University of Strathclyde Engineering

AUTOMATED NDT INSPECTION FOR LARGE AND COMPLEX GEOMETRIES OF COMPOSITE MATERIALS

Carmelo Mineo

Centre for Ultrasonic Engineering

Department of Electronic and Electrical Engineering

University of Strathclyde

Glasgow

A dissertation submitted for the degree of Doctor of Philosophy

September 2015

DECLARATION

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: ______________________________

Date: ________________________________

To my parents Salvatore and Caterina for nourishing my childhood and kindling my curiosity. To them and my brother Francesco, I dedicate this work as reparation for all my attention it has deprived them of.

Ai miei genitori Salvatore e Caterina per aver nutrito la mia infanzia e aver stimolato la mia curiosità. A loro e a mio fratello Francesco, dedico questo lavoro in riparazione per tutte le attenzione che questo ha sottratto loro.

"If error is corrected whenever it is recognized as such, the path of error is the path of truth."

Hans Reichenbach (1951)

ABSTRACT

Large components with complex geometries, made of composite materials, have become very common in modern structures. To cope with future demand projections, it is necessary to overcome the current non-destructive testing (NDT) bottlenecks encountered during the inspection phase of manufacture.

This thesis investigates several aspects of the introduction of automation within the inspection process of complex parts. The use of six-axis robots for product inspection and non-destructive testing systems is the central investigation of this thesis. The challenges embraced by the research include the development of a novel controlling approach for robotic manipulators and of novel path-planning strategies. The integration of robot manipulators and NDT data acquisition instruments is optimized. An effective and reliable way to encode the NDT data through the interpolated robot feedback positions is implemented.

The viability of the new external control method is evaluated experimentally. The observed maximum position and orientation errors are respectively within 2mm and within 1 degree, over an operating envelope of $3m^3$.

A new software toolbox (RoboNDT), aimed at NDT technicians, has been developed during this work. RoboNDT is intended to transform the robot path-planning problem into an easy step of the inspection process. The software incorporates the novel path-planning algorithms developed during this research and is shaped to overcome practical limitations of current OLP software. The software has been experimentally validated using scans on real high value aerospace components. RoboNDT delivers tool-path errors that are lower than the errors given by commercial off-line path-planning software. For example the variability of the standoff is within 10 mm for the tool-paths created with the commercial software and within 4.5 mm for the RoboNDT tool-paths, over a scanned area of $1.6m^2$.

The output of this research was used to support a 3-year industrial project, called IntACom and led by TWI on behalf of major aerospace sponsors. The result is a demonstrator system, currently in use at TWI Technology Centre, which is capable of inspecting complex geometries with high throughput. The IntACom system can scan real components 2.8 times faster than traditional 3-DoF scanners deploying phased-array inspection and 6.7 times faster than commercial gantry systems deploying traditional single-element inspection.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my supervisors. They provided continuous support to my research, with lot of patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me throughout the research and the writing-up of this thesis. Therefore, special thanks go to my academic supervisors, Dr Stephen Gareth Pierce and Professor Anthony Gachagan, who kept a sense of humour when I had lost mine. Heartfelt thanks to my industrial supervisor at TWI, Dr Pascual Ian Nicholson, whose selfless time and care were sometimes all that kept me going. His maddening attention to detail drove me to finally learn to punctuate prose.

Thanks to the TWI experienced technology fellow, Mr Ian Cooper, for involving me in the development of the IntACom project and for providing many other challenging tasks and learning opportunities. They were worth the efforts.

A great thank to the brilliant TWI software programmer, Dr Ben Wright, for the stimulating discussions, for inspiring several areas of my research, for the successful work carried out together under short deadlines, and for enduring my plaintive and grammatically not correct speeches during the shared car commutes.

I am grateful to the administrative assistants at the University of Strathclyde, Rhona Nicholson and Lynn Morrison, and to the administrators at TWI Technology Centre, Lorna Morris and Maureen John, for keeping me organized and being always ready to help.

I would like to thank the colleagues of the robotics research group at the University of Strathclyde for keeping me involved and updated, despite of the geographical distance from the academic department.

I gratefully acknowledge the funding sources that made my PhD work possible. I was funded by the Prince of Wales Innovation Scholarship Scheme (POWIS). My research was also supported by the IntACom project and its sponsors (Rolls-Royce, Bombardier Aerospace and GKN Aerospace and Welsh Government).

Lastly but not the least, I would like to thank all my friends and my family for all their encouragement and love (the most important ingredient of the recipe). Thank you.

Carmelo Mineo

September 2015

ALPHABETICAL LIST OF ACRONYMS

ACUI	Air-Coupled Ultrasonic Inspection
AO	Acusto-Optic
AOS	Advanced OEM Solutions
ASCII	American Standard Code for Information Interchange
AUSS	Automated Ultrasonic Scanning System
BINDT	British Institute of Non-Destructive Testing
BVID	Barely Visible Impact Damage
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CATT	Computer Aided Tap Test
CFRP	Carbon Fibre Reinforced Plastic
CMM	Coordinate Measuring Machine
CMOS	Complementary Metal Oxide Semiconductor
CNRP	Carbon Nanotube Reinforced Polymer
CNT	Carbon Nanotube
COSHH	Control of Substances Hazardous to Health
CPD	Crash Protection Device
CPU	Central Processing Unit
CUE	Centre for Ultrasonic Engineering
DAS	Data Acquisition System
DLL	Dynamic Link Libraries
DOF	Degrees Of Freedom
DOTNAC	Development and Optimization of THz NDT on Aeronautics Composite
DSL	Diagnostic Sonar Ltd
EC	Eddy Current
EtherCAT	Ethernet for Control Automation Technology
EXE	Executable File
FFT	Fast-Fourier Transform
FMC	Full Matrix Capture
FPS	Frames per second
GC	Garbage Collection
GE	General Electric
GNTP	South Korea Research Centre for Composite Materials for the Aeronautical Industry
GRP	Glass Reinforced Plastic
GS	Giga-samples
GUI	Graphical User Interface
HCT	High Critical Temperature
HMI	Human-Machine Interface
HTS	High-Temperature Superconducting
HW	Hollow Wrist
ICCST	International Conference on Composite Science and Technology

IEC	International Electrotechnical Commission
IF	Inconvenience Factor
IFFT	Inverse Fast-Fourier Transform
ILSS	Interlaminar Shear Strength
IP	Internet Protocol address
IPOC	Interpolation Cycle counter
IR	Infrared
ISO	International Organization for Standardization
IUT	Immersion Ultraonic Testing
KCT	KUKA Control Toolbox
KRC	KUKA Robot Controller
KRC2	Second generation of KUKA Robot Controllers
KRC4	Fourth generation of KUKA Robot Controllers
KRL	KUKA Robot Language
LIN	Linear
LU	Laser Ultrasound
MAUS	Mobile Automated Scanner
MCS	Motion Control System
MIT	Massachusetts Institute of Technology
MNDT	Marietta Non-Destructive Testing
NDE	Non-destructive Evaluation
NDI	Non-Destructive Inspection
NDT	Non-Destructive Testing
NDTS	NDT Solutions Inc.
NRPB	National Radiological Protection Board
OEM	Original Equipment Manufacturer
OLP	Off-Line Programming
OS	Operating System
PA	Phased Array
PAUT	Phased Array Ultrasonic Testing
PC	Personal Computer
PE	Pulse-Echo
PMC	Polymer Matrix Composites
POD	Probability Of Detection
PTFE	Polytetrafluoroethylene
PTP	Point-to-point
QNDE	Quantitative Non-Destructive Evaluation
RCNDE	Research Centre in Non-destructive Evaluation
RMS	Root Mean Square
RSI	Robot Sensor Interface
RTOS	Real-Time Operating System
SI	International System of Units
SNR	Signal to Noise Ratio
SQUID	Superconducting Quantum Interference Devices
STL	Stereo-Lithography

SVD	Singular Value Decomposition
TCP	Tool Central Point
TCP/IP	Transmission Control Protocol / Internet Protocol
TOF	Time-Of-Flight
TOFD	Time-Of-Flight Diffraction
TRL	Technology Readness Level
TT	Through-Transmission
TTU	Through-Transmission Ultrasonic
TWI	The Welding Institure
UDP	User Datagram Protocol
UK	United Kingdom
UKAEA	United Kingdom Atomic Energy Authority
UoS	University of Strathclyde
USA	United States of America
USB	Universal Serial Bus
USL	Ultrasonic Sciences Ltd
UT	Ultrasonic Testing
XML	Extensible Markup Language
YAG	Yttrium Aluminium Garnet

CONTENTS

1. Introduction ***1***

1.1. Context of research **1**

1.2. Contributions to knowledge **3**

1.3. Thesis structure **5**

1.4. Publications arising from this thesis **7**

1.4.1. Journal papers 7

1.4.2. Conference papers 7

References **9**

2. Research Background and Motivations ***10***

2.1. Introduction to non-destructive testing of composites **10**

2.2. Composite materials **12**

2.2.1. Types and defects 12

2.2.2. Geometries of aeronautical components 15

2.3. Techniques overview **16**

2.3.1. Contact and coupled ultrasonic techniques 17

2.3.2. Non-Contact ultrasonic technique 23

2.3.3. Other relevant techniques 24

2.3.4. Emerging techniques 29

2.4. Semi-automated and Automated NDT **33**

2.4.1. Current needs for automated NDT solutions 33

2.4.2. General overview of manipulator types 33

2.4.3. Commercial systems 35

2.5. Motivations and weaknesses of the current systems **39**

References **42**

3. Interfacing challenges for automating NDT with 6-axis manipulators ***46***

3.1. Introduction 46

3.2. Describing 6-axis robot arms 47

3.2.1. Kinematics 49

3.2.2. Singularities 51

3.2.3. Working envelope 52

3.2.4. Dynamics 53

3.2.5. Accuracy and repeatability 53

3.2.6. Communications 55

3.2.7. Robot programming 56

3.2.8. Synchronization of cooperative robots 58

3.3. Robotic hardware used in this work 59

3.3.1. Robots at the University of Strathclyde 60

3.3.2. Robots at TWI Technology Centre 60

3.4. Review of inspection strategies 64

3.4.1. Discrete point measurements 66

3.4.2. Continuous path measurements 73

3.5. External control 79

3.5.1. Experimental tests 80

3.6. Accuracy investigations 81

3.6.1. Comparison of target and feedback positions 81

3.6.2. Independent positional accuracy measurements 85

3.7. Enabling external control for KRC4 robots 89

3.7.1. Upgrading the C# server application 89

3.7.2. Testing the KRC4 system 90

3.8. Interfacing the ultrasonic acquisition system 91

3.9. Encoding UT data acquired from curved surfaces 98

3.10. Wrapping up 101

References 102

4. Fundamental considerations for NDT path-planning 104

4.1. Available solutions 104

4.2. Motivations for the investigation of new solutions 108

4.3. The MATLAB programming environment 111

4.4. Feasibility studies for new approaches 112

4.4.1. Command coordinate transfer 112

4.4.2. Curvilinear coordinates for severe curvature surfaces 115

4.5. Developing a full NDT path-planning toolbox 118

4.5.1. Start-up module 119

4.5.2. Path-planning module 124

4.5.3. Kinematics & Simulation module 130

4.5.4. Output module 133

4.6. Metrology experiments – Path accuracy verification 135

4.7. Ultimate approximation-free path-planning approach 139

4.7.1. Disadvantages of the analytical approach 139

4.7.2. A mesh-following approach 140

4.7.3. Testing the reliability and the flexibility 145

4.8. Inverse kinematics for simulations 146

4.8.1. Disadvantages of the open-source function 146

4.8.2. A full solution to the kinematics formulation 147

4.8.3. Selection of the most appropriate configuration 152

4.8.4. Experimental testing 155

4.9. Wrapping-up 158

References 160

5. A graphical interface for practical NDT implementations 162

5.1. RoboNDT Software 162

5.1.1. Modular GUI architecture 163

5.1.2. Libraries 166

5.1.3. Calibration of the sample position 180
5.1.4. Path-planning GUI module 184
5.1.5. Avoidance of surface obstacles and voids 187
5.1.6. Interactive definition of tool-path features 188
5.1.7. Kinematics features 192
5.1.8. Approach and retract motions 193
5.1.9. Flexible display 193
5.1.10. Output 194
5.1.11. Executable application package 195

5.2. Validation experiments 195

5.2.1. Aerospace winglet sample 196
5.2.2. Reverse engineered titanium fan blade 199

5.3. Future development 208

5.4. Wrapping-up 209

References 211

6. Systems engineering and integration 212

6.1. The IntACom project 212

6.2. Setting up the robotic inspection prototype system 214

6.2.1. Magnetic tool holder 214
6.2.2. Water-jet nozzle prototypes 220
6.2.3. Draining trays 225
6.2.4. Water circulation system 226
6.2.5. Jig table 227

6.3. The inspection process 228

6.4. The data acquisition software 229

6.4.1. Selecting the most suitable programming language 230
6.4.2. Software architecture 231

6.5. IntACom path-planning solutions 234

6.5.1. Generation of the main inspection tool-paths 234

6.5.2. Built-in capability to rescan areas of interest 235

6.6. Future development 244

6.7. Wrapping-up 245

References 247

7. System performance evaluation - High value aerospace inspection 248

7.1. Introduction 248

7.2. Pulse-echo applications 249

7.2.1. Scan of flat surfaces 249

7.2.2. Investigating the optimum stand-off 251

7.2.3. CFRP fan blade 255

7.3. Through-transmission inspections 257

7.4. Final performances 260

7.4.1. Scanning speeds 261

7.4.2. Robot feedback rate and data encoding 263

7.5. Wrapping-up 266

References 268

8. Conclusions and Future work 269

8.1. Conclusions 269

8.2. Future work 271

APPENDICES 273

A1 – Implementation of external control for KRC2 robots 274

A2 – Ensuring a reliable connection 282

A3 – Configuration of external control for KRC4 robots 287

A4 – Performance optimization of external control code 291

A5 – Rotational matrices 294

B1 – Main program of the path-planning MATLAB toolbox 295

B2 – Setting up automatic execution of approach and retract motions 296

C1 – Setting a software interrupt to stop the robot motion....................................... 301

C2 – Water-jet solutions 303

C3 – Software configuration to automate the return to areas of interest 308

D1 – Template of KRL modules for Master-Slave synchronization 312

References... 313

1. INTRODUCTION

1.1. Context of research

Non-destructive Testing (NDT) and Non-destructive Evaluation (NDE) are highly multidisciplinary groups of analysis techniques used throughout science and industry to evaluate the properties of materials, and/or to ensure the integrity of components/structures, without causing damage to them [1]. In civil aerospace manufacturing, the increasing deployment of composite materials demands a high integrity and traceability of NDT measurements, combined with a rapid throughput of data. Modern components increasingly present challenging shapes and geometries for inspection. Using traditional manual inspection approaches produce a time-consuming bottleneck in the industrial production [2]. There is therefore a fundamental motivation for increased automation for NDT in production environment.

Modern Computer-Aided Design (CAD) is used extensively in composite manufacture. Additionally whereas it was once necessary to construct large items from many smaller parts, Computer-Aided Manufacturing (CAM) now allows these large items to be produced easily from one piece of raw material (either through traditional subtractive approaches, or with more recent additive manufacturing processes [3]). As a result, large components with complex geometries have become very common in modern structures, and the aerospace industry is a typical field where wide complex shaped parts are very frequently used. Moreover the use of composite materials, which are notoriously challenging to inspect [4], is becoming widespread in the construction of new generations of civilian aircraft. To cope with future demand projections for these operations, it is therefore essential to overcome the current NDT bottlenecks.

A fundamental issue with composites manufacturing compared to conventional light alloy materials lies in the process variability. Often parts that are designed as identical, will have significant deviations from the CAD, and also suffer from inherent but different part to part springiness out of the mould. This presents a significant challenge for precision NDT measurement deployment, which must be flexible to accommodate these manufacturing issues.

NDT inspection is still often performed manually by technicians who typically have to move appropriate probes over the contour of the sample surfaces. Manual scanning requires trained technicians and results in a very slow inspection process for large samples. The repeatability of a test can be challenging in structures where complex setups are necessary to perform the inspection (e.g. orientation of the probe, constant standoff, etc.) [5]. While manual scanning may remain a valid approach around the edges of a structure, or the edges of holes in a structure, developing reliable automated solutions has become an industry priority to drive down inspection times. The fundamental aim of automation within the inspection process is to minimize downtimes due to the higher achievable speed, and in parallel to carry out 100% inspection coverage of the sample, including all edge areas.

Semi-automated inspection systems have been developed to overcome some of the shortcomings with manual inspection techniques, using both mobile and fixed robotic platforms. The use of linear manipulators and bridge designs has for a number of years provided the most stable conditions in terms of positioning accuracy [6, 7]. The use of these systems to inspect parts with noncomplex shapes (plates, cylinders or cones) is widespread; typically, they are specific machines which are used to inspect identically shaped and/or sized parts.

More recently, many manufacturers of industrial robots have produced robotic manipulators with excellent positional accuracy and repeatability. An industrial robot is defined as an automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes [8]. In the spectrum of robot manipulators, some modern robots have suitable attributes to develop automated NDT systems and cope with the challenging situations present in the aerospace industry [9]. They present precise mechanical systems, the possibility to accurately master each joint, and the ability to export positional data at high update rates. Some applications of 6-axis robotic arms in the

NDT field have been published during the last few years and many manufacturers within the aerospace sector [2] are interested in using such automation solutions.

Despite these previous efforts, key areas of research for robot deployed NDT remain unexplored and effective fully automated NDT inspections of composite parts are not commonplace. The key challenges, found by the author of the present thesis, include the development of suitable path-planning strategies for robot manipulators interfaced with NDT acquisition instruments, high speed NDT data collection, integration of surface metrology measurements, and overall visualisation of results in a user friendly fashion. This thesis is the result of a three year PhD project. The project embraces the visions of the Centre for Ultrasonic Engineering (CUE) at the University of Strathclyde and of TWI Technology Centre (Wales), where most of the research was carried out.

The PhD project has used the opportunity to prove and apply the outcomes of the research to an industrial project, called IntACom and led by TWI on behalf of major aerospace sponsors. The objective was to achieve a fourfold increase in the inspection throughput of complex aerospace components [2]. Additional input has come from the UK Research Centre in Non-destructive Evaluation (RCNDE) consortium that conducts research into integration of metrology with NDT inspection [10, 11]. Both the IntACom project and the RCNDE consortium have deemed the optimal robotic tool-path generation over complex curved surfaces as a fundamental requirement and highlighted other critical areas to be addressed in the future.

1.2. Contributions to knowledge

Previous research within the field of automated NDE focused on understanding and quantifying the constraints to remote mobile, (both wheeled and aerial) NDE inspection, specifically addressing issues with relation to localisation, mapping, path-planning and sensing [12, 13]. In the field of robot deployed NDE of complex surfaces through fixed 6-Degrees Of Freedom (DOF) robotic arms, some investigations have demonstrated the possibility of using industrial robots to deploy NDT applications, integrating them in an overall inspection apparatus [9, 14-16].

This body of work investigates and addresses solutions to the deployment of phased-array ultrasonic hardware integrated to robotic arms for the inspection of complexly shaped parts. The PhD project drove the fundamental research. Therefore the aims and objectives of this thesis were to understand and quantify the constraints of robotic inspection. The weaknesses of early infant systems were examined and opportunities for improvement were sought and applied in the development of an automatic robotic inspection prototype system. They are both hardware and software solutions.

This thesis presents a number of unique and novel, industry focused contributions to the fields of robotic NDE. They are bulleted in short below.

- The literature review gives a general presentation of the research field and makes a synthesis of the state-of-the art. It is useful to establish the metre of comparison and the benchmarking for new robotic inspection systems.
- The robot tool-paths are usually created by commercially available software and outputted in the specific robot language of the hardware in use. An external control approach is here exploited for the first time to run the original command tool-path, leaving open the possibility to add real-time correction. It allows controlling the position of the robotic end-effector very accurately and it is suitable to follow curved contours; a fundamental requirement for reliable NDT inspections.
- The external control of a robotic arm is used simultaneously with the deployment of leading edge metrology equipment, such as the LEICA laser tracker, to get figures of the accuracy of the robot manipulators.
- The advantages of the innovative way to control robotic arms, sending command coordinate packets from an external server computer, are highlighted. It allows developing seamless NDT data acquisition software for the control of all aspects of robotic inspection, from the management of the part virtual model to the display and analysis of the NDT results.
- Flexible path-planning solutions are investigated to support the new approach of controlling robot manipulators. Several path-planning algorithms were tested and a novel algorithm for creating tool-paths from meshed CAD files is presented. The algorithm is able to accurately follow the real curvature of challenging surfaces, compute curvilinear distances and generate raster tool-paths with parallel inspection trajectories. The capability to work with meshed surfaces makes the algorithm suitable to be used for geometries obtained through reverse engineering and surface mapping.
- A complete application, named RoboNDT, is developed using the MATLAB programming platform. The software embeds the new path-planning algorithm

and has some unique features that cannot be found in current commercially available path-planning solutions. The application can be used as a stand-alone solution to address the path-planning needs or be implemented as a software add-on within NDT data acquisition software to extend their functionalities. Validation experiments, showing the software performances, are presented and comparisons with commercial path-planning software are made.

- Higher levels of automation are achieved by setting a new way of commanding approach and retract robot movements. This avoids the operator having to write specific robot language programs.
- The developed external control server application is optimized to run in a real-time fashion and allow simultaneous collection of NDT data. A novel strategy for encoding NDT data with robot positional feedback, developed for the TWI IntACom project, demonstrates NDE imaging quality for ultrasonic phased-array inspections.
- An innovative automated inspection prototype system has been developed. The design of the main elements of the system is presented and the final outcomes are discussed through example applications. The system is suited to both research & development (R&D) and production inspection. Two robot arms deploy ultrasonic transducers allowing ultrasonic pulse-echo and through-transmission ultrasonic inspection of complex geometry aerospace composite components.
- Two cooperative robot arms are used to run ultrasonic through-transmission inspection of challenging geometries. A suitable approach has been developed to maintain the two robots aligned at all times during the inspection. This, in conjunction with the high performances of the IntACom acquisition software, enables an excellent quality of through-transmission scans.
- The solutions developed within this PhD have already attracted the attention of the industry for new opportunities and the delivery of future high-value projects.

1.3. Thesis structure

The remainder of this thesis is structured as follows.

Chapter 2 introduces the research background and highlights the motivations behind the work described in this thesis. An introduction to non-destructive testing and a classification of composite materials is provided. An overview of both traditional and modern testing techniques is presented. The final part of the chapter is focused on semi-automated and automated NDT solutions with a review of the state-of-the-art.

Chapter 3 focuses on the robotics challenges to be addressed for the deployment of robotic arms as manipulators of NDT probes. Some early NDT systems, which provided a starting point to the research, are described. The chapter ends focusing on the implementation of external control, the presentation of accuracy results and the inverse kinematics models.

Chapter 4 describes the investigations for the development of fundamental path-planning algorithms. It presents an innovative and effective path-planning approach and highlights obstacles and solutions to several difficulties and challenges.

Chapter 5 is an overview of the RoboNDT software that has been developed. The chapter presents the software graphical user interface (GUI), through the modular structure of the application. The final section of the chapter contains the outcomes of the validation experiments put in place to test the performance of the software.

Chapter 6 presents a portrait of the IntACom project, developed by TWI and sponsored by major aerospace companies and the Welsh government. The chapter presents the design of the hardware of the IntACom robotic inspection prototype system and the developed software solutions, including the developments provided under the PhD.

Chapter 7 provides a final performance validation of all the techniques introduced by using the system to undertake reference ultrasonic NDT scans on a number of high value aerospace components with complex geometry. The improvements in scan times over the currently implemented industrial scanning systems are quantified.

Finally, Chapter 8 provides an overall summary, conclusions and suggestions for future work.

A series of appendices offer additional details regarding the implementation of fundamental approaches and algorithms described in the thesis.

1.4. Publications arising from this thesis

1.4.1. Journal papers

- C. Mineo, S. G. Pierce, P. I. Nicholson, B. Wright, I. Cooper, *Robotic path planning for non-destructive testing - a custom MATLAB toolbox approach*, Robotics and Computer-Integrated Manufacturing Journal – Accepted for publication on 18th May 2015.
- C. Mineo, B. Wright, P. I. Nicholson, I. Cooper, S. G. Pierce, *PAUT inspection of complex shaped composite materials through 6 DOFs robotic manipulators*, Insight Journal, Volume 57, No 3, March 2015.

1.4.2. Conference papers

- C. Mineo, J. Riise and S. G. Pierce, *Robotic path planning for non-destructive testing through RoboNDT*, 54th BINDT Annual Conference, Telford, UK, September 2015.
- I. Cooper, C. Mineo, B. Wright, R. Freeman, P. I. Nicholson, D. Liaptsis, *Rapid Inspection of Composite and Additive Manufactured Components using Advanced Ultrasonic Techniques*, 26th Advanced Aerospace Materials and Processes (AeroMat) Conference, Long Beach, California (USA), May 2015.
- I. Cooper, P. I. Nicholson, D. Liaptsis, B. Wright, C. Mineo, *Development of a fast inspection system for complex aerospace structure*, 6th International Symposium on NDT in Aerospace, Madrid, Spain, November 2014.
- C. Mineo, B. Wright, P. I. Nicholson, I. Cooper, S. G. Pierce, *PAUT inspection of complex shaped composite materials through 6 DOFs robotic manipulators*, 53rd BINDT Annual Conference, Manchester, UK, September 2014.
- C. Mineo, G. Pierce, P. I. Nicholson, I. Cooper, *Robotic path planning for non-destructive testing of complex shaped surfaces*, QNDE Conference 2014, Boise, Idaho (USA), July 2014.
- I. Cooper, P. I. Nicholson, D. Liaptsis, C. Mineo, B. Wright, *The Development of a Fast Inspection System for Complex Aerospace Composite Structure*, 25th Advanced Aerospace Materials and Processes (AeroMat) Conference, Orlando, Florida (USA), June 2014.

- I. Cooper, P. I. Nicholson, D. Yan, B. Wright, D. Liaptsis, C. Mineo, *Development of a Fast Inspection System for Complex Composite Structures*, 5th International Symposium on NDT in Aerospace, Singapore, November 2013.
- C. Mineo, S. G. Pierce, M. Morozov, P. I. Nicholson, I. Cooper, *Computer-Aided Tool Path Generation for Robotic Non-Destructive Inspection*, 52nd BINDT Annual Conference, Telford, UK, September 2013.
- I. Cooper, P. I. Nicholson, D. Yan, B. Wright, C. Mineo, *Development of a Fast Inspection System for Aerospace Composite Materials - the IntACom Project*, 9th International Conference on Composite Science and Technology (ICCST-9), Sorrento, Italy, July 2013.
- C. Mineo, D. Herbert, M. Morozov, S. G. Pierce, P. I. Nicholson, I. Cooper, *Robotic Non-Destructive Inspection*, 51st BINDT Annual Conference, Daventry, UK, September 2012.

References

[1] L. Cartz, *Nondestructive testing*. United States: ASM International, 1995.

[2] I. Cooper, P. I. Nicholson, D. Yan, B. Wright, and C. Mineo, "Development of a Fast Inspection System for Aerospace Composite Materials - The IntACom Project," presented at the 9th International Conference on Composite Science and Technology (ICCST-9), Sorrento (Italy), 2013.

[3] I. Gibson, D. W. Rosen, and B. Stucker, *Additive manufacturing technologies: rapid prototyping to direct digital manufacturing*. New York: Springer, 2010.

[4] Y. Bar-Cohen, "Emerging NDE Technologies and Challenges at the Beginning of the 3 rd Millennium--Part II, Part I," 2000.

[5] T. Sattar, "Robotic non-destructive testing," *Industrial Robot: An International Journal,* vol. 37, 2010.

[6] M. Schwabe, A. Maurer, and R. Koch, "Ultrasonic Testing Machines with Robot Mechanics - A New Approach to CFRP Component Testing," presented at the 2nd International Symposium on NDT in Aerospace, Germany, 2010.

[7] P. Louviot, A. Tachattahte, and D. Garnier, "Robotised UT Transmission NDT of Composite Complex Shaped Parts," presented at the 4th International Symposium on NDT in Aerospace, Berlin (Germany), 2012.

[8] B. Djordjevic, "Remote Non-Contact Ultrasonic Testing of Composite Materials," presented at the 15th World Conference on Nondestructive Testing, Roma (Italy), 2000.

[9] E. Cuevas, M. López, and M. Garcìa, "Ultrasonic Techniques and Industrial Robots: Natural Evolution of Inspection Systems," presented at the 4th International Symposium on NDT in Aerospace, Ausburg, Germany, 2012.

[10] S. G. Pierce, G. Dobie, R. Summan, L. Mackenzie, J. Hensman, K. Worden, and G. Hayward, "Positioning challenges in reconfigurable semi-autonomous robotic NDE inspection," in *SPIE 7650, Health Monitoring of structural and Biological Systems 2010*, San Diego, California, USA, 2010, p. 76501C.

[11] C. Mineo, D. Herbert, M. Morozov, S. G. Pierce, P. I. Nicholson, and I. Cooper, "Robotic Non-Destructive Inspection," presented at the 51st Annual Conference of The British Institute of Non-Destructive Testing, Daventry (UK), 2012.

[12] R. Clark, G. Punzo, G. Dobie, R. Summan, C. N. MacLeod, G. Pierce, and M. Macdonald, "Autonomous swarm testbed with multiple quadcopters," in *1st World Congress on Unmanned Systems Enginenering, 2014-WCUSEng*, 2014.

[13] R. Clark, G. Punzo, G. Dobie, C. MacLeod, R. Summan, G. Pierce, M. Macdonald, and G. Bolton, "3D Model Generation using an Airborne Swarm," presented at the 41st Annual Review of Progress in Quantitative Nondestructive Evaluation, , Boise, Idaho (USA), 2014.

[14] F. Bentouhami, B. Campagne, E. Cuevas, T. Drake, M. Dubois, T. Fraslin, P. Piñeiro, J. Serrano, and H. Voillaume, "LUCIE - A flexible and powerful Laser Ultrasonic system for inspection of large CFRP components.," presented at the 2nd International Symposium on Laser Ultrasonics, Talence (France), 2010.

[15] A. Maurer, W. D. Odorico, R. Huber, and T. Laffont, "Aerospace composite testing solutions using industrial robots," presented at the 18th World Conference on Nondestructive Testing, Durban, South Africa, 2012.

[16] J. T. Stetson and W. D. Odorico, "Robotic inspection of fiber reinforced aerospace composites using phased array UT," presented at the 40th Annual Review of Progress in Quantitative NDE, Baltimore, Maryland, 2013.

2. RESEARCH BACKGROUND AND MOTIVATIONS

2.1. Introduction to non-destructive testing of composites

The non-destructive testing (NDT) industry emerged in response to the need to test metals and metallic components for manufacturing and in-service defects. Scientists and engineers are continually seeking to develop materials with improved mechanical, thermal and other properties, reflecting the desire to operate components at more extreme temperatures or subject them to ever-greater in-service stresses and loading. This has led to the emergence of a range of non-metallic engineering materials, such as composites and ceramics. Their subsequent use in critical applications, together with the emerging families of nanomaterials, has highlighted the need for new and improved NDT methods. Some of the traditional techniques cannot be applied to these materials, others can be adapted but in several instances new methods are required [1]. Composite materials are nowadays widely used in a number of industrial sectors from aviation, space, to marine, automotive, and sports goods. In recent years composite structures have seen a substantial increase of their use in the new generation of airplanes [2]. Much of the fuselage of the new Boeing 787 Dreamliner (Figure 2.1) and Airbus A350-XWB is composed of carbon fibre reinforced plastic (CFRP), making these aircraft lighter than those with a comparable aluminium fuselage, with the added benefit of lower maintenance requirements due to CFRP’s superior damage resistance [3] and lower fuel consumption.

Figure 2.1 – A disassembled carbon fibre fuselage section of the Boeing Dreamliner 787 in Everett, Washington (Source: Wikipedia)

Many conventional and emerging NDE tools are used for NDT of composites. Ultrasonic inspection methods have been demonstrated as very effective tools in characterization of anisotropic composite materials and structures [4, 5]. Due to increasingly strict quality standards in all industrial sectors for both production and maintenance, non-destructive testing must provide quantitative data. The flaws detected can then be monitored throughout the service life of the device or component. Automated testing is more efficient and it can considerably boost productivity, thus keeping industries competitive. Automated gantry systems can fulfil the automation requirements in a number of applications, but six-axis industrial robots are necessary for automated inspection of complex-shaped parts [6]. There are several techniques for automated ultrasonic transmission inspection. Ultrasound testing can be carried out in pulse-echo (PE) or in through transmission ultrasonic (TTU) mode; for the pulse-echo technique there is only one movement to be managed and only one robot is required. Also for TTU only one robotic arm can be sufficient for simple and small geometries, using a fork with a squirter at the end of each branch. This method is suitable for simple-geometry parts (flat or rotational), but not for complex geometries. For such parts, it is necessary to use two synchronised robots that travel opposite each other.

Automated ultrasonic testing of complex aerospace components is usually carried out with the squirter technique [7]. In advanced applications and for composite process control it is advantageous to perform non-contact ultrasonic measurements that do not require traditional ultrasonic coupling and contact requirements [8]. However, water coupling may deliver disadvantages like pressure variations, air-bubbles, lime scales, algae and corrosion of the mechanics. *Sonatest* has developed the RapidScan system, where the probe and the coupling water are confined within a rubber wheel [9]. The wheel probe can be dragged to follow the contour of curved surfaces; it is equipped with an encoder and is able to display the three most common data presentation formats, known in the NDT world: A-scan, B-scan and C-scan presentations. Each presentation mode provides a different way of looking at and evaluating the region of material being inspected. The A-scan presentation displays the amount of received ultrasonic energy as a function of time. The relative amount of received energy is plotted along the vertical axis and the elapsed time (which may be related to the sound energy travel time within the material) is displayed along the horizontal axis. The B-scan presentation is a profile (cross-sectional) view of the test specimen. In the B-scan, the time-of-flight (travel time) of the sound energy is displayed along the vertical axis and the linear position of the transducer is displayed along the horizontal axis. From the B-scan, the depth of the reflector and its approximate linear dimensions in the scan direction can be determined. The B-scan is typically produced by establishing a trigger gate on the A-scan. The C-scan presentation provides a plan-type view of the location and size of test specimen features. The plane of the image is parallel to the scan pattern of the transducer. C-scan presentations are produced with an automated data acquisition system, such as a computer controlled immersion scanning system.

The air-coupled ultrasonic technique introduces a large acoustical mismatch between air and solids. The mismatch can be reduced with special transducers, but in spite of these optimisations the test frequencies remain lower than 1 MHz [10] because of attenuation with air. An emerging technology is the hybrid ultrasonic test configuration using laser ultrasound and air/gas coupled ultrasonic transducers [8].

Due to the lack of a fully automatic production chain, the industry is presently not being able to fulfil the drastically increasing demand of the components [11]. The actual goal, embraced by the PhD project, is to improve the automation level in the manufacturing technology, reduce the manufacturing cost and improve the part's quality.

2.2. Composite materials

2.2.1. Types and defects

The various forms of composite materials, with their advantages of light weight, design flexibility, and high specific stiffness and specific strength, are widely used as structural materials in the aviation, space, marine, automotive, and sports industries. Composite structures can be divided into approximately two large categories: solid laminates (see

Figure 2.2a) and bonded sandwich structures. The use of composites in the aviation industry started, by employing honeycomb panels (see Figure 2.2b) as flight control surfaces, fairings, and engine nacelles. Composite sandwiches with foam core (see Figure 2.2c) have been used in boat building and on helicopter rotor blades. In recent years, thick laminates are increasingly employed as primary load-bearing structures on aircraft.

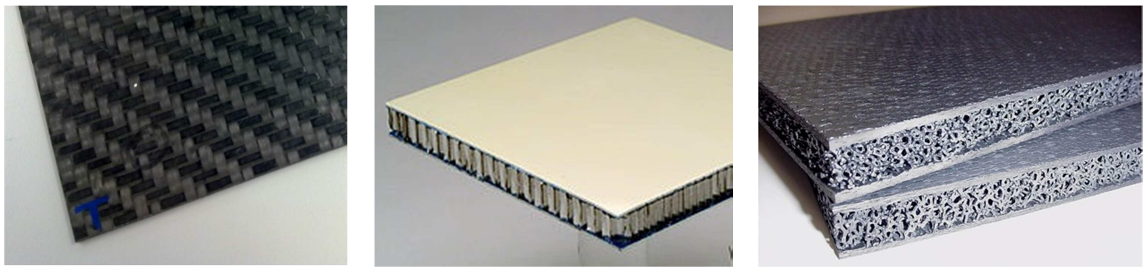

(a) (b) (c)

Figure 2.2 – a) Carbon fibre reinforced plastic or CFRP solid laminate, b) honeycomb panel (Source: NASA), c) ceramic matrix composite/foam core sandwich structure (Source: NASA).

Some honeycomb composite structures on the aircraft, such as the composite rudder of the Airbus 300, are now considered primary structures [2]. Since the detection of the different flaws and defects affecting composite materials is absolutely important in order to carry out a manual inspection, it is considered to be even more important when developing an automatic NDT system, especially if it is supposed to perform an intelligent identification of failure areas and support sentencing the results. Imperfect manufacturing conditions of solid laminates can lead to flaws and defective conditions of a fabricated composite part. The list of flaws and defects that can be encountered in manufacturing includes embedded foreign objects, regions of excessive porosity, errors in ply orientation or layup sequence, ply waviness, and interlaminar delamination. Foreign objects in a laminate compromise the uniformity and act as stress concentrators under load; they may lead to disbond or delamination while in service. Porosity is a volumetric distribution of microscopic voids that occur in the resin matrix of a fibre reinforced composite. The size and morphology of the porosity depend on the type of the laminate construction and the layup design. Since the micro-porosities are located in the resin matrix, they tend to concentrate in the interfacial resin rich layer in a laminate. Such porosity can considerably degrade the interlaminar shear strength (ILSS) of the composite laminate. Ply orientation and layup sequence errors are more likely to occur in hand laid up parts than in parts fabricated with automated tape laying machines or fibre placement machines. Composite parts containing layup errors have to be rejected because the errors can have a large effect on the strength of the part. Ply waviness refers to undulations of the plies; such anomalies can occur locally in both the in-plane direction and in the out-of-plane direction. The ply waviness condition is detrimental to the performance of the laminate as the wavy fibres can stretch under load, putting stress on the matrix. Finally, composites suffering large temperature variations during manufacturing can develop micro-cracks in the direction normal to the plies. Adhesively bonded sandwich structures (e.g. Figure 2.2b-c) consist of a pair of facesheets that are usually fibre reinforced composite laminates and a composite or metallic honeycomb core or a foam core. The facesheets are adhesively bonded to the core. For facesheets made of composite laminates, defects similar to those described above may occur. The most important inspection task for bonded structures is to verify the integrity of the bond between the core and the facesheets. Although highly desirable, the strength of an adhesive bond cannot be determined by a simple NDT test [2]. The primary focus in the inspection of bonded sandwich structures is the detection and characterization of the in-service damage.

While in service, the composite structures may suffer damage from a number of natural and man-made causes. For both solid laminates and sandwich structures, the most significant damages are caused by mechanical impact. Low velocity impacts on composites often lead to non-visual damage on the surface but significant damage internally. Impact damage can result from dropped tools in a maintenance hangar or from collision with ground handling equipment. Bird strikes in flight can cause considerable damage to the radome[1] on to the wing leading edge. The damage morphology of an impact on a composite solid laminate is well known [12, 13]. The damage takes the form of a series of interlaminar delaminations that increase in size with depth. The delaminations are usually accompanied by micro-cracking of the matrix. For honeycomb sandwich panels, impact damage typically results in a fracture or bucking of the core, often with an upward concave profile [14]. The lowest point of the fracture may occur at a depth approaching one half of the core thickness. At higher impact energy, delamination or fibre breakage may occur in the facesheets. Composite components, especially the radomes on aircrafts, can also be damaged by lightning strike or the discharge of static charge build-up. Such damage can produce pin holes that can then serve as path for water ingression, leading to secondary damage. Thermal degradation of composites can occur on structures near hot exhaust and chemical degradation of the adhesive bond can result from hydraulic fluid damage. A schematic representation of defects and flaws in composites is given in Figure 2.3.

[1] A radome (which is a blend of radar and dome) is a structural, weatherproof enclosure that protects a microwave (e.g. radar) antenna. The radome is constructed of material that minimally attenuates the electromagnetic signal transmitted or received by the antenna.

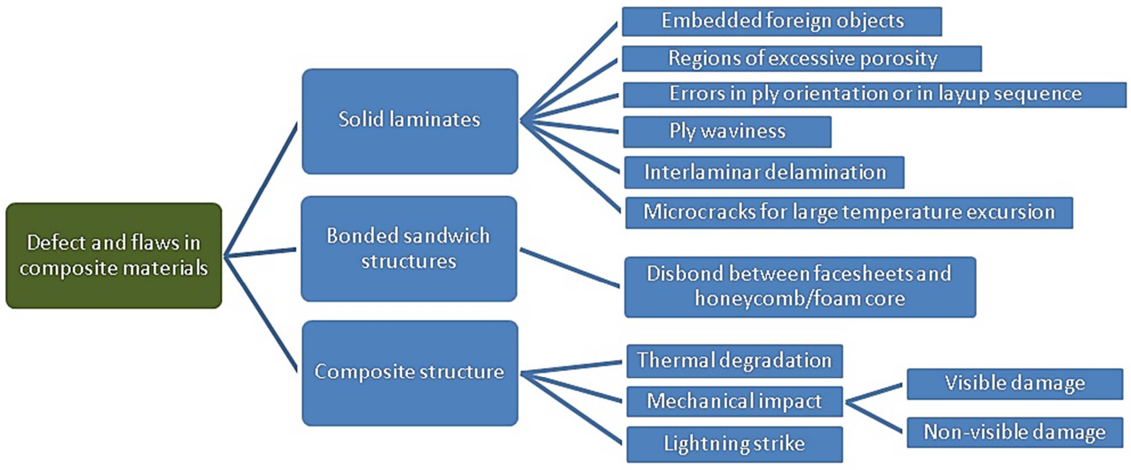

Defect and flaws in composite materials
Solid laminates
Bonded sandwich structures
Composite structure
Embedded foreign objects
Regions of excessive porosity
Errors in ply orientation or in layup sequence
Ply waviness
Interlaminar delamination
Microcracks for large temperature excursion
Disbond between facesheets and honeycomb/foam core
Thermal degradation
Mechanical impact
Lightning strike
Visible damage
Non-visible damage

Figure 2.3 – Schematic representation of defects and flaws in composite materials.

2.2.2. Geometries of aeronautical components

An aspect that has a fundamental influence on the NDT inspection of aeronautical components is their geometry (see Figure 2.4). In this respect we may have to deal with the complex geometries generated by parts of great size and length. Figure 2.5 shows the typical geometry configurations of aerospace parts. Generally speaking, the geometries to be inspected are often the sum of a series of defined profiles of the L, T, U, H and closed H type. These are combinations of flat and curved surfaces. The following figures show conceptual and actual examples of the aforementioned geometries [15].

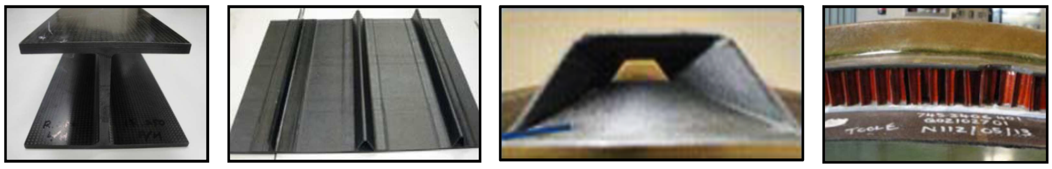


Figure 2.4 – Some examples of aeronautical components [15].

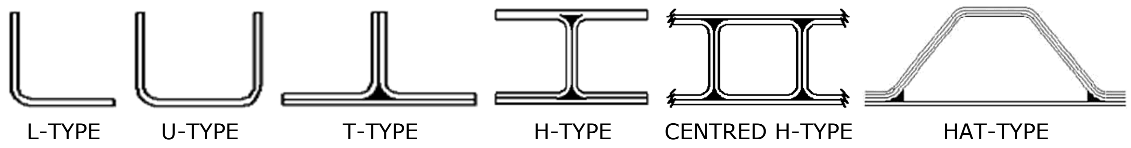


Figure 2.5 – Most common configurations for aeronautical geometries.

The geometric parameters that are essential to define an adequate inspection technique are the thicknesses, radii and angles [16]. These parameters may vary considerably, as a result of which it will be necessary to study each case individually before defining the inspection solution to be applied. The usual ranges are given below.

- Thicknesses: 1-50 mm
- Radii: 3 mm – without limit

- Inner angles: between 80 and 180 degrees (it influences the physical space available for the probe)
- Outer angles: between 0 and 180 degrees.

2.3. Techniques overview

The inspection of composite materials is a challenge and inherently troublesome due to the nature of the material itself. This is compounded by the increasing complexity of individual parts and the sheer size of the area to be inspected. The new generation of safety-critical and geometrically complex composite parts require fast and reliable NDT methods applied in production as well as during maintenance. Due to the large variety of composite materials and the diversity of the NDT techniques, it is difficult to address all aspects of the composite non-destructive inspection (NDI) topic. An attempt is made to briefly review the NDT techniques available today [17] for composite structures used in the industry and to comment on their relative advantages and limitations. The most basic form of non-destructive testing is visual inspection. It provides a mean of detecting and examining a variety of external flaws on many items either in-service or at various stages during manufacture. Visual inspection is the most widely used method for detecting and examination of surface cracks and their relationship to structural failure. A component that is not rejected by visual inspection undergoes further inspections.

The inspection of composite materials presents challenges to conventional NDT inspection techniques that require specialist equipment to overcome. While the term "*composite*" does not specify a particular material or method of production, it is assumed for this thesis that the emphasis is toward polymer matrix composites (PMC) that comprise materials such as carbon fibre reinforced epoxy.

Fragility of composite material to impact loading limits their application in aircraft structures. In particular, low velocity impacts can cause a significant amount of delamination, even though the only external indication of damage may be a very small surface indentation. This type of damage is often referred to as barely visible impact damage (BVID), and it can cause significant degradation of structural properties [18]. Therefore, the need to develop improved and more efficient means of detecting such damage, together with the concepts of maintainability, reparability and inspectability, is starting driving the design of modern composite components (design for maintenance).

2.3.1. Contact and coupled ultrasonic techniques

Critical composite structures must pass through the non-destructive inspection process before assembly onto an aircraft. The primary method used by the manufacturers is ultrasonic testing (UT). UT permits the detection of small flaws with only single surface accessibility and is capable of estimating location and size of the defect providing both surfaces are parallel. On microscopically homogenous materials (i.e. non-composite) it is commonly used in the frequency range 20 kHz to 20 MHz. With composite materials, the testing range is significantly reduced because of the increased attenuation, so the operating frequency limit is usually 5 MHz or less. However, the ability to resolve small flaws is also reduced.

A couplant is necessary to permit the transmission of ultrasonic waves from the probe into the test specimen by reducing the impedance mismatch between air and the inspection surface. In contact UT, as is typically used in manual inspection, a thin film of oil, glycerine or water is generally used to displace the air gap between the transducer and the test surface. The thickness of the coupling fluid layer must be maintained within a narrow tolerance to reduce signal amplitude fluctuation and possible decoupling of the probe. Immersion ultrasonic testing (IUT) further improves reliability by totally submerging the test piece and probe in water, therefore maintaining a continuous column of couplant in all probe orientations. However, when the specimen is very large, submersion becomes impractical due to the excessive cost and size of immersion tanks. Another problem is that if the specimen is buoyant, large forces are needed to keep it submerged. An alternative to submersion is the squirter jet technique, where the ultrasound is coupled by a water jet applied by specially designed nozzles. The ultrasound waves travel through a narrow column of water, which is constantly projected onto the specimen surface. This is much more tolerant of changes in surface contours and is clearly a more practical solution for large or highly buoyant specimens. The ultrasonic inspection of composite parts of large size and with complex geometries is typically carried out in a scan gantry with squirters, in pulse-echo (PE) or through-transmission ultrasonic (TTU) techniques. In the latter case, two water jets carry the ultrasonic beam from the transmitting transducer to the solid laminate or honeycomb sandwich structures, through the structure and then to the receiving transducer. The scan images produced by the ultrasonic squirter systems can reveal the presence of foreign object inclusions, interlaminar delaminations, skin-to-core disbonds, and regions of excessive porosity. TTU scan images of solid laminates based on the amplitude of the transmitted ultrasonic signal (or “dB drop”) are often used in the estimation of the porosity volume fraction.

2.3.1.1. Pulse-echo

Ultrasonic pulse-echo is a well-established and widely used NDT technique. Ultrasonic pulses are transmitted through a couplant into the specimen in a direction normal to the surface and received on the same side of the test piece after being reflected from the opposite face. The signals which travel back towards the probe are detected and the position and size of a flaw is determined from the total pulse travel time and detected amplitude respectively. This is the “A-scan” display and it consists of a series of peaks, the position of which along the horizontal axis can be calibrated in terms of the depth in the composite. The amplitude of each echo will give some indication of the size and nature of the reflector, which might be a flaw or a specimen boundary. Typically, immersion testing is used to carry out area scans where the ultrasonic data can be represented in ways that make it easier to interpret. In the “B-scan” display a 2D “slice” through the specimen is produced by scanning the probe along the surface and capturing an A-scan at each point along the scanning direction. A “D-scan” is also a slice through the specimen, the plane of section is perpendicular to the B-scan plane. In B and D scans, the amplitude at each point is displayed as a colour or a grey-scale value according to a predetermined palette. In general, the higher the amplitude the brighter or more intense the displayed point. C-scans may be thought of as “Top down” or “Plan” views of the component. Here the A-scan is “gated” to display either the amplitude or the depth of signals occurring within a predefined depth range within the component.

Pulse-echo scans require minimal part preparation and exhibits high sensitivity allowing the detection of very small flaws. It also allows accurate determination of defect position, size, shape and depth which leads to a good estimation of defect severity. It is sensitive to both surface and subsurface discontinuities and can also be utilised for thickness measurement. This technique only requires one surface access which is a particularly useful advantage as such testing situations often occur in practice.

As far as the disadvantages are regarded, firstly the surface must be accessible to transmit ultrasound. UT demands a high level of operator skill so requires more extensive training than some other methods. Materials that are rough, irregular in shape, very small, exceptionally thin or not homogeneous prove very difficult to inspect. There is also a time period approximately equal to the pulse length during which the probe cannot detect signals because it is either emitting a pulse or reverberating. Consequently, there is a region near the probe in which defects cannot be detected. This is sometimes referred to as the dead zone. The problem can be limited by using a shorter pulse, immersion testing, a stand-off probe, or employing a twin crystal probe. Linear defects oriented parallel to the sound beam may also go undetected unless the appropriate beam path angle has been attained in the set-up phase. Reference standards are used for both equipment calibration and the characterization of flaws.

2.3.1.2. Through-transmission

Through-transmission is an ultrasonic technique that uses two transducers: one emitter and one receiver, the receiver being placed on the opposite side of the component and facing the transmitting probe. Where a water jet system is not available a standard immersion tank may be used to carry out double through transmission. Here the pulse passes through the material and is bounced off a reflector, such as a glass plate, before passing back through the specimen to the original transducer. Through-transmission can also be applied with the contact coupling technique where the two probes are usually held in a jig capable of adjusting itself to the specimen's width.

Compared with pulse-echo there is no "dead zone" which means that flaws can be detected at all depths throughout the thickness of the specimen at the measurement location.

One disadvantage of through-transmission is that it does not provide any information about flaw depth, and is less sensitive to small defects than pulse-echo inspection. Furthermore, the transmitter and detector must be kept well aligned making the inspection of contoured surfaces very difficult, especially when the traditional coupling technique is utilised. Also, it requires access from both sides and this may not be possible in certain applications. Highly attenuating plastic materials, like CFRP have a high frequency dependent attenuation and phase velocity dispersion of ultrasonic waves resulting in reduced accuracy of results.

2.3.1.3. Phased Array

Phased array (PA) probes typically consist of an assembly containing from 16 to as many as 256 individual piezoelectric crystals that can each be activated separately. The crystals are usually arranged in a linear array, although 2D arrays (annular, circular, etc.) are available, as well as more complex arrangements and shapes for specialised applications. PA probes require a computer-based instrument loaded with software that is capable of driving the multi-element transducer, receiving and digitizing the returning signals. The instrument then plots the echo information as B, C and S scans (or sector scans). The ability to display multiple scans on the instrument screen at any one time allows for better interpretation of the scan data [19].

PA systems are capable of sweeping and steering an ultrasonic beam through a range of steering angles or along a linear path, as well as being able to dynamically focus at different depths. These capabilities increase the flexibility and capability of inspection setups and are achieved utilising the phasing principle. Beam steering, commonly referred to as sectorial scanning, can be used for mapping components at appropriate angles. This can greatly simplify the inspection of components with complex geometries as-well as increasing the probability of detection[2] (POD) for awkwardly orientated defects. Electronic focusing permits beam shape and size optimisation at the expected defect location, thus further increasing POD. The ability to focus at multiple depths also improves the ability for sizing critical defects for volumetric inspections. Focusing can significantly improve signal-to-noise and electronic scanning across many groups of elements allows for C-Scan images to be produced very rapidly.

Electronic scanning is many times faster than mechanical scanning. Electronic focusing allows the use of a single probe for working at different depths, and electronic deflection allows the angles of incidence to be varied.

However, PA systems require a large initial outlay of capital due to the cost of the probes, the dedicated computer control unit and the operator training cost.

2.3.1.4. Full Matrix Capture

Full Matrix Capture (FMC) is a technique derived from PA. It allows the maximum amount of ultrasonic data to be captured from a standard phased array probe. In the FMC technique each element transmits a single ultrasonic pulse in turn [20]. After each pulse the signal received at each of the other elements is recorded. The resulting information matrix must then undergo post-processing so that the raw data can be displayed effectively, allowing interpretation by the operator.

[2] The concept of Probability of Detection (POD) was developed mainly in NASA in the USA during the 1970s. POD tries to assess a minimum flaw size that will be reliably detected by a specific NDT technique. This is best done by plotting the accumulation of flaws detected against the flaw size of all the flaws "detected" or that produce a response over some threshold. Ideally all flaws over some critical size will be detected and flaws smaller than that are not "detected". The tool most commonly used for POD description is the POD curve.

FMC has generally resulted in greater inspection sensitivity and hence increased probability of detection of defects [21]. Nevertheless, FMC requires complex signal processing to effectively render and display the ultrasonic data. This requires advanced equipment with high processing power. FMC is currently in its infancy and therefore is predominantly laboratory based. Due to the need to pulse from each element in the array before moving to the next pulse point, scanning times are generally longer than conventional phased array.

2.3.1.5. Time-of-flight-diffraction

The time-of-flight diffraction (TOFD) method is an ultrasonic sensitive and accurate testing method for the non-destructive testing of welds. TOFD was invented in the UK in the 1970s initially as a research tool [22]. It originated from tip diffraction techniques, which were first published by Silk and Lidington in 1975 [23].

Bray and Stanley (1997) summarized TOFD as tip-diffraction techniques which utilized the principle that the tips of a crack when struck by a wave will diffract the signals back to the other location on the surface. The depth of these tips can be determined from the diffracted energy [24].

Although TOFD is mainly targeted to the inspection of welds in metals, the United Kingdom Atomic Energy Authority (UKAEA) has recently adapted the technique and developed inspection procedures to locate, identify and size defects in glass reinforced plastic (GRP) structures [25]. The procedures cover defects such as delamination, flange cracks, fractures, cracks, tears, incorrect lamination, disbonding, lack of adhesive, material degradation, porosity and voids, inclusions and foreign bodies, and incorrect dimensions. Tests have shown that TOFD can accurately identify and size defects in GRP laminates of practical construction and thickness. The technique also allows for layup verification, as it is possible to identify the individual layers in the laminates.

Compared to conventional methods of ultrasonic testing, TOFD enabled crack sizes to be measured more accurately, so that expensive components could be kept in operation as long as possible with minimal risk of failure.

However TOFD has two blind zones where it is not sensitive to defects, hence TOFD ought always to be supplemented by a conventional pulse-echo examination of the near and far walls. The pulse-echo probes are commonly mounted on the same buggy as the TOFD probes. Another drawback is the need of ultrasound technicians with advanced training.

2.3.1.6. Back-Scatter

Fibre orientations and their stacking order determine the final properties of a composite material sample. In its most common configuration, back-scatter is a variation on pulse-echo in which the transducer is inclined at an acute angle (α) with respect to the normal to the test structure surface. The echoes received by the transducer are monitored as the component or transducer is rotated about an axis normal to the component surface, the angle α being kept constant. When the angle of rotation, β, is such that the transducer is normal to the fibre direction in any of the layers of the structure, the back-scattered signal reaches a maximum. A plot of the signal intensity versus the angle of rotation is recorded.

This technique is one of the few ways to actually assure the ply lay-up sequence. It may be used to check the stacking sequence. For example, the fibre orientation of graphite epoxy laminates in several lay-ups is easily resolved. It can also detect porosity because if it is present, the back-scattered signal is higher at all angles of rotation compared with a sound specimen. This technique will also detect local fibre waviness, ply-end discontinuities and trans-laminar cracks. It can also be used to detect matrix cracks due to thermo-elastic stress.

However, the technique is very surface-orientation sensitive and a sophisticated set-up is required to maintain correct alignment. This inherently makes inspection of complex geometries troublesome. A near perfect surface is also needed for this type of inspection which makes surface preparation critical.

2.3.1.7. Ultrasonic Spectroscopy

Ultrasonic spectroscopy is a form of ultrasonic test in which the pulse arriving at the detecting transducer is analysed harmonically. Information relating to the composite sample or features within it can be deduced from characteristic changes of the frequency spectrum. When the specimen is much thicker than the pulse length, the pulse spectrum can be analysed to reveal frequencies at which the ultrasound is attenuated or scattered by features in the composite. At the identified frequencies, the reduced amplitude can be correlated with flaws, provided a calibration standard is available for comparison. When the pulse length is greater than the specimen thickness, e.g. for thin-walled specimens, interference can occur between transmitted and internally reflected beam components, so that the detected amplitude is the superposition of these.

As well as thickness scans, a spectroscopic system can be used to determine the delamination depths as differences in these will also affect the resonance frequency. If the thickness is stable and well defined, the ultrasonic velocity changes can be used to monitor the location (in the scanning direction) of poor matrix curing. Regions of porosity may also be detected with ultrasonic spectroscopy. Ultrasonic spectroscopy can be better than pulse-echo for determining the thickness of thin specimens. This is because for such specimens it may be difficult to resolve pulses reflected from the front and back-walls in the time domain. By comparison, resolution in the frequency domain may be much easier, hence spectroscopy would be preferable.

The frequency spectrum of the returning signal will be affected by attenuation within the material and therefore comparisons between different specimens are only possible when the attenuation is well known in both. Also, most techniques use a wide-band, pulsed input which can suffer from poor signal-to-noise ratio.

2.3.2. Non-Contact ultrasonic technique

Although the squirter-operated, water-coupled ultrasonic technique remains the prevalent method of quality assurance inspection for manufacturing or maintenance of composite parts, air-coupled ultrasonic scan have become increasingly practical. Air-coupled ultrasonic inspection (ACUI) avoids the disadvantages of coupling liquids like water, corrosion, air-bubbles in the coupling path, algae in the water, problems with filter system for water, influence of force of gravity and others. It is particularly suited for inspecting honeycomb sandwiches with perforated facesheets and other water-sensitive composite structures. However ACUI produces a large acoustic mismatch between the transducers and the test component so that a special system with adapted transducers is necessary. The development of ACUI started for aerospace components at the end of the 1990`s [26]. A recent paper [10] describes the application of ACUI for a complex helicopter composite tail boom.

The advantages of ACUI are dearly paid by a large acoustic impedance mismatch between air for coupling and solids such as transducers and the test component. In a typical inspection configuration, four interfaces are encountered as the acoustic waves travel from the transducer face, through the air into the specimen, and then back through the air to the receiving transducer. Signal losses at these interfaces are large and the resulting low signal to noise ratio (SNR) has been problematic. Moreover the sound attenuation of the air has to be taken in account, which is proportional to the frequency. In order to avoid additional signal damping, the useful frequency range is situated below 1 MHz. Materials with a high sound damping such as sandwich components can only be inspected with low frequencies in this range. Therefore ACUI even delivers advantages because of the smaller wavelength in the coupling medium (air in comparison to water) and the smaller sound beam diameter by the same test frequency. Usually air-coupled ultrasonic testing is carried out in through-transmission technique with separate transmitter and receiver transducers on opposite side of the test component. There are also methods with a single sided access (pitch and catch arrangement). Usually a Lamb wave is generated in the component. The conversion from the transmitted longitudinal wave to guided waves is carried out by an angular incidence with same orientation of the transducers [27]. Due to the requirement of precise adjustment of the angles at all positions (also of large and curved components) this method is usually limited to laboratory applications with flat components.

2.3.3. Other relevant techniques

2.3.3.1. Acoustography

In this approach, a wide-area acousto-optic (AO) sensor is employed to provide real-time ultrasonic images over the area of the sensor [28]. The UT field is generated by a conventional probe and focused onto the sensor by a system of lenses after having interrogated the test specimen [17]. The most important advantage that acoustography has over other UT inspection methods is speed. However the resolution of images obtained by acoustography is generally grainier than typical conventional UT C-Scans. Improvements can be achieved but only at the expense of time by allowing the sensos to dwell longer at each scan point. The other main drawback of acoustography is the inability to obtain information on depth of features detected in the specimen.

2.3.3.2. Low frequency methods

Low frequency vibration methods are those in which bulk disturbances are excited in the structure [29]. The frequencies of these vibrations are dependent on the dimensions of the structure but for composites the frequency range below 20 kHz is often used. They can be grouped into global and local methods [17].

Global methods measure the response of the entire structure from a single point. The measured global properties are modal frequency, modal damping or mode shape measurement. In local methods the specimen is excited and its measured response is only representative of its local condition. These methods include measurement of mode shape changes, the coin tap-test, and the mechanical impedance method. In both global and local methods, the excitation source may be a transient or continuous signal. As the acoustic impedance difference between air and composite structures is less at the lower frequencies, actuators (vibration transmitters) and sensors can be mechanically coupled, i.e. acoustic coupling fluid is not necessary. However, the lower frequency means that these methods cannot detect flaws as small as ultrasonic techniques.

2.3.3.3. Optical Holography

The mode shape may be measured by optical holography, which is a non-contact method of measurement [30, 31]. This is a highly expensive approach, and unless a pulsed system is used the specimen must be mounted on a table to isolate it from extraneous vibration. Flawed regions produce local changes in the mode shape (hence this method is otherwise known as local amplitude measurement). The illumination source has to be highly coherent for holography and hence a laser is used.

2.3.3.4. Tap Test

In the original version of this test the surface was tapped by a coin or metal hammer and the difference in local stiffness was detected by the difference in the sound produced [32]. In fact this method is still widely used but it is limited by the inability to identify the type of flaw and to quantify its size. It is also subject to inconsistencies due to ambient noise, operator hearing sensitivity and tapping force. The use of semi-automated systems that measure the contact duration, such as the Mitsui “Woodpecker” and the “WichiTech RD3” instrumented hammer [33], or the Computer Aided Tap Test (CATT) system [34] can overcome these limitations. In the modern tap test, the structure is tapped with an instrumented hammer in various locations and a load cell mounted in the tip records the interacting force. Where there is a difference in the local stiffness between two regions the spectra will look markedly different. The most common measure of the local stiffness is the contact duration, which can be plotted as a function of position on the surface. This can be used to produce a C-Scan of the inspected area providing a permanent record of the inspection and enabling intuitive interpretation of the results. This test is fast, low cost and efficient for composites.

2.3.3.5. Mechanical Impedance Analysis

Mechanical impedance analysis is like the coin-tap test except that the excitation is a continuous harmonic rather than an impulse. The physical principles of the two techniques are very similar and therefore they have similar sensitivities [35, 36]. For both, their sensitivities fall off with increasing flaw depth in the specimen and so they are better at detecting near-surface flaws. The relationship between the phase and amplitude of the driving and received voltages is dependent on the local impedance, so changes in the amplitude or phase of the received signal can indicate local changes in structural impedance. The probe is usually mounted within a spring loaded holder to ensure constant contact force. Samples containing known good and bad areas are required in order to configure the test.

2.3.3.6. Membrane resonance method

The membrane resonance method is similar to the mechanical impedance method except that it specifically measures local resonant vibration, and the excitation source is non-contact [37, 38]. It works on the principle that collectively the layers above a delamination constitute a membrane constrained at the perimeter of the delamination. It follows that the thickness of the membrane is the same as the depth of the delamination and that this together with the composite properties and the membrane equivalent diameter determine the frequency at which the membrane resonates. Therefore at frequencies around the membrane resonance, the vibration response for a given input force amplitude will be greater near the delamination than in a sound region. The difference in dynamic impedance between a good region and a defective region is significantly greater for the membrane resonance method than observed for the mechanical impedance method. Consequently it is not necessary to place such stringent controls on the input force making it a potentially simpler method.

2.3.3.7. Radiographic Inspection

Radiography inspection is the deployment of ionising radiation and detection media to the non-destructive imaging of structures [39]. For composite inspection two types of radiation are relevant: X-rays and neutrons [40-43]. The basic principle is that parts of the specimen, with differing material densities and hence radiation absorption properties, can be discriminated in an image formed by the beam transmitted through the specimen and reaching the detection media.

By far the most common form of radiography uses X-rays and falls into two broad categories: conventional X-radiography, which uses only the radiation beam; and enhanced X-radiography in which specially formulated liquids are used to enhance the contrast of the radiographic images. Because the polymer matrix and some fibres such as carbon have low absorption cross-sections, good contrast is difficult to achieve and low energy X-rays must be used to ensure that some absorption takes place (10 kV to 50 kV). In conventional radiography, a beam of X-rays bombards the specimen and the unabsorbed radiation passing through the specimen hits a radiation-sensitive target (a film or digital X-ray detector). The radiation exposes the film emulsion in a similar manner to the way that light exposes film in photography. Development of the film produces an image that is a two dimensional "shadow picture" of the specimen. Variations in the intensity of unabsorbed radiation, and therefore of radiation absorption in the specimen, appear as shades of grey in the developed film. Evaluation of the radiograph is based on comparison of the differences in density produced by an unknown specimen and that produced by a similar specimen of acceptable quality or neighbouring area. In micro-focus X-radiography, the x-ray beam is emitted from a focal spot of 100 μm or less. This is typically 20 times smaller than a conventional source, hence producing a sharper image. Some systems have focal spot sizes as low as 5 μm allowing geometric magnification to be used to image very small defects without the associated penumbra effect.

Flaws that do not have appreciable depth in the direction of the beam cannot be detected by conventional radiography that uses a single X-ray shot. Cracks cannot be detected unless they lie parallel to the radiation beam. For similar reasons, delaminations perpendicular to the beam are almost impossible to find with conventional radiography. Fibre volume fraction variations are difficult to detect in CFRP composites with conventional radiography because both the matrix and the reinforcement have similar radiation absorption properties. Ionising radiation is particularly hazardous to health if exposure is not carefully controlled. Consequently, it is necessary to ensure that adequate shielding is provided for technicians and to follow the guidelines set out by the national regulatory body on radiation protection (the National Radiological Protection Board or NRPB in the UK) [44]. Furthermore the chemicals used for enhancement are covered by the COSHH (Control of Substances Hazardous to Health) regulation [45]. This means that the total cost both in terms of time and money is significantly greater for radiographic inspection methods than for other methods of NDT.

2.3.3.8. Computed Tomography

X-ray computed tomography allows 3D images of components to be generated [46]. A series of typically 360 to 720 radiographs are taken by rotating the object about one axis. The resulting data cloud is reconstructed in software to produce a 3D image of the component that can be viewed through any image plane. By assigning appropriate grey values, components such as metal fixings can be extracted from the image and viewed separately [47]. The main advantage is the possibility to detect cracks of any orientation. The drawbacks are the high cost and the limited sample sizes.

2.3.3.9. Shearography

A near-surface flaw will decrease a composite component's local strength and therefore the surface will deform differently under loading if a flaw is present. These differences are very small so a technique based on optical interference is suitable to detect them [48, 49]. In shearography, the specimen is illuminated with a divergent laser beam and the scattered light is projected onto an image plane by a special shearing lens. This lens effectively shears the image of the specimen in the plane of the lens allowing interference between the sheared and direct images. This pattern is effectively a map of the distribution of local surface strain and represents a *reference image*. A stress distribution is then applied to the surface (using vibration, sound, pressure, vacuum or thermal loading) and a second interference pattern is recorded. This stress distribution can be either in plane or out-of-plane. Superposition or subtraction of the images obtained with the structure in the stressed and unstressed states will produce a fringe pattern. Each fringe is a line of iso-strain and areas of increased strain are shown by concentrations of fringes.

2.3.3.10. Thermography

Thermographic methods are those in which the presence of flaws is determined by monitoring the flow of heat over the surface of a structure after some external introduction of a temperature gradient [50]. The presence of flaws disrupts the normal pattern of heat flow that would be expected in a sound structure. The method is more sensitive to flaws near to the surface. Modern thermographic systems commonly use infrared (IR) cameras to detect radiated heat and are controlled by video electronics which sample the field of view at a typical rate of 50 Hz, allowing temperature variations on a 20ms time-scale to be resolved [11]. The camera is sensitive to temperature changes of about 0.005°C and covers a chosen range of temperature, 4°C and 8°C being commonly suitable, although operation is possible between -50°C and +100°C. Liquid crystal coatings and pyroelectric detectors have also been used to detect IR radiation [51].

Thermographic methods fall broadly into two groups: active methods, and passive methods. Active methods are those in which the thermal gradient is produced and continuously maintained by the application of cyclic stress. Passive methods are those in which the thermal gradient results from a transient change. Passive methods are the most widely applied NDT technique in composites inspection.

2.3.4. Emerging techniques

Although conventional methods are in most cases capable of detecting the sought flaws, the speed of inspection can be too slow. Faster and more reliable methods are constantly developed in order to improve the inspection throughput, reduce inspection variability and drive production costs down.

2.3.4.1. Developments in ultrasonic testing

While commercial systems for air-coupled ultrasound are available, considerable research is still being carried out into further developing the technique. These developments include the use of alternative modes of wave propagation such as lamb waves, and hybrid techniques that combine air-coupling, lasers and electromagnetic acoustic transducers (EMATs) into a single system.

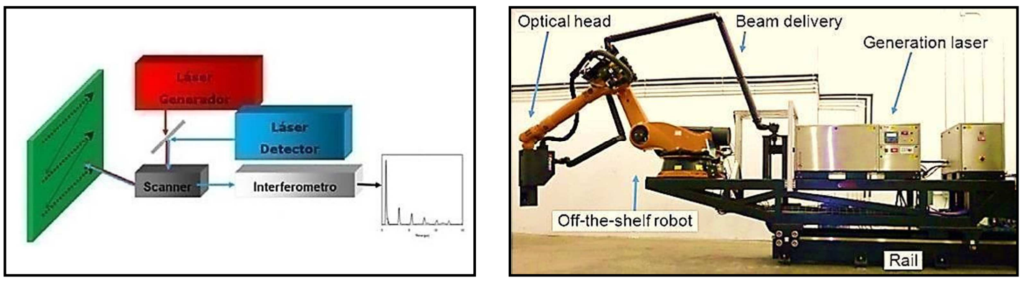
- *High-power pulser systems for air-coupled technique*

One particular area of development in air-coupled systems is in high-power, flexible ultrasonic pulser systems to improve the transmission of acoustic waves into the specimen. One high-power pulser, capable of delivering up to 800V in a spike excitation to a transducer, has been developed [52]. It can also operate using a double pulse mode to increase the axial resolution of ultrasonic inspection through a pulse cancellation technique. The limitation on the driving voltage is given by the depolarization of the piezoelectric ceramic element of the transducers [53]. These developments can improve the non-contact ultrasonic inspection of aircraft structures with honeycomb cores, foam filled materials, polymer matrix composites (PMCs), solar cells and other porous materials that may be damaged using conventional liquid coupling based inspection techniques.

- *Laser ultrasound*

After the invention of powerful pulsed lasers in 1960s, laser sources have been used for generating waves in the ultrasonic range [54].

A number of different physical processes take place when a solid surface is illuminated by a laser. At lower incident powers these include heating, the generation of thermal waves, elastic waves (ultrasound) and, in materials such as semiconductors, electric current may be caused to flow. At higher powers, material may be ablated from the surface and plasma formed, while in the sample there may be melting, plastic deformation and even the formation of cracks. If lasers are operated below the power of ablation regime, they are suitable for non-destructive testing. Electromagnetic radiation from the laser is absorbed in the surface region of a sample, causing heating. Thermal energy then propagates into the specimen as thermal waves. The heated region undergoes thermal expansion, and thermoelastic stresses generate elastic waves (ultrasound) which propagate deep within the sample. It is possible to develop NDT systems based on laser generated ultrasound [1, 55]. This technology combines features of optical inspection (contactless) with those of conventional ultrasonic testing (internal examination). A laser pulse is directed onto the surface of the component under test and generates an ultrasonic wave, propagating in the material. As for conventional ultrasonic testing, the signal is reflected by flaws and interfaces but it can be detected optically, using a second (long pulse) laser and an interferometer (Figure 2.6a).



(a) (b)

Figure 2.6 – a) Diagram of ultrasonic laser generation and detection, b) Configuration of Ultrasonic Laser Equipment [15].

The main advantage of this technique is that the waves always propagate normally to the component's surface, independent of the incidence angle of the laser light, and is therefore ideal for inspecting parts with complex geometry where conventional multi-channel systems cannot follow the profile or where manual testing is not economic.

A laser-ultrasonic system made of two units, a detection unit and a generation unit linked by optical fibres, was presented in 1998 by *Ultra-Optec* for the detection of corrosion in airframes and for the measurement of the thickness of thermal barrier coatings [55].

In May 2011, *Tecnatom* [15] delivered an inspection system based on laser ultrasound, the configuration of which is shown in Figure 2.6b. This means that laser ultrasound (LU) systems are available for industrial composite inspection.

- *Hybrid systems*

Early research has shown the potential of LU to generated Lamb waves which are detected using air-coupled transducers [56]. The use of two technologies explains the term "hybrid systems". Monitoring structures with long-range guided waves is attractive because it is possible to test complete material volumes at distances of over 100m from the point of measurement. Guided-wave ultrasonic inspection differs from conventional ultrasonic technology since the ultrasound wave along the length of the component rather than through it. The process of fibre placement of thermoplastic matrix composite structures requires real time process control and rugged non-destructive evaluation (NDE) sensors. The surface acoustical stress wave methods combined with remote laser/air coupled ultrasonic transducers are in development for on-line, in process control of fibre placing operations [8]. Figure 2.7 is a diagram of the hybrid ultrasonic test configuration using laser ultrasound and air/gas coupled ultrasonic transducers. These advanced non-contact ultrasonic methods are adapted for surface wave testing in the frequency range of MHz that is not readily achievable using contact transducers.

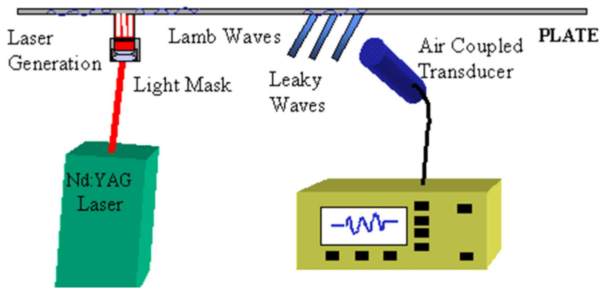


Figure 2.7 – Diagram of the hybrid, remote, non-contact ultrasonic test configuration using laser generation and air coupled detection [8].

Non-contact ultrasonic configuration enables non-interfering defect detection and material characterization for in-process inspection and on-line process control. Recent work explores a two-dimensional propagation of Lamb waves generated using a laser and a shearography system to capture the distortions in CFRP samples [57]. This system applies a 20mJ Nd:YAG laser to a 4mm quasi-isotropic CFRP plate consisting of 8 layers.

Regions containing delaminations are visually identifiable in the 2-D images as the Lamb waves interact with the edges of the defect. The defect is still found to be visible after the main wave front has passed as the mode converted waves continue to resonate.

- *Eddy Current Testing*

Ultrasonic methods and other techniques all have their limitations and a number of research groups have investigated the use of a modified form of the eddy current (EC) technique to test CFRPs [58]. This is based on HCT (high critical temperature) superconducting quantum interference devices (SQUIDs). Traditional EC testing is challenging with CFRP because of its low electrical conductivity but SQUIDs can detect the very small changes in magnetic field arising from material defects. These replace the conventional coils used in EC testing. A group from the University of Naples has studied this approach for some years [59, 60]. Multidirectional cross-ply composites based on an epoxy matrix reinforced with layers of carbon fibres were successfully inspected for fatigue and impact damage using this technique [61]. The SQUID was cooled to 77K in liquid nitrogen and exhibited very high magnetic field sensitivity in an unshielded environment – less than 0.3 pT/√Hz rms for frequencies above 100 Hz. Carbon nanotube (CNT) reinforced polymer (CNRP) is a significantly stronger and lighter material than CFRP. In 2011, Lockheed Martin announced that it would be using this in its F-35 Lightning II multi-role fighter aircraft for several non-load-bearing airframe components [1]. Significantly, the EC/SQUID concept has recently been extended by the Naples group for testing CNRP nano-composites comprising multi-walled CNTs in an epoxy matrix [1].

- *Terahertz (THz) imaging*

THz radiation has wavelengths in the range 100mm (IR) to 1.0mm (microwave) and has the ability to penetrate and interact with non-metals. While enjoying growing use in security screening, its use in NDT is still at an early stage. THz imaging technology has been used by NASA for detecting flaws in the foam of the Space Shuttle’s external tank thermal protection system [1]. Several groups are investigating its role as a means of detecting defects in composite materials, using reflection, transmission and time-domain spectroscopic techniques. An example of a recent collaborative project is the European *DOTNAC* programme (Development and Optimization of THz NDT on Aeronautics Composite Multi-layered Structures) whose stated aim is to develop a fast, high resolution, non-invasive and non-contacting inspection system for assessing composite aeronautical components during production [62]. As yet, relatively little is known of how THz radiation responds to composite defects such as delamination, porosity and inclusions, so the DOTNAC programme will break new ground.

2.4. Semi-automated and Automated NDT

2.4.1. Current needs for automated NDT solutions

The previous section explained that various NDT techniques may be used to perform non-destructive inspection of composites materials. Even if new techniques are continuously being developed, the ultrasonic techniques are nowadays the most used methods. Non-destructive inspection may be performed manually by technicians who typically move ultrasonic probes over the structures. Manual scanning requires trained technicians to move sensors over all portions of the structure needing inspection. While manual scanning may be required around the edges of the structure and the edges of holes in a structure, there is a need in the industry of composite materials for developing reliable automatic solutions.

The tendency in recent years has been to automate the entire process of manufacturing aeronautical components, including the NDT inspection phases. The fundamental objective of automation within the inspection process is to reduce the time needed for scanning the samples, post-processing and analysing the acquired data and sentencing the results. Although necessary, NDT should never be the bottleneck in production processes. Manual inspection in a production environment is not practical in terms of time and operator costs. High throughput systems allow inspection to take place early in the manufacturing process, before too much value is added to the products. Automated systems allow fuller inspection on large areas.

2.4.2. General overview of manipulator types

One of the first elements to consider in the design of semi-automated or automated NDT solutions is the selection of appropriate manipulators to move the probes over the areas of interest. In order to design a semi-automatic or automatic system, it is important to choose the right manipulator type for the required inspection application. The current alternatives are the followings:

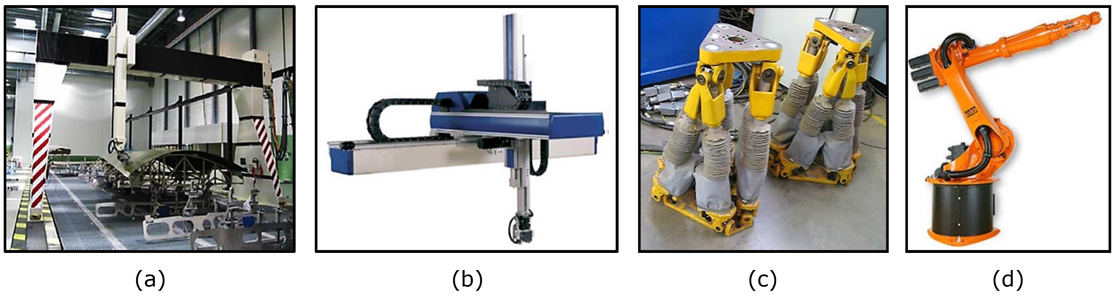
- Bridge-type mechanism
- Linear manipulators

- Parallel manipulators
- Robotic arms

Bridge designs provide the most stable conditions in terms of positioning accuracy. The main degrees of freedom (DOF) are linear X-, Y-, Z-; C-axes are used to adapt to the part surface [63] (Figure 2.8a). The use of these systems to inspect parts with noncomplex shapes (planes, cylinders or cones) is widespread [6]. These systems usually consist of specific machines or machines, which are used to inspect identically shaped and/or sized parts.

Linear manipulators (Figure 2.8b) are usually used to move sensors over simple geometries. Three linear manipulators can be easily used to obtain 3-DOF machines.

A parallel manipulator (Figure 2.8c) is a mechanical system that uses computer-controlled parallel chains to support a single platform, or end-effector[3]. Perhaps, the best know parallel manipulator is formed from six-linear actuators that support a moveable base for devices such as flight simulators. Their "parallel" distinction, as opposed to a serial manipulator (or robotic arm), is that the end-effector (or "hand") is connected to its base by a number of separate linkage working in parallel. The control of the two-link parallel manipulator was investigated in 2005 [64], obtaining a reduced order nonlinear analytical model that can be used as a basis for accurate and computationally inexpensive control for maximized displacements within three-dimensional space.


(a) (b) (c) (d)
Figure 2.8 – a) Bridge type mechanics example [65], b) Example of 3 axis conventional system, c) Hexapod positioning systems and d) KR16 L6-2 Six Axis Robot from KUKA.

An industrial robot (Figure 2.8d) is defined as an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes [66]. One of the most important parameters is the number of axes or degrees of freedom. Two

[3] A robotic end-effector is any object attached to the robot flange (wrist) that serves a function. This would include robotic grippers, robotic tool changers, robotic collision sensors, robotic rotary joints, robotic production tooling, etc. Robot end-effectors are also known as robotic peripherals, robotic accessories, robot tools, or robotic tools, end-of-arm tooling, or end-of-arm devices. End-effector may also be unhyphenated as "end effector".

axes are required to reach any point in a plane; three axes are required to reach any point in space. To fully control the orientation of the end of the arm (i.e. the wrist) three more axes (yaw, pitch, and roll) are required, so that the minimum number of axes required for a robot to reach any arbitrary point in its working envelope is six. A robot with six axes is also known as a general manipulator. For through-transmission ultrasonic inspection, a semi-automated inspection system requires access to both sides of surfaces of a structure which, at least in some circumstances, will be problematic, if not impossible, particularly for semi-automated systems that use a fixed frame for control of automated scan heads. A device such as an Automated Ultrasonic Scanning System (AUSS®) [65] may be used. Conventional automated scanning systems, such as the AUSS-X system, therefore require two robotically controlled probe arms that can be positioned proximate the opposed surfaces of the structure undergoing inspection with one probe arm moving an ultrasonic transmitter along one surface of the structure, and the other probe arm correspondingly moving an ultrasonic receiver along the opposed surface of the structure.

In the spectrum of industrial robotic manipulators, KUKA robots seem to offer the best capability to develop automatic NDT systems for the aerospace industry [15]. They present precise mechanical system, the possibility to accurately master each Joint and the ability to export positional data at frequency up to 1 kHz.

2.4.3. Commercial systems

Semi-automated inspection systems have been developed over the years to overcome some of the shortcomings with manual inspection techniques. This section reports about the commercially available automated NDT systems for ultrasonic testing of composites, with particular emphasis to the aerospace industrial sector.

2.4.3.1. Boeing AUSS

The Boeing family of Automated Ultrasonic Scanning Systems (AUSS) products are fully automated C-Scan ultrasonic inspection systems. Numerous models have been developed culminating in the most recent development, the AUSS XVII [65]. This iteration utilises ultrasonic phased array probes and is capable of full digital waveform capture. It features two 6-axis robot arms suspended from a sliding gantry as seen in Figure 2.9, resulting in a 13-axis system with a potential scan envelope of 15 x 6 x 6 metres.

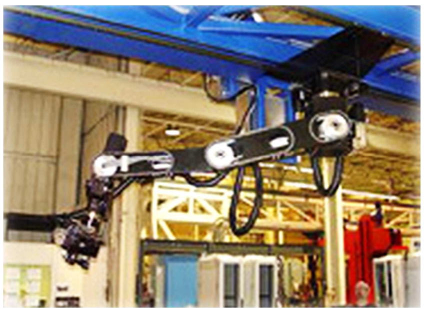


Figure 2.9 – AUSS 6-axis robot suspended on sliding gantry.

The multi-processor configuration allows simultaneous operations during scanning such as through-transmission and pulse-echo inspections from each side in a single scan, even on non-parallel surfaces. The system features seamless re-start, re-scan, sub-scan functions. Scan paths can either be CAD generated or taught and can achieve speeds of 1 metre per second with a 1mm acquisition interval. Adjustable end-effectors can be used for increased "in-side" reach.

2.4.3.2. MAUS System

The Mobile Automated Scanner (MAUS®) System [67] is a mobile scanning system that generally employs a fixed frame and one or more automated scanning heads typically adapted for ultrasonic inspection. A MAUS system may be used with pulse-echo, shear wave, and through-transmission sensors. The fixed frame may be attached to a surface of a structure to be inspected by vacuum suction cups, magnets, or like affixation methods. Smaller MAUS systems may be portable units manually moved over the surface of a structure by a technician.

2.4.3.3. RABBIT

RABBIT is the name of a group of systems developed by TECNATOM in collaboration with KUKA that makes it possible to incorporate the use of industrial robots in NDT inspection systems [15]. These systems adapt the characteristics of industrial robots and incorporate them in an overall inspection system, bringing together all the hardware and software elements required to plan and to configure ultrasonic inspections (specification of probes, definition of the part to be inspected obtaining also geometries by 2D Laser, definition and generation of focal laws, definition of inspection parameters, ultrasonic calibration). The features of these systems include the generation of inspection trajectories (off-line path-planning and 3D robot movement simulations), the control of the safety of all the elements of the inspection cell, the automatic change of probe-holder modules, the acquisition of ultrasonic data with on-line monitoring and the analysis of them [11]. The final result is a group of flexible systems that incorporate one or two KUKA industrial robots integrated with linear tracks, gantries and/or turntables. That allows for adaptation to different non-destructive testing configurations, ultrasound through-transmission (Figure 2.10) or pulse-echo, using conventional or phased-array technology and providing the possibility of automatic exchanging ultrasonic modules (with different phased-array probe configurations).

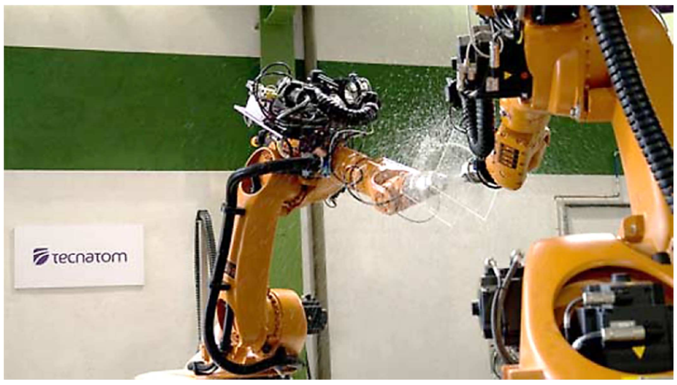


Figure 2.10 – RABBIT Inspection system with two robots operating in transmission mode [15].

At the end of 2013, Tecnatom successfully completed the process of acceptance of a system, shown in Figure 2.11, supplied to Aerocomposit (a Russian technology centre dedicated to the manufacturing of aeronautical components in composite materials) [68].

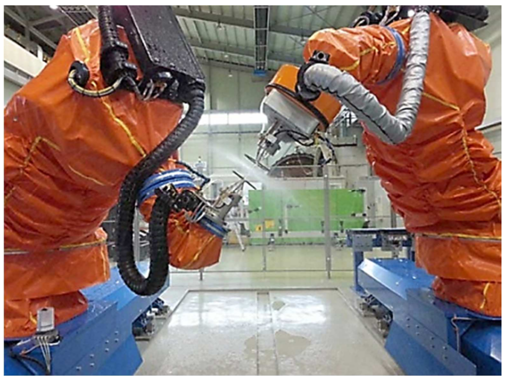


Figure 2.11 – Tecnatom RABBIT system supplied to Aerocomposit [68].

Aerocomposit purchased the RABBIT system and mounted it on a linear track in its factory in Ulianovsk. The system is used for the inspection of wing spars and skins manufactured from composites to support the programmes of the Russian aircraft manufacturer Sukhoi.

In addition to this, another system was supplied to GNTP, The South Korea Research Centre for Composite Materials for the Aeronautical Industry. In early 2013, this technology centre acquired an ultrasonic inspection system based on Twin-Robot equipment running on a linear track and on transmission by water jet or independent inspection in local immersion. The system is dedicated to the inspection of complex geometry helicopter and fixed wing aircraft components manufactured in composites.

COMAC, China's largest passenger aircraft manufacturer, have also procured one of the systems for the control of metallic and composite parts at their plant in Shanghai.

Airbus, Embraer and Bombardier, are some of the main manufactures said to have purchased the RABBIT systems.

2.4.3.4. USL NDT system

Ultrasonic Sciences Ltd (USL) have produced a new system for robotic NDT [69] (Figure 2.12). The system comes with some interesting features to achieve fast scanning speeds.

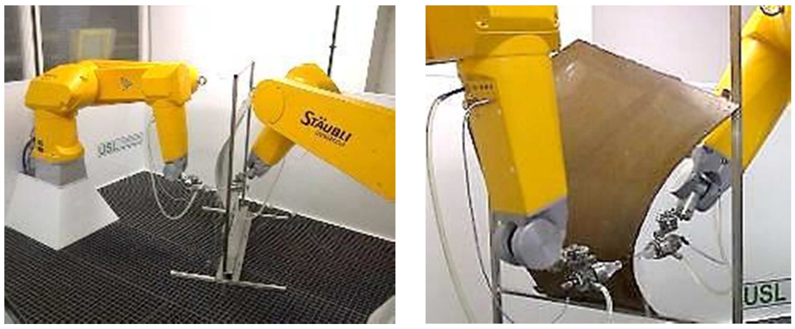


Figure 2.12 – Overview picture (a) and detail (b) of the new robotic USL system [69].

For ultrasonic imaging applications it is necessary to monitor the probe position with sufficient accuracy in real time, so that image pixels can be accurately placed on a C-scan image. This is necessary to ensure that the size and position of defects and features can be precisely recorded and measured. Current standard robot controllers can provide this position data at a maximum rate of 1 kHz. The main thrust of the USL development was to develop a robot motion controller which achieves high position feedback rate, so that scanning speeds can match those used in traditional scanning systems based on Cartesian mechanisms.

The system has been developed using two Stäubli 6 axis robots with the UNIval drive interface and the USL controller. The Stäubli robots have the necessary accuracy, rigidity and robustness for the application, with the added advantage of being inherently waterproof to IP65/67 and with the cabling being fed inside the arm. The combined system has sufficient capacity to control not only the 12 robot axes, but also the linear tracks which are required for long travel applications and the water pumps which supply the squirter heads. Control of the water flow is an important factor, to compensate for the position of the squirters as they follow the contour and for gravitational effects when operating at a high level. The robotic systems maintain the standard characteristics of existing USL ultrasonic systems, such as simultaneous through-transmission and pulse-echo testing (using logarithmic and linear amplifiers), multi gate data acquisition and full waveform capture.

The manufacturing time for robot based systems can be significantly reduced in comparison with bespoke manufacture and there are also benefits in terms of capital cost. In addition, the fact that well established, standard parts are used means that reliability, servicing and maintenance are improved. The first application of this new system was installed in the USA for a fuselage inspection, in conjunction with USL's partners in that project, Arcadia Aerospace. Other applications are expected in UK and elsewhere.

2.4.3.5. Genesis NSpect 210

NDT Solutions Inc. (NDTS) and Genesis Systems Group have collaborated to develop an automated Robotic Non-Destructive Inspection system [70]. The NSpect 210 is a single 6-axis robotic work-cell designed for automated NDT processes. The system combines ultrasonic inspection data with robot positional data and presents it as A-Scan, B-Scan and C-Scan images. Parts are robotically inspected using phased-array ultrasonic equipment configured both in through-transmission and pulse-echo modes.

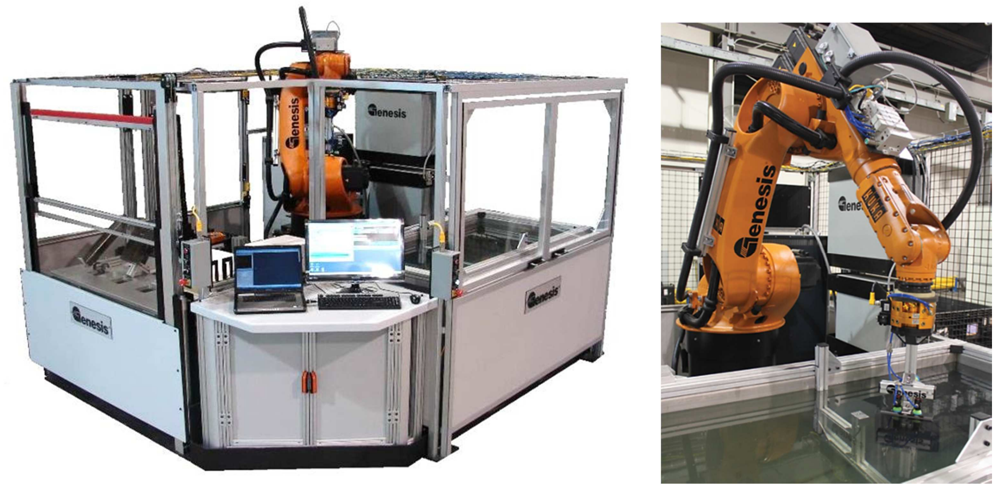


Figure 2.13 – NDTS & Genesis NSpect210 system.

2.5. Motivations and weaknesses of the current systems

A lack of flexible automated solutions for the inspection of large composite parts with complex geometry has been identified. Therefore the body of work contained in this thesis was required and motivated by the following main drives:

- The needs to automate the entire process of manufacturing aeronautical components, including the NDT inspection phases;
- Necessity to push the technology readiness level (TRL) of automated NDT inspection towards higher levels to meet the requirements of the aerospace industry (in terms of cost reduction, inspection speed and throughput);
- It was clear that new generations of robotic arms with enhanced positional accuracy had become available. NDT instruments with high speed acquisition rate had also been released into the market. The two fields of hardware were disconnected and great innovation possibility was foreseen by the idea of integrating the hardware and let the NDT acquisition field and the robotics field work together.

The recent efforts carried out by many companies in different parts of the world have changed the initial scenario. Most of the systems described above have become available in the last three years, concurrently to the development of this PhD research. It shows the strong and industrially driven interest arising worldwide for the achievement of better automated NDT inspection of composite structures and the high competitiveness of this research niche.

The author of this thesis kept an eye on the release of new systems, in order to periodically evaluate the novelty of the research against the outbreak of new solutions. Although very limited technical information is available for some of the aforementioned systems and a direct comparison is often not possible, some considerations can be drawn here and references to them will be found in the remaining part of the thesis.

The little information that is available about the Boeing AUSS suggests that the system uses a sliding gantry to move two 6-axis robots manipulating the NDT probes. It makes possible to inspect large parts, but the flexibility of the path-planning software and of NDT acquisition and data display software is unknown.

The USL system is designed for single-piezoelectric-element ultrasonic probes that require to move fast to increase the extension of surface inspected over time. Phased array probes cannot be moved as fast as single element probes for physical limitations; it is necessary to slow down the speed to allow the collection of all the A-Scans composing each B-Scan in a linear array, without distorting the B-Scan too much in the direction of motion (the passive axis of the transducer). Despite the fact ultrasonic phased array probes are moved at slower speed, the inspection of a given surface takes less time than the traditional single element inspection. This is due to the electronic sweeping (the progressive firing of piezoelectric elements in an array) being more time-effective than the mechanical movement required with single element probes.

The NSpect 210 system was clearly designed to achieve a high throughput for batch inspections of smaller components. The level of integration achieved is obviously very high. However changes to the ultrasonic data acquisition hardware suggest the system is still a development one; the original hardware from *Diagnostic Sonar Ltd* (DSL) has been recently replaced with an acquisition unit from *Advanced OEM Solutions* (AOS).

It is certain that Tecnatom systems achieve good performances. However, it is not clear the level of integration between robotic manipulators and the NDT equipment that has been achieved by Tecnatom. The capabilities of the acquisition software are not described in Tecnatom publications.

The flexibility of a bespoke robotic NDT system is mainly given by the driving software. First of all, the path-generation software is highly critical for the definition of an accurate and suitable toolpath targeted to the chosen inspection techniques. The acquisition software is in charge to exploit the flexibility of motion of industrial robots within their working envelope and the performances of the NDT probes to collect consistent data. The collected information must be accurately encoded through the positional data to provide meaningful output. Finally, the way the output is displayed and the freedom left to the user to customize the plots and gate different areas of interest can make one system more attractive than another. The efforts carried out within the present PhD project introduce novelty factors spanning beyond the features of the aforementioned systems.

References

[1] R. Bogue, "New NDT techniques for new materials and applications," *Assembly Automation,* vol. 32, pp. 211-215, 2012.

[2] D. K. Hsu, "Nondestructive inspection of composite structures: methods and practice," 2008.

[3] T. Yokozeki, Y. Aoki, and T. Ogasawara, "Experimental characterization of strength and damage resistance properties of thin-ply carbon fiber/toughened epoxy laminates," *Composite structures,* vol. 82, pp. 382-389, 2008.

[4] R. E. Green Jr, "Special Problems Associated with Ultrasonic Wave Propagation in Anisotropic Materials," *Nondestructive Testing And Evaluation,* vol. 8, pp. 469-476, 1992.

[5] E. S. Boltz and R. E. Green, "Ultrasonics in Thick Anisotropic Materials," in *Review of Progress in Quantitative Nondestructive Evaluation*, ed: Springer, 1993, pp. 1241-1248.

[6] P. Louviot, A. Tachattahte, and D. Garnier, "Robotised UT Transmission NDT of Composite Complex Shaped Parts " presented at the NDT in Aerospace, 2012.

[7] W. Hillger, L. Buhling, and D. Ilse, "Air-coupled Ultrasonic Testing-Method, System and practical Applications," in *11th European Conference on Non-destructive Testing*, 2014.

[8] B. B. Djordjevic, "Remote Non-Contact Ultrasonic Testing of Composite Materials," in *Proceedings of the 15th World Conference on Non-destructive Techniques, Rome, www. ndt. net*, 2000.

[9] M. Reilly, "Half a century of pioneering ultrasonics for Sonatest," *Insight,* vol. 50, p. 587, 2008.

[10] W. Hillger, R. Stößel, S. Lang, J. Schuller, R. Oster, L. Bühling, D. Ilse, J. Bosse, and B. Thaler, "Automated Air-Coupled Ultrasonic Technique for the Inspection of the EC145 Tail Boom," in *4th International Symposium on NDT in Aerospace, November13th to 15th*, 2012.

[11] T. Schmidt and S. Dutta, "Automation in production integrated NDT using thermography," *NDT in Aerospace, Augsburg,* 2012.

[12] C. Buynak and T. Moran, "Characterization of impact damage in composites," in *Review of Progress in Quantitative Nondestructive Evaluation*, ed: Springer, 1987, pp. 1203-1211.

[13] T. Moran and C. Buynak, "Correlation of Ultrasonic Imaging and Destructive Analyses of Low Energy Impact Events," in *Review of Progress in Quantitative Nondestructive Evaluation*, ed: Springer, 1989, pp. 1627-1634.

[14] D. K. Hsu, "Nondestructive evaluation of sandwich structures: a review of some inspection techniques," *Journal of Sandwich Structures and Materials,* vol. 11, pp. 275-291, 2009.

[15] E. Cuevas, M. López, and M. Garcìa, "Ultrasonic Techniques and Industrial Robots: Natural Evolution of Inspection Systems," presented at the 4th International Symposium on NDT in Aerospace, Ausburg, Germany, 2012.

[16] M. García, F. J. Fernández, and E. Cuevas, "Design of ultrasonic inspection techniques for complex geometries in composites," presented at the 12th NDT Spanish Congress.

[17] A. Kapadia, "Non Destructive Testing of Composite Materials," TWI Ltd, National Composite Network.

[18] U. Polimeno and M. Meo, "Detecting barely visible impact damage detection on aircraft composites structures," *Composite structures,* vol. 91, pp. 398-402, 2009.

[19] D. Kass. *Intro to Ultrasonic Phased Array*. Available: http://www.olympus-ims.com/en/ultrasonics/intro-to-pa/ - Accessed 16/11/2014

[20] C. Holmes, B. W. Drinkwater, and P. D. Wilcox, "Post-processing of the full matrix of ultrasonic transmit–receive array data for non-destructive evaluation," *NDT & E International,* vol. 38, pp. 701-711, 2005.

[21] C. Beine, C. Boller, U. Netzelmann, F. Porsch, S. Ramanan, M. Schulze, A. Bulavinov, and H. Heuer, "NDT for CFRP aeronautical components a comparative study," *Proceedings of NDT in Aerospace, November,* pp. 22-24, 2010.

[22] J. P. Charlesworth and J. A. G. Temple, *Engineering applications of ultrasonic time-of-flight diffraction*: Research Studies PressLtd, 2001.

[23] M. Silk and B. Lidington, "The potential of scattered or diffracted ultrasound in the determination of crack depth," *Non-destructive Testing,* vol. 8, pp. 146-151, 1975.

[24] T. Kundu, *Ultrasonic nondestructive evaluation: engineering and biological material characterization*: CRC press, 2004.

[25] J. Oosthuizen, C. Stanton, and Z. McCann. *The application of the time-of-flight diffraction inspection technique on GRP structures*. Available: http://www.netcomposites.com/ikb/Topics/-Examples/TO%20Example%20Inspection%20of%20GRP%20vessels%20with%20TOFD/grp_paper.pdf - Accessed 13/05/2015

[26] W. Hillger, "Ultrasonic Testing of Composites-From laboratory Research to In-field Inspections," in *15th world conference on Nondestructive Testing, Rome (Italy)*, 2000, pp. 15-21.

[27] M. Castaings and P. Cawley, "The generation, propagation, and detection of Lamb waves in plates using air-coupled ultrasonic transducers," *The Journal of the Acoustical Society of America,* vol. 100, pp. 3070-3077, 1996.

[28] D. J. Roth, A. Mandlik, J. Sandhu, and L. Hertert, "Approaches for non-uniformity correction and dynamic range extension for acoustography," in *Nondestructive Evaulation for Health Monitoring and Diagnostics*, 2005, pp. 124-134.

[29] F. J. Fahy and P. Gardonio, *Sound and structural vibration: radiation, transmission and response*: Academic press, 2007.

[30] P. Hariharan, *Optical Holography: Principles, techniques and applications* vol. 20: Cambridge University Press, 1996.

[31] R. Collier, *Optical holography*: Elsevier, 2013.

[32] P. Cawley and R. Adams, "The mechanics of the coin-tap method of non-destructive testing," *Journal of Sound and Vibration,* vol. 122, pp. 299-316, 1988.

[33] H. Wu and M. Siegel, "Correlation of accelerometer and microphone data in the “coin tap test”," *Instrumentation and Measurement, IEEE Transactions on,* vol. 49, pp. 493-497, 2000.

[34] D. K. Hsu, D. J. Barnard, and J. J. Peters, "Nondestructive evaluation of repairs on aircraft composite structures," in *6th Annual International Symposium on NDE for Health Monitoring and Diagnostics*, 2001, pp. 100-107.

[35] P. Cawley, "The sensitivity of the mechanical impedance method of nondestructive testing," *NDT international,* vol. 20, pp. 209-215, 1987.

[36] P. Cawley, "The impedance method of non-destructive inspection," *NDT international,* vol. 17, pp. 59-65, 1984.

[37] P. Cawley and C. Theodorakopoulos, "The membrane resonance method of non-destructive testing," *Journal of Sound and Vibration,* vol. 130, pp. 299-311, 1989.

[38] P. Cawley, "The sensitivity of an NDT instrument based on the membrane resonance method," *NDT international,* vol. 22, pp. 209-216, 1989.

[39] R. Halmshaw, R. Honeycombe, and P. Hancock, *Non-destructive testing*: E. Arnold, 1991.

[40] I. Scott and C. Scala, "A review of non-destructive testing of composite materials," *NDT international,* vol. 15, pp. 75-86, 1982.

[41] E. Birt, "Applicability of X-radiography to the inspection of composites," *NDT'99 and UK Corrosion'99,* pp. 449-454, 1999.

[42] M. Strobl, I. Manke, N. Kardjilov, A. Hilger, M. Dawson, and J. Banhart, "Advances in neutron radiography and tomography," *Journal of Physics D: Applied Physics,* vol. 42, p. 243001, 2009.

[43] P. Rinard, "Neutron interactions with matter," *Passive Nondestructive Assay of Nuclear Materials,* pp. 357-377, 1991.

[44] K. Davies, "Industrial radiography- managing radiation risks," *Insight: Non-Destructive Testing and Condition Monitoring,* vol. 43, pp. 46-48, 2001.

[45] HSE, "Information for the procurement and conduct of NDT - Part 3: Radiographic Inspection in Industry," ed, 2008.

[46] P. Reimers, W. Gilboy, and J. Goebbels, "Recent developments in the industrial application of computerized tomography with ionizing radiation," *NDT international,* vol. 17, pp. 197-207, 1984.

[47] J.-P. Kruth, M. Bartscher, S. Carmignato, R. Schmitt, L. De Chiffre, and A. Weckenmann, "Computed tomography for dimensional metrology," *CIRP Annals-Manufacturing Technology,* vol. 60, pp. 821-842, 2011.

[48] Y. Hung, "Shearography: a new optical method for strain measurement and nondestructive testing," *Optical Engineering,* vol. 21, pp. 213391-213391-, 1982.

[49] Y. Hung and H. Ho, "Shearography: An optical measurement technique and applications," *Materials science and engineering: R: Reports,* vol. 49, pp. 61-87, 2005.

[50] W. Reynolds, "Thermographic methods applied to industrial materials," *Canadian Journal of Physics,* vol. 64, pp. 1150-1154, 1986.

[51] C. Filloy, W. Liu, and B. Yang, "Thermography techniques for integrated circuits and semiconductor devices," *Sensor Review,* vol. 27, pp. 298-309, 2007.

[52] J. Salazar, A. Turo, J. A. Chávez, J. A. Ortega, and M. J. Garcia, "High-power high-resolution pulser for air-coupled ultrasonic NDE applications," *Instrumentation and Measurement, IEEE Transactions on,* vol. 52, pp. 1792-1798, 2003.

[53] J. Waanders, *Piezoelectric ceramics: properties and applications*: NV Philips' Gloeilampenfabrieken, 1991.

[54] C. B. Scruby and L. E. Drain, *Laser ultrasonics techniques and applications*: CRC Press, 1990.

[55] J.-P. Monchalin, C. Neron, P. Bouchard, and R. Heon, "Laser-ultrasonics for inspection and characterization of aeronautic materials," *Journal of Nondestructive Testing & Ultrasonics(Germany),* vol. 3, p. 002, 1998.

[56] D. Cerniglia, K. Jhang, and B. Djordjevic, "Non-contact ultrasonic testing of aircraft lap joints," in *15th Worl Conference on NDT,, Rome, Italy, Editor AIPnD, NDT. net October*, 2000.

[57] A. Forbes, L. Botha, N. du Preez, and T. Drake, "Influence of laser parameters on laser ultrasonic efficiency," in *XVI International Symposium on Gas Flow, Chemical Lasers, and High-Power Lasers*, 2006, pp. 63462X-63462X-7.

[58] X. E. Gros, "An eddy current approach to the detection of damage caused by low-energy impacts on carbon fibre reinforced materials," *Materials & Design,* vol. 16, pp. 167-173, 1995.

[59] C. Bonavolontà, G. Peluso, G. Pepe, and M. Valentino, "Detection of early stage damage in carbon fiber reinforced polymers for aeronautical applications using an HTS SQUID magnetometer," *The European Physical Journal B-Condensed Matter and Complex Systems,* vol. 42, pp. 491-496, 2004.

[60] C. Bonavolonta, M. Valentino, G. Peluso, and A. Barone, "Non destructive evaluation of advanced composite materials for aerospace application using HTS SQUIDs," *Applied Superconductivity, IEEE Transactions on,* vol. 17, pp. 772-775, 2007.

[61] M. Valentino, C. Bonavolonta, G. Peluso, and G. Pepe, "Real-time monitoring of fatigue damage in Carbon Fiber Reinforced Polymers for aeronautical applications using HTS SQUID magnetometer," in *Journal of Physics: Conference Series*, 2006, p. 1231.

[62] M. Vandewal, J. Depauw, K. Rombaut, R. Beigang, J. Jonuscheit, P. Mounaix, I. S. de Ocáriz, R. M. Edo, A. Priegue, and Y. Sternberg, "Development and optimization of THz NDT on aeronautics composite multilayered structures," in *SPIE Defense, Security, and Sensing*, 2012, pp. 83630Y-83630Y-9.

[63] M. Schwabe, A. Maurer, and R. Kock, "Ultrasonic Testing Machines with Robot Mechanics – A New Approach to CFRP Component Testing," presented at the NDT in Aerospace 2010, 2010.

[64] M. P. Cartmell, A. Whittaker, and A. Krimly, "The kinematics and dynamics of a dextrous parallel manipulator for structural inspection applications," *Key Engineering Materials,* vol. 293, pp. 653-660, 2005.

[65] *Automated Ultrasonic Scanning Systems (AUSS)*. Available: http://www.boeing.com/boeing/defense-space/support/maintenance/commercial/auss.page

[66] I. Standard, "8373: 1994," *Manipulating Industrial Robots–Vocabulary*.

[67] G. E. Georgeson, "Recent advances in aerospace composite NDE," in *NDE For Health Monitoring and Diagnostics*, 2002, pp. 104-115.

[68] Tecnatom, "Tecnatom passes the acceptance tests for two new aeronautical systems," *NDT.net e-Journal,* vol. 19, 2013.

[69] USL, "A new NDT development using high speed robots," *NDT.net e-Journal,* vol. 19, 2013.

[70] *Robotic Non Destructive Inspection NSpect 210*. Available: www.genesissystems.com - Accessed 10/12/2012

3. Interfacing challenges for automating NDT with 6-axis manipulators

3.1. Introduction

This chapter focuses on the robotics challenges to be addressed for the deployment of robotic arms as manipulators of NDT probes. The features of 6-axis robotic arms are explained. Some early NDT systems, which provided a starting point to the research, are described. Moreover, the chapter presents different robot control strategies, the interfacing challenges for automating NDT inspections and the encoding of the acquired data. Experimental results are described to compare the performances of the different robot control strategies and investigate the accuracy of the executed robot tool-paths. For the sake of helping the comprehension of the motivations of the experiments reported in this chapter, they are briefly described in Table 3.1.

Name of experiment		Position	Motivation	Results
1	Scan of a flat metal block with a discrete point robot system	Section 3.4.1.1.	Observing the performances of the system described in Section 3.4.1.	The system is able to coordinate robot motion and UT data collection to perform raster scans; Scan speed too low; Communications not suitable to achieve a good level of automation.
2	Scan of a flat metal block with a continuous motion robot system	Section 3.4.2.1.	Observing the performances of the system described in Section 3.4.2.	Scanning time reduced by a factor of 12 with no loss of resolution; Data acquisition rata still lower than the one of commercial systems.

3	Scan of a flat metal block with a externally controlled robot system	Section 3.5.1.	Observing the performances of the system described in Section 3.5.	New robot control approach successfully tested.
4	Comparison of target and feedback positions of externally controlled KRC2 robot	Section 3.6.1.	Accuracy investigation	RMS of positional error always smaller than 1mm. RMS of orientation error lower than 0.36 degrees.
5	Laser-tracking accuracy measurements of externally controlled KRC2 robot	Section 3.6.2.	Independent positional accuracy measurement.	The manual calibration of the robot tool central point is the main source of error.
6	Comparison of target and feedback position of externally controlled KRC4 robot	Section 3.7.2.	Testing the external control approach on KRC4 robot	The implementation of the external control on KRC4 robots is successful; The values of the error show that the KR16 L6-2 robot is more accurate than the KR5arcHW robot.
7	Test of the approach to encode NDT data with robot feedback positions	Section 3.8.	Validation of the approach and of its robustness	The software approach described in Figure 3.34 proves to be reliable and robust.

Table 3.1 – List of experiments described in this chapter.

3.2. Describing 6-axis robot arms

The extension of the use of 6-axis robots for product inspection and NDT is central to the investigations in this thesis. This section contains a basic breakdown of how a 6-axis robot works, the way it moves, how it is programmed, the interfacing challenges for automating NDT and all related topics necessary to this thesis.

Six-axis industrial robots are able to perform a wide range of applications with great flexibility. In their simplest terms they consist of two units; the robotic arm and the control system. They are called six-axis robots because the arm has 6 axes, arranged as a serial link, which allow movement to any point within a working envelope (Figure 3.1a).

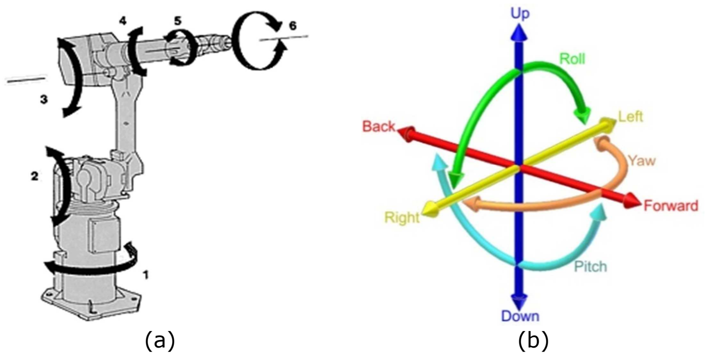


Figure 3.1 – a) 6-axis manipulator; b) degrees of freedom (DoF) (wikipedia.org).

Six-axis robot arms are also called 6-DoF arms, since they enable 6-degrees of freedom of movement of a rigid body (e.g. the robot tool) in the three-dimensional space (Figure 3.1b). Moreover, 6-axis robots are known as serial-link manipulators, as opposed to the parallel-link manipulators briefly described in Section 2.4.2.

The following list specifies the functions of each axis.

- Axis 1 – It is located at the base of a robot and allows rotation from left to right.
- Axis 2 – It helps the lower arm of a robot to extend forward and backward.
- Axis 3 – It allows the upper arm of a robot to raise and lower.
- Axis 4 – This axis is known as wrist roll, and it rotates the upper arm of a robot in a circular movement.
- Axis 5 – It permits the wrist of the robot's arm to raise and lower.
- Axis 6 – It allows the wrist of the robot's arm to rotate freely in a circular motion.

For industrial robots, their movement is driven by servo motors. The control system is a computer that allows precise movement of the arm by controlling the power supplied to the motors. A 7th axis can also be incorporated if needed. When working on complex parts, or with complex processes, the reach provided by a 6-axis robot may be insufficient. For these situations an additional positioner (rotating table, track, etc.) can be introduced and fully coordinated with the movement of the robots. Manufacturers of commercially available industrial robots include:

- Intelligent Actuator
- Adept
- Epson Robots
- Yaskawa-Motoman
- ABB
- EPSON-SEIKO
- igm Robotersysteme
- KUKA
- Kawasaki
- FANUC Robotics

Typical applications where industrial robots have already been integrated include welding, painting, ironing, assembly, pick and place and palletizing; all accomplished with high endurance, speed, and precision.

The extension of the use of six-axis robots for product inspection and non-destructive testing systems is the central investigation of this thesis. In this work KUKA robots have been used, for their combination of costs, performances and high data transfer rate.

3.2.1. Kinematics

The robot arms are designed to manipulate an end-effector with six degrees of freedom, consisting of three in translation and three in orientation. There is a direct relationship between actuator positions and the configuration of the manipulator defined by its forward and inverse kinematics.

3.2.1.1. Forward kinematics

Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters (joint coordinates) [1]. The kinematics equations for the series chain of a robot are obtained using a rigid transformation matrix (Z_j) to characterize the relative movement allowed at the j-th joint and another transformation matrix (X_j) to define the dimensions of the j-th link. The result is a sequence of rigid transformations alternating joint and link transformations from the base of the chain to its end link, which is equated to the specified position for the end link:

$$T = Z_1 \cdot X_1 \cdot Z_2 \cdot X_2 \cdot \ldots \cdot X_{n-1} \cdot Z_n \cdot X_n \qquad \text{Eq. 3.1}$$

where T is the transformation locating the end-link. These equations are called the kinematics equations of the serial chain [2]. In 1955, *Jacques Denavit* and *Richard Hartenberg* [3, 4] introduced a convention for the definition of the joint matrices Z and link matrices X to standardize the coordinate frame for spatial linkages. This convention positions the joint frame so that it consists of a screw displacement along the Z-axis:

$$Z_j = Trans_{Z_j}(d_j) \cdot Rot_{Z_j}(\theta_j) \qquad \text{Eq. 3.2}$$

and it positions the link frame so it consists of a screw displacement along the X-axis:

$$X_j = Trans_{X_j}(a_j) \cdot Rot_{X_j}(\alpha_j). \qquad \text{Eq. 3.3}$$

Using this notation, each transformation-link goes along a serial chain robot and can be described by the coordinate transformation:

$$T_j^{j-1} = Z_j \cdot X_j = Trans_{Z_j}(d_j) \cdot Rot_{Z_j}(\theta_j) \cdot Trans_{X_j}(a_j) \cdot Rot_{X_j}(\alpha_j) \qquad \text{Eq. 3.4}$$

where θ_j, d_j, a_j and α_j, shown in Figure 3.2, are known as the Denavit-Hartenberg parameters.

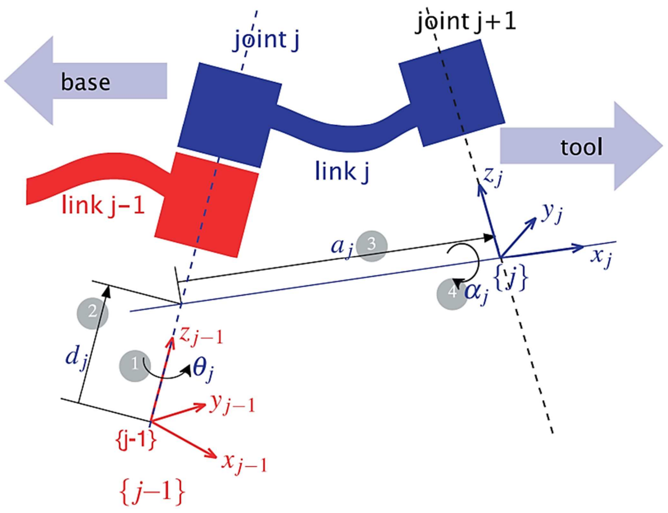


Figure 3.2 – Definition of standard Denavit and Hartenberg link parameters [5].

The kinematics equations of a serial chain of n links, with joint parameters θ_j are given by:

$$T = T_n^0 = \prod_{j=1}^{n} T_j^{j-1}(\theta_j) \qquad \text{Eq. 3.5}$$

where $T_j^{j-1}(\theta_j)$ is the transformation matrix (known as the *Denavit-Hartenberg matrix*) from the frame of link *j* to link *j-1*. The matrices associated with these operations are:

$$Trans_{Z_j}(d_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \quad Rot_{Z_j}(\theta_j) = \begin{bmatrix} \cos\theta_j & -\sin\theta_j & 0 & 0 \\ \sin\theta_j & \cos\theta_j & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ;$$
$$Trans_{X_j}(a_j) = \begin{bmatrix} 1 & 0 & 0 & a_j \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; \quad Rot_{X_j}(\alpha_j) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_j & -\sin\alpha_j & 0 \\ 0 & \sin\alpha_j & \cos\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad \text{Eq. 3.6}$$

The use of the Denavit-Hartenberg convention yields the link transformation matrix:

$$T_j^{j-1}(\theta_j) = \begin{bmatrix} \cos\theta_j & -\sin\theta_j \cdot \cos\alpha_j & \sin\theta_j \cdot \sin\alpha_j & a_j \cdot \cos\theta_j \\ \sin\theta_j & \cos\theta_j \cdot \cos\alpha_j & -\cos\theta_j \cdot \sin\alpha_j & a_j \cdot \sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad \text{Eq. 3.7}$$

3.2.1.2. Inverse kinematics

Whereas forward kinematics uses the joint parameters to compute the configuration of the chain; inverse kinematics reverses this calculation to determine the joint parameters that achieves a desired configuration of the end-effector [1].

The use of inverse kinematics formulas has become commonplace in the modern computer animation industry. The formulas allow calculation of the positions of the skeleton of an animated character that is to move in a particular way.

For industrial robots, the inverse kinematics models are used by off-line path-planning software for the simulation of the movements of the robot during the execution of a particular task. The movement of a kinematic chain whether it is a robot or an animated character is modelled by the kinematics equations of the chain. These equations define the configuration of the chain in terms of its joint parameters.

The kinematic analysis of a mechanical chain is the first step in the design of most industrial robots. Kinematic analysis allows the designer to obtain information on the position of each component within the mechanical system. This information is necessary for subsequent dynamic analysis along with path control. Analytical solutions to the inverse kinematics problem exist for a wide range of kinematic chains, including serial-link 6-DoF manipulators.

The development of an inverse kinematics model was one of the challenges overcome by this work. The motivations behind the creation of the kinematics model and the mathematical details will be described in Chapter 4.

3.2.2. Singularities

The singularities are kinematics problems that emerge when robot axes are redundant (more axes than necessary) or when the robot is positioned in certain configurations that require extremely high joint rates to move at some nominal speed in Cartesian space. Industrial robots have 6 or less joints, thus, redundancy is not inherent to their design. However, for six-axis robots, there are certain joint arrangements that can lead to redundancy for certain orientations. Mathematically, a *jacobian* mathematical matrix formulation can be used to relate the motion in joint space to the motion in Cartesian space [1, 3, 4]. The singularity occurs when the inverse *jacobian* becomes singular (matrix determinant equal to zero) [2].

The types of singularity configurations that can be experienced by a robot depend on the physical relationships between the robot joints. The most common types of singularities that can occur are bulleted here [2]:

- Alignment (Shoulder or Overhead) Singularity occurs when Joint 6 (wrist) and Joint 1 axes are aligned (Figure 3.1a).

- Elbow Singularity occurs when the arm is fully extended. In this case, as the elbow joint becomes further extended, higher joint speeds are required to maintain constant Cartesian speed (Figure 3.1b). The robot cannot extend beyond its reach.
- Wrist Singularity occurs when the axes of joints 4 and 6 are aligned (Figure 3.1c).

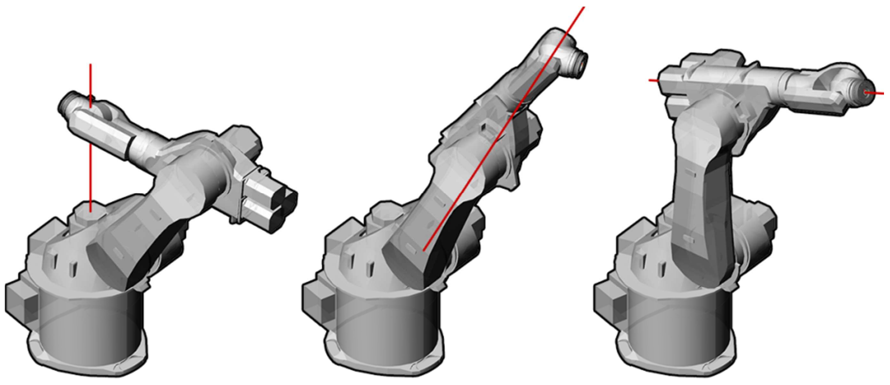


Figure 3.3 – Types of singularity: overhead (a), elbow (b) and wrist (c).

It is not easy to always avoid singularities when the robot is programmed manually. When a 6-DoF robot suddenly jerks to a halt because the controller has anticipated motion through a singularity, the operator is often left mystified, despite the clarity of the error message that a singular configuration has been reached. A 6-DoF robot can approach a point in the 3D space using 8 different configurations of its joints [2]. An experienced programmer should be able to select the most convenient configuration to avoid singularity problems.

Good off-line path-planning software is usually used to help the robot programmer to avoid the singularity configurations. A viable way to avoid robot singularities is also to control the robot movements with axis coordinates rather than Cartesian coordinates; this can ensure the robot behaviour remains singularity-free during the entire task.

3.2.3. Working envelope

The kinematic model must take into account the pose of the robot to plan a movement which is physically possible. For most 6-DoF manipulators, each of the joints (which are typically all of different lengths) is driven by an electrical motor. Each joint has one degree of freedom (in this case only rotational) with two limit positions (maximum and minimum rotation).

The space which contains all the reachable points of the robot is known as the robot's working envelope.

3.2.4. Dynamics

The robot controller is responsible to execute the command lines contained in the robot program, developed manually by the programmer or by path-planning software. The actual act of the robot controller is to use the kinematic model to drive the motors and move the end effector. Once the robot controller knows its current position, given a new position in space, it calculates how to get there. This takes place by application of power to the electric motors driving its gears, resulting in the rotation of one or more joints by certain numbers of degrees. Due to the difference in the position of the motors and lengths of the segments that make up the robot, each motor will usually drive at a different speed to execute a desired motion.

The robot controller task of moving a robot arm accurately is not trivial and needs to encompass a large number of variables. Acceleration and deceleration curves are applied to the robot trajectory to minimise wear on the motors as well as the dynamic effects arising from inertial loads given by the masses of the joints and of the tool attached to the end effector.

The inertial forces have an influence on the dynamic accuracy of the robot arm. Despite the fact the accelerations values of robot tool-paths for NDT applications are usually restrained to low values, the dynamic issues related to the robot movements were considered in several phases of this work.

3.2.5. Accuracy and repeatability

The catalogues of robot manufacturers specify the repeatability of each robot arm, among numerous other parameters. The accuracy is not reported, because it depends on the calibration of the robot tool central point operated by the robot user. Accuracy and repeatability are different measures. Repeatability is usually the most important criterion for a robot. The ISO 9283 standard sets out a method whereby both accuracy and repeatability can be measured [6]. Typically a robot is sent to a commanded position a number of times and the error is measured at each return to the position after visiting 4 other positions. Repeatability is then quantified using the standard deviation of those position samples in all three dimensions.

Accuracy defines how close a robot can reach a commanded position [7]. When the absolute position of the robot is measured and compared to the commanded position the error is a measure of accuracy. The accuracy can vary with speed and position within the working envelope and with payload.

Repeatability is how well the robot will return to a programmed position [7]. This is not the same as accuracy. It may be that when told to go to a certain X-Y-Z position the robot gets only to within 1 mm of that position. This would be its accuracy which may be improved by calibration. But if that position is programmed into the robot controller memory and each time it is sent there it returns to within 0.1mm of the taught position then the repeatability will be within 0.1mm.

Some robot manufacturers report the resolution (or precision) of their robots. The resolution of a robot is a feature determined by the design of the control unit and is mainly dependent on the position feedback sensor. It is important to distinguish the programming resolution from the control resolution. The programming resolution is the smallest allowable position increment in robot programs. The control resolution is the smallest change in position that the feedback device can sense. The best performances are obtained when programming resolution is equal to control resolution. In this case both resolutions can be replaced with one term: the system resolution or precision.

The definition of the Denavit-Hartenberg parameters becomes essential in ensuring that the robot is able to move accurately and precisely to a specified point in space. Small deviations in manufacturing can lead to the robot moving inaccurately. This is why robots are calibrated upon delivery to ensure that the accuracy of the robot matches that stated by the manufacturer. Static and dynamic calibrations are necessary. Static calibration identifies static parameters such as link lengths, coupling factors and deviations in axis orientations. In addition, the load of the end effector can significantly change according to the applications and this can also affect accuracy and repeatability. Dynamic calibration identifies parameters such as friction and loading forces [8].

The calibration procedure involves moving the robot to a number of positions and using an external metrology system capable of 3-DoF or 6-DoF measurements to record these movements. Coordinates of the robot controller are compared to those obtained using an external metrology system for a large dataset of points (typically over 100). The difference in position between the kinematic model prediction and the observed points are then minimised by updating the kinematic model parameters through a non-linear least squared method [9]. The Denavit-Hartenberg parameters of the robot can be periodically updated to ensure accurate motion throughout the entire life of the robot.

According to Conrad et al [10], a length deviation of ± 0.1mm and angle uncertainty of 0.000375° can lead to an end-effector inaccuracy of approximately ± 0.3mm. This is only due to machining tolerances and does not include weight loading or temperature effects. The deviations in segment length are one of the main factors influencing robot accuracy. Further additions to the overall error include friction, inertia, gear backlash, lack of rigidity and the fundamental limit of how accurately each encoder can describe a movement. It is also worth noting that different poses (orientations) put slightly different strains on each joint motor, so the inaccuracy varies within the working envelope. Typical inaccuracies of modern six-axis industrial robots depend on the size of the robot (due to the larger loads it can carry and manufacturing tolerances) but with better off-line planning, compensation models and calibration procedures, even large robots can attain a positional accuracy of approximately ± 0.5mm [11].

3.2.6. Communications

KUKA robots have been used in the delivery of the work in this thesis. Like most modern robots, KUKA robots are able to output their current position with update rates up to 1 kHz [12]. The current pose is usually output through the Ethernet port of the robot controller. The communication can be based on two network protocols: User Datagram Protocol (UDP) or Transmission Control Protocol / Internet Protocol (TCP/IP) communication. These two protocols are similar but have some important differences [13].

UDP (*User Datagram Protocol*) is a “connectionless” protocol, meaning that data can be transmitted without confirmation that there is a server ready to receive them. In addition, different packets of data may be routed differently through the network and may arrive in a different order to which they were sent – similar to sending more than one letter in the post.

TCP/IP (*Transmission Control Protocol / Internet Protocol*) is known as being a “connection-orientated” protocol which means that a connection must be established between client and server before and then data can be sent. This is a reliable protocol, where data packets can be guaranteed to be delivered and are re-sent if not delivered correctly. In addition, it is structured meaning that data packets will be routed along the same network path and will arrive in order.

Therefore, when available, TCP/IP is selected as the preferred option due to the increased robustness of the communication.

For KUKA robots, communication is supported by a software add-on known as the *Robot Sensor Interface*, or RSI [14, 15]. RSI was purposely developed by KUKA to enable the transmission of data from the robot. Using RSI also makes it possible to influence the robot motion or program execution via sensor data. The sensor data and signals can be read by a field bus on the robot controller. This data is processed in the RSI context and forwarded to the robot controller. Optionally, the data can also be exchanged via Ethernet as XML strings.

3.2.7. Robot programming

Each robot manufacturer adopts a specific machine programming language that its robot controllers are able to interpret.

KUKA robots are programmed through the KUKA Robot Language (KRL), which is a proprietary programming language with similarities to Pascal [16]. Unlike G-Code[4], it does not contain just tool and machine movement commands, but can also declare variables and work with conditional clauses [17]. Basically, a control file for KUKA consists of two separate files: a *.dat file where variables such as the home position are stored and a *.src file which contains the various commands, like movement commands, loops, sensor queries, etc.

There are various ways of giving movement commands to a robot, the most common ones are the following [18]:

- Joint coordinate programming consists of absolute axis rotation commands, instructing the robot to move each of its six axes to a defined rotation value; e.g. (A1: 0, A2: 10, A3: 90, A4: 20, A5: 60, A6: 25).
- Cartesian coordinate programming defines the location and orientation of the end-effector in a previously defined Cartesian coordinate system (*world* or a

[4] The G-Code is the common name for the most widely used numerical control programming language. It is used mainly in computer-aided manufacturing for controlling automated machine tools. The first implementation of a numerical control programming language was developed at the MIT Servomechanisms Laboratory in the late 1950s.

base reference system). This can either be point-to-point (PTP) commands where the end-effector moves from one position to the next with the least amount of axis rotation or linear (LIN) commands where the end-effector moves from one position to the next along a straight line; e.g. (X: 10, Y: 20, Z: 40, A: 45, B: 75, C: 15).

In order to enable 6-axis robot arms to follow the contour of complex surfaces, it is necessary to generate the optimum robot scanning paths. To date, there are two main approaches to the path-planning problem:

1. The simplest programming method for industrial robots is to use a teach pendant. It is a handheld control and programming unit; a human-machine-interface (HMI) device (typically touch-screen). It is used to program in a matrix of points and to interpolate between these points using predefined movement types. The points can be included in a program by inserting their coordinates or by teaching their position to the robot. The teaching approach is based on jogging the robot arm to all points of interest. The programmer has to know the specific robot language to manually develop the program by inserting the necessary command lines. Therefore, in addition to being very time-consuming and not suited to complex geometries, this approach requires specialized technicians. The obtained tool-paths are usually not very precise.
2. Path-planning is better achieved through off-line programming (OLP) software. "Off-line" means that the programming takes place without contact with the robot. This approach reduces the set-up time and the robots can remain operative during the path-planning of new tasks. Several commercial OLP applications exist; examples of programs are: ABB Robot Studio, Delcam, Delmia, FastSurf, RobCAD, Robot3D, MasterCAM-RobotMaster, KUKA|prc. The OLP software is usually able to load the component geometry CAD file into the virtual environment of the application, and it allows the operator to select the surfaces of interest. Therefore the software generates the tool-path for the robots. This kind of OLP software provides the ability to "build" a virtual robot cell, complete with components and manipulators, in a three dimensional virtual environment. Many kinematics models of the robots currently in use are available and these can be incorporated into the virtual environment. Many

OLP programs also provide the possibility to simulate several different robots at the same time and establish any potential robot collisions safely in a virtual environment.

Theoretically, there should be another viable way of executing a robot tool-path. It would involve the capability of a robot controller to exchange information with an external computer ("external" to the robot controller). A simple robot program executing in the robot controller could be responsible to manage the connection and receive sequential packets of coordinates from the external computer. These coordinates would command the robot controller the next point to reach during the scanning motion. This approach differs from the traditional OLP, where a specific robot language programme is compiled by the OLP software and transferred into the memory of the robot controller before the execution.

3.2.8. Synchronization of cooperative robots

Multiple robot systems engaged in cooperative tasks are becoming widespread in the manufacturing industry. Such systems can be of interest for several reasons:

- Tasks may be inherently too complex for a single robot to be accomplished (or impossible to be realized);
- Performance benefits can be gained from using multiple robots;
- Building and using several simple robots can be easier, cheaper, more flexible and more fault tolerant than having a single powerful robot for each separate task.

Studies concerning multiple-robot systems naturally extend the research developed on single-robot systems and represent also a discipline by themselves. Multiple-robot systems can accomplish tasks that no single robot can accomplish, since ultimately a single robot (no matter how capable) is spatially limited [19]. Modelling and controlling multiple robotic manipulators handling a constrained object requires more sophisticated techniques compared with a single robot working alone. Many benefits can be obtained by using them in industrial manufacturing. A typical example is in a flexible assembly, where the robots assemble two parts into a product [20].

However, there is minimal literature regarding experimental results on modelling, simulation and control of systems of multiple manipulators. The problem of controlling multiple robots holding a common object is often solved with force-control techniques [21].

In the field of robotic non-destructive testing, the handling of rigid objects is not the main concern. The use of multiple robot systems can be of interest for speeding up the inspection of large and complex components. More importantly, cooperating robots can also become necessary to execute particular NDT techniques (e.g. the ultrasonic through-transmission inspection).

Force-control techniques are not suitable to model and control cooperating robots for most of the NDT applications, since the robots manipulate independent probes rather than a common object. Each 6-axis robot arm has its own control unit; a basic degree of synchronization can be achieved through using triggering signals between the digital I/O panels of the controllers. It only allows the synchronization of key positions of the trajectories of multiple robots sharing a working space. This is good enough for making sure the robots start or finish specific tasks at precise times.

However, the ultrasonic through-transmission inspection technique requires two probes (one generator and one receiver) to be constantly aligned during the scan of a given surface. A good level of alignment cannot be obtained through using the previous approach. The robots can easily lose the synchronization and the alignment during the execution of a task, even if they start it at the same time. This can be caused by small differences in the speeds and/or in the length of the curved trajectories.

More robust robot synchronization approaches are based on Ethernet communication [22]. Some software solutions of this kind have been developed by robot manufacturers for their robots. For example, *RoboTeam* is a software add-on for KUKA robots that enables precisely coordinated teamwork between up to 15 robots through fast synchronization of the path motions. Each robot keeps its standard controller. This is connected to a high-speed local network (Ethernet) via which the controllers communicate with one another and synchronize themselves [23].

3.3. Robotic hardware used in this work

As mentioned in Chapter 1, the work carried out for the PhD project has taken place at University of Strathclyde (UoS) and TWI.

3.3.1. Robots at the University of Strathclyde

A robotic setup was already in existence at the University of Strathclyde at the starting date of the PhD project that led to the present thesis. The University of Strathclyde established a robotic cell in 2010 to support the research and a wide project undertaken by staff and students in the Centre for Ultrasonic Engineering (CUE), part-funded by industrial partners. The overall aim was to create a robotic NDT system capable of scanning a wide variety of complex geometries. The robot cell at the University of Strathclyde, used for carrying out part of the research leading to this thesis, consists of two industrial robotic arms. They are two KR5arcHW robots [24], manufactured by KUKA. An image of one of the two twin robots is shown in Figure 3.4.

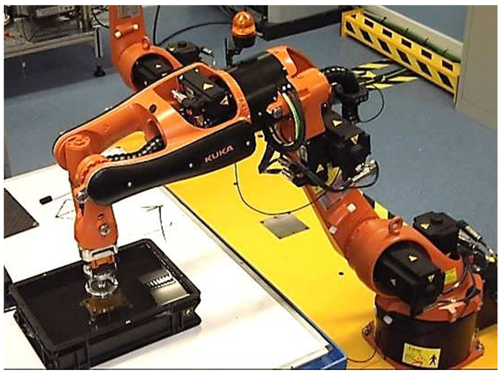


Figure 3.4 – KUKA KR5arcHW Robotic Arm.

3.3.2. Robots at TWI Technology Centre

A new robotic facility was established at TWI Validation Centre in January 2013. The system comprises two KR16 L6-2 six-axis robots [25], each one mounted on an 800 mm pedestal to maximise the robot working envelope. It is enclosed by safety guarding to form a robotic cell. Each robot is powered and controlled from its own dedicated controller. The two controllers can execute independent tasks or communicate over an EtherCat[5] link when running in co-operating mode. The robots are monitored and managed using a single KUKA SmartPad (teach pendant), which is able to connect to each controller individually. Figure 3.5 shows two pictures of the TWI robot cell as it was soon after its establishment.

[5] EtherCAT (Ethernet for Control Automation Technology) is an Ethernet-based fieldbus system, invented by Beckhoff Automation. The protocol is standardized in IEC 61158 and is suitable for both hard and soft real-time requirements in automation technology.

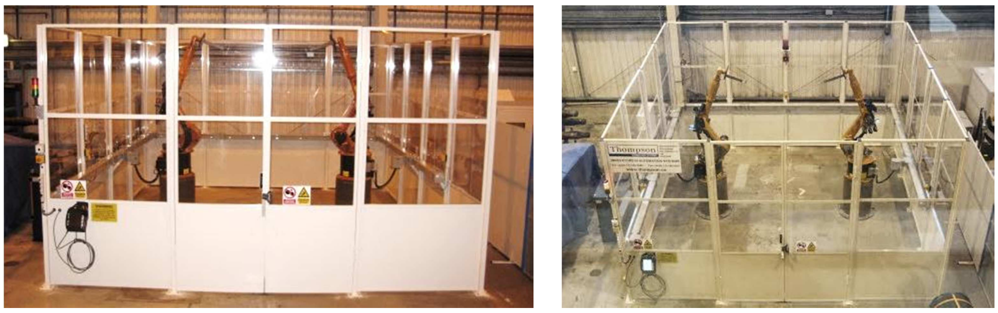


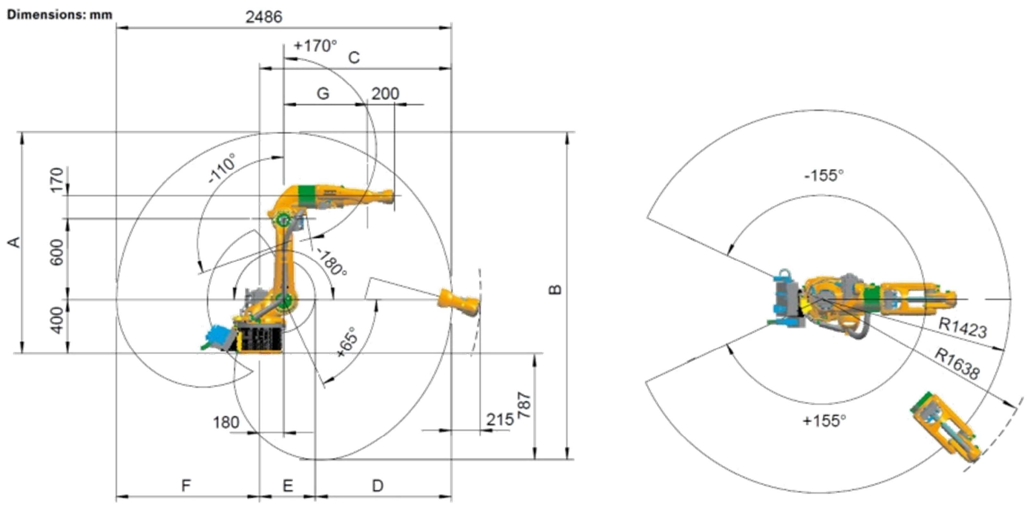
Figure 3.5 – Overviews of the TWI Robotic cell at the time of its establishment, in January 2013.

This system supported most part of the research carried out by the PhD programme: modelling of scan paths for NDT purposes, investigation of robot positioning systems, development of end-effectors and final integration of different technologies into a stand-alone automatic NDT device for complex surfaces.

Like the KUKA KR5ArcHW at Strathclyde, the KR16 L6-2 robots are 6-axis articulated arm robots. The features of KR5ArcHW and KR16 L6-2 robots are shown in Table 3.2, Table 3.3 and Table 3.4 [24, 25].

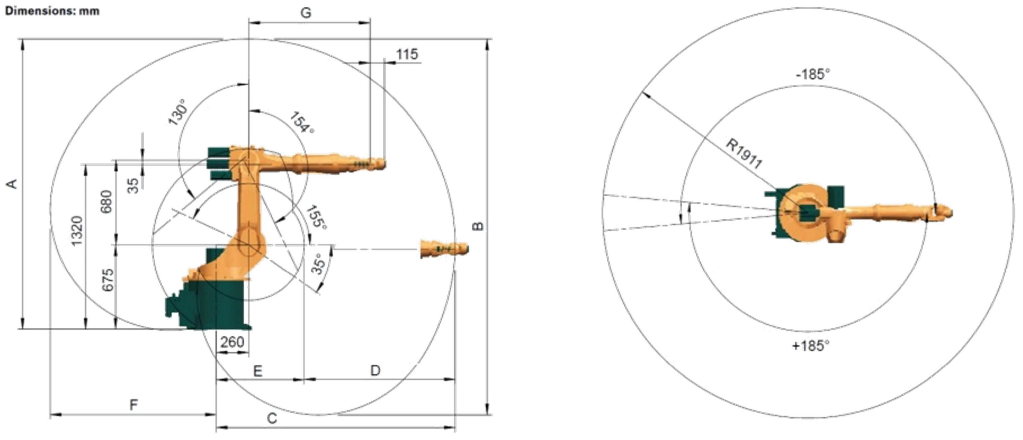
	KR5 Arc HW	**KR16 L6-2**
Payload	5 kg	6 kg
Maximum total load	37 kg	36 kg
Max. reach	1423 mm	1911 mm
Number of axes	6	6
Position repeatability	<±0.04 mm	<±0.05 mm
Controller	KRC2	KRC4
Weight	126 kg	240 kg
Operational temperature	+10 to +55 °C	+5 to +55 °C
Protection classification	IP 54	IP 65
Connection	7.3 kvA	7.3 kvA
Noise level	< 75 db	< 75 db
Mounting positions	floor, ceiling	floor, wall, ceiling

Table 3.2 – Technical details of KUKA KR5 Arc HW and KR16 L6-2 robotic arms.



Work envelope	Dimensions							Volume
	A	B	C	D	E	F	G	
KR 5 arc HW	1,643 mm	2,430 mm	1,423 mm	967 mm	456 mm	1,063 mm	620 mm	9.8 m³

Table 3.3 – Working envelope of the KR5arcHW Robot.



Work envelope	Dimensions							Volume
	A	B	C	D	E	F	G	
KR 16 L6-2	2,326 mm	3,011 mm	1,911 mm	1,206 mm	705 mm	1,327 mm	970 mm	24.0 m³

Table 3.4 – Working envelope of the KR16 L6-2 Robot.

The KR16 L6-2 robots are based on a newer generation of robot controllers (KRC4 – fourth generation of KUKA controllers). Moreover, the two robots at TWI are configured in a master-slave relationship[6] and each controller contains the following software add-ons:

- Robot Sensor Interface - RSI V3.1.2
- EthernetKRL V2.1.1
- RoboTeam V1.0.2

[6] In the control of multiple robot cooperation systems, the master-slave relationship is still the state of art for industrial robots [26]. The master-slave relationship identifies one robot as master, the others as slaves.

The Robot Sensor Interface (RSI) software is also in use with the KRC2 robots at UoS. RSI makes it possible to influence the robot motion or program execution via sensor data. The sensor data and signals can be read by a field bus, processed in the RSI context and forwarded to the robot controller. Optionally, the data can also be exchanged via Ethernet as XML strings. EthernetKRL makes it possible to set up, under the Robot Sensor Interface, a cyclic Ethernet link between a robot controller and up to nine external systems (e.g. sensors). The data are transmitted via the Ethernet TCP/IP protocol as XML strings. The received data are saved in a buffer so that no information can be lost. RSI and EthernetKRL have been implemented in the system to allow streaming of current positions from the controllers.

As mentioned in Section 3.2.8, the RoboTeam application package enables precisely coordinated teamwork between up to 15 robots through fast synchronization of the path motions. Each robot can keep its standard controller. A high-speed local network (Ethernet) is used by the controllers to communicate with one another and synchronize their motion. RoboTeam groups can be programmed conveniently and transparently using inline forms that contain all the command parameters and prevent the possibility of incorrect entries.

The RoboTeam software installed on each Robot controller allows the following types, or combinations, of program options [23]:

- Independent operation
- Synchronization points
- Synchronized motion
- Direct Geometric coupling
- Indirect Geometric coupling

In independent operations, each robot may run its own complete independent program steps with no consideration for the other robot within the cell. Neither the master robot nor the slave robot has visibility of the other and so, when operating in this mode, responsibility for avoidance of collisions lies entirely with the operator/programmer.

Whilst operating independently, it is possible to introduce synchronisation reference points into programs running on each robot. Using a unique identifier, when a robot reaches a synchronisation point it will wait for all robots in the cell to reach this point before continuing. Once this has occurred, each robot continues with its programmed commands. As with independent operation, each robot is unaware of the physical location of the other before or after a synchronisation point.

Synchronised Motion is the mode allowing individual linear or circular motion instructions to run in a synchronised manner. Using a unique identifier against the motion step in each program, the co-operating software asserts that the motion of each robot will start and stop synchronously. As with independent operation, each robot is unaware of the physical location of the other before, during or after a synchronisation point; only the timing is co-ordinated.

Utilising the Master-Slave relationship, the co-operating software allows geometric coupling of the Slave robot to the Master, when Direct Geometric Coupling is programmed. Using a unique identifier, a geometric linking command is inserted into each program. From this point forward, the Slave will follow the motion of the Master by maintaining the orientation and position of its active tool to that of the Master. Following the linking command, the Master program contains motion instructions to be followed. The Slave program contains no motion commands but follows every movement of the Master. When co-operation is no longer required, geometric tracking is cancelled with a similar linking command.

Indirect Geometric Coupling extends from direct geometric coupling; the Slave robot can follow its own motion path whilst still geometrically linked to the Master. This scenario is facilitated by the Indirect Geometric Coupling. For example, a part can be held and manoeuvred by the Master robot whilst the Slave robot completes some operations. This is achieved in a similar manner to direct coupling, with the addition of motion steps in the Slave program. Whilst coupled, the Slave moves through motion commands, changing position and orientation relatively to the Master robot base reference system. In an adaptation to the previous co-operating mode, the Slave program now includes motion instructions whilst coupled to the Master. Therefore, the Slave robot will complete its programmed motion, executing it relatively to the Master robot base.

3.4. Review of inspection strategies

Between 2010 and 2012, some investigations were carried out at UoS to develop infant systems for the robotic ultrasonic inspection of simple geometries (e.g. small flat components); this section describes the most relevant of those systems. The setups are explained and some relevant testing outcomes are discussed. This review provides an overview of the status of the research, found at the beginning of the PhD project related to the present thesis. The attention is focused on two infant inspection strategies in particular. The first allows discrete motion of a robot arm to carry out raster scan tool-paths for the ultrasonic inspection of small plates. The robotic motion type achieved through this system is discontinuous, with the robot stopping at each of the sampling points. The second system paves the way to continuous motion of the robotic manipulator, reducing the overall scanning time.

Since the necessity of using six-axis robotic arms arises from the task of being able to follow complex curved paths, maintaining a perpendicular orientation and a constant distance from the surface with ease, the potentiality of the robotic manipulators will not be used until the system will be able to automatically generate optimum scanning paths with the minimum human intervention. The wider scopes of the core research programme at the UoS and the industrial project at TWI (IntACom) were taken into account at all times during the research. The conceptual representation of the ideal system is given in Figure 3.6.

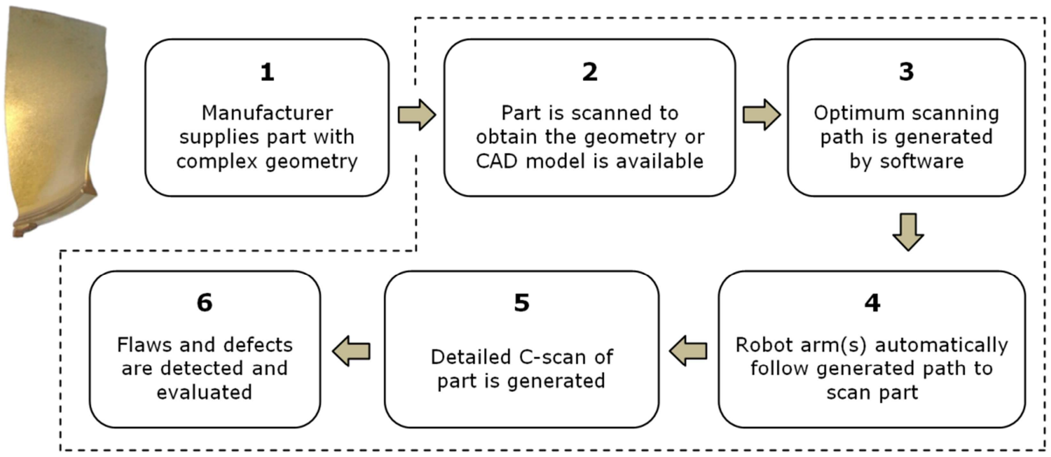

1
Manufacturer supplies part with complex geometry
2
Part is scanned to obtain the geometry or CAD model is available
3
Optimum scanning path is generated by software
4
Robot arm(s) automatically follow generated path to scan part
5
Detailed C-scan of part is generated
6
Flaws and defects are detected and evaluated

Figure 3.6 – Overall intended system architecture.

The remaining part of this chapter is organized to give an overview of all main elements necessary to enable the inspection process described in the scheme above. These elements are related to robot control strategies, interfacing with NDT instruments and data encoding.

The system described in Section 3.4.1, simply focuses on block 5 of the scheme in Figure 3.6. A raster scan of a flat surface is manually programmed via a KRL module and the ultrasound data is acquired by an external computer. Through the second configuration, in Section 3.4.2, block 4 is considered. The novel robot control strategy explained in Section 3.5 allows the potential implementation of an automated process covering blocks 2 to 6.

3.4.1. Discrete point measurements

The setup of this robotic inspection system comprises two control units working together:

- Data Acquisition System (DAS)
- Motion Control System (MCS)

Each control unit is controlled by a separate computer. The MCS is responsible for all the robot movements, and the DAS is responsible for collecting and storing data. The MCS is designed to be a "dumb machine". In this system, the robot has no knowledge of what is attached as its tool, and neither does it know how to interact with it. However the entire scan motion is defined in the MCS. In short, the system worked in the following manner:

1. DAS starts script
2. MCS starts module
3. DAS waits for MCS to reach first point
4. MCS moves to a new position
5. MCS tells DAS it has reached position
6. DAS confirms, performs scan, and tells MCS to move on
7. MCS confirms, and unless end of scan is reached, repeats back to 4.
8. MCS tells DAS end of scan is reached; the robot moves to the initial position (HOME position) and the robot programme ends.
9. DAS confirms end of scan, and exits script.

The MCS determines all the movement. The MCS is controlled by the KUKA Robot Controller (KRC), where all the movements are defined through a KRL module. Figure 3.7 shows a block diagram with a representation of the overall implementation.

The KUKA Robot Language (KRL) module is created by the user and stored in the memory unit of the KUKA Robot Controller (KRC) to command the robot to follow predefined paths. All of the MCS is controlled by the KRC. In other words, the KRL module executed within the embedded Windows XP in the KRC. Since the definition of the tool central point (TCP) and of the base reference system influences the precision of all the robot movements, great care is taken in the calibration of both.

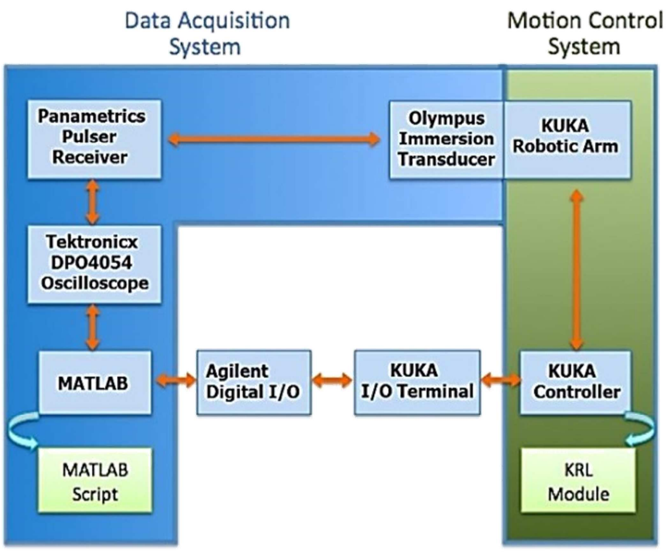


Figure 3.7 – Diagram of implemented system architecture

Figure 3.8 shows a representation of the inspection setup and one moment during the calibration of the tool. A steel rod with the same length of the transducer was mounted on the robot wrist and the TCP calibration was carried out by the standard KUKA calibration procedure, jogging the TCP to a fixed reference point using 4 different orientations.

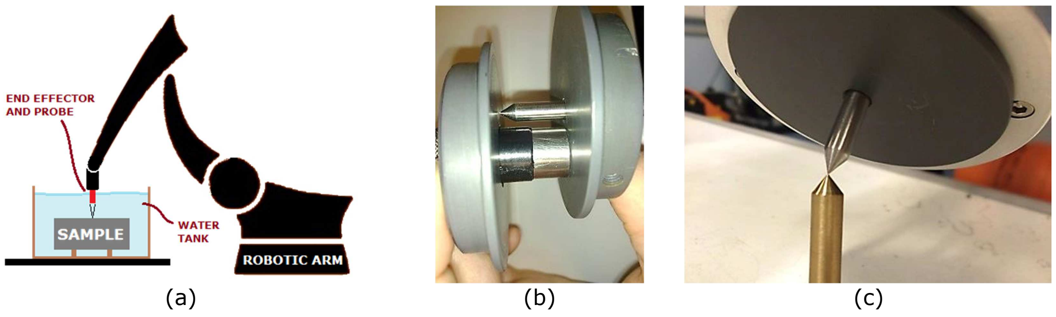


Figure 3.8 – a) Inspection setup; b) Calibration rod with same probe length; c) Tool calibration.

An initial MATLAB software script was developed for the PhD work. The software was used both to manage the acquisition system, trigger the movements of the robotic arm and to synchronize it with the start and the end of the signal collection. A key aspect of this system, regarding the establishment of a basic level of automation, is the communication between the two control systems. This is achieved by ensuring the two systems have a means to communicate in a full-duplex manner. The digital I/O (Input/Output) terminals of the respective systems are used. MATLAB with Virtual Instrument Toolbox was interfaced with a signal sampling unit (Tektronix DPO4054 Oscilloscope) and with a device (Agilent U2121A) for the management of the digital I/O terminals of the robot controller. The main features of the sampling unit are given below.

- 500 MHz Bandwidth

- 4 Channel
- Sample Rates Up to 5 GS/s
- 10 Mega-sample Record Length
- Built-in Ethernet Port

This device provides the possibility to operate with an x-cross Ethernet connection (instead of the more common, but slower, USB connection) and with a high sampling rate. On the other hand Agilent U2121A is a USB digital I/O peripheral, used to establish communication between MATLAB and the KUKA controller. U2121A has the following features:

- 32 terminals (16bits per Input/Output)
- USB driven
- Optically isolated
- Input range: 0 to 24 V
- Output Voltage load: 24V max
- Compatible with MATLAB

In order to test the system, a 5 MHz single element immersion transducer with a 50.8mm focal length was mounted to the end effector of the robotic arm and a pulse-receiver (*Parametrics* 5052PR in Figure 3.9b) was used to activate the transducer (in Figure 3.9a). This unit has external output for "+Sync" and "Signal Output" and a bi-directional terminal for "T/R" (standing for Transmit and Receive). Every time a pulse is outputted by the pulse-receiver unit, the unit automatically sends out a synchronisation pulse via the +Sync terminal — this terminal was connected to the data acquisition unit. The synchronisation pulse allows the data acquisition unit to trigger its trace, so there is no delay in the data acquisition.

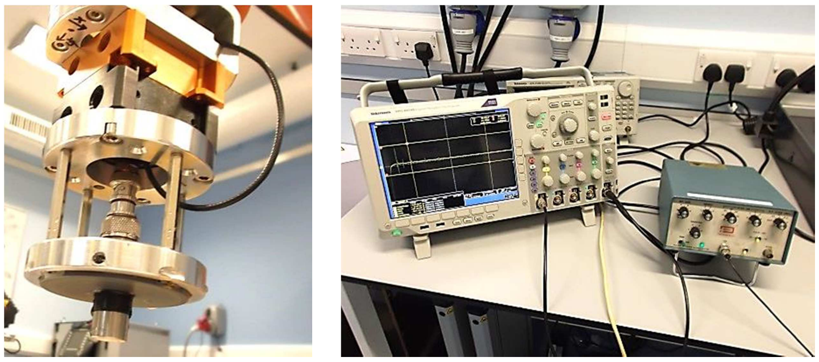

(a) (b)

Figure 3.9 – a) 5 MHz transducer mounted on the robotic arm wrist; b) Pulser and Oscilloscope.

While Figure 3.7 shows the diagram of the implemented system architecture, Figure 3.10 provides an explicit diagram of all the connections among the elements of the system.

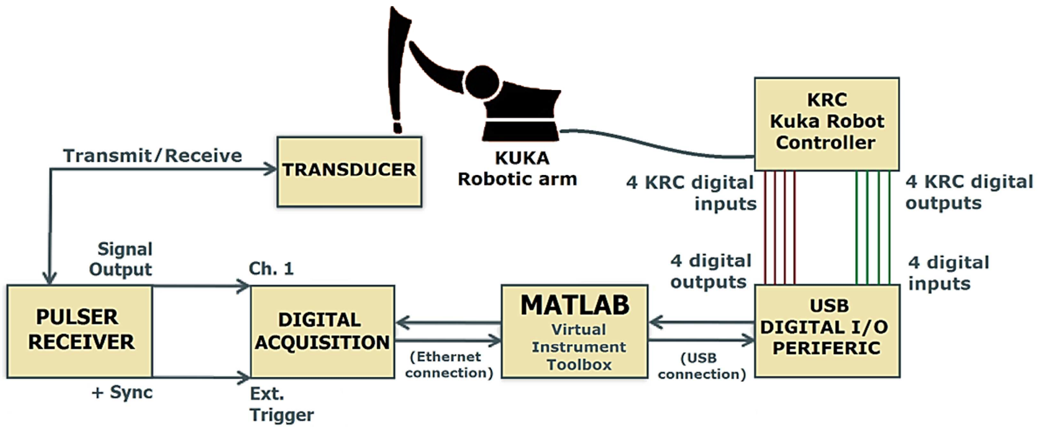


Figure 3.10 – Diagram of all the connections among the elements of the system.

In summary, MATLAB can alter Agilent digital outputs, and check for the logic levels of the digital inputs. Likewise, the KUKA controller can change the logic states of its digital outputs whilst checking for its digital input logic. Agilent U2121A's output is an active low device – thus when an instruction is sent to change internal logic of terminal 1 at channel 1 to LOW, the KUKA digital input observes logic HIGH value.

The system is configured so that all the data acquired at every position of the scan gets dumped immediately into a predefined folder, along with a *.mat* file which stores some key variables of the scan. All the stored data are loaded in by a MATLAB script to perform the desired post-processing and data analysis. The system is implemented so that the scan area is not limited by the available computer memory space and very basic hardware requirements are needed. The collected A-Scans were saved into the hard disk of the acquisition computer as binary files soon after their acquisition. If all scan data were stored into one MATLAB variable as the scan was performed, the scan resolution and area would be limited by the space allocated for MATLAB and thus most likely cause an error during a larger scan. Also, this way the full set of raw data sampled by the Tektronix DPO4054 Oscilloscope gets stored; so if the user wishes to perform a custom post-processing algorithm, it can be accommodated.

3.4.1.1. Results

In order to observe the capabilities of the system to perform raster scans, a metal test-piece was scanned (Figure 3.11). The metal piece was a steel block (100 x 70 x 12 mm) with 4 through holes drilled near the corners and a blind hole drilled half-way through the thickness in the middle of the block. The 4 holes in the corners have a diameter of 10 mm, whereas the centre hole has a diameter of 7 mm. The metal block was placed in an immersion tank with the centre hole not visible when observed from above. The standard KUKA method for the calibration of a base reference system, the *3-point method*, was utilised to define a base for the robot tool-path.

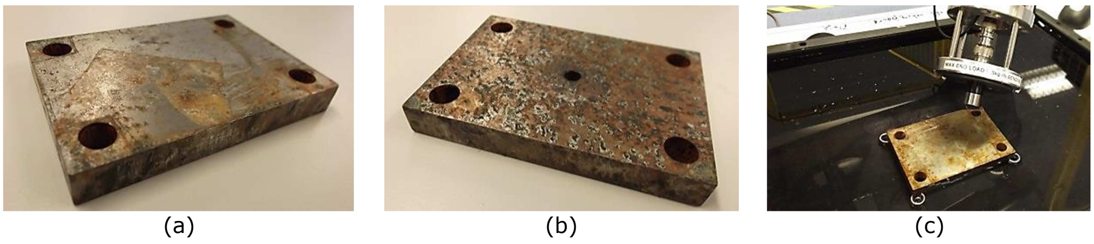


Figure 3.11 – Metal block from top (a) and from beneath (b). KUKA Robotic arm approaching the starting point of the raster scan (c).

The tank was then filled with water (Figure 3.8a and Figure 3.11c), and the system performed a raster scan where the focal point of the transducer was set to 6 mm from the metal block's surface (half of the sample thickness). The A-scan signals were recorded with resolution of 1 mm in both *x* and *y* direction.

A post-processing MATLAB script was developed to evaluate the amplitude and the Time-of-Flight (TOF) of the waveforms. The script is able to measure the maximum, the minimum and the peak-to-peak amplitude in a given time window of interest, or register the differences in the TOF for the first echo of the longitudinal wave. The TOF is measured in correspondence of the peak of the signal Hilbert transform, in order to make the measure insensible to the wave phase. Since the Hilbert transform is calculated inside the script to perform the TOF measurement, its maximum is also outputted by the post-processing process. Figure 3.12 shows all the resulting C-scans.

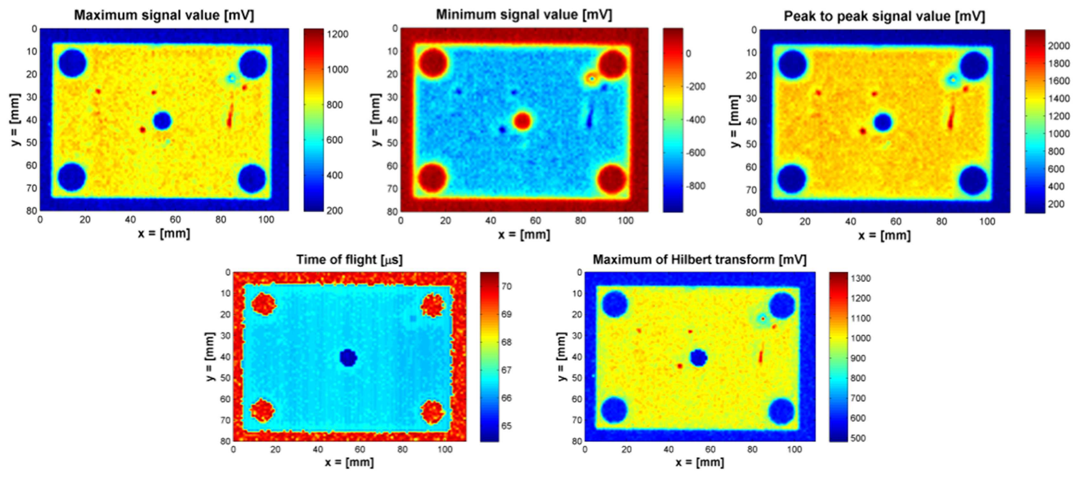


Figure 3.12 – C-scans of metal block at the end of post-processing. On the top, minimum wave amplitude (left), maximum amplitude (middle) and peak to peak amplitude (right). On the bottom, Time of flight of the echo (left) and maximum of Hilbert transform (right).

The resolution of the C-scans is good enough to conclude that the centre hole is smaller in diameter than the rest of the holes, with the four corners having a diameter of 10 mm whilst the centre hole having a diameter of roughly 7 mm. This means the system can be used to analyse flaw sizes within materials.

The system was also tested scanning a carbon-fibre-reinforced polymer (CFRP) plate. Figure 3.13 shows the sample, a 4 mm thick CFRP plate with a circular depression in the middle.

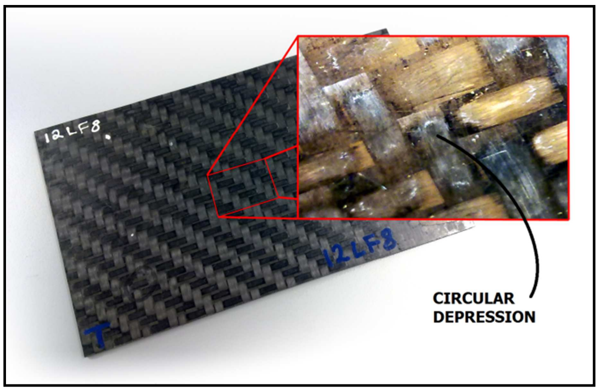


Figure 3.13 – 4 mm thick CFRP plate with a circular depression in the centre.

The system operated a raster scan of a rectangular area (21 x 30 mm) around the defect and the signals were recorded at intervals of 0.25 mm in both x and y direction. The same post-processing program used for the metal block was deployed to output the results. Figure 3.14 shows the relative C-scans.

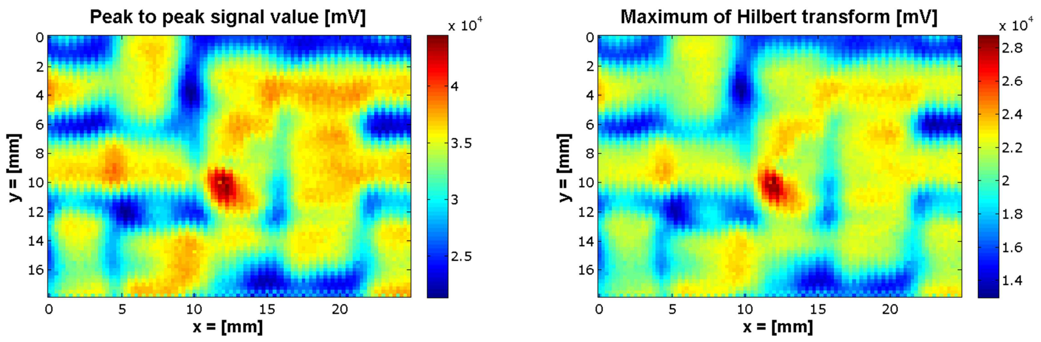


Figure 3.14 – C-scan images of CRFP plate with a circular depression. Peak-to-peak amplitude value (left) and maximum of Hilbert transform (right).

It can be seen in both the peak-to-peak value and maximum of Hilbert transform C-scans that the circular depression in the centre is clearly recognizable. Looking at both the C-scans, the red region near the centre demonstrates that a higher energy was reflected back at the focussed layer – which suggests that there is more acoustic impedance present in the area. This suggests structural damage.

The raster scan of the same area was carried out with a commercial ultrasound microscope system (SAM 300 – PVA TePla), using the same transducer. The device was able to record the signals at intervals of 0.1 mm in both *x* and *y* direction. Figure 3.14 shows the resulting peak-to-peak map and the map with the maximum of the Hilbert transform.

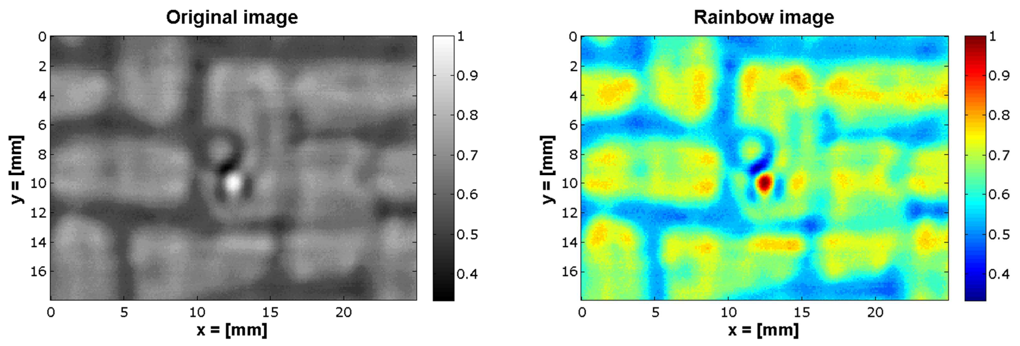


Figure 3.15 – C-scan image obtained with an Ultrasound Microscope (SAM 300 – PVA TePla); Original image outputted by the system (left) and Rainbow-scale image obtained with MATLAB (right).

3.4.1.2. Highlights

One of the most important objectives for the development of robotic inspection of complex geometries is achieved in this first system, although with infant performances (slow speed and basic level of automation). The objective of coordinating a system to perform a raster scan was successfully reached both on homogenous and composite materials. This paved the way for future developments.

Important necessary improvements to this system were easy to establish. The following is a list of the main drawbacks of the system:

- The scanning speed is not sufficient to compete with the speed of other commercial systems. The average speed of the system, evaluated as number of scanned points per minute, was equal to 88 A-scans/min (for 100μs long A-scans sampled at 50MHz). This is very slow speed. The cause of this drawback is evident in the fact the robot manipulator must halt at each scan position (typically moving only a fraction of a millimeter each time), indicate to MATLAB that it is in position and wait for MATLAB to complete the data acquisition before progressing to the next scan point. This results in a scan time of up to several hours for a modestly sized plate.
- A better communication between DAS and MCS is required. Four-signal full-duplex communication is not sufficient to transmit both triggers and

geometrical coordinates (robot feedback); it requires at least 16 signals or using Ethernet ports.

- The DAS MATLAB script relies on the operator manually entering the scan dimensional data which has been manually programmed into the KUKA module. The script will only function if a flat plate is being scanned using a raster scan path and if the details are entered correctly. Although this works for flat plates, neither the data acquisition script nor the post-processing code allow for future developments to include complex geometries or any other scanning path other than a simple raster path.
- The KRL modules used for this system have to be manually edited for every different set of dimensions (width and length) of the rectangular area of interest. It is time-consuming and prone to human errors.

3.4.2. Continuous path measurements

The possibility of using continuous scanning motion was achieved in the second system. The previous data acquisition system, based on a MATLAB script, was modified to acquire ultrasonic A-scans continuously throughout the scanning process. The code triggers a pulse from the *Panametrics* Pulser/Receiver which is transmitted from the ultrasonic transducer on the end of the robot arm.

To enable the robot to move in a continuous, fluid motion it is necessary to know the exact position of the robot when an A-scan is captured. At the basis of the continuous motion usage is the introduction of the time-stamps. The ultrasonic waves are picked up by the transducer and the resulting waveforms are captured by the oscilloscope. Each waveform is imported into MATLAB, where it is time-stamped and saved.

An Ethernet connection was established between the robot controller and an external PC. This allowed the transmission of the robot's positional data (the six coordinates: X, Y, Z, A, B and C) in real time. Likewise for each collected waveform, every data packet from the robot was time-stamped and saved. Once a scan was completed, a MATLAB post-processing script was able to match the time-stamp of every A-scan waveform to the corresponding robot position and then process every data point to produce a C-scan graphic.

The second generation of the KUKA controller (KRC2) of the KR5arcHW robot is divided into three main systems. The graphic user interface (GUI) allows the user to write and execute robot programs, through defining robot bases, tool parameters and by jogging the robot arm. This GUI is run from an embedded version of Windows XP. However, hidden from the user is a separate operating system called VXworks. This is a real-time operating system (RTOS)[7] which is designed for embedded applications and is developed by Wind River Systems [27]. The VXworks system controls all the robot's server controllers and is used because of its multi-tasking capabilities, real-time performance and reliability.

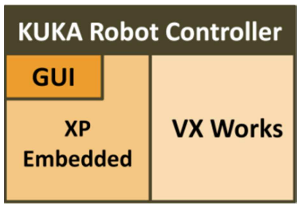


Figure 3.16 – KUKA Controller architecture.

Although running on the same processor, the XP and VXworks systems are entirely separate from each other. Any information which is passed between them – for example when a command button on the GUI is pressed – is sent over a virtual TCP/IP connection within the KRC's architecture. There is no physical network cable but information is packed up, transmitted over a virtual connection, received and unpacked by the other system to be processed.

The KRC has a number of Ethernet ports available. However, the port which is mounted directly onto the motherboard is the only port visible to the Windows XP system and the only port which can be controlled by XP. Theoretically, this port could be used and any communications received on that port could be routed through the internal virtual connection to be processed by VXworks. This introduces a time delay, however, and can lead to a loss of integrity of the data packets while also possibly interfering with the robot's own necessary internal communications. The remaining Ethernet ports are mounted on network adaptors manufactured by 3COM and installed by KUKA on request. These network adapters are controlled by VXworks and therefore they are not visible from Windows XP. It is to one of these ports that the cable from the external PC must be connected for RSI communication.

The RSI software running in the KRC2 was configured to simply transmit the robot's positional data and digital output states over a TCP/IP connection but not to expect any

[7] A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application process data as it comes in, typically without buffering delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter. A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application task.

data or commands in return. Data are sent in XML file format and the structure of the packet is dictated by an XML file located in the robot controller.

Once a connection has been established with a server, the KRC2 robot controller transmits data packets according to the XML configuration file in a 12ms interpolation cycle. Appended to the end of every packet is a number identified as the IPOC[8]. This is a time-stamp for the packet. An increase of 6 in the IPOC number equates to a 12ms time difference, it is one interpolation cycle of the RSI software. The robot expects the server to extract this time-stamp and append it to the return packet which the robot expects to receive within the same 12ms cycle. If it does not receive the IPOC number back within the specified time then the packet is deemed late – which could mean that it was not delivered at all – and if the total number of late packets exceeds the set maximum then the robot automatically stops [14].

It was initially planned to use the MATLAB Instrument Control Toolbox to receive communications from the robot since it has inbuilt TCP/IP and UDP capabilities. This would involve creating a TCP object inside the MATLAB script, linked to the robot's IP address. However, this code cannot work with the robot for three fundamental reasons:

1. This code, and the Instrument Control Toolbox, can only allow the external PC to act as a client. That is, the PC would request a connection from a server and would fail if there was no server waiting to accept its connection. However, the robot is also acting as a client and not a server and therefore will also fail if there is not a server awaiting its connection request. It is not normally possible to have client to client communication.
2. Moreover, writing a MATLAB Java script which would act as a server to receive communication (employing a convenient feature of MATLAB where Java functions can be imported and called), a problem arises when the MATLAB code includes many functions. The part of code to trigger a pulse to the ultrasonic transducer, capture the data from the oscilloscope and display it in a MATLAB plot, distracts the communication code from listening continuously. The result is that the script would read every received data packet in order, but they would be out of date.

[8] A uint32 value representing the lower 32 bits of the 64-bit Interpolation Cycle counter (IPOC). The IPOC indicates the current time-stamp of the communication data packet from the robot controller that generates it automatically. As long as the time-stamp value changes and is updated, the RSI application is running and connected.

3. The IEEE Standard 1003.1 reveals that a feature of the TCP/IP protocol is to create a buffer where store data received by a socket until they are removed by a read call [28]. Essentially this means that for the data to be current then MATLAB must be continually reading the input stream to keep the socket receive queue empty. MATLAB, however, is a single-threading application meaning it can only perform one action at a time and therefore would not be able to sustain communications and also control the ultrasonic equipment.

Instead, a server application was developed using the C# programming language. It is possible to write multi-threading programs in C# which allows multiple actions to take place simultaneously meaning that a queue of packets would not build up. The basic operation of the code is as follows:

- The code spawns four threads: initially the main thread from where the three other threads are started. One thread handles receiving communications from the robot, another handles mirroring the IPOC time-stamp back to the robot and the last saves the received data.
- The data which is received is not processed in any way by this code beyond numbering each data packet and prepending this number to the packet. Two arrays are used as containers for the data packets, each 1000 lines long. The communications thread fills the first array until it is full. At this point, it passes the array to the saving thread while immediately beginning to fill the next array. This means that not a single packet is missed. The saving thread writes the array to a text file and stores it in a predefined directory. This process takes a few seconds so if this were not to be in a separate thread, it would interrupt the communications.
- Since all the saved data is time-stamped, it is possible to know the robot's exact position at any given time during a motion.

A multi-threaded C# application can be launched from MATLAB and it can then run independently from what MATLAB is doing in its command window. This led to the system architecture shown in Figure 3.17.

The MATLAB script developed for the initial system was edited to include features compatible with the new scanning approach. When initially run, MATLAB creates a new directory to save the scan data to. This was edited to create two directories: one for the ultrasonic data saved by MATLAB and another for the positional data from the KUKA.

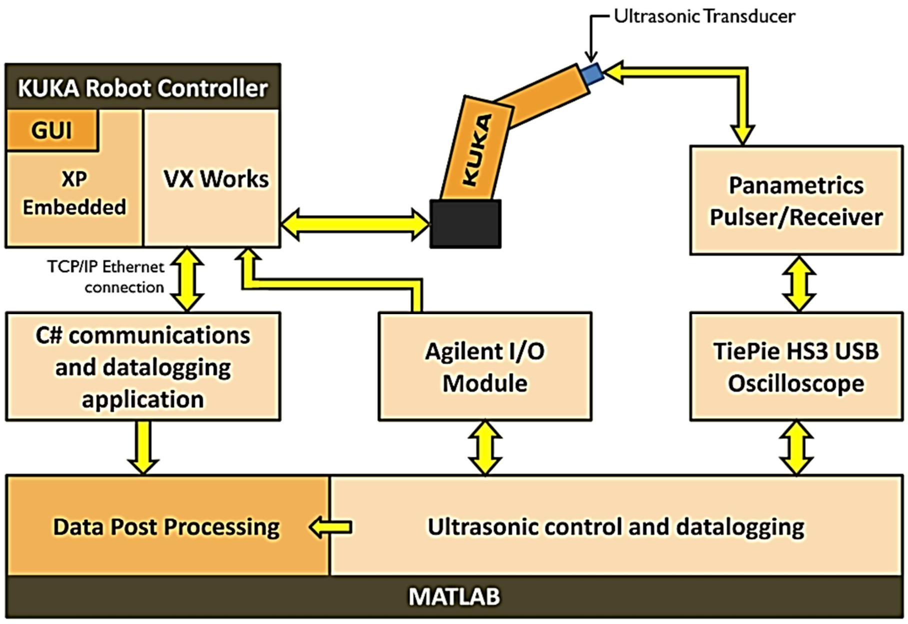

Ultrasonic Transducer
KUKA Robot Controller
GUI
XP Embedded
VX Works
KUKA
Panametrics Pulser/Receiver
TCP/IP Ethernet connection
C# communications and datalogging application
Agilent I/O Module
TiePie HS3 USB Oscilloscope
Data Post Processing
Ultrasonic control and datalogging
MATLAB

Figure 3.17 – System architecture.

The directory of the KUKA data is saved to a text file which can then be read by the C# program so that it will save to the correct folder. MATLAB then launches the C# program as a standalone console application. Both programs have visibility of the robot's digital outputs: MATLAB through the Agilent I/O module and the C# program through the information included in each data packet. This is used to synchronise the time-stamps between the two programs. It is also used to prevent the programs saving data that is unnecessary for example once communications have been established but before the robot has reached the starting point of the scan. When scanning is underway, MATLAB continually loops the code which performs ultrasonic A-scans and saves the data, no longer waiting for instruction from the robot on when to fire each pulse, like it happened with the discrete path system.

The previous post-processing script was modified to be used in the new system. The program first loads each of the text files containing 1000 data packets from the KUKA and copies the coordinates and the time-stamps of every data packet into a large array. Once this parsing is complete, the program loads each A-scan data in turn, matching that time-stamp to a KUKA time-stamp. The code supports gating the ultrasonic data, allowing the waveforms to be analysed.

Once every A-scan has been loaded and matched to the corresponding robot position, the program contains four arrays: three with X, Y and Z coordinates and a fourth with the peak values of the gated ultrasonic data. These four arrays can then be used to plot an ultrasonic C-scan using a scatter plot.

A scatter plot is used instead of a surface plot as the latter assumes constant spacing between each A-scan (each pixel). However, since the time taken for MATLAB to save an A-scan varies slightly and since the robot does not follow an exactly straight path, a scatter plot is more suitable as it allows each pixel to be exactly placed in 3D space according to where the data was measured during the scan. In this way, this system can also display complex geometries with ease.

3.4.2.1. Results

The same test piece shown in Figure 3.11, which was scanned with the first system, was also scanned in a continuous raster tool-path using the new scanning setup. The resultant C-scans are shown in Figure 3.18.

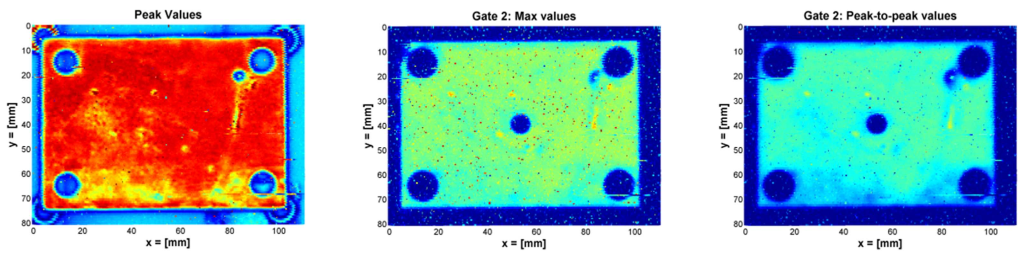


Figure 3.18 – Section images of metal block at the end of post-processing. Peak value of the first wave (left), maximum value (middle) and peak to peak value (right) of the second wave.

The C-Scans on the left hand side of Figure 3.18 was obtained gating the first ultrasound reflection from the surface and it shows the external geometry. The blind hole can be clearly seen in the other two C-scans, obtained gating the second wave of the signals (back-wall reflection).

The scan of the CFRP plate with a circular depression in the centre (see Figure 3.13) was also carried out. The result is shown in the following Figure 3.19.

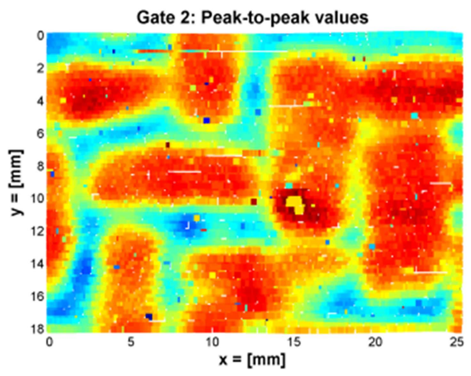


Figure 3.19 – C-scan of CRFP plate with a circular depression.

3.4.2.2. Highlights

Through using Ethernet communication to communicate the robot's positional data, the option of using continuous motions of the KUKA arm was opened. The way to display the c-scans, through using scatter plots, allows displaying c-scans of complex geometries with ease.

The acquisition rate of the system is 1056 A-scans/min, when 100μs long A-scans are sampled at 50MHz. This reduces scanning times by a factor of 12 with no loss of resolution, comparing to the system with discontinuous robot motion.

The scanning speed is faster than the system described in Section 3.4.1. However the data acquisition rate is still lower than the rate of commercially available systems. A USL system and a GE system based on 3-DoF gantry manipulators were tested at TWI; the USL system was running the *USL Scanner acquisition software (Version 4.51)* and the GE system was equipped with the *GE Nutronik UTxx software*. Both the systems proved to be capable of acquiring 60000 A-scans/min, when each A-scan is 100μs long and is sampled at 50MHz.

Therefore the data collection in commercial 3-DoF systems is 57 times faster than in the continuous motion robotic scanner introduced in Section 3.4.2. Moreover, the acquisition rate of commercial systems hits the limit of the physics of the ultrasound waves generated by single crystal piezoelectric probes. Faster acquisition rate would not be possible, since the pulse repetition frequency is limited by the *ultrasonic ghosting*.

Commercial gantry inspection systems able to collect 60000 A-scans/min (3.6 million A-scans/hour) provide a scanning rate of 3.6 m^2/hour (assuming a scan resolution of 1mm). The following part of the thesis presents the research strategies adopted to program 6-DoF robotic manipulators with ease and to develop acquisition software able to equate the acquisition speed provided by gantry systems. The final performances of a robotic inspection system prototype capable of inspection curved surfaces will be described in Section 7.4. It will be shown how the use of ultrasonic phased-array probes significantly improved acquisition rates, compared to current systems.

3.5. External control

The approach enabled by the second inspection strategy, in Section 3.4.2, allowed a better degree of automation. The robotic arm could be programmed to start and finish the entire scan and complete it in a smooth motion with no pauses. Scatter plots, used instead of a surface plots, allowed each output value to be placed in 3D space according to where the data was measured during the scan; in such way the post-processing script can also display C-scans of complex geometries with ease. Despite of the potentiality of scanning complex geometries, the motion of the arm was still programmed manually. This prevented the possibility to scan complex surfaces.

In order to allow complex geometries it is necessary to generate the optimum scanning path. Section 3.2.7 introduced the two traditional approaches to the path-planning problem and it speculated about the possibility to execute complex robot movements through establishing a new method to control the robot manipulator; the external real-time control of the robot controller from a server computer.

The capability of externally controlling the motion of KRC2 robots was investigated and its implementation is described in Appendix A1. A custom C# application was written to provide external control of KRC2 robots. The application is a multi-threading code receiving communications from the robot and mirroring the IPOC time-stamp of the robot feedback packets. For each interpolation cycle, the application reads a new target position from a text file and attaches the position coordinates to the outgoing XML packet to be sent to the robot controller.

3.5.1. Experimental tests

The post-processing MATLAB code described in Section 3.4.2 was used to evaluate the features of the new external control system. To prove the system, the generated file of command coordinates, “coordinates.txt”, was used to scan the test piece showed in Figure 3.11. The part was scanned in a continuous and externally controlled raster tool-path.

Through troubleshooting some software issues, reported in Appendix A2, reliable continuous motion of the robotic arm was enabled; this guaranteed high quality C-scan images. Figure 3.20 shows the resulting C-scans.

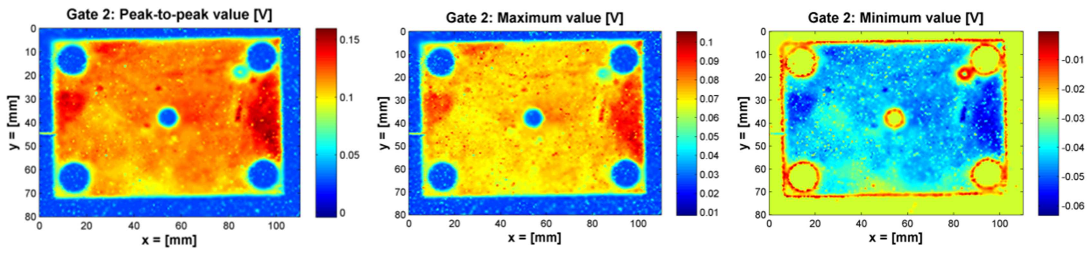


Figure 3.20 – C-scans obtained with the externally controlled motion, after achieving reliable connections.

3.6. Accuracy investigations

The new C# code, described in detail in Appendix A2, enables the possibility of carrying out investigations of the error in the execution of the robot tool-paths, comparing them with the commanded target paths.

Since a single programme thread (see Appendix A2) is responsible for managing both the target coordinates and the reception of the current robot's position, this paves the way to the development of on-line (or real-time) path correction approaches.

Moreover, since the received stream of robot’s position data is saved in text files inside a directory of the external computer, an off-line comparison of the motion accuracy can be arranged with ease. It is important to highlight how this is a direct advantage given by the new structure of the code. It was not possible to ensure a direct correspondence between the sent packets of coordinates and the received data, when the activity of sending a command packet and the activity of receiving a feedback packet were managed by two different threads. There was no assurance that all the packets could be received and each error in the reception contributed in removing any trace of correspondence with the sent command coordinates.

The new code realizes a perfect correspondence between the outgoing target coordinates and the incoming feedback positions, since each new packet of coordinates is sent only after the reception of the current position, every 12ms.

3.6.1. Comparison of target and feedback positions

Using the possibility given by the new code, an off-line comparison between the desired path and the real path has been carried out and the results are presented in this section. When the C# code is started, the first string that is received from the KRC is related to the initial position of the robotic arm, since no packet of coordinates has yet been sent to say the KRC to reach the first point of the desired path. So the external computer sends the first packet of coordinate to tell KRC to start moving and continues receiving the next incoming string from the KRC. At first it was considered sufficient to compare each i-th sent packet of coordinates with the (i+1)-th received current position, conjecturing that each commanded position was reached in 12ms, that means only 1 interpolation cycle. Two error values were calculated: the position error (the distance between the target point and the real point in the Cartesian space) and the orientation error (the angle between the vector of the target normal direction and the vector of the obtained direction). Calculating these two errors for each point along the path it was possible to know the maximum error and the average error and obtain a final evaluation of the motion accuracy.

However, in order to validate the conjecture that each sent position is reached within a single KRC interpolation cycle, different comparisons were developed introducing an offset between the number of the sent packet and the number of the cycle where the target position is supposed to be reached. If *j* indicates the value of this offset (e.g. j=1,2,…,13,14) and *i* is the number of the i-th target position, the i-th position error and the i-th orientation error are given by the following:

$$PosError(i) = \sqrt{[Xreal(i+j) - Xtar(i)]^2 + [Yreal(i+j) - Ytar(i)]^2 + [Zreal(i+j) - Ztar(i)]^2}$$

Eq. 3.8

$$OrientError(i) = \cos^{-1}\left[\big(NXreal(i+j) \cdot NXtar(i)\big) + \big(NYreal(i+j) \cdot NYtar(i)\big) + \big(NZreal(i+j) \cdot NZtar(i)\big)\right]$$

Eq. 3.9

where *Xreal*, *Yreal*, *Zreal* and *Xtar*, *Ytar*, *Ztar* are respectively the Cartesian coordinates of the feedback position and of the target point. *NXreal*, *NYreal*, *NZreal* and *NXtar*, *NYtar*, *NZtar* are respectively the Cartesian component of the real angular direction of the robot wrist and of the target normal surface vector.

The maximum position error, the maximum orientation error, the root mean square (RMS) of the position error and the RMS of the orientation error can be calculated and plotted in function of the offset (*j*) in the cycle number. Figure 3.21 shows a plot of the errors versus the offset.

As it is possible to see in Figure 3.21, the minimum error is reached after 7 interpolation cycles of the robot controller from the cycle that send the target coordinates. This simply means that the KRC2 controller considered 7 packets of coordinates to effectively control the robotic arm. The knowledge of this aspect of the external control can be very useful for future development of on-line error evaluations and of real-time path correction systems.

Further information to understand more about the origin of this time offset between the transmission of the command packet and the attainment of the demanded position was found in the KUKA reference manuals [14, 29].

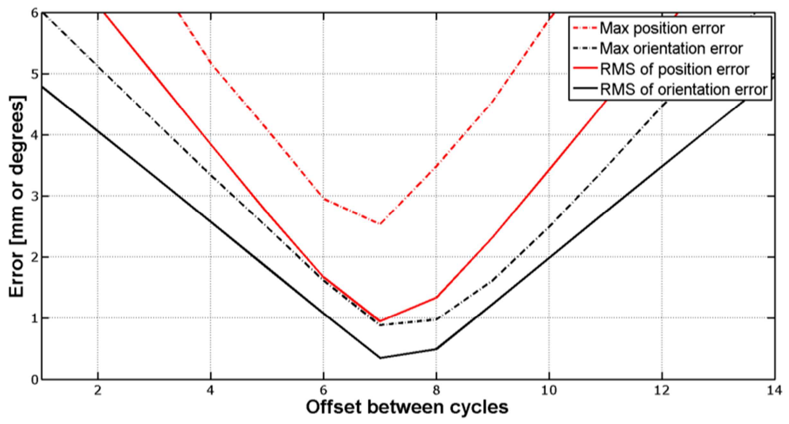


Figure 3.21 – Errors vs the offset in the cycle number.

The general implementation of RSI in the KUKA system architecture is shown in Figure 3.22. The tool-paths programmed in KUKA Robot Language (KRL) modules are executed by the drives after travelling through the first interpolation stage that runs in 12ms cycles. The resulting commands coordinates are filtered before undergoing a fine interpolation. Such interpolation is carried out by the second stage interpolator that produces coordinate packets every 4ms. RSI with 4ms interpolation cycle is only supported by KRC4 robot controllers.

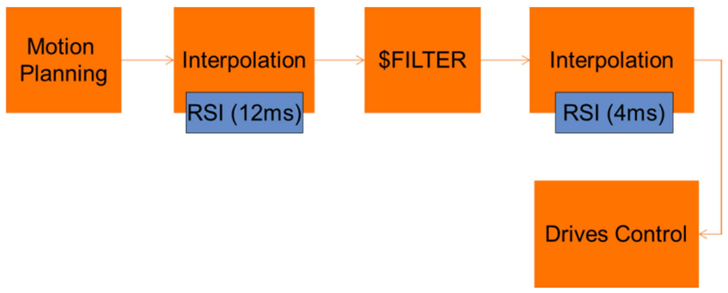


Figure 3.22 – Implementation of RSI in the KUKA system architecture.

Because of the architecture of the KUKA system, there is a delay between the command coordinates issued by the KRL module and the time when the command position is reached. This delay is usually called "dead-time". KUKA has carried out experimental work, obtaining an empirical relationship between dead-time and the value stored into the filter variable ($FILTER) [15]:

$$Deadtime = \begin{cases} (24\text{ms}) + \frac{\$FILTER}{2} & \text{for KRC2} \\ (36\text{ms}) + \frac{\$FILTER}{2} & \text{for KRC4} \end{cases} \qquad \text{Eq. 3.10}$$

When the RSI software operates in the 4ms interpolation cycle mode, the second part of the sum is omitted and the dead-time is minimum, equal to 36ms.

The empirical formulas clarify the reason behind the delays observed during our investigations. For the curves given in Figure 3.21, obtained with the KRC2 $KR5arcHW$ robot at UoS, the dead-time is equal to:

$Deadtime_{KRC2} = 7 \cdot 12ms = 84ms$ Eq. 3.11

The values of $FILTER calculated accordingly to the empirical formula is:

$\$FILTER_{KR5arcHW} = 120$ Eq. 3.12

The real value of the filter variable stored within the robot controllers was checked and it validated this result.

The external control was tested with a generated set of command packets, suitable for scanning a spherical surface. Figure 3.23 shows the robotic arm at the beginning of the raster tool-path. Four raster tool-paths were executed, in order to evaluate the accuracy of the system running at different speeds and with different steps in the x-direction. The tool-paths had speeds of 100, 50, 20 and 10 mm/sec and the respective steps equal to 2, 1, 0.5 and 0.25 mm. The errors of position and orientation were calculated comparing the command packets and the feedback packets of coordinates.

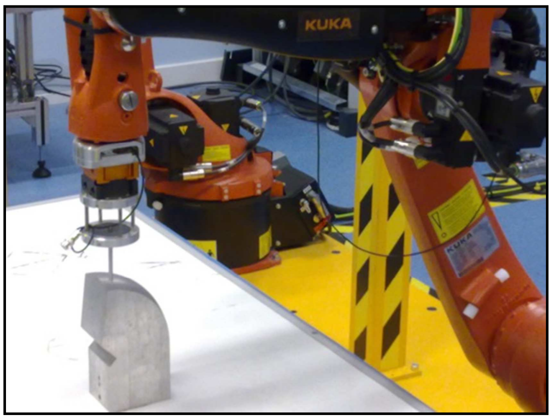


Figure 3.23 – Robotic arm at the beginning of the raster scan.

Figure 3.24 gives the resulting error maps for the four tool-paths. Both the position and the orientation errors are shown in three-dimensional scatter plots, presenting the most critical areas.

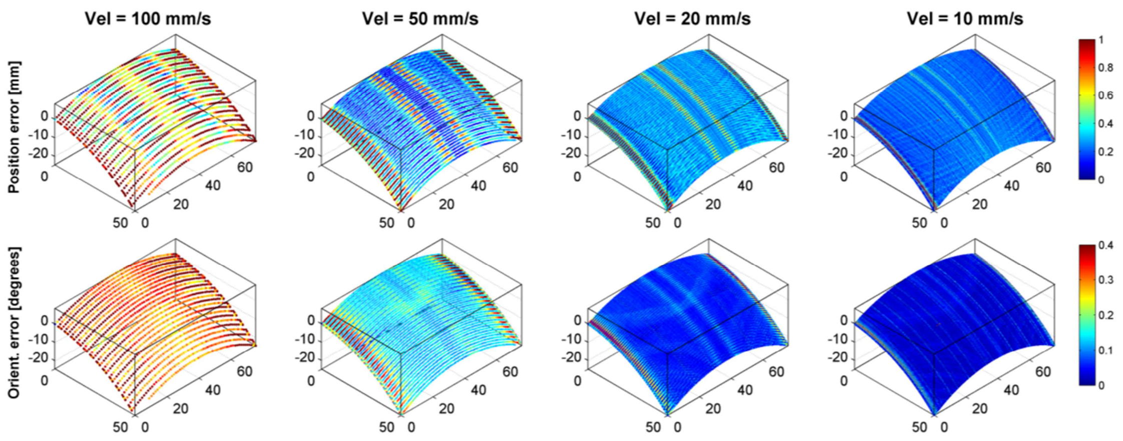


Figure 3.24 – Graphical representation of the system accuracy in the four raster scans.

The values of the maximum errors and the RMS values are reported in the following table for an easier comparison.

	100 mm/sec Step = 2 mm	50 mm/sec Step = 1 mm	20 mm/sec Step = 0.5 mm	10 mm/sec Step = 0.25 mm
RMS position error (mm)	0.95	0.52	0.38	0.27
RMS orient. error (degrees)	0.35	0.17	0.08	0.04
Max position error (mm)	2.53	1.97	1.84	1.71
Max orient. error (degrees)	0.88	0.52	0.39	0.86

Table 3.5 – RMS values and maximum errors obtained in the 4 raster scans.

The resulting values of the error were considered a good starting point for future improvements. The RMS of the position error is always smaller than the tool-path step for speeds above 10 mm/sec and slightly bigger for 10 mm/sec. The RMS of the orientation error is very small and under 5 hundredths of a degree for the smallest speed. The speed profile used to acquire the experimental data was uniform. This explains why the maximum deviations are much larger than the RMS deviations; the command tool paths were not optimized in terms of speed. The maximum orientation error is also presenting an outlier value for the slowest speed, where the error happens to be larger than the errors found at the higher speeds. An optimization of the path around the critical areas, through the introduction of acceleration and deceleration ramps, can smooth the tool-paths and bring an improvement of the accuracy.

3.6.2. Independent positional accuracy measurements

Following the results coming from the comparison between command and feedback coordinates, it was deemed useful to check the positional accuracy of the robotic arm with an external system. A method to track the position of the robot end-effector was selected. The LEICA laser tracker available at UoS was used. Its features are reported in Table 3.6 [30]. Since the system has an acquisition frequency rate of 1000 Hz, it is suitable to measure the accuracy of the robot in a continuous motion, being able to acquire the position at least once in a KRC interpolation cycle. Moreover, it is possible to implement the control of the LEICA system in the same C# code managing the robot motion control.

However, it was deemed useful to start the development of such integrated metrology system looking at the accuracy measurement for a discontinuous motion.

	Laser Tracker
Accuracy	ADM:±10μm IFM:±0.2μm+0.15μm/m
Volume Coverage	Radius: 80 m θ hoz: 360° θ ver: ±45°
Update Rate (Hz)	1000

Table 3.6 – Features of LEICA Absolute Tracker AT901 at UoS.

After having generated a suitable file of command coordinates, the C# application controlled both the robot motion and the LEICA system. The task was to follow a large 3D raster tool-path within the working envelope of the KR5arcHW robotic arm, triggering the LEICA system at each stop position to acquire and store the positional data. The robotic arm travelled a distance of 25 m within its working space, interrupting the continuous movement every 49.2 mm along the path. A reflector was attached to the robotic arm, in order to allow the LEICA system to track the motion of the robot. Figure 3.25 shows the head of the LEICA system tracking the position of the reflector, attached to the robotic arm in the background.

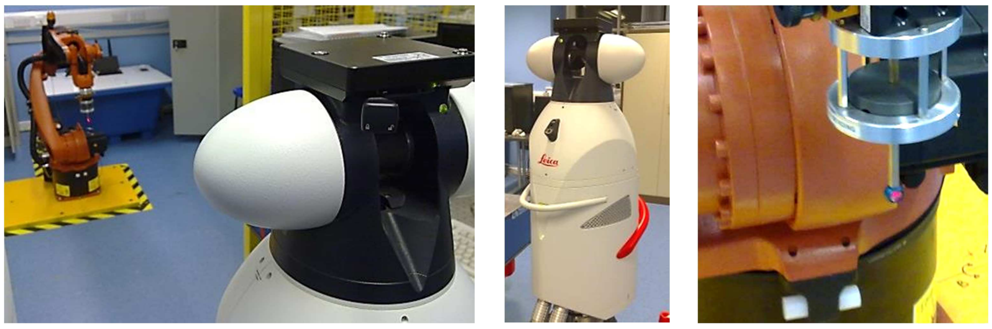


Figure 3.25 – a) Laser tracking of the robot position, b) LEICA laser tracker, c) Reflector.

The total number of collected points was 505. At the end of the acquisition, there are three clouds of points: the first cloud contains the desired points (the commanded stop positions), the second is derived by the robot's feedback data and the third is generated by the positions acquired by the external tracking system. Since the KUKA Robot and the LEICA tracker have different coordinate reference systems, the cloud of points acquired by LEICA was translated and rotated before calculating the distance between the points (Figure 3.26).

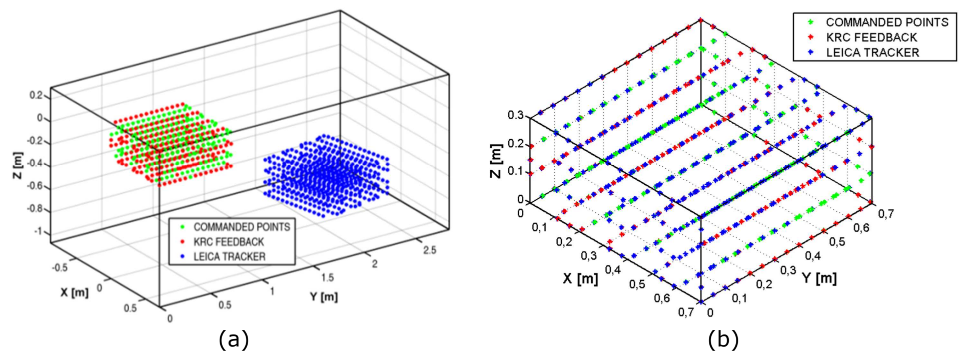
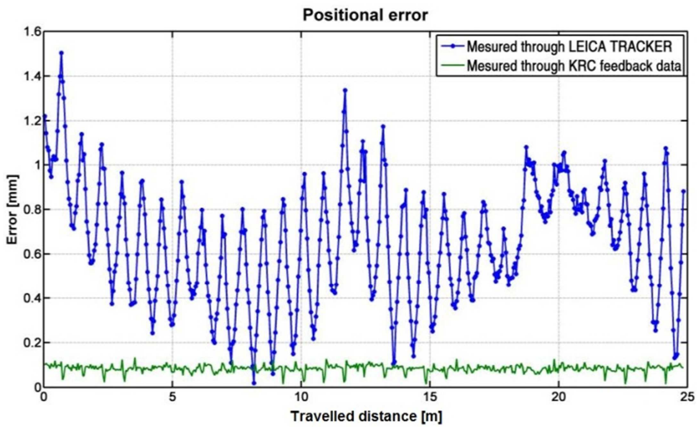


Figure 3.26 – Original clouds (a); clouds after the roto-translation of the LEICA's dataset (b).

Figure 3.27 shows a plot of the error along the travelled distance. Both the error estimated by the KRC feedback and the other evaluated by the laser tracker are presented.



	LEICA	KUKA
Mean error (mm)	0.65	0.08
Max error (mm)	1.50	0.14

Figure 3.27 – Positional error evaluated with the LEICA tracker and the KRC feedback.

The error measured by the LEICA tracker is much bigger than the error estimated by the KRC feedback. A possible cause for the bigger error reported by the laser tracker was found in the manual approach to calibrate the position of the reflector. The centre of the reflector was taught to the robot through calibrating the tip of a rod that was assumed to be in the same position of the reflector centre.

A simple test demonstrated how the error in the manual calibration of the tool central point (TCP) the main source of the deviation between the error ranges in Figure 3.27. The robotic arm was externally controlled to move the reflector around the calibrated TCP. Three rotations were carried out around the x, y and z-axis. During the three rotations, the position of the TCP was monitored by the LEICA tracker. Figure 3.28 can clarify the types of motion that were executed.

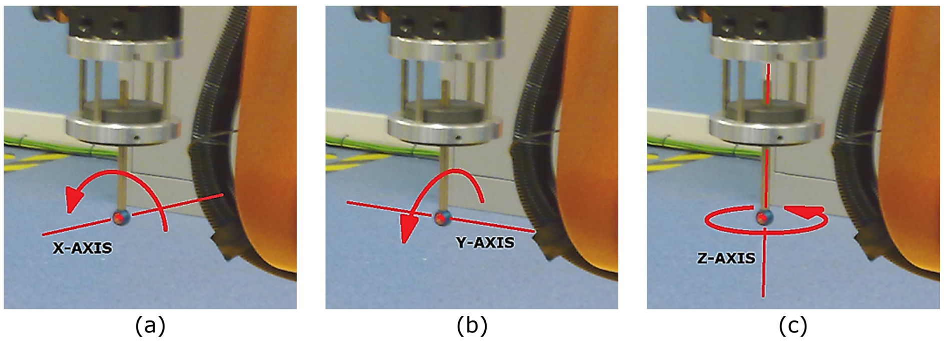


Figure 3.28 – Rotation of the TCP around the three Cartesian axes.

The reflector was rotated by 270 degrees around x-axis and 30 degrees around y and z-axis. The positions acquired by the laser tracker during the rotations were plotted in a 3D Cartesian plot. If the centre of the reflector was calibrated accurately, small errors deviations would be presented around the origin of the plot. However large clouds of points are shown in the plot of Figure 3.29.

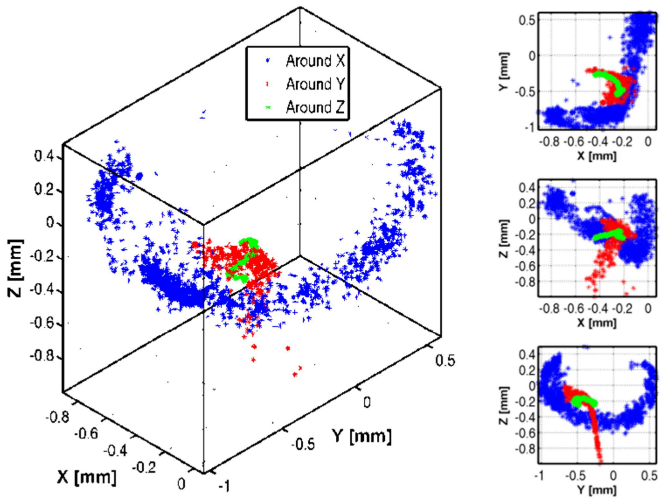


Figure 3.29 – 3D plot of the points acquired during three rotations around x, y and z-axis.

The previous point clouds suggest interesting elements for discussion. Calculating the average distance between the origin of the Cartesian plot and each point of the clouds, the following values are obtained for the three rotations: 0.808, 0.612 and 0.553 mm. The overall average value is equal to 0.658 mm, almost equal to the mean error related to the laser tracker in Figure 3.27. It demonstrates the origin of the error being caused by the imprecise teaching of the TCP.

The development of a method based on optical measurements, for correcting the calibration of the tool, would allow a consistent reduction of the error measured by the laser tracker and a good agreement with the positional deviation obtained from the KRC feedback. An algorithm may be developed, through further investigations, in order to use the laser tracker data to calibrate the centre of the reflector and consequently the TCP of any other robot tool. To reduce the chances of positioning error, the necessity of calibrating the robot precisely using a laser system was also presented by some authors at the International Symposium on NDT in Aerospace [31], in 2012.

3.7. Enabling external control for KRC4 robots

The previous sections of this chapter explained how external control was deployed for the KRC2 robots at UoS, achieving continuous and fluid robot tool-paths. A stand-alone C# application allowed Ethernet communication between a robotic arm and a server computer; through the communication socket it was possible to receive robots current positions and command the desired motion. A schematic representation of the external control system is given in Figure 3.30.

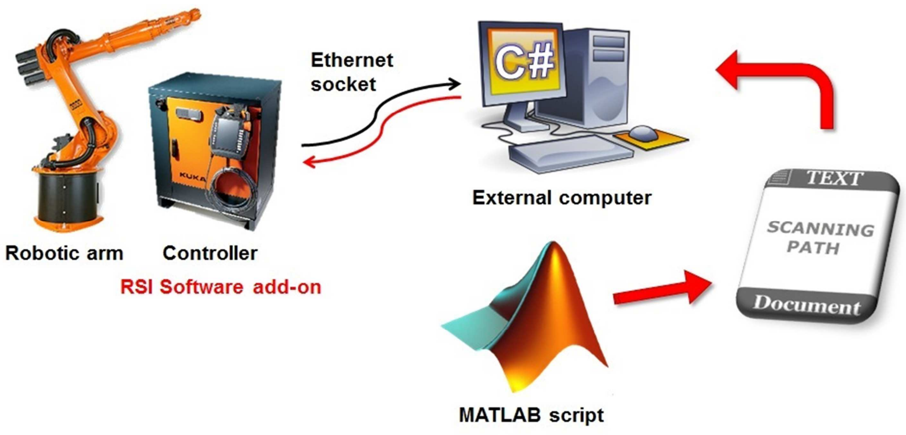


Figure 3.30 – Schematic representation of the external control system.

This section describes the necessary modifications of the C# code, which were applied to make suitable operation with the KRC4 controllers (for the KR16 L6-2 robots at TWI).

3.7.1. Upgrading the C# server application

The version of the RSI Software add-on for KRC2 controllers supports both Transmission Control Protocol (TCP/IP) and User Datagram Protocol (UDP) [14]. These two protocols are similar but have some important differences [13]. TCP/IP had been selected and used for the Strathclyde system. It was the preferred option due to the increased robustness of the communication.

Unfortunately, TCP/IP is not supported by the new RSI version for KRC4 controllers and UDP protocol becomes the only available alternative [29]. The configuration of the KRL module to enable RSI double-direction communication for KRC4 robots is described in Appendix A3.

It was also necessary to upgrade the UoS C# code to make it able to manage the movement of KRC4 robots by external control. Despite the different Ethernet connection protocol used, the resulting code spawned the same actions of the code as developed for the KRC2 controllers. The performances of the external control application were examined and they were optimized through conversion of the C# code to a C++ script. The reasons behind this are described in Appendix A4.

3.7.2. Testing the KRC4 system

Section 3.6.1 described how the comparison between the target coordinates and the received coordinates was carried out for the KRC2 robots at UoS. Two error types were analysed: the position error (the distance between the target point and the real point in the Cartesian space) and the orientation error (the angle between the vector of the target normal direction and the vector of the obtained direction). For KRC2 controllers the minimum error was found at 7 RSI signal-processing cycles from the interpolation cycles that sent the target coordinates.

The same investigation was carried out with the KRC4 robots at TWI. The same tool-paths used at Strathclyde for speeds of 50, 20 and 10 mm/sec were used at TWI. The minimum errors were always found at 5 interpolation cycles after the cycles that sent the command coordinates. Using Eq. 3.10 and equating it to:

$$Deadtime_{KRC4} = 5 \cdot 12ms = 60ms \qquad \text{Eq. 3.13}$$

The values of $FILTER for the KR16 L6-2 robots at TWI is equal to:

$$\$FILTER_{KR16\ L6-2} = 48 \qquad \text{Eq. 3.14}$$

The real value of the filter variable stored within the robot controllers was checked and it validated this result.

Figure 3.31 gives a graphical representation of the system accuracy in the three raster tool-paths. The position and the orientation errors are shown in three-dimensional scatter plots, giving evidence of the most critical areas.

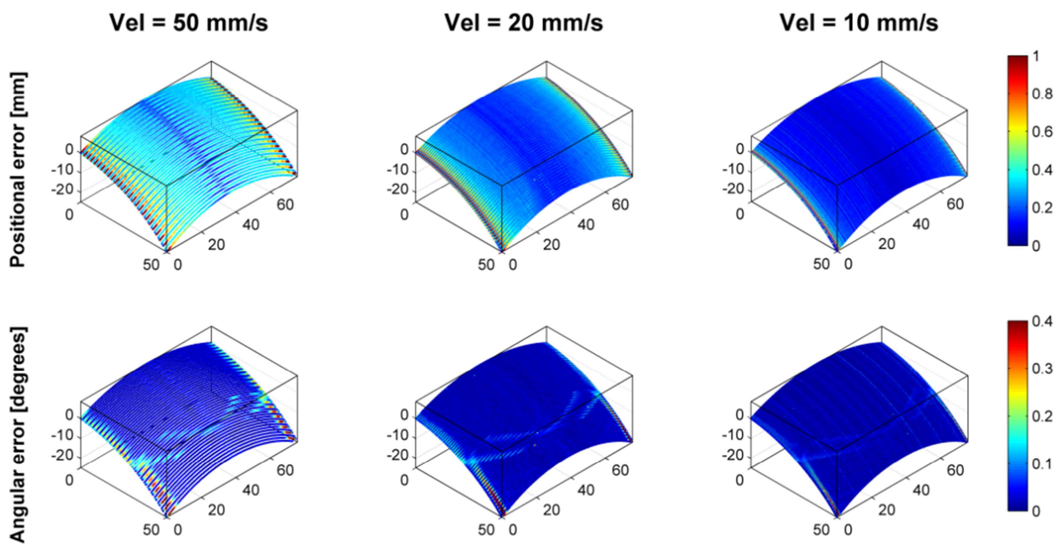


Figure 3.31 – Position and orientation error maps of the raster tool-path.

The values of the maximum errors and the RMS values, obtained with the KR16 L6-2 robot (KRC4 controller) at TWI, are reported in Table 3.7 and compared with the respective values obtained at Strathclyde with the KR5arcHW robot (KRC2 controller).

	Speed = 50mm/sec Step = 1 mm		Speed = 20mm/sec Step = 0.5 mm		Speed = 10mm/sec Step = 0.25 mm	
	KRC2	KRC4	KRC2	KRC4	KRC2	KRC4
RMS position error (mm)	0.52	0.39	0.38	0.29	0.27	0.17
RMS orient. error (degrees)	0.17	0.07	0.08	0.04	0.04	0.03
Max position error (mm)	1.97	1.25	1.84	1.06	1.71	0.89
Max orient. error (degrees)	0.52	0.71	0.39	0.51	0.86	0.46

Table 3.7 – Maximum errors and the RMS values.

The values of the error show that the KR16 L6-2 robot at TWI is generally more accurate than the KR5arcHW robot at Strathclyde. The RMS of the position and angular error and the maximum positional error of the KRC4 robot are always smaller than the relative values obtained with KRC2 robot. Although the maximum orientation error is bigger for 50 mm/sec and 20 mm/sec, the RMS of the orientation error is usually very small and always below 7 hundredths of a degree.

3.8. Interfacing the ultrasonic acquisition system

Whereas the previous part of the chapter covered the strategies to control the motion of industrial robots, presenting the novel approach based on the external control, this section focuses on the problem of interfacing the ultrasonic instrument to enable NDT data acquisition.

Many probes and sensors can be attached to the end effector of a robotic arm and considered like specific tools, after teaching their geometric sizes. The difficulty of pursuing a final useful automated inspection system is hidden in the task of matching the sensor data with the accurate position of acquisition; this is a fundamental requirement and a key output of this thesis.

Unlike other automated manipulators using encoders (e.g. gantries), recent works [31] have underlined the difficulty of identifying probe position in real time when using a 6-axis industrial robot. Current industrial robots can inform about their position only in regular time intervals lasting some milliseconds, so data concerning the probe's true position can only be captured at limited frequency rates.

A common idea is interpolation. With interpolation it is possible to create virtual positions in between the known positions. The interpolation algorithms are relatively easy to develop and set up. The idea consists in time-stamping both the position information received from the robot controller and the signals received by the acquisition system. Since the sensor data and the robot positions come from two independent controllers (the sensor data acquisition device and the robot controller), they arrive to the user (usually a server application running on a third system) at different rates and without any sort of synchronization. It is necessary to interpolate the position patterns through the time of reception of each sensor signal, in order to evaluate the position of its registration.

A simple concept, like the aforementioned interpolation, can lead to some inconveniences if it is not carefully developed. Each controller may time-stamp every outgoing data packet with its own instantaneous time. It might be possible to think it is sufficient to record the time when the server application starts receiving position information and sensor signals and rely on the time-stamp of each packet to calculate the elapsed time. This would lead to the necessity of fulfilling two conditions:

- The signal acquisition frequency must be constant;
- The robot's probe speed must be constant.

The acquisition frequency can be controlled quite well on most systems. However the probe speed can be considered as constant in only two cases:

- Outside the acceleration and deceleration phases (the scanning area must therefore be extended beyond the zone that needs to be inspected);
- In the zones where the changes in direction are not too extreme. However, robotic arms are designed to be able to follow complex trajectories. By their

very nature, complex-geometry parts have zones that require significant changes in path direction.

The author of this thesis investigated the problem and worked together with TWI software engineers, in order to develop a reliable method for accurate matching of NDT data and collection positions.

The aim was to integrate a MicroPulse 5PA phased array ultrasonic system (designed by PeakNDT) [32] with KRC16 L6-2 KUKA robots. The MicroPulse 5PA is an advanced ultrasonic instrument that is available with both conventional and phased array channels. It also has 4 axes of position encoder inputs and can be remotely controlled via a high-speed Ethernet connection. These features have made it popular for field inspections, using both crawlers and hand-held probes. It is also widely used for in-process inspections. Figure 3.32 summarizes the features of the MicroPulse 5PA.

Features	
General	25 MHz bandwidth. Low noise receiver, many filtering options. 100 MHz sampling at 12 bits vertical resolution. 4-Axis encoder inputs. Enables high-speed scanning in pulse-on-position mode or position stamped mode.
Conventional	Square-Wave 300 V pulser for optimum transducer excitation 16 channels, expandable up to 64 channels
Phased Array	32/32 64/64 or 128/128 channels. All channels may be used for beam forming Dynamic Depth Focusing Apodization.

Figure 3.32 – Picture and features of the MicroPulse 5PA.

Several attempts were carried out to develop a C# server application, running on a server machine connected to the MicroPulse 5PA and to the robot controller via Ethernet sockets. The server application has to be capable of simultaneously receiving ultrasonic phased array data and the position feedback from the robotic arm, time-stamp both in a congruous way and save all data to output files for successive post-processing phase.

The choice of the time-stamping method is crucially important. Every ultrasonic signal has to be accurately associated to the relative collection point in 3D space. The first attempt was based on the following conceptual flow scheme (see Figure 3.33).

When the C# application is launched, it firstly initializes the counter of the elapsed time (t). The time is measured in milliseconds. The first reception of data from the KUKA Robot Controller (KRC) marks the starting time (ts). Since that time instant, the robot position is time-stamped with time *t_KRC* (initially set equal to ts + 4ms) and incremented by 4ms at each new received position. This assumes the cycle time of the robot controller being constant and exactly equal to 4ms. Each ultrasound signal is instead time-stamped with the elapsed time (t) at the moment when the command is sent to the MicroPulse 5PA to fire the probe. The robot positions are interpolated according to time-stamps during the post-processing procedure, in order to estimate the exact position of collection of each A-Scan signal.

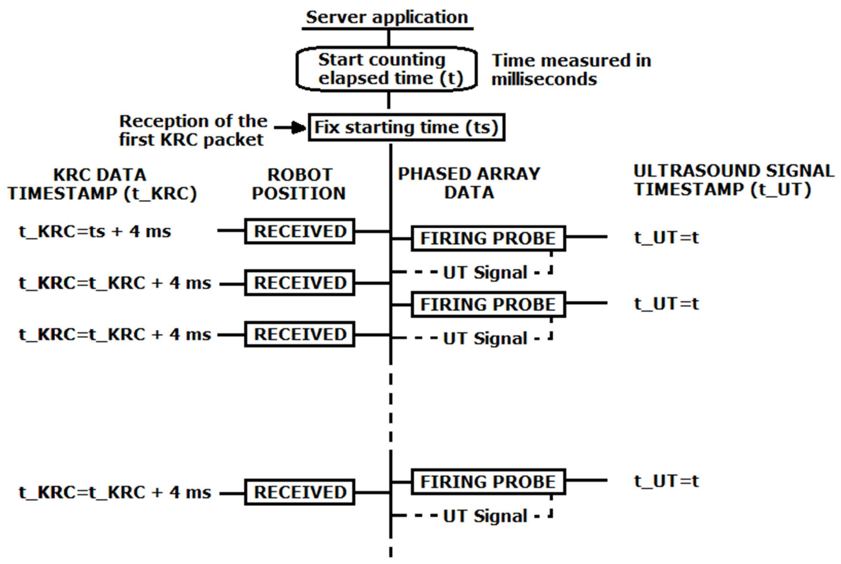


Figure 3.33 – Conceptual scheme of the first time-stamping method.

The quality of the C-Scans, obtained through using this approach, was quite poor. The ultrasound data appeared to be badly encoded. This issue was considered to be caused by wrong time-stamping approach rather than to an interpolation error. Since the robot speed can be assumed constant during each cycle of the robot controller (lasting 4ms), the interpolation of the positional data results in a linear law and it is quite easy to program.

Significant improvements were obtained through developing another approach to time-stamp the incoming ultrasound and positional data. The conceptual scheme of the new time-stamping method is shown in Figure 3.34. Both the positional data and A-Scan signals are time-stamped with the elapsed time (t). Moreover, the time is measured in “ticks” rather than in milliseconds. It was found that a “tick” is the smallest time unit available in the C# language. There is no correspondence between real time and “ticks”. The ticks are machine dependent; however, since both robot position and A-Scan are time-stamped with the same clock, every time unit (even machine dependent) is a congruent unit for measure of time.

For the computer used for our development work, the number of ticks per millisecond was equal to 2435.66 ticks/ms; it gives an idea of how using "ticks" is far more accurate than using milliseconds for accurate measure of elapsed time.

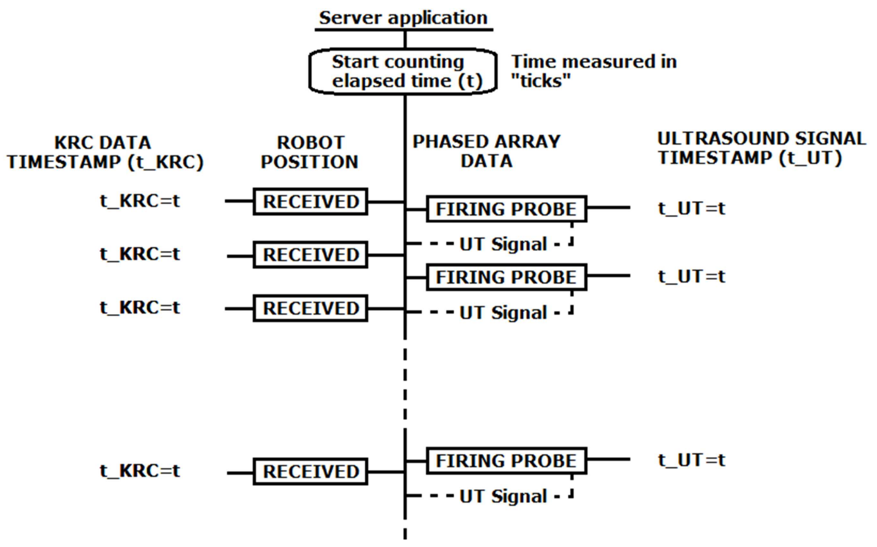


Figure 3.34 – Conceptual scheme of the latest time-stamping method.

Four metallic plates were immersed in a tank filled with water, held at different levels above a flat surface. A raster scan path was used to manipulate the phased array probe, whilst scanning a large rectangular area of 500 x 500 mm (Figure 3.35).

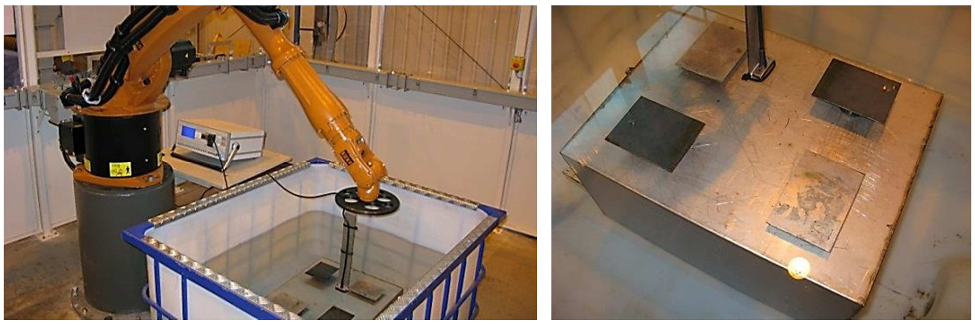


Figure 3.35 – Experimental setup

Figure 3.36 shows the Time of Flight (TOF) C-Scan obtained with the first and the second time-stamping method. It is evident how the first method produced a poor matching between ultrasound signals and positions. Compared to the first time-stamping approach, the edges of the plates are quite well defined using the second approach.

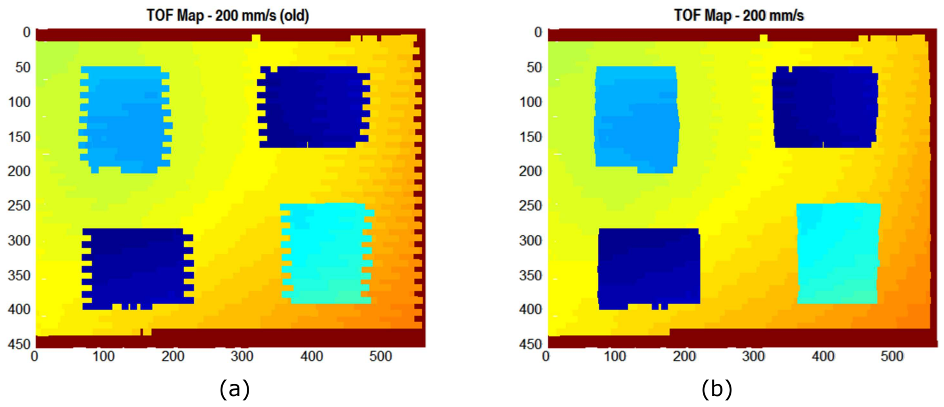


Figure 3.36 – TOF maps obtained with the first (a) and the latest time-stamping method (b).

Further investigations demonstrated the failure of the first time-stamping method being related to the inconstancy of the robot controller cycle time. It is expected to last exactly 4ms for KRC4 robot controllers, but it was seen that this is just an average value. It spans between 3.4ms and 4.7ms as showed by Figure 3.37. Despite of the fact the IPOC number is incremented by 4 at each cycle, assuming a constant cycle time of 4ms is not possible.

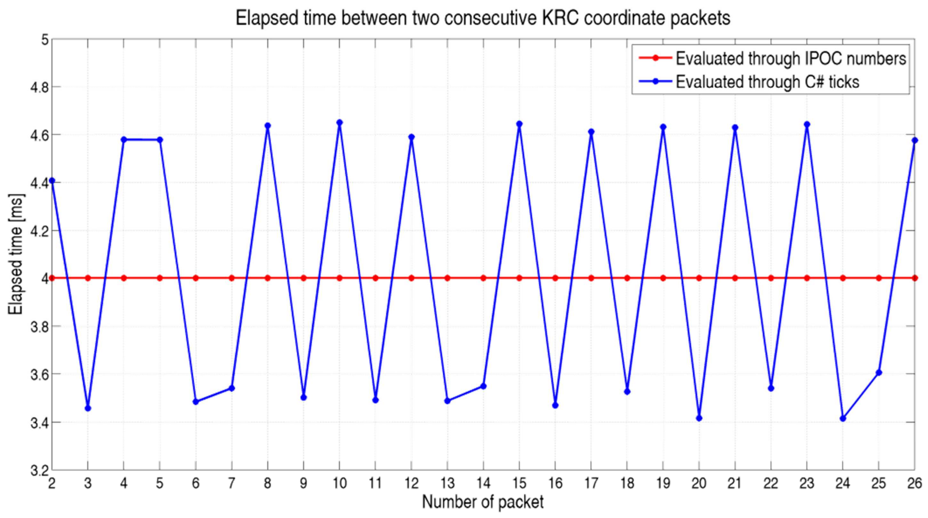


Figure 3.37 – Elapsed time between consecutive robot controller cycles.

Figure 3.38 compares three TOF C-Scans of the same area, obtained using different scanning velocities and the second time-stamping algorithm. The focal law was set to produce an unfocused beam from a 32 element - 5.0MHz - 1mm pitch phased array probe. A 10-element sub-aperture was used.

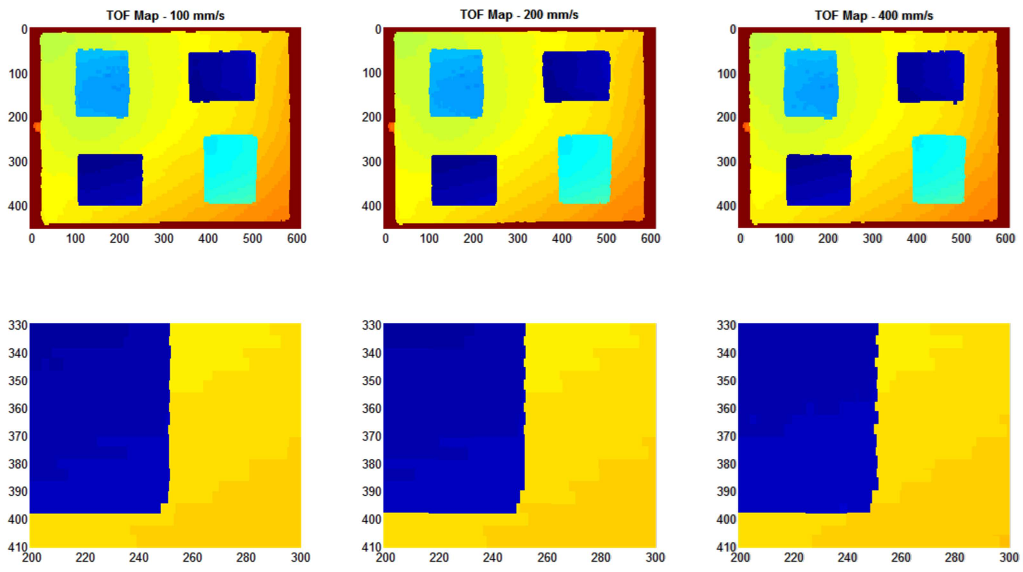


Figure 3.38 – Raster scans obtained at 100, 200 and 400 mm/s. The bottom pictures show the same zoomed area for each C-Scan. Scales are in millimetres.

Figure 3.39 shows the same set of C-Scans obtained sweeping the ultrasound beam along the whole active axis of the phased array probe. Each B-Scan consists of 23 A-Scans.

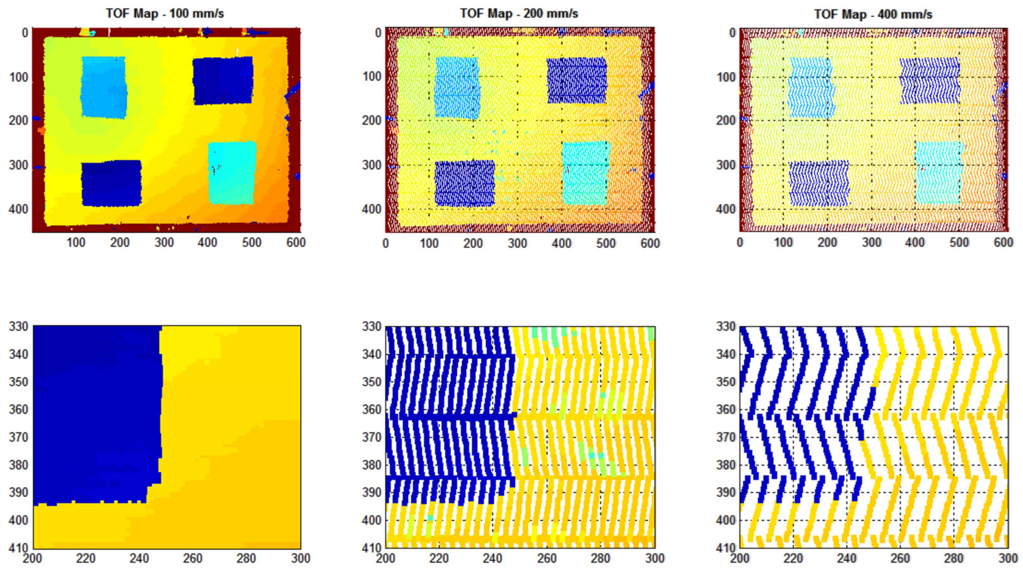


Figure 3.39 – Raster scans obtained firing 23 blocks of elements along the phased array probe with pitch of 1 mm. The robotic arm moved with speeds of 100, 200 and 400 mm/s.

The second encoding algorithm ignores the robot time-stamp (the IPOC – Interpolation Counter), contained in each received position data packet. Both the position information and the ultrasound signal are time-stamped with the local time of the system running the server application, expressed in a machine dependent unit (ticks). Using the smallest unit ("tick") related to the internal clock of the system it is possible to go beyond the aforementioned limitations found in previous reported work [31]. Ticks are dependent on the hardware hosting the server application; however more than 1000 ticks per millisecond are usually generated by modern processors. It enables the possibility of enhancing the encoding performances. This allows significant changes in the robot end effector speed and in the sampling rate of the ultrasound data throughout the scan, without affecting the matching performance between ultrasonic A-Scans and interpolated coordinates.

The aforementioned experimental tests validated the time-stamping method. Further tests were carried out through scanning a carbon fibre composite sample with a flat front surface and some reinforcements (stringers and ribs) on the back surface (see Figure 3.40). The focal law was set to sweep the ultrasound beam along the whole active axis of the phased array probe (the 32 elements - 5.0 MHz - 1 mm pitch phased array probe).

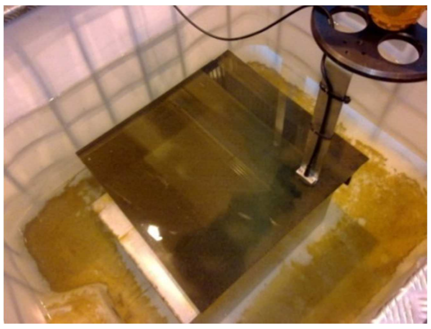
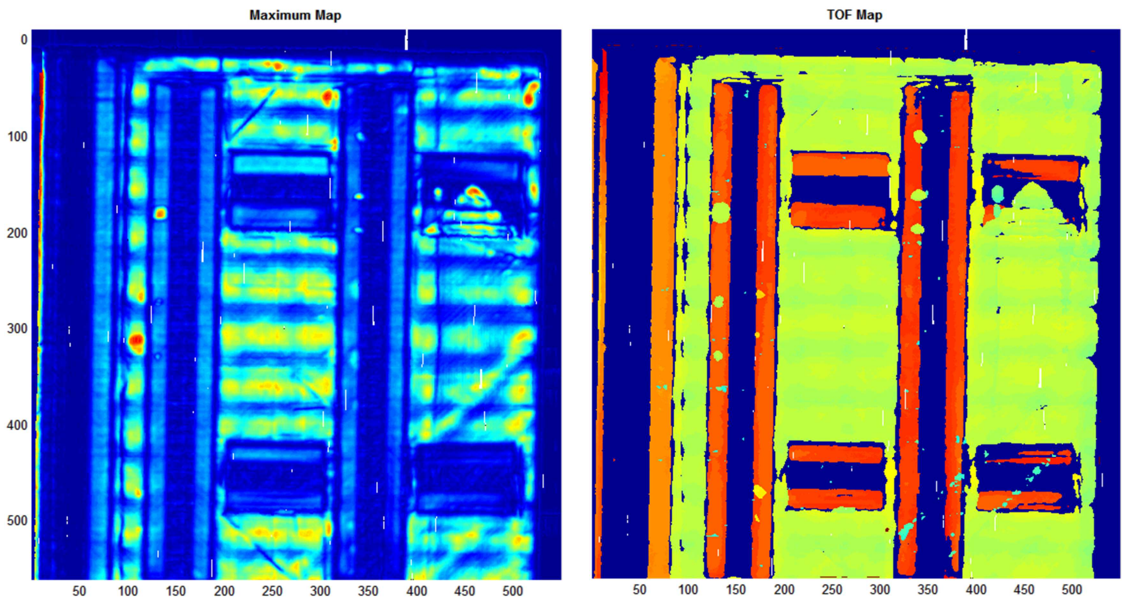


Figure 3.40 – Carbon fibre composite sample during the scan.

Figure 3.41 shows the C-Scan of the sample. Artificial defects are clearly visible in the time of flight map.



(a) (b)

Figure 3.41 – C-Scan of the composite sample: Maximum amplitude map (a) and TOF map (b).

3.9. Encoding UT data acquired from curved surfaces

The robot controller transmits the position of the Tool Central Point (TCP) during the execution of a scan. The X, Y and Z coordinate are sufficient to encode ultrasonic signals coming from single-element (non-phased array) probes that emit a single ultrasound beam.

In contrast, phased array probes generate sweeping or steering beams. The beam source and direction are controllable because a phased array probe is made up of multiple small elements, each of which can be pulsed individually at a computer-calculated timing. It increases the complexity for encoding ultrasonic data, since the robotic Tool Central Point of a phased array probe is rarely coincident with the focal point of the ultrasound beam.

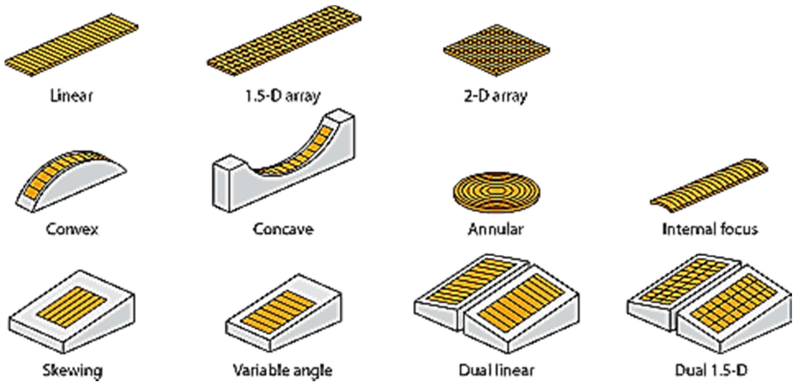


Figure 3.42 – Types of phased array probe geometries [33].

A wide variety of phased array probe geometries is emerging, with arrays spanning in one, two or three dimensions according to linear, circular or angled patterns. Different types of probes are shown in Figure 3.42.

The robot controller transmits the TCP coordinates relatively to its active reference system. If the orientation of the probe does not change during the scan and the directions of the probe axes (active, passive and normal axis) are equal to the directions of the robot reference system, the encoding problem is very much simplified. This leads to the schematic representation given in Figure 3.43a. Assuming Xt and Yt respectively corresponding to the active and the passive axis of a linear phased array probe and Zt the normal to the first two directions, the orientation of the active axis Xt relatively to the robot reference system can be determined from the axes Xr, Yr and Zr. In this case it is sufficient to use the current coordinates of the TCP and the number of scan during a B-scan (i.e. the group of fired elements in the array for a certain signal during the B-scan sweep). In a linear phased array probe, with the active axis in the Xt direction, it is easy to work out the offset value to associate to the Xr component of the TCP coordinate. The Yr and the Zr component remain constant for encoding all signals of the B-scan. This was the straightforward approach used during our early inspections of flat surfaces. An example is given by the scan shown in Figure 3.41.

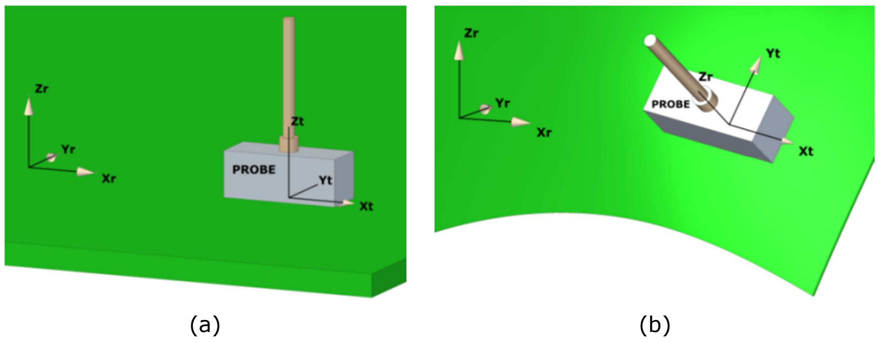


Figure 3.43 – Probe manipulated over a flat surface with constant orientation of its axes (a) and probe inspecting a curved surface maintaining a perpendicular direction to the surface (b).

When the probe is manipulated to follow a curved surface, the tool reference system is constantly changing its orientation relatively to the robot reference system, as it is shown in Figure 3.43b. In this case it is necessary to develop a more complex encoding algorithm. The robots return six positional components, the first three of which are the Cartesian coordinates of the tool point: X, Y and Z. The final three components are named A, B and C and give the rotation relatively to the Z, Y and X axis. It is necessary to use these rotation components to work out what the direction of the phased array active axis is for each signal collected during the scan.

The following describes the mathematical formulation that was implemented into the acquisition software to encode signals acquired by linear phased array probes. The formulation was parameterized to allow using probes with different size, pitch and number of elements. The direction of the active axis relatively to the tool reference system is given by the vector:

$$\mathrm{probeTransform} = \{\mathrm{xa} \quad \mathrm{ya} \quad \mathrm{za}\} \qquad \text{Eq. 3.15}$$

for the probe shown in Figure 3.43a it is equal to:

$$\mathrm{probeTransform} = \{\pm 1 \quad 0 \quad 0\} \qquad \text{Eq. 3.16}$$

The cosine and sine values of the rotation coordinates (A, B and C) are herein indicated with cosA, cosB, cosC and sinA, sinB, sinC. The relations between the rotational matrices ($\mathrm{R_X}$, $\mathrm{R_Y}$ and $\mathrm{R_Z}$) and the angles A, B and C are given in Appendix A5.

Rotational operations are not commutative, i.e. performing the rotations in a different order will give a different result, so defining the correct order is very important when interpreting the final three coordinates. The correct transformation, for the tool reference system in relation to the robot reference system, consists in rotating firstly around Z, then around Y and finally around X. The mathematical formulation is given by the following matrix [34]:

$$\mathrm{R} = \mathrm{R_Z R_Y R_X} = \begin{bmatrix} \mathrm{cosA \cdot cosB} & \mathrm{cosA \cdot sinB \cdot sinC - sinA \cdot sinC} & \mathrm{cosA \cdot sinB \cdot cosC + sinA \cdot sinC} \\ \mathrm{sinA \cdot cosB} & \mathrm{sinA \cdot sinB \cdot sinC + cosA \cdot cosC} & \mathrm{sinA \cdot sinB \cdot cosC - cosA \cdot sinC} \\ \mathrm{-sinA} & \mathrm{cosB \cdot sinC} & \mathrm{cosB \cdot cosC} \end{bmatrix} \qquad \text{Eq. 3.17}$$

Thus the offset vector is:

$$\mathrm{offsetVector} = \mathrm{R} * \mathrm{probeTransform}' \qquad \text{Eq. 3.18}$$

from where:

$$\mathrm{offsetVector(1)} = \mathrm{(cosA \cdot cosB) \cdot probeTransform(1)} + \mathrm{(cosA \cdot sinB \cdot sinC - sinA \cdot sinC)} \cdot \mathrm{probeTransform(2)} + \mathrm{(cosA \cdot sinB \cdot cosC + sinA \cdot sinC) \cdot probeTransform(3)} \qquad \text{Eq. 3.19}$$

The final calculation of the encoding coordinates for each signal is:

$$\mathrm{X} = \mathrm{coordPoint}(1) + \mathrm{offsetVector}(1) \cdot \left(\frac{\mathrm{num}-\mathrm{numScan}}{2}\right) \cdot \mathrm{p} \qquad \text{Eq. 3.20}$$

$$\mathrm{Y} = \mathrm{coordPoint}(2) + \mathrm{offsetVector}(2) \cdot \left(\frac{\mathrm{num}-\mathrm{numScan}}{2}\right) \cdot \mathrm{p} \qquad \text{Eq. 3.21}$$

$$\mathrm{Z} = \mathrm{coordPoint}(3) + \mathrm{offsetVector}(3) \cdot \left(\frac{\mathrm{num}-\mathrm{numScan}}{2}\right) \cdot \mathrm{p} \qquad \text{Eq. 3.22}$$

where *num* is the total number of signals in a B-scan, *numScan* is the current signal number and *p* is the distance between consecutive groups of fired elements in the array (*pitch*).

3.10. Wrapping up

The use of six-axis robots for product inspection and non-destructive testing systems is the central investigation of this thesis.

This chapter gave a basic breakdown of how a 6-axis serial manipulator works, the way it moves and how it is programmed. The key aspects to work with 6-DoF robots were discussed: kinematics, dynamics, robot singularities, robot communication and synchronization, programming, control, accuracy and repeatability.

The robotic hardware used for the development of the present thesis was described in Section 3.3. Two different robot control strategies for the robotic NDT of simple geometries (e.g. small flat components) were discussed. Relevant testing outcomes were presented.

A new method, established for external real-time control of robot manipulators, is presented. The viability of the new control method was proved by testing carried out on both KUKA KRC2 and KRC4 robots. Accuracy measurements were obtained through comparing target positions with feedback positions and through laser tracking of the robot end-effector.

The external control paves the way to the development of new advantageous approaches to integrate robot manipulators with NDT acquisition systems for automated NDT inspections. The deployment of the new robot control strategy and its inherent advantages are discussed in the following chapters.

The chapter ends presenting the integration of an ultrasonic phased-array instrument for automatic NDT with 6-axis manipulators. The mathematical formulation to encode the NDT data through the interpolated robot feedback positions is also described.

References

[1] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*: Richard Paul, 1981.

[2] M. Hayes, M. Husty, and P. Zsombor-Murray, "Singular configurations of wrist-partitioned 6R serial robots: A geometric perspective for users," *Transactions of the Canadian Society for Mechanical Engineering,* vol. 26, pp. 41-55, 2002.

[3] A. Jorge, "Fundamentals of robotic mechanical systems, Theory, Methods and Algorithms," ed: Springer-Verlag NY USA, 1997.

[4] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*: Springer Science & Business Media, 2000.

[5] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB* vol. 73: Springer Science & Business Media, 2011.

[6] E. ISO, "Manipulating Industrial Robots–Performance Criteria and Related Test Methods," ed: European Committee for Standardization Geneva, 1998.

[7] H. A. Youssef, H. A. El-Hofy, and M. H. Ahmed, *Manufacturing Technology: Materials, Processes, and Equipment*: CRC Press, 2011.

[8] R. Bernhardt, R. Bernhardt, and S. Albright, *Robot calibration*: Springer Science & Business Media, 1993.

[9] S. Aoyagi, *Selection of optimal measuring points using genetic algorithm in the process to calibrate robot kinematic parameters*: INTECH Open Access Publisher, 2012.

[10] K. L. Conrad, P. S. Shiakolas, and T. Yih, "Robotic calibration issues: Accuracy, repeatability and calibration," in *Proceedings of the 8th Mediterranean Conference on Control and Automation (MED2000), Rio, Patras, Greece*, 2000.

[11] T. Brogårdh, "Present and future robot control development—An industrial perspective," *Annual Reviews in Control,* vol. 31, pp. 69-79, 2007.

[12] KUKA, "KUKA FastSendDriver 4.0 manual," 2014.

[13] G. Xylomenos and G. C. Polyzos, "TCP and UDP performance over a wireless LAN," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, pp. 439-446.

[14] KUKA, *KUKA.RobotSensorInterface 2.3 Documentation - Version: KST_RSI_2.3_V1_en*, 2009.

[15] KUKA, *KUKA.RobotSensorInterface 3.1 Documentation - Version: KST_RSI_3.1_V1_en*, 2010.

[16] KUKA, *KRC2 / KRC3 Expert Programming*. Augsburg: KUKA Roboter GmbH, 2003.

[17] J. Braumann and S. Brell-Cokcan, "Parametric Robot Control," *Integrated CAD/CAM for Architectural Design,* pp. 242-251, 2011.

[18] K. Takase, R. P. Paul, and E. J. Berg, "A structured approach to robot programming and teaching," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 11, pp. 274-289, 1981.

[19] F. Caccavale and M. Uchiyama, "Cooperative manipulators," in *Springer Handbook of Robotics*, ed: Springer, 2008, pp. 701-718.

[20] J. Gudiño-Lau and M. A. Arteaga, "Dynamic model and simulation of cooperative robots: a case study," *Robotica,* vol. 23, pp. 615-624, 2005.

[21] S. Anton, "Integrating multiple robot-vision systems in intelligent assembly/disassembly structures," PhD thesis, UPB Automatics and Industrial Informatics Department, 2008.

[22] F. Anton and S. Anton, "Synchronizing Robot Motions in Cooperative Tasks," *Journal of Control Engineering and Applied Informatics,* vol. 13, pp. 43-48, 2011.

[23] KUKA, "KUKA Roboteam 1.0 manual," 2012.

[24] KUKA. *KR5 arc HW Specifications*. Available: http://www.kuka-robotics.com - Accessed 12/8/2012

[25] KUKA. *KR 16 L6-2 Specifications*. Available: http://www.kuka-robotics.com - Accessed 5/12/2012

[26] Y. GAN, X. DAI, and J. Li, "Cooperative Path Planning and Constraints Analysis for Master-Slave Industrial Robots," *International Journal of Advanced Robotic Systems,* vol. 9, p. 88, 2012.

[27] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio, "Performance comparison of VxWorks, Linux, RTAI and Xenomai in a hard real-time application," in *Real-Time Conference, 2007 15th IEEE-NPSS*, 2007, pp. 1-5.

[28] IEEE, "The Open Group Base Specifications Issue 6, IEEE Std 1003.1, IEEE, Ch2.10," ed, 2004.

[29] KUKA, *KUKA.RobotSensorInterface 3.1 Documentation, Version: KST_RSI_3.1_V1_en*, 2010.

[30] Leica. (2012). *Leica Absolute Tracker AT901*. Available: http://metrology.leica-geosystems.com - Accessed 21/10/2012

[31] P. Louviot, A. Tachattahte, and D. Garnier, "Robotised UT Transmission NDT of Composite Complex Shaped Parts " presented at the NDT in Aerospace, 2012.

[32] Utex. Available: http://www.utex.com (Accessed 26/09/2013)

[33] OLYMPUS, "PHASED ARRAY INSPECTIONS - Phased Array Probes and Wedges," ed.

[34] KUKA. *ARM TUTORIAL 2*. Available: http://www.kuka-robotics.com/en/products/software/ /educational_framework/arm_tutorials/PS_Content_Arm2.htm (Accessed: 20/08/2013)

4. Fundamental considerations for NDT path-planning

4.1. Available solutions

The previous chapter gave an overview of 6-axis robot arms, a breakdown of how they work. Moreover, it presented a new viable strategy to control the motion of 6-DoF robots. Section 3.1.7 was a general introduction to the path-planning problem. The first part of this chapter focuses back on the meaning of *path-planning*, analysing the current available solutions. The chapter continues with the identification of the weaknesses of current systems and the key motivations for the development of new path-planning tools and the presentation of our investigations.

Robotic NDT will inevitably be embedded in the manufacturing processes, in the near future. As manufacturing comes to be more responsive, products with shorter life cycles and smaller batch quantities become commonplace. Therefore robot programming times become critical, and hence an area to be addressed in order to seek improved productivity. Off-line Programming (OLP) is an approach that could reduce the required skill levels of a programmer, reduce the programming times, give the operator a "natural" interface through which the operator would conduct the task in the real world, and reduce the boredom factor [1].

Due to on-line programming having some deficiencies, off-line programming has become popularly used in the industries. The currently available off-line programming software is developed by some CAD/CAM software or provided by the robot manufacturers. Collaborations between software developers and robot manufacturers are emerging; one example is the collaboration between Grasshopper and KUKA that has recently produced KUKA|prc. Table 4.1 shows examples of off-line programming software provided by some software companies and directly by some robot manufacturers.

	Name of company	**Off-line programming software**
Software developers	Cenit	FastSurf
	MasterCam	RobotMaster
	Delcam	PowerMILL
	Delmia	Robotics OLP
Robot manufacturers	ABB	RobotStudio
	Fanuc	RoboGuide
	Motoman	MotoSim
	Kawasaki	PC-Roset
	Adept	Adept DeskTop
	Staubli	Staubli Studio
	KUKA	KUKASim
Collaborations	Grasshopper & KUKA	KUKA|prc

Table 4.1 – Examples of off-line programming software.

The number of industrial robots that have been applied in the manufacturing production and other areas in the past decade is big but significantly smaller than the number predicted at the beginning of the decade [2]. Robot programming is the main problem to blame; barriers include legislation, safety, and the steep learning curve for robot programmers. Programming robots for various tasks in not the least trivial problem and thus the word "off-line programming" catches researcher's eye for further investigation.

Programming includes the generation of algorithms and data. It is possible to outline five objectives of structured programming that need to be considered and fulfilled before any of robot programming is done [3]:

1. Program readability and clarity
2. Increased programmer productivity
3. Reduced cycle time
4. Reliability
5. Maintainability

It is possible to say there are four main categories among the available robot programming techniques [4]:

1. Manual on-line programming – It consists in physically teaching a robot the desired trajectory through interaction with a teach pendant or other similar devices. It is also known as “teach by showing” or “lead through” programming [5].
2. Use of explicit robot programming languages – In this approach the entire motion cycle of the robot is specified explicitly by a computer generated program containing specific commands.
3. Task-level programming – In this approach the robotic arm is commanded using various artificial intelligence techniques, in order to perform certain tasks.
4. Off-line programming – This approach combines computer simulations and graphics to produce a desired trajectory plan without need for physical direct access to the manipulator or its movement.

On-line programming consists in physically moving the robot through the task by an operator. Computer-integrated manufacturing operations require off-line programming and simulation in order to layout production facilities, model and evaluate design concepts, optimize motion of devices, avoid interference and collisions, minimize process cycle times, maximize productivity, and ensure maximum return on investment [6]. Current commercial pieces of software, as those quoted in Table 4.1, provide support for 3D workable layouts including robots, end effectors, fixtures, conveyors, part positioners, and simulations.

Off-line or indirect programming means generating a robot program freed from a robot control or a programming device, remote from the industrial robot's workplace, for instance in central programming development [7]. Programming of the robot for various tasks in production can be done in two ways: either requiring the industrial robots to be mounted in the working environments or by animation using off-line programming methods. Both methods require significant investments but the off-line programming system has the main advantage to enable prediction and avoidance of possible numerous collision situations, before running the final operation with the real system. The need to buy a particular programming package and the corresponding hardware brings the customers to carefully think about the appropriate or suitable software to meet their needs. Time and costs for robot introduction, education and training on the programming systems for the person who will be conducting the robot are also items to be considered. These factors lead to significant costs, expenditure of money and time, especially for small and medium size companies.

The use of off-line programming and simulation systems to program industrial robots enables the possibility to carry out creative programming and optimization, away from production sites. This, thereby, facilitates the reduction of downtimes in production. Furthermore, by using the Off-line programming, a simulation program can reproduce a variety of scenarios and processes for each robot specific task. Thus, this allows optimizing the process without affecting the production flow of the real robotic cell.

Hodges outlines the following advantages offered by off-line programming [8]:

a) Improved robot and programmer safety. The programmer can stay far from potentially hazardous environments.

b) The programmer and the robot are not placed at risk by the accidental operation of the wrong controls, as when the robot is very close to the work piece.

c) Operator does not need to be close to the tool point to achieve the desired accuracy of positioning. Conventional programming places the operator in a vulnerable position inside the working envelope. High-precision tasks require this risk period to be long and continuous, particularly if the program consists of several hundred points. Control mistakes, operator fatigue and errors induced by visual problems such as restricted view and eye strain can easily cause mistakes in the program and poor accuracy.

A large variety of robot types can be programmed from one workstation.

- The system permits verification, assisted by the graphical simulation system, of robot programs in terms of positional data and program logic. This may include input/output signals for control of peripheral equipment, the processing of sensory inputs, path control information and programming structures such as loops, branches and wait instructions, and the inclusion of previously written subroutines.
- In off-line programming the robot program is developed, partially or completely, without requiring direct use of the robot, which can remain in production while its next task is being programmed. This makes the use or robots economic for small batch productions.
- The use of off-line programming enables automated workflows to be planned, built up and modified to suit the desired requirements. This can include the layout of machine tools, materials handling devices such as bowl feeders and

conveyors system, as well as the evaluation of different robots and their individual positioning and work envelope considerations.

- Off-line programming allows data built up in CAD and production control systems to be incorporated into the design of the robot installation.

Given the numerous advantages of off-line to on-line programming, it is clear that the former is worth of being investigated to be applied in the area of robotic NDT.

4.2. Motivations for the investigation of new solutions

The deployment of commercial OLP software in the context of robotic NDT, for the inspection of complex geometry parts, leads to systems with inspection procedures that can be represented by the schema in Figure 4.1.

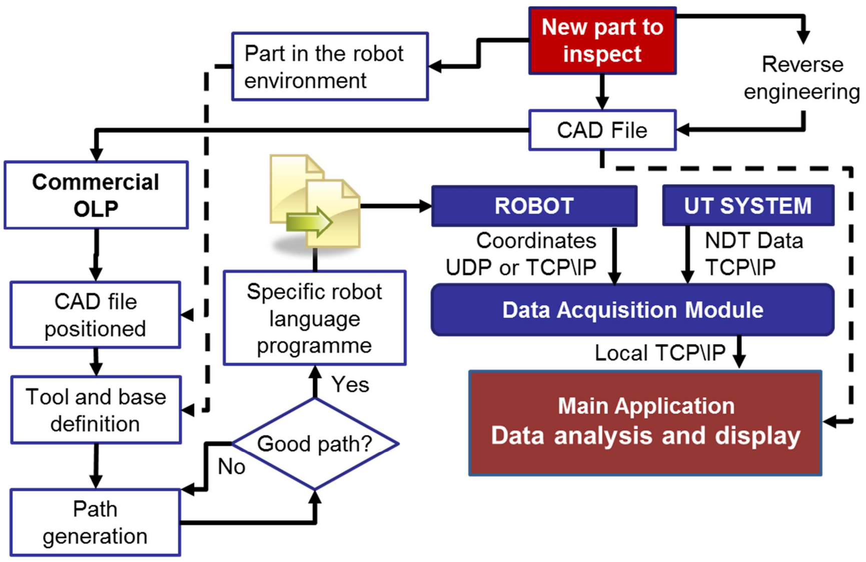


Figure 4.1 – Schematic representation of the robotic inspection procedure, using commercial OLP software.

Using current OLP software to generate tool-paths for NDT purposes is a viable route. It can appear quite straightforward at first inspection; however a series of serious inadequacies appear, when we consider the path-planning task in the context of the whole inspection procedure:

1. Path-planning for automated NDT inspections is a very specific task. As previously mentioned, much commercial software for off-line robot programming draws its origin from the need to use the advantageous flexibility of general robotic manipulators to replace the more traditional and usual machining tools (milling machines, lathes, etc.). As a result, many commercial software applications for off-line programming of robots are

expensive tools, incorporating lots of unnecessary functions for CAD/CAM purposes and machining features. Despite the abundance of functions, a tool-path generated via a CAD/CAM commercial software usually has to be subjected to some amendments, before fulfilling all the requirements for an effective NDT inspection. A number of problems are often present in the original path, being generated by software functions expressly developed for machining and production operations rather than for NDT tasks.

2. Commercially available OLP software is usually designed to optimize the tool-paths for rigid robot tools (e.g. welding guns, cutting and drilling tools). However the NDT tooling is substantially different from manufacturing tools. The NDT techniques are often not expecting any contact between the probes and the part of interest; for example, it is the case of radiographic inspections, laser ultrasound inspections or traditional ultrasonic inspections with water-jet coupling methods. If a long water column is used to guide the ultrasound energy from the probe to the surface, an additional problem arises for the correction of the tool-path to compensate the parabolic deflection of the water column under the gravitational law. Some software is available to fulfill the requirements of the specific applications like spray-painting, but not all NDT non-contact methods can be easily analogized to those applications in order to bend the software to the field of robotic NDT.
3. Commercial OLP always remains separate from the NDT data acquisition software module. The inspection parameters that are fundamental for the generation of suitable tool-paths (e.g. probe standoff, inspection speed, raster step, part position, etc.) have to be manually transferred by the user, with consequent downtimes and lack of automation.
4. Lack of automation is also due to the need of transferring the specific robot language tool-path program file, generated by current OLP, to the robot controller. This is still often accomplished through copying the program file to a USB stick.
5. Significant complications exist when two or more robotic arms need to be synchronized in order to perform a specific NDT inspection. The Ultrasonic Through-Transmission (UTT) technique, for example, uses two transducers: one emitter and one receiver; the receiver being placed on the opposite side of the component and facing the transmitting probe. Many commercial

pieces of software (e.g. Delcam and Mastercam) do not offer any support for co-operating robots. FastSurf, an add-on from CENIT for Delmia (Dassault Systems), allows partial synchronization of robotic movements (e.g. at start or end points of complex paths) but not full synchronization over the complete path, required for the UTT technique.

6. Current OLP software lacks fundamental capability in conditional programming. Typically very specific code is generated for each tool-path, and changes to this path due to changing operation conditions requires re-downloading a complete new path to the robot controller. NDT inspection often requires re-inspection of a particular area of interest of a sample in order to carry out more detailed investigation after an initial screening. This requirement places an additional demand of a more adaptive approach to the path planning that has the provision for conditional modification in response to externally measured data.
7. The possibility to output command coordinates to be used with the external control approach (investigated in Chapter 3) is not enabled by any piece of software to date, with the resultant limitation on true real-time operation.
8. Current software is geared towards manufacturing applications where the task is the production of a specific component. In contrast, the result of an automated NDT inspection is a collection of digital data coming from one or more transducers, and stored in convenient ways onto a hard disk or any other portable memory. These data results are only meaningful when merged with the precise position of acquisition; in other words, it is necessary to record the current position of the robot throughout the whole inspection time in order to encode the NDT data [9]. Modern industrial robotic arms are equipped with encoders monitoring the position of each joint; they can inform about their position at regular time intervals lasting some milliseconds. The captured positions can be further interpolated to extrapolate the probe's location at the collection time of each piece of NDT information. An external computer (separate to the robot controller) is generally required to process the robot positional data and perform synchronization to the externally measured NDT data. If the same computer was used to provide control commands through the same communication socket used to receive feedback, this would lead to better use of resources and higher achievable level of automation. An open-source path-planning

software platform purposely designed for NDT could be embedded within the NDT data acquisition software to provide seamless automated solutions.

9. The deployment of the external control of robot arm, would allow simultaneous management of command and feedback coordinate from one single workstation. This would constitute a pathway to implement real-time path-correction approaches that make the most of data coming from machine vision systems or other type of metrology equipment. The implementation of similar ideas is not made easy by current software.
10. Commercial path-planning software packages are very expensive but not very flexible. Robotics researchers wanting to implement new approaches often struggle with the lack of possibilities that commercial software offers to them. Researchers and universities need robot path-planning research platforms.

4.3. The MATLAB programming environment

The remainder part of this chapter and next chapter present the outcomes of extensive work, undertaken to fulfil the aforementioned motivations and pave the way to the solution of all weaknesses of current software. MATLAB® was selected as the main platform to embrace the challenges and develop new path-planning software solutions. It allows for flexibility of new sections of code (e.g. fluid dynamics, image processing, etc.).

MATLAB is a common platform for the modelling and simulation of various kinds of systems. It is a flexible and extensively used programming software environment for matrix manipulations, plotting of data, implementation of algorithms, and creation of user interfaces. It can also be interfaced with programs written in other languages, including C, C#, C++, Java and FORTRAN. As such MATLAB can be elected as a good choice for programming and simulating robotic systems.

Specific toolboxes (collections of dedicated MATLAB functions) have been developed in the past few years for research and teaching in almost every branch of engineering, such as telecommunications, electronics, aerospace, mechanics and control. As far as robotics is concerned, several toolboxes have been presented for the modelling of robot systems [10-14]. These software tools have been inspired by various application scenarios, such as robot vision [13, 14], and space robotics [11], and have addressed both industrial [12] and academic/educational [10, 13, 14] targets. Off line programming has been investigated by [2] using a combined Simulink/SimMechanics approach.

To date, no specific toolbox has been presented as targeted to the field of robotic NDT inspection; no toolbox has been developed to specifically address the current needs of robotic NDT related to effective tool-path generation, to fulfil the deficiencies of existing off-line programming as outlined above.

4.4. Feasibility studies for new approaches

4.4.1. Command coordinate transfer

The first MATLAB script, which can be included among the investigations for new path-planning approaches, regards simple code developed to allow outputting of command coordinates to be used with the external control of KUKA Robots (introduced in Chapter 3). This script was used to execute the robot tool-path of the accuracy investigations presented in Section 3.5.1 and 3.6.3.

As described in the previous chapter, for the sake of operating the robot through external control, the command coordinates of the desired robot path have to be outputted to a text file through a certain number of coordinate packets. A simple program (a KRL module) runs inside the KUKA Robot Controller (KRC) and is in charge to receive the sequential packets of coordinates sent by a C# application on the external computer. These coordinates say to the robot controller the next point to reach during the scanning motion. A position in a base coordinate system is given by a packet of 6 coordinate (X, Y, Z, A, B, C), where X, Y and Z indicate the position of the Tool Central Point (TCP), A, B and C the orientation of the robot wrist respect to the axes z, y and x.

For each position also the angular coordinates (A, B and C) can be derived from the Cartesian components of the vector normal to the surface of interest. Since the KRC moves the robotic arm to reach the destination position within the interpolation time, the velocity is controlled by changing the distance between the points.

Figure 4.2 shows a metal sample, chosen for testing the script. A portion of its top surface was considered to produce a test raster tool-path. The considered surface was a section of a spherical cup, with radius of 90 mm.

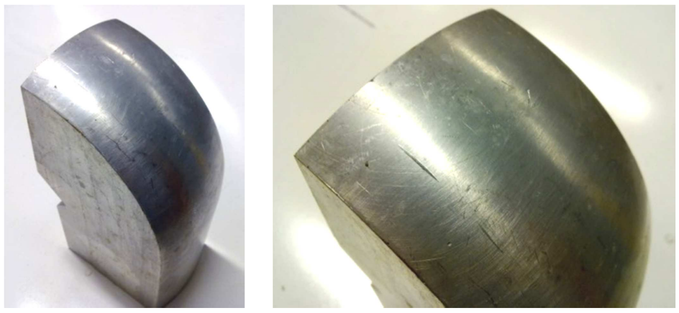
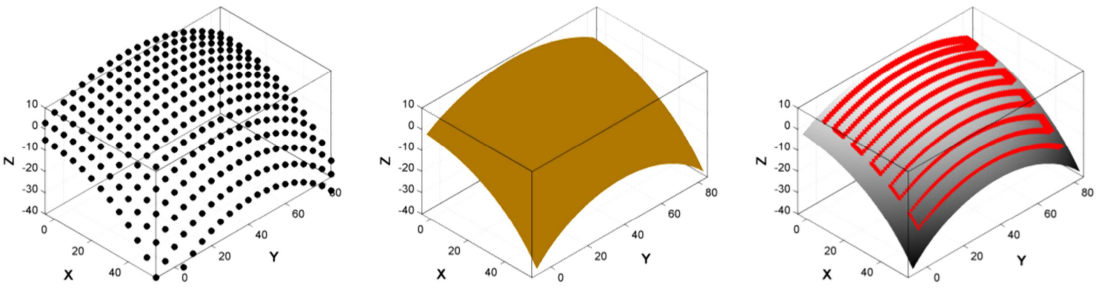

(a) (b)
Figure 4.2 – a) Sample; b) Particular of the top surface.

Figure 4.3a shows the point cloud (input dataset for the script), the approximated surface and a calculated tool-path. Point clouds can be acquired with coordinate measuring machines, CMM (e.g. laser scanners), when the CAD model of the samples is not available or it differs from the real geometry. The point cloud in Figure 4.3a was numerically generated for the spherical surface, since a precise model of the surface was not required in this case. The example tool-path given in Figure 4.3c was generated setting a speed of 20 mm/s and a raster-step of 5 mm. Each point of the path is highlighted by a red vector representing the normal direction to the surface.



(a) (b) (c)
Figure 4.3 – a) Cloud of points; b) Surface; c) Calculated path.

The MATLAB script was developed to receive a point cloud representing the digitalized surface of interest. The cloud had to be in the form *pc=(x, y, z)*, a vector of 3 columns with the number of rows equal to the number of points in the cloud. 3D vectors, normal to the surface related to the point cloud, were computed through the execution of the script.

The first step of the script consisted in the calculation of an analytical surface. This step was based on an open source MATLAB function, GRIDFIT [15], capable of fitting a surface of the form z(x, y) to scattered (or regular) points. GRIDFIT is not an *interpolant* function; it allows the existence of replicates in the cloud of points with no problems. An *interpolant* is a code designed to always exactly predict all supplied data. GRIDDATA and INTERP1 are examples of interpolant functions in MATLAB.

GRIDFIT is more accurately described as an approximant function. It produces a surface which represents the behaviour of the supplied data as closely as possible, allowing for noise in the data and for replicate data. A nice feature of GRIDFIT is its ability to smoothly extrapolate beyond the convex hull of your data, something that GRIDDATA cannot do[9].

For understanding the philosophical underpinnings of GRIDFIT, it is possible to imagine a thin, flexible plate, attached to the data points by elastic bands [15]. Each of these elastic bands draws the plate towards its corresponding data point, stretching only in the z direction. If the data points were to fall on a simple planar surface $z(x, y) = a_0 + a_1 x + a_2 y$, then the result will be a least squares plane. This works because the potential energy stored in an elastic band[10] will be proportional to the square of its extension.

Now it should be imagined data points that do not fall on a plane, but arise from some curved surface, $z(x, y)$. While the plate itself is flexible, it is possible to allow some finite and non-zero bending rigidity that the user can control. The bands connecting the plate to our data points will pull the plate into a curved shape, while the bending rigidity of the plate resists deformation. The relative stiffness of the plate, as compared to the strength of the elastic bands will cause some trade-off between fitting the data as well as possible and a smooth overall surface. It is this plate stiffness that allows GRIDFIT to smoothly extrapolate beyond the data itself. Extrapolation beyond the data will be well behaved, since the plate will tend to become locally planar wherever there is no data to deform it.

When the surface is computed, the SURFNORM function is used to calculate the components of the 3D vector normal to the surface. The normal vectors are normalized to length 1.

The final task of the MATLAB script consisted in the generation of the tool-path. The raster path was initially generated on the plane subtended by the surface. Therefore it was projected upon the surface, evaluating the z-altitude and deriving the angular coordinates (A, B and C) for each point of the planar raster scan (x, y). Although this approach did not ensure a good constant spacing between the command points and constant robot speed along the curved surface, it was deemed satisfactory to allow generation of command coordinates to test the external control.

[9] Except by the slow, memory intensive "v4" method. GRIDDATA can be used with 4 different interpolation methods. The "nearest" and "linear" methods have discontinuities in the zero-th and first derivatives respectively, while the "cubic" and "v4" methods produce smooth surfaces. All the methods except "v4" are based on a Delaunay triangulation of the data.

[10] Perfect, linearly elastic.

The conversion of the vector Cartesian components (N_X, N_Y, N_Z) into the angular coordinates for the robot (A, B, C) was obtained using the following equations:

$$A = -\mathrm{rad2deg}\left[\mathrm{atan2}\left(\frac{N_Y}{\sqrt{N_X^2+N_Z^2}}\right)\right] \quad \text{Eq. 4.1}$$

$$B = \mathrm{rad2deg}\left[\mathrm{atan2}\left(\frac{N_X}{N_Z}\right)\right] \quad \text{Eq. 4.2}$$

$$C = -\mathrm{rad2deg}\left[\mathrm{atan2}\left(\frac{N_Y}{N_Z}\right)\right] \quad \text{Eq. 4.3}$$

A is the angle that the robot tool forms with the X-Z plane; B and C are the angles around the y-axis and x-axis respectively.

4.4.2. Curvilinear coordinates for severe curvature surfaces

The only scope of the script described above was to provide a quick way to get text files of command coordinates to be used to test the external control system. In that script the tool-paths were generated on the plane subtended by the surface. The z-altitude of each packet of coordinates was evaluated projecting the planar raster scan upon the surface of interest (analytically computed). The angular coordinates (A, B and C) were calculated through deriving the vectors normal to the surface for each point.

The generation of a raster tool-path on the plane subtended by the surface, and the following projection upon it, may provide an acceptable solution for a narrow range of slightly curved surfaces. However the approach fails for more complex curved surfaces.

A new approach was developed through a more systematic mathematical investigation of the problem. In order to obtain a good control of the travelled distances on a curved surface *curvilinear* rather than *Cartesian* coordinates were used. Curvilinear coordinates can be obtained from the partial derivatives of a surface in its analytical form. Since the GRIDFIT function used in the first version is not able to provide the analytical form of a surface, it was substituted with a general n-dimensional polynomial fitting function, POLYFITN [16]. It is capable of fitting the point cloud to a double variable polynomial expression. The function calculates a finite number (n) of terms to approximate the dataset with a truncated Taylor series[11]. Therefore, the coefficients of the polynomial approximation of the surface $z = f(x, y)$ are:

[11] The Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

$pCoef = POLYFITN([x, y], z, n)$ Eq. 4.4

The coefficients of the polynomial functions approximating the partial derivatives are calculated with the POLYDERN function [16]:

$pDx = POLYDERN(pCoef, 1)$ Eq. 4.5

$pDy = POLYDERN(pCoef, 2)$ Eq. 4.6

Therefore the partial derivatives can be calculated for any point of the domain with the POLYVALN function [16]:

$\frac{\partial f}{\partial x} = POLYVALN(pDx, [x, y])$ Eq. 4.7

$\frac{\partial f}{\partial x} = POLYVALN(pDy, [x, y])$ Eq. 4.8

The infinitesimal distance (ds) between the point $O = [x_0, y_0, z_0]$ and the point $P = [x_0 + dx,\ y_0 + dy,\ z_0 + dz]$, both on the surface $f(x, y)$, is given by:

$ds = \sqrt{dx^2 + dy^2 + dz^2}$ Eq. 4.9

If $\vec{n} = (u, v, w)$ is a vector for the point O of a given curve on the surface (a reference edge), perpendicular to the curve and tangent to the surface, the infinitesimal curvilinear distance from the curve can be expressed with:

$ds = \sqrt{du^2 + dv^2 + dw^2}$ Eq. 4.10

Where dw is the directional derivative of the surface in O relative to the vector $\vec{n}$. For the formula of the gradient (∇f), it results:

$dw = \frac{\partial f}{\partial \vec{n}}(x_0, y_0) = \nabla f(x_0, y_0) \cdot \vec{n} = \nabla f(x_0, y_0) \cdot \begin{bmatrix} u \\ v \end{bmatrix}$ Eq. 4.11

The gradient of the surface is given by:

$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$ Eq. 4.12

Therefore:

$dw = \frac{\partial f}{\partial x} \cdot u + \frac{\partial f}{\partial y} \cdot v$ Eq. 4.13

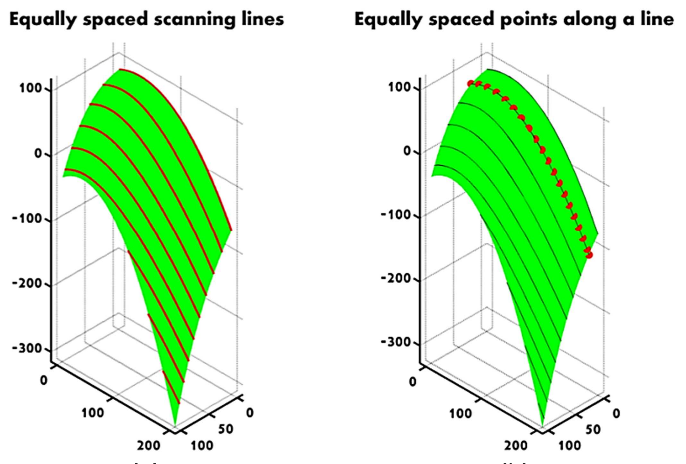
Considering Eq. 4.7 and Eq. 4.8:

$dw = POLYVALN(pDx, [x_0, y_0]) \cdot u + POLYVALN(pDy, [x_0, y_0]) \cdot v$ Eq. 4.14

The curvilinear distance (d) from the point O of the reference edge is finally given by:

$d = \int ds = \iint \sqrt{du^2 + dv^2 + dw^2}$ Eq. 4.15

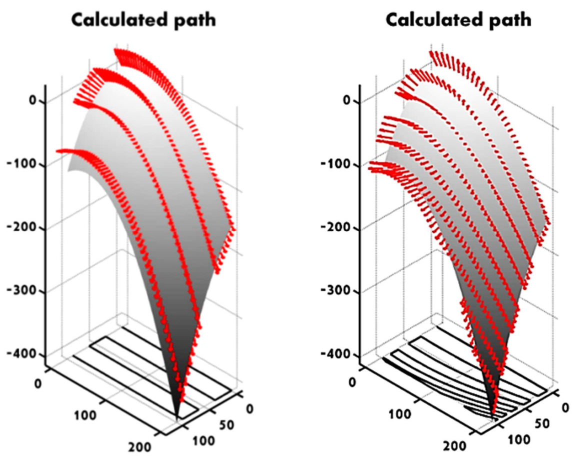
Figure 4.4 shows a surface with a large curvature, representing a challenging situation for path-planning. Monitoring the value of the distance for every point a given reference edge through Eq. 4.15, it was possible to develop an algorithm for the generation of equally spaced raster path passes.



(a) (b)

Figure 4.4 – Equally spaced lines and points on a curved surface.

Constant spacing between the passes of a raster path is important to allow uniform collection of NDT data and constant speed of the probe. Furthermore the scanning lines need to be linked with transition paths in order to get a continuous raster scanning pathway. The algorithm was made capable of linking consecutive lines, following the border of the surface. Figure 4.5 shows a comparison of the scanning path obtained with the Cartesian and the curvilinear approach; in both cases a step of 30 mm between lines and a speed of 400 mm/sec were used as desired parameters for the path.



(a) (b)

Figure 4.5 – Scanning path obtained with the old (a) and the new (b) MATLAB script.

The projection of the tool-paths onto the subtended X-Y plane shows the profound difference between the paths in Figure 4.5a and Figure 4.5b.

4.5. Developing a full NDT path-planning toolbox

Further investigations were undertaken to expand the potentialities of the MATLAB scripts described above, in order to allow path-planning for a wide range of surfaces, beyond those subtending a simply rectangular X-Y domain. A flexible path-planning platform would require a proper application with a friendly Graphic User Interface (GUI) able to manage many robot models, samples, tools and settings.

For the sake of pursuing the vision of a flexible software solution, it was decided to start the development from the establishment of a MATLAB toolbox for off-line path-planning targeted to NDT. Although a MATLAB toolbox is typically not equipped with a GUI and is deployed through the MATLAB console, the development of the fundamental kit of functions was deemed as the most reasonable starting point to investigate and explore the necessary algorithms.

The remaining part of this chapter gives an overview of the work undertaken to establish a bespoke MATLAB toolbox, with particular emphasis to the description of the encountered difficulties and the relative solutions.

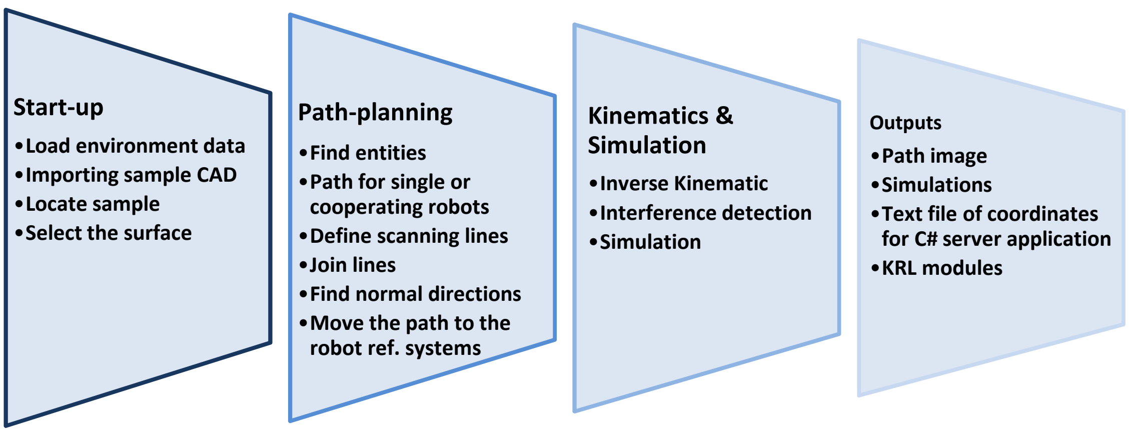


Figure 4.6 – Schematic representation of the path-planning toolbox modular structure.

The workflow can be broken into a series of specific subtasks, shown in Figure 4.6. The developed toolbox consists of the functions given in Table 4.2.

Start-up	Path-planning	Kinematics & Simulation	Output
F_LoadSample	F_QueryEdge	F_ShowPath	F_Output
F_RecognizeSurfaces	F_FindPathLines	F_ShowDemo	
F_SelectSurface	F_JoinLines		
F_Int_LocateSample	F_FindNormals		
F_LocateSample			
F_FindEntities			
F_ShowEntities			

Table 4.2 – Functions of path-planning MATLAB toolbox target to robotic NDT.

All the functions were conveniently used in the main MATLAB script (*MainProgram.m*), given in Appendix B1.

The following pages explain how the current toolbox works, using an example of scanning a large curved surface (1.6 m^2) from a composite aerospace winglet available at TWI, shown in CAD form in Figure 4.7.

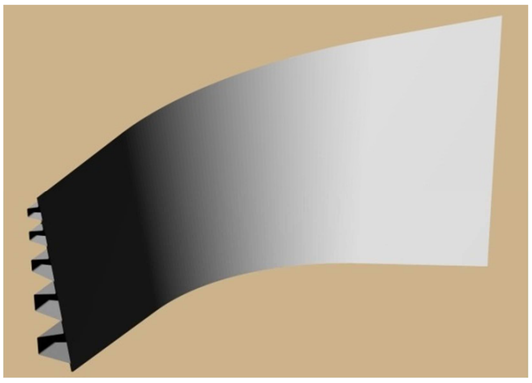


Figure 4.7 – Composite aerospace winglet.

4.5.1. Start-up module

The *start-up* module is addressed to load environment data (the robotic cell), import the CAD file of the sample, start an interactive procedure to locate the sample in the right place within the cell and ask to the user the sample's surface he wants to scan.

The function responsible for importing the CAD file is *F_LoadSample*. It is able to load sample geometry in STL format. STL (STereoLithography) is a file format native to the stereolithography CAD software created by 3D Systems [17]. STL is also known as Standard Tessellation Language [17]. This file format is supported by many other software packages; it is widely used for rapid prototyping and computer-aided manufacturing. *F_LoadSample* can load STL files in their ASCII and binary representation. The binary files are more common since they are more compact [18]. An STL file describes a raw unstructured triangulated surface by the unit normal vector and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system (Figure 4.8).

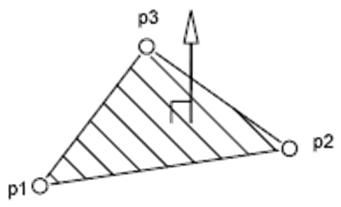


Figure 4.8 – STL Triangle.

An ASCII STL file is a human readable format; it begins with the line:

```
solid name
```

where *name* is an optional string (though if *name* is omitted there must still be a space after solid). The file continues with any number of triangles, each represented as follows:

```
facet normal nᵢ nⱼ nₖ
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
```

where each *n* or *v* is a floating point number in sign-mantissa (e-sign-exponent format), e.g. "-2.648000e-002". The file concludes with:

```
endsolid name
```

The structure of the format suggests that other possibilities exist (e.g. facets with more than one “loop”, or loops with more than three vertices) but in practice, all facets are simple triangles. White space (spaces, tabs, newlines) may be used anywhere in the file except within numbers or words. The spaces between “facet” and “normal” and between “outer” and “loop” are required. Because ASCII STL files can become very large, a binary version of STL exists. A binary STL file has an 80 character header (which is generally ignored – but which should never begin with “solid” because that will lead most software to assume that this is an ASCII STL file). Following the header is a 4 byte unsigned integer indicating the number of triangular facets in the file. Following that is data describing each triangle in turn. The file simply ends after the last triangle. Each triangle is described by twelve 32-bit-floating point numbers: three for the normal and then three for the X/Y/Z coordinate of each vertex – just as with the ASCII version of STL.

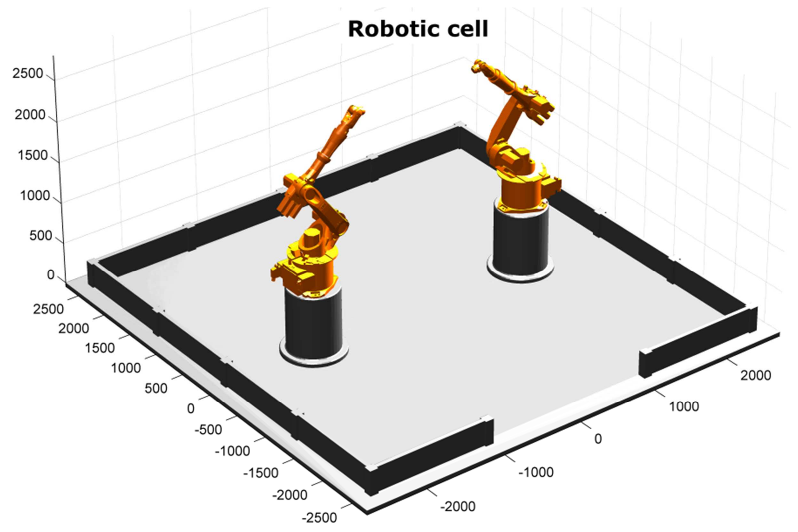


Figure 4.9 – CAD model of the TWI robotic cell.

The *start-up* module imports the geometry of the robotic cell in the same way it loads the geometry of the sample. Therefore the cell and the robots that compose the robotic environment are embedded in the toolbox as STL files. Figure 4.9 shows how the TWI robotic cell, imported by the *start-up* module, looks like. The CAD models of the KR16 L6-2 robots were kindly provided by KUKA Robotics UK Ltd.

The function *F_RecognizeSurfaces* was developed to recognise all different surfaces present in a three-dimensional model, loaded by *F_LoadSample,* and divide them in different memory arrays during its execution. The function analyses the triangular mesh, originally coming from STL CAD file, to identify groups of contiguous triangles. Since the number of triangles in the triangular mesh of the whole model can be very big, the function operates through indexes. All triangles and nodes of the mesh are indexed, in order to compare integer numbers rather than double precision variables. This allows the algorithm to be fast and to differentiate the surfaces in a reasonable amount of time.

After loading the STL file of the sample, the user is required to locate the sample within the virtual robot cell. *F_Int_LocateSample* is the function responsible for locating the sample via an interactive procedure. *F_LocateSample* is instead a non-interactive function able to use previous location data to set the sample within the robotic cell. With the interactive function, the *start-up* module asks the user to inform about the robot in use. The user can decide to use either the Master or the Slave robot.

As it was anticipated, the toolbox has no GUI. However the user can manage the progress of the workflow through the MATLAB command window.

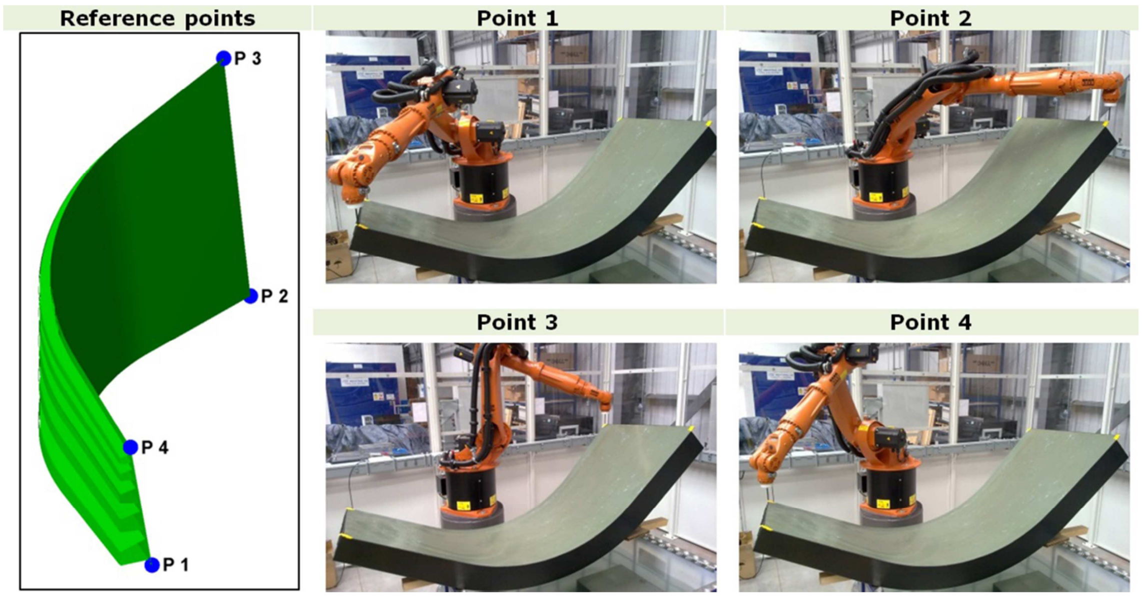


Figure 4.10 – Robotic arm conducted to the four reference points of the TWI winglet sample.

The interactive procedure to transfer the real position of the sample to the MATLAB environment is based on the acknowledgement of the Cartesian coordinates of four points of the sample. The user is interactively asked to select four points of the sample model and to provide their coordinates. These coordinates can be obtained through jogging the robot to each of the selected points and reading the relative coordinates from the screen of the teach-pendant. Figure 4.10 shows the robotic arm jogged to the four reference points selected to assess the position of the TWI winglet sample.

After having inserted the measured Cartesian coordinates in the algorithm, through typing them directly into the MATLAB command window, the function positions the sample model in the virtual robot cell environment.

Existing software usually considers a CAD model to strictly correspond to the real part. This can be tolerated for well machined metallic samples, but it is sometimes the source of unacceptable errors for large composite components. Therefore the function implemented in the toolbox uses a positioning algorithm originally proposed for 3D point cloud data registration [19]. It calculates the optimum position of the STL mesh within the virtual robot cell in order to minimize the square errors of the distances between the four points selected in the real sample and the relative points in the CAD model. The centroids of the two datasets (P_r and P_e), named cP_r and cP_e, are given by:

$$cP_r = [cP_{rx} \quad cP_{ry} \quad cP_{rz}] = \left[\frac{\sum_{i=1}^{4} P_{rx}^{i}}{4} \quad \frac{\sum_{i=1}^{4} P_{ry}^{i}}{4} \quad \frac{\sum_{i=1}^{4} P_{rz}^{i}}{4}\right]; \qquad \text{Eq. 4.16}$$

$$cP_e = [cP_{ex} \quad cP_{ey} \quad cP_{ez}] = \left[\frac{\sum_{i=1}^{4} P_{ex}^{i}}{4} \quad \frac{\sum_{i=1}^{4} P_{ey}^{i}}{4} \quad \frac{\sum_{i=1}^{4} P_{ez}^{i}}{4}\right]; \qquad \text{Eq. 4.17}$$

Therefore the point clouds with the centroids removed (mP_r and mP_e) are:

$$mP_r = P_r - cP_r; \qquad \text{Eq. 4.18}$$

$$mP_e = P_e - cP_e; \qquad \text{Eq. 4.19}$$

A correlation matrix (H) is calculated as:

$$\text{H} = \sum_{i=1}^{4} mP_r(i,:)^T * mP_e(i,:); \qquad \text{Eq. 4.20}$$

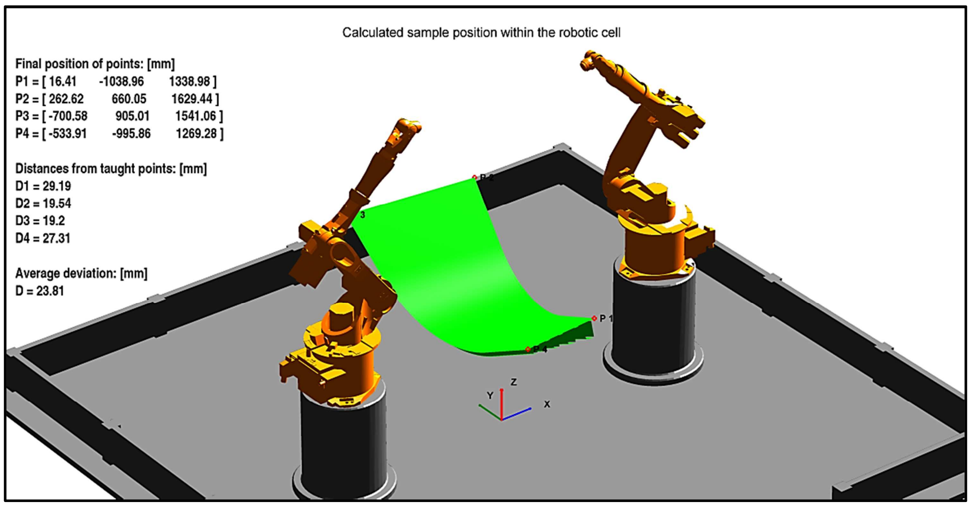
The singular value decomposition (SVD) function, applied to the correlation matrix, produces a diagonal matrix S of the same dimension as H, with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that $H = U * S * V'$.

The rotation matrix (R) and the translation vector (t), necessary to match the experimental cloud of points (P_e) with the reference cloud (P_r), are then calculated from the SVD output.

$R = V * U';$ Eq. 4.21

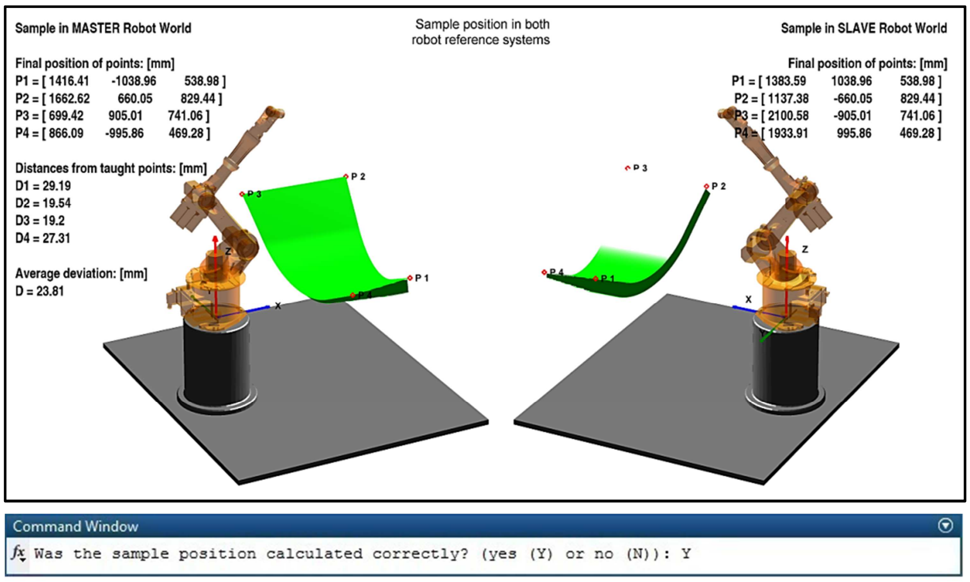
$t = cP_e' - R * cP_r';$ Eq. 4.22

The calculated rotation matrix and translation vector are used to move the meshed sample to the optimum position. The result of this operation is then displayed, so that the user can verify and accept before proceeding. Figure 4.11 and Figure 4.12 are screenshots of the 3D plots produced by the MATLAB toolbox at the end of the interactive calibration procedure.



```
Final position of points: [mm]
P1 = [ 16.41     -1038.96     1338.98 ]
P2 = [ 262.62      660.05     1629.44 ]
P3 = [ -700.58     905.01     1541.06 ]
P4 = [ -533.91    -995.86     1269.28 ]

Distances from taught points: [mm]
D1 = 29.19
D2 = 19.54
D3 = 19.2
D4 = 27.31

Average deviation: [mm]
D = 23.81
```

Figure 4.11 – Calculated position of the sample with coordinates referenced to the origin of the robot cell virtual environment. The MATLAB console waits for the user to approve.



```
Sample in MASTER Robot World

Final position of points: [mm]
P1 = [ 1416.41   -1038.96     538.98 ]
P2 = [ 1662.62     660.05     829.44 ]
P3 = [ 699.42      905.01     741.06 ]
P4 = [ 866.09     -995.86     469.28 ]

Distances from taught points: [mm]
D1 = 29.19
D2 = 19.54
D3 = 19.2
D4 = 27.31

Average deviation: [mm]
D = 23.81
```

```
Sample in SLAVE Robot World

Final position of points: [mm]
P1 = [ 1383.59    1038.96     538.98 ]
P2 = [ 1137.38    -660.05     829.44 ]
P3 = [ 2100.58    -905.01     741.06 ]
P4 = [ 1933.91     995.86     469.28 ]
```

```
fx Was the sample position calculated correctly? (yes (Y) or no (N)): Y
```

Figure 4.12 – Calculated position of the sample with coordinates referenced to the origin of each robot. The MATLAB console waits for the user to approve.

The display of the results facilitates the evaluation of the completed calibration procedure. The measured points are numbered and highlighted through red markers. The coordinate reference systems of both master and slave robot are displayed. The calculated coordinates and the deviations from the taught points are reported. The calculated coordinates can be used by the user to teach a BASE reference system for the sample of interest to the robot controllers.

The approval of the assessed sample position and the selection of the surface of interest mark the end of the *start-up* module tasks and the start of the *path-planning* operations.

4.5.2. Path-planning module

F_FindEntities is the function responsible for approximating the triangular mesh, coming from the STL file, with an analytical surface. It is able to increase the order of the approximating polynomial function, in order to guarantee that the maximum error (err_max) is smaller than a target error indicated by the user (ERR_MAX). This is done by the following programmed "*while loop".*

```
PC=unique([X_coor Y_coor Z_coor],'rows');
XY=[PC(:,1),PC(:,2)];

err_max=ERR_MAX+1;

while err_max>ERR_MAX
    i=i+1;
    Coef_surface = polyfitn(XY,PC(:,3),i);
    z = polyvaln(Coef_surface,PC(:,1:2));
    err = abs(z-PC(:,3));
    err_max=max(err);
end
```

The function "*unique"* is used before the loop to remove duplicate nodes. Since a node can be linked to many triangles, each node is present many times as the number of triangles linked to it. The deletion of duplicate nodes reduces the length of the array and the duration of each loop.

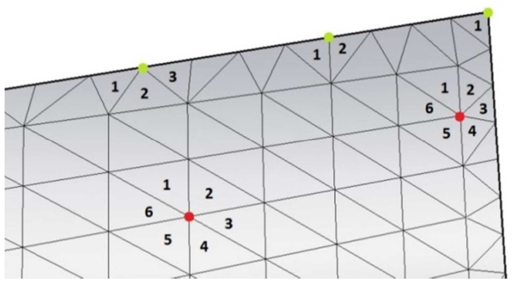
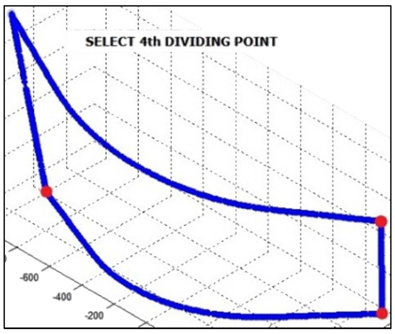
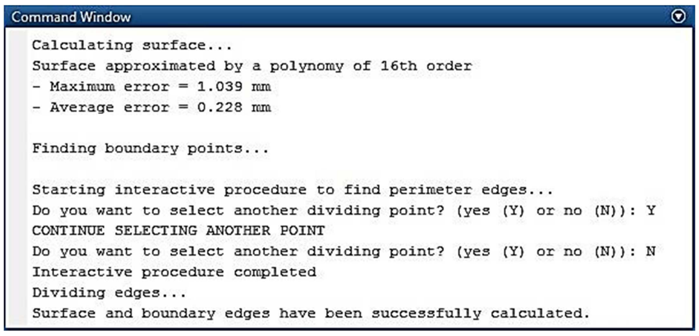


Figure 4.13 – Typical number of triangles attached to each node.

The same function is also responsible for finding the free boundary of the meshed surface. This is done evaluating the number of triangles linked to each node of the mesh. Each node shared among less than 4 triangles is recognised as a point belonging to the perimeter of the surface. Figure 4.13 can better clarify how this algorithm works. The green marked boundary points are always attached to less than 4 triangles whilst red marked points within the surface are usually surrounded by 6 triangles.

Thus, the algorithm starts an interactive procedure to divide the surface perimeter into separate continuous edges, according to the corner points indicated by the user (Figure 4.14).




Command Window

```
Calculating surface...
Surface approximated by a polynomy of 16th order
- Maximum error = 1.039 mm
- Average error = 0.228 mm

Finding boundary points...

Starting interactive procedure to find perimeter edges...
Do you want to select another dividing point? (yes (Y) or no (N)): Y
CONTINUE SELECTING ANOTHER POINT
Do you want to select another dividing point? (yes (Y) or no (N)): N
Interactive procedure completed
Dividing edges...
Surface and boundary edges have been successfully calculated.
```

Figure 4.14 – The algorithm divides the perimeter into continuous edges.

F_ShowEntities is an optional function able to show the calculated analytical surface and the perimeter divided into separate edges. It is not a strictly necessary function, but gives the user the possibility to evaluate the found entities and check if something went wrong.

Once the geometric entities are correctly recognised, the algorithm asks the user to indicate the reference edge to generate the scanning path. F_QueryEdge is the function responsible for an interactive selection of the reference edge. The user is enabled to select one edge clicking on any of its points. The selected edge gets highlighted in green colour, as showed in Figure 4.15.

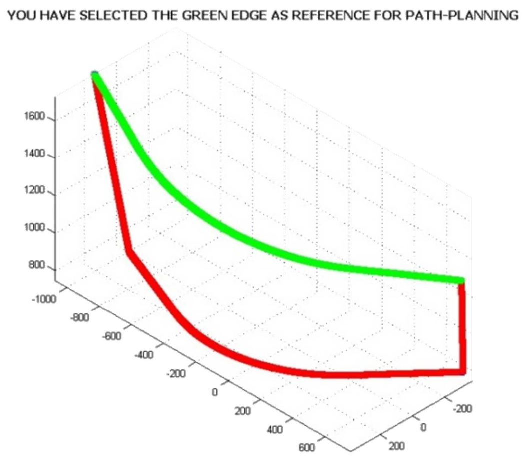
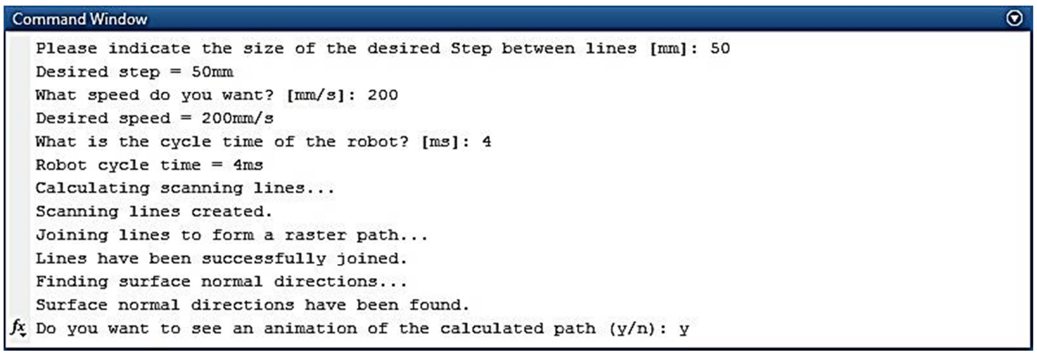


Figure 4.15 – Selected edge highlighted in green colour.

Once the analytical surface is available, the boundary is defined and the reference edge is selected, the *Path-planning* module is ready to generate the scanning path. The *F_FindPathLines* function asks the user to know fundamental parameters (*desired step* in the raster scanning path and *speed* of the probe/end effector), before proceeding with the calculation of the tool-path.



```
Please indicate the size of the desired Step between lines [mm]: 50
Desired step = 50mm
What speed do you want? [mm/s]: 200
Desired speed = 200mm/s
What is the cycle time of the robot? [ms]: 4
Robot cycle time = 4ms
Calculating scanning lines...
Scanning lines created.
Joining lines to form a raster path...
Lines have been successfully joined.
Finding surface normal directions...
Surface normal directions have been found.
fx Do you want to see an animation of the calculated path (y/n): y
```

Figure 4.16 – MATLAB Command Window showing the progress of the path-generation.

4.5.2.1. Generation of the raster path – analytical surface approximation approach

The calculation of the lines composing a raster tool-path for a given surface is based on the creation of equally spaced parallel trajectories starting from a selected reference edge. The first line is calculated according to the selected reference edge. The path-planning module implemented within the toolbox is based on the analytical approach described in Section 4.4.2. The reference edge and each consecutive new line of the raster path are parametrically defined by analytical functions before deriving the next parallel line.

4.5.2.2. Speed patterns

One problem mentioned in Section 3.6.1 regarded the dynamic accuracy of externally controlled a robot arm. When the robotic arm is forced (by the external control via the RSI Ethernet communication) to suddenly change its direction of travel whilst maintaining a constant speed, poor dynamic accuracy is noticed in proximity of the trajectory cusps (see Figure 3.24 and Figure 3.31 - the larger errors are located at the two sides of the raster scan, where the robotic arm inverts the direction of travel).

The path-planning function developed for the toolbox adds acceleration and deceleration ramps at the start and at the end point of each parallel trajectory composing the raster scan. If α is the duration of the increasing and decreasing ramps of the speed pattern in a normalized time scale t, $v(\mathrm{t})$ is the normalized speed function and $v'(\mathrm{t})$ is its derivative, the following conditions are applied to obtain a continuous speed pattern:

$$\begin{cases} v(0) = 0 \\ v(\alpha) = 1 \\ v(1-\alpha) = 1 \\ v(1) = 0 \end{cases} \quad \begin{cases} v^{'}(0) = 0 \\ v^{'}(\alpha) = 0 \\ v^{'}(1-\alpha) = 0 \\ v^{'}(1) = 0 \end{cases} \qquad \text{Eq. 4.23}$$

The typical speed pattern is given in Figure 4.17a. The increasing ramp of the speed pattern is described by the function:

$$v(t) = -\frac{2}{\alpha^3}t^3 + \frac{3}{\alpha^2}t^2 \qquad \text{for } 0 \le t \le \alpha \qquad \text{Eq. 4.24}$$

The speed is kept constant between $t = \alpha$ and $t = 1 - \alpha$. The decreasing ramp of the speed pattern is governed by:

$$v(t) = \frac{2}{\alpha^3}(t+\alpha-1)^3 - \frac{3}{\alpha^2}(t+\alpha-1)^2 + 1 \qquad \text{for } (1-\alpha) \le t \le 1 \qquad \text{Eq. 4.25}$$

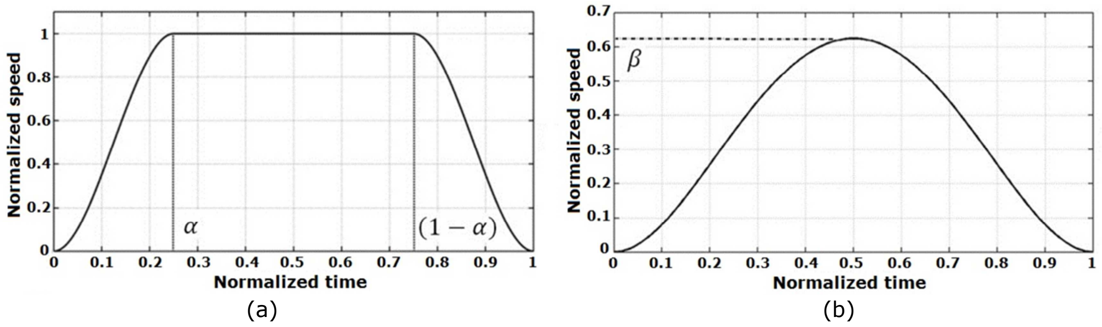


Figure 4.17 – Typical speed pattern of each line of a raster path (a) and of the links between lines (b).

The algorithm continues by starting the function *F_JoinLines*, responsible for joining the lines in a single raster scan path. The end of each line is linked to the first point of the next line, inserting a connecting path. Likewise the speed of the raster lines, the speed pattern of the tool central point during the step between two lines is optimized, applying the following conditions:

$$\begin{cases} v(0) = 0 \\ v(0.5) = \beta \\ v(1) = 0 \end{cases} \quad \begin{cases} v^{'}(0) = 0 \\ v^{'}(0.5) = 0 \\ v^{'}(1) = 0 \end{cases} \qquad \text{Eq. 4.26}$$

where β is a percentage of the target speed used for the raster scan. This parameter spans between 0 and 1 according to the length of the trajectory linking two consecutive lines of the scan path. Small values of β are used for short trajectories, to let the robot smoothly abandon the end point of the finished line and reach the starting point of the next line. The speed function results:

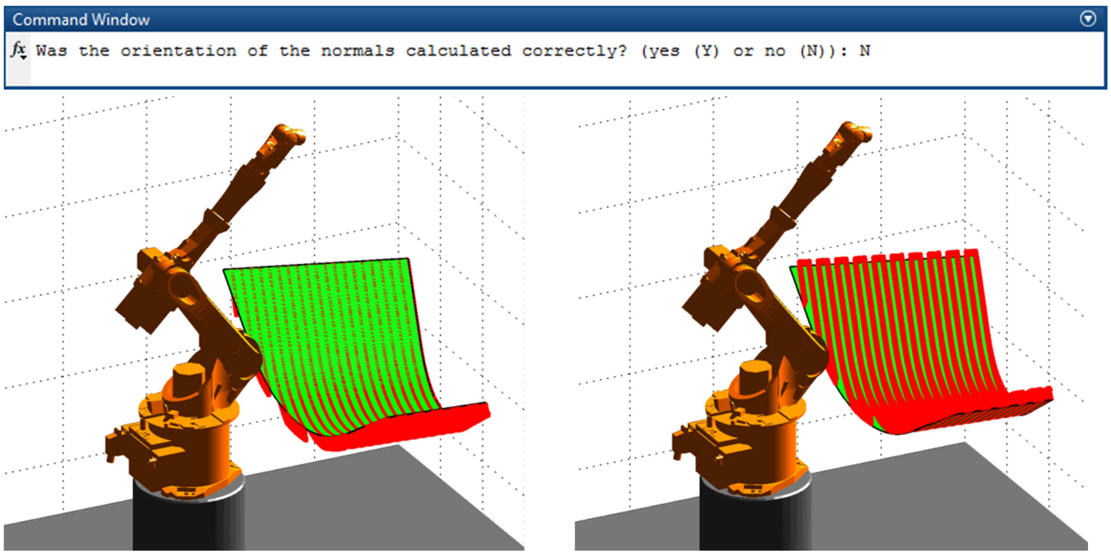
$$v(t) = 16\beta t^4 - 32\beta t^3 + 16\beta t^2 \quad \text{for } 0 \le t \le 1 \qquad \text{Eq. 4.27}$$

The typical pattern of the speed is given in Figure 4.17b.

4.5.2.3. Computation of the angular coordinates

As soon as the path is defined, the function *F_FindNormals* calculates the normal direction to the surface for each point of the path. The MATLAB function *SURFNORM* is used to calculate the components of the 3D vector normal to the surface for each point of the tool-path. The normal vector is normalized to length 1.

An interactive procedure to check the orientation of the displayed orientation of the normals allows its correction if necessary. A wrong orientation of the normals would result in collisions between the robot and the sample, since the robotic arm would try to approach the surface from its back wall. Figure 4.18 shows the correction of the normal orientation, by requesting it through the command window.



```
fx Was the orientation of the normals calculated correctly? (yes (Y) or no (N)): N
```

(a) (b)

Figure 4.18 – Correction of the normal orientation.

The conversion of the vector components (N_X, N_Y, N_Z) into the angular coordinates for the robot (A, B and C) was already described in Section 4.4.1, where the problem is addressed with the empirical equations Eq. 4.1-4.3. That empirical formulation was initially considered correct, until it was revealed that it leads to acceptable results only when a tool-path prevalently lying on a horizontal plane is generated. It produced unacceptable robot wrist orientations for tool-paths extending in vertical direction.

A conversion formulation valid for all scenarios was searched. A generally valid formulation was found in a didactic KUKA tutorial [20]; it is based on the following rotational matrix:

$$R = \begin{bmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} \end{bmatrix} = -\begin{bmatrix} \mathrm{V_X} & \mathrm{T_X} & \mathrm{N_X} \\ \mathrm{V_Y} & \mathrm{T_Y} & \mathrm{N_Y} \\ \mathrm{V_Z} & \mathrm{T_Z} & \mathrm{N_Z} \end{bmatrix} \qquad \text{Eq. 4.28}$$

The third column of the matrix contains the components of the normal vector (calculated by SURFNORM). The second column contains the tangential vector, representing the direction of travel calculated as:

$$T = \begin{bmatrix} \mathrm{T_X} \\ \mathrm{T_Y} \\ \mathrm{T_Z} \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{dx}}{\sqrt{\mathrm{dx}^2+\mathrm{dy}^2+\mathrm{dz}^2}} \\ \frac{\mathrm{dy}}{\sqrt{\mathrm{dx}^2+\mathrm{dy}^2+\mathrm{dz}^2}} \\ \frac{\mathrm{dz}}{\sqrt{\mathrm{dx}^2+\mathrm{dy}^2+\mathrm{dz}^2}} \end{bmatrix} \qquad \text{Eq. 4.29}$$

where dx, dy and dz are the gradients of the trajectory in the three dimensions. The first column contains the bi-normal vector:

$$V = \begin{bmatrix} \mathrm{V_X} \\ \mathrm{V_Y} \\ \mathrm{V_Z} \end{bmatrix} = \begin{bmatrix} \mathrm{N_Y T_Z - N_Z T_Y} \\ \mathrm{N_Z T_X - N_X T_Z} \\ \mathrm{N_X T_Y - N_Y T_X} \end{bmatrix} \qquad \text{Eq. 4.30}$$

Thus the angular coordinates (A, B and C) are calculated through the following general formulation:

$$B = atan2\left(\frac{-R_{31}}{\sqrt{R_{11}^2+R_{21}^2}}\right) \qquad \text{Eq. 4.31}$$

$$\mathrm{A} = \begin{cases} 0, & \text{for } |B| = \pi/2 \\ atan2\left(\frac{R_{21}}{R_{11}}\right), & \text{for } |B| \neq \pi/2 \end{cases} \qquad \text{Eq. 4.32}$$

$$\mathrm{C} = \begin{cases} C = \frac{\mathrm{B}}{\mathrm{abs(B)}} \cdot atan2(R_{12}), & \text{for } |B| = \pi/2 \\ C = atan2\left(\frac{R_{32}}{R_{33}}\right), & \text{for } |B| \neq \pi/2 \end{cases} \qquad \text{Eq. 4.33}$$

The new formulation allows keeping the probe axis perpendicular to any surface. The new algorithm was validated after its implementation through approaching the surfaces of a rectangular prism sample.

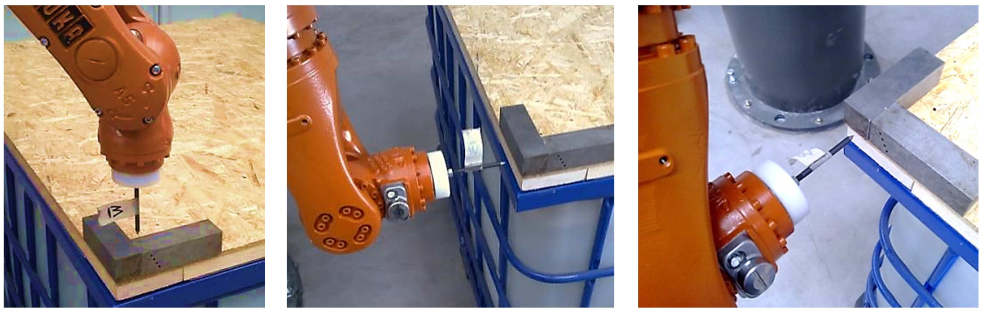

(a) (b) (c)

Figure 4.19 – Approach of different planes through an externally controlled robot, to check the perpendicularity of the robot tool; X-Y plane (a), Y-Z plane (b) and X-Z plane (c).

The surfaces of the prism lying on different planes were approached by the externally controlled KR16 L6-2 robot at TWI, in order to check the perpendicularity between the robot tool (a simple spike attached to the robot end-effector) and each of the surfaces. Figure 4.19 shows the robot tool correctly pointing towards the three different surfaces of the prismatic sample. The incorrectness of the old formulation is clearly highlighted by Figure 4.20, showing the evident wrong orientation of the robot tool when approaching the vertical Y-Z plane.

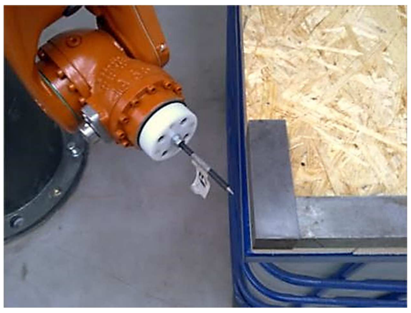


Figure 4.20 – Wrong orientation of the robot tool given by the old formulation.

The robot tool-path is originally generated coherently to the reference system of the robotic cell. The function *F_PathToRobotSpace* moves the generated tool-path to the reference system of the master robot or of the slave robot, according to the preference expressed by the user to scan the surface through one of the available robots in the working space. It enables the toolbox to generate tool-paths suitable to all robots able to reach the part.

The translation of the tool-path to the selected robot space marks the end of the *Path-planning* module.

4.5.3. Kinematics & Simulation module

The *Kinematics & Simulation* module of the path-planning toolbox was developed to simulate the robot movement for the generated tool-paths.

As it is explained in Section 3.2.1, simulations are created through the implementation of the inverse kinematic laws of the mechanic chains. The development of the simulation function of the toolbox (*F_ShowDemo*) was initially based on an existing open-source MATLAB toolbox, the KUKA Control Toolbox (KCT), produced by the University of Siena (Italy) in 2011 [21]. KCT is not a toolbox for off-line path-planning purposes, which is one of the novelty aspects of this work, but it includes more than 40 interesting functions, carrying out operations such as forward and inverse kinematics computation, point-to-point axial and Cartesian real-time control. The toolbox has an internal library, which make it compatible with all 6 degrees of freedom (DoF) small and low-payload KUKA robots. The library includes the model of the KUKA KR16 L6-2 and of the KUKA KR5arcHW robot.

The *Path-Planning* module of the toolbox was made able to calculate the position and roll-pitch-yaw orientation of the end effector of the robot arm (the pose: $\vec{p} = [X, Y, Z, A, B, C]^T$) for each point of the tool-path. It was sufficient to use just the following function of the KCT toolbox to compute the inverse kinematics solution (**q**) from the vector **p**.

q=*kctikinerpy*(**p**) Eq. 4.34

The vector $\vec{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$ contains the 6 angular coordinates for the each robot joint. *F_ShowDemo* loads the CAD file of each joint of the robot and assembly it according to the angular coordinates. If *Ri* and *Ti* are respectively the rotational matrix and the translation vector of the *i*-th joint to be linked to the *(i-1)*-th joint, the following leads to the correct simulation of the robot position for each pose during the raster scan. The distance of the linking point (*Pi*) of each joint, with respect to the origin of the robot reference system, is named as *Li*.

```
%% Portion of MATLAB Script to set the robot in the correct pose

R1=[cos(q1)      -sin(q1)       0;
    sin(q1)      cos(q1)        0;
    0            0              1];
T2=R1*L1';

R2=[cos(q2)       0     sin(q2);
    0             1     0;
    -sin(q2)      0     cos(q2)];
R2=R1*R2;
T2=R2*L2';
T3=T3+T2;

R3=[cos(q3)       0     sin(q3);
    0             1     0;
    -sin(q3)      0     cos(q3)];
R3=R2*R3;
T3=R3*L3';
T4=T4+T3;

R4=[1     0             0;
    0     cos(q4)       -sin(q4);
    0     sin(q4)       cos(q4)];
R4=R3*R4;
T4=R4*L4';
T5=T5+T4;

R5=[cos(q5)      0      sin(q5);
    0            1      0;
    -sin(q5)     0      cos(q5)];
R5=R4*R5;
T5=R5*L5';
T6=T6+T5;
```

```
R6=[1      0           0;
    0      cos(q6)     -sin(q6);
    0      sin(q6)     cos(q6)];
R6=R5*R6;

ROBOT_1=transformSTL(Robot_1,R1);
ROBOT_2=transformSTL(Robot_2,R2,T2);
ROBOT_3=transformSTL(Robot_3,R3,T3);
ROBOT_4=transformSTL(Robot_4,R4,T4);
ROBOT_5=transformSTL(Robot_5,R5,T5);
ROBOT_6=transformSTL(Robot_6,R6,T6);
```

The function *transformSTL* returns the triangular mesh of each robot joint after its rotation and translation according to the rotational matrix and the translation vector *Ri* and *Ti*. The reason for the rotational matrices to not have all the same formulation descends from the observation they have to describe rotations around different axes, as showed in Figure 4.21. The rotations around axis Z, Y and X are described by the matrices given in Appendix A5.

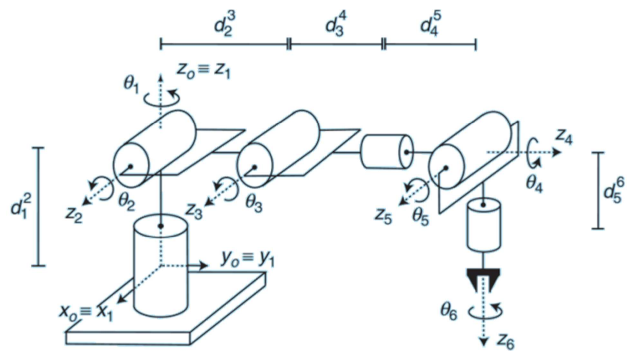


Figure 4.21 – Reference robot: 6-DoF manipulator with a spherical wrist.

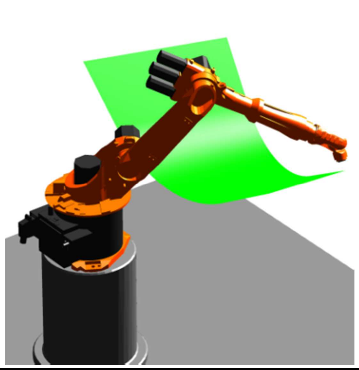
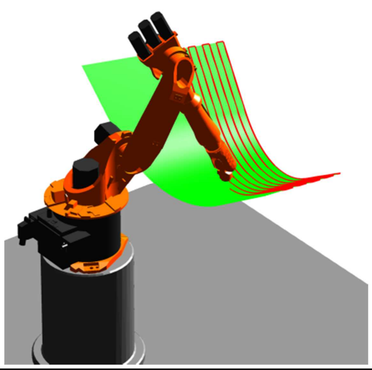
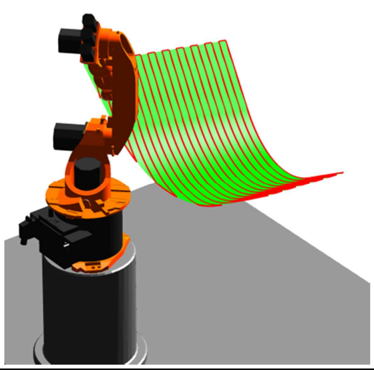
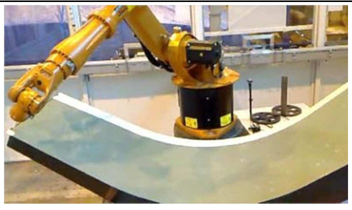
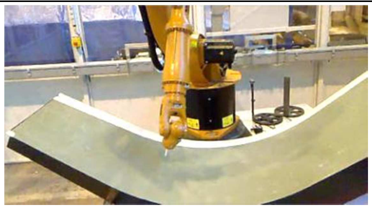
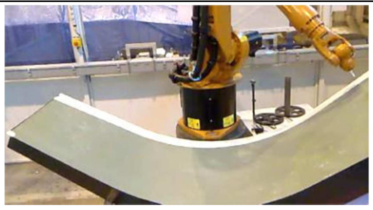
0% Completed – Time 0.0 sec	50% Completed – Time 172.7 sec	100% Completed – Time 345.4 sec
		
		

Figure 4.22 – Snapshots of the simulation video and of the real raster scan executed in the lab.

Figure 4.22 shows three snapshots of a simulated tool-path and of the recorded video of the robot execution. The raster path was generated using the top edge of the surface as reference edge, raster step of 50 mm and speed of 200 mm/s.

Some limitations of the KCT open-source function to compute the inverse kinematics will be highlighted in Section 4.8, where a full solution to the inverse kinematics formulation is presented.

4.5.4. Output module

F_Output, was developed to output the final text files, in order to let externally controlled robot arms execute the planned tool-paths. The function generates the necessary files for each generated tool-path to be executed through an externally controlled robot. Each generated tool-path requires two files to be executed: the first is a text file containing all packets of command coordinates; the second file ("*LogFile_xxxxxx.txt*") contains the necessary coordinates (the succession of robot poses) to set up suitable trajectories to safely approach the sample from the HOME position at the beginning of the inspection task and retract to the HOME at the end of the task.

The second output file enables deploying a new method, developed to automatically communicate the succession of approach and retract poses to the robot. It avoids the need for the operator to copy coordinates to the KRL module and it was put in place through creating an appropriate software configuration of both the RSI interface and the external server application. It brings several advantages:

1. It removes the risk of introducing human typing errors and copying wrong coordinates.
2. The robot manipulator cannot follow unpredictable trajectories, which are extremely worrying for the health and safety of the working environment.
3. It is much less time-consuming than manual typing.
4. It does not require different KRL modules. The KRL module can always remain the same across completely different tool-paths. The only role left to the KRL module is to open the Ethernet communication with the external computer, in order to receive instructions about the approach and retract motion and the real-time communication of command path coordinates.

5. It enables the development of seamless software for completely automated inspection processes, from the positioning of the sample within the robot cell to the evaluation of the results.

Details about the implementation of this solution can be found in Appendix B2.

The output function was also designed to pave the way to the use of two or more robots running independent or synchronized tool-paths. For the specific case of the master-slave robotic system at TWI, when the user assigns a target surface to the Master Robot and another surface to the Slave Robot to be inspected, the function generates three couples of text files. The first two couples contain the tool-paths for the Master and for the Slave Robot. They can be independently used. The third couple of files puts together the command coordinates to be sent simultaneously to the two robot controllers to achieve synchronized movements. Each line of this last file comprises two packets of coordinates; one addressed to the Master controller and the other to the Slave controller. This paves the way to the development of external control for more robots running simultaneously in a synchronized manner. The maximum positional delay of the synchronized robots would be equal to the maximum distance the robot with the longer interpolation cycle can cover in one cycle. For example, for synchronized robots running with 4ms and 12ms interpolation cycles and moving at 100mm/s, the maximum positional delay would be equal to 1.2mm.

Figure 4.23 shows a screenshots of the simulation of an ultrasonic through-transmission inspection for a fan blade at 50% completion.

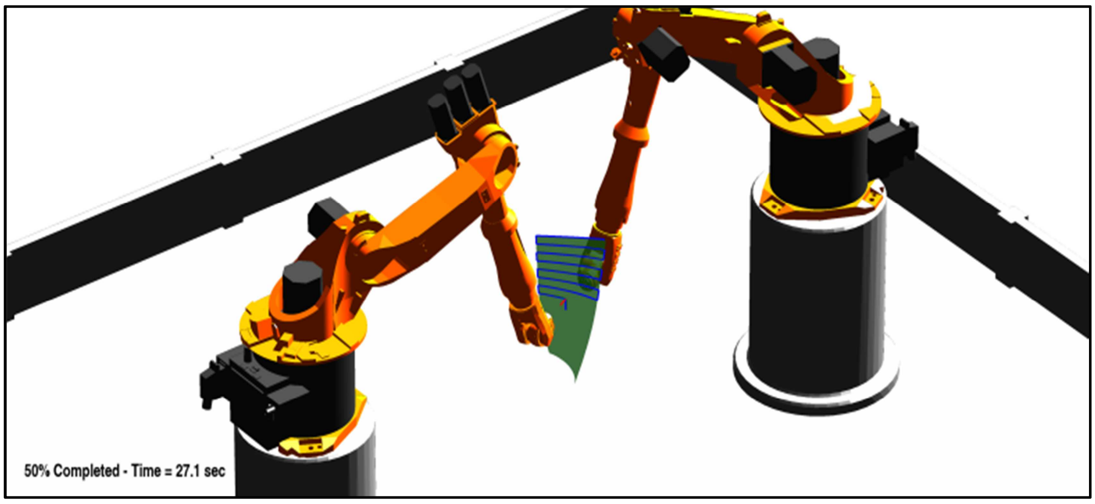


Figure 4.23 – Simulation of an ultrasonic through-transmission inspection of a turbine blade.

4.6. Metrology experiments – Path accuracy verification

To evaluate the accuracy of the tool-paths generated through the developed path-planning toolbox, a precision non-contact laser distance measurement probe (Acuity AR200-100) was mounted to the end effector of the KUKA KR16 L6-2 robotic arm at TWI (Figure 4.24). This allowed monitoring the probe separation from the sample curved surface during a full surface scan.

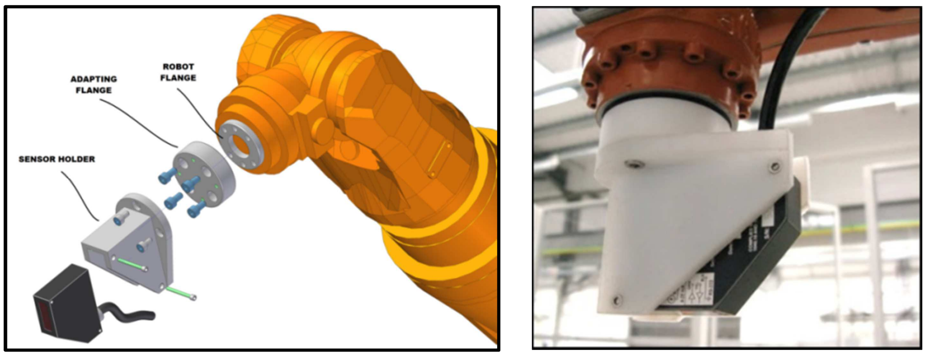


Figure 4.24 – Acuity AR200-100 laser distance meter mounted to the extremity of the robot.

The AR200 uses triangulation to measure distance [22]. The laser beam is projected from the housing and is reflected from a target surface to a collection lens. The lens focuses an image of the spot on a linear array camera. Reflected light from the surface is viewed from an angle by a CMOS line scan camera [23] inside the AR200 sensor. The camera views the measurement range from an angle that varies from 45 to 65 degrees at the centre of the measurement range, depending on the model. The position of the spot image on the pixels of the camera is then processed to determine the distance to the target. The camera integrates the light falling on it, so longer exposure times allow greater sensitivity to weak reflections. The exposure time and laser power level are controlled to optimize the accuracy of the measurements for the signal strength and environmental light level measured. If the sample rate set allows time for internal averaging of multiple exposures before transmission that is done. Exposure time and laser power are adjusted from sample to sample, so rapidly changing conditions may result in momentary loss of signal or overexposure. If the sensor cannot take another exposure before it is time to transmit a sample, a zero value will be output. AR200 sensors emit visible laser light (650nm wavelength - 1 mW red beam).

The laser sensor was established as the robot tool, setting its TCP (tool centre point) along the laser beam at the central position of the sensor span (according to dimensional specifications of sensor and holder). The selected laser distance meter used triangulation to measure distance. It had a 101.6 mm detection span with 79 mm standoff to the middle of span, a resolution of 30.5μm. The measurement laser spot size was smaller than 250μm and the maximum sampling frequency was equal to 1250Hz.

The capability to control the laser sensor via the RS232 serial communication was exploited to integrate sensor data acquisition in the same C++ server application controlling the execution of the robot tool-path (see Appendix A4). The concept of the experimental setup is given in Figure 4.25.

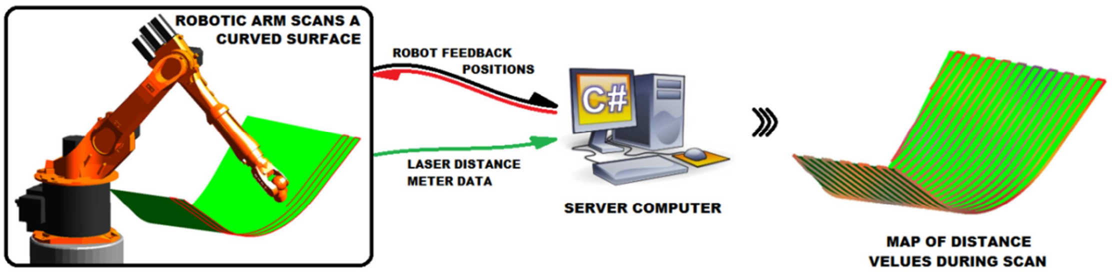

ROBOTIC ARM SCANS A CURVED SURFACE
ROBOT FEEDBACK POSITIONS
LASER DISTANCE METER DATA
C#
SERVER COMPUTER
MAP OF DISTANCE VELUES DURING SCAN

Figure 4.25 – Concept for future experimental tests.

The C++ script was made able to receive data during the execution of a robotic path programmed in a KUKA Robot Language (KRL) module or commanded via the coordinates calculated by the path-planning MATLAB toolbox. The laser distance meter data were received simultaneously to the reception of the robot feedback positions. The sensor data and the robot position feedback were time-stamped to allow successive encoding of the data through interpolated positions.

Before using the laser distance meter (manipulated by the robotic arm) to measure the distance between the robot end-effector the surface of the aerospace winglet (the sample in Figure 4.7), some initial tests were carried out to ensure the proper operation of the setup. The sensor was manipulated to develop a raster scan on a plane over some featured surfaces; it was basically used as a reverse engineering tool for surface mapping. Figure 4.26 and Figure 4.27 show the scan of a perforated and of a sinusoidal surface and the relative maps of the acquired distances.

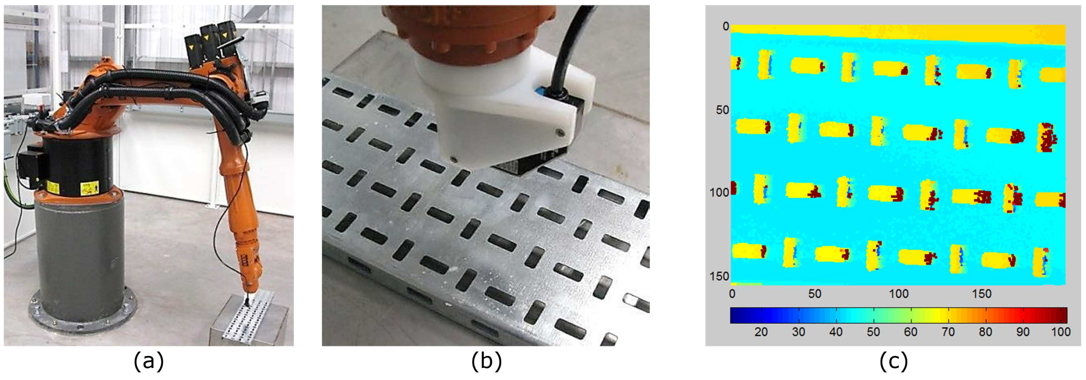


Figure 4.26 – Scanning of perforated surface (a and b) and scatter plot of the acquired data (c).

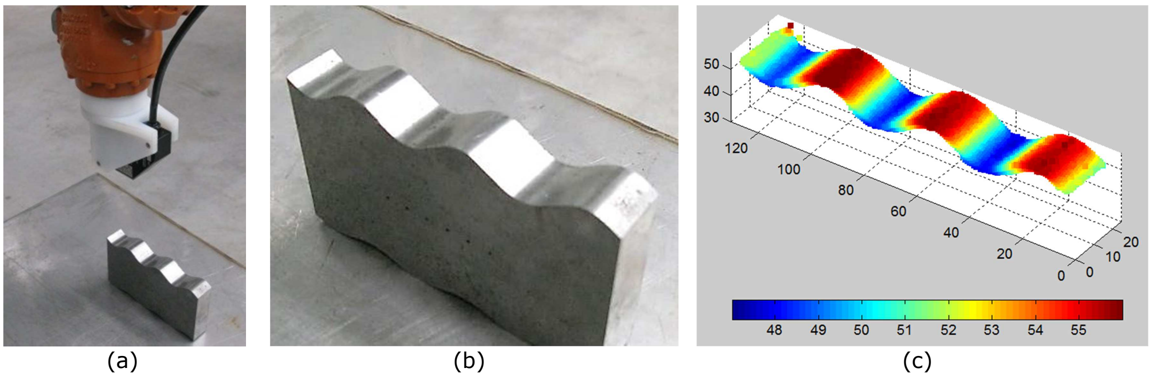


Figure 4.27 – Scan of a sinusoidal surface (a and b); scatter plot of the acquired distances (c).

Therefore the laser distance meter was used to investigate the accuracy of a tool-path, generated by the MATLAB toolbox for the wide curved surface of the aerospace winglet.

Raster scan paths were generated through the MATLAB toolbox and used to finely follow the curved contour of the target surface. The STL mesh of the sample was approximated allowing a maximum error of 0.5 mm, and three separate tool-paths were generated to travel equally spaced trajectories (10mm pitch) at constant speeds of 100, 200 and 300mm/s, respectively. The C++ server application transmitted command positions to the robot controller and received actual TCP coordinates over the Ethernet interface. The sampling rate of the laser sensor was set to 250 Hz.

Figure 4.28 shows the recorded maps of difference between measured and theoretical TCP for the 3 different robot speeds.

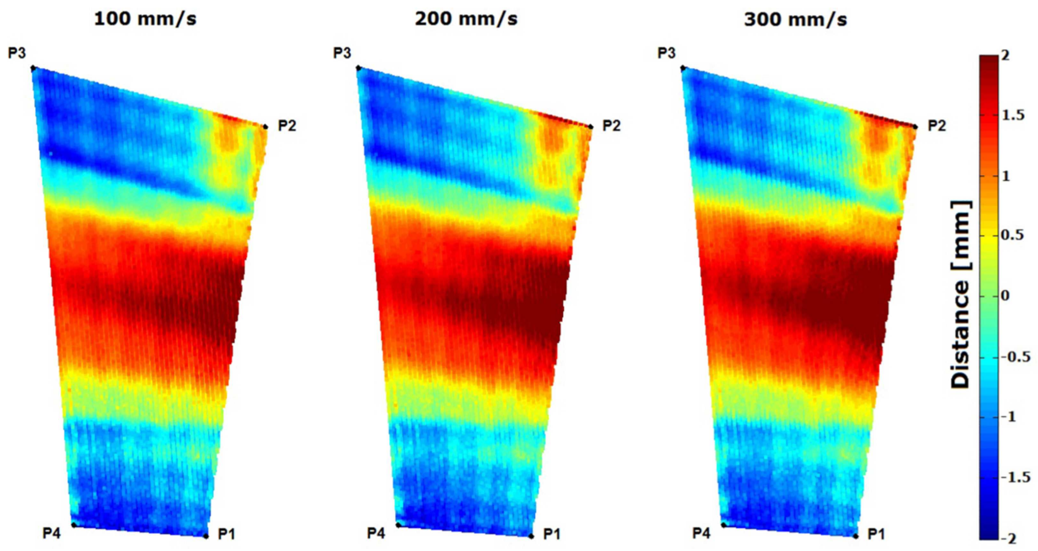


Figure 4.28 – Maps showing distance deviation from theoretical TCP for different robot speeds.

The results show that the deviation of the measured distance from the theoretical TCP spans a range between ± 2mm. This error range was deemed like a fairly good result; however it is sufficiently large to impact some NDT techniques potentially deployable through the robot manipulators. For example eddy current inspection [24] requires maintaining stand-off distance to better than 1 mm [25] for consistent results.

Therefore the possible sources of the standoff error are examined below.

1. Firstly dimensional differences are expected between the CAD manufacturing model and the manufactured part, particularly for the large composite sample as considered herein. During manufacture, as the composite part is removed from the mould, the part can exhibit a spring back effect due to fiber deviation, residual stress and strain [26]. This error can be several mm [27] and is likely the principal source of error in this experiment.
2. The next most significant source lays in the approximation of the CAD mesh with the polynomial surface used for path planning operations. The error can obviously be decreased by increasing the order of the polynomial fitting function, at the expense of increased computation time. In the present work the maximum allowed error (err_target) was set to 0.5 mm for the path-planning of the given surface, leading to a polynomial fitting function of 21st order (Figure 4.29). The numerical stability of the code was assured by monitoring the mean error. The order of the polynomial function was no longer increased if the mean error began to diverge.

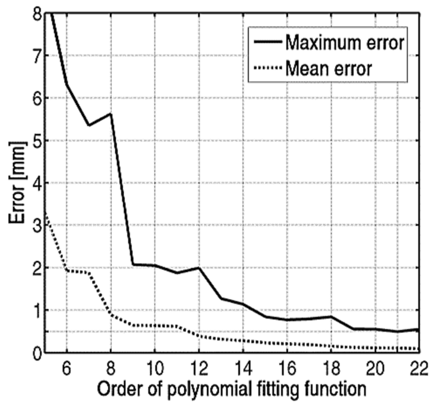


Figure 4.29 – Maximum and mean CAD approximation error against the order of the polynomial fitting function (for the given surface).

3. Finally the effects of the accuracy of the actual robot controller should be considered – the actual robot path always deviates to some degree from the instructed robot path. It was possible to directly estimate this effect for the full scan area, as the C++ application manages both the command and feedback positional data. The maps in Figure 4.30 show the error between

the command and the feedback coordinates throughout the tool-path. It is clear that the robot path error ranges between 0 and 0.6 mm and increases as end effector velocity is increased.

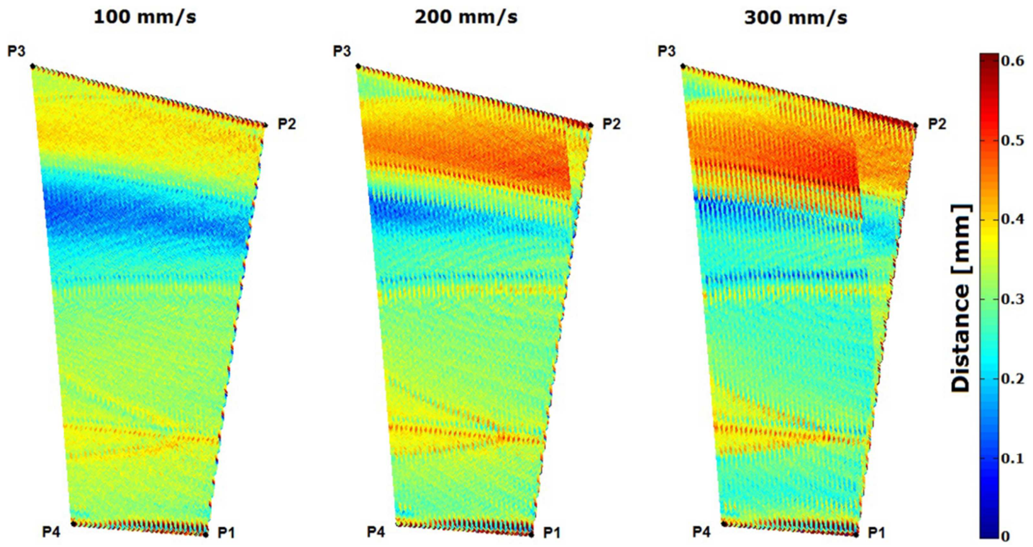


Figure 4.30 – Maps of distance between target and feedback positions for different robot speed.

So the overall supposition is that, for the considered case, the path error was dominated by the deviation between the real sample and the CAD model. To reduce this source of error, a full surface metrology scan of the part would be required for each and every part prior to the generation of NDT tool-paths.

The results obtained by experimental testing gave fundamental indications to improve the path-planning algorithms. The remainder part of the chapter presents the ultimate solution sought by this research to develop a robust and flexible approach. Crucially, it completely eliminates the error caused by approximation and it is suitable to a wider range of surfaces. The chapter ends with the description of a full kinematics model that overcomes the limitations of the open-source formulation.

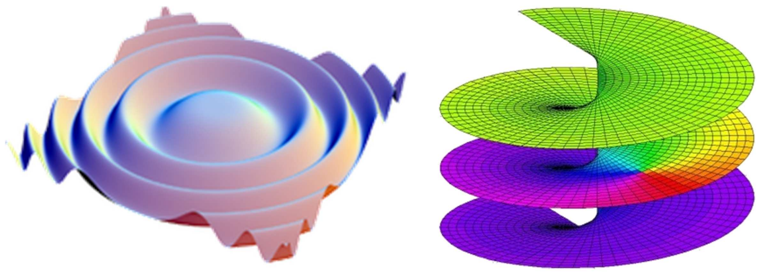
4.7. Ultimate approximation-free path-planning approach

4.7.1. Disadvantages of the analytical approach

The background path-planning algorithms implemented into the MATLAB toolbox (see Section 4.4.2) have some limitations. The limitations originate from the intrinsic nature of the path-planning approach; it is based on an analytical formulation. The tessellated surfaces are approximated with polynomial surfaces, necessary to the computation of the tool-path command points. Likewise, each portion of the tool-paths is described by polynomial formulations.

Each approximation introduces an error by definition. The error can obviously be decreased by increasing the order of the polynomial fitting function, at the expense of increasing the computation time. It was seen that to maintain the approximation error lower than 0.5 mm, for a 1.6m^2, a polynomial fitting function of 21st order was required (Figure 4.29). The numerical stability of the approximation was assured by monitoring the mean error. The order of the polynomial function was no longer increased if the mean error began to diverge.

Even more importantly, the approximation of a meshed surface with a polynomial surface is only possible when the surface can be mathematically described by a *surjective*[12] function. A surjective function, $z = f(x, y)$ with X-Y domain and codomain in Z, is surjective (or a surjection) if every element z in Z has a corresponding x-y couple such that $z = f(x, y)$. The function f may map more than one couple of X-Y to the same element of Z, but not the opposite. The inverse of a surjective function is not surjective. As a result, the approximation of a meshed surface fails if the surface is not surjective and it is influenced by the orientation of the surface in the 3D Cartesian space.


(a) (b)
Figure 4.31 – Example of surjective surface (a) and non-surjective surface (b).

4.7.2. A mesh-following approach

A new approach was investigated. Since the CAD files are imported as tessellated surfaces (meshed objects), the new approach is based on the idea that the mesh of each specific surface can be used as a guide to let the path-planning algorithm operate directly on the

[12] A function is called injective (or one-to-one, or an injection) if $f(a) \neq f(b)$ for any two different elements a and b of the domain. It is called surjective (or onto) if $f(X) = Y$. That is, it is surjective if for every element y in the codomain there is an x in the domain such that $f(x) = y$. Finally f is called bijective if it is both injective and surjective. This nomenclature was introduced by the Bourbaki group in the 20th century.

curved contour of the surface and calculate parallel curves, intersections between curves and every other geometrical entity useful to obtain flexible and robust path-planning.

The fundamental mathematical concepts of this new approach are presented below. The surface shown in Figure 4.32 was used during the development of the new path-planning algorithms. Three irregular holes were intentionally created in the mesh, in order to increase the challenge and increase possibilities to debug the codes under development.

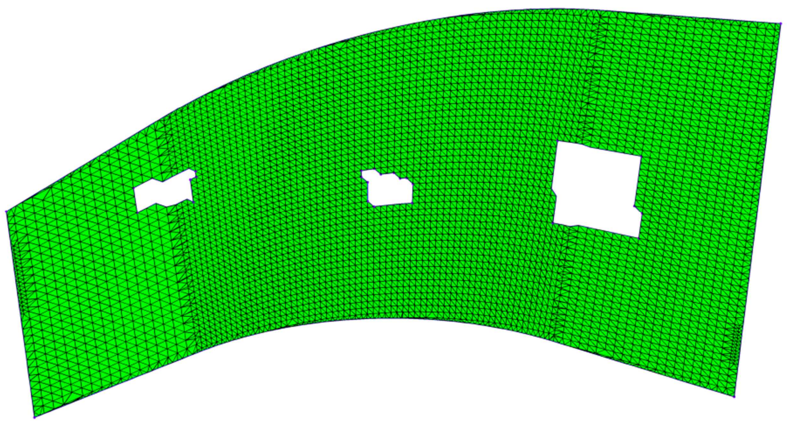


Figure 4.32 – Sample surface used for development of the path-planning algorithms.

Let us consider a reference edge of the given surface to create a parallel line at distance d. The vertical edge on the left hand side will be used for the following descriptions. The target distance between the reference edge and the new line is set to $d = 100mm$.

The reference edge consists of segments, whose extremities coincide with two corners of the adjacent triangles. Let us consider the directional vector of the i-th segment (s_i) given by:

$$\vec{v}_i = [u_i, v_i, w_i] \qquad \text{Eq. 4.35}$$

The middle point of the segment is:

$$P_i = (x_i,\ y_i,\ z_i) \qquad \text{Eq. 4.36}$$

Therefore the plane (π_i), perpendicular to the segment s_i for the point P_i, is given by the equation:

$$\pi_i : \quad (u_i \cdot x) + (v_i \cdot y) + (w_i \cdot z) + d_i = 0 \qquad \text{Eq. 4.37}$$

where d_i is equal to:

$$d_i = -(u_i \cdot x_i + v_i \cdot y_i + w_i \cdot z_i) \qquad \text{Eq. 4.38}$$

For each segment of the reference edge it is possible to find the intersection points between the calculated plane and the edges of the triangles of the surface mesh. Figure

4.33 shows the intersection points (given in black colour) originated from the plane relative to the first segment (s_1) of the reference edge and the edges of the tessellated mesh.

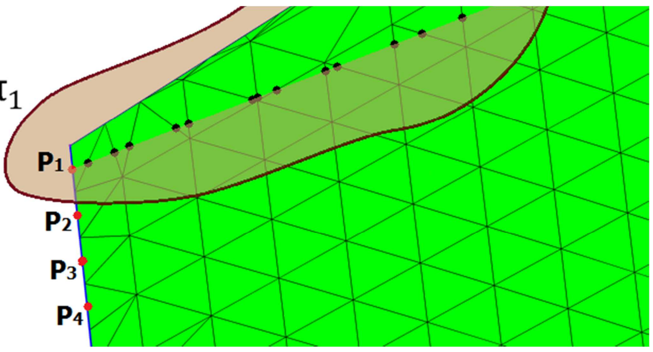


Figure 4.33 – Intersection points (given in black colour).

Starting from point P_i and following the succession of the intersection points found on the relative plane π_i, the curvilinear distance from the reference edge is calculated. The resulting distance is the proper distance along the surface contour. The integration of the distance stops when the distance exceeded the set target distance d. The farther intersection points are ignored. Since the last remaining point is farther than the set distance and the second-last point is closer, a point that is exactly at the set distance is calculated through interpolation between the two points. This point lies on one of the mesh triangles (Figure 4.34a). The process is repeated for all the segments in the reference edge (Figure 4.34b). The final net result is a curve, parallel to the selected reference edge (Figure 4.34c). Because all of the points of the curve lie on the mesh triangles, the accuracy with which the curve follows the contour of the surface is directly related to the maximum deviation between the mesh and the sample surface. This eliminates the approximation error introduced by the old approach.

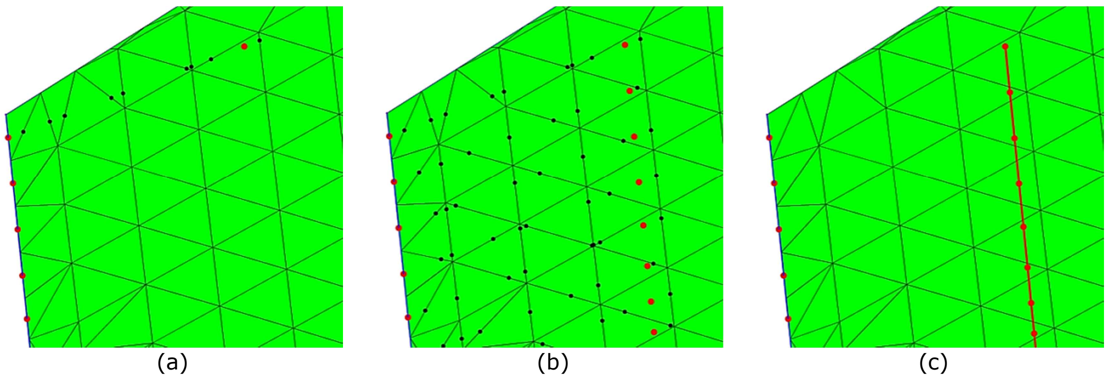


Figure 4.34 – Step-by-step process for the generation of a curve parallel to the reference edge.

After the generation of the curve parallel to the reference edge, its extremities do not intersect the surface outer boundary. This can be inconvenient when one of the requirements for the inspection scanning path is to cover 100% of the surface. An algorithm has been implemented to extend the extremity segments of the curve to the boundary of the surface. Therefore two additional points, one for each extremity, are added to the parallel curve in order to make sure its start and its end point are at the surface boundary (Figure 4.35).

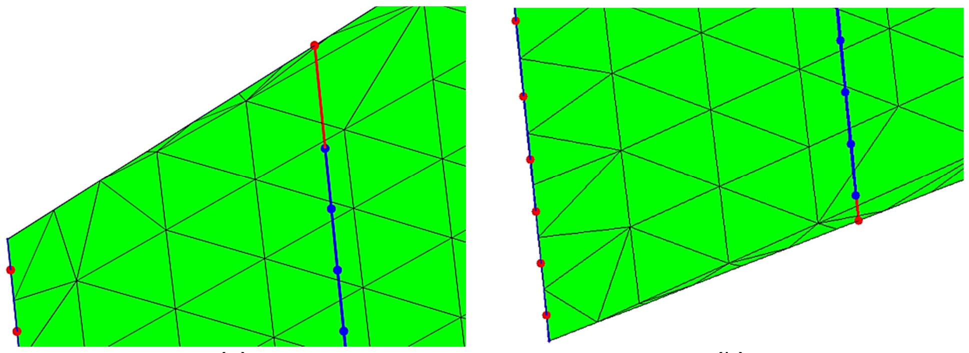


Figure 4.35 – Extension to the surface boundary. First extremity (a) and second extremity (b).

The algorithm is based on the calculation of the minimum distance between two skew lines (r and s). Let us consider the two lines containing the extremity segment (r) of the parallel curve and the neighbour segment (s) of the boundary. Each line can be identified by the start point of the relative segment and by the direction vector that links the start point to the end point:

r: $R = (x_R, y_R, z_R)$; $\overrightarrow{v_r} = (u_r, v_r, w_r)$ Eq. 4.39

s: $S = (x_S, y_S, z_S)$; $\overrightarrow{v_s} = (u_s, v_s, w_s)$ Eq. 4.40

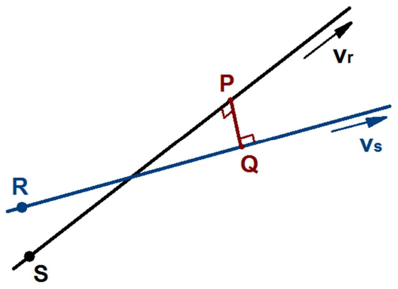


Figure 4.36 – Minimum distance between two skew lines.

Let us consider the generic segment $\overline{PQ}$ from line r to line s (see Figure 4.36). The length of $\overline{PQ}$ is equal to the minimum distance between the skew lines if it forms right angles with each of the two lines. In other words, $\overline{PQ}$ has to be perpendicular to both lines. Introducing the parameters t and h, the general coordinates of point P and Q are:

$$P = \begin{pmatrix} x_R + t \cdot u_r \\ y_R + t \cdot v_r \\ z_R + t \cdot w_r \end{pmatrix}$$ Eq. 4.41

$$Q = \begin{pmatrix} x_S + h \cdot u_s \\ y_S + h \cdot v_s \\ z_S + h \cdot w_s \end{pmatrix} \qquad \text{Eq. 4.42}$$

Therefore the segment is given by:

$$\overline{\mathrm{PQ}} = \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \qquad \text{Eq. 4.43}$$

Applying the condition of perpendicularity between $\overline{\mathrm{PQ}}$ and both lines:

$$\begin{cases} \overline{\mathrm{PQ}} \cdot \overrightarrow{v_r} = 0 \\ \overline{\mathrm{PQ}} \cdot \overrightarrow{v_s} = 0 \end{cases} \rightarrow \begin{cases} \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \cdot \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = 0 \\ \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \cdot \begin{pmatrix} u_s \\ v_s \\ w_s \end{pmatrix} = 0 \end{cases} \rightarrow \qquad \text{Eq. 4.44}$$

$$\begin{cases} (x_S + h \cdot u_s - x_R - t \cdot u_r) \cdot u_r + (y_S + h \cdot v_s - y_R - t \cdot v_r) \cdot v_r + (z_S + h \cdot w_s - z_R - t \cdot w_r) \cdot w_r = 0 \\ (x_S + h \cdot u_s - x_R - t \cdot u_r) \cdot u_s + (y_S + h \cdot v_s - y_R - t \cdot v_r) \cdot v_s + (z_S + h \cdot w_s - z_R - t \cdot w_r) \cdot w_s = 0 \end{cases}.$$

The solution of the system of equations gives the values of the parameters t and h. The substitution of h into Eq. 4.42 gives the coordinates of point Q. This point belongs to the boundary of the surface and can be used to extend the extremity of the parallel line.

Once the parallel curve crosses the meshed surface from side to side, the path-planning algorithm finds the triangles of the mesh crossed by the curve (Figure 4.37a).

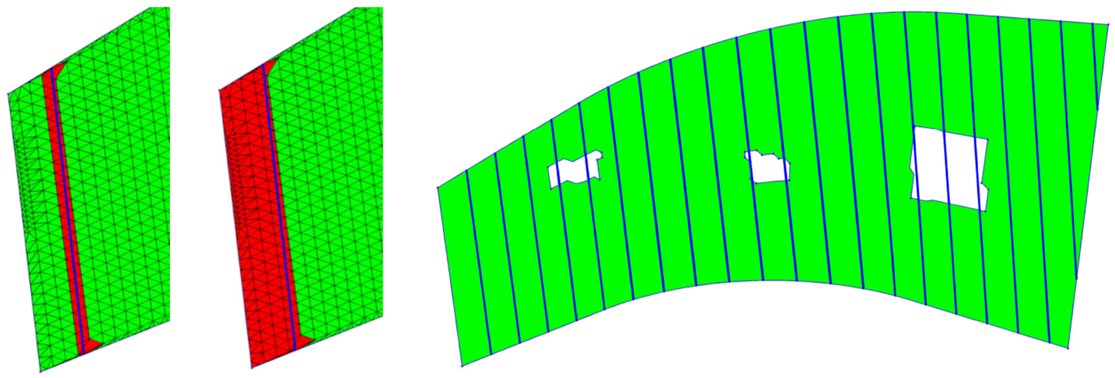


Figure 4.37 – Triangles crossed by the parallel curve (a), meshed region to be excluded (b) and parallel curves given by the iteration of the algorithm (c).

For the creation of raster scan paths it is important to iterate the algorithm described so far to generate other parallel lines, equally spaced, covering the entirety of the meshed surface. The row of crossed triangles, shown in Figure 4.37a, divides the mesh in two regions: the region that has already been swept, between the reference edge and the generated curve, and the remaining part of the mesh. The former region is identified (Figure 4.37b) and excluded from the domain of interest for the iteration of the algorithm.

The generated curve is then used as reference edge to compute the next parallel curve. Figure 4.37c shows the result of the iteration of the algorithm to reach the entire coverage of the meshed surface.

Once the parallel curves are generated, the final objective is relative to the calculation of the direction perpendicular to the mesh for each point of the curves. Since every point of the obtained curves lies on one (and only one) of the triangles of the mesh, the perpendicular direction associated with each point is given by the vector normal to the relative triangle. The normal vectors associated to the generated curves for the given surface are shown in Figure 4.38. The portions of the curves that travel across the holes of the mesh are recognised. The intersection points between the curves and the inner boundaries of the surface are also computed through the previously described method, based on the minimum distance between skew lines.

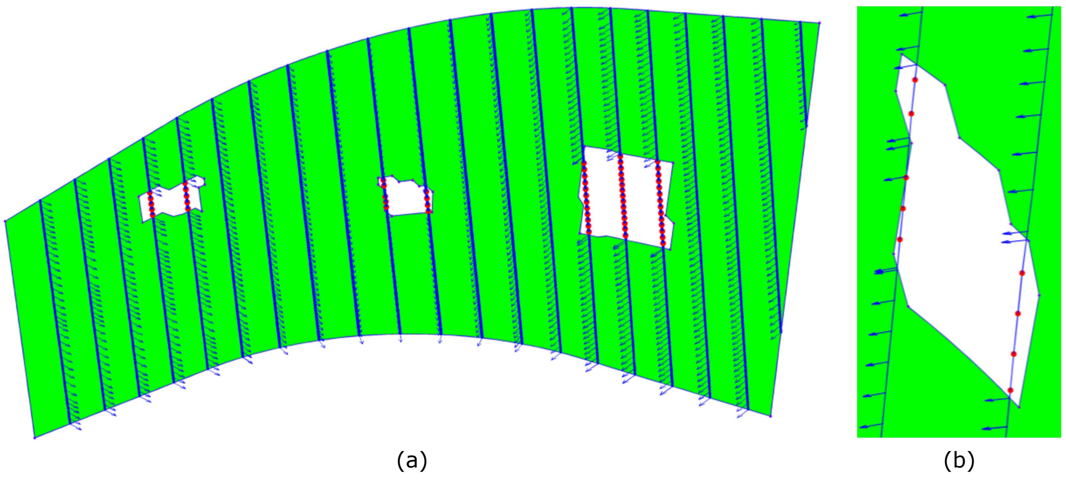


Figure 4.38 – Parallel trajectories complete of normals to the surface (a) and points of intersection with the inner boundary of the middle hole (b).

4.7.3. Testing the reliability and the flexibility

For the sake of testing the path-planning algorithm, it was put within a loop and executed for 100 times. A different set of input parameters was selected at each one of the iterations. This testing procedure allowed debugging of the code to obtain a reliable working status of the algorithms.

Figure 4.39 shows 12 of the generated random tool-paths, obtained at the end of the optimization period.

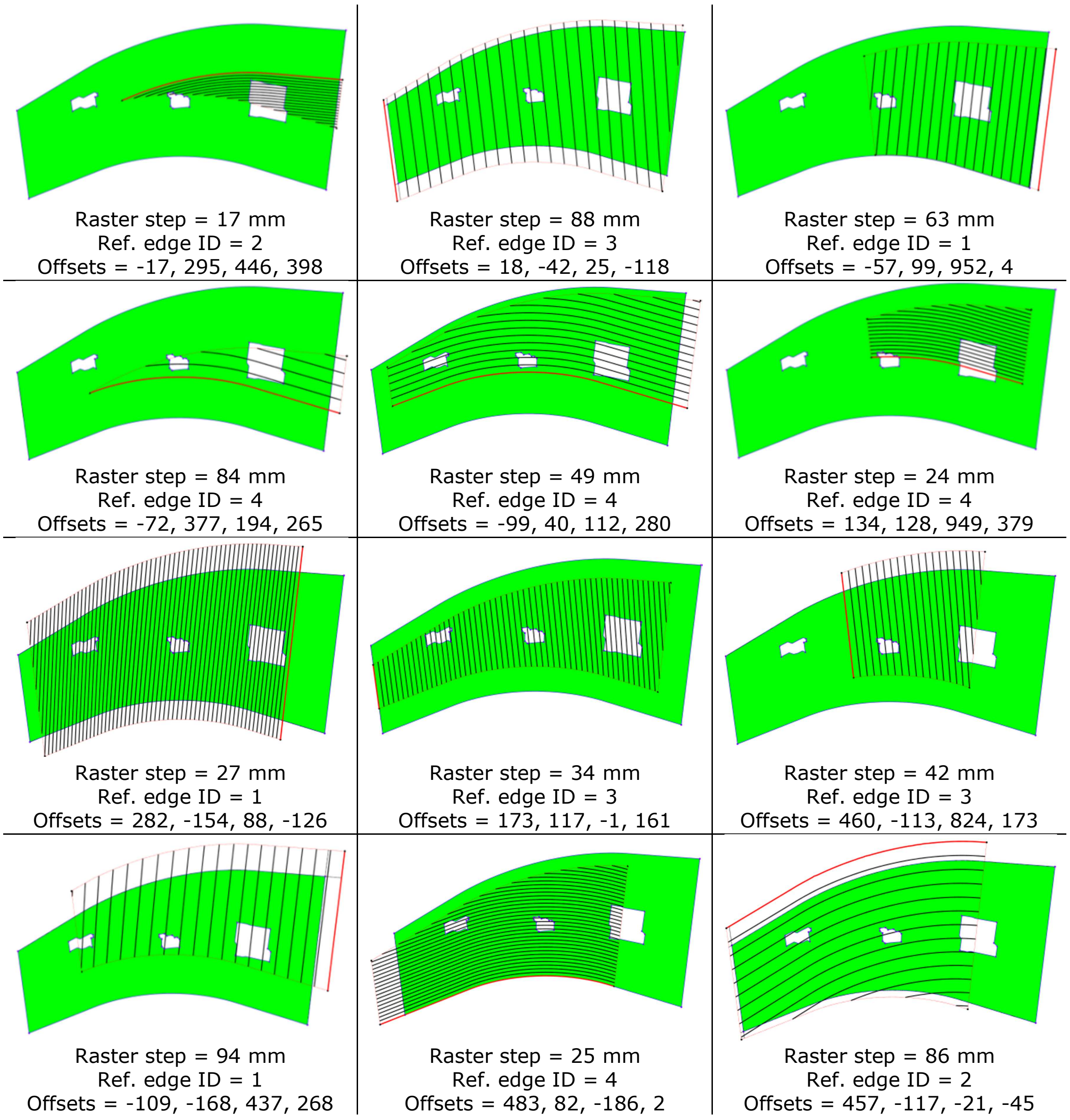


Figure 4.39 – Some of the generated raster tool-paths.

4.8. Inverse kinematics for simulations

4.8.1. Disadvantages of the open-source function

Section 4.5.3 described the development of the simulation module of the path-planning toolbox that was initially based on a function of an existing open-source function of the KUKA Control Toolbox (KCT) [21].

However, two fundamental problems related to this adopted formulation were found:

- Inaccurate matching between the simulated TCP of robot and the target position;
- Impossibility in selecting the robot configuration (a six-axis robot can theoretically reach any point within its working envelope using 8 different configurations of the joints).

These inconveniences make the KCT function unsuitable to be used for the development of acceptable off-line path-planning software. An attempt was made to correct the behaviour of the function inherited from the KUKA Control Toolbox, but the reason for the inaccurate positioning was not completely clear. Moreover, the structure of the function did not allow an easy update to support all the possible configurations of the robot kinematics.

4.8.2. A full solution to the kinematics formulation

A new formulation was developed from scratch by the author of this thesis to overcome the aforementioned inconveniences. The resultant algorithm is described herein. The inverse kinematics formulation is based on a geometric approach [28].

For a given Cartesian position ([x, y, z, A, B, C]) and orientation of the TCP ([x_t, y_t, z_t, A_t, B_t, C_t]), it is possible to use a method that calculates the joint angles for a six axis robot. For a single TCP position there are 8 possible configurations of the robot joints (see Figure 4.40).

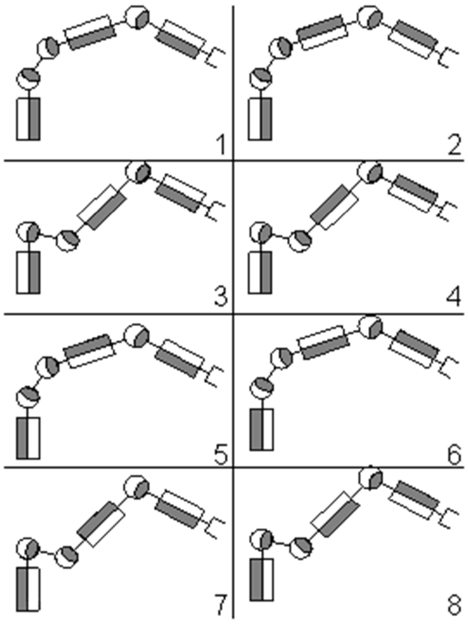


Figure 4.40 – Possible solutions of the inverse kinematics calculation.

4.8.2.1. Joint 1

The angle Θ_1 can be calculated from the projection of frame K_4 on the x-z plane of frame K_0 (see Figure 4.41). Hence at first it is necessary to compute the position of K_4. Twist joints do not change the position but only the orientation of the subsequent frame. This property simplifies the calculations. The three twist joints are: Joint 1, 4 and 6.

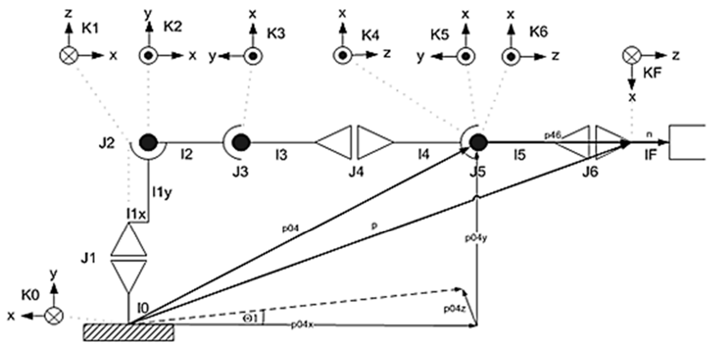


Figure 4.41 – Joint angle 1.

As shown in Figure 4.40, the angular state of joint 6 influences just the orientation of the TCP, not its position. Through the equation $\vec{p}_{46} = (l_5 \cdot \vec{n}) + (R \cdot \vec{t})$, we get the vector $\vec{p}_{46}$. The unit vector $\vec{n}$ points towards the z-axis of K_6. It is retrieved from the third column of the rotation matrix R. This matrix is the same of the matrix in Eq. 4.13, but it needs to be calculated as a function of the angular coordinates of the robot pose (A, B and C) and the angles of the tool frame (A_t, B_t and C_t). Using the abbreviation s for sine and c for cosine, the matrix R is given by:

$$R_C = \begin{bmatrix} cA \cdot cB & cA \cdot sB \cdot sC - sA \cdot sC & cA \cdot sB \cdot cC + sA \cdot sC \\ sA \cdot cB & sA \cdot sB \cdot sC + cA \cdot sC & sA \cdot sB \cdot cC - cA \cdot sC \\ -sB & cB \cdot sC & cB \cdot cC \end{bmatrix} \qquad \text{Eq. 4.45}$$

$$R_T = \begin{bmatrix} cA_t \cdot cB_t & cA_t \cdot sB_t \cdot sC_t - sA_t \cdot sC_t & cA_t \cdot sB_t \cdot cC_t + sA_t \cdot sC_t \\ sA_t \cdot cB_t & sA_t \cdot sB_t \cdot sC_t + cA_t \cdot sC_t & sA_t \cdot sB_t \cdot cC_t - cA_t \cdot sC_t \\ -sB_t & cB_t \cdot sC_t & cB_t \cdot cC_t \end{bmatrix} \qquad \text{Eq. 4.46}$$

$$R = R_C \cdot R_T \qquad \text{Eq. 4.47}$$

Therefore:

$$\vec{n} = R(:,3) \qquad \text{Eq. 4.48}$$

The vectors $\vec{p}$ and $\vec{t}$ are equal to:

$$\vec{p} = [x \quad y \quad z]^T \qquad \text{Eq. 4.49}$$

$$\vec{t} = [x_t \quad y_t \quad z_t]^T \qquad \text{Eq. 4.50}$$

The first is the vector of the Cartesian coordinates of the target position and the second is the vector of the tool's Cartesian parameters.

Therefore, it results:

$$\vec{p}_{04} = \vec{p} - \vec{p}_{45} - \vec{p}_{56} = \vec{p} - \left[(l_5 \cdot \vec{n}) + (R \cdot \vec{t})\right]. \qquad \text{Eq. 4.51}$$

The two solutions θ_1 and $(\theta_1 + \pi)$ give the same position for K_4. Therefore we get all 8 configurations for Joint 1:

$$\theta_{1,1} = \theta_{1,2} = \theta_{1,3} = \theta_{1,4} = \text{atan2}(-p_{04,z}\,, p_{04,x}) \qquad \text{Eq. 4.52}$$
$$\theta_{1,5} = \theta_{1,6} = \theta_{1,7} = \theta_{1,8} = \text{atan2}(-p_{04,z}\,, p_{04,x}) + \pi$$

4.8.2.2. Joint 3

Again we take advantage of a twist joint property. As shown in Figure 4.42, the angular state of joint 4 influences the orientation of frame K_4, not its position. Hence vector $\vec{p}_{14}$ can be calculated.

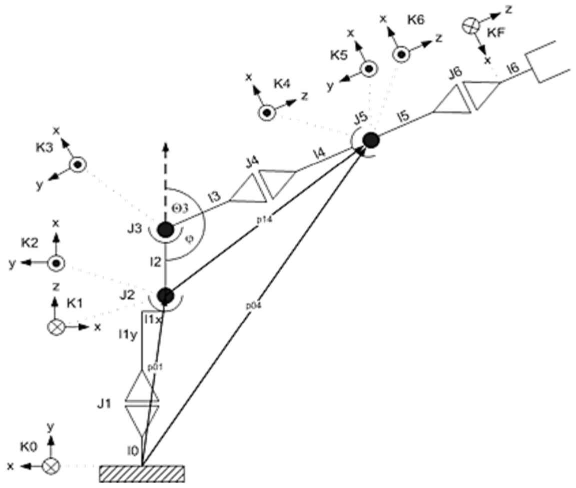


Figure 4.42 – Joint angle 3.

Through the knowledge of $\vec{p}_{14}$, we get angle φ which leads to joint angle θ_3. We obtain the position $\vec{p}_{01}$ from:

$$\vec{p}_{01} = R_1 \cdot [l_{1x} \quad 0 \quad l_{1y}]^T \qquad \text{Eq. 4.53}$$

where:

$$R_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad \text{Eq. 4.54}$$

Thus, $\vec{p}_{14} = \vec{p}_{04} - \vec{p}_{01}$.

We compute angle φ through the *law of cosine*:

$$\varphi = \text{acos}\left(\frac{l_2^2 + (l_3 + l_4)^2 - |\vec{p}_{14}|^2}{2 \cdot l_2 \cdot (l_3 + l_4)}\right) \qquad \text{Eq. 4.55}$$

Finally, the knowledge of φ offers four possible solutions for θ_3:

$$\theta_{3,1} = \theta_{3,2} = \pi - \varphi + \alpha \qquad \text{Eq. 4.56}$$
$$\theta_{3,3} = \theta_{3,4} = \pi + \varphi + \alpha$$
$$\theta_{3,5} = \theta_{3,6} = -(\pi - \varphi) + \alpha$$
$$\theta_{3,7} = \theta_{3,8} = -(\pi + \varphi) + \alpha$$

where is $\alpha = atan2(-l_{34,z}, l_{34,x})$.

4.8.2.3. Joint 2

As shown in Figure 4.43, angle θ_2 can be computed through β_1 and β_2 which in turn we get through the vector $\vec{p}_{14}$. We obtain β_1 from:

$$\beta_1 = atan2\left(p_{14,z}, \sqrt{p_{14,x}^2 + p_{14,y}^2}\right), \qquad \text{Eq. 4.57}$$

and β_2 from the law of cosine:

$$\beta_2 = acos\left(\frac{l_2^2 + |\vec{p}_{14}|^2 - (l_3 + l_4)^2}{2 \cdot l_2 \cdot |\vec{p}_{14}|}\right) \qquad \text{Eq. 4.58}$$

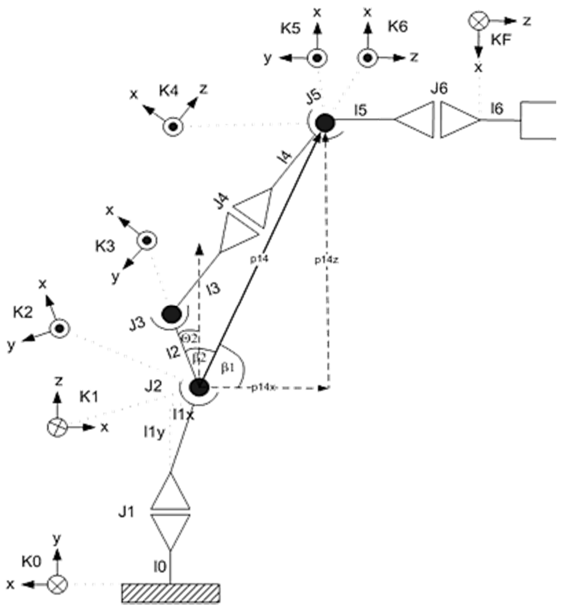


Figure 4.43 – Joint angle 2.

Finally all solutions for θ_2 come out from:

$$\theta_{2,1} = \theta_{2,2} = -(\beta_1 + \beta_2) \qquad \text{Eq. 4.59}$$
$$\theta_{2,3} = \theta_{2,4} = -(\beta_1 - \beta_2)$$
$$\theta_{2,5} = \theta_{2,6} = \beta_1 + \beta_2 + \pi$$
$$\theta_{2,7} = \theta_{2,8} = \beta_1 + \beta_2 - \pi.$$

4.8.2.4. Joint 4

In order to calculate joint angle 4, θ_4, we need again the vector $\vec{p}_{04}$ and a unitary vector $\vec{v}$, lying on the plane perpendicular to joint 3 for P_4. This point originates from the projected direction of joint 5. The vector $\vec{v}$ is given by:

$$\vec{v} = \mathrm{inv}(A) * \vec{b} \qquad \text{Eq. 4.60}$$

where, naming $\vec{n}_3$ the unit vector giving the direction of joint 3, the matrix A is:

$$A = \begin{bmatrix} n_{3,x} & n_{3,y} & n_{3,z} \\ n_{3,z} & 0 & -n_{3,x} \\ 0 & n_{3,z} & -n_{3,y} \end{bmatrix} \qquad \text{Eq. 4.61}$$

and the vector $\vec{b}$ can be calculated from:

$$\vec{p}_{05} = \vec{p} - \vec{p}_{56} = \vec{p} - (R \cdot \vec{t}) \qquad \text{Eq. 4.62}$$

$$\vec{b} = \begin{pmatrix} n_{3,x} \cdot p_{04,x} + n_{3,y} \cdot p_{04,y} + n_{3,z} \cdot p_{04,z} \\ n_{3,z} \cdot p_{05,x} - n_{3,x} \cdot p_{05,z} \\ n_{3,z} \cdot p_{05,y} - n_{3,y} \cdot p_{05,z} \end{pmatrix} \qquad \text{Eq. 4.63}$$

The coordinates of the vectors $\vec{p}_{04}$ and $\vec{v}$ can be transformed to be relative to the plane perpendicular to joint 3 for P_4:

$$\vec{p}_{04}^{(3)} = (R_1 * R_2 * R_3)^T * \vec{p}_{04} \qquad \text{Eq. 4.64}$$

$$\vec{v}^{(3)} = (R_1 * R_2 * R_3)^T * \vec{v} \qquad \text{Eq. 4.65}$$

Finally all solutions for θ_4 come out from:

$$\theta_{4,1} = \theta_{4,3} = \theta_{4,6} = \theta_{4,8} = \mathrm{atan2}\left(v_y^{(3)}, p_{04,z}^{(3)} - v_z^{(3)}\right) \qquad \text{Eq. 4.66}$$

$$\theta_{4,2} = \theta_{4,4} = \theta_{4,5} = \theta_{4,7} = \mathrm{atan2}\left(v_y^{(3)}, p_{04,z}^{(3)} - v_z^{(3)}\right) + \pi$$

4.8.2.5. Joint 5

Joint angle 5 can be calculated transforming $\vec{p}_{04}$ and $\vec{p}_{05}$ to make them relative to the frame K_4:

$$\vec{p}_{04}^{(4)} = (R_1 * R_2 * R_3 * R_4)^T * \vec{p}_{04} \qquad \text{Eq. 4.67}$$

$$\vec{p}_{05}^{(4)} = (R_1 * R_2 * R_3 * R_4)^T * \vec{p}_{05} \qquad \text{Eq. 4.68}$$

Therefore, the solutions for θ_5 are:

$$\theta_{5,1-8} = -\mathrm{atan2}\left(p_{05,z}^{(4)} - p_{04,z}^{(4)}, p_{05,x}^{(4)} - p_{04,x}^{(4)}\right) \qquad \text{Eq. 4.69}$$

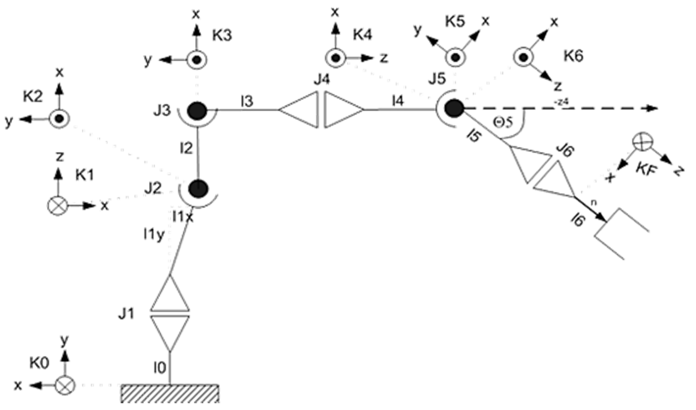


Figure 4.44 – Joint angle 5.

4.8.2.6. Joint 6

Finally, joint angle 6 is obtained from the unit vector of the z-axis of frame K_5, transformed to the frame K_6 (the TCP reference system).

$$R^{(5)} = R_1 * R_2 * R_3 * R_4 * R_5 \qquad \text{Eq. 4.70}$$

$$\vec{n}_x = (R)^T * R^{(5)}(:,3) \qquad \text{Eq. 4.71}$$

Thus:

$$\theta_{6,1} = \theta_{6,3} = \theta_{6,5} = \theta_{6,7} = -\mathrm{atan2}(n_{x,y}, n_{x,x}) \qquad \text{Eq. 4.72}$$

$$\theta_{6,2} = \theta_{6,4} = \theta_{6,6} = \theta_{6,8} = -\mathrm{atan2}(n_{x,y}, n_{x,x}) + \pi$$

4.8.3. Selection of the most appropriate configuration

By way of example, Figure 4.45 shows a KR16 L6-2 robot model keeping the TCP in the target position through all the 8 possible configurations.

The knowledge of the correct inverse kinematics formulation is fundamental to allow external controlling of the robot arm through the axis coordinates. This provides a pathway for the avoidance of singularity problems and let the user decide which kinematic configuration is most suitable to the execution of a certain robot task.

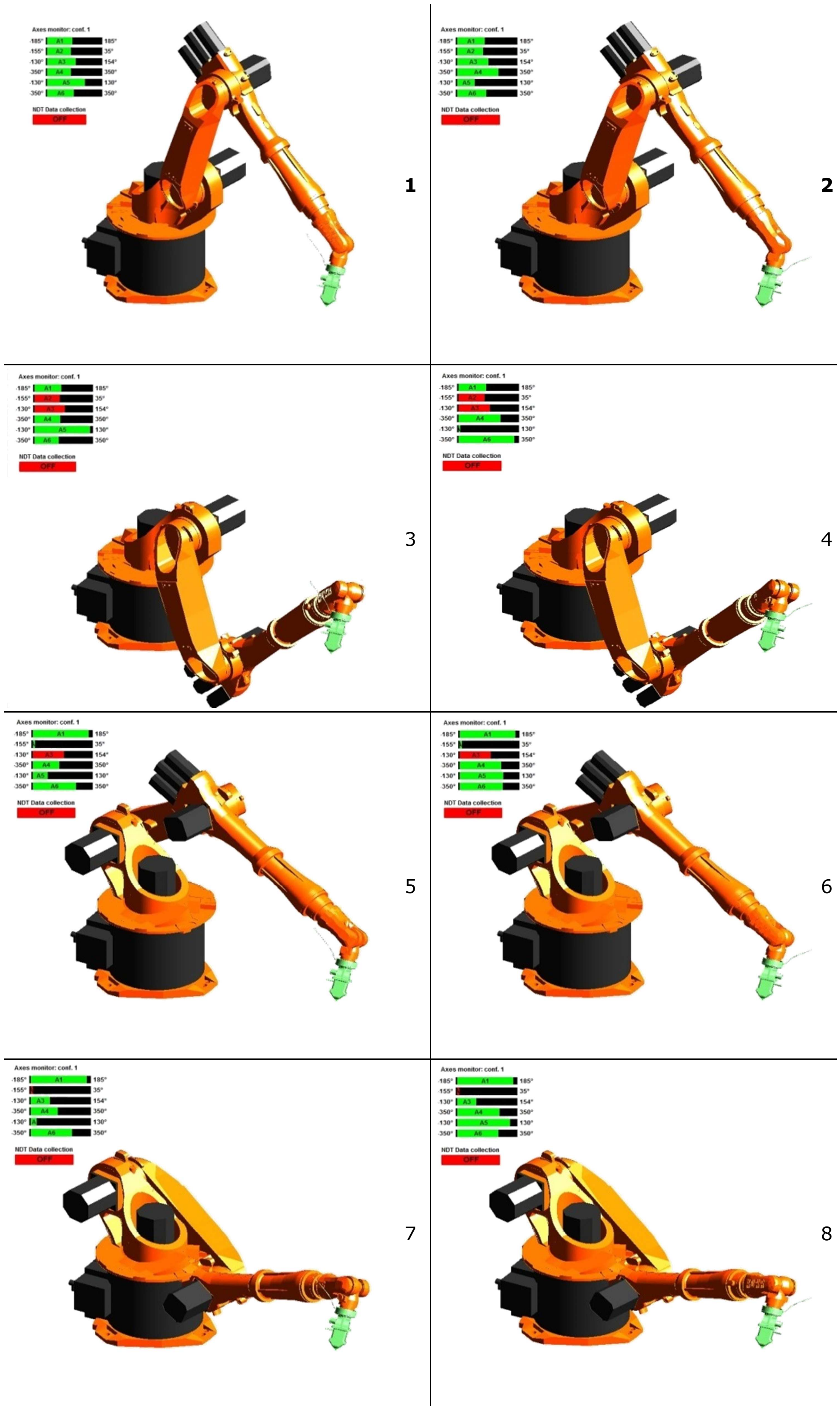


Figure 4.45 – KRC16 L6-2 reaching a target point in 8 different configurations.

When the robot is controlled through Cartesian coordinates, instead, the robot controller automatically decides the appropriate configuration to be used to reach a commanded position. Since the robot can change its configuration only whilst executing a point-to-point (PTP) movement, the robot controller selects the configuration in the initial PTP motion that links the HOME position to the first point of the approaching motion. This selection is based on the comparison of how "comfortable" the robot arm is for each of the configurations, once the commanded position is reached. The most comfortable configuration is chosen.

In order to replicate the behaviour of the robot controller, the following formulation of the so called Inconvenience Factor (IF) was implemented within the simulation function of the path-planning toolbox:

$$\mathrm{IF} = \sum_{1}^{6} \left| \frac{\theta_i - \theta_i^{LL}}{\theta_i^{UL} - \theta_i^{LL}} - \frac{1}{2} \right| \qquad \text{Eq. 4.73}$$

θ_i, θ_i^{LL} and θ_i^{UL} are respectively the angular coordinate, the lower and upper limits of the possible angle range for the i-th axis. This factor is a factor of "inconvenience", because the factor increases when one or more axis angles get close to their lower or upper angular limits.

When all axis angles are in the middle of their possible range IF is equal to 0 ($\mathrm{IF} = 0$). The value of the factor becomes bigger when the angles of one or more joints get closer to the lower or upper range limits. The maximum value of IF is equal to 6. The inconvenience factor is calculated for each possible configuration and the configuration that produces the minimum IF is chosen for the simulation of the generated tool-path. Even if all configurations are possible from a mathematical point of view, some of them might be impossible rather than inconvenient. This happens when one or more axes are out of range. The impossible configurations are excluded.

For the robot position in Figure 4.45, the IF of the 8 configurations is given in Table 4.3.

Configuration	Inconvenience Factor (IF)
1	0.37
2	0.52
3	1.82
4	1.96
5	2.00
6	2.15

Table 4.3 – Inconvenience factor for the 8 configurations in Fig 34.

Four configurations, from the 3rd to the 6th, are practically unavailable because of one or more joint angles being out of range. Therefore only the 1st and the 2nd configuration are valid and the 1st is the most convenient to use.

The simulation function (*F_ShowDemo*) chooses the most convenient configuration according to this approach.

The investigations undertaken to develop a way to select the most convenient configuration provided a deeper understanding of the inverse kinematics matter. For example, it is now clear why the initial function from the KCT toolbox was not suitable to be used for our scopes. The KCT function chooses the arm configuration on the basis of a different quality factor that takes into account the previous joint angle values ($\tilde{\theta}_1$, $\tilde{\theta}_2$, … , $\tilde{\theta}_6$). It uses the follow quality factor:

$$\varphi = \sum_{1}^{6} |\theta_i - \tilde{\theta}_i| \qquad \text{Eq. 4.74}$$

The arm configuration with the smallest quality factor value becomes the new target arm configuration. This would work fine for a succession of point-to-point (PTP) motions between distant points, but it is not suitable for the external control of inspection tool-paths where we want to maintain a constant configuration during the inspection task. Changing from one to another configuration in a single interpolation cycle of the robot controller would be a very fast and unpredictable movement that has to be avoided. This does not mean that changing configuration within one robot task cannot be addressed, but it needs to be managed outside the execution of the RSI functions responsible for real-time path-correction.

It is worth highlighting that, in the calculation of the most suitable configuration, the robot dynamics and the joint torques were not considered; this was justified by the fact that robotic arms used for NDT inspections usually move at slow speeds (in comparison to their maximum nominal speeds). However, future development of this work may involve the implementation of algorithms to monitor the dynamics features of the generated tool-paths. Such algorithms are based on recursive formulas involving quantities called articulated-body inertias, which represent the inertia properties of rigid bodies connected together by joints allowing constrained relative motion between the bodies [29].

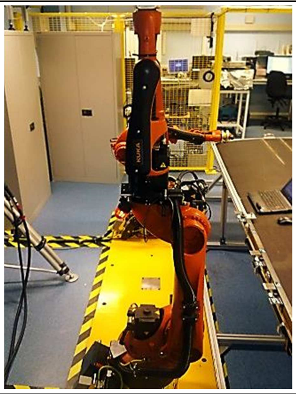
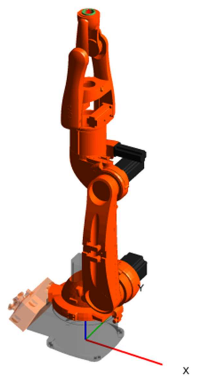
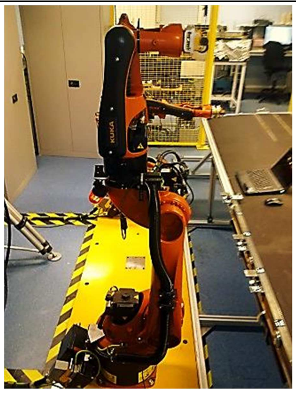
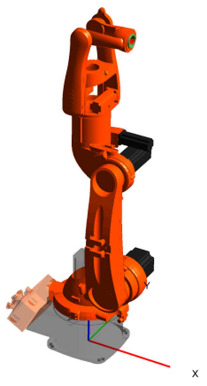
4.8.4. Experimental testing

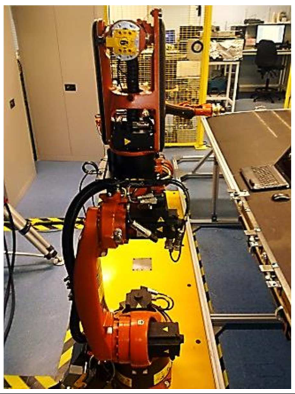
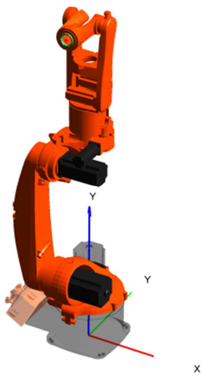
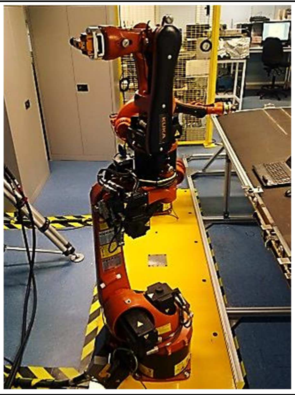
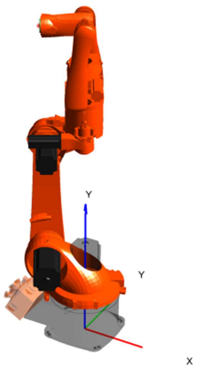
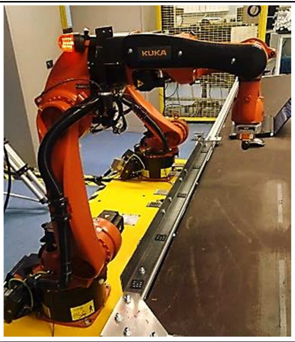
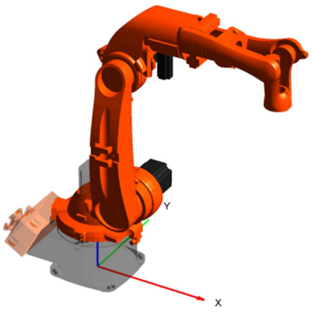
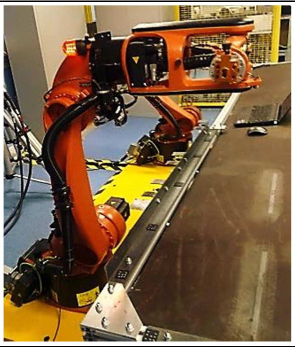
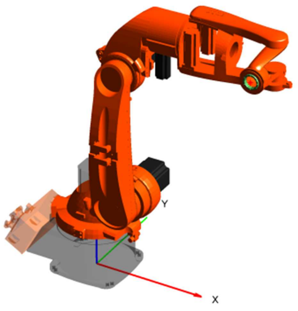
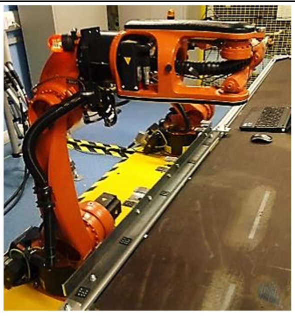
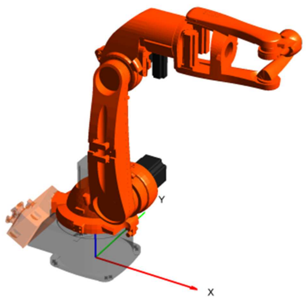
Experimental testing was used to validate the new inverse kinematics formulation and confirm that the simulated configurations always mirror the real behaviour of the robot arm. The usage of the IF factor to predict the behaviour of the robot proved to be a robust method.

Nine sets of robot Cartesian coordinates and their relative axis coordinates were collected through jogging one of KR5arcHW robots at the University of Strathclyde. They are given in Table 4.4. The output of the inverse kinematics function was compared against the collected coordinates. The inverse kinematics function was applied to the KR5arcHW robot model. All the nine sets of axis coordinates were correctly output when the relative Cartesian coordinates were used as inputs. Figure 4.46 shows the photographs of the real poses and the virtual models produced by the tested inverse kinematics formulation.

Pose	**Cartesian Coordinates**						**Axis Coordinates**					
	X	Y	Z	A	B	C	A1	A2	A3	A4	A5	A6
1	10	0	1820	0	0	0	0	-90	0	0	0	0
2	210	0	1620	0	90	0	0	-90	0	0	90	0
3	0	-210	1620	0	90	90	90	-90	0	0	90	0
4	-181.9	-105	1620	0	90	150	150	-90	0	0	90	0
5	800	0	970	-180	0	-180	0	-90	90	0	90	0
6	800	-200	1170	-180	0	-90	0	-90	90	90	90	0
7	800	200	1170	180	0	90	0	-90	90	-90	90	0
8	-200	800	1170	-90	0	90	-90	-90	90	-90	90	0
9	200	800	1170	90	0	90	-90	-90	90	-90	-90	0

Table 4.4 – Nine couple of relative Cartesian and axis coordinates collected jogging a KR5arcHW robot at the University of Strathclyde.

	Lab photos	**Virtual model pose**
1		
2		

3		
4		
5		
6		
7		

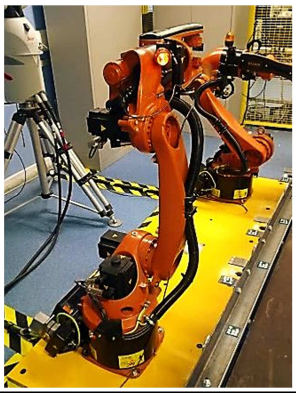
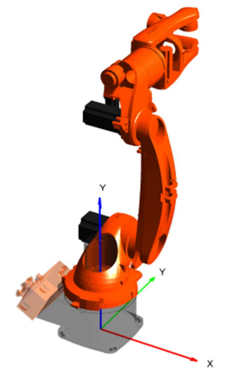
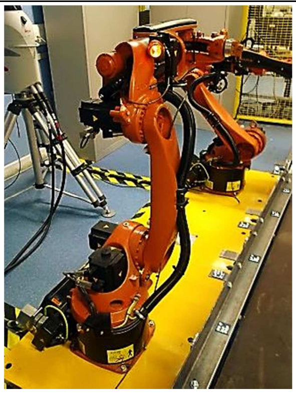
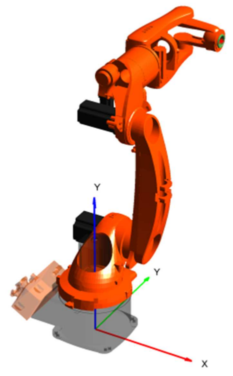
8		
9		

Figure 4.46 – Comparison between real poses and virtual replications.

The obtained robust inverse kinematics formulation can be very useful in the near future. Although it was initially developed to enable toolpath simulations, it can be used in order to compute axis command coordinates to drive the robots in axial mode.

4.9. Wrapping-up

The analysis of commercially available path-planning solutions showed the deficiencies of the current solutions. They make current OLP software not very suitable to be used in the field of robotic NDT. This chapter presented the development of path-planning tools. Extensive research was embraced to investigate the possibility to solve the deficiencies of the commercial systems. The developed path-planning toolbox provided a path-way to generate tool-paths executable through externally controlled robot arms.

A precision non-contact laser distance measurement probe was mounted to the end effector of the KUKA KR16 L6-2 robotic arm, in order to evaluate the accuracy of the tool-paths generated by the MATLAB toolbox. This allowed monitoring the probe separation from the sample curved surface during a full surface scan. The results showed that the deviation from the theoretical TCP spans a 4mm range. The sources of the standoff error were analysed. For the executed tool-paths, the deviation of the robot feedback positions from the commanded positions ranged between 0 and 0.6mm and increased with increasing robot speed.

Following the indications given by the experimental results, a novel path-planning algorithm was developed to increase robustness and flexibility and remove the error associated with the analytical approximation of the surfaces.

Moreover, the chapter presented the solution to key challenges, like the establishment of an automatic approach to transfer instructions to the robots for the execution of approaching and retracting movements. An inverse kinematics formulation was developed to simulate the real execution of the generated tool-paths. Crucially, this formulation allows selecting the best robot configuration to deliver specific tasks and it is capable of supporting the external control of robotic arms in axis mode; a viable way to overcome singularity problems.

References

[1] A. Boud and S. Steiner, "A new method for off-line robot programming: Applications and limitations using a virtual environment," in *Factory 2000-The Technology Exploitation Process, Fifth International Conference on (Conf. Publ. No. 435)*, 1997, pp. 450-455.

[2] A. Hamzah, "Feasibility study on robot off-line programming and simulation using matlab tools: simmechanics and simulink packages," Universiti Malaysia Pahang, 2004.

[3] V. D. Hunt, *Industrial robotics handbook*: Industrial Press Inc., 1983.

[4] Y. Huang and C. Lee, "An automatic assembly planning system," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 1990, pp. 1594-1599.

[5] C. T. Leondes, *Computer-Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications, Volume II, Computer-Integrated Manufacturing* vol. 2: CRC Press, 2000.

[6] F. Lewis and A. Et, "Robotics Mechanical Engineering Handbook. Ed. Frank Kreith," ed: Boca Raton: CRC Press LLC, 1999.

[7] E. Roos and A. Behrens, "Off-line programming of industrial robots—Adaptation of simulated user programs to the real environment," *Computers in industry,* vol. 33, pp. 139-150, 1997.

[8] B. Hodges and P. Hallam, *Industrial robotics*: Newnes, 1992.

[9] P. Louviot, A. Tachattahte, and D. Garnier, "Robotised UT Transmission NDT of Composite Complex Shaped Parts," presented at the 4th International Symposium on NDT in Aerospace, Berlin (Germany), 2012.

[10] P. I. Corke, "A robotics toolbox for MATLAB," *Robotics & Automation Magazine, IEEE,* vol. 3, pp. 24-32, 1996.

[11] K. Yoshida, "The SpaceDyn: a MATLAB toolbox for space and mobile robots," in *Proceedings 1999 IEEE/RSJ International Conference on intelligent robots and systems*, Kyongju, Korea, 1999, pp. 1633-1638.

[12] A. Breijs, B. Klaassens, and R. Babuška, "Automated design environment for serial industrial manipulators," *Industrial Robot: An International Journal,* vol. 32, pp. 32-34, 2005.

[13] G. L. Mariottini and D. Prattichizzo, "EGT for multiple view geometry and visual servoing: robotics vision with pinhole and panoramic cameras," *Robotics & Automation Magazine, IEEE,* vol. 12, pp. 26-39, 2005.

[14] P. I. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB.* vol. 73. Berlin: Springer, 2011.

[15] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio, "Performance comparison of VxWorks, Linux, RTAI and Xenomai in a hard real-time application," in *Real-Time Conference, 2007 15th IEEE-NPSS*, 2007, pp. 1-5.

[16] John D'Enrico. PolyfitN function [Online]. Available: www.mathworks.com

[17] F. Bentouhami, B. Campagne, E. Cuevas, T. Drake, M. Dubois, T. Fraslin, P. Piñeiro, J. Serrano, and H. Voillaume, "LUCIE - A flexible and powerful Laser Ultrasonic system for inspection of large CFRP components.," presented at the 2nd International Symposium on Laser Ultrasonics, Talence (France), 2010.

[18] M. Burns, *Automated fabrication: improving productivity in manufacturing*: Prentice-Hall, Inc., 1993.

[19] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in *Robotics-DL tentative*, 1992, pp. 586-606.

[20] KUKA. *ARM TUTORIAL 2*. Available: http://www.kuka-robotics.com/en/products/software/ /educational_framework/arm_tutorials/PS_Content_Arm2.htm (Accessed: 20/08/2013)

[21] F. Chinello, S. Scheggi, F. Morbidi, and D. Prattichizzo, "Kuka control toolbox," *Robotics & Automation Magazine, IEEE,* vol. 18, pp. 69-79, 2011.

[22] Acuity. Available: http://www.acuitylaser.com (Accessed 08/04/13)

[23] S. H. Lim, "Video processing applications of high speed CMOS image sensors," Stanford University, 2003.

[24] B. Lebrun, Y. Jayet, and J.-C. Baboux, "Pulsed eddy current signal analysis: application to the experimental detection and characterization of deep flaws in highly conductive materials," *NDT & E International,* vol. 30, pp. 163-170, 1997.

[25] S. Giguere, B. Lepine, and J. Dubois, "Pulsed eddy current technology: Characterizing material loss with gap and lift-off variations," *Journal of Research in Nondestructive Evaluation,* vol. 13, pp. 119-129, 2001.

[26] G. Fernlund, "Spring-in of angled sandwich panels," *Composites science and technology,* vol. 65, pp. 317-323, 2005.

[27] W.-K. Jung, W.-S. Chu, S.-H. Ahn, and M.-S. Won, "Measurement and compensation of spring-back of a hybrid composite beam," *Journal of composite materials,* vol. 41, pp. 851-864, 2007.

[28] W. Weber, *Industrieroboter: Methoden der Steuerung und Regelung; mit... 33� bungsaufgaben sowie einer begleitenden Internetseite*: Hanser Verlag, 2009.

[29] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *The international journal of robotics research,* vol. 2, pp. 13-30, 1983.

5. A GRAPHICAL INTERFACE FOR PRACTICAL NDT IMPLEMENTATIONS

5.1. RoboNDT Software

This chapter describes the features of the Graphical User Interface (GUI) designed and developed for a path-planning software solution targeted to robotic NDT. The software application was named *RoboNDT* (short for Robotic NDT).

Chapter 4 explored fundamental considerations for robotic path-planning targeted to NDT inspections. The path-planning toolbox described in Section 4.5 was a valuable tool to test externally controlled robot tool-paths. However it does not fulfil all the motivations highlighted in Section 4.2 (Motivations for the investigation of new solutions).

The RoboNDT GUI was designed to provide a user-friendly platform to get the most from the functionalities investigated and implemented by the author. The resulting software overcomes the deficiencies of current commercial OLP. It is flexible enough to be used as a research tool, support future expansions of functionalities and approach commercial exploitations.

RoboNDT runs outside the MATLAB environment, through compiling it as a .EXE or a .DLL application. The .EXE format (standing for *executable* file) produces a stand-alone version of the software that can be used for independent path-planning. The .DLL format stands for *Dynamic Link Libraries*, which means that the application can be integrated to other software modules to ease the sharing of data and variables. This allows the achievement of the vision pursued by this work, which enables the possibility of establishing seamless robotic NDT software platforms. The conceptual representation is shown in Figure 5.1.

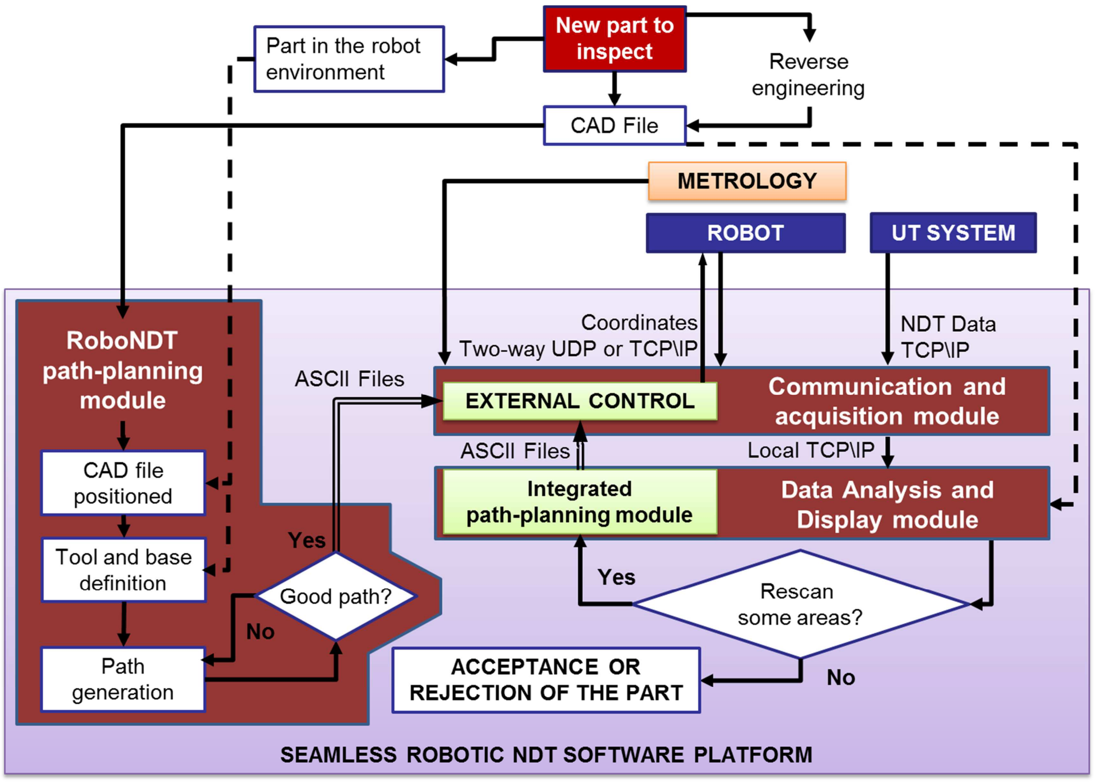


Figure 5.1 – Seamless robotic NDT software platform.

The development of RoboNDT covered a big part of the work described in this thesis and provided the ideal container to deliver the research of the industrial PhD program in the form of a tool, which remains available to the industry and to the research and scientific community.

5.1.1. Modular GUI architecture

The start interface of the GUI (Figure 5.2) is quite austere. It has a menu bar with four menus: File, Libraries, Options and Help. Below the menu bar there is a toolbar with four main tool buttons and five additional graphic tool buttons. The main buttons can be used to create a new project, load an existing project, save the current project and print the screen to memory. The start interface enables the user to create a new project or load an old one.

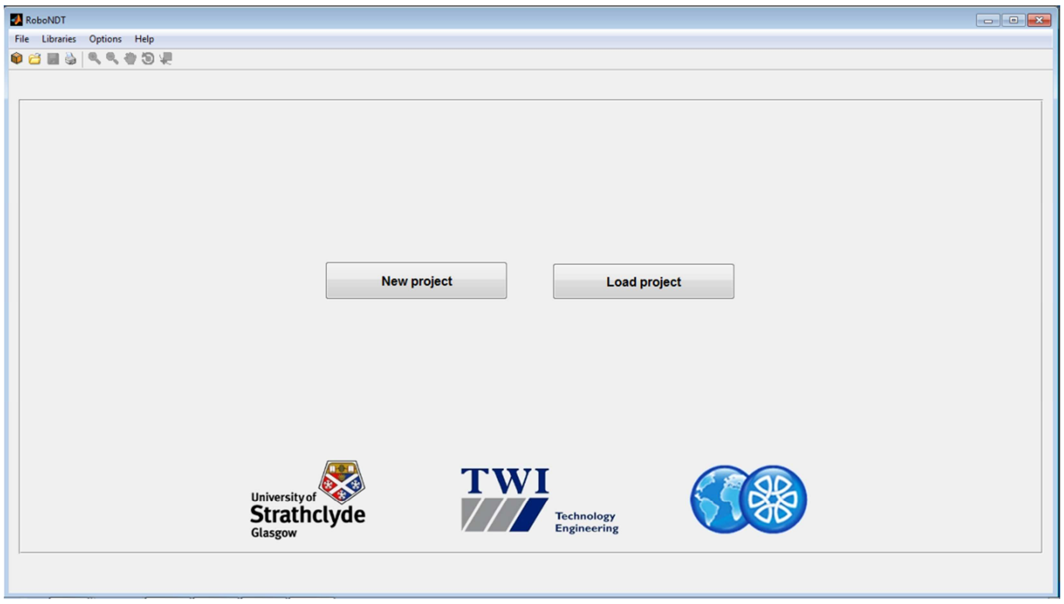

RoboNDT
File Libraries Options Help
New project
Load project
University of Strathclyde Glasgow
TWI Technology Engineering

Figure 5.2 – Start interface of RoboNDT.

If the user decides to create a new project, he is prompted to save the project file. The user can navigate to the preferred directory and the project file is stored with the desired name followed by the extension ".ndt". Once the file is saved the project is initialized and a navigation bar appears below the toolbar (see Figure 5.3).

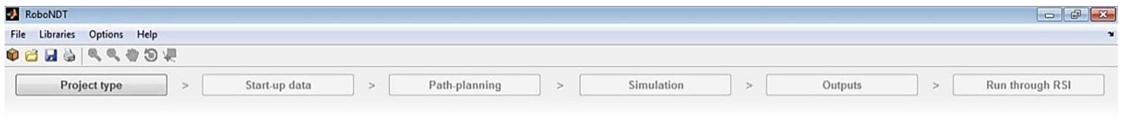
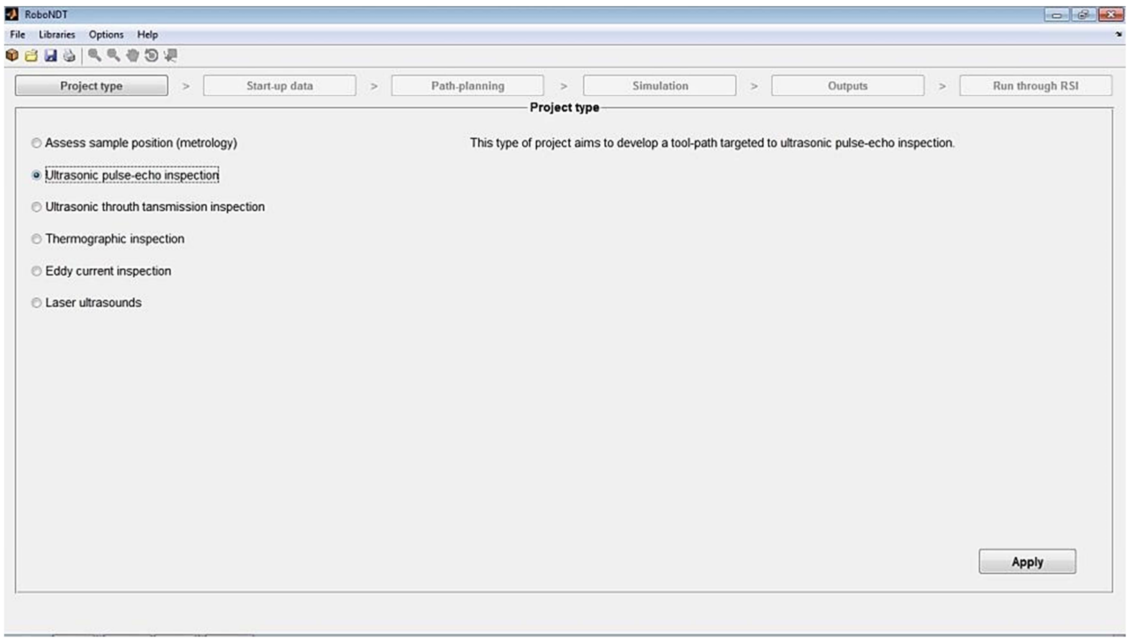


Figure 5.3 – Navigation bar populating the GUI when a project is open.

The navigation bar indicates the progress status during the development of the project and will permit the movement between the different project steps. The names of the steps are likely to change during the future further development of the GUI; the current names remind the modular structure of the MATLAB toolbox. They are:

1. Project type
2. Start-up data
3. Path-planning
4. Simulation
5. Outputs
6. Run through RSI

The first action following the creation of a new project consists in the choice of the Project Type. This phase is supported by a list of possible project types displayed as in Figure 5.4.


RoboNDT
File Libraries Options Help
Project type > Start-up data > Path-planning > Simulation > Outputs > Run through RSI
Apply

Project type

( ) Assess sample position (metrology)
(●) Ultrasonic pulse-echo inspection
( ) Ultrasonic throuth tansmission inspection
( ) Thermographic inspection
( ) Eddy current inspection
( ) Laser ultrasounds

This type of project aims to develop a tool-path targeted to ultrasonic pulse-echo inspection.

Figure 5.4 – List of possible project types displayed in the Project Type context.

The user needs to select one of the project types. A short description of the project is given on the right hand side of the GUI. The list of available project types can be customized in the future. The current types give an example of the possible items that are likely to be maintained for the future; other types can be added.

1. Assessment of sample position (metrology project)
2. Ultrasonic pulse-echo inspection
3. Ultrasonic through-transmission inspection
4. Thermography inspection
5. Eddy-current inspection
6. Laser ultrasounds

The first project type aims at providing a valid procedure to assess the position of the sample to be scanned. This project type is not enabled in the current version of RoboNDT, but the software is ready for its development. This project type could be set up to create a robot path-planning suitable to a get data from metrology equipment attached to the end-effector of the robot arm. The idea that could be pursued is to generate a coarse robot tool-path for checking some critical points of the sample and/or execute a laser scan of the sample surfaces. The collected metrology data can be used for the generation of accurate tool-paths for NDT inspections. This would potentially set up a procedure that can be used in manufacturing environments with cost-effective equipment.

The other project types (2 to 6) aim to generate specific tool-paths fulfilling the different needs of each NDT technique. The selection of a project type leads the user to the "Start-up data" context. This module of the software is designed to guide the user through the selection of the robot environment, the robot tool, the assessment of the sample position.

The project can be saved at any time using the save button in the toolbar or in the File menu of the menu bar (Figure 5.5).

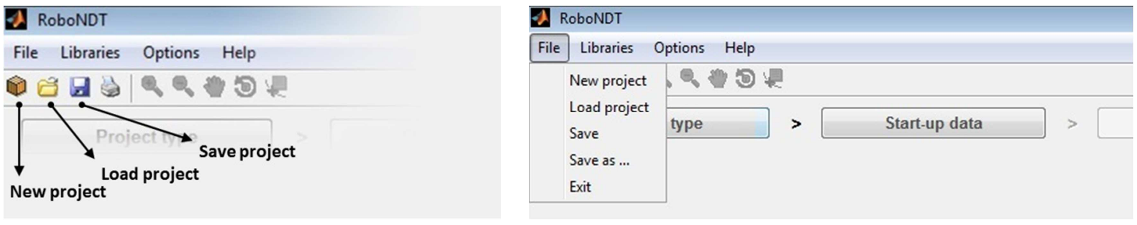
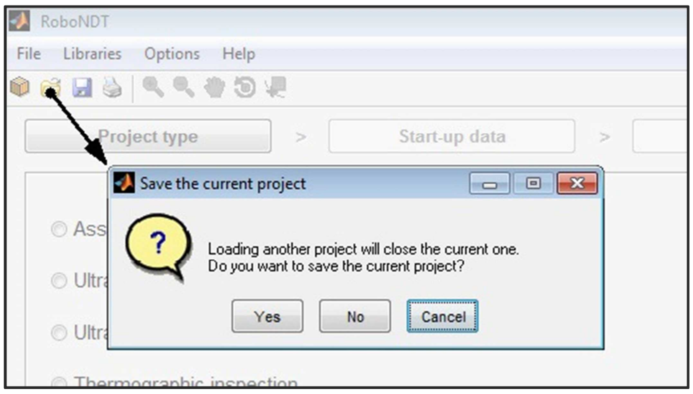


Figure 5.5 – Toolbar and File menu of the menu bar.

The GUI was developed paying particular attention to the data robustness and integrity. Messages like the one given in Figure 5.6 are programmed to be displayed every time a potential hazard for the data integrity is detected. The user is asked to take a conscious decision. This approach was maintained throughout the whole development of the GUI.



Loading another project will close the current one.
Do you want to save the current project?

Figure 5.6 – Example of message displayed for ensuring data integrity.

5.1.2. Libraries

Before presenting the features of the main body of the software, it is necessary to introduce the secondary functionalities of the GUI; despite being defined as "secondary", they constitute the foundations for the primary functionalities.

The *Libraries* menu in the menu bar gives access to 5 important modules of the GUI. They are the libraries of the robots, the tools (probes and sensors) and the environments and the appropriate contexts to manage the robot cells and the samples of interest.

It was deemed that most of the flexibility of the path-planning platform relies on the appropriate definition of the aforementioned elements. The different libraries and management contexts allow the user to easily reproduce the real working environment for the robotic NDT inspection and model the real equipment.

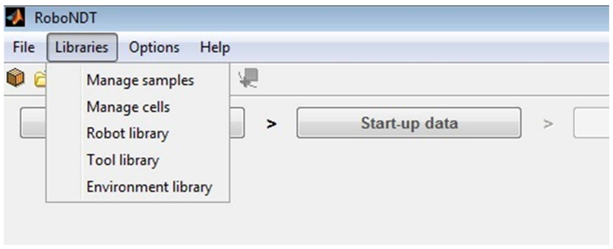


Figure 5.7 – Content of the Library menu in the main window.

These “secondary” elements are presented below.

5.1.2.1. Robot library

Selecting Robot library in the Libraries menu of the main window, the Robot Library window is displayed. It is given in Figure 5.8.

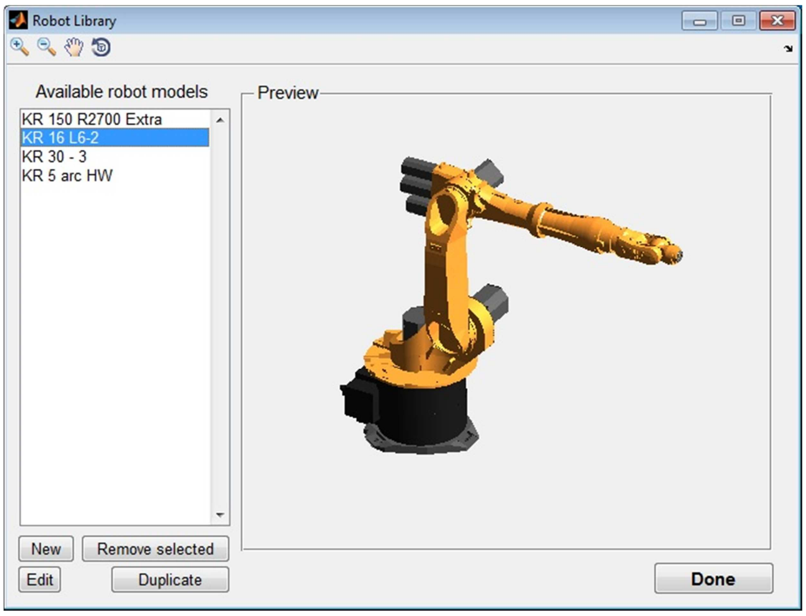
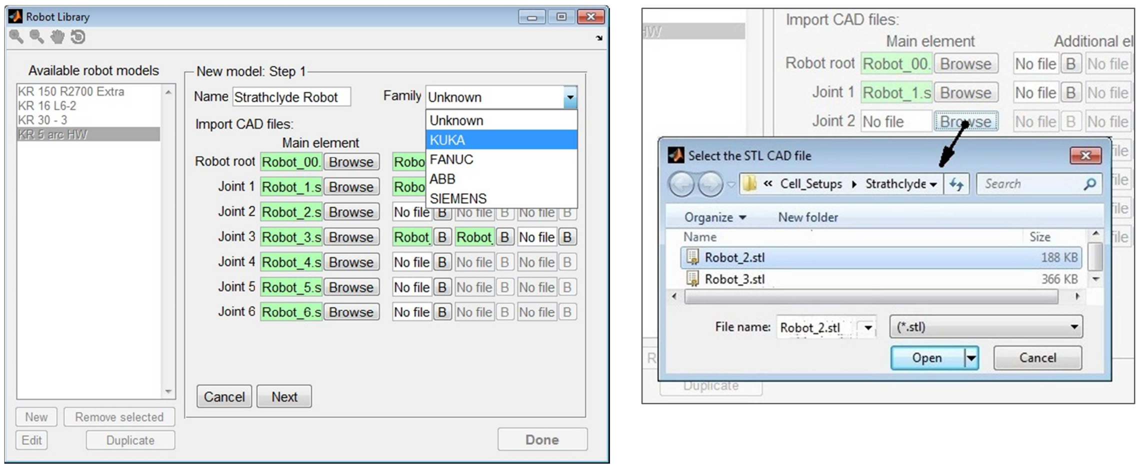


Figure 5.8 – Robot library window.

The list of available robot models is on the left hand side of the window. The user can select one robot to display its relative 3D model in the preview section. The graphic buttons on the top left corner can be used to zoon in, zoom out, pan and rotate the 3D model. The push buttons on the bottom left corner can be used to remove, edit or duplicate the selected robot model or to create a new robot model. Figure 5.8 shows the KUKA KR16 L6-2 robot used at TWI. The *Robot Library* is further explained below by going through the procedure to create a new robot model for the KUKA KR5arcHW in use at the University of Strathclyde.

Once the "*New button"* is pushed, all the previously described elements of the window are disabled and a new panel is shown to set up the new robot model. Firstly the robot name and the robot family have to be specified. In our case, "Strathclyde Robot" is specified as name and the KUKA family is selected (Figure 5.9). The user has to load the CAD files (in STL format) for all the 6 robot joints composing the robot, plus the robot root (the fixed element of the robot known as Base). They are necessary files; all of them have to be loaded to proceed. Additionally, up to three CAD elements can be loaded and associated to each robot joint. They can be ancillary elements (hoses, wires, actuators) that we might want to include to the virtual model to reproduce the real robot. The Browse button, next to each text box, opens the selection window to indicate the relative CAD file.


Robot Library

Available robot models:
- KR 150 R2700 Extra
- KR 16 L6-2
- KR 30 - 3
- KR 5 arc HW

New model: Step 1
Name: Strathclyde Robot
Family: Unknown (list: Unknown, KUKA, FANUC, ABB, SIEMENS)
Import CAD files:

	Main element		Additional elements		
Robot root	Robot_00.	Browse	Robo		
Joint 1	Robot_1.s	Browse	Robo		
Joint 2	Robot_2.s	Browse	No file	No file	No file
Joint 3	Robot_3.s	Browse	Robot	Robot	No file
Joint 4	Robot_4.s	Browse	No file	No file	No file
Joint 5	Robot_5.s	Browse	No file	No file	No file
Joint 6	Robot_6.s	Browse	No file	No file	No file

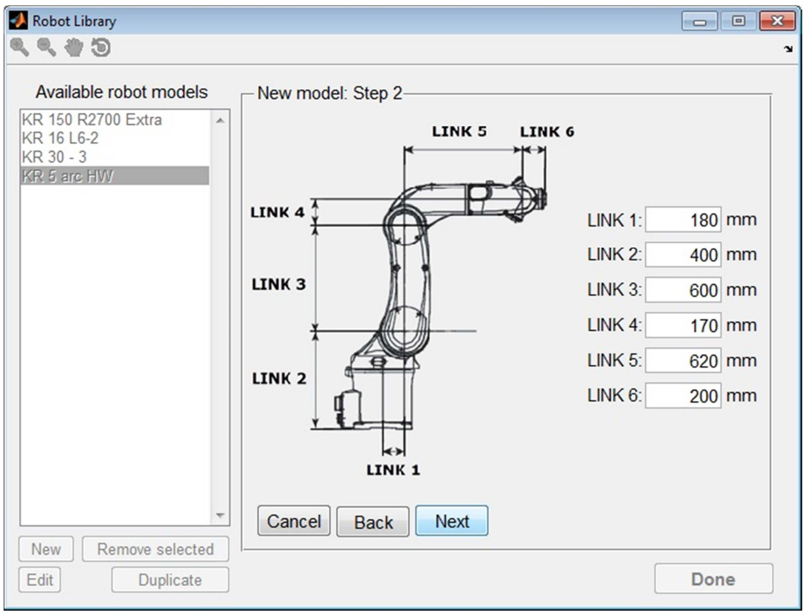
Select the STL CAD file
Cell_Setups ▸ Strathclyde

Name	Size
Robot_2.stl	188 KB
Robot_3.stl	366 KB

File name: Robot_2.stl
(*.stl)

Figure 5.9 – Creation of a new robot model – definition of name, family and CAD files selection.

The following panel (Figure 5.10) is used to provide the geometric details of the robot model. They are necessary to assembly the robot and simulate the robot movements. The required details are reported in the reference manual of each robot model, written by the robot manufacturer.

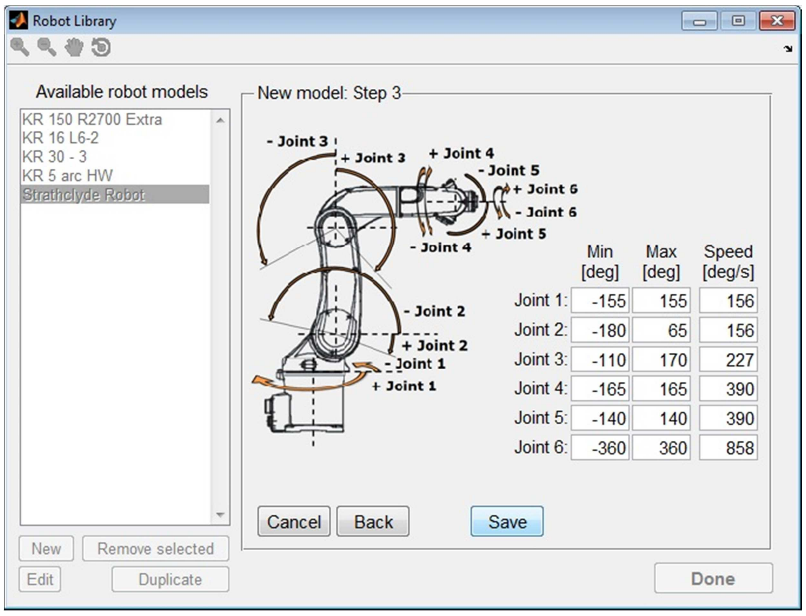


Available robot models:
KR 150 R2700 Extra
KR 16 L6-2
KR 30 - 3
KR 5 arc HW

LINK 1:	180	mm
LINK 2:	400	mm
LINK 3:	600	mm
LINK 4:	170	mm
LINK 5:	620	mm
LINK 6:	200	mm

Figure 5.10 – Geometric details.

The last panel of parameters concerns the kinematics features: the angular working interval of each joint and the relative maximum angular speed (Figure 5.11). These parameters are necessary for accurate simulation of robot tasks, where the consideration of the robot working envelope and the kinematics limits are important.

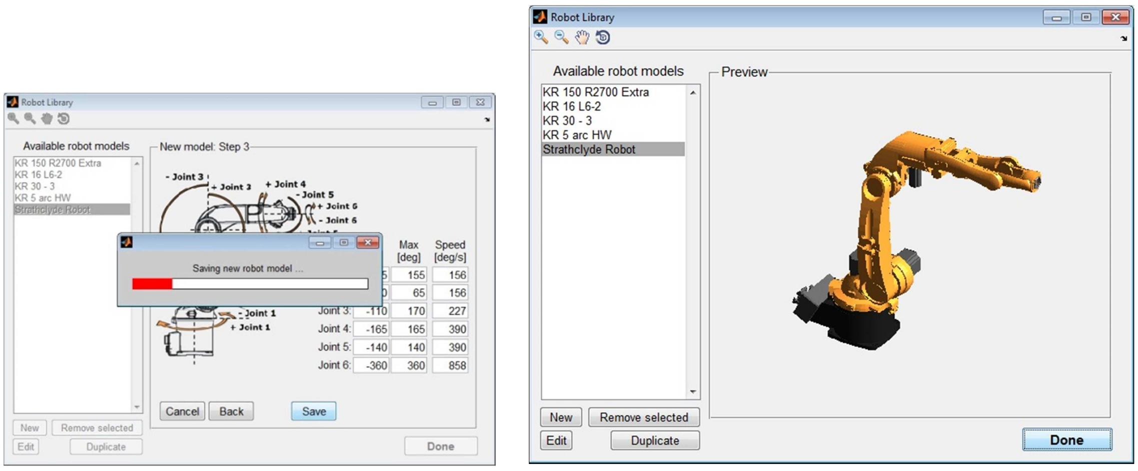


Available robot models:
KR 150 R2700 Extra
KR 16 L6-2
KR 30 - 3
KR 5 arc HW
Strathclyde Robot

	Min [deg]	Max [deg]	Speed [deg/s]
Joint 1:	-155	155	156
Joint 2:	-180	65	156
Joint 3:	-110	170	227
Joint 4:	-165	165	390
Joint 5:	-140	140	390
Joint 6:	-360	360	858

Figure 5.11 – Kinematics limits.

Finally, all data is saved and the new robot model is inserted into the list of available robot models. The Robot Library returns to the default view and the newly loaded model is displayed in the Preview section of the GUI (Figure 5.12).



		Max [deg]	Speed [deg/s]
		155	156
		65	156
Joint 3:	-110	170	227
Joint 4:	-165	165	390
Joint 5:	-140	140	390
Joint 6:	-360	360	858

Figure 5.12 – The new robot model is saved and inserted in the list of available models.

5.1.2.2. Environment library

Selecting *Environment library* in the *Libraries menu* of the main window (Figure 5.7), the Environment Library window is displayed. It is given in Figure 5.13.

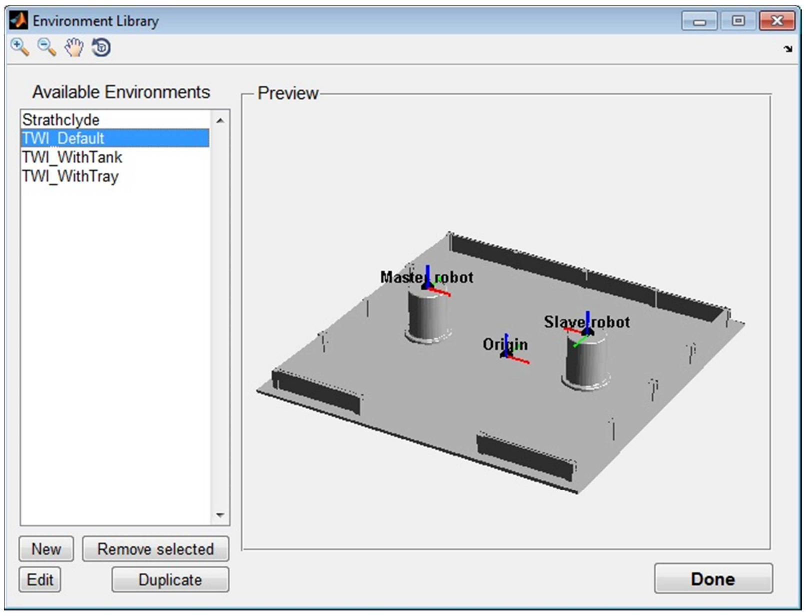
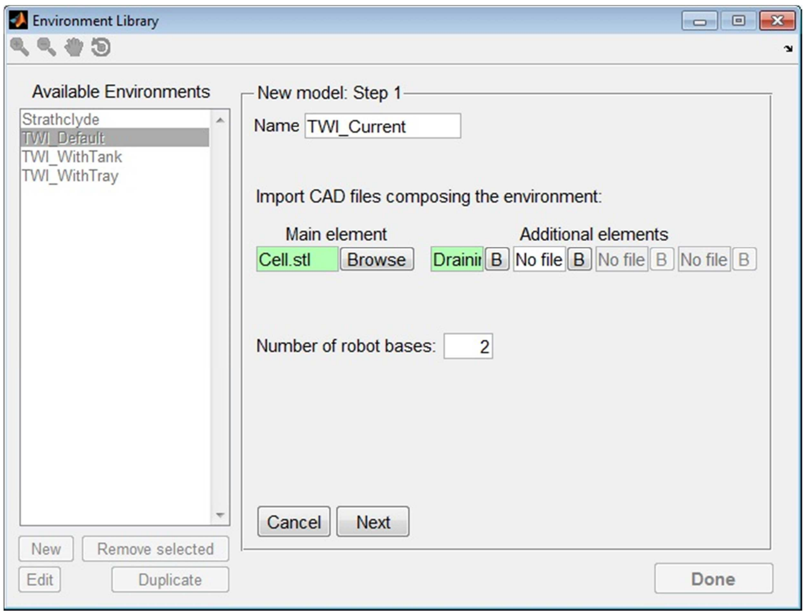


Figure 5.13 – Environment library window.

The window is similar to the *Robot Library* window. The list of available environments is on the left hand side of the window. The user can select one environment to display the relative 3D model in the preview section and the graphic buttons on the top left corner can be used to zoon in, zoom out, pan and rotate the 3D model. The push buttons on the bottom left corner can be used to remove, edit or duplicate the selected environment or to create a new one. Figure 5.13 shows the basic TWI environment constituted by a 5m x 5m enclosure and two 0.8m high plinths (pedestals for the robots). The Environment Library is further explained below, through showing the procedure to create a new model for the TWI robot environment with a water collection tray.

Once the *New button* is pushed, all the previously described elements of the window are disabled and a new panel is shown to set up the new environment model. Firstly, the environment name has to be specified. In our case, "TWI_Current" is specified as name (Figure 5.14). The user has to load the CAD file (in STL format) of the main element composing the environment. Additionally, up to four elements can be loaded to be considered within the environment. In this case, we load the CAD file of a draining tray. Next chapter will show how a robotic NDT prototype system, which uses water jet columns to couple ultrasonic probes, was developed at TWI.

The *Browse button*, next to each text box, opens the selection window to indicate the appropriate CAD file. Then, the number of robot bases in the environment is specified before proceeding.



Name TWI_Current

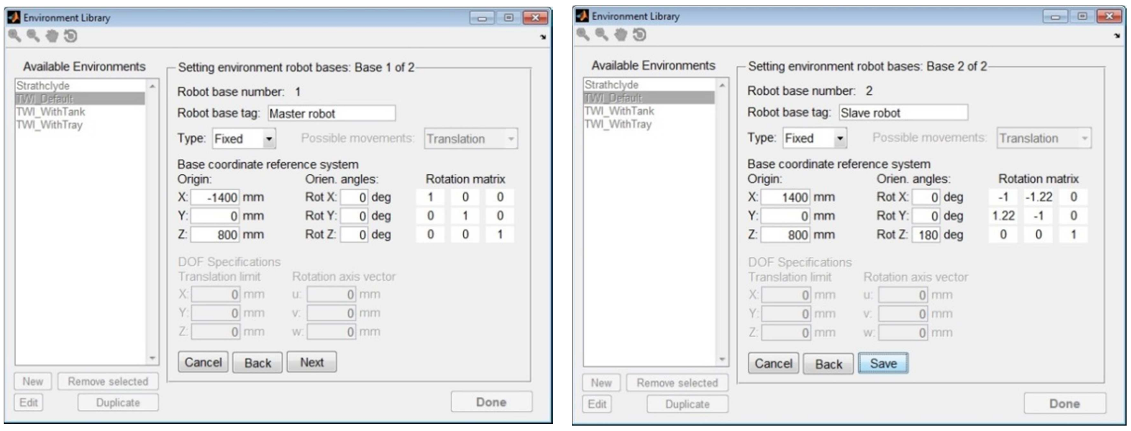
Main element: Cell.stl

Additional elements: Draini | No file | No file | No file

Number of robot bases: 2

Figure 5.14 – Creation of a new environment model – definition of name and CAD files.

The following panels are used to specify the robot type and set the coordinate reference system of each robot base. Fixed and mobile robot bases can be specified. Mobile bases can be translational, rotational or translational & rotational together. The number of panels that are displayed is equal to the number of specified bases (Figure 5.15), so each base has its own panel and the user can focus on a robot base at a time. The user gives a name to each base, its location and orientation relatively to the origin of the environment.



Setting environment robot bases: Base 1 of 2

Robot base number: 1

Robot base tag: Master robot

Type: Fixed   Possible movements: Translation

Base coordinate reference system

Origin:		Orien. angles:		Rotation matrix		
X:	-1400 mm	Rot X:	0 deg	1	0	0
Y:	0 mm	Rot Y:	0 deg	0	1	0
Z:	800 mm	Rot Z:	0 deg	0	0	1

DOF Specifications

Translation limit		Rotation axis vector	
X:	0 mm	u:	0 mm
Y:	0 mm	v:	0 mm
Z:	0 mm	w:	0 mm

Setting environment robot bases: Base 2 of 2

Robot base number: 2

Robot base tag: Slave robot

Type: Fixed   Possible movements: Translation

Base coordinate reference system

Origin:		Orien. angles:		Rotation matrix		
X:	1400 mm	Rot X:	0 deg	-1	-1.22	0
Y:	0 mm	Rot Y:	0 deg	1.22	-1	0
Z:	800 mm	Rot Z:	180 deg	0	0	1

DOF Specifications

Translation limit		Rotation axis vector	
X:	0 mm	u:	0 mm
Y:	0 mm	v:	0 mm
Z:	0 mm	w:	0 mm

(a) (b)

Figure 5.15 – Definition of the characteristics of a robot base.

As far as the TWI environment is regarded, the master robot base is located at X=-1.4m, Y=0m and Z=0.8m and the slave robot base is at X=1.4m, Y=0m and Z=0.8m. The master robot reference system is oriented like the reference system of the virtual environment model. Therefore it gives RotX=RotY=RotZ=0 degrees. The slave robot reference system is rotated by 180 degrees around the z-axis (RotX=RotY=0 and RotZ=180 degrees). The coordinates of the robot bases are necessary to translate any robot model to the specified base. The rotational matrix is automatically calculated according to the following:

$$R = R_z \cdot R_y \cdot R_x = \qquad \text{Eq. 5.1}$$

$$\begin{bmatrix} cos(RotZ) & -sin(RotZ) & 0 \\ sin(RotZ) & cos(RotZ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(RotY) & 0 & sin(RotY) \\ 0 & 1 & 0 \\ -sin(RotY) & 0 & cos(RotY) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(RotX) & -sin(RotX) \\ 0 & sin(RotX) & cos(RotX) \end{bmatrix}$$

It is equal to the diagonal identity matrix for RotX=RotY=RotZ=0.

$$R = R_z \cdot R_y \cdot R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad \text{Eq. 5.2}$$

Finally, the provided data are saved and the new environment model is added to the list of available environments. The Environment Library returns to the default view and the newly loaded environment is displayed in the Preview section of the GUI (Figure 5.16).

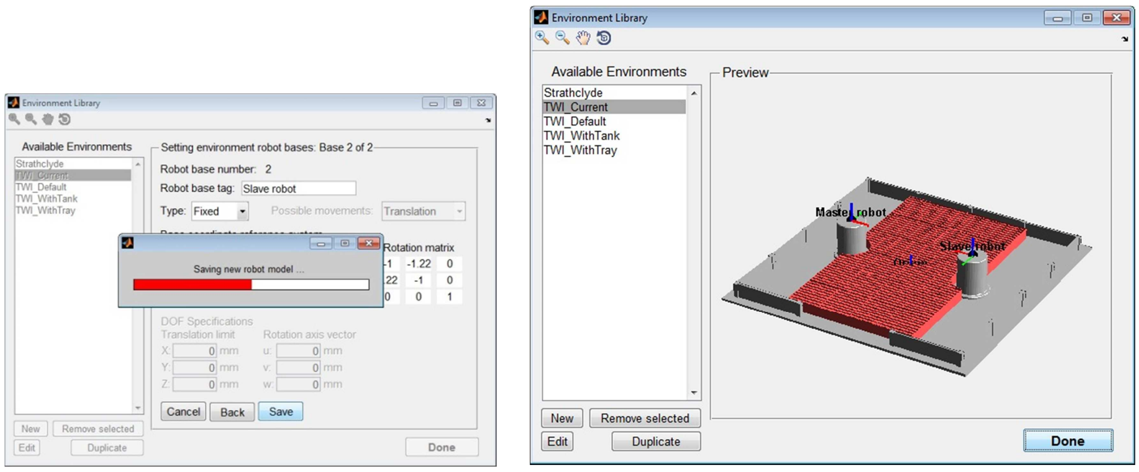


Figure 5.16 – The new environment is saved and added to the list of available environments.

5.1.2.3. Management of robotic cells

The *cells manager* was developed to allow populating a given environment with the desired robot models, in order to create complete robot cells to be used as platforms for the tool-path generation. The window for the management of the robot cells is displayed through selecting *Libraries menu > Manage cells* from the main RoboNDT window. It is given in Figure 5.17.

The layout of the window is again similar to the one used for the other libraries. The list of available robotic cells is on the left hand side of the window. The user can select one cell to display the relative 3D model in the preview section. Figure 5.17 shows the KUKA KR5arcHW robot placed in a very simple environment. The cells manager is further explained below, showing the procedure to populate the "*TWI_Current"* environment with two KUKA KR16 L6-2 robots.

Firstly, starting the procedure to create a new cell, the cell name has to be specified. In our case, "*TWI_Current*" is specified as name (Figure 5.18). The environment to be used has to be selected from the pop-up menu. Herein all the environments present in the environment library are given as possible choices. If a new environment is needed, the option "Go to the library…" can redirect to the *Environment Library* to let the user create a new one.

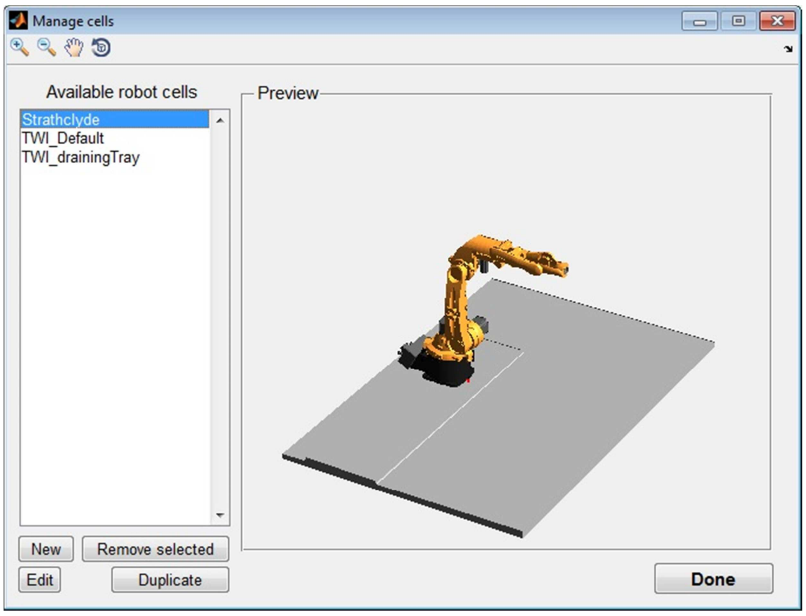


Figure 5.17 – Cells management window.

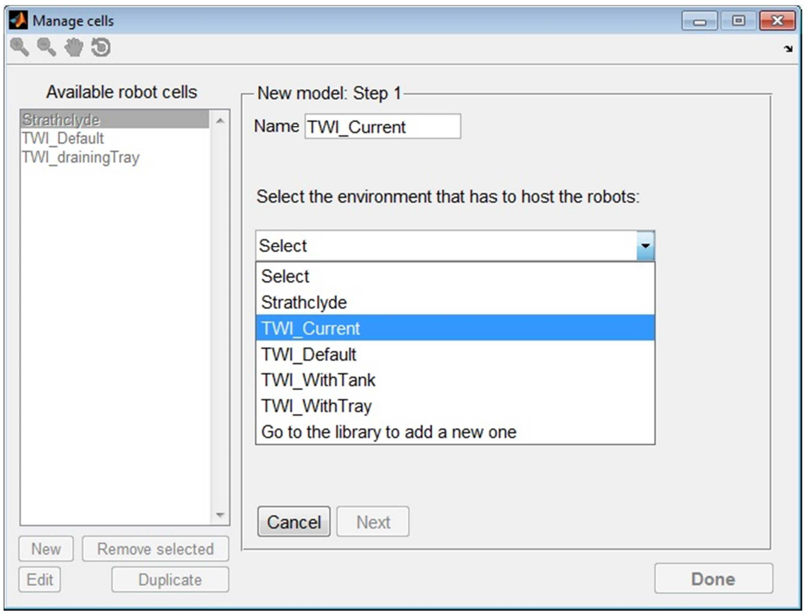
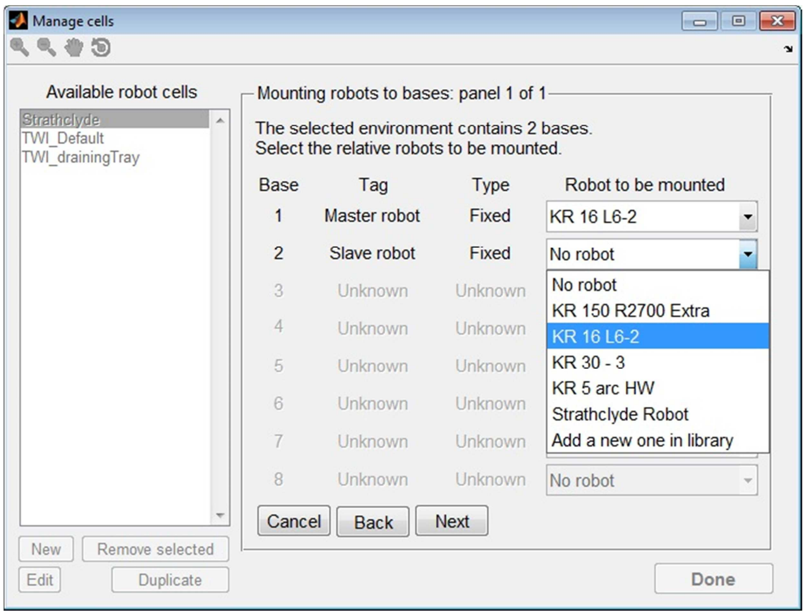


Figure 5.18 – Definition of the cell name and selection of the environment to use.

The following panel (Figure 5.19) is used to specify the robot model that has to be mounted on each base present in the environment. The user can select the desired robot model from the pop-up menu relative to every robot base. All the robots present in the robot library are given as possible choices. If a different robot is needed, the option “Go to the library…” redirects to *the Robot Library* to let the user insert a new robot model. It is also possible to not associate any robot to some bases. Of course, at least one robot has to be included to create a valid robot cell.


Manage cells

Available robot cells
- Strathclyde
- TWI_Default
- TWI_drainingTray

Mounting robots to bases: panel 1 of 1

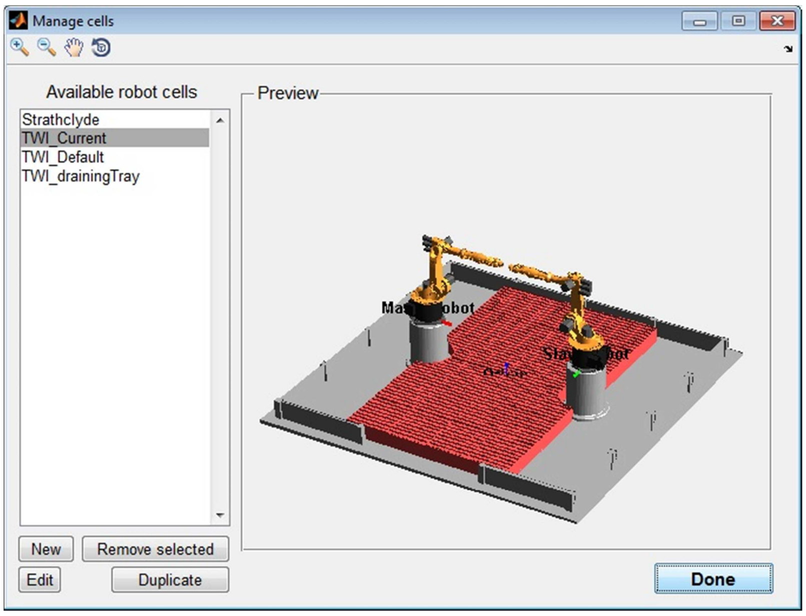
The selected environment contains 2 bases.
Select the relative robots to be mounted.

Base	Tag	Type	Robot to be mounted
1	Master robot	Fixed	KR 16 L6-2
2	Slave robot	Fixed	No robot
3	Unknown	Unknown	
4	Unknown	Unknown	
5	Unknown	Unknown	
6	Unknown	Unknown	
7	Unknown	Unknown	
8	Unknown	Unknown	No robot

Figure 5.19 – Matching the environment bases with the respective robot models.

The panel can be used to match up to 8 bases with the respective robots. If the environment contains more than 8 bases, additional panels are displayed to match all base-robot pairs. Therefore the number of bases in the environments is not restricted to a maximum number.

Finally, the cell is saved and its name is added to the list of available robotic cells. The cell management context returns to the default view and the newly loaded cell is displayed in the preview section of the GUI (Figure 5.20).


Manage cells

Available robot cells
- Strathclyde
- TWI_Current
- TWI_Default
- TWI_drainingTray

Figure 5.20 – TWI current robotic cell shown by the cell management window.

5.1.2.4. Tool library

Every robot arm needs to be equipped with a suitable tool to execute a specific task. Therefore the *Tool Library* is another important library of RoboNDT. It can be accessed through *Tool Library > Libraries menu*, in the main programme window. It allows the user to create virtual models of robot end-effectors (tools, probes or sensors) to be mounted on the robots. When the Tool Library is displayed, it looks like the window given in Figure 5.21.

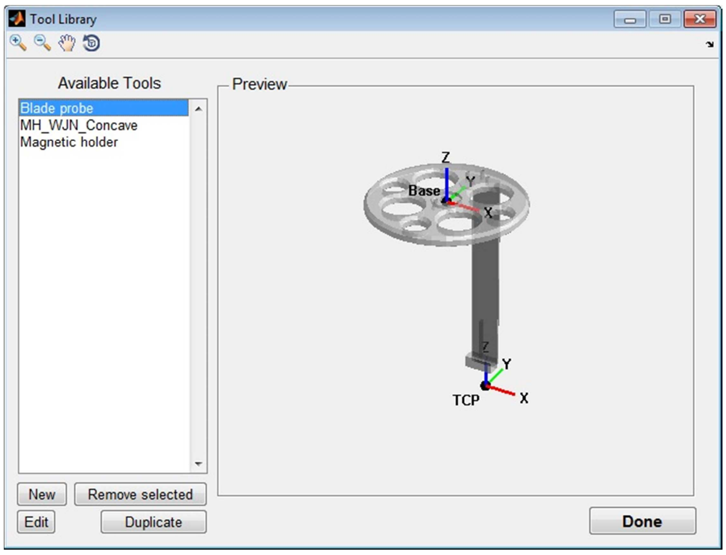
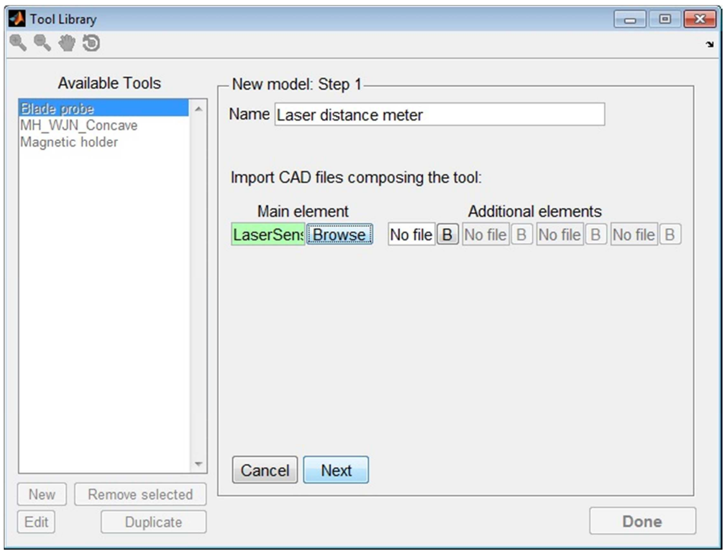


Figure 5.21 – Tool library window.

Figure 5.21 shows a robot tool consisting of a blade shaped support holding an ultrasonic phased array probe. The Tool Library is further explained below through showing the procedure to create a new tool model for a laser distance meter sensor (the one used for the metrology experiments described in Section 4.7).


Tool Library

Available Tools
Blade probe
MH_WJN_Concave
Magnetic holder

New model: Step 1
Name: Laser distance meter

Import CAD files composing the tool:

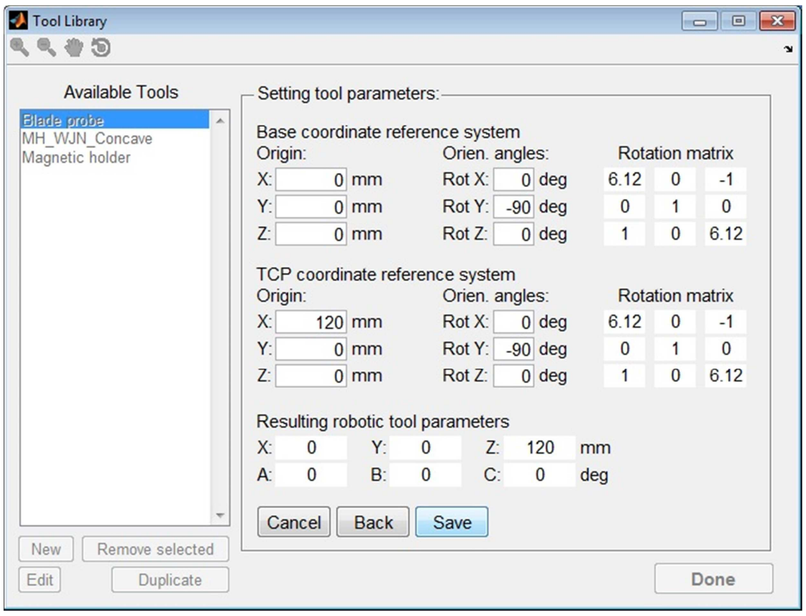
Main element: LaserSen: Browse
Additional elements: No file B | No file B | No file B | No file B

Cancel | Next

New | Remove selected | Edit | Duplicate | Done

Figure 5.22 – Creation of a new tool.

Firstly, the name of the new tool is specified. Together with the necessary main CAD file of the tool model, up to four additional elements can be loaded to be part of the resultant tool. The following panel (Figure 5.23) is used to specify the reference system of the *tool base* and of the *tool central point* (TCP). The correct specification of the tool base relatively to the coordinate reference system of the CAD is essential for the exact assembly of the tool model with the robots. The TCP reference system is instead important for the correct orientation of the tool during the execution of any tool-path. The correct specification of the TCP is crucial to obtain the desired tool orientation and standoff between the robot end-effector and the part to inspect.



Available Tools: Blade probe, MH_WJN_Concave, Magnetic holder

Setting tool parameters:

Base coordinate reference system

Origin:		Orien. angles:		Rotation matrix		
X:	0 mm	Rot X:	0 deg	6.12	0	-1
Y:	0 mm	Rot Y:	-90 deg	0	1	0
Z:	0 mm	Rot Z:	0 deg	1	0	6.12

TCP coordinate reference system

Origin:		Orien. angles:		Rotation matrix		
X:	120 mm	Rot X:	0 deg	6.12	0	-1
Y:	0 mm	Rot Y:	-90 deg	0	1	0
Z:	0 mm	Rot Z:	0 deg	1	0	6.12

Resulting robotic tool parameters

X:	0	Y:	0	Z:	120	mm
A:	0	B:	0	C:	0	deg

Figure 5.23 – Inserting the parameters for the tool base and TCP reference systems.

Using the data provided by the user, the code behind the GUI automatically updates the Base rotational matrix (R_B) and the TCP rotational matrix (R_T) according to the following:

$$\mathrm{R_B} = \mathrm{R_{Bz}} \cdot \mathrm{R_{By}} \cdot \mathrm{R_{Bx}} = \qquad \text{Eq. 5.3}$$

$$= \begin{bmatrix} cos(Rot_{Bz}) & -sin(Rot_{Bz}) & 0 \\ sin(Rot_{Bz}) & cos(Rot_{Bz}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(Rot_{By}) & 0 & sin(Rot_{By}) \\ 0 & 1 & 0 \\ -sin(Rot_{By}) & 0 & cos(Rot_{By}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(Rot_{Bx}) & -sin(Rot_{Bx}) \\ 0 & sin(Rot_{Bx}) & cos(Rot_{Bx}) \end{bmatrix}$$

$$\mathrm{R}_T = \mathrm{R}_{Tz} \cdot \mathrm{R}_{Ty} \cdot \mathrm{R}_{Tx} = \qquad \text{Eq. 5.4}$$

$$= \begin{bmatrix} cos(Rot_{Tz}) & -sin(Rot_{Tz}) & 0 \\ sin(Rot_{Tz}) & cos(Rot_{Tz}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(Rot_{Ty}) & 0 & sin(Rot_{Ty}) \\ 0 & 1 & 0 \\ -sin(Rot_{Ty}) & 0 & cos(Rot_{Ty}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(Rot_{Tx}) & -sin(\mathrm{Rot}_{Tx}) \\ 0 & sin(Rot_{Tx}) & cos(Rot_{Tx}) \end{bmatrix}$$

Most importantly, the tool parameters ([X, Y, Z, A, B, C]) suitable to be copied into the robot controller are also calculated. The angles (A, B, C) are given by the following:

$$\mathrm{R} = \mathrm{R_T}/\mathrm{R_B} \qquad \text{Eq. 5.5}$$

$$B = atan2\left(\frac{-R_{31}}{\sqrt{R_{11}^2+R_{21}^2}}\right) \qquad \text{Eq. 5.6}$$

$$\mathrm{A} = \begin{cases} 0, & \text{for } |B| = \pi/2 \\ atan2\left(\frac{R_{21}}{R_{11}}\right), & \text{for } |B| \neq \pi/2 \end{cases} \qquad \text{Eq. 5.7}$$

$$\mathrm{C} = \begin{cases} C = \frac{\mathrm{B}}{\mathrm{abs(B)}} \cdot atan2(R_{12}), & \text{for } |B| = \pi/2 \\ C = atan2\left(\frac{R_{32}}{R_{33}}\right), & \text{for } |B| \neq \pi/2 \end{cases} \qquad \text{Eq. 5.8}$$

And the tool Cartesian parameters (X, Y, Z), given the coordinates of the tool Base (X_B, Y_B, Z_B) and of the tool TCP (X_T, Y_T, Z_T), are equal to:

$$O_B = [X_B \quad Y_B \quad Z_B] * R_B \qquad \text{Eq. 5.9}$$

$$O_T = [X_T \quad Y_T \quad Z_T] * R_T \qquad \text{Eq. 5.10}$$

$$\begin{cases} X = O_{Bx} - O_{Tx} \\ Y = O_{By} - O_{Ty}. \\ Z = O_{Bz} - O_{Tz} \end{cases} \qquad \text{Eq. 5.11}$$

Finally, the tool is saved and its name is added to the list of available tools. The Tool Library window returns to the default view and the newly loaded tool is displayed in the Preview section of the GUI (Figure 5.24).

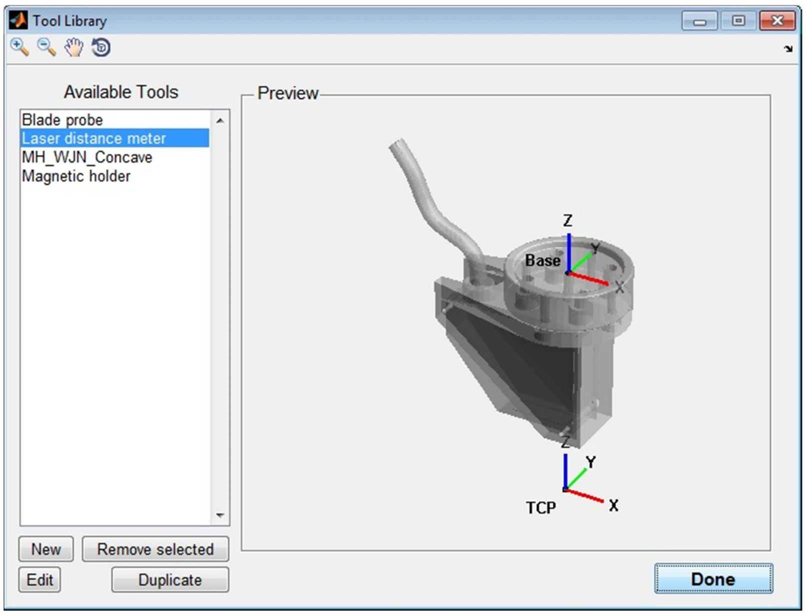
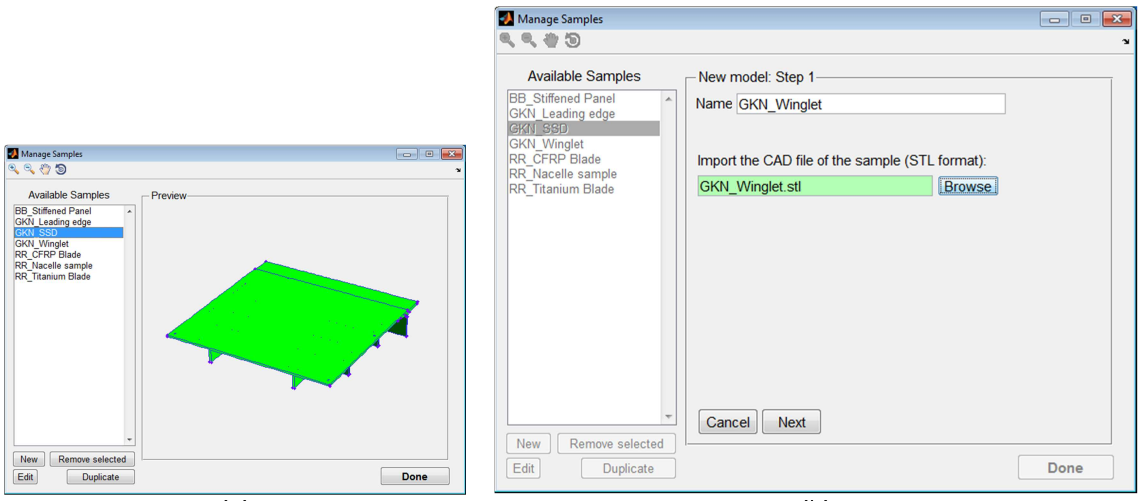


Figure 5.24 – Laser distance meter loaded in the tool library.

5.1.2.5. Management of the samples

Figure 5.25a shows the layout of the samples management window. The list of available samples is displayed on the left hand side and the user can select one sample to show the relative 3D model in the preview section. The graphic buttons on the top left corner can be used to zoon in, zoom out, pan and rotate the 3D model. The push buttons on the bottom left corner can be used to remove, edit or duplicate the selected sample or to add a new one. When the button *New* is pushed, all the previously described elements of the window are disabled and a new panel is shown to set up the new sample model. Figure 5.25b shows the specification of the sample's *Name* and *File Directory*, required to add the new sample model.



(a) (b)

Figure 5.25 – Samples management window (a) and creation of new sample model (b).

During the importation of the CAD file, the software is able to recognise the main geometrical entities. The path-planning MATLAB toolbox presented in Chapter 4 was already able to identify the surfaces of a STL file, but it was capable of recognising the boundaries and the corners for the sample model. The recognition of all the geometrical entities is necessary to facilitate both the operations of calibration of the sample position and the path-planning tasks.

An efficient and correct identification of all geometric entities of the sample facilitates the selection of good reference points for the preliminary assessment of the position of the specimen within the robot working envelope. It can also considerably help the user in the definition of the areas of interest in the path-planning operations.

Figure 5.26 shows the newly imported model of the aerospace winglet sample, where the boundary of each surface is highlighted by blue lines and the corners by red points.

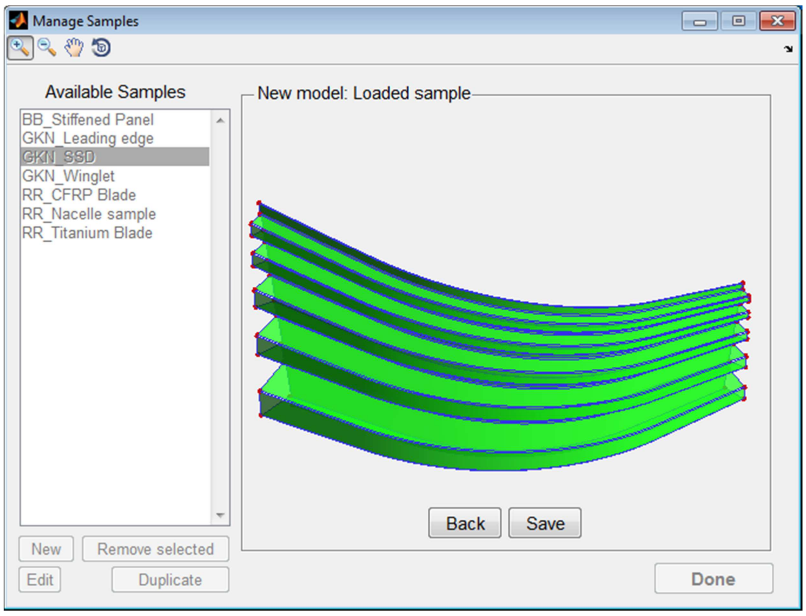


Figure 5.26 – Preview of the loaded sample model.

5.1.3. Calibration of the sample position

The triangular mesh of the sample, imported from the STL file, needs to be positioned correctly within the virtual robotic cell. The mathematical formulation for the calibration of the sample position, implemented in the MATLAB toolbox, was based on a positioning algorithm originally proposed for 3D point cloud data registration [1]. It was described in Section 4.5.1.

The algorithm calculates the optimum position translation and rotation to apply to the tessellated sample model to minimize the square errors of the distances between the points selected in the real sample and the relative points selected from the model. A minimum number of three points is required.

Differently from the original toolbox, devoid of graphical interface, the Start-Up module integrated into RoboNDT take advantage of the many robot cells and samples that the user can define in the relative libraries of the software. Figure 5.27 shows the specification of the robot cell.

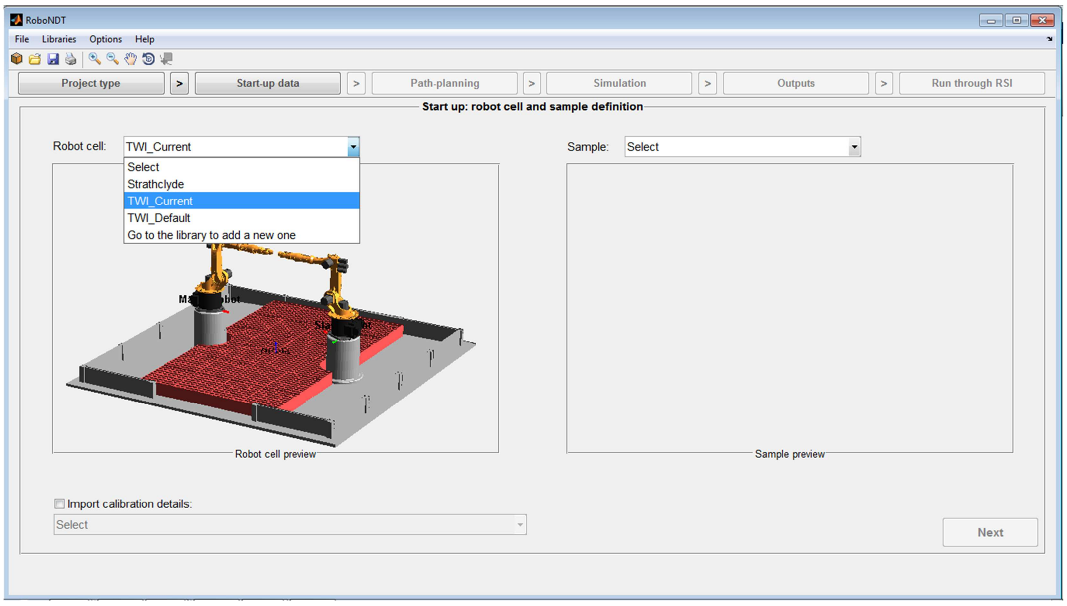


Figure 5.27 – Robot cell specification in the Start-Up module of RoboNDT software.

Figure 5.28 shows the specification of the sample of interest. Both the robot cell and the sample are selected from pop-up menus that give the items predefined in the relative libraries.

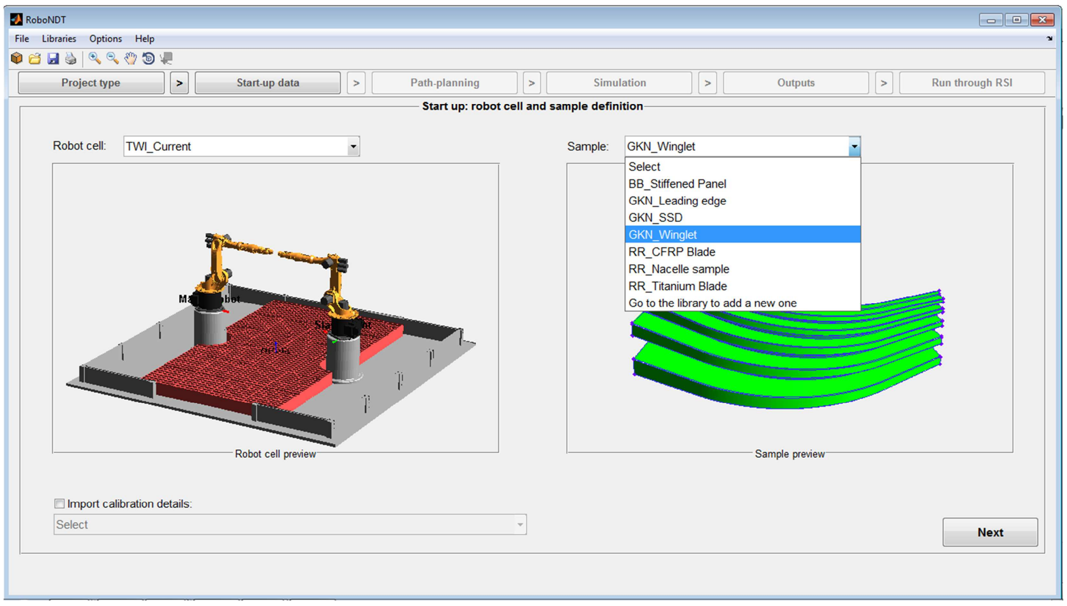
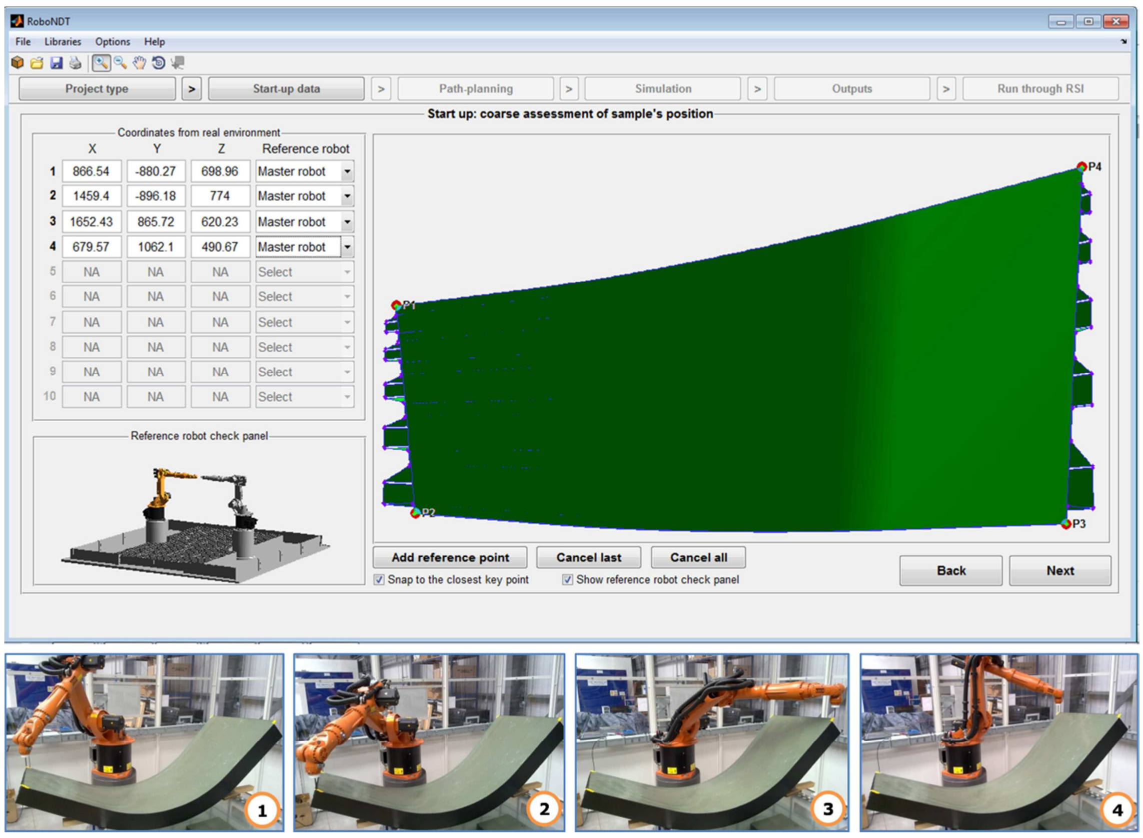


Figure 5.28 – Selection of the sample of interest in the Start-Up module of RoboNDT software.

Figure 5.29 shows the insertion of the points' Cartesian coordinates and the photos of the robot arm jogged to the points. The original interactive procedure implemented into the MATLAB toolbox was based in the definition of four calibration points; the number of points was fixed. However the accuracy of the sample calibration increases when more points are selected.



	X	Y	Z	Reference robot
1	866.54	-880.27	698.96	Master robot
2	1459.4	-896.18	774	Master robot
3	1652.43	865.72	620.23	Master robot
4	679.57	1062.1	490.67	Master robot
5	NA	NA	NA	Select
6	NA	NA	NA	Select
7	NA	NA	NA	Select
8	NA	NA	NA	Select
9	NA	NA	NA	Select
10	NA	NA	NA	Select

[✓] Snap to the closest key point
[✓] Show reference robot check panel

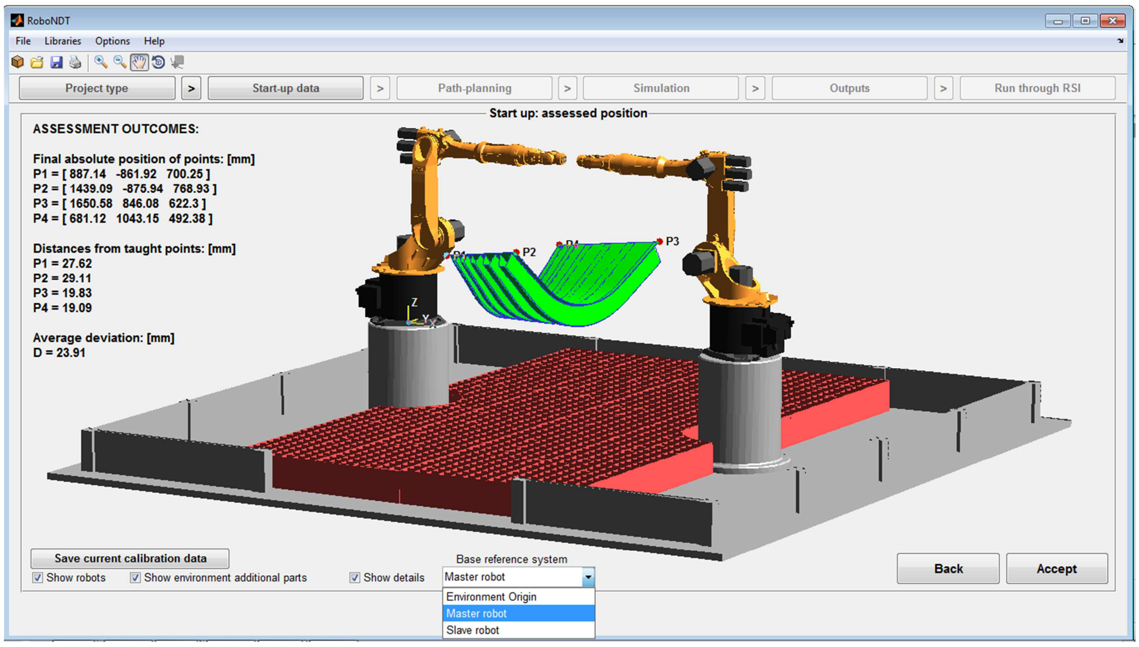
Figure 5.29 – Insertion of points' coordinates. The photos at the bottom show the robot jogged to the points.

The Start-Up module of RoboNDT allows selecting up to 10 points from the virtual model of the sample. Each selected points enables one row in the list of points on the left hand side of the window. Since the key points of the sample are recognised during the loading of the CAD model, operated by the samples management window, the key points can be used to facilitate the selections. If a wrong point is selected it can immediately be removed using the button "*Cancel last*". If all points need to be reselected the button "*Cancel all*" can be used to clean the list of selected points.

The check "*Snap to the closest key points*" enables the software to replace the rough position interactively selected by the user with the nearest key point. The user can decide to use or not to use this option to ease the work.

Each row on the numbered list of points on the left hand side of the window has a pop-up menu to specify the reference robot, used to get the Cartesian coordinates of the relative point. The Cartesian coordinates given by each robot arm, when it is jogged to a specific position, are referenced to its global reference system (the root reference system). Therefore, when more than one robot is available in the selected robot environment, it is fundamentally important to specify the reference robot used to retrieve the coordinates of each point. Moreover, this enables the technicians to use different robots to reach all the selected points. This can be very useful when some of the points cannot be reached by only one robot or when the sample is not entirely contained within the working envelope of none of the available robots. The robot names shown by the pop-up menus are the names specified in the definition (operated through the environments library) of the robot environment in use. The virtual model of the robot cell is also given at the bottom left corner of the window as an additional aid for the user to identify the reference robot and avoid wrong selections.

Figure 5.30 shows the virtual model of the sample finally placed within the virtual model of the robotic cell.



```
ASSESSMENT OUTCOMES:

Final absolute position of points: [mm]
P1 = [ 887.14   -861.92   700.25 ]
P2 = [ 1439.09   -875.94   768.93 ]
P3 = [ 1650.58   846.08   622.3 ]
P4 = [ 681.12   1043.15   492.38 ]

Distances from taught points: [mm]
P1 = 27.62
P2 = 29.11
P3 = 19.83
P4 = 19.09

Average deviation: [mm]
D = 23.91
```

Figure 5.30 – Assessment of the sample position.

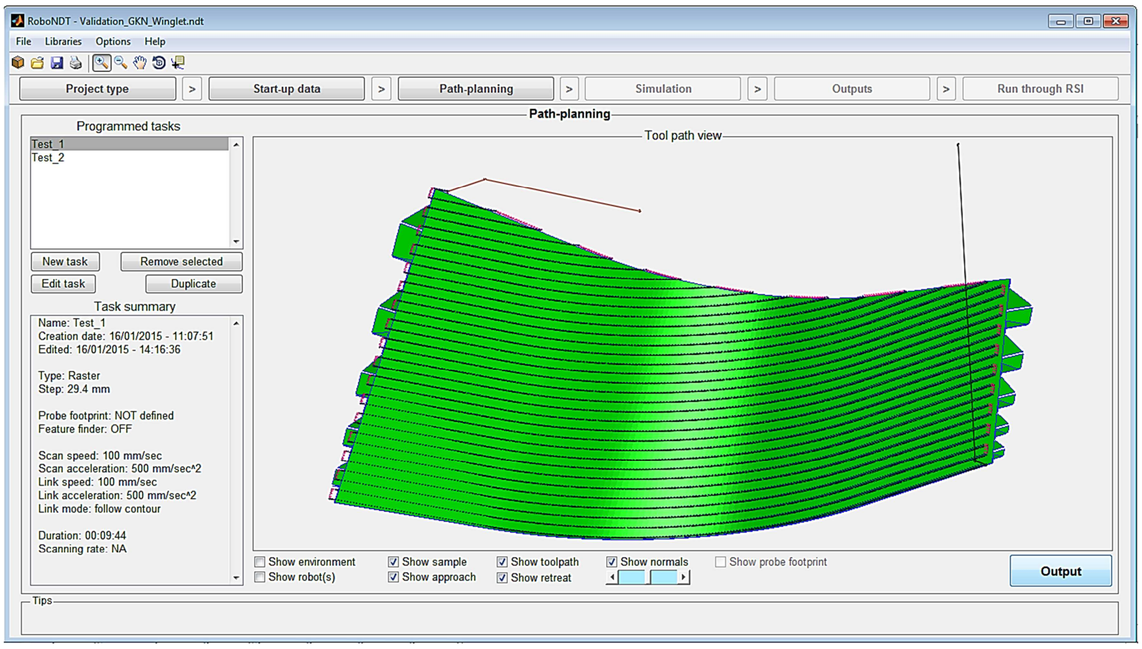
The virtual representation given in Figure 5.30 is used to check and accept the result of the sample's position. The user can choose to enable or disable the visualization of the robots and the additional parts of the environment (e.g. the water collection trays in the TWI environment). Moreover, the calibration details can be displayed on the left hand side of the window. They report the final coordinates of the selected points and the distances from the original points. The average deviation is also given. The user has the possibility to display the Cartesian coordinates relatively to the environment reference system or to the robot root systems. The selection of the preferred reference system can be done through the pop-up menu at the bottom.

The calibration of a sample is an activity that can take several minutes. The final result can be saved. Therefore, this allows loading of saved calibration data, in order to entirely skip the calibration procedure. The option to load saved calibrations is given at the bottom of the window shown in Figure 5.28.

5.1.4. Path-planning GUI module

The enhanced stability and flexibility of the new path-planning algorithms (described in Section 4.7), together with its new features, inspired the layout of the path-planning module of the software GUI. RoboNDT enables a range of different types of tool-paths, going far beyond the old MATLAB toolbox that was just able to generate raster type tool-paths.

The acceptance of the result of the sample calibration (Figure 5.30) enables the path-planning module of the software. The layout is given in Figure 5.31.



Name: Test_1
Creation date: 16/01/2015 - 11:07:51
Edited: 16/01/2015 - 14:16:36

Type: Raster
Step: 29.4 mm

Probe footprint: NOT defined
Feature finder: OFF

Scan speed: 100 mm/sec
Scan acceleration: 500 mm/sec^2
Link speed: 100 mm/sec
Link acceleration: 500 mm/sec^2
Link mode: follow contour

Duration: 00:09:44
Scanning rate: NA

[ ] Show environment [✓ Show sample] [✓ Show toolpath] [✓ Show normals] [ ] Show probe footprint
[ ] Show robot(s) [✓ Show approach] [✓ Show retreat]

Figure 5.31 – Layout of the path-planning module.

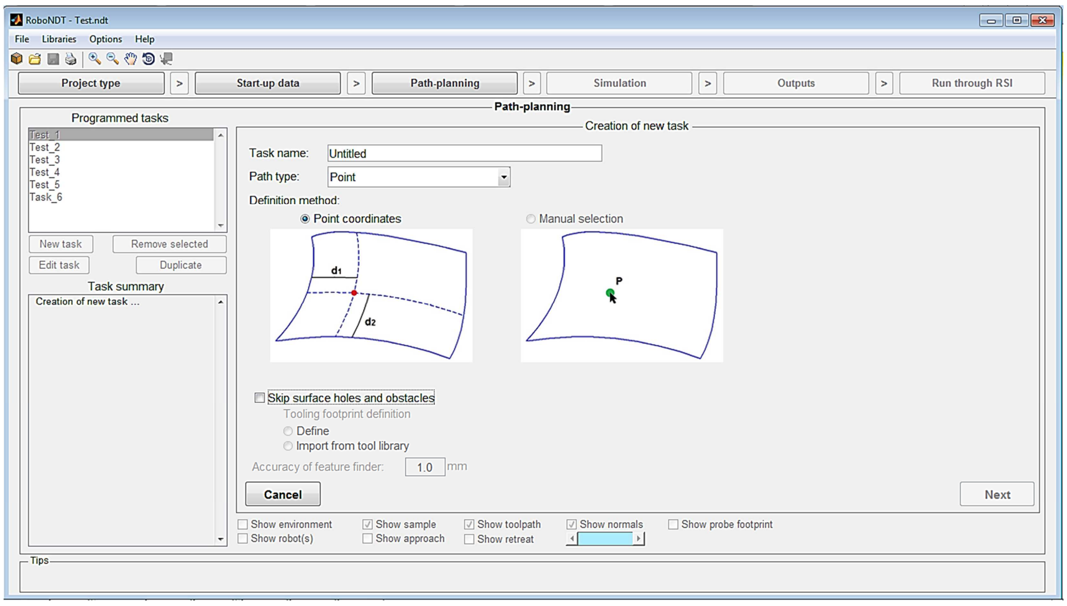
The fundamental idea behind the path-planning module is that the users might want to define many inspection tool-paths for each sample. The full-definition of a tool-path creates a *task*. The list box, on the left hand side, is to contain all the programmed *tasks*. The main panel is dedicated to the 3D visualization of the tool-path contained into the selected task. A text box (*Task summary*), below the list box, shows the details of the task.

The push buttons placed in between the list of *Programmed Tasks* and the *Task Summary* enable removing, editing or duplicating the selected task or the creation of a new one.

When the button *New* is pushed, a new panel replaces the visualization panel to initialize the creation of a new task. Therefore, the user defines the *task name* and the *path type*. Three categories of tool-path types have been foreseen for the full-developed version of RoboNDT: *Point*, *Segment* and *Raster*.

5.1.4.1. Point path

The *Point* path type is to jog the robot manipulator to a single position. This can be obtained in two different ways: by giving the distance of the target point from two sides of the surface of interest or through an interactive mode, selecting the point directly through the mouse (Figure 5.32).



Programmed tasks: Test_1, Test_2, Test_3, Test_4, Test_5, Task_6

Task summary: Creation of new task ...

Task name: Untitled
Path type: Point
Definition method: [✓ Point coordinates] Manual selection

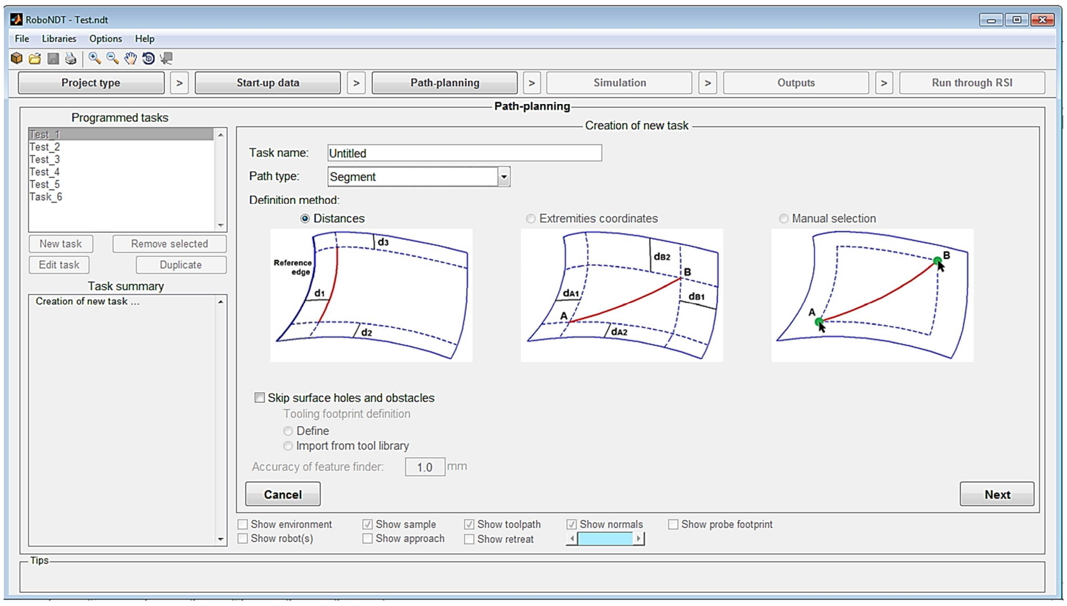
☐ Skip surface holes and obstacles
Tooling footprint definition: Define / Import from tool library
Accuracy of feature finder: 1.0 mm

☐ Show environment ☐ Show robot(s) ☑ Show sample ☐ Show approach ☑ Show toolpath ☐ Show retreat ☑ Show normals ☐ Show probe footprint

Figure 5.32 – Possible definition methods for the Point tool-path type.

5.1.4.2. Segment path

The Segment tool-path type is to move the robot end effector along a single curve segment. This tool-path type can be defined in three different ways (Figure 5.33). The first definition method consists in giving the distance of the curve from the reference edge and the distances of the curve extremities from the adjacent surface boundaries. The second way consists in providing the coordinates of the starting and end point of the curve, as distances from the surface boundary. The last method is based on the interactive selection of the curve extremities.



Test_1
Test_2
Test_3
Test_4
Test_5
Task_6

Creation of new task ...

Path-planning

Creation of new task

Task name: Untitled
Path type: Segment
Definition method:
(●) Distances ( ) Extremities coordinates ( ) Manual selection

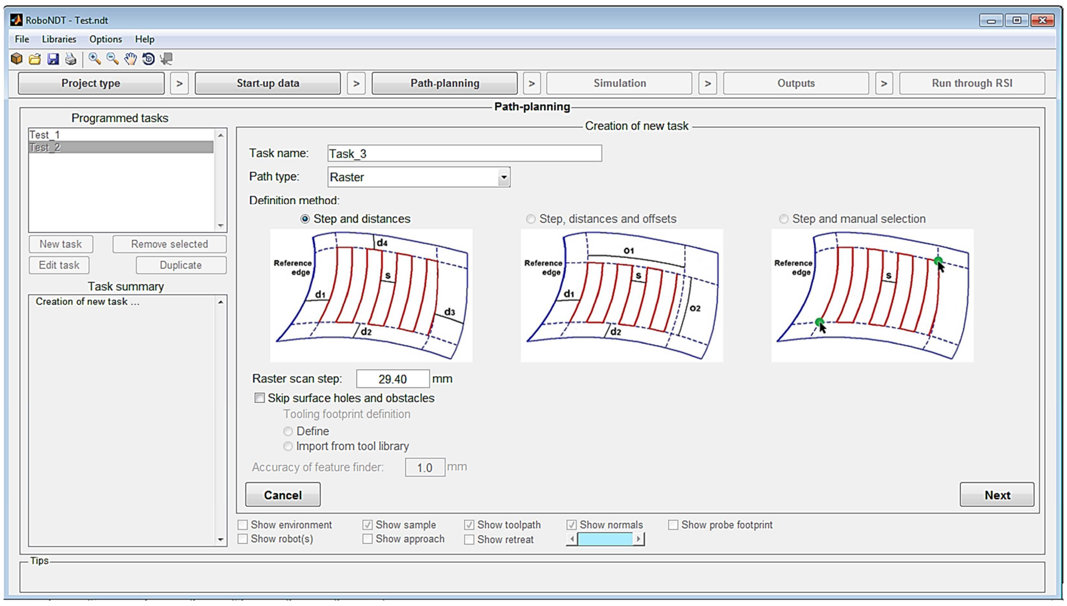
☐ Skip surface holes and obstacles
Tooling footprint definition
( ) Define
( ) Import from tool library
Accuracy of feature finder: 1.0 mm

☐ Show environment ☑ Show sample ☑ Show toolpath ☑ Show normals ☐ Show probe footprint
☐ Show robot(s) ☐ Show approach ☐ Show retreat

Figure 5.33 – Possible definition methods for the Segment tool-path type.

5.1.4.3. Raster path

The Raster tool-path type is to move the robot end effector according to a raster fashion. This tool-path type can be defined in three different ways (Figure 5.34). The first definition method consists in giving the step and the distances of the raster tool-path from the sides of the surface boundary. The second way consists in providing the step and the coordinates of one corner of the area to scan and two relative extensions (offsets), to define a rectangular area of interest. The last method is based on the interactive selection of two opposite corners of the rectangular area of interest and the definition of the step.



Test_1
Test_2

Creation of new task ...

Path-planning

Creation of new task

Task name: Task_3
Path type: Raster
Definition method:
(●) Step and distances ( ) Step, distances and offsets ( ) Step and manual selection

Raster scan step: 29.40 mm
☐ Skip surface holes and obstacles
Tooling footprint definition
( ) Define
( ) Import from tool library
Accuracy of feature finder: 1.0 mm

☐ Show environment ☑ Show sample ☑ Show toolpath ☑ Show normals ☐ Show probe footprint
☐ Show robot(s) ☐ Show approach ☐ Show retreat

Figure 5.34 – Possible definition methods for the Raster tool-path type.

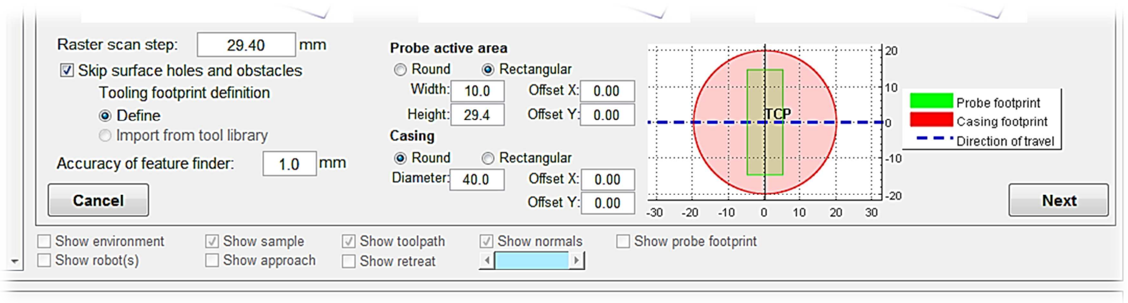
5.1.5. Avoidance of surface obstacles and voids

Collision avoidance has traditionally been the most important problem of off-line and on-line path-planning for fixed and mobile robots [2, 3]. However the traditional problem found in the literature and solved by many authors concerns the computation of the optimal trajectory of a robotic manipulator, when the manipulator must move from an initial position to a given final state without colliding with surrounding obstacles [4]. This problem is different from the problem we face in robotic NDT inspections.

A robot arm manipulating a NDT probe is not only required to move the probe from an initial position to a given final state; in doing so, the manipulator has to follow the contour of the part of interest, whilst maintaining constant standoff and orientation between the part surface and the probe. Therefore the robot manipulator has a limited freedom of changing the trajectory that links the initial position to the final state.

A collision avoidance algorithm has been developed and implemented in RoboNDT. The idea behind the algorithm developed in this work is that obstacles must be *skipped* when they are encountered, rather than being *avoided*. The NDT inspection tool-paths cannot be developed as a function of the obstacles to be avoided; they are more appropriately developed according to the surfaces of interest and the paths are locally modified when they intersect an obstacle.

This kind of obstacle avoidance requires the knowledge of the size of the casing of the NDT probes to avoid the collision between the surface protrusions and the robot end-effector. The interface to support the insertion of the parameters for the obstacle avoidance algorithm can be enabled during the creation of a new task. After selecting the tool-path type the check box "*Skip surface holes and obstacles*" can be ticked. This enables the definition of the NDT probe active area and of the casing footprint (Figure 5.35).



Raster scan step: 29.40 mm
[✓] Skip surface holes and obstacles
Tooling footprint definition
[✓ Define]
Import from tool library
Accuracy of feature finder: 1.0 mm

Probe active area
Round [✓ Rectangular]
Width: 10.0 Offset X: 0.00
Height: 29.4 Offset Y: 0.00

Casing
[✓ Round] Rectangular
Diameter: 40.0 Offset X: 0.00
Offset Y: 0.00

Show environment
[✓] Show sample
[✓] Show toolpath
[✓] Show normals
Show probe footprint
Show robot(s)
Show approach
Show retreat

Figure 5.35 – Definition of the parameters for the obstacles and holes avoidance algorithm

The collision avoidance algorithm considers the profile of the casing footprint at each computed point of the tool-path and looks for intersections between the footprint boundary and the internal boundaries of the surface of interest. If intersections are found for a given point, the point is appended to the array of positions that must be skipped.

The capability of the implemented algorithm, to check for intersections between profiles centred on each point of the tool-path and the internal boundaries of the surface of interest, is also deployed to optimize the tool-path around surface voids. This is the reason for specifying the footprint of the NDT probe active area. This can be particularly useful when the surface has large holes; in this situation it is desirable to stop the acquisition of NDT data meanwhile the robot end-effector travels across voids to not store useless data into memory. Firstly the tool-path points that are inner to the internal surface features are found. Among these points, the points that produce no intersection between the probe profile and the internal boundaries are appended to the array of positions for which the data collection can be disabled.

Figure 5.35 shows that the collision avoidance algorithm requires the specification of a parameter called “Accuracy of feature finder”. This is the accuracy with which the user wants to avoid the collision points. The tool-path is interpolated to set the distance between the points equal to the specified accuracy, before the points are analysed to avoid collisions. The obstacle avoidance algorithm is always conservative, meaning that the collisions are always avoided and the portions of surface left uninspected along the scanning trajectories have a maximum width equal to the specified accuracy.

5.1.6. Interactive definition of tool-path features

Given a sample, the selection of the surface of interest, the reference edge, the definition of the offsets and all other entities is carried out in an interactive way.

The definition of a raster tool-path is described herein. It was implemented as an interactive procedure based on 4 steps. The first step is the selection of the tool-path type at the beginning of the creation of the new task. Figure 5.34 shows the selection of the raster tool-path type and of the definition mode based on “Step and distances”. The *task name* is specified in the text box at the top of the screen and the *raster step* is given in millimetres.

The second step consists in selecting the surface of interest. The selected surface turns from green to purple. If the wrong surface is selected, the user can reset the selection and select the correct surface (Figure 5.36).

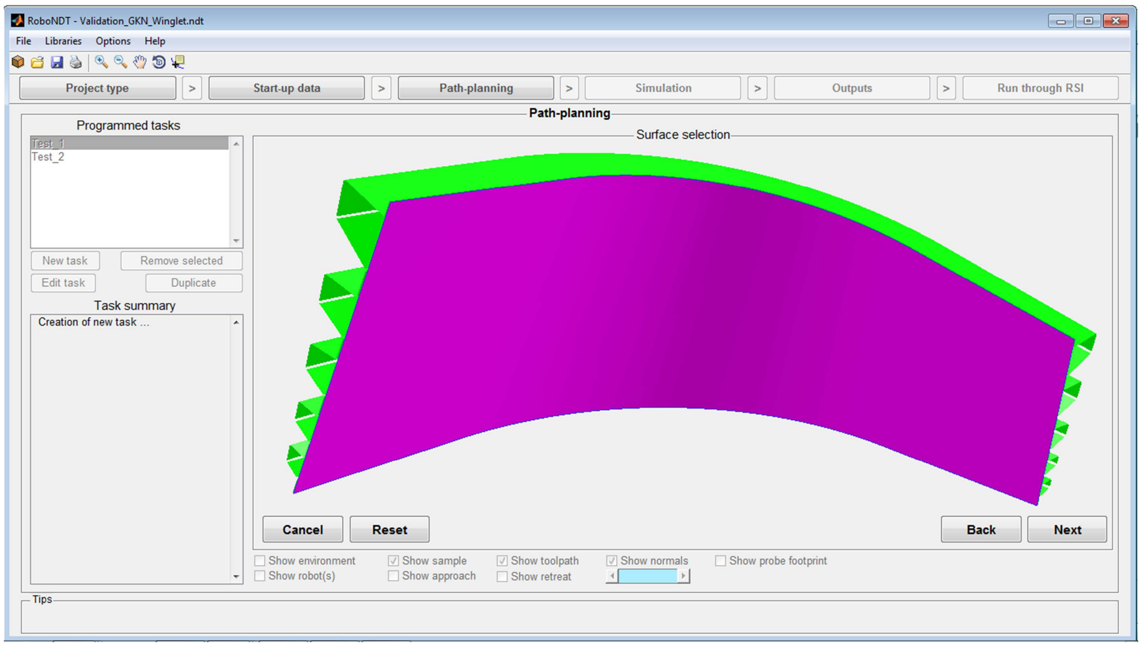


Figure 5.36 – Selection of the surface of interest.

The third step regards the selection of the reference edge. The user can drive the cursor nearby the edge of interest and click the left hand side button of the computer mouse. The edge is highlighted with a thick blue line and the extremities are indicated by the descriptions: "*Start point*" and "*End point*". Since the reference edge is used for the generation of the first pass of the raster scan, the user might want to change the verse of the edge. This can be done by ticking the check box "*Flip edge*" at the bottom of the panel. If the wrong reference edge is selected, the user can reset the selection and select another edge (Figure 5.37).

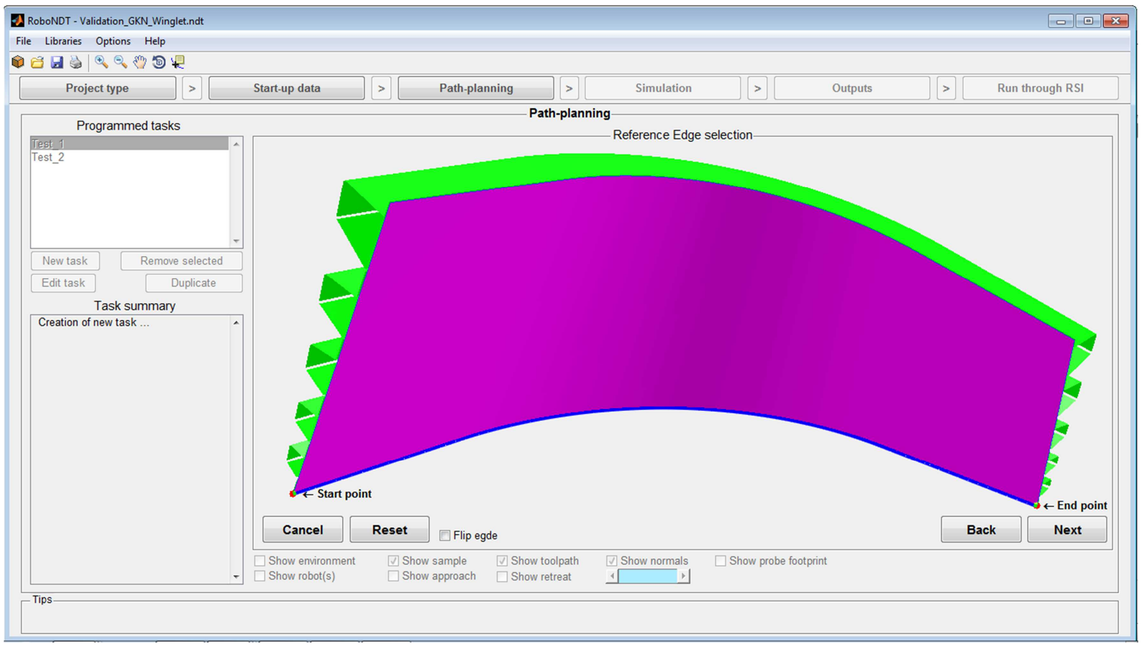


Figure 5.37 – Selection of the reference edge.

The last step concerns the specifications of the offsets. The user can drive the cursor nearby any of the surface edges. A left hand side mouse click triggers a pop-up window, where the offset relative to the closest edge can be specified (in millimetres). A positive offset is intended to point towards the surface, a negative offset is pointing outside the surface. Any value other than "0.00" is transferred to a reminder text box, displayed on the middle point of the relative edge. Each offset can be modified at any time. Figure 5.38 shows a screenshot during the application of a negative offset (-30mm) to the right hand side edge of the surface.

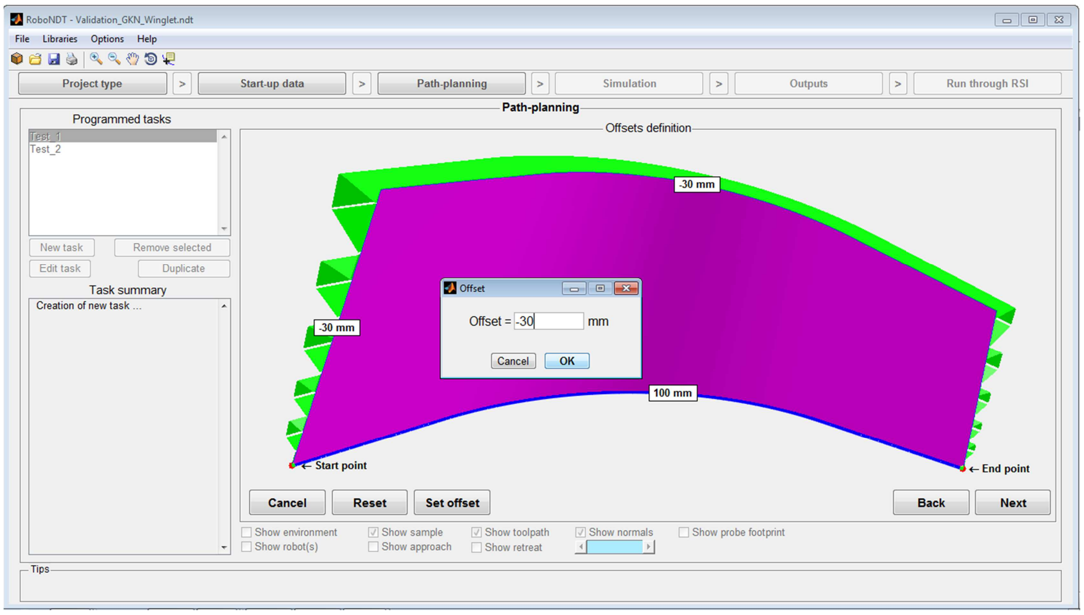


Figure 5.38 – Specification of the offsets.

The fundamentals of the path-planning algorithm were described in Section 4.7.2, for the generation of parallel lines that follow the curved mesh of the sample model without the need for approximating polynomial surfaces. The final code for the generation of the tool-paths was developed through extensive programming work to support the level of flexibility shown by the GUI. The user-friendly GUI helps the user to transfer his requirements to the software, but the effectiveness of the path-planning algorithms plays a crucial role in guaranteeing a high degree of robustness and usefulness. Figure 5.39 shows the raw tool-path generated according to the parameters specified in Figure 5.35-47.

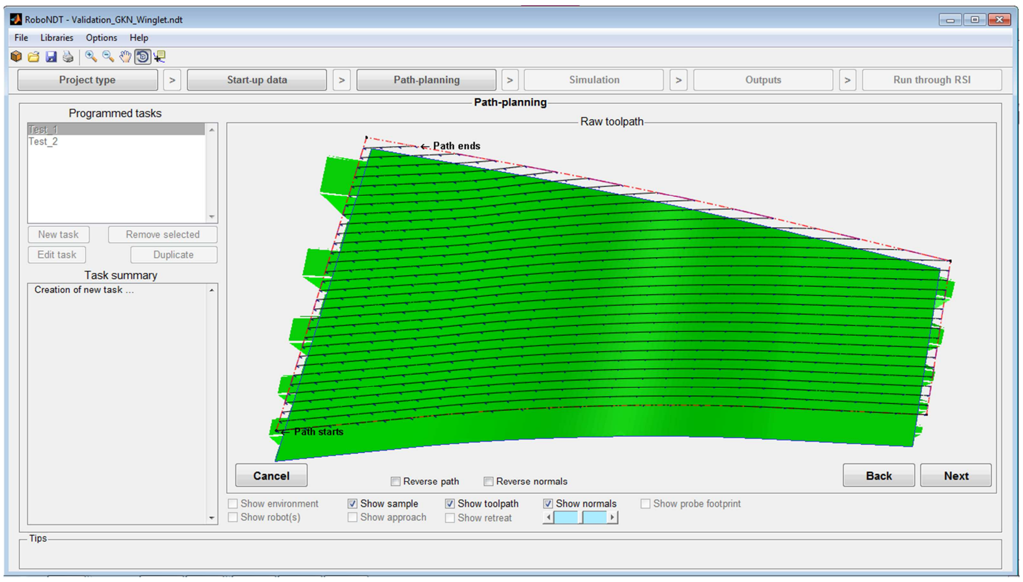


Figure 5.39 – Generated raw too-path.

When the check box to avoid obstacles and holes is selected and the surface of interest contains some internal feature an additional step is required, before obtaining the raw tool-path. It concerns the specification of the features types. The internal boundaries are automatically found by the software, but the user is asked to specify if these internal features are relative to obstacles or voids. This is better clarified by Figure 5.40, showing a surface of interest with 9 internal boundaries (features). Feature 3, 4 and 5 are small holes; the other features are 80mm tall obstacles.

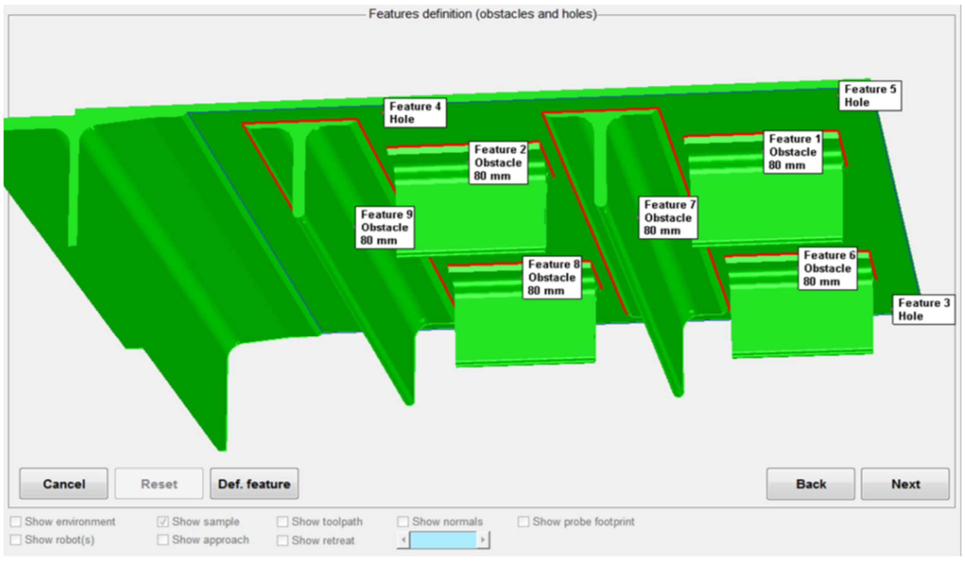
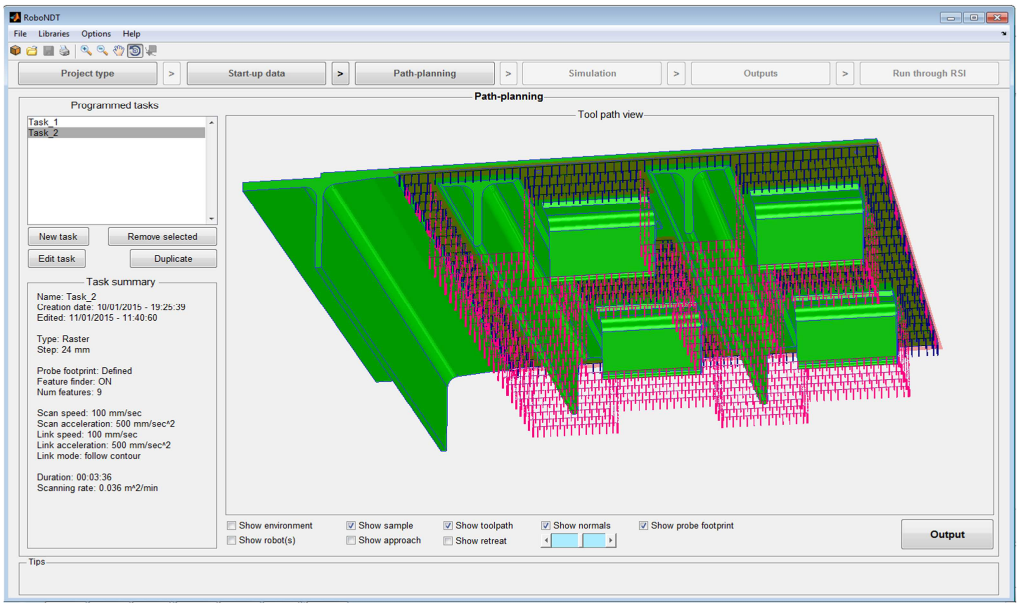


Figure 5.40 – Definition of internal features.

Figure 5.41 shows the resulting tool-path for the surface with internal features.



Path-planning

Task summary

Name: Task_2
Creation date: 10/01/2015 - 19:25:39
Edited: 11/01/2015 - 11:40:60

Type: Raster
Step: 24 mm

Probe footprint: Defined
Feature finder: ON
Num features: 9

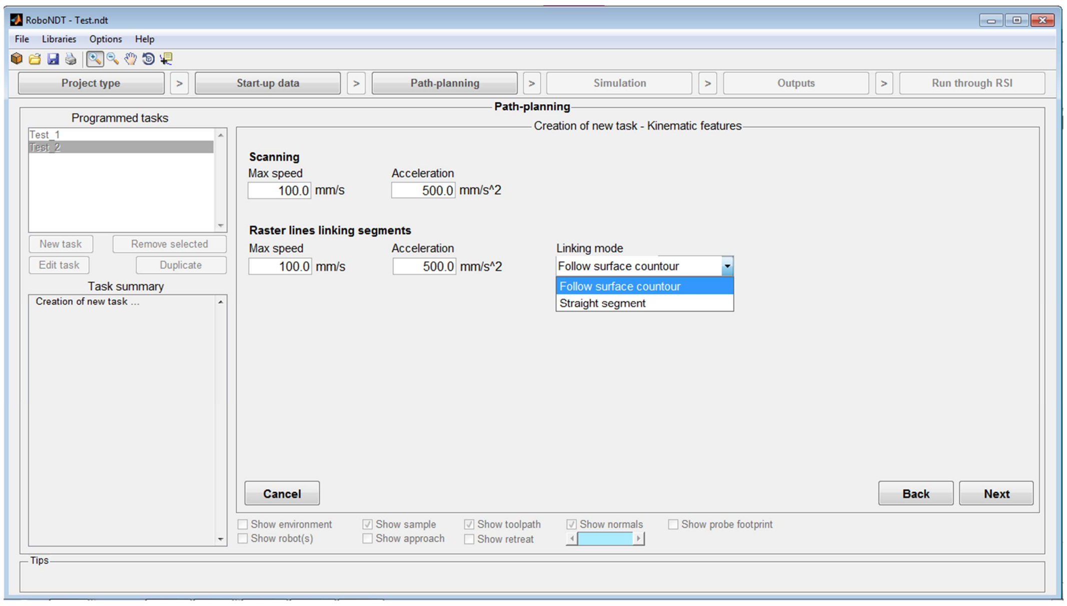
Scan speed: 100 mm/sec
Scan acceleration: 500 mm/sec^2
Link speed: 100 mm/sec
Link acceleration: 500 mm/sec^2
Link mode: follow contour

Duration: 00:03:36
Scanning rate: 0.036 m^2/min

Figure 5.41 – Raw tool-path for the surface with internal features.

5.1.7. Kinematics features

The tool-path generated after the interactive definition of the tool-path properties is a *raw tool-path*, meaning that it is not yet ready to be used to be executed through the robot arm. This is achieved through defining the kinematics features. Speed and acceleration for the scanning portions of the tool-path and for the linking portions are specified through the interface shown in Figure 5.42. The user can also select the *linking mode*. It is the way the portions of an inspection tool-path are linked: through straight segments of through following the surface contour.



Path-planning

Creation of new task - Kinematic features

Creation of new task ...

Scanning
Max speed 100.0 mm/s
Acceleration 500.0 mm/s^2

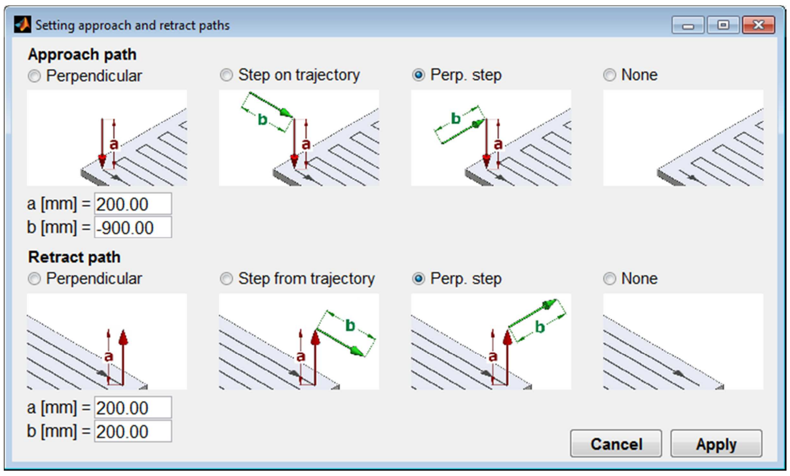
Raster lines linking segments
Max speed 100.0 mm/s
Acceleration 500.0 mm/s^2
Linking mode: Follow surface countour
- Follow surface countour
- Straight segment

Figure 5.42 – Kinematics features.

5.1.8. Approach and retract motions

The robot manipulators need to always approach the starting point of an inspection tool-path in a safe way, avoiding collisions with the sample or with any other element present in the working envelope. Likewise, the same is valid for the retracting point from the end point of the inspection to the HOME position.

RoboNDT allows specification of the approach and retract motion for each robot task through the interface shown in Figure 5.43. A range of possible approaches and retracts is available and the user can customize the length of the vectors.



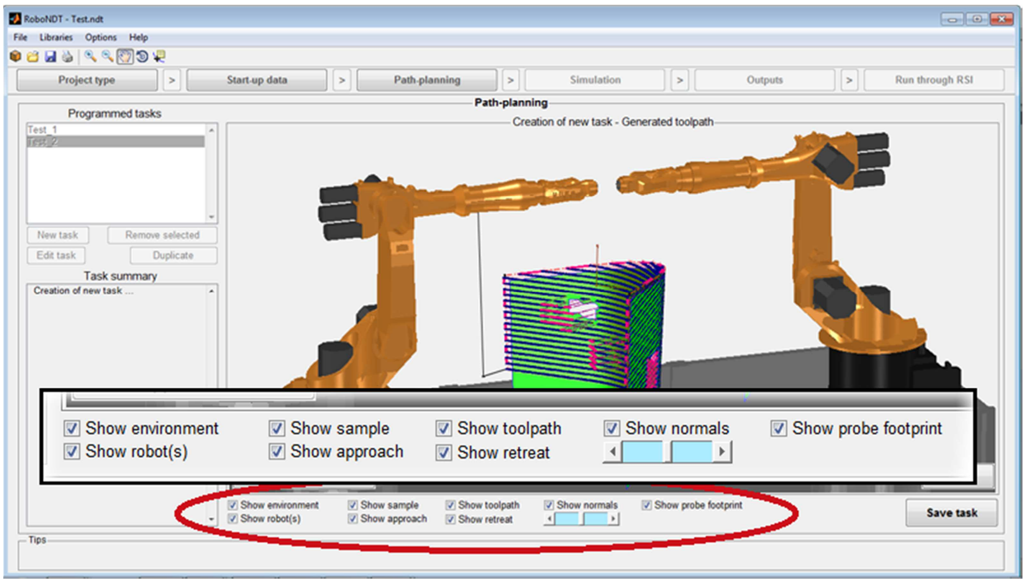
Approach path
Perpendicular / Step on trajectory / [✓ Perp. step] / None
a [mm] = 200.00
b [mm] = -900.00

Retract path
Perpendicular / Step from trajectory / [✓ Perp. step] / None
a [mm] = 200.00
b [mm] = 200.00

Figure 5.43 – Setting approach and retract paths.

5.1.9. Flexible display

A range of display options was implemented to enable flexible display.



[✓] Show environment
[✓] Show robot(s)
[✓] Show sample
[✓] Show approach
[✓] Show toolpath
[✓] Show retreat
[✓] Show normals
[✓] Show probe footprint

Figure 5.44 – Display options.

Figure 5.44 shows a close up of the display options at the bottom of the programme window. It is possible to choose to show or hide the robot environment, the robots, the model of the sample, the tool-path, the normals, the inspection footprint, the approach and the retract paths. The number of the displayed normals can be adjusted through using a slide control.

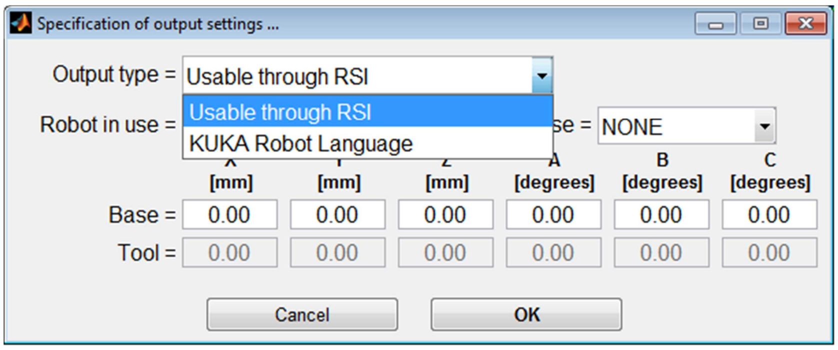
5.1.10. Output

RoboNDT is functional in its development status achieved by the end of the research program reported by this thesis; however the software has not yet reached its full development. The full-developed version should present the possibility to operate a *simulation* and an *output* module, as it is described in Section 5.1.1 (Modular GUI architecture).

The output of executable tool-paths is currently allowed through using the *output* button at the bottom right corner of the *path-planning* module (Figure 5.31 – Layout of the path-planning module.). This button enables the output of any task selected by the user in the list of *Programmed tasks*.

Figure 5.45 shows the GUI pop-up window implemented as temporary solution to the lack of a proper, more flexible, output module. The window allows specifying the type of output (usable through RSI or KUKA Robot Language), the destination robot and the robot tool. The user can customize the robot base parameters, if a local base is preferred to the robot root (X=Y=Z=A=B=C=0). The tool parameters can also been specified when a CUSTOM tool type is selected from the relative pop-up menu.

(a)


Specification of output settings ...

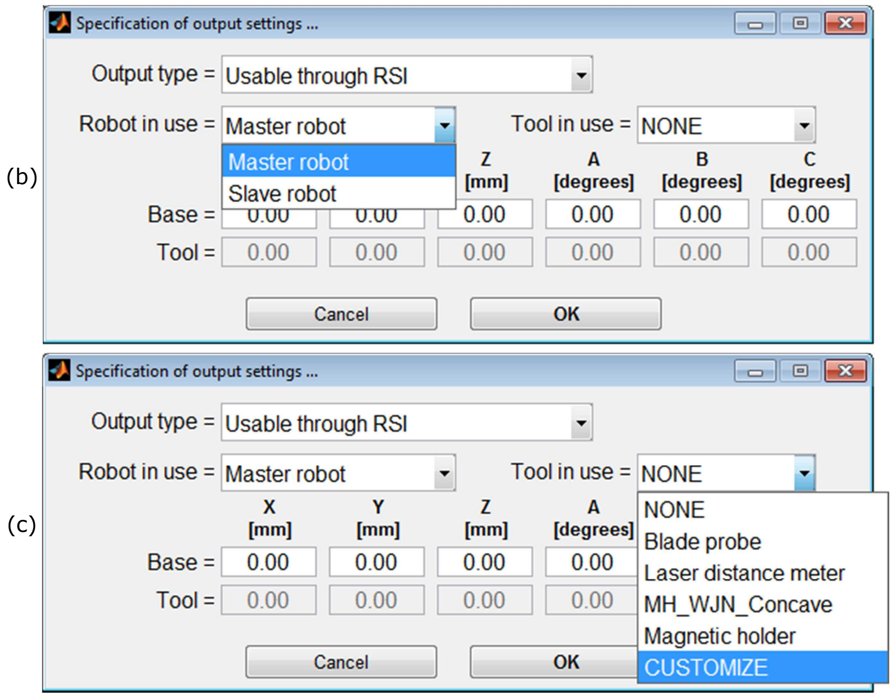
Output type = Usable through RSI

Usable through RSI
KUKA Robot Language

Robot in use =

se = NONE

	X [mm]	Y [mm]	Z [mm]	A [degrees]	B [degrees]	C [degrees]
Base =	0.00	0.00	0.00	0.00	0.00	0.00
Tool =	0.00	0.00	0.00	0.00	0.00	0.00


(b)
Specification of output settings ...
Cancel
OK
(c)
Specification of output settings ...
Cancel
OK

Output type = Usable through RSI
Robot in use = Master robot
Master robot
Slave robot
Tool in use = NONE

	X [mm]	Y [mm]	Z [mm]	A [degrees]	B [degrees]	C [degrees]
Base =	0.00	0.00	0.00	0.00	0.00	0.00
Tool =	0.00	0.00	0.00	0.00	0.00	0.00

Output type = Usable through RSI
Robot in use = Master robot
Tool in use = NONE
NONE
Blade probe
Laser distance meter
MH_WJN_Concave
Magnetic holder
CUSTOMIZE

Figure 5.45 – Specification of output settings.

If *"usable through RSI"* is selected for the *Output type*, the software generates two output text files for each robot task: the first contains all command coordinates the robot needs to receive to inspect the target surface, and a second short log file containing the points to set the initial and final motion to approach the starting point of the inspection and to abandon the endpoint. These two files have very simple syntax; each line merely contains 6 coordinates (x, y, z, A, B, C) to drive the robotic arm to a specific pose.

5.1.11. Executable application package

The latest version of RoboNDT can be downloaded as an executable application package from: http://www.strath.ac.uk/eee/research/cue/downloads/.

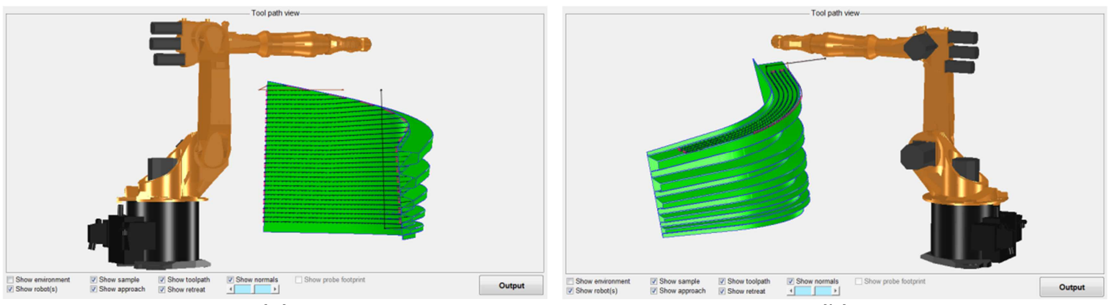
5.2. Validation experiments

This section presents the experimental tests carried out at TWI Technology Centre to validate the accuracy of the tool-paths generated through RoboNDT. The experiments were also designed for the final validation of the reliability of the novel approach introduced by this research. It is based on external control of the robot motion and simultaneous collection of feedback coordinates and NDT data carried out by the C++ context.

RoboNDT was integrated to an experimental version of the data acquisition software developed by IntACom (see Chapter 6). This allowed interfacing RoboNDT with a powerful NDT data acquisition and post-processing platform and with an effective phased-array ultrasonic receiver (the Micropulse 5PA from PeakNDT). The software experimental setup was equivalent to the schematic representation given by Figure 5.1.

5.2.1. Aerospace winglet sample

Figure 5.46 shows the inspection tool-paths generated through RoboNDT for the experimental tests to be carried out with an aerospace winglet. For the sake of testing the software with surfaces curving in different directions, the main skin of the winglet (Figure 5.46a) and the top surface of one of its back wall beams (Figure 5.46b) were considered for path-planning. The main skin surface has an area of 1.6 m^2; the beam surface has an area of 0.5 m^2. The generated tool-paths are raster scans with a 29.4 mm raster step. This step is suitable for phased-array ultrasonic inspection (PAUT) when a 64 element, 0.6 mm pitch phased-array probe is employed and its elements are fired with a focal law that uses a linear sweep with sub-aperture of 16 elements.



(a) (b)

Figure 5.46 – Tool-paths for the validation experiments carried out with the aerospace winglet.

The phased-array focal law and the setting of the ultrasonic receiver (the Micropulse 5PA) were set to acquire C-scans with resolution of 1.2 mm in all directions. The tool-paths accuracy is evaluated through comparing the feedback coordinates received from the robot encoders and the command coordinates. Two sets of tool-paths were created to execute the NDT inspection at 100 mm/s and 300 mm/s, maintaining the same robot acceleration equal to 500mm/s^2. The positional error is calculated as the distance between the commanded tool central points (TCPs) and the reached points as there were measured by the robot encoders. The orientation error is calculated as the mismatch angle between the commanded rotation matrix and the rotation matrix computed from the feedback roll, pitch and yaw angles.

Table 5.1 shows the maps of position and orientation errors for all tool-paths generated through RoboNDT. For the sake of helping the comparison, the same colour scale has been maintained where possible. As expected it can be seen that with faster speed errors are worse, because of the dynamic factors affecting the robotic hardware (e.g. inertia). The bigger errors localized at the start and end of the raster tool-path passes are due to the unavoidable backlash present in the robot gears.

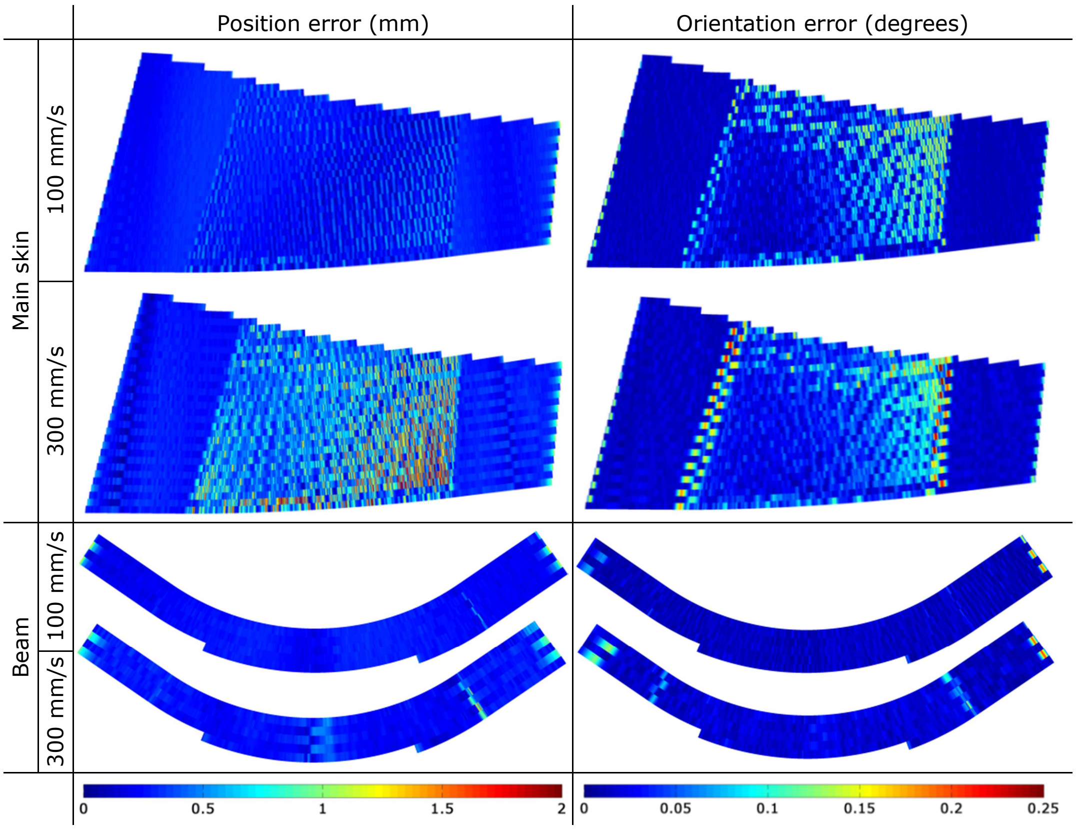


Table 5.1 – Maps of position and orientation errors.

Table 5.2 reports the maximum and Root Mean Square (RMS) errors. The maximum position error is equal to 2.70 mm and the maximum orientation error is equal to 0.29 degrees.

		Inspection speed			
		100 mm/s		300 mm/s	
		Main skin	Beam surface	Main skin	Beam surface
Position error (mm)	Max	1.37	1.18	2.70	1.53
	RMS	0.30	0.27	0.52	0.33
Orientation error (degrees)	Max	0.21	0.20	0.29	0.24
	RMS	0.04	0.02	0.05	0.03

Table 5.2 – Maximum and Root Mean Square (RMS) errors.

The variability of the standoff between the probe and the surfaces is shown in Table 3 with Time-Of-Flight (TOF) maps of the ultrasonic wave reflected from the scanned surface to the probe. The TOF values have been divided by the speed of sound in the sample to quantify the standoff variability in millimetres. The standoff relative to the RoboNDT tool-path is compared to that relative to the tool-path generated through commercially available path-planning software based on the Dassault Delmia V5 platform (FastSurf). The comparison is made for the same travelling speed of 300 mm/s and acceleration of 500 mm/s^2.

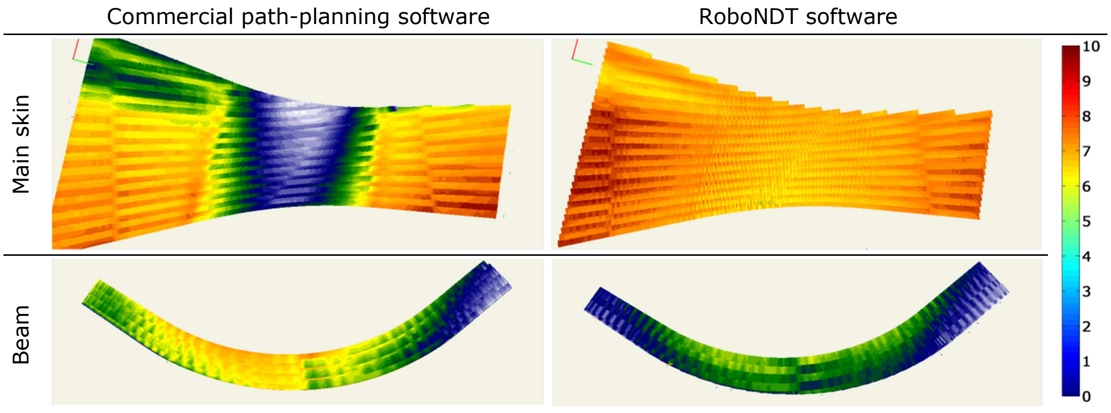


Table 5.3 – Maps of standoff between probe and scanned surface.

The variability of the standoff is within 10 mm for the tool-paths created with the commercial software and within 4.5 mm for the RoboNDT tool-paths. The experimental data demonstrates that the path errors, achievable through externally controlled robots are lower than the errors given by the traditional approach.

5.2.1.1. NDT Results

The accuracy of the RoboNDT tool-paths is higher than the tool-paths obtained through the commercial path planning software. However, the comparison of the NDT results (Table 5.4) does not show any visible difference between the inspection carried out through the traditional and the new approach. For the current application, the level of accuracy is sufficient as the intended NDT delivery is accomplished using a water jet coupling approach [5, 6]. The water path from water nozzle to sample surface can easily accommodate such tool-path inaccuracies. The bottom row of Table 4 shows the close-up of an array of artificial squared delaminations embedded within the thickness of the winglet main skin. The smallest delaminations have a size of 5 mm and are visible in both cases.

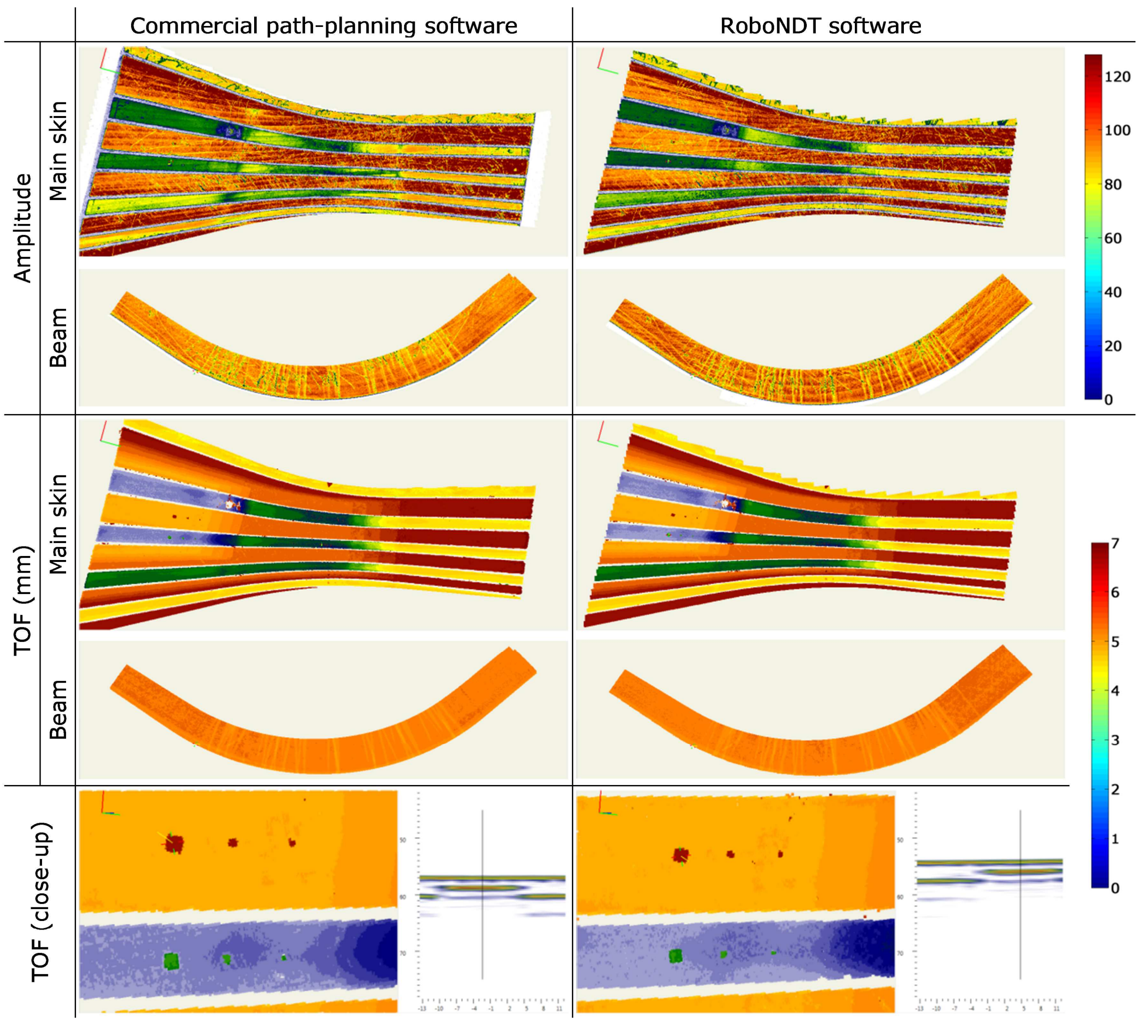


Table 5.4 – NDT results.

5.2.2. Reverse engineered titanium fan blade

A second set of experiments was carried out with a different sample. RoboNDT was used to generate tool-paths for a reverse engineered surface. A way to generate a suitable STL file from the raw point cloud was investigated. In order to reduce the noise coming from the cloud points, a smoothing algorithm was implemented. Ultrasonic phased array scans were carried out to monitor the variability of the standoff between the ultrasonic probe and the test surface.

5.2.2.1. Getting a suitable meshed surface from a point cloud

A component with challenging geometry was selected to carry out more validation experiments for the RoboNDT software. It is a titanium fan blade, 640 mm long and 300 mm wide (in average). Since the CAD file of the component is not available, the surface of the blade was digitalized at the University of Strathclyde using the FARO Quantum. This is a piece of reverse engineering equipment. The FARO Quantum was used in conjunction with the laser line probe V3 [7]. With a volumetric maximum deviation of ±55μm for contact probe measurements, and ±74μm for non-contact laser scanned points, the FARO arm proved ideal to carry out the digitisation of the complex surface.

A point cloud with 139757 points was obtained (Figure 5.47b), resulting in an average of 72 points per square centimetre.

The component was shipped from the University of Strathclyde to TWI Technology Centre (Wales), to allow ultrasonic phased array scanning to be undertaken through the IntACom data acquisition software and the external control of the KUKA KR16 L6-2 robot available at TWI.

Commercially available off-line path-planning software requires precise vectorial CAD models of the components of interest to generate suitable robot tool-paths. These CAD formats are usually proprietary. RoboNDT was developed to support the STL file format. This format describes geometries through tessellated (or mesh) models and is not proprietary. An additional advantage foreseen since the beginning of RoboNDT lies in the fact that an STL file can be generated from a point cloud with appropriate software (most of which is open-source). The algorithms to convert a point cloud into a tessellated surface are also well-known in the literature [8]. The possibility to operate path-planning for surfaces digitalized as point clouds is important to pursue the introduction of metrology and surface mapping to the area of robotic NDT inspection of complex geometries, in the near future.

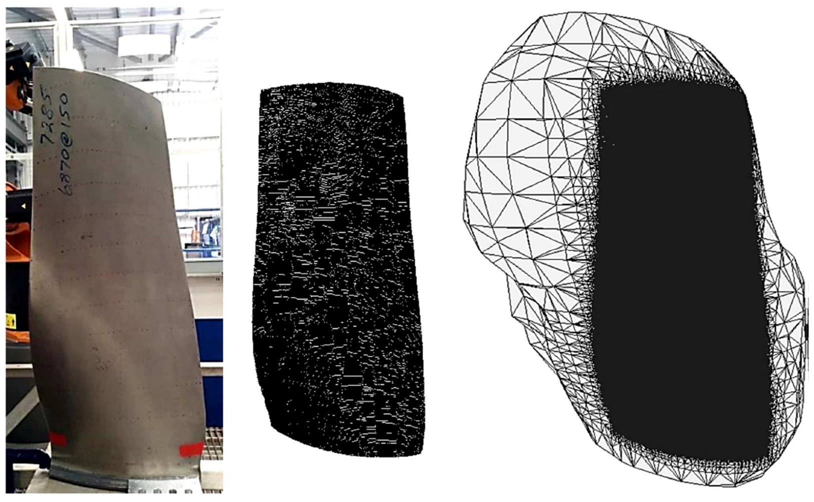
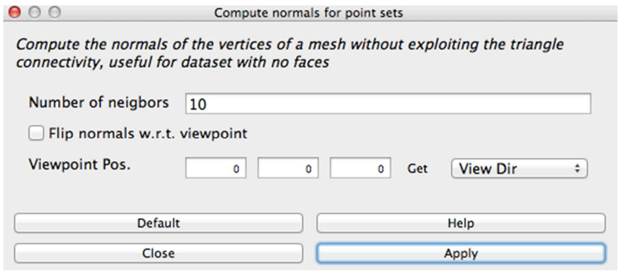

(a) (b) (c)

Figure 5.47 – Fan blade sample (a), point cloud (b) and raw tessellated surface (c).

MeshLab is an open-source piece of software for creating, processing and editing of unconstructed 3D triangular meshes [9, 10]. MeshLab allows importing point clouds and the generation of triangular meshes.

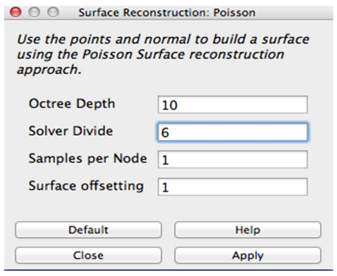
After importing the point cloud of interest, it is necessary to convert it to a cloud of oriented points. This is done with MeshLab, through clicking on *Filters -> Normals, Curvatures and Orientation -> Compute Normals for Point Sets*. The window shown in Figure 5.48a is displayed. The number 10 represents the number of points used to estimate the tangent plane at each point set. From this, the normal direction is calculated for each point.


Compute normals for point sets

Compute the normals of the vertices of a mesh without exploiting the triangle connectivity, useful for dataset with no faces

Number of neigbors: 10
Flip normals w.r.t. viewpoint: ☐
Viewpoint Pos.: 0 0 0 Get View Dir

(a)


Surface Reconstruction: Poisson

Use the points and normal to build a surface using the Poisson Surface reconstruction approach.

Octree Depth: 10
Solver Divide: 6
Samples per Node: 1
Surface offsetting: 1

(b)

Figure 5.48 – Computation of normals (a) and Poisson surface reconstruction (b).

Surface reconstruction from oriented points can be cast as a spatial Poisson problem [8]. MeshLab uses a Poisson formulation that considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. This function can be used by clicking on *Filters -> Remeshing, Simplification and Reconstruction -> Surface Reconstruction: Poisson*.

The Poisson algorithm takes the normals from each point to calculate the surface. The Poisson parameters used for the generation of the mesh of the blade are shown in Figure 5.48b. The Octree Depth is the fundamental parameter. Larger values of Octree Depth lead to more accurate meshes, but also to more computational time needed. Good values for the Octree Depth span between 6 and 10.

The obtained mesh (Figure 5.47c) contains 128069 vertexes and 256072 triangles; it was exported to an STL file. The mesh generated through the Poisson algorithm is not ready to be used for path-planning. The boundary of the tessellated surface does not coincide with the boundary of the point cloud. Although the Poisson algorithm is perfect for mesh reconstruction, it is designed for closed surfaces (e.g. statues and all-around sampled objects). The problem of trimming the mesh to faithfully reproduce the boundary of the point cloud is not addressed by the algorithm. The mesh could be trimmed manually, by deleting the triangular faces lying outside the point cloud, but it would produce jagged edges not suitable to path-planning.

The problem of trimming the mesh using the boundary of the point cloud is not simple. In fact, the identification of the point cloud borderline is not a trivial task. The human brain is able to infer the border of a point cloud by simply looking at the arrangement of the sparse points. For a machine, a point cloud is an entity without a well-defined boundary. A MATLAB script has been developed to find the boundary points by analysing the vectors linking each point to its neighbours. Figure 5.49a shows the close-up of a small area of a point cloud. The vectors of an internal point depart from it to any direction in 360 degrees (Figure 5.49b). For a boundary point, the vectors point towards other external points or towards the inner part the point cloud; the vectors cluster within a smaller range of angles (Figure 5.49c).

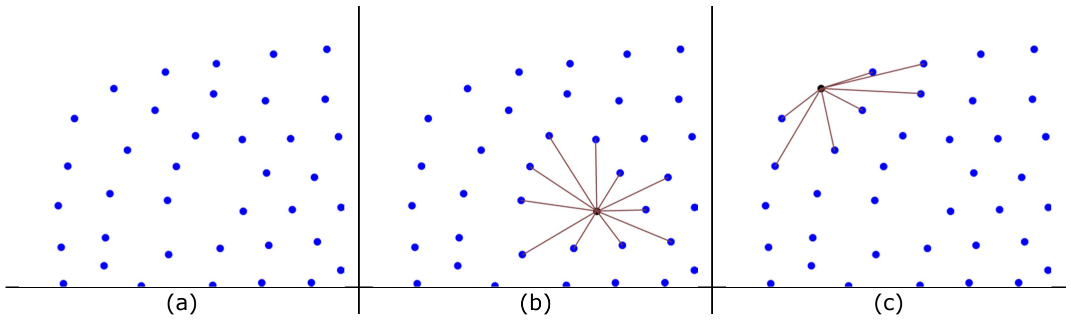


Figure 5.49 – Close up of a group of points (a), internal point (b) and boundary point (c).

Since the point cloud is not bi-dimensional, but is in the 3D space, the arrangement of the vectors has to be observed from above each point, using the local normal as view direction. A MATLAB script capable of using this peculiar difference between internal and boundary points has been developed to find the boundary of the fan blade point cloud.

The result is shown in Figure 5.50a; the boundary is divided into the four continuous edges of the fan blade. The boundary given by the recognised boundary points is quite jagged, as it is evident by looking at the close-up in Figure 5.50b. Such a boundary is not yet suitable to trim the mesh.

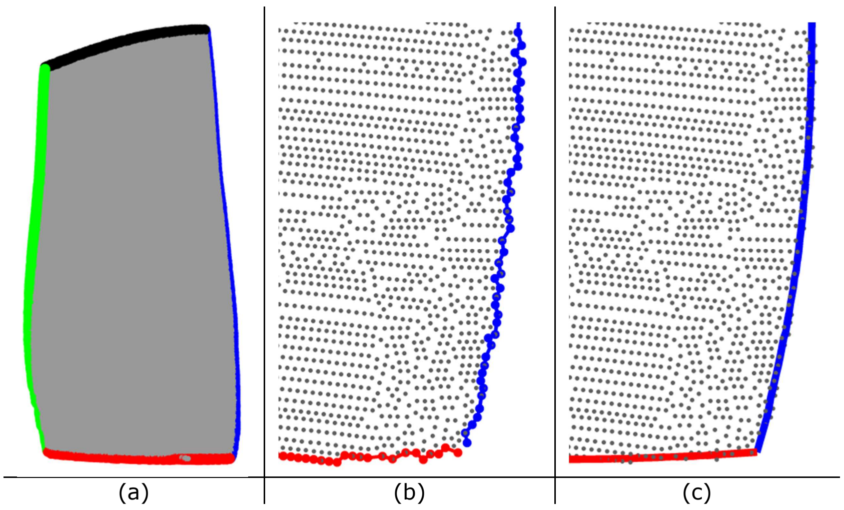


Figure 5.50 – Recognised boundary points into four edges (a), close-up of the raw boundary (b) and of the smoothed boundary (c).

The boundary points could be interpolated through spline curves; however the polynomial grade of the curves is unknown. It was decided to implement a more robust approach.

The Fast Fourier Transform (FFT) of a waveform is a well-known way to display the frequency content of a waveform. It allows filtering of specific frequencies or the application of high or low-pass filters, to respectively damp down low or high frequencies.

The pursued idea consists in filtering each of the Cartesian components of the boundary edges in the frequency domain and in returning to the Cartesian components by applying the Inverse Fast Fourier Transform (IFFT). The edges with the filtered Cartesian components give the smoothed boundary.

The FFT works on periodic waveforms. Figure 5.51 clarifies how a periodic waveform is obtained from the original pattern of each Cartesian component of a given edge. The figure shows the z-component of one of the edges of the blade; the original pattern is quite noisy. It is initially translated along the direction of the ordinate axis to move the first point of the pattern to the origin of the plot. The pattern is then rotated by the angle α to move the last point of the pattern on the horizontal axis. A copy of the translated and rotated pattern is inverted, flipped and appended to the end; it constitutes a joining portion. The resulting pattern is a periodic waveform.

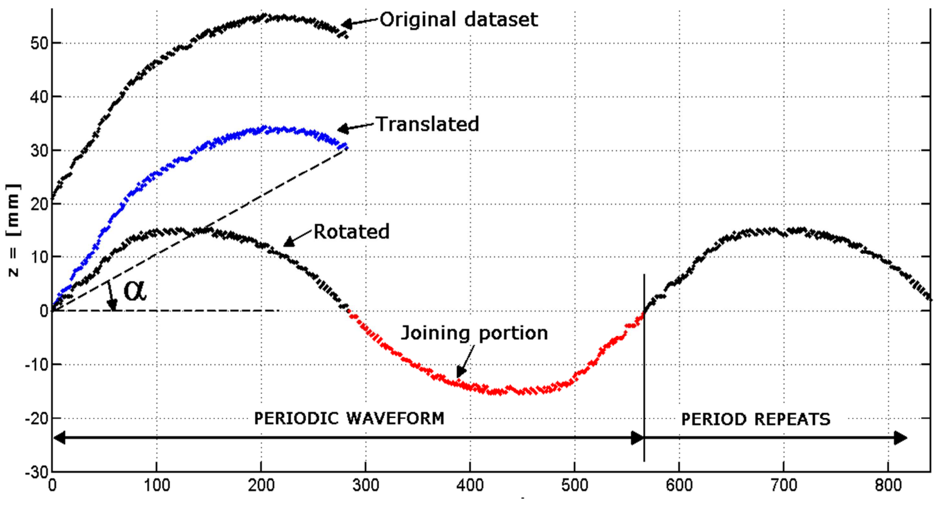


Figure 5.51 – Creation of a periodic waveform from the original pattern.

The FFT is applied to the periodic waveform of the component to get its spectral content. Then, a low pass filter is applied and the IFFT is used to get the smoothed waveform. The first part of the waveform (with the same length of the original pattern) is rotated by $-\alpha$ and translated back to the position of the original dataset.

By filtering all the Cartesian components of all the edges, the smooth boundary of the fan blade was obtained. A portion of the smoothed boundary is shown in Figure 5.50c. This boundary was used to trim the original mesh (Figure 5.52a). The resulting mesh of the fan blade, suitable for path-planning is shown in Figure 5.52b.

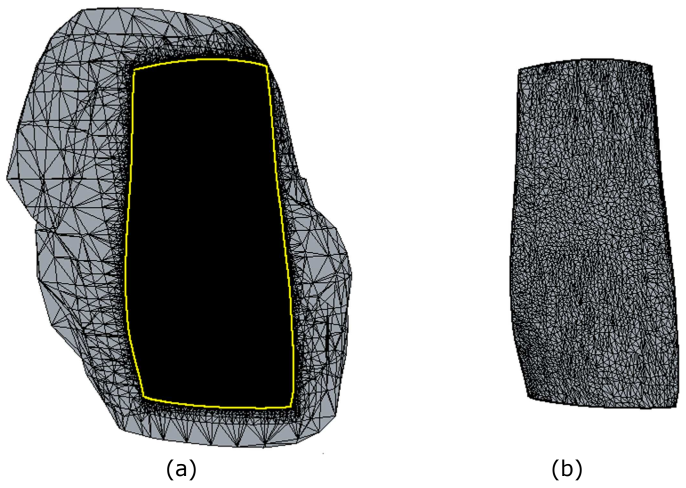
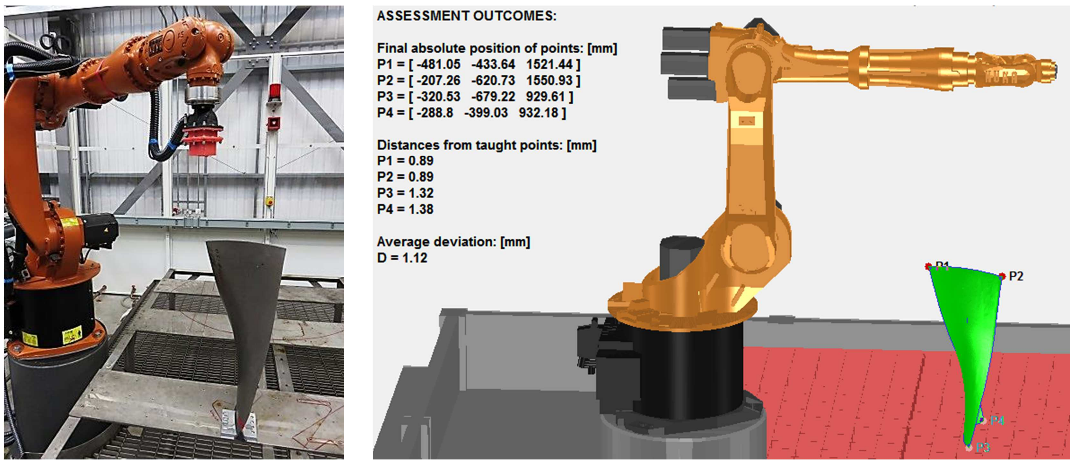


Figure 5.52 – Trim of original mesh with the smoothed boundary (yellow curve) (a) and resulting mesh (b).

5.2.2.2. Path-planning

The mesh was imported into RoboNDT and its position was assessed to match with the real position of the component within the robot cell. The four corners of the surface were used as reference points for the sample positioning. The real coordinates of the points were obtained by jogging the robot arm to each point and reading the robot coordinates, displayed on the teach-pendant.

The positioning algorithm embedded in RoboNDT was able to position the virtual model of the blade with an average mismatch of 1.12mm from the real sample position (Figure 5.53b).

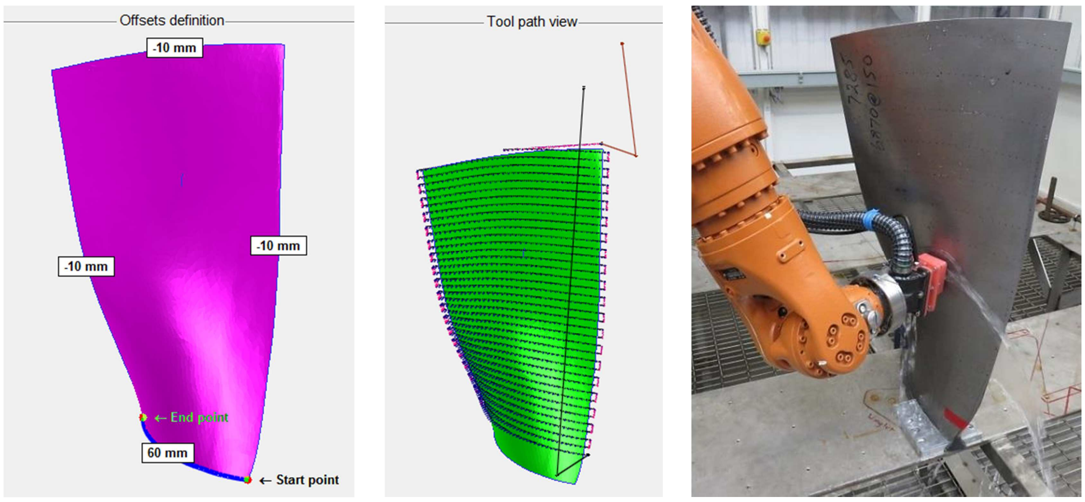


```
ASSESSMENT OUTCOMES:

Final absolute position of points: [mm]
P1 = [ -481.05   -433.64   1521.44 ]
P2 = [ -207.26   -620.73   1550.93 ]
P3 = [ -320.53   -679.22   929.61 ]
P4 = [ -288.8   -399.03   932.18 ]

Distances from taught points: [mm]
P1 = 0.89
P2 = 0.89
P3 = 1.32
P4 = 1.38

Average deviation: [mm]
D = 1.12
```

(a) (b)

Figure 5.53 – Assessment of sample position.

Figure 5.54a shows the key features of the path-planning. The bottom edge was used as reference edge for the generation of the raster tool-path. An offset of 60mm from the bottom was applied to avoid collisions between the robot wrist and the table supporting the sample. A negative offset of -10mm was applied to the other edges to allow complete scanning of the blade boundary. Figure 5.54b shows the generated tool-path after setting the approach and retract paths for the robot to safely approach the first point of the scan and come back to the HOME position.



(a) (b) (c)

Figure 5.54 – Path-planning (a & b) and snapshot of the PAUT inspection (c).

Figure 5.54c shows a snapshot taken during the execution of the tool-path.

5.2.2.3. Results – Robot accuracy

The tool-path was initially carried out at 100 mm/s. It was found that the robot movement was quite jerky and the cause of this was found in the instrumental noise present in the original point cloud being transmitted to the reconstructed tessellated surface.

A solution to this problem was successfully investigated. The filtering method, developed for the mesh boundary and based on the Fast Fourier Transformation, was applied to each pass of the raster tool-path. Both the Cartesian and the angular components of the command tool-path were filtered to remove the high frequency content in the frequency domain.

Therefore, the tool-path was repeated at 100 mm/s showing a smooth and fluid robot movement. The same tool-path was also carried out at 300 mm/s.

The following tables show quantitative and qualitative comparisons between the command and the feedback robot coordinates.

		100 mm/s Original tool-path	100 mm/s Smoothed tool-path	300 mm/s Smoothed tool-path
Position error (mm)	Max	4.165	1.772	2.592
	RMS	0.544	0.371	0.470
	Mean	0.445	0.331	0.413
Orientation error (degrees)	Max	0.447	0.292	0.355
	RMS	0.054	0.041	0.062
	Mean	0.042	0.029	0.050

Table 5.5 – Position and angular errors.

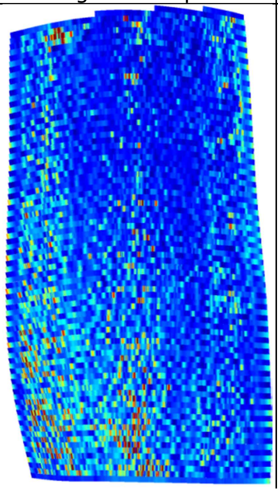
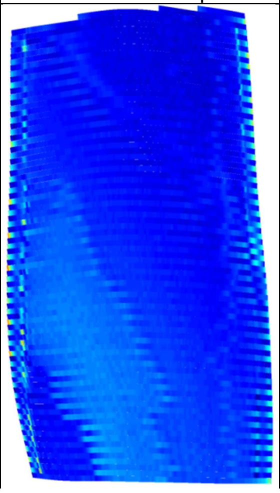
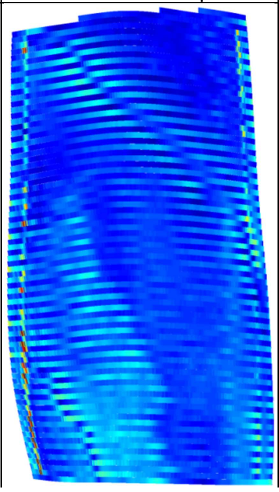
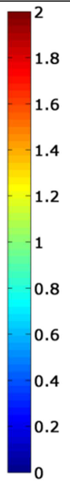
100 mm/s Original tool-path	100 mm/s Smoothed tool-path	300 mm/s Smoothed tool-path	(mm)
			

Table 5.6 – Maps of position errors.

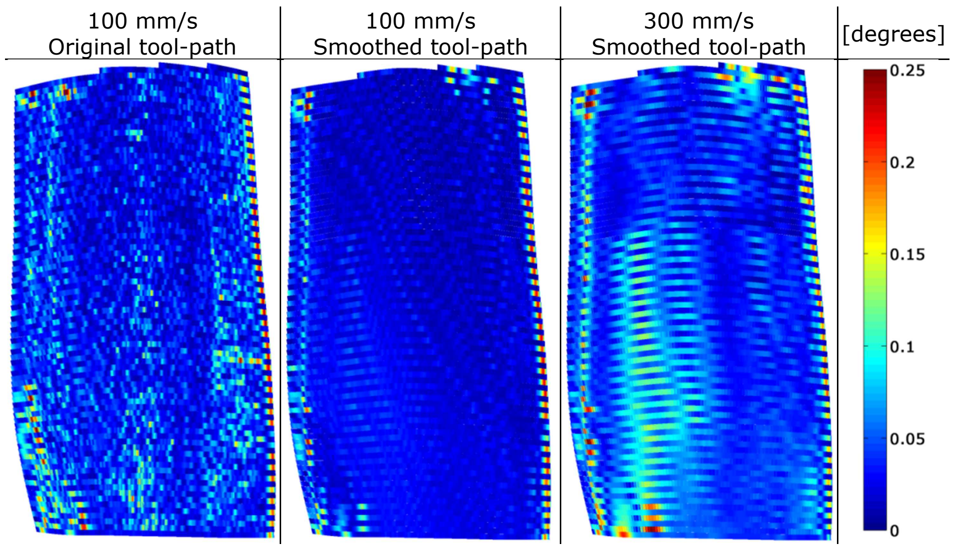


Table 5.7 – Maps of orientation errors.

5.2.2.4. Results – NDT C-scans

The tool-paths were used to carry out a pulse-echo ultrasonic inspection of the blade. A 64-element, 5 MHz, 0.6 mm pitch phased array probe was used. The focal law was set to an unfocused linear sweeping scan with a probe sub-aperture of 16 elements. The scan was carried out using the water-jet coupling method.

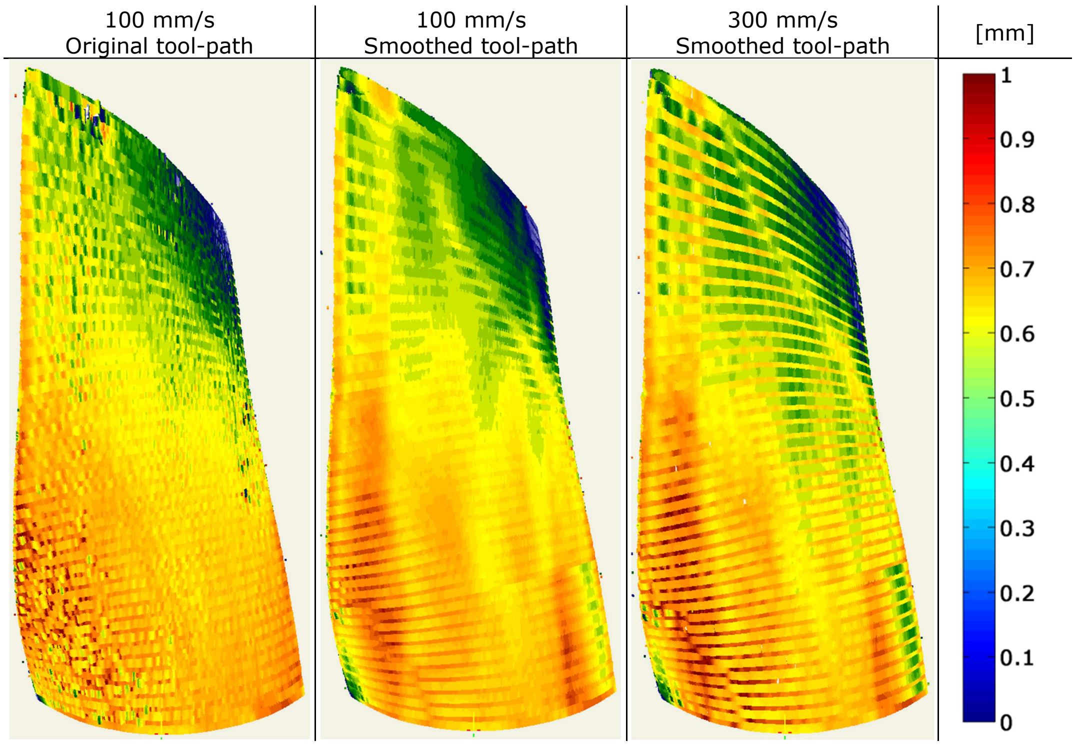


Table 5.8 – Time of Flight of the front wave (standoff variability).

The Time of Flight (TOF) maps of the front wall wave were obtained setting the speed of propagation equal to the speed in water (1.47 km/s). Therefore the maps show the variability of the standoff between the probe and the surface. The variability is within 1 mm, 4 times more accurate than the standoff variability obtained for the GKN Winglet (where it was within 4 mm).

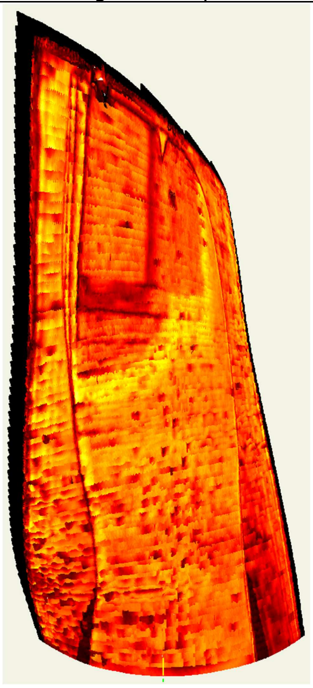
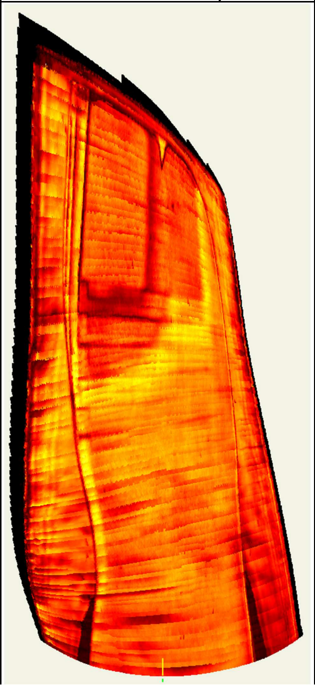
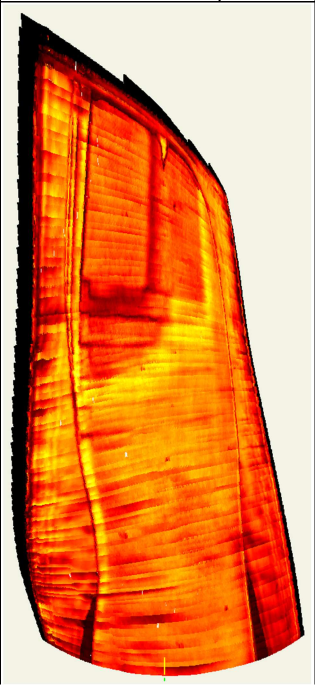
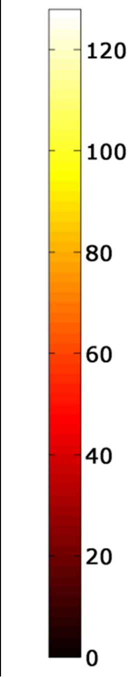
100 mm/s Original tool-path	100 mm/s Smoothed tool-path	300 mm/s Smoothed tool-path	Amplitude of quantized signal
			

Table 5.9 – Amplitude C-scans of the back wall echo.

The maps of the back-wall echo amplitude, in Table 5.9, show a dramatic improvement moving from the original tool-path to the smoothed tool-paths. No big differences are notable between the 100 mm/s scan and the 300 mm/s scan, meaning that the tool-path remains accurate enough for the kind of ultrasonic inspection carried out in this work.

5.3. Future development

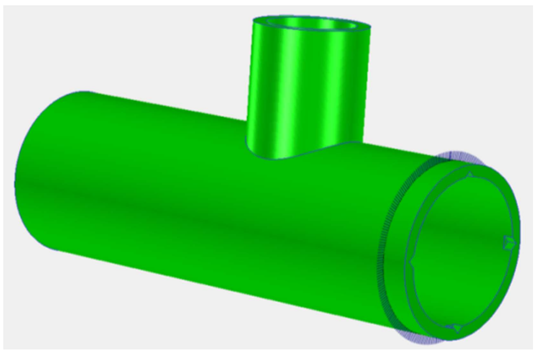
RoboNDT was designed to provide a user-friendly interface for robotic NDT. Technicians often face steep learning curves to become fully trained NDT inspectors. RoboNDT can be integrated to NDT data acquisition software to create seamless platforms for the robotic inspection.

RoboNDT is intended to transform the robot path-planning problem into an easy step of the inspection process. The latest version of the software developed within this work encapsulates novel path-planning algorithms, the outcomes of the research and it is shaped to overcome practical limitations of current OLP software.

Future development for RoboNDT should focus on short term and long term objectives. Among the short term objectives it is possible to include the optimization of the performances of the background code (e.g. bug fixing, speeding up computation time, etc.) and the implementation of the simulation module according to the full kinematics formulation presented in Section 4.8.2.

Long term objectives can include commercial exploitations. Furthermore, the MATLAB nature of RoboNDT makes it suitable to support research activities and/or teaching activities in academic institutions.

The software, already capable of supporting fixed robot manipulators, is extendable to mobile robot manipulators (6-DoF manipulators mounted on tracks, rotary tables, etc.) and to sister platforms, like mobile robots. An immediate use of RoboNDT in this direction could concern the generation of mobile robot path for the inspection of big diameter pipework junctions. The path-planning module of RoboNDT is already capable of coping with *folded* or *closed* surfaces (e.g. cylindrical, conical surfaces). Figure 5.54 shows examples of inspection paths for butt-welded and T-welded pipework junctions.


(a)

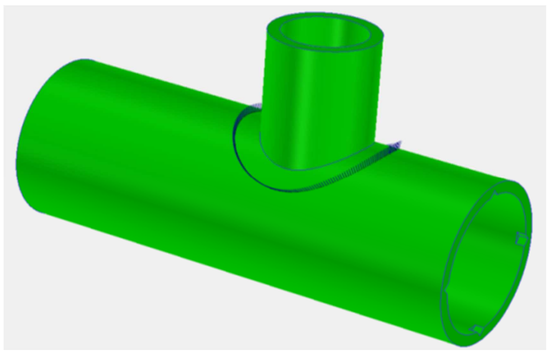

(b)

Table 5.10 – Examples of inspection paths for butt-welded (a) and T-welded (b) pipework junctions.

5.4. Wrapping-up

This chapter described the features of the RoboNDT software. The development of the software was embraced to overcome the deficiencies of current commercial OLP.

RoboNDT was developed with the MATLAB programming language; however it can run outside the MATLAB environment through compiling it as an .EXE or a .DLL application. It can be integrated to NDT data acquisition software modules to achieve the visions pursued by this work. Crucially, it enables the possibility of establishing seamless robotic NDT software platforms.

RoboNDT is intended to transform the robot path-planning problem into an easy step of the inspection process. The latest version of the software developed within this work encapsulates novel path-planning algorithms, the outcomes of the research and it is shaped to overcome practical limitations of current OLP software.

Future development for RoboNDT includes the optimization of the performances of the background code (e.g. general bug fixing, speeding-up of computation time, etc.) and the implementation of simulation capabilities.

Long term objectives can comprise commercial exploitations. RoboNDT is also suitable to support research and/or teaching activities in academic institutions.

The idea to extend the software to support mobile robot manipulators (6-DoF manipulators mounted on tracks, rotary tables, etc.) and sister platforms, like mobile robots, makes RoboNDT a very promising application.

References

[1] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in *Robotics-DL tentative*, 1992, pp. 586-606.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research,* vol. 5, pp. 90-98, 1986.

[3] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation*: MIT press, 2005.

[4] M. Gerdts, R. Henrion, D. Hömberg, and C. Landry, "Path planning and collision avoidance for robots," *Numerical Algebra, Control and Optimization,* vol. 2, pp. 437-463, 2012.

[5] E. Cuevas, M. López, and M. Garcìa, "Ultrasonic Techniques and Industrial Robots: Natural Evolution of Inspection Systems," presented at the 4th International Symposium on NDT in Aerospace, Ausburg, Germany, 2012.

[6] A. Maurer, W. Haase, and W. De Odorico, "Phased array application in industrial scanning systems," presented at the ECNDT, Berlin (Germany), 2006.

[7] J.-P. Monchalin, C. Neron, P. Bouchard, and R. Heon, "Laser-ultrasonics for inspection and characterization of aeronautic materials," *Journal of Nondestructive Testing & Ultrasonics(Germany),* vol. 3, p. 002, 1998.

[8] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[9] MeshLab. Available: http://meshlab.sourceforge.net/

[10] P. Cignoni, M. Corsini, and G. Ranzuglia, "Meshlab: an open-source 3d mesh processing system," *Ercim news,* vol. 73, pp. 45-46, 2008.

6. SYSTEMS ENGINEERING AND INTEGRATION

6.1. The IntACom project

As it was explained in the introductive chapter of this thesis, the research in the field of robotic NDT inspections has been strongly driven by industrial need arising in recent decades.

In the aerospace industry the uptake of composites has been driven by the desire to improve stiffness to weight ratios and develop lighter structures, to improve corrosion, impact and fatigue resistance. Big steps have been made in these areas, but the cost of manufacture of composite structures is still a concern. This is partially caused by the cost of raw materials, but mainly due to the labour intensive manufacturing techniques. The aerospace regulatory requirements to inspect every aerospace part can result in the NDT process being the cause of production slowdown. The possibilities given by modern technologies enable the development of evermore complex component geometries. This then necessitates the deployment of more advanced and fast NDT inspection in a production environment.

The IntACom project was a 3-years project developed by TWI and supported by a number of major aerospace partners (Rolls-Royce, GKN and Bombardier) and Welsh government. The aim of the IntACom development project was to reduce the time taken for inspection of complex geometry composite components by a factor of four. This reduced inspection time was achieved by addressing three areas: automation of inspection, employment of advanced PAUT and software enhancement through the use of techniques, such as assisted defect recognition and scan display management.

The project produced a robotic NDT inspection prototype system capable of inspecting complex geometry in an improved manner compared to the traditional UT immersion tanks. The heart of the system is an inspection cell comprising two 6-axis robotic arms, capable of working independently and cooperatively. TWI established the robotic cell in its facilities in South Wales (Port Talbot). The cell was defined to be able to inspect all areas of a 3x1x1m volume. Despite the limited workspace of the cell, the robotic inspection prototype system was designed to be fully scalable and replicable in production environments. The safety enclosure of the cell surrounds a 5m x 5m space. Figure 6.1 shows the final developed IntACom robot cell.

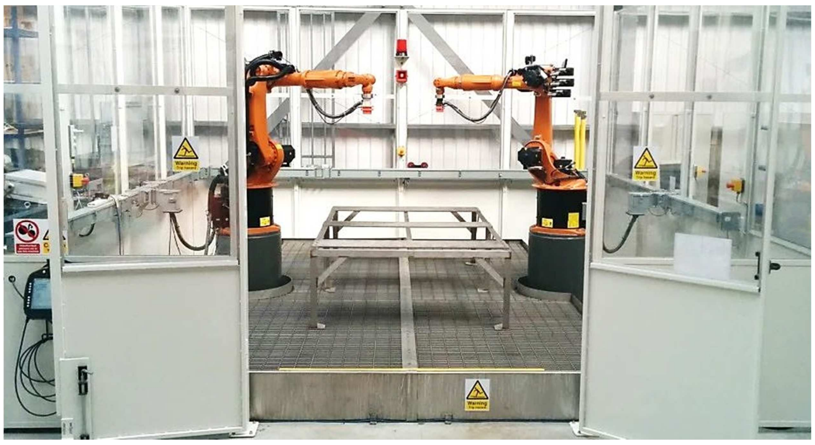


Figure 6.1 – IntACom robot cell.

The robots integrated into the robot cell are two KUKA KR16 L6-2 robot arms [1], already described in Section 3.3.2. These were selected for their maximum reach, accuracy and payload capability. Although it is intended that many inspection methods can be deployed on the robotic manipulator system, the main objective is a system able to carry out inspections using PAUT. The reason to use phased array ultrasonic transducers rather than single element probes is to take advantage of the wide area coverage available and increase the scanning speed. Both pulse-echo and through-transmission inspections have been implemented.

The robotic arms deploy end-effectors carrying ultrasonic transducers, mounted into water-jet nozzles that provide suitable water columns to guide the ultrasonic beams from the probes to the surfaces of the samples. The developed software enables data acquisition for effective robotic PAUT. Many unique features, like the real time B-scan for optimization of PAUT settings and the external control of the robotic manipulators to allow returning to points of interest, increase the usefulness of the inspection process.

The research and the investigations described in previous chapters, in conjunction with the work of TWI engineers, was very instrumental for the achievement of the interesting outcomes of IntACom. This chapter presents an overview of the development the robotic NDT inspection prototype system, which engaged the author of this thesis. Although the IntACom project integrated inputs coming from many source, an effort is made to discriminate the work developed by the author of this thesis and the work done by other TWI engineers.

The PhD research supported the IntACom project through the design of specific hardware and software solutions. As far as the hardware solutions are regarded, the author developed a magnetic tool holder for the robot end-effector (Section 6.2.1), the specification of the requirements for suitable water-jet nozzles and the design of early prototypes (Section 6.2.2), the design of water collection trays (Section 6.2.3) and of a jig table (Section 6.2.5). The core module of the IntACom software for NDT data acquisition and display was developed by TWI engineers; its architecture is described in Section 6.4 and it is included in this thesis to help contextualizing the author's software contributions. The novel external robot control, described in Section 3.5, was incorporated in the IntACom software. The author developed an add-on software module to achieve fast rescanning of areas of interest of a part under inspection, through the exploitation of the external control method (see Section 6.3 and 6.5.2). The add-on module is capable of introducing substantial advantages to speed-up the industrial inspection processes. It was a valuable and remarkable contribution of the author to the final performances of the IntACom system.

6.2. Setting up the robotic inspection prototype system

6.2.1. Magnetic tool holder

An end effector is the device at the end of a robotic arm, designed to interact with the environment. The exact nature of this device depends on the application of the robot. In the strict definition, which originates from serial robotic manipulators, *end-effector* means the last link (or end) of the robot. At this endpoint the tools or probes are attached. In a wider sense, an end effector can be seen as the part of a robot that interacts with the work environment. This does not refer to the wheels of a mobile robot or the feet of a humanoid robot which are not end effectors - they are part of the robot's mobility.

The last link of the robots used at TWI has a flange with threaded holes that provides possibility for a rigid attachment of a tool, as showed in Figure 6.2.

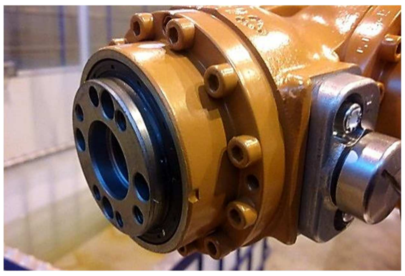


Figure 6.2 – Extremity of KUKA KR16 L6-2 robotic arm.

A rigid attachment of a tool has many disadvantages. It does not allow rapid tool change and it does not provide any robot crash protection feature. A Robotic Tool Changer is usually an end-effector with two mating parts normally called a Master Side and Tool Side that have been designed to lock or couple together automatically and are able to pass utilities such as electrical signals, pneumatic, water, etc. The Master Side of the tool changer mounts to the robot; the Tool Side mounts to the tooling, such as grippers, welders and deburring tools or to probes and sensors.

When using expensive tools or if it is necessary to protect the system from collisions with objects in the robot working envelope, it is important to have crash protection features embedded in the robot end-effector. In this case, a crash can be detected before or during a collision and a signal can be sent back to the robot controller to stop any motion and avoid further consequences. A Robotic Crash Protection Device (CPD) is a collision sensor that detects the collision during the crash. A collision sensor is also known as a robot safety joint, robot overload protection device, *QuickSTOP* (trademark of *Applied Robotics*), robot safety mount, robotic clutch, and robotic collision protector [2].

Most robot couplers use pneumatics to lock the Master and Tool side together. They use the vacuum or the Bernoulli's law[13] (for non-contact holders) to keep the tooling locked to the master part of the end-effector (Figure 6.3).

[13] In fluid dynamics, the Bernoulli's law states that for an inviscid flow of a nonconducting fluid, an increase in the speed of the fluid occurs simultaneously with a decrease in pressure or a decrease in the fluid's potential energy.

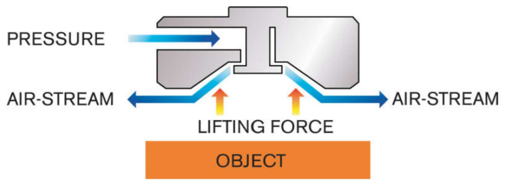


Figure 6.3 – Schematic representation of Bernoulli's law usage for a robotic end-effector.

ATI Industrial Automation has patented a pneumatic robotic collision sensor designed to prevent costly damage to robotic end-effectors resulting from robot crashes. It is shown in Figure 6.4. The Protector's features include: automatic reset, high repeatability and large moment rotation [3]. The automatic reset feature particularly benefits workplace safety. After the robot crashes, ATI's collision sensor will automatically reset when the robot removes the tooling from the crash-causing object and when the torsional rotation does not exceed 20-25 degrees. Compliant in the X, Y and Z axes, this capability provides a safer method for resetting by making it unnecessary for personnel to enter the robotic work cell.

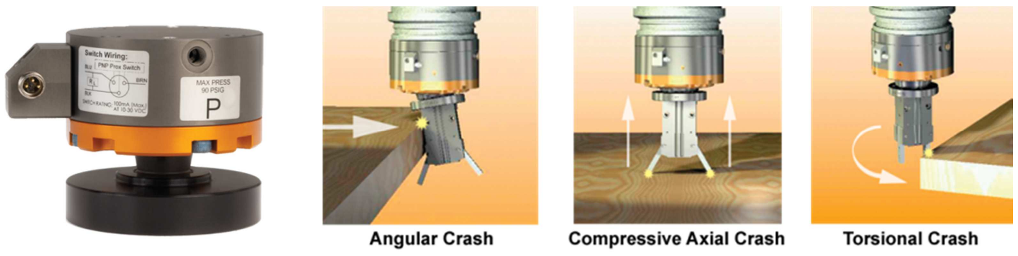


Figure 6.4 – ATI Protector end-effector.

Along with the Protector's superior angular compliance, a spring option provides three different breakaway moment settings. This feature provides additional compliance when a low moment trip point is needed. Pneumatic pressure can be added to increase moment capacity for high-inertia motions, and the pressure can be released—relying on the spring alone for low-inertia moves.

6.2.1.1. A customized solution

Despite of many pneumatic holders being commercially available and having many advantages, they need a source of compressed air or vacuum in order to work properly.

It was decided to investigate other concepts and develop a different solution. Since the final system was expected to have a water jet nozzle attached to the robot end effector, it was agreed to not further increase the amount of utilities that had to be transferred to the extremity of the robotic arm. In addition, the maximum payload of the robots in use is only 6 kg [4].

The possibility to develop a magnetic holder was investigated. A concept was developed. It comprises an austenitic stainless steel body for the Master and Slave part of the holder, a permanent magnet on the Master side and a ferromagnetic disc on the Tool side. The coupling force can be adjusted changing the thickness of the ferromagnetic disc. Three push button switches connected in series provide the safety information to the robot controller to stop the motion of the arm if the Slave side is released for any reasons.

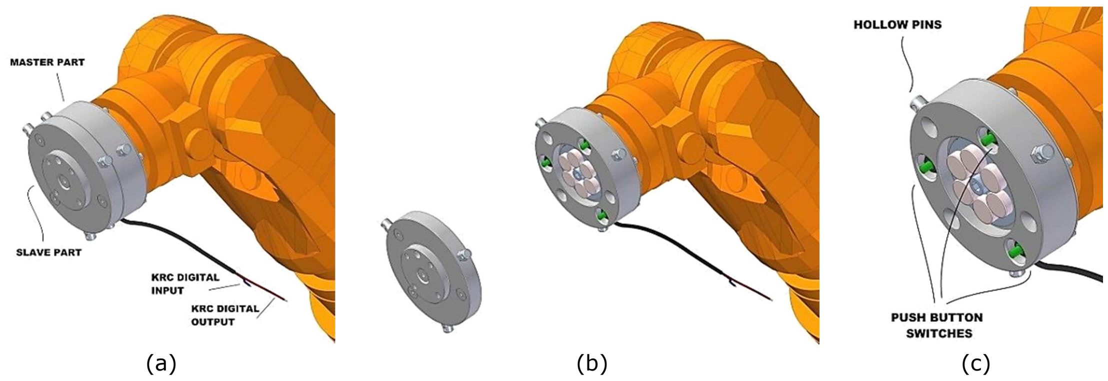


Figure 6.5 – a) Assembled view of the magnetic holder first design; b) Exploded view; c) Magnified view showing location of push button switches and hollow pins.

Figure 6.5 shows the first design of the holder. It included three radially disposed hollow pins on each side to allow the possibility to secure the tool side to the master part with long ring tapes and to avoid falling of the tooling to the ground, when the magnetic attraction becomes insufficient to keep the holder assembled. Moreover, the Master part comprised three conical holes non-axial-symmetrically disposed on its front surface, in order to host the conical protuberances of the slave part for reference purpose.

The final solution was obtained after applying some improvements to the first design. The bodies of the two parts where lightened through adding holes to their designs. Since these holes can be used to host safety strings, the radially disposed hollow pins where removed. The conical reference holes where transferred from the master part to the slave part and the conical protuberances from the slave to the master part. This last shrewdness was put in place to reduce the manufacturing cost of the slave part. Many *slave* parts may be needed in the final system, each one for a different tool or probe. Some rendering pictures are shown in Figure 6.6.

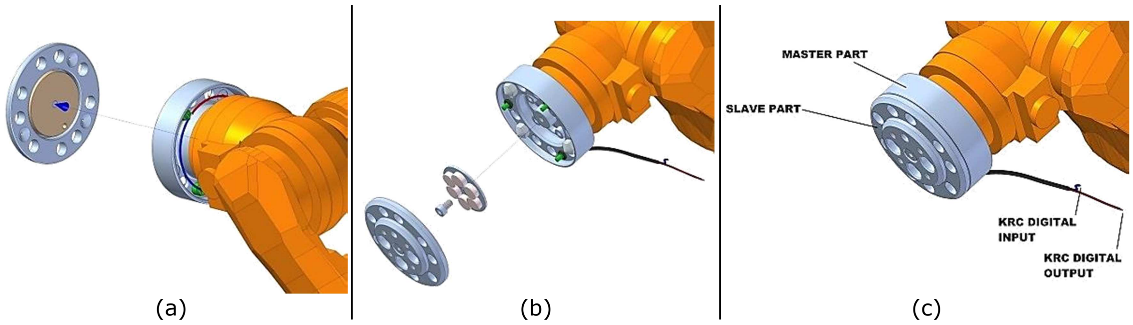
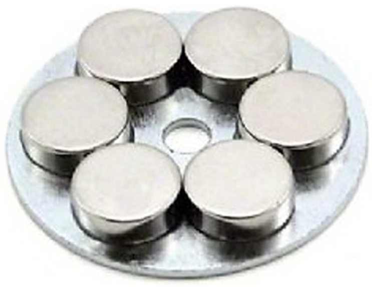


Figure 6.6 – Final design of end-effector magnetic holder. Exploded views (a and b) and assembled view (c).

The magnet used in the design is commercially available [5]. It is a 50 mm diameter x 7 mm thick N42 Neodymium medium pole limpet magnet. It comprises six small round magnets. The North Pole is facing on three magnets and the South Pole is facing on the other three magnets creating a striped magnetic field (N-S-N-S). Each magnet is plated with 3 layers of protective coatings - Nickel + Copper + Nickel (Ni-Cu-Ni). Figure 6 shows a picture of the limpet magnet plus a table reporting its features.



Shape:	Disc/Ring
Magnetic Face:	50 mm diameter
Thickness:	7 mm
Grade:	N42
Plating:	Ni-Cu-Ni (Nickel)
Material:	NdFeB
Vertical Pull (kg):	50
Slide Resistance (kg):	10
Fixing:	M6 Screw

Figure 6.7 – Picture of the chosen limpet magnet plus a table reporting its features.

The slide resistance of a magnet is always smaller than the vertical pull capability. The aforementioned magnet has a reasonable slide resistance of 10 kg and a very high vertical pull capability of 50 kg.

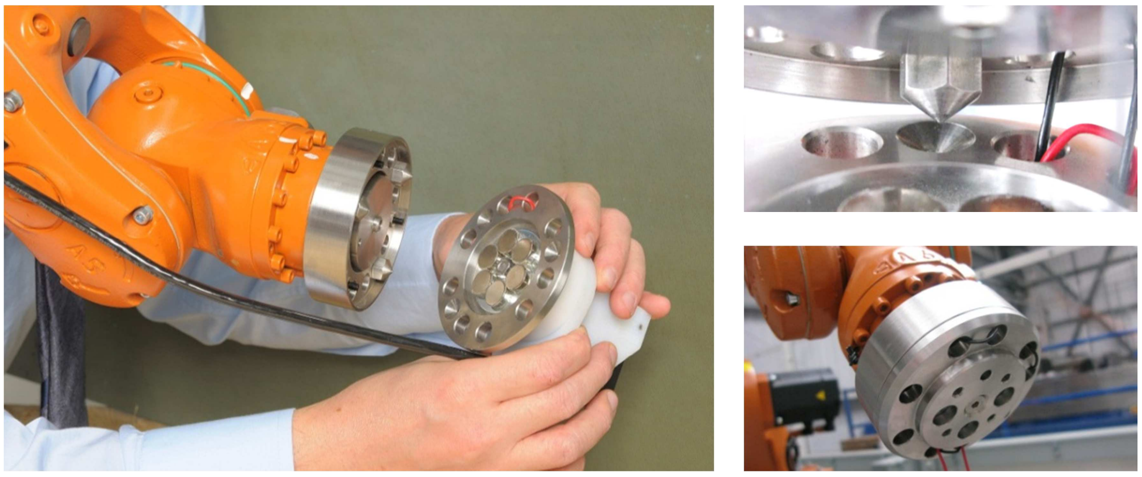


Figure 6.8 – Magnetic holder mounted to the KUKA KR16 L-6 robot extremity.

All components of the holder were manufactured and assembled. Figure 6.8 shows the magnetic holder mounted to the KUKA KR16 L-6 robot extremity, during the phase of locking the Tool part into the Master part.

6.2.1.2. KRL code to trigger the interrupt

The three push button switches of the magnetic holder of the IntACom system where designed to send a digital signal to the robot controller, to stop the motion of the arm every time the Slave side is released for any reasons.

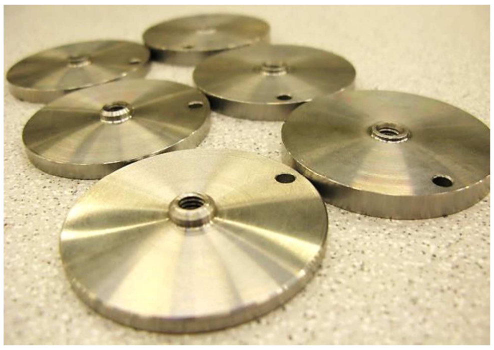
It was investigated how to set an interrupt into a KUKA Robot Language (KRL) programme. The robot controller processes the programme code line by line; the task of ensuring the magnetic holder is constantly locked during the execution of robotic movements has to be defined in a sub-routine running in background.

A method to set up a software interrupt through KRL is described in Appendix C1.

6.2.1.3. Experimental characterization

Experimental tests were carried out to characterize the response of the magnetic holder when different forces are applied.

Six ferromagnetic disks of increasing thickness were used (Figure 6.9a). The disks were manufactured in ferromagnetic stainless material 431, which is a high chromium-low nickel high hardenability Martensitic stainless steel with high strength and good corrosion resistance (suitable to be used in wet environments) [6].


(a)

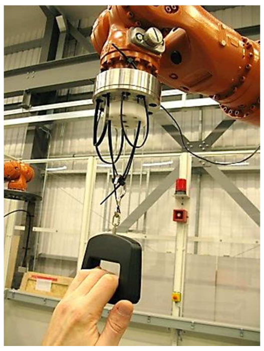
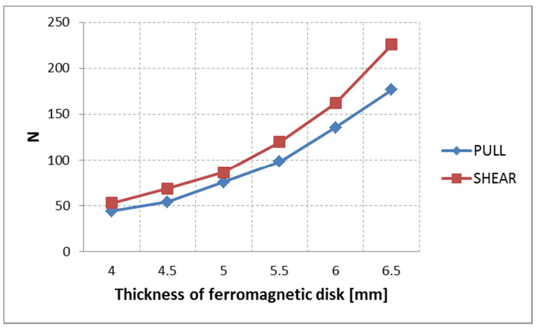

(b)

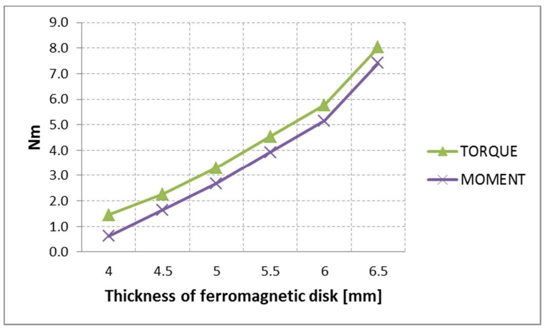
Figure 6.9 – a) Ferromagnetic disks for the magnetic holder; b) Estimation of pulling resistance.

The magnetic holder was tested to characterize its behaviour under pulling, shear, torque and moment. Figure 6.9b shows a moment during the estimation of the pulling force by using a commercially available electronic weighing machine. The results are given in

Figure 6.10. All the resistance patterns are considered satisfactory for the practical usage of the magnetic holder.



(a)



(b)

Figure 6.10 – Magnetic holder characterization: a) Pulling and shear; b) Torque and moment.

Therefore, knowing the pulling and shear forces and the torque and moment resistance, it is possible to select the appropriate magnet disk thickness accordingly to end effector load.

6.2.2. Water-jet nozzle prototypes

The automated robotic manipulator system specified by the IntACom project had to be equipped with a phased array ultrasonic inspection system. A way to couple ultrasonic probes with complex surfaces was investigated.

Simple geometries can be inspected using the full immersion ultrasound techniques; however, since the automated NDT system aims to cope with complex surfaces, the immersion techniques are not suitable.

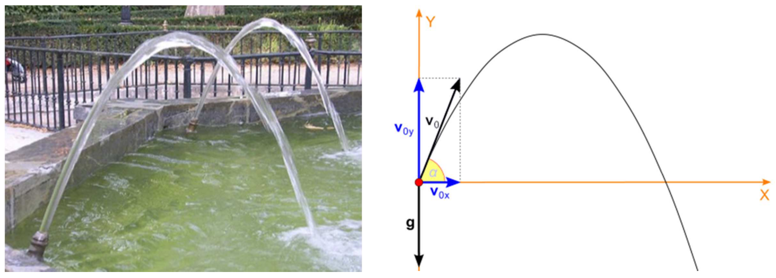
In order to accommodate small deviations between the virtual models and the real parts, it was decided that the probes should not come into contact with the component to be inspected. They are to follow a non-contact path along a path relative to the inspected component and an ultrasound coupling water column has to be provided by water-jet nozzles.

Free water jet coupling is entirely wear-less and it is especially suitable for the inspection of complex profiles. Between probe and specimen, a free water column of approximately 10-30 mm has to be generated to guide and couple ultrasound. Designing probe holders that produce a water column which is bubble-free for undisturbed guidance of the ultrasound is a key aspect for the success of this technique. Appendix C2 reports an overview of water-jet systems developed by some companies in the past few years.

6.2.2.1. Requirements and considerations

This section reports the preliminary investigations carried out to develop suitable water jet nozzles to fulfil the requirements of the IntACom project. Considering the lack of information regarding the working parameters (flow rates, pressures, water speed, etc.), of existing water-jet systems (see Appendix C2), it was necessary to investigate such parameters to get an initial grasp for the development of customized solutions for the TWI robotic inspection system.

The water irrigation system required by the IntACom project had to be flexible enough to allow pulse-echo and through-transmission ultrasound inspections executed through single-element or phased-array probes. A maximum active area of 6 x 40mm was considered for phased array probes. The water jet coupling system had to provide water columns free from air bubbles and allow standoff of 10-30mm between the probe and the sample surfaces. A free water jet follows a particular trajectory akin to that of a *projectile motion*. It is a form of motion where a particle (called *projectile*) is thrown obliquely near the earth's surface. The particle moves along a curved path under the action of gravity.



(a) (b)

Figure 6.11 – Trajectory of free water jet.

We can refer to a water particle as representative of a water flow. If the water particle is launched with an initial vector velocity $\mathbf{v}_0$ and an angle Θ, the horizontal component of the velocity remains unchanged throughout the motion. The vertical component of the velocity changes linearly, because the acceleration is constant. At any time *t*, the components of the velocity are:

$$v_x = v_0 \cos\theta \qquad \text{Eq. 6.1}$$

$$v_y = v_0 \sin\theta - gt \qquad \text{Eq. 6.2}$$

And the particle's horizontal and vertical displacements are:

$$x = v_0 t \cos\theta \qquad \text{Eq. 6.3}$$

$$y = v_0 t \sin\theta - \frac{1}{2} g t^2 \qquad \text{Eq. 6.4}$$

If we eliminate *t* between the previous two equations we will obtain the following:

$$y = x \tan\theta - \frac{g}{2\, v_0^2 \cos^2\theta} x^2 \qquad \text{Eq. 6.5}$$

From Eq. 6.1 and Eq 6.2 it is possible to express:

$$\tan\alpha = \frac{v_y}{v_x} = v_0 \tan\theta - \frac{gt}{v_0 \cos\theta} \qquad \text{Eq. 6.6}$$

and substituting *t* from Eq. 6.3:

$$\tan\alpha = \frac{v_y}{v_x} = v_0 \tan\theta - \frac{gx}{v_0^2 \cos^2\theta} \qquad \text{Eq. 6.7}$$

The previous formulas allow some preliminary calculations.

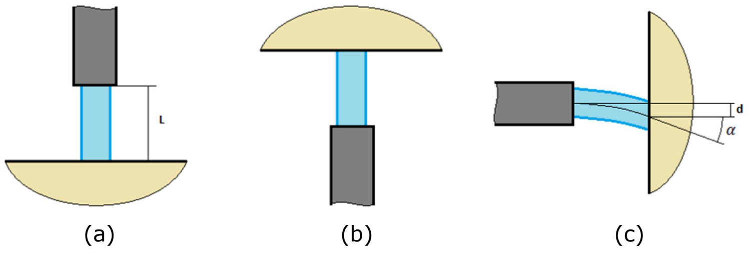


Figure 6.12 – Free water jet addressed to a surface from three different directions.

Figure 6.12 shows free water jet addressed to a surface vertically from below, from above and horizontally from one side. Let us imagine that the jet originates from a squirting pipe at L=30 mm from the surface. The easiest scenario is associated with the projection of the water jet from above the surface. In this case the jet falls perpendicularly to the surface whatever is the velocity of its water particles. If the jet is directed to the sample from below, it is necessary to guarantee $v_y(L) > 0$. It means, from Eq. 6.2 and Eq. 6.4:

$$v_0 > \sqrt{2gL} \;\rightarrow\; v_0 > 0.767\ \frac{m}{s} \qquad \text{Eq. 6.8}$$

In the third case, we may want to ensure that the deviation (d) from the horizontal direction or/and the angle (α) are smaller than certain values. If we desire to have d<1mm, it is possible to calculate from Eq. 6.5:

$$v_0 > 1.40\ \frac{m}{s} \qquad \text{Eq. 6.9}$$

Instead, if we want to guarantee α<1.0 degree, from Eq. 6.7:

$$v_0 > 4.11\ \frac{m}{s} \qquad \text{Eq. 6.10}$$

These values are in the order of magnitude of the water velocity in the water jet system of a 3-DOF inspection gantry that TWI has in its laboratory. The flow rate of the through transmission pump was measured being equal to Q=13.8 litre/min. The pump serves two nozzles for single element ultrasonic probes with 6 mm diameter aperture. The relative velocity of the outgoing water flow is equal to:

$$\mathrm{v_0} = \frac{\mathrm{Q}}{2 \cdot 60 \cdot 1000} \cdot \frac{1}{\pi \mathrm{r}^2} = 4.07\ \frac{\mathrm{m}}{\mathrm{s}} \qquad \text{Eq. 6.11}$$

So, $v_0^* = 4.00\ m/s$ is assumed to be a good design parameter for the following calculations.

In fluid mechanics, the Reynolds number (Re) is a dimensionless number that gives a measure of the ratio of inertial forces to viscous forces and consequently quantifies the relative importance of these two types of forces for given flow conditions. Reynolds numbers frequently arise when performing dimensional analysis of fluid dynamics problems, and as such can be used to determine dynamic similitude between different experimental cases. They are also used to describe different flow regimes, such as laminar or turbulent flow: laminar flow occurs at low Reynolds numbers, where viscous forces are dominant, and is characterized by smooth, constant fluid motion; turbulent flow occurs at high Reynolds numbers and is dominated by inertial forces, which tend to produce chaotic eddies, vortices and other flow instabilities.

The concept of the Reynolds number can be used to evaluate the possibility of obtaining laminar flows (free of turbulence and air bubbles) for water jets coupling phased array probes.

For water flowing in a round-section pipes, the Reynolds number is generally defined as [7]:

$$\mathrm{Re} = \frac{\rho\ \mathrm{v}\ \mathrm{D_H}}{\mu} \qquad \text{Eq. 6.12}$$

where:

- $\mathrm{D_H}$ is the hydraulic diameter of the pipe;
- v is the mean velocity of the fluid (SI unit: m/s);
- μ is the dynamic viscosity of the fluid (SI unit: Pa·s or N·s/m^2 or kg/(m·s));
- ρ is the density of the fluid (SI unit: kg/m^3).

For square or rectangular pipe sections, the characteristic dimension for internal flow situations is taken to be the hydraulic diameter, $\mathrm{D_H}$, defined as:

$$\mathrm{D_H} = \frac{4\mathrm{A}}{\mathrm{P}} \qquad \text{Eq. 6.13}$$

where A is the cross-sectional area and P is the wetted perimeter. The wetted perimeter for a channel is the total perimeter of all channel walls that are in contact with the flow.

It is common practice, according to experimental observation, to assume that a liquid flowing in a pipe is in laminar regime when $\mathrm{Re} < 2300$, transient regime when $2300 < \mathrm{Re} < 4000$ and turbulent regime when $\mathrm{Re} > 4000$. For water at T=15°C, the dynamic viscosity and the density are respectively: $\mu = 0.001139\ kg/ms$ and $\rho = 999.19\ m^3/s$.

The hydraulic diameter to cover the active area of the phased-array probe is:

$$\mathrm{D_H} = \frac{4(6 \cdot 40)}{2(6+40)} = 10.3\ \mathrm{mm} \qquad \text{Eq. 6.14}$$

Therefore, calculating the necessary flow rates to guarantee the velocity being equal to v_0^* and the relative Reynolds number, we get:

$$\mathrm{Q} = 57.6\ \mathrm{litre/min} \qquad \text{Eq. 6.15}$$

$$\mathrm{Re} = 36143. \qquad \text{Eq. 6.16}$$

The Reynolds number, greater than 4000, predicts strong turbulent flows. In order to maintain $Re < 2300$, the flow velocity and rate should be:

$$\mathrm{v}_0 < 0.254\ \mathrm{m/s} \qquad \text{Eq. 6.17}$$

$$\mathrm{Q} < 3.65\ \mathrm{litre/min.} \qquad \text{Eq. 6.18}$$

However the previous velocity would be too small to ensure good coupling conditions, according to the limits calculated in Eq. 6.8, Eq. 6.9 and Eq. 6.10.

The Reynolds number related to the nozzles of the USL system is $Re = 21422$. It is also bigger that the expected value for laminar flows. It suggests that clever nozzle designs have to be considered to ensure a reliable coupling water jet.

6.2.2.2. Water jet nozzle prototypes

3D printing has been used extensively to enable rapid prototyping of multiple versions of water jet nozzles. Figure 6.13 shows two of the water jet nozzle 3D-printed prototypes, designed to support flat and concave ultrasonic phased array probes and generate suitable coupling with high curvature surfaces, through laminar water columns able to support the ultrasonic beams.

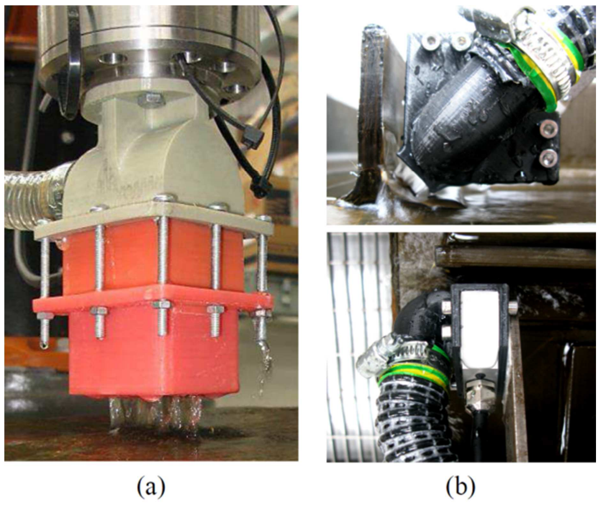


Figure 6.13 – Two of the water jet nozzle prototypes, designed to support flat (a) and concave (b) ultrasonic phased array probes.

Figure 6.14 and Figure 6.15 show some B-scans of regions with defect and without defect, collected with a 5MHz and a 2.5MHz concave probe.

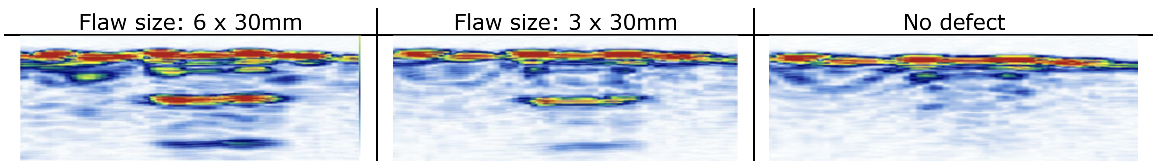


Figure 6.14 – B-scans of regions with defect and without defect, collected with a 5MHz concave probe.

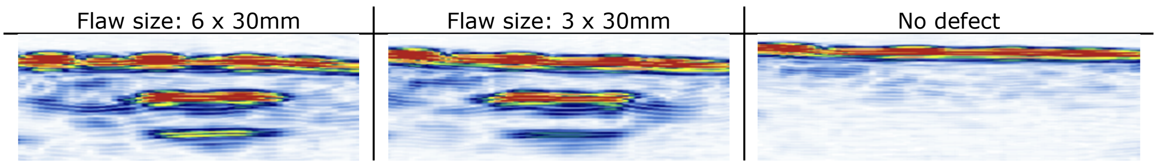


Figure 6.15 – B-scans of regions with defect and without defect, collected with a 2.5MHz concave probe.

6.2.3. Draining trays

The KR16 L6-2 robots are IP65[14] rated and all cabling of the IntACom cell was fitted within IP65 conduits to prevent water ingress. A suitable draining tray was designed to collect the water and reuse it in a closed hydraulic circuit.

[14] According to the IP Code, Ingress Protection Rating, sometimes also interpreted as International Protection Rating. It classifies and rates the degree of protection provided against the intrusion of solid objects (including body parts like hands and fingers), dust, accidental contact, and water in mechanical casings and with electrical enclosures.

The tray was designed to cover as much floor area as possible in front and around the robots, in order to collect the water squirted by the water jet nozzles. The space above the tray had to be walkable to allow the technicians to safely create the inspection setups.

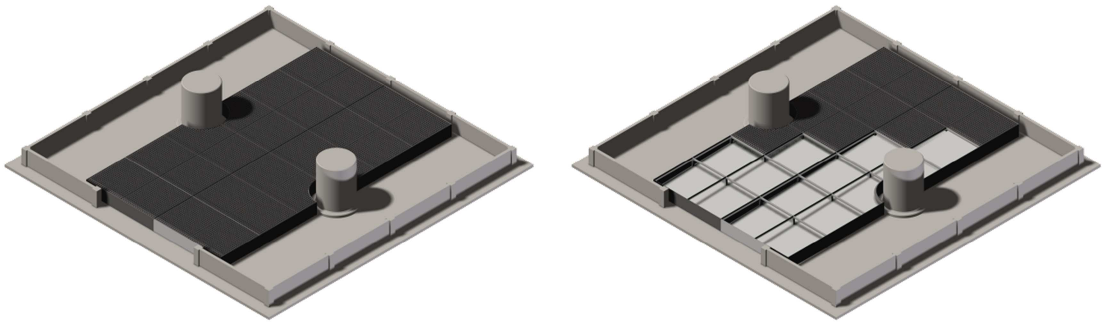


Figure 6.16 – Draining tray: concept 1.

Figure 6.16 shows design model for the draining tray. It covers the central area of the cell between the two robot plinths. The whole tray consists of two parts with sloping surfaces that direct the water to the collection points. The tray has a robust steel structure that provides backing for 24 steel gratings.

The draining trays were manufactured in Stainless Steel 316 and installed in the robot cell, as it is shown in Figure 6.17.

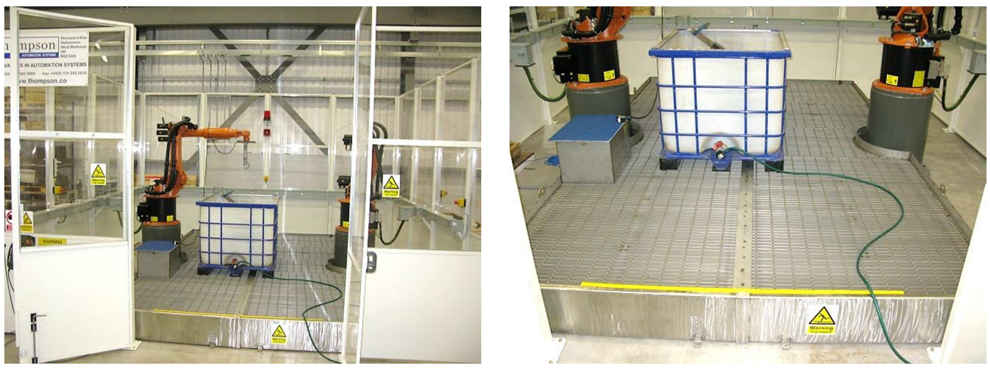


Figure 6.17 – The draining tray installed within the enclosure of the TWI robot cell.

6.2.4. Water circulation system

The water circulating system was developed according to the following scheme.

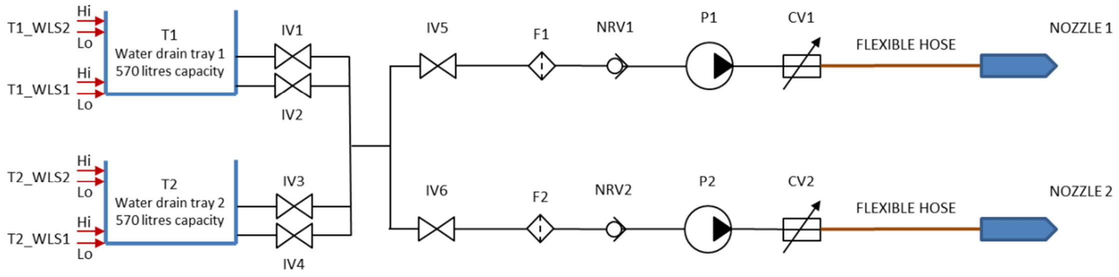
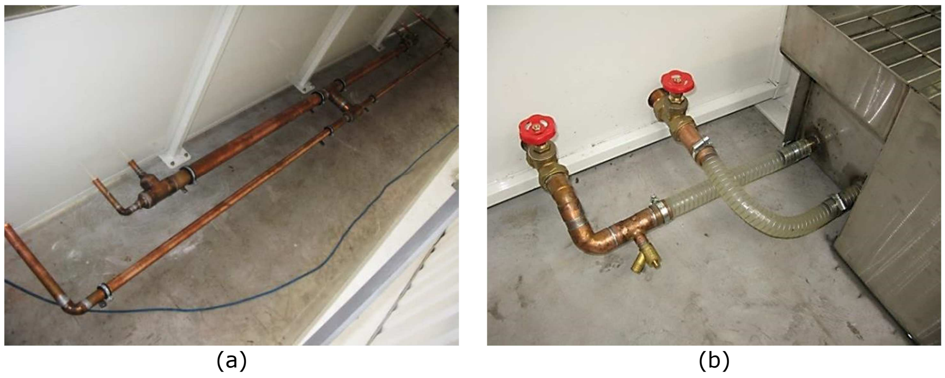


Figure 6.18 – Schematic representation of the integrated water jet system hardware.

The scheme in Figure 6.18 shows the essential components required for setting up the IntACom water circulation system. The two water collection trays are used as a large reservoir tank. They stay filled with water and linked by a 3-inch diameter copper pipe to help maintaining their level of water as equal as possible and prevent overflows when a great amount of water is squirted into just one of the two trays.

Figure 6.19a shows the copper pipework established at the back of the robotic cell. Figure 6.19b shows the two intersection valves (IV1 and IV2) mounted on the outlets of the left hand side water tray.


(a) (b)
Figure 6.19 – Pictures of the integrated pipework.

6.2.5. Jig table

The accurate calibration of the samples' position is necessary to generate accurate robotic inspection and achieve repeatable results. Manually assessment of the sample position is always possible by jogging the robot end effector to three or more points of the sample. Each of these points is reached by a rod tip attached to the robot and calibrated as robot tool central point (TCP). Therefore the Cartesian coordinates of each point are read from the robot teach pendant and used to translate and rotate the virtual model of the sample to the optimum position for path-planning.

This approach works fairly well but it is time consuming. In order to ease this procedure and remove the need to recalibrate the sample position innumerable times, a jig table was designed and manufactured. The table was mounted on the water collection trays. It allows the robot programmer to place any sample in repeatable positions within the robots' working envelope and calibrate its position less often.

The final concept is given in Figure 6.20. It consists in a 1.5m x 2.5m x 0.6m frame with three 400mm wide rectangular plates. Both the table frame and plates were made of stainless steel to avoid corrosion caused the wet environment.

The plates are mounted on the frame with bolts screwed into the threaded holes present along the long sides of the table. The position of the plates can be adjusted and changed according to the needs. The plates themselves have three rows of 6mm threaded holes running along the longitudinal direction of the plates; the holes are used to hold the samples in position.

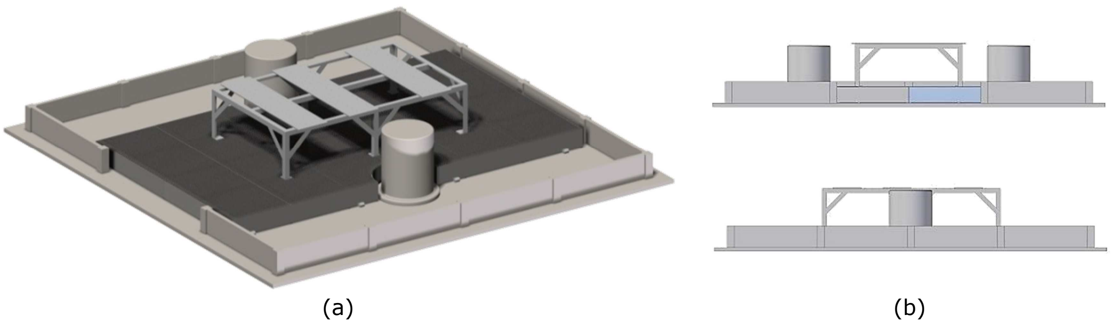


Figure 6.20 – Final concept of the jig table. Isometric view (a) and lateral views (b).

The IntACom robot cell, complete with draining trays and calibration table is shown at the beginning of this chapter (Figure 6.1).

6.3. The inspection process

The overall inspection process designed for the IntACom prototype inspection system is schematically shown in Figure 6.21. The process has hybrid features, in between the basic system described in Figure 4.1 and the completely seamless solution (Figure 5.1) achievable by using RoboNDT.

Since the original requirements for the development of the IntACom system were defined before the foundations of the RoboNDT software had been laid down, IntACom was intended to be based on the inspection procedure represented by the schema in Figure 4.1. This schema allows deployment of commercial OLP software in the context of robotic NDT, although it implies the limitations explained in Section 4.2.

The novel approach, investigated by the author of this thesis, to deploy external or robot manipulators, led to a review of the original concept. The external control capability was incorporated in the IntACom main application to enable the possibility to achieve fast rescanning of areas of interest after an initial screening of the parts under inspection.

The workflow for the inspection of a part is organized around the acquisition module and the graphical user interface (the main application) of the IntACom software. It is the core of the system, described in the following section. Off-line path-planning is deployed to generate and simulate the inspection tool-paths. It is achieved through using a commercial solution, described in Section 6.5.1.

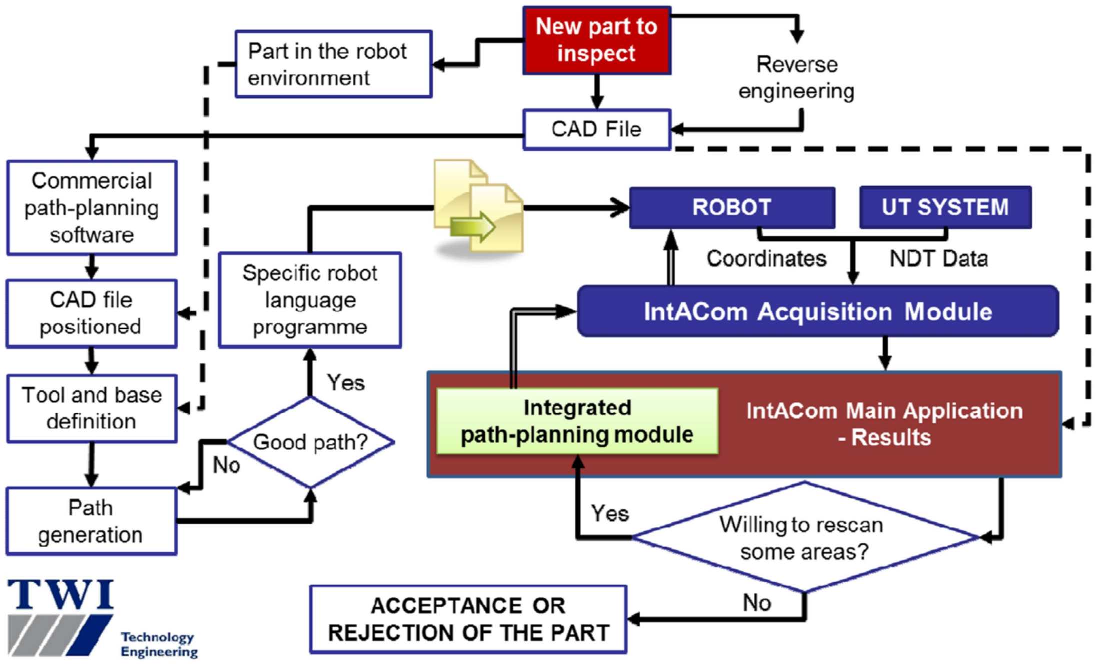

New part to inspect
Part in the robot environment
Reverse engineering
CAD File
Commercial path-planning software
CAD file positioned
Tool and base definition
Path generation
Specific robot language programme
Good path?
Yes
No
ROBOT
UT SYSTEM
Coordinates
NDT Data
IntACom Acquisition Module
Integrated path-planning module
IntACom Main Application - Results
Yes
Willing to rescan some areas?
No
ACCEPTANCE OR REJECTION OF THE PART
TWI
Technology Engineering

Figure 6.21 – Schematic representation of the robotic inspection procedure.

The probe is manipulated by the robot arm according to the predefined tool-path. At the same time, the robot positional feedback and the ultrasound phased-array data are collected by the acquisition module. Once the data is collected, it is sent to the GUI for post-processing and analysis.

The user can decide to rescan any area of interest of the parts under inspection without need to return to the commercial OLP software, thanks to the integrated path-planning capabilities described in Section 6.5.2.

6.4. The data acquisition software

The robot manipulators and the Micropulse 5PA provided a strong potential to achieve a great deal of flexibility for fast and effective NDT inspections of large curved samples. However the ultrasonic phased-array Micropulse system comes with basic focal law calculation and imaging software and a fully functioning phased array imaging and analysis system was needed.

Moreover the software developed under the IntACom project has the fundamental function of integrating the two systems, through encoding the ultrasound data coming from the Micropulse with the positional information coming from the robot controller.

One of the main objectives of the IntACom project was to develop a fully integrated data acquisition software solution, allowing NDT experts to easily collect data through the robotic system and analyse it with new types of data imagining for curved surfaces and complex geometries. The software development exploited the research described in Section 3.8 and 3.9, concerning data acquisition and encoding algorithms.

6.4.1. Selecting the most suitable programming language

The data acquisition and encoding algorithms are the core of the IntACom software; they were initially programmed in C# language.

Although the C# application deploying the second encoding algorithm proved to work well as a stand-alone application, some issues were found when it was tried to integrate it with a more complex GUI able to give real-time feedback. It was seen that even a small amount of graphical activity for real-time visualization can have a negative impact on the core data acquisition process. The machine processor can suddenly become too busy because of the plotting related operations, resulting in unwanted pauses during data acquisition.

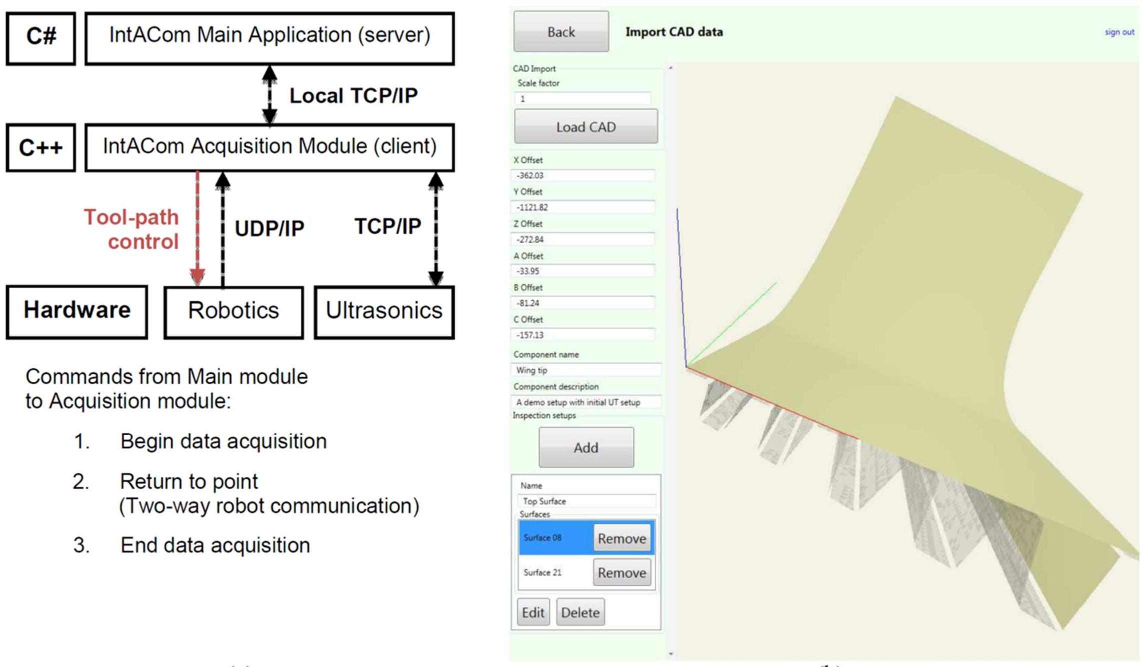
A pausing data acquisition code produces a lack of data that is eventually found as holes in the C-Scan maps. Many attempts have been made to isolate the GUI computational load from the acquisition process, e.g. running the GUI on a different machine and sending data via an Ethernet communication, assigning higher CPU priority to the acquisition process and asking to the graphic card to manage the GUI. None of these attempts sorted out the issue completely.

The cause of the issue was eventually identified in the not perfect *garbage collection* managed by C# scripts. In computer science, garbage collection (GC) is a form of automatic memory management. The *garbage collector*, or just *collector*, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. Unlike C#, C++ is suitable to develop real-time data acquisition algorithms that run in a reliable manner. The programmer can avoid the periodic, automated creation and disruption of allocated memory, which is known as garbage collection [8, 9].

Therefore the C++ was chosen as programming language to develop the critically important real-time modules of the IntACom software. The C# code was translated into a C++ code. This led to a very reliable acquisition data module enhanced by a flexible GUI capable of showing real-time feedback data.

6.4.2. Software architecture

Figure 6.22a shows the fundamental structure of the IntACom software. It has been carefully designed to enable fully integrated communication with the robots and phased array controllers. The main application, developed in the C# programming language, controls the GUI and behaves as a server application. The C++ language controls the acquisition module.



Commands from Main module to Acquisition module:

1. Begin data acquisition
2. Return to point (Two-way robot communication)
3. End data acquisition

CAD Import
Scale factor: 1
X Offset: -362.03
Y Offset: -1121.82
Z Offset: -272.84
A Offset: -33.95
B Offset: -81.24
C Offset: -157.13
Component name: Wing tip
Component description: A demo setup with initial UT setup
Inspection setups
Name: Top Surface
Surfaces: Surface 08, Surface 21

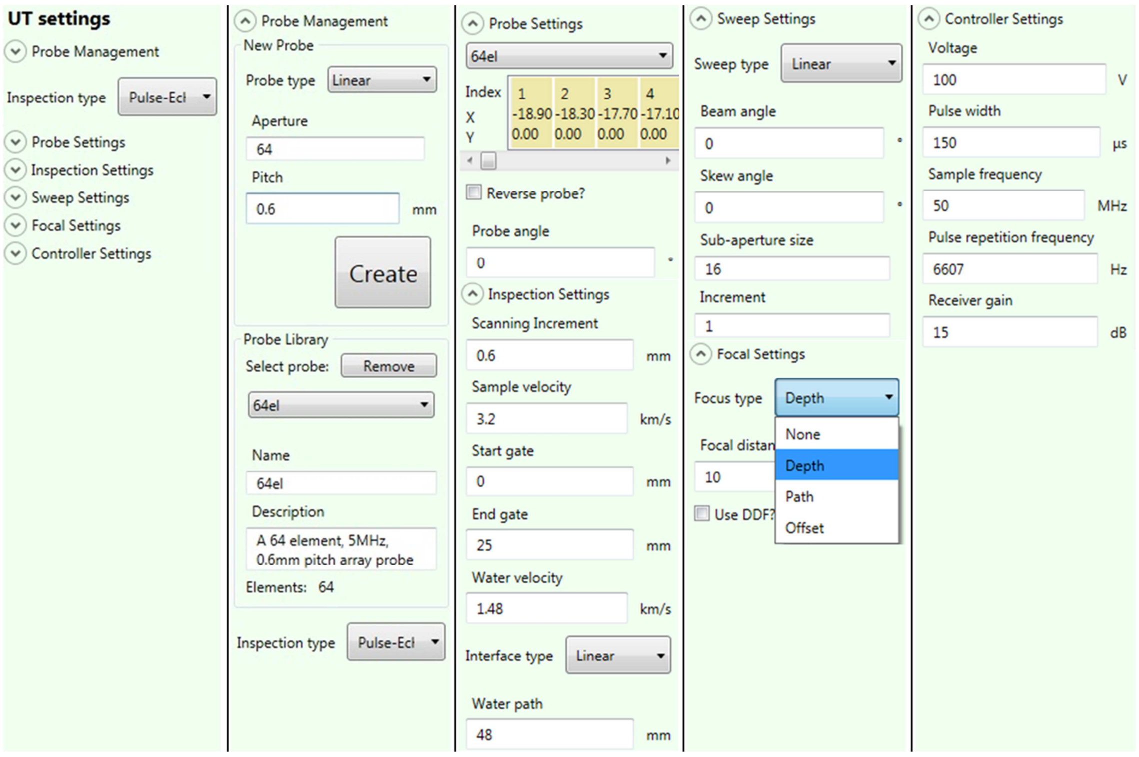
(a) (b)

Figure 6.22 – Software structure (a) and GUI with a newly imported sample model (b).

The main application receives data from the acquisition module through a local TCP/IP connection. The acquisition module connects to the robot controller through a one-way or two-way UDP/IP Ethernet connection and to the Micropulse with a TCP/IP connection.

Figure 6.22b shows the GUI during the definition of a new part. The software can import Standard Tessellation Language (STL) CAD files [10]. The STL format was chosen because it is supported by the majority of the existing software packages; it is widely used for rapid prototyping and computer-aided manufacturing. The format only describes the surface geometry of a three-dimensional object without any representation of colour, texture or other common CAD model attributes. The STL file format can specify a CAD part in either ASCII or binary. The software analyses the nodes of the STL mesh while importing the CAD model; all the disjointed surfaces of the sample are automatically recognised through counting the number of triangles that share each of the nodes in the mesh.

Every sample has multiple surfaces and each one of them requires specific inspection settings. Therefore the operator is able to select the surfaces of interest, name them with tags and associate the desired Ultrasonic Testing (UT) settings. The complete definition of the UT settings involves the specification of a well-organized list of options, ordered in six groups (Figure 6.23): probe management, probe settings, inspection settings, sweep settings, focal settings and controller settings. The probe management group allows the definition of linear or curved phased array probes with the specification of number of active elements and pitch.


UT settings
Probe Management
Inspection type: Pulse-Ech
Probe Settings
Inspection Settings
Sweep Settings
Focal Settings
Controller Settings

Probe Management
New Probe
Probe type: Linear
Aperture: 64
Pitch: 0.6 mm
Create
Probe Library
Select probe: Remove
64el
Name: 64el
Description: A 64 element, 5MHz, 0.6mm pitch array probe
Elements: 64
Inspection type: Pulse-Ech

Probe Settings
64el

Index	1	2	3	4
X	-18.90	-18.30	-17.70	-17.10
Y	0.00	0.00	0.00	0.00

Reverse probe?
Probe angle: 0 °
Inspection Settings
Scanning Increment: 0.6 mm
Sample velocity: 3.2 km/s
Start gate: 0 mm
End gate: 25 mm
Water velocity: 1.48 km/s
Interface type: Linear
Water path: 48 mm

Sweep Settings
Sweep type: Linear
Beam angle: 0 °
Skew angle: 0 °
Sub-aperture size: 16
Increment: 1
Focal Settings
Focus type: Depth (None / Depth / Path / Offset)
Focal distan: 10
Use DDF?

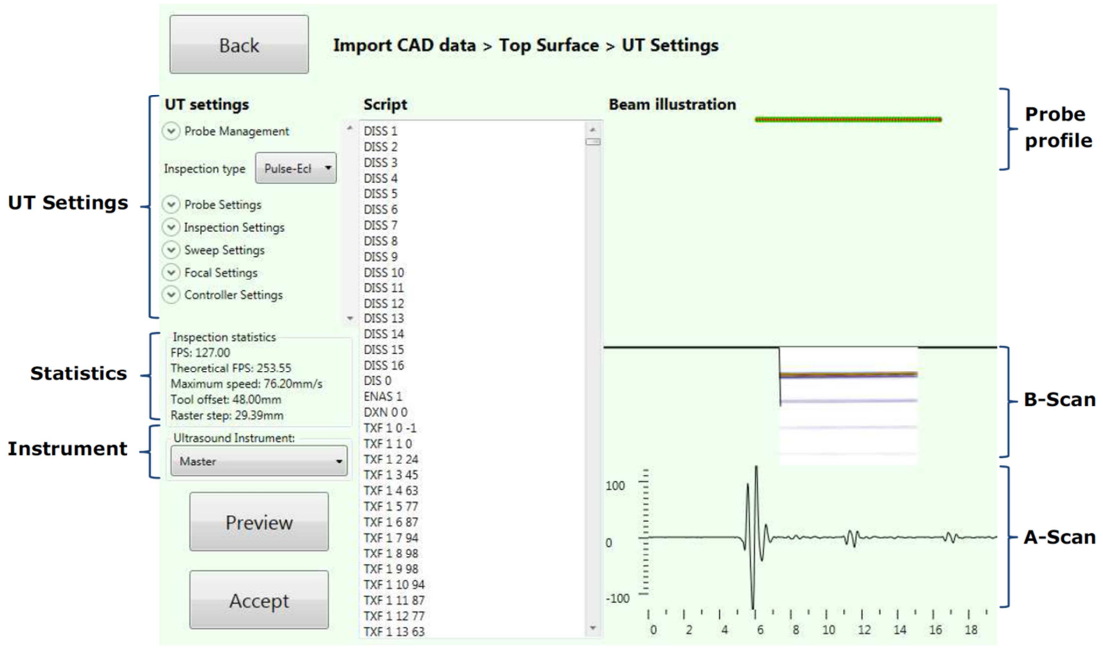
Controller Settings
Voltage: 100 V
Pulse width: 150 µs
Sample frequency: 50 MHz
Pulse repetition frequency: 6607 Hz
Receiver gain: 15 dB

Figure 6.23 – List of options for the complete definition of the UT settings.

The probe settings group gives a graphic representation of the probe elements and of their Cartesian coordinates, relatively to the centre of the array. The inspection settings group allows parameters such as the water path and the speed of sound to be set. The sweep settings, focal settings and the controller settings are used to define the focal laws.

An ASCII based communication protocol over Ethernet is used to command the Micropulse. The software automatically generates the ASCII command script to be sent to the Micropulse. It is a set of instructions for the phased array controller to generate the necessary focal laws, fire the PA probe with the defined voltage and pulse repetition rate and acquire A-scans according to the correct time window, sampling rate and gain.

The operator does not need to use extra portable screen-equipped PA instruments to check if the inspection setup is correct. The IntACom software can send the command script to the Micropulse and trigger the start of the data acquisition. This allows the operator to get a preview of the probe profile, the real-time B-scan and the A-scans. This is very useful for optimizing the position of the probe above the sample surface. Figure 6.24 shows a screenshot of the preview data, obtained after setting the parameters to the values given in Figure 6.23.



Inspection type: Pulse-Ecl

Inspection statistics
FPS: 127.00
Theoretical FPS: 253.55
Maximum speed: 76.20mm/s
Tool offset: 48.00mm
Raster step: 29.39mm

Ultrasound Instrument: Master

```
DISS 1
DISS 2
DISS 3
DISS 4
DISS 5
DISS 6
DISS 7
DISS 8
DISS 9
DISS 10
DISS 11
DISS 12
DISS 13
DISS 14
DISS 15
DISS 16
DIS 0
ENAS 1
DXN 0 0
TXF 1 0 -1
TXF 1 1 0
TXF 1 2 24
TXF 1 3 45
TXF 1 4 63
TXF 1 5 77
TXF 1 6 87
TXF 1 7 94
TXF 1 8 98
TXF 1 9 98
TXF 1 10 94
TXF 1 11 87
TXF 1 12 77
TXF 1 13 63
```

Figure 6.24 – Preview window with the probe profile, real-time B-scan and A-scan.

The activation of the data acquisition module, to get preview information through the UT instrument, also gives the possibility to obtain statistics about the acquisition performance. The acquisition speed, in terms of number of B-scans per second or frames per second (FPS), depends on several parameters: the number of elements in the probe, the sub-aperture width, the step (in a linear sweep), the length of the water path (tool offset) and the length of the A-scans (related to the thickness of the specimen). The theoretical FPS is calculated just considering physics, computing the time it takes for the ultrasound to travel to the back wall surface of the specimen and come back to the probe and multiplying it by the number of A-scans in the B-scan. The real FPS is obtained by monitoring the actual quantity of data the UT instrument is able to stream to the acquisition software. The real FPS is always smaller than the theoretical FPS, because of the delays caused by the necessary commands that the software needs to send to the Micropulse, in order to trigger the acquisition of each B-scan, and the time taken by the Micropulse to process and return the UT data. The statistics include the maximum robot speed; it is calculated according to the FPS and the desired scanning increment (the resolution in the passive direction of the probe), defined in the inspection settings.

The statistics given in Figure 6.24 show a typical inspection using a 64 element phased array transducer. The tool offset, the probe coverage and the probe speed are important input parameters for the generation of the robot tool-path, described in the following section.

6.5. IntACom path-planning solutions

6.5.1. Generation of the main inspection tool-paths

Off-line path-planning was achieved within the IntACom project using commercial robotic simulation and programming software. The chosen software was CENIT-FastSurf [11], based on a Delmia platform [12].

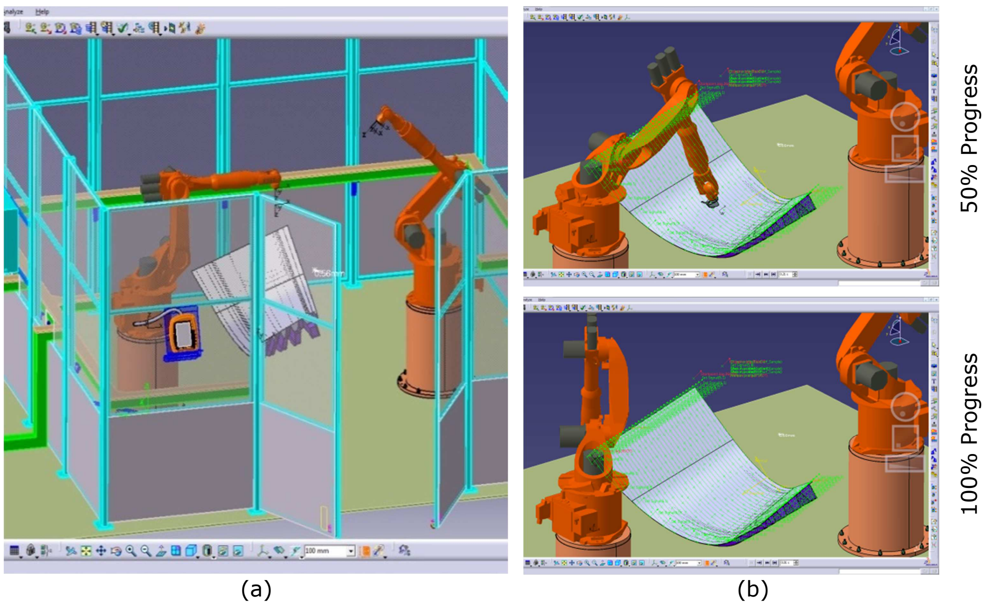


Figure 6.25 – TWI cell as it is mapped out in the FastSurf environment (a); path simulation (b).

FastSurf provides the ability to “build” a robot cell, complete with components and manipulators, in a three dimensional virtual environment. Many simulation models of the robots currently in use are available and each model contains kinematics data.

Added functionality in the simulation and OLP software allows for 3D CAD data from the component to be imported, resulting in automatic generation of scan paths on identified surfaces on the 3D CAD representation of the component. Figure 6.25 shows the TWI robot cell as it is mapped out in the FastSurf virtual environment and two moments during the simulation of the tool-path for the inspection of the main skin of a carbon fibre composite material aerospace winglet.

6.5.2. Built-in capability to rescan areas of interest

It is usual for NDT operators to double check some suspect areas of a part, after an initial inspection. For such situations, generating specific tool-paths for all the areas of interest through commercial path-planning software would be time consuming and not very practical.

A MATLAB based path-planning module was purposely developed and integrated into the IntACom software. The path-planning software add-on is able to use the original tool centre point (TCP) data, received from the robot during the initial scan, in order to generate a specific tool-path for returning to the point of interest and executing what is called a "sub-scan". The original robot trajectory is interpolated to generate the desired type of sub-scan tool-path: raster, segment or single point.

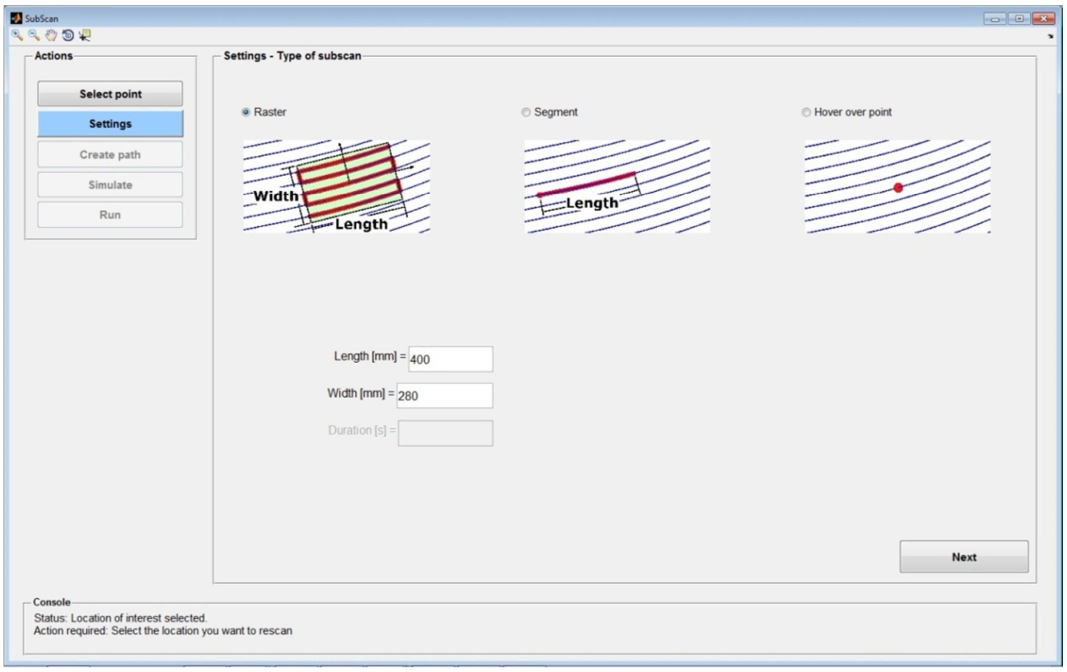
Since the *sub-scan tool-paths* are generated from the original coordinates of the Tool Central Point (TCP) of completed scans, the sub-scans are safe to execute (no crashes, singularities, etc.).

A completely automated approach to execute the sub-scan tool-paths was implemented through exploiting the external control approach, described in Section 3.7. When a specific area of interest is selected directly from the NDT data superimposed to the virtual model of the part, the integrated path-planning module is launched. The generated sub-scan tool-path is sent to the robot via the Ethernet communication and executed. Robot positional feedback and NDT data are collected similarly to the collection of the data from the main scan. Details about this implementation are given in Appendix C3.

The remainder part of this section describes the graphical user interface of the developed add-on application (SubScan.exe).

The GUI layout is organized in four areas: at the top of the window is a tool bar containing 5 graphic tools (zoom in, zoom out, pan, rotate and select point). The "Actions" panel contains the push buttons to proceed step by step from the TCP coordinates of the initial tool path to the execution of the new sub-scan. The largest part of the window is dedicated to the active context (point selection, application of settings, path creation or simulation). The bottom portion of the GUI is occupied by a console, where the user can read about the application status, some tips or errors.

Three types of sub-scan tool-path types were considered relevant and implemented: raster scan, segment scan and the possibility to halt over the point of interest. The latest is not a proper scan tool-path; however it can allow the user to drive the robot manipulator to a point of interest and check the ultrasonic signal in that particular location. Figure 6.26 shows the window for the choice of the scan type.



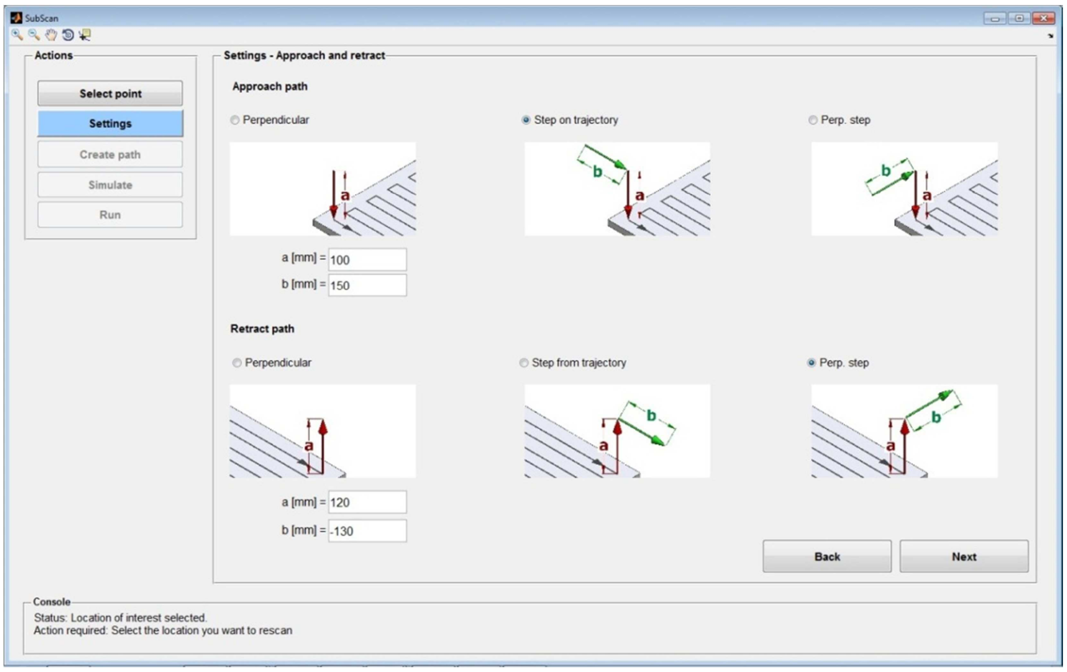
Length [mm] = 400
Width [mm] = 280
Duration [s] =

Status: Location of interest selected.
Action required: Select the location you want to rescan

Figure 6.26 – Selection of scan type.

For each type the relative text boxes for the characteristic parameters are enabled. The raster type requires length and width of the area of interest (in millimetres). A rectangular area, centred on the selected point, with the specified length and width will be considered. The length is relative to the direction of the original scan lines, the width is respected in the direction of the steps. The segment type requires the length of the segment, positive or negative depending on the preferred direction of travel (original or opposed to the original). The halt over the point requires the duration, in seconds.

The next setting menu (see Figure 6.27) regards the specification of the approaching and retracting motion. Three types of motions have been implemented and the user can select any of them for the approaching and the retracting motion. The perpendicular motion is a displacement vector perpendicular to the surface in the start point of the scan. The user is required to prompt the length of the vector in millimetre. In order to avoid collisions with the surface of the part, the right orientation of the vector is programmatically selected to make the robot approach the surface from the same side of the original tool path. The "step on trajectory" consists in a vector parallel to the scan trajectory followed by a vector perpendicular to the surface in the approaching version of the path. In the retracting motion vector parallel to the scan trajectory is preceded by the vector perpendicular to the surface. The user has to provide the length of both vectors. The "perpendicular step" motion is similar to the previous one, with the exception that the vector that precedes or follows the vector perpendicular to the surface in the approaching or in the retracting motion is perpendicular to the scan trajectory. The user can play with the sign of the vectors to flip their directions.



Settings - Approach and retract

Approach path

Perpendicular [✓ Step on trajectory] Perp. step

a [mm] = 100
b [mm] = 150

Retract path

Perpendicular Step from trajectory [✓ Perp. step]

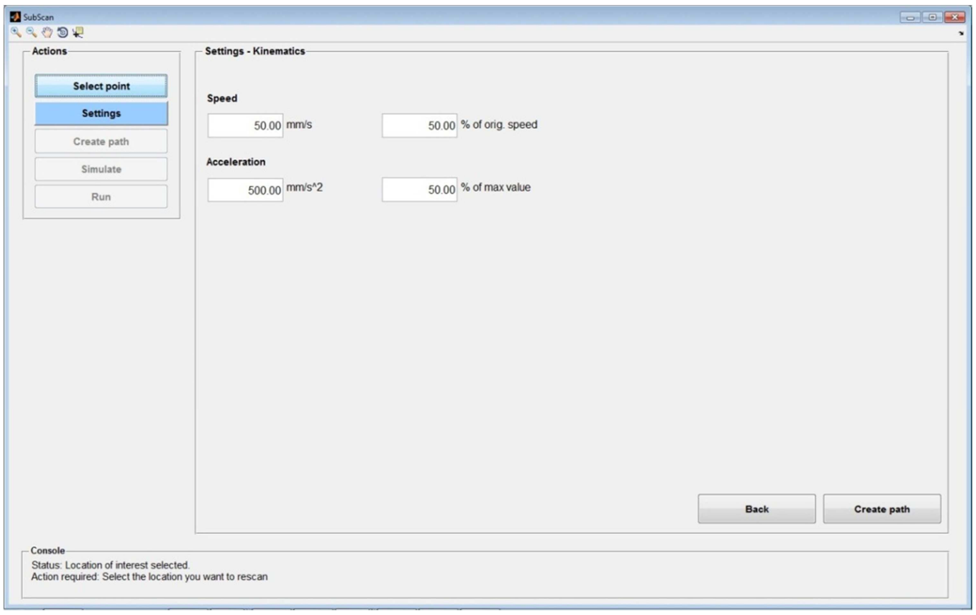
a [mm] = 120
b [mm] = -130

Console
Status: Location of interest selected.
Action required: Select the location you want to rescan

Figure 6.27 – Selection of approaching and retracting motion types.

The last settings menu regards the kinematic properties of the tool path. The user can choose a speed for the new sub-scan tool path. The software is able to estimate the speed of the original path from the feedback coordinates received from the robot controller. For safety reasons the value of the speed is limited to 100% of the original speed. This is not a big limitation, since the robot motion should be slower than the original to get a better resolution of the area subjected to rescanning. Anyway, the possibility to select a speed value up to 200% or 300% of the original one can be easily enabled in the software if it is considered useful in the future.

As far as the acceleration is regarded, this value is very important at each sudden change in the direction of travel during the path. Rapid changes in the direction can cause high inertial forces; they can damage the probes (because of the induced vibrations) and are not healthy for the longevity of the robot joints. Many models of KUKA robots allow a maximum acceleration of 10 m/s^2, but this acceleration magnitude is too high for the manipulation of NDT tooling and probing. Therefore the maximum value is limited by software to 1 m/s^2 (1000 mm/s^2); the user can choose any value of acceleration between 1 and 1000 mm/s^2.



Settings - Kinematics

Speed

50.00 mm/s 50.00 % of orig. speed

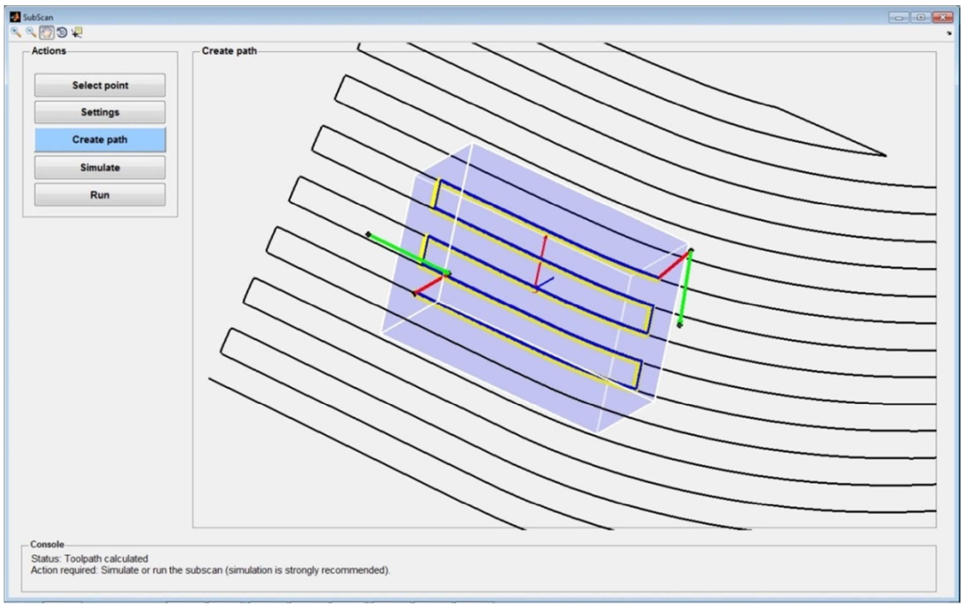
Acceleration

500.00 mm/s^2 50.00 % of max value

Console
Status: Location of interest selected.
Action required: Select the location you want to rescan

Figure 6.28 – Kinematics properties.

Once all settings are specified, the application proceeds with the creation of the path and with the show of the path preview. Figure 6.29 shows a screenshot of the path preview obtained applying the settings given in the previous figures. It is a raster scan of a 400mm wide and 280mm high rectangular region, with a "step on trajectory" approaching path and a "perpendicular step" as retracting path.



Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

Figure 6.29 – Tool path preview.

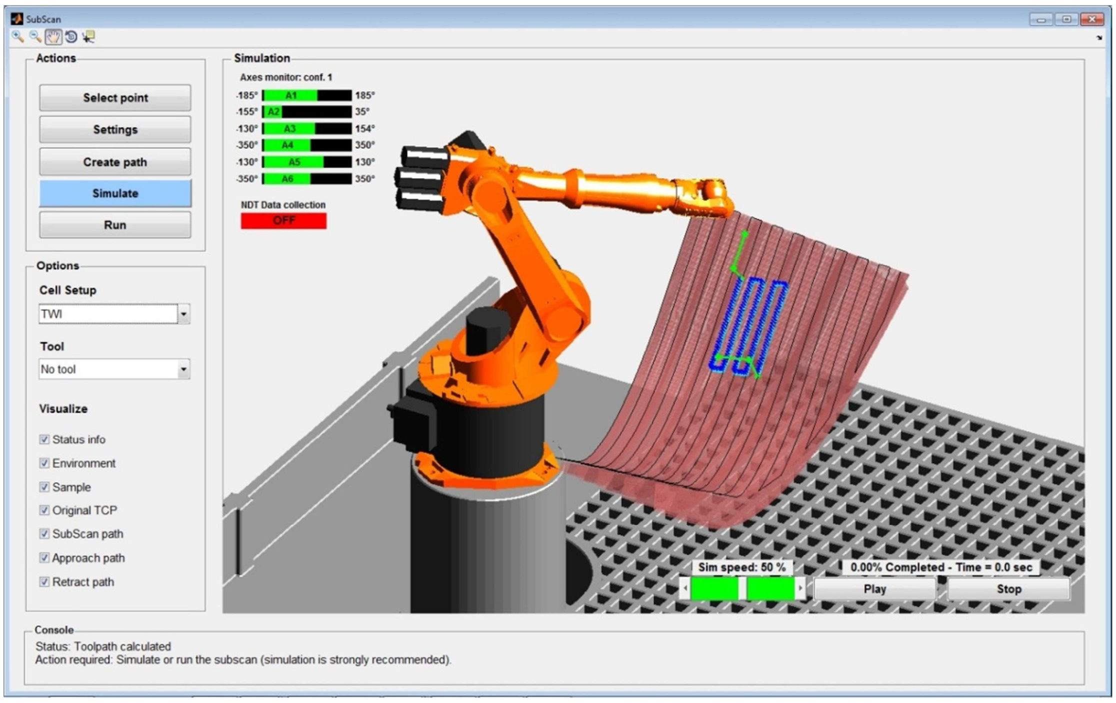
The last phase before the execution of the sub-scan regards the simulation of the generated tool path. The simulation module is based on the kinematics formulation described in 4.8.2. It demonstrates the possibility to implement simulation capabilities within executable applications compiled through MATLAB; some of the future work foreseen for further development of the RoboNDT software (see Section 5.3).

The result is an efficient simulation interface that enables the user to select a specific robot environment and mount a specific tool (probe plus casing) to the robot model. The interface shows the kinematic status of each robot joint during the simulation of the tool-path. The user can decide to display or hidden each one of the elements presents in the simulation window (robot status details, environment, sample, original TCP, tool-path trajectory, approaching and retracting paths).

The robot status details show also the status of a flag telling the user where the NDT data will or will not be collected. The value of this flag is transferred to the IntACom software to consequently trigger the code responsible for the collection of NDT data.

The CAD model of the sample showed in the simulation environment is loaded from an XML file generated by the IntACom software each time a new inspection is defined. A specific function was developed to parse the XML file and load the sample geometry within the simulation environment of the path-planning add-on application. This ensures consistency between the IntACom main application and its built-in path-planning module. Moreover, the correct sample is always loaded within the simulation environment and placed in the right position relatively to the robot manipulator.

The objective was to free the user as much as possible from the redefinition of anything that can be affected by errors. For this reason the software uses the *base* and *tool* parameters received from the robot controller during the execution of the initial tool path and redefinition of robot base and tool parameters is not enabled. Figure 6.30 shows the simulation of a raster sub-scan, before execution. The user can choose the right cell setup and robot tool (probe) from two pop-up menus present in the option panel of the simulation module. If it is necessary to define a new cell setup (e.g. for a new industrial partner), this can be done creating a new folder in a specific directory of the software (..\SubScanData\Cell_Setups) named with the desired name for the new setup and containing the CAD files of the cell environment and of the robot joints. The name of the cell setup will appear as a new option in the pop-up menu. Similarly, a new tool can be defined storing its CAD file in the relative directory (..\SubScanData\Tools).



Cell Setup: TWI
Tool: No tool
☑ Status info
☑ Environment
☑ Sample
☑ Original TCP
☑ SubScan path
☑ Approach path
☑ Retract path

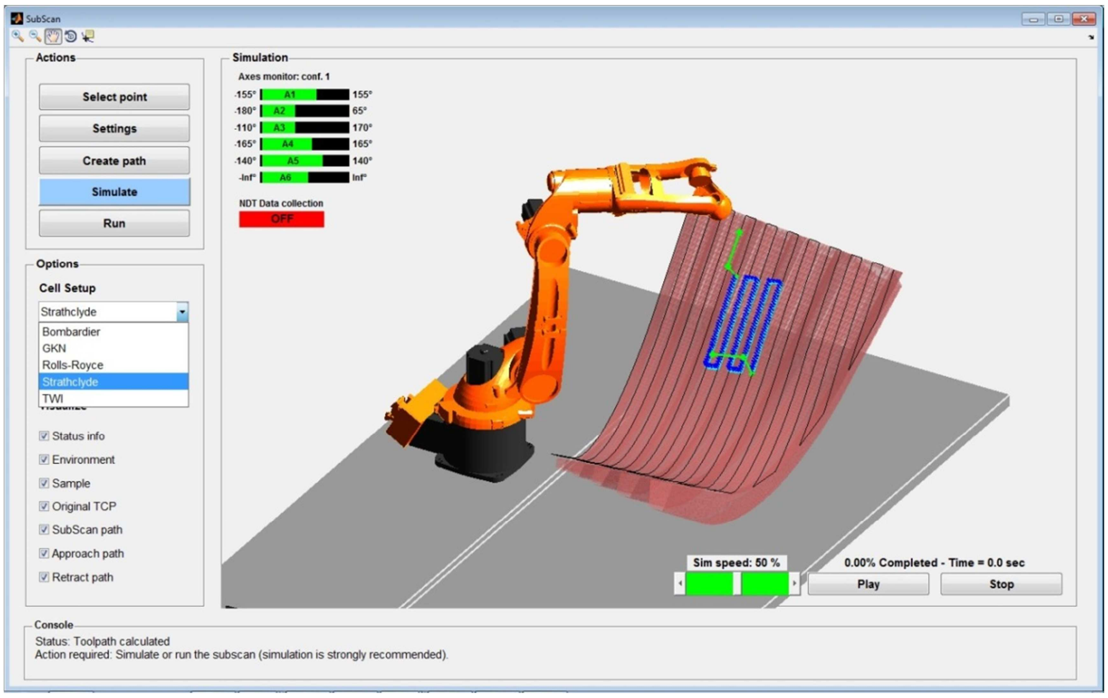
Axes monitor: conf. 1

-185°	A1	185°
-155°	A2	35°
-130°	A3	154°
-350°	A4	350°
-130°	A5	130°
-350°	A6	350°

Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

Figure 6.30 – Simulation environment with the TWI cell setup.

Figure 6.31 shows the same simulation environment given by Figure 6.30, where the TWI cell setup was replaced by robot setup available at the University of Strathclyde. The TWI setup comprising the KUKA KR16 L6-2 placed on a plinth is replaced by the KR5arcHW robot arm placed on a floor platform. The values of the maximum and minimum angle for each joint of the robot are automatically updated as soon as the robot cell is changed.



Cell Setup: Strathclyde
Bombardier
GKN
Rolls-Royce
Strathclyde
TWI
☑ Status info
☑ Environment
☑ Sample
☑ Original TCP
☑ SubScan path
☑ Approach path
☑ Retract path

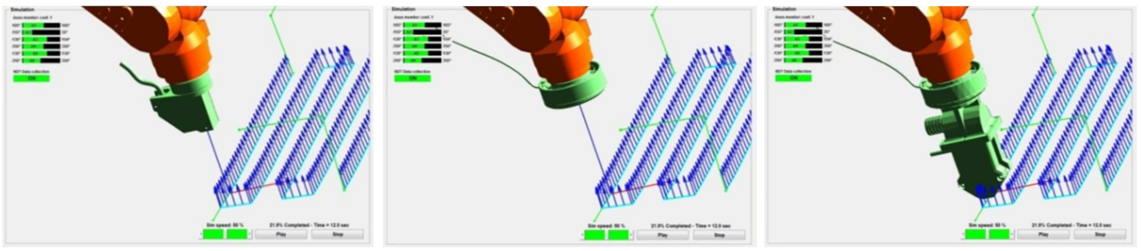
Axes monitor: conf. 1

-155°	A1	155°
-180°	A2	65°
-110°	A3	170°
-165°	A4	165°
-140°	A5	140°
-Inf°	A6	Inf°

Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

Figure 6.31 – Simulation environment with the University of Strathclyde cell setup.

Figure 6.32 shows three different tools attached to the robot arm. The choice of a specific tool is only for simulation purposes, since for the execution of the sub-scan the robot will use the tool parameters of the initial tool path.



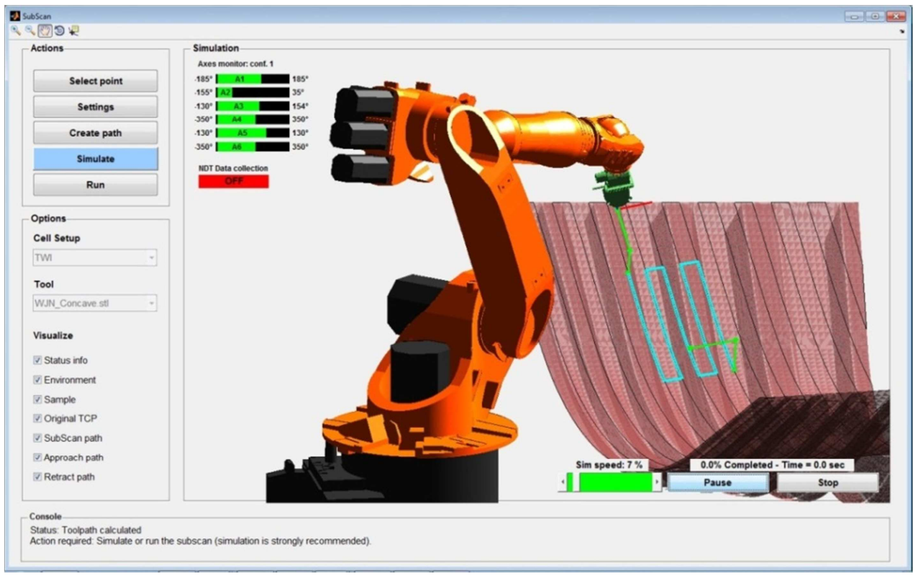
(a) (b) (c)

Figure 6.32 – Three different tools attached to the robot arm: a) laser distance meter, b) magnetic holder and c) magnetic holder and water jet nozzle.

The simulation window contains a sliding bar, to set the simulation speed, and the *Play* and *Stop* buttons. The user can start, pause, stop and restart the simulation and adjust the simulation speed even halfway through the simulation. It is also possible to zoom in, zoom out, pan and rotate the view to get enough details.

Figure 6.33 shows five screenshots taken during a simulation at 0, 25, 50, 75 and 100% of completion. The simulation starts from the first point of the approaching motion and ends with the last point of the retracting motion, where the NDT data collection is OFF. The collection of NDT signals is also OFF at 25% and 75% of completion where the robot arm is manipulating the probe across the step. The data collection is instead ON at 50% of completion where the probe is following the middle inspection line.

0 %



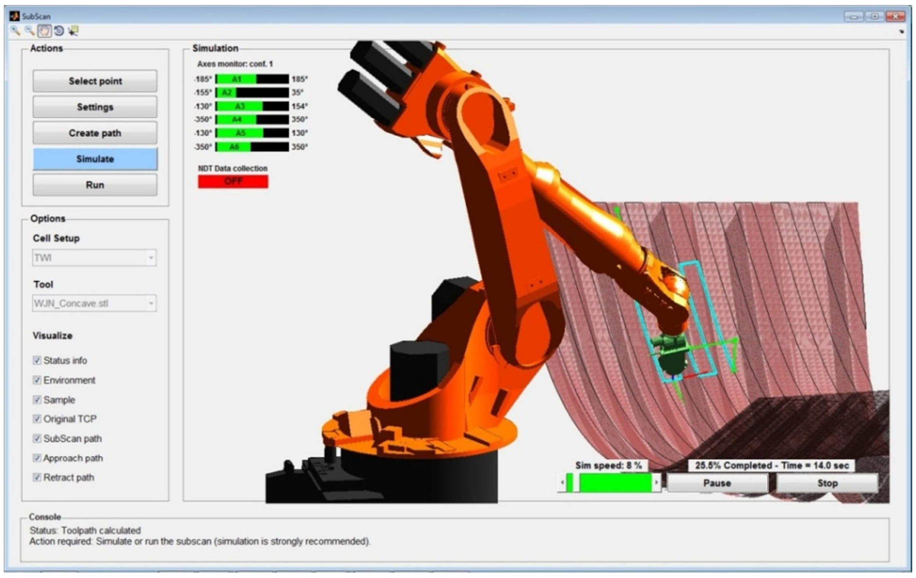
Cell Setup: TWI

Tool: WJN_Concave.stl

☑ Status info
☑ Environment
☑ Sample
☑ Original TCP
☑ SubScan path
☑ Approach path
☑ Retract path

Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

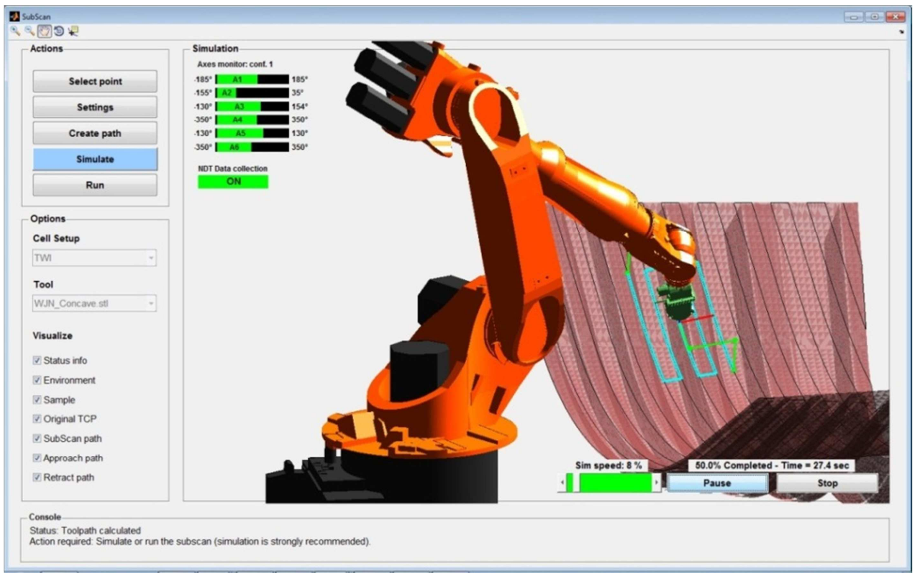
25 %



Options
Cell Setup: TWI
Tool: WJN_Concave.stl
Sim speed: 8 %
25.5% Completed - Time = 14.0 sec

Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

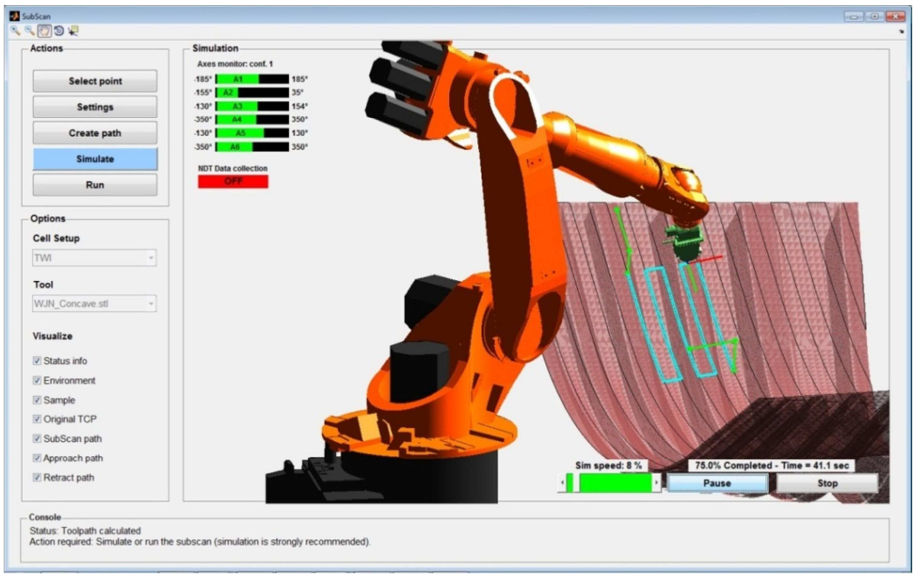
50 %



Options
Cell Setup: TWI
Tool: WJN_Concave.stl
Sim speed: 8 %
50.0% Completed - Time = 27.4 sec

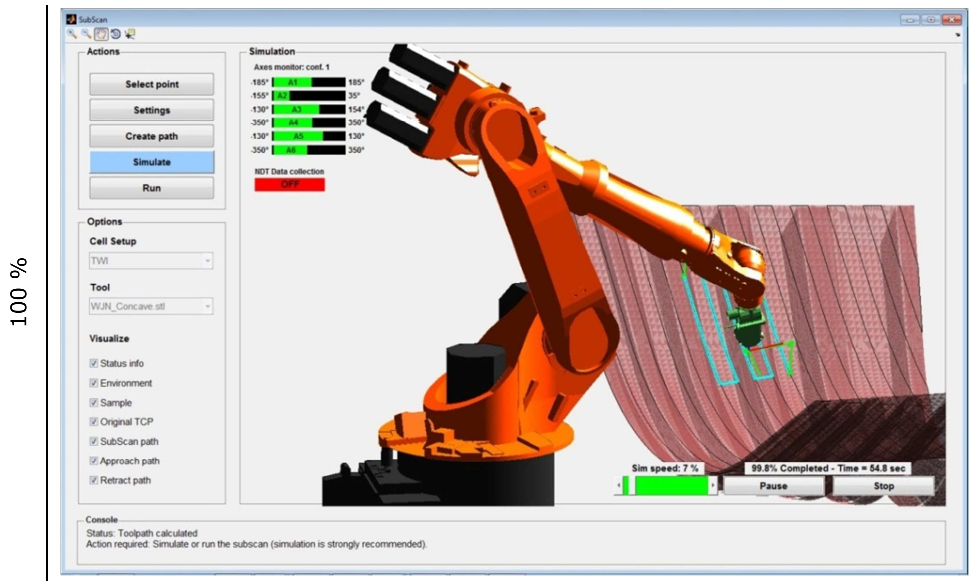
Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

75 %



Options
Cell Setup: TWI
Tool: WJN_Concave.stl
Sim speed: 8 %
75.0% Completed - Time = 41.1 sec

Console
Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).



Status: Toolpath calculated
Action required: Simulate or run the subscan (simulation is strongly recommended).

Figure 6.33 – Five screenshots during the simulation at 0, 25, 50, 75 and 100% of completion.

6.5.2.1. Data exchange with the IntACom software main application

The developed MATLAB based path-planning module is able to use a dataset of received robot feedback positions to arrange a "sub-scan" tool-path and execute it via the external control based on the Robot Sensor Interface (RSI). Since the new tool path is generated from the original coordinates of the Tool Central Point (TCP), the user can easily generate a reliable and effective robot path free from singularities. The exploitation of the external control is a novel way to speed up the process to return to numerous points of interest, after completing the main scan.

The application was initially developed as a stand-alone application, debugged and then integrated with the IntACom software. It constitutes a real example of how the integration of external control capabilities has helped to increase the level of automation and to create a seamless platform for NDT testing. The overall inspection process becomes faster, since there is no need to redefine most of the inspection parameters when the user moves from the acquisition and post-processing data visualization module to the path-planning functionalities.

All the necessary variables are transferred from the main IntACom GUI to the Matlab software removing downtime and possible errors that can be caused by re-definition. These variables are:

- TCP dataset from the original robot feedback;
- Index of the point of interest selected by the user from the main C-scan, superimposed to the CAD model of the part;

- Robot base reference system and tool parameters;
- Tessellated CAD model of the sample under inspection;

Figure 6.34 shows the schematic representation of data exchange between the main IntACom software platform and the MATLAB add-on application.

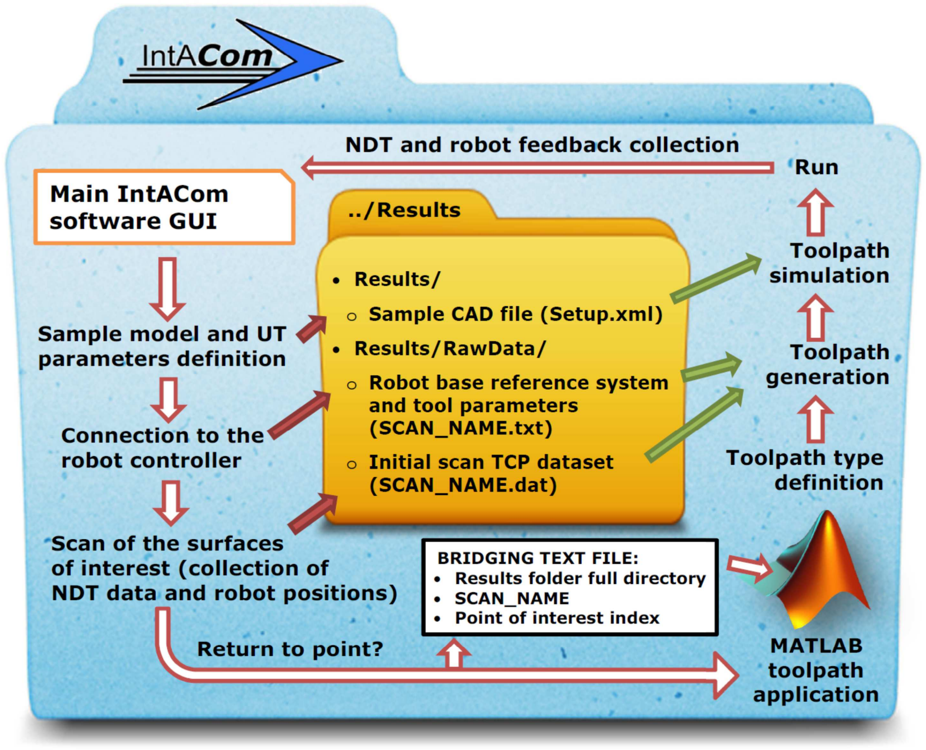

IntACom
NDT and robot feedback collection
Main IntACom software GUI
../Results
Run
Results/
Sample CAD file (Setup.xml)
Results/RawData/
Robot base reference system and tool parameters (SCAN_NAME.txt)
Initial scan TCP dataset (SCAN_NAME.dat)
Toolpath simulation
Toolpath generation
Toolpath type definition
Sample model and UT parameters definition
Connection to the robot controller
Scan of the surfaces of interest (collection of NDT data and robot positions)
BRIDGING TEXT FILE:
Results folder full directory
SCAN_NAME
Point of interest index
Return to point?
MATLAB toolpath application

Figure 6.34 – Data exchange between the main application and the path-planning module.

6.6. Future development

Future work will concentrate on the integration of metrology equipment to provide automated part identification and verification. It will allow parts to be positioned in the optimum position with better accuracy. An interesting feature to be investigated will be the automatic detection of part conformity to CAD data within set limits, in order to either adjust scan paths or reject the part.

The development of real-time dynamic UT adaptation for probes scanning highly curved surfaces is to be investigated. Improvements of UT through-transmission inspection for samples with variable thickness are expected.

IntACom industrial partners have provided funding support for another year, while a proposal for a larger program of work for a further three years is submitted. Work will include the inspection of thicker components and the detection of damage under metal fasteners or erosion shields using full matrix capture (FMC) techniques, optimized for use on composites.

The research presented in this thesis has been instrumental to TWI to start looking ahead. TWI aims at establishing a larger robot cell with multiple robots on tracks for the inspection of larger components such as wing spars and skins, wind turbine blades and boat hulls. Development scope is expected to extend to include high accuracy inspection of complex metal parts such as those produced by the latest additive manufacturing methods (e.g. laser metal deposition).

6.7. Wrapping-up

This chapter presented the contribution of this research to the development of the IntACom project. It was a 3-years project developed by TWI and supported by a number of major aerospace partners (Rolls-Royce, GKN and Bombardier) and Welsh government.

The IntACom project aimed to develop a robot based system for the rapid automated inspection of complex geometry composite components with the objective of increasing inspection throughput by a factor of four. The result is a demonstrator system which is capable of inspecting complex geometries, and with a high throughput.

The development of the system was presented. The system uses integrated software that intuitively steers the user through the inspection process. The core module of the IntACom software for NDT data acquisition and display was developed by TWI engineers. A simple-to-use graphical user interface (GUI) was developed to control all aspects of the robotic inspection, from initial loading of part data, through scanning of the part to data analysis. Many unique features, including the real time B-scan for optimization of PAUT settings and the external control of the robotic manipulators to allow returning to points of interest, increase the usefulness of the inspection process.

Although the IntACom project integrated inputs coming from many source, an effort was made to discriminate the work developed by the author of this thesis and the work done by other TWI engineers.

The PhD research supported the IntACom project through the design of specific hardware and software solutions. As far as the hardware solutions are regarded, the author developed a magnetic tool holder for the robot end-effector, suitable water-jet nozzles prototypes, and the design of water collection trays and of a jig table. The novel external robot control, described by the main body of this thesis, was incorporated in the IntACom software. The author developed an add-on software module to achieve fast rescanning of areas of interest of a part under inspection, through the exploitation of the external control method. The add-on module is capable of introducing substantial advantages to speed-up the industrial inspection processes. It was a valuable and remarkable contribution of the author to the final performances of the IntACom system

Future work is related to the development of real-time dynamic UT adaptation for probes scanning highly curved surfaces, support for UT through-transmission inspection of samples with variable thickness and implementation of metrology for part position assessment and surface mapping.

References

[1] KUKA. *KR 16 L6-2 Reference manual*. Available: http://www.kuka-robotics.com/en/products/industrial_robots/low/kr16_l6_2/ - Accessed on 01/07/2014

[2] T. J. Petronis, "Tool changers, collision and force sensing," *Industrial Robot: An International Journal,* vol. 23, pp. 29-34, 1996.

[3] ATI. Available: http://www.ati-ia.com/Products/collision_sensor/robot_collision_sensor.aspx (Accessed 25/09/2013)

[4] KUKA. *KR 16 L6-2 Specifications*. Available: http://www.kuka-robotics.com - Accessed 5/12/2012

[5] First4Magnets. Available: http://www.first4magnets.com (Accessed 22/06/2013)

[6] W. F. Smith and J. Hashemi, *Foundations of materials science and engineering*: Mcgraw-Hill Publishing, 2006.

[7] J. Happel and H. Brenner, *Low Reynolds number hydrodynamics: with special applications to particulate media* vol. 1: Springer, 1983.

[8] K. D. Nilsen, "Reliable real-time garbage collection of C++," *Computing Systems,* vol. 7, pp. 467-504, 1994.

[9] K. Houstoun and E. Briggs. Rapid Addition leverages Microsoft .NET 3.5 Framework™ to build ultra-low latency FIX and FAST processing. Available: http://www.rapidaddition.com/news/microsoft-a-rapid-addition-publish-white-paper-on-building-low-latency-applications-for-financial-markets.html - Accessed 05/08/2014

[10] M. Burns, "Automated fabrication," *Englewood Cliffs,* 1993.

[11] CENIT. *FastSurf*. Available: http://www.cenit.com/en_EN/plm/digital-factory/software/-fastsurf.html - Accessed 18/07/2014

[12] E. Cuevas, M. López, and M. García, "Ultrasonic Techniques and Industrial Robots: Natural Evolution of Inspection Systems," presented at the 4th International Symposium on NDT in Aerospace, Berlin (Germany), 2012.

7. SYSTEM PERFORMANCE EVALUATION - HIGH VALUE AEROSPACE INSPECTION

7.1. Introduction

The previous chapter presented the development of the prototype robotic NDT system, designed within the IntACom project. The successful outcomes of the project pleased the founding bodies (Rolls-Royce, GKN, Bombardier and the Welsh Government), which have encouraged an extension of the original project and future phases of the research.

Moreover the project attracted the attention of the scientific journalism. An article in Aerospace Testing International magazine [1], published in April 2015, provided an overview of state-of-the-art automated NDT systems incorporated into the production process of new airliner models. The article highlights the IntACom system as one of the most advanced systems.

This chapter presents some specific applications of the IntACom prototype robotic NDT system. The chapter describes some experimental tests carried out throughout the different phases of the system integration. This helps to describe relevant practicalities to consider during the employment of the IntACom robotic system for phased-array inspection of complex shape high value aerospace parts.

Although many tests were carried out during the development of the IntACom system, this chapter reports only those experimental tests that engaged the author of this thesis and were relevant to optimize the performances of the system. The author was involved in the creation of the physical experimental setups and in the research of solutions to both substantial and contingent problems. The early tests described in Section 7.2.1 were used to characterize the performances of the water-jet nozzle prototype. The optimum stand-off between the nozzle and the part surfaces was found through the experimental tests in Section 7.2.2. The scan of one of the most challenging IntACom target samples (a CFRP fan blade) is described in Section 7.2.3. A suitable approach to achieve accurate alignment of two robots for executing through-transmission inspections was found and it is described in Section 7.3; it fulfilled an important requirement of the industrial project.

The applications are grouped into two sections, respectively dedicated to the pulse-echo and the through-transmission technique. The chapter ends with a quantitative evaluation and discussion of the IntACom system final performances, developed by the author.

7.2. Pulse-echo applications

7.2.1. Scan of flat surfaces

The first relevant experimental results, collected through the IntACom robot manipulators, date back to the time the water circulation system described in Section 6.2.4 was integrated. The water jet nozzles enabled the possibility to carry out ultrasonic phased-array inspections of target samples without the necessity of placing the samples within immersion tanks. The water jet creates a local coupling area that moves with the probe during the scan. Figure 7.1 shows the water jet nozzle during the scan of the flat surface of a composite sample (500 x 500 mm).

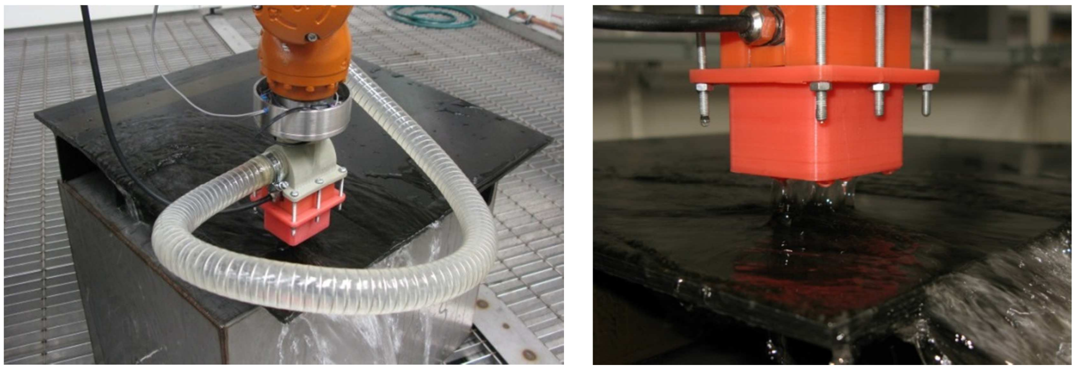

(a) (b)

Figure 7.1 – Pictures of the water jet nozzle during the scan of the flat sample surface.

Figure 7.1a is a close-up of the water jet column, with 25mm gap between the nozzle and the sample surface.

The performances of the water jet coupling system were tested for different speeds of the robot motion (between 50mm/s and 300mm/s). The sample was placed horizontally and vertically. Figure 7.2 shows the Time Of Flight (TOF) C-scans of the sample obtained with the water jet nozzle. They are compared with the C-scans obtained through the immersion technique. The water jet added visible noise to the UT results, especially for in the case with horizontal water column (sample in vertical position).

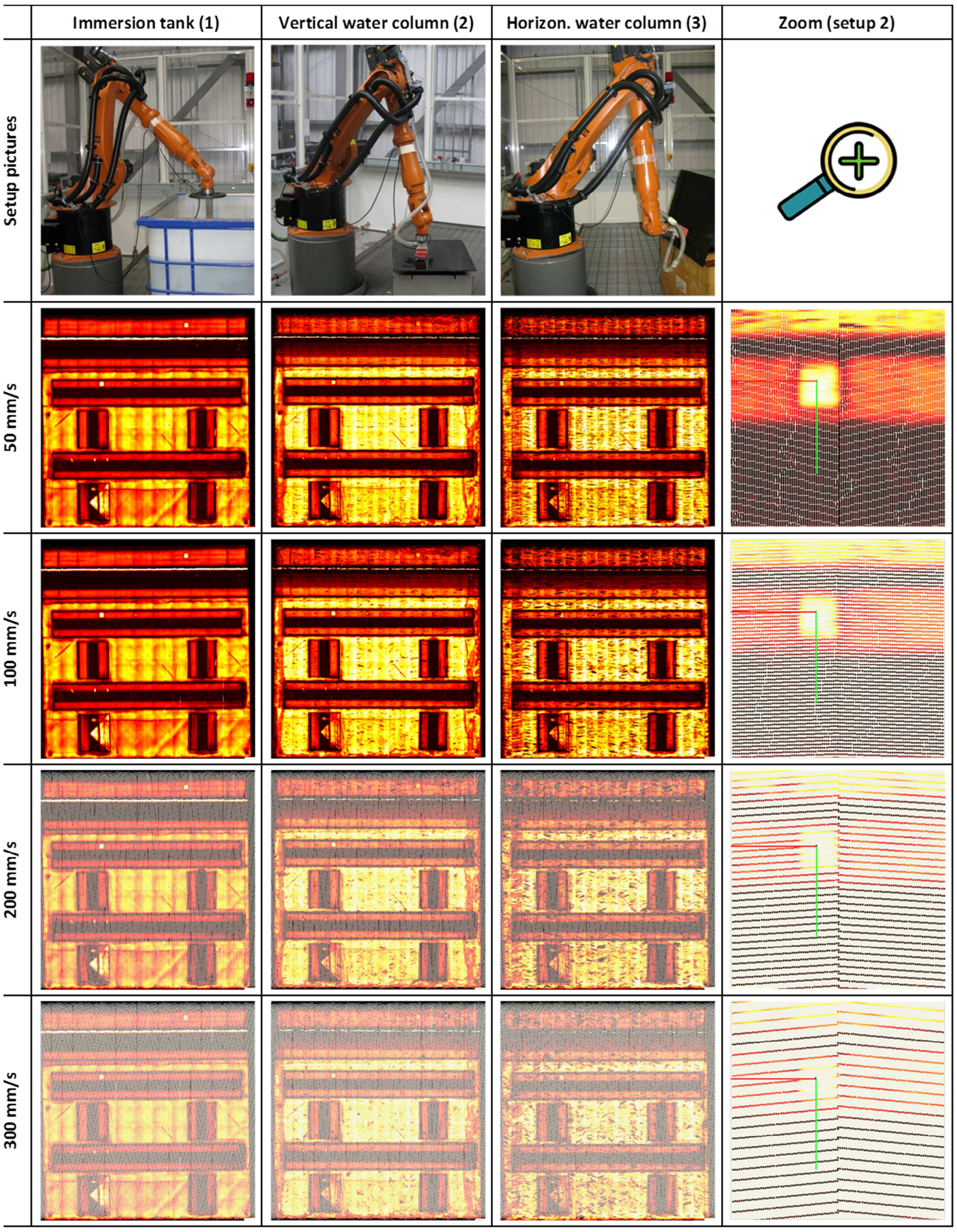


Figure 7.2 – Comparison of TOF C-Scans.

The perimeter of the C-scan obtained with the sample placed horizontally appears to be affected by noise. It was discovered to be caused by the squirted water going around the edges of the sample when the water nozzle nears the perimeter. This changes the acoustic impedance of the sample back wall. Moreover the ultrasonic waves travelling across additional thickness, due to the layer of water beyond the back wall, can also produce a longer time of flight of the reflected waves.

Figure 7.2 also compares the distribution of the A-Scan points on a smaller area that was zoomed through the same magnification factor for all the scans. It can be noticed how 100mm/s speed produces a uniform resolution, since the points seem to be equally spaced across all directions. The C-scan obtained at 50mm/s seems to contain an excess of points along the direction of scan. The spacing between the B-lines is smaller than the spacing between the points along each B-scan; the bigger the scanning speed the bigger the distance between the B-lines. The B-lines appear to be clearly separate in the scan obtained at 200mm/s. The transversal resolution becomes unacceptable, compared to the B-scan resolution, in the 300mm/s scan.

Therefore, 100mm/s was the speed limit for the scanning settings used in this experimental setup. This limit regards the scans carried out with a 64-element phased-array transducer, when the probe elements are fired in groups of 16 and the active area is electronically scanned across the active axis of the probe with step of 1 element at a time. It produces 49 A-Scans per each B-Scan. The phased-array probe gave a resolution of 0.6mm along the B-Scans and a 29.4mm wide inspection area for every C-Scan pass. The scanning speed is limited by the UT data acquisition rate. A reduction of the number of elements of the phased array probe or an increase of the probe pitch produces the possibility to increase the scanning speed. In other words, the scanning speed suitable to get uniform resolution would be greater.

7.2.2. Investigating the optimum stand-off

Section 5.2.1 presented some experiments carried out to validate the path-planning operated by RoboNDT; the scans of an aerospace winglet obtained through the IntACom system.

Section 5.2.1 aimed at comparing the C-scans obtained through using RoboNDT tool-paths with those obtained using FastSurf paths, this section reports about antecedent investigations undertaken to define the optimum stand-off for the water jet nozzle given in

Figure 7.3b. The main-skin 1.6 m^2 surface of the sample was scanned. The setup of the scan is shown in Figure 7.3a.

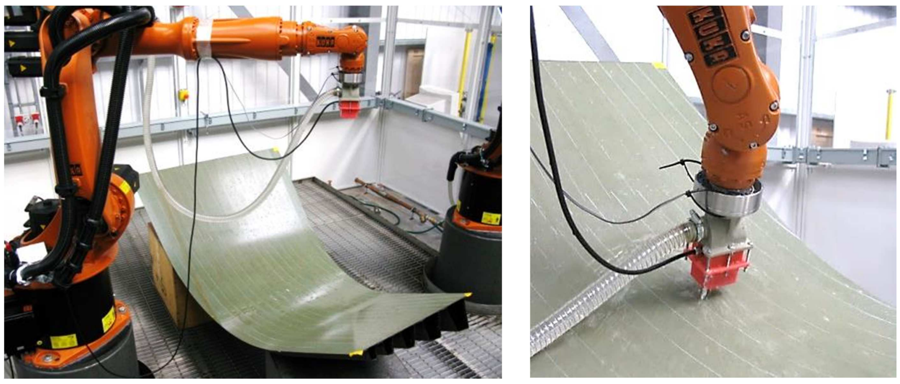
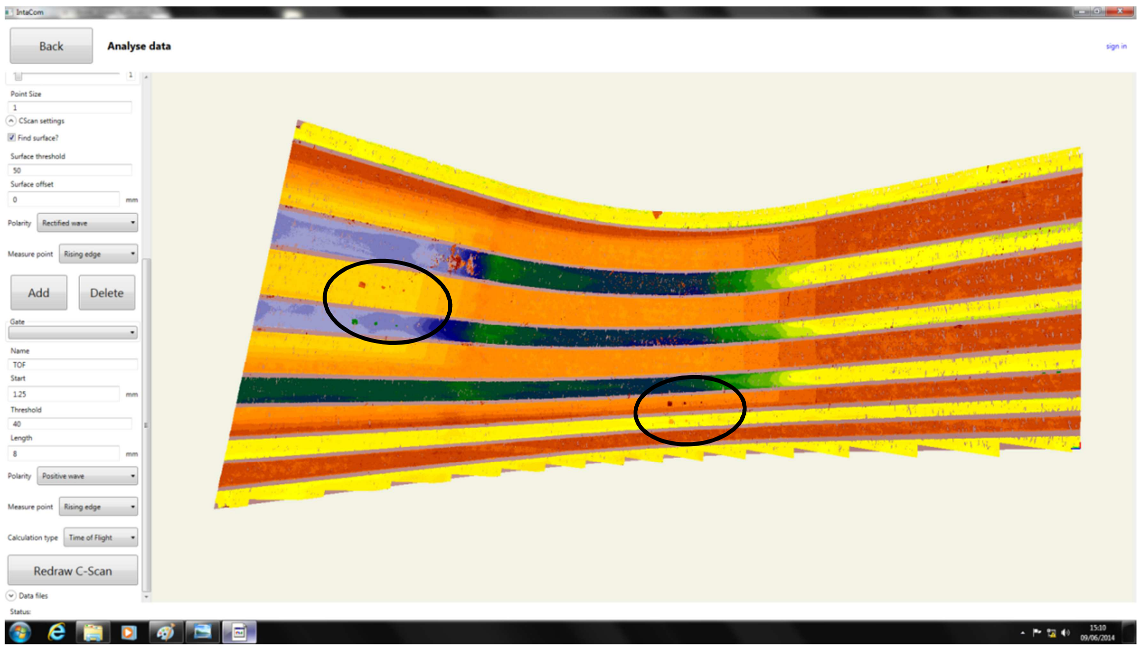

(a) (b)

Figure 7.3 – Scan of the GKN winglet main skin surface.

Figure 7.4 shows the C-scan obtained from the first test. The data was acquired using a scanning speed of 100mm/s and a scanning increment of 29.4mm. The resolution was set equal to 0.6mm. The main feature of the C-Scan is the clear presence of longitudinal areas with different thickness. The thickness differences are nicely shown by the TOF map. The sample contains defects, intentionally introduced by the manufacturer.

One group of 6 square delamination defects, with different sizes, appear on the left hand side of the map. The smallest defect is 5mm wide and the biggest is 15mm. A similar group of defects is present at the bottom of the map, roughly in the middle.

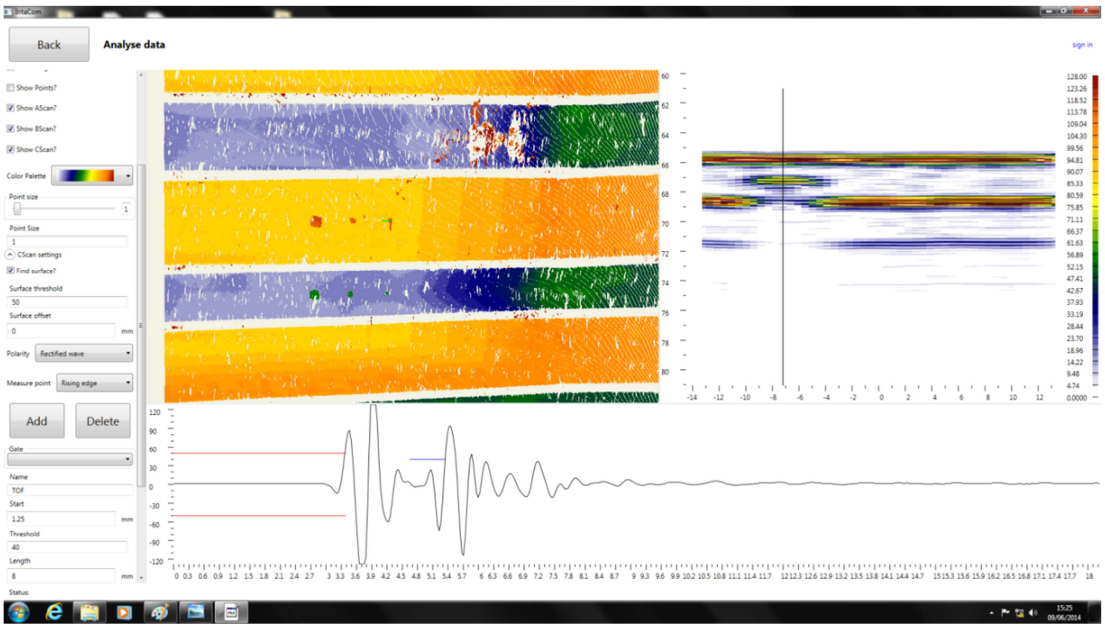


Point Size: 1
CScan settings
[✓] Find surface?
Surface threshold: 50
Surface offset: 0 mm
Polarity: Rectified wave
Measure point: Rising edge
Gate:
Name: TOF
Start: 1.25 mm
Threshold: 40
Length: 8 mm
Polarity: Positive wave
Measure point: Rising edge
Calculation type: Time of Flight

Figure 7.4 – Initial TOF C-Scan of the GKN winglet main-skin surface.

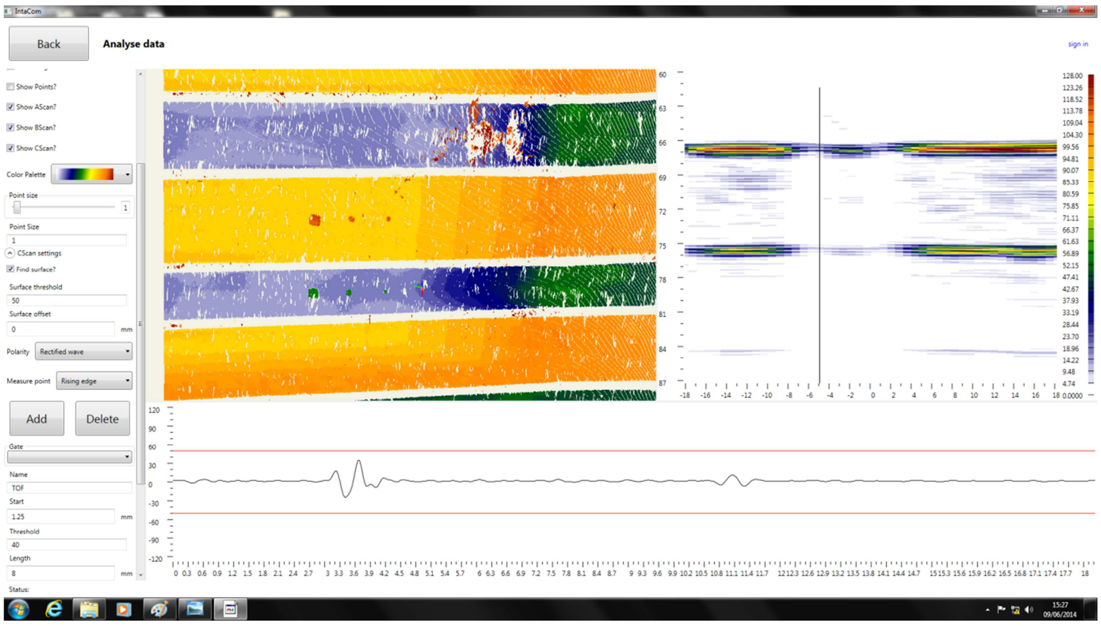
Figure 7.5 shows a close up of the first group of defects. The GUI of the IntACom software lets the user analyse the collected data through visualization of B-Scans and the A-Scans. The B-Scan is given on the right hand side of the screen; the A-Scan section is at the bottom. The B-Scan is very useful to size the C-Scan features and the potential defects. The screenshot given in Figure 7.5 shows the B-Scan and the A-Scan obtained from the smallest delamination. The noisy C-Scan area close to the top-right corner of the zoomed region is area of the carbon fibre composite skin with dry-fibres.

The first results coming from the scan of the curved surface were promising; however the overall quality of the C-Scan was quite poor. The zoomed C-Scan region given in Figure 7.5 reveals how the acquired scan is affected my missing data that creates large gaps in the resulting map. This was discovered to be caused by the poor quality of the water column created by the nozzle. The ultrasound waves were refracted by the ripples of the water column. The refraction that took place in the water column caused the dissipation of most part of the waves' energy. This is shown in the B-scan and A-scan given in Figure 7.6, where the information of the front surface and the back wall is strongly attenuated by the described phenomenon.



☐ Show Points?
☑ Show AScan?
☑ Show BScan?
☑ Show CScan?
Color Palette
Point size: 1
Point Size: 1
CScan settings
☑ Find surface?
Surface threshold: 50
Surface offset: 0 mm
Polarity: Rectified wave
Measure point: Rising edge
Gate
Name: TOF
Start: 1.25 mm
Threshold: 40
Length: 8 mm
Status:

Figure 7.5 – Close-up of the first group of defects.


IntaCom

Back
Analyse data

- [ ] Show Points?
- [x] Show AScan?
- [x] Show BScan?
- [x] Show CScan?

Color Palette

Point size: 1

Point Size: 1

CScan settings

- [x] Find surface?

Surface threshold: 50

Surface offset: 0 mm

Polarity: Rectified wave

Measure point: Rising edge

Add
Delete

Gate:

Name: TOF

Start: 1.25 mm

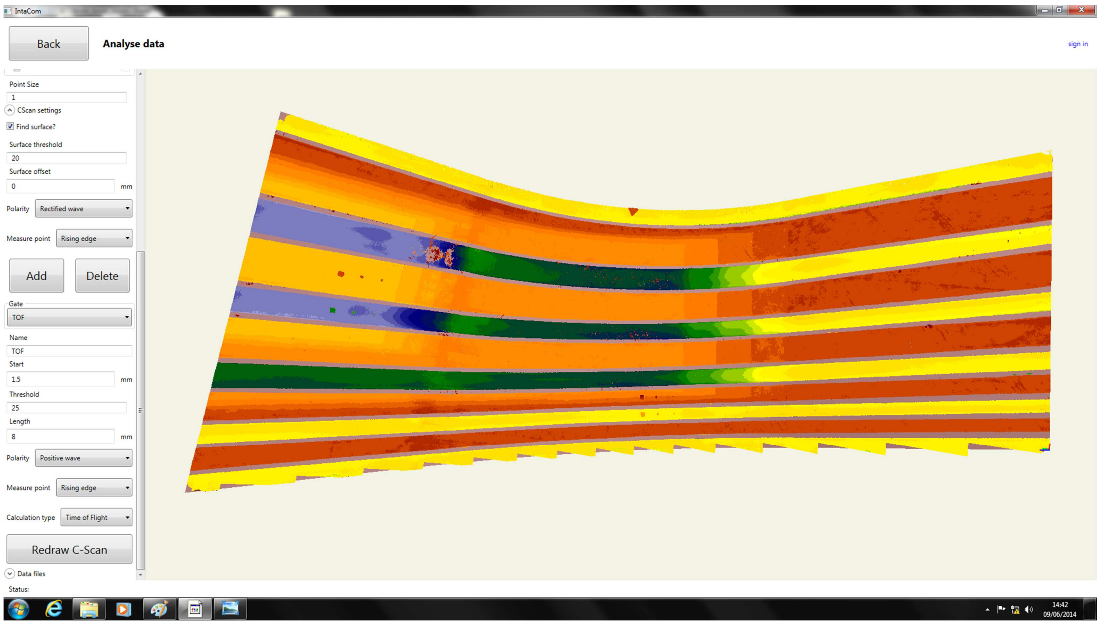
Threshold: 40

Length: 8 mm

Status:

Figure 7.6 – UT signals strongly attenuated by the refraction caused by ripples of the water jet.

As an attempt to improve the quality of the C-Scan map, the gap between the water jet nozzle and the surface was progressively reduced from 25mm to 8mm. This action was driven by the idea that the shorter the water column the less the number of ripples in it. Good C-scans were obtained for standoff smaller than 15mm. Figure 7.7 shows the C-Scan acquired with the shorter free column of water (8 mm). It is visible smoother than the initial scan. The zoomed region given in Figure 7.8 also confirms a great improvement.


IntaCom

Back
Analyse data

Point Size: 1

CScan settings

- [x] Find surface?

Surface threshold: 20

Surface offset: 0 mm

Polarity: Rectified wave

Measure point: Rising edge

Add
Delete

Gate: TOF

Name: TOF

Start: 1.5 mm

Threshold: 25

Length: 8 mm

Polarity: Positive wave

Measure point: Rising edge

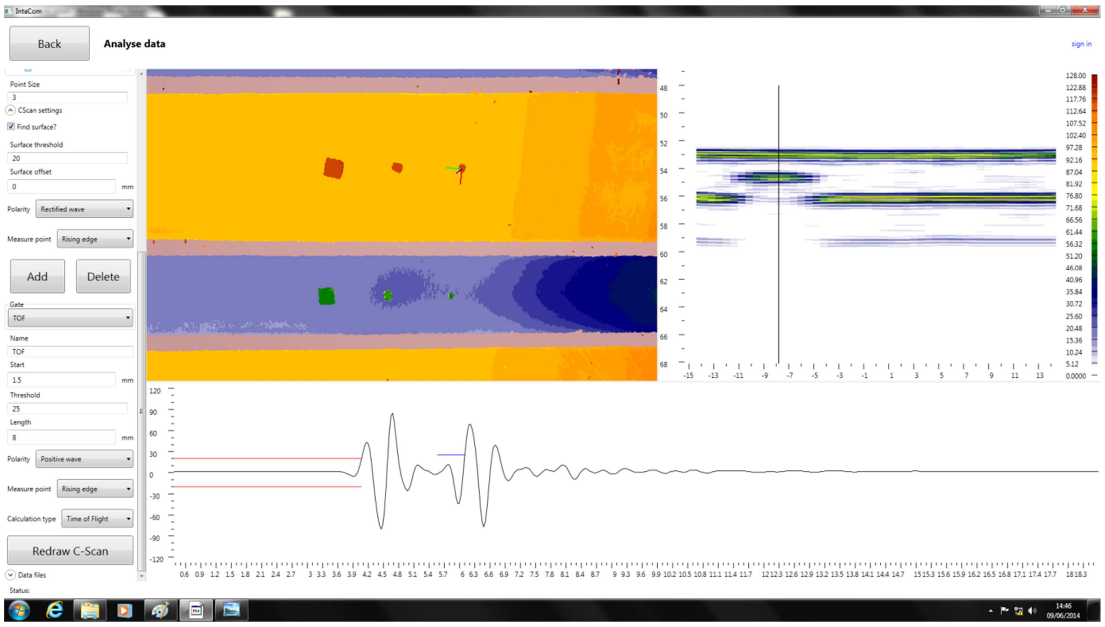
Calculation type: Time of Flight

Redraw C-Scan

Data files

Status:

Figure 7.7 – C-Scan acquired with the shorter column of water (8 mm).



Point Size: 3
CScan settings
Find surface? ✓
Surface threshold: 20
Surface offset: 0 mm
Polarity: Rectified wave
Measure point: Rising edge
Gate: TOF
Name: TOF
Start: 1.5 mm
Threshold: 25
Length: 8 mm
Polarity: Positive wave
Measure point: Rising edge
Calculation type: Time of Flight

Figure 7.8 – Close-up of the C-Scan acquired with the shorter column of water (8 mm).

The developed water jet nozzle prototype allowed the achievement of the target inspections required by the industrial partners of IntACom; however short coupling water columns can only be used for non-highly curved surfaces.

Future work will regard the improvement of the design of the water jet nozzle to maximize the length of the free water column.

7.2.3. CFRP fan blade

A carbon fibre reinforced plastic (CFRP) blade was one of the most challenging target samples provided by the industrial partners to prove the prototype system designed through the IntACom project.

The main challenges given by the CFRP blade are the strong anisotropy of the material and the variable thickness. The ultrasound wave is not reflected back along the incident direction, when the back-wall is not parallel to the front surface. Moreover the variable thickness of the sample and the strongly attenuated back-wall reflections produce difficult gating of the A-scan signals across the geometry of the blade.

The difficulties were overcome through a software trick. The possibility to dynamically adjust the length of the inspection gate was implemented to the IntACom software by TWI engineers. The length of the gate is adjusted for each A-Scan to follow the position of the back-wall. The position of the back-wall is monitored through the CAD model of the part imported into the IntACom software.

For each point of the C-scan, the software was made able to use the angular coordinates (roll, pitch and yaw angles) received from the robot controller, in order to calculate the vector perpendicular to the surface. The intersection points between the perpendicular direction and the mesh of the part model are found to estimate the local thickness. Figure 7.9 shows the experimental setup used to prove the software. FastSurf was used to generate a tool-path to manipulate a 64 element – 5MHz phased array probe. The probe was manipulated with a speed of 300mm/s. Only the central 16 elements of the probe were fired to reduce the width of the active area and avoid any striping effect that may be caused by excessive curvature of the inspected surface.

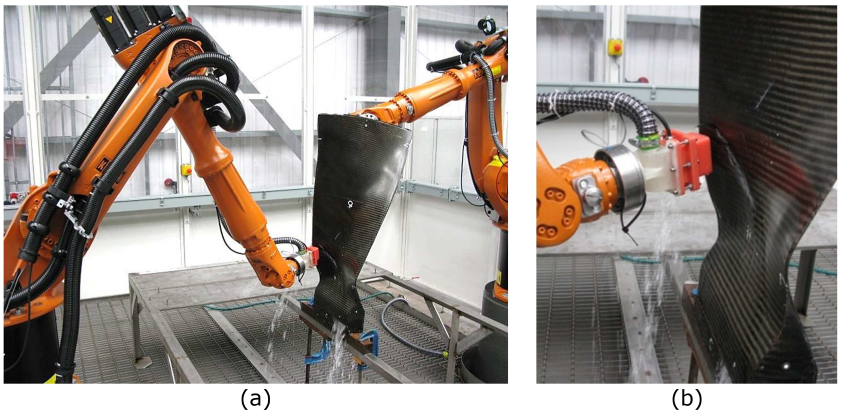
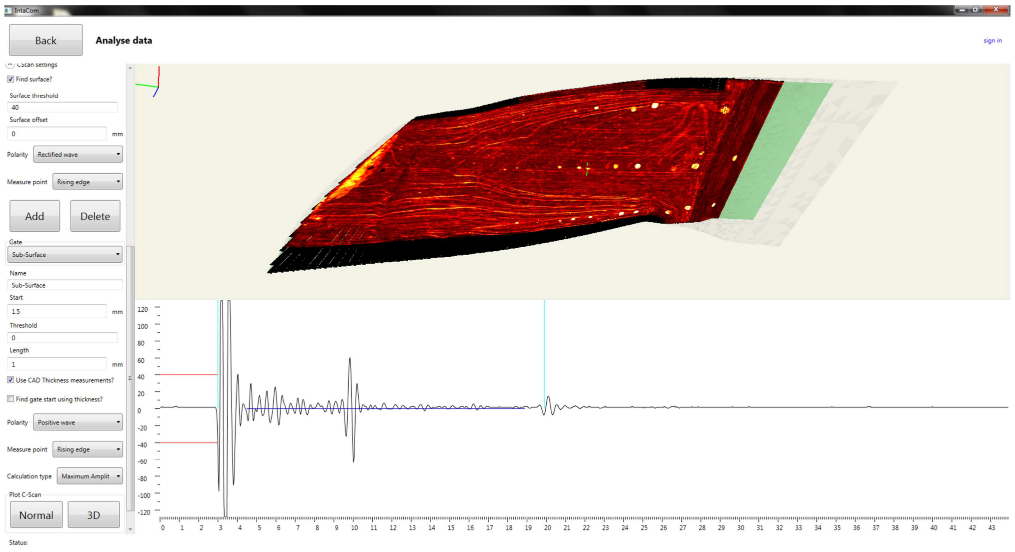


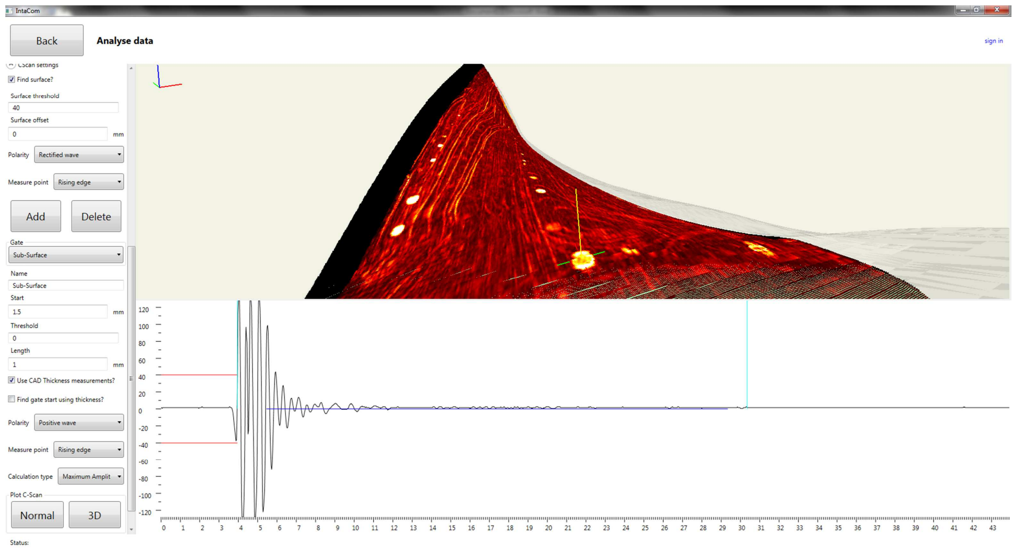
Figure 7.9 – Inspection setup for the CFRP blade (a) and close up of the water jet nozzle (b).

Figure 7.10 and Figure 7.11 show two screenshots of the IntACom software GUI, taken during the analysis of the inspection results. The sample contains artificial delaminations; they are round inserts of polytetrafluoroethylene (PTFE) with diameters between 5 and 20mm. All defects artificially embedded in the sample by the manufacturer are visible and recognizable. This was deemed as a very good result for a pulse-echo inspection.



CScan settings
☑ Find surface?
Surface threshold: 40
Surface offset: 0 mm
Polarity: Rectified wave
Measure point: Rising edge
Gate: Sub-Surface
Name: Sub-Surface
Start: 1.5 mm
Threshold: 0
Length: 1 mm
☑ Use CAD Thickness measurements?
☐ Find gate start using thickness?
Polarity: Positive wave
Measure point: Rising edge
Calculation type: Maximum Amplit
Plot C-Scan
Status:

Figure 7.10 – Screenshot of IntACom software during analysis of the results.



[✓] Find surface?
Surface threshold: 40
Surface offset: 0 mm
Polarity: Rectified wave
Measure point: Rising edge
Gate: Sub-Surface
Name: Sub-Surface
Start: 1.5 mm
Threshold: 0
Length: 1 mm
[✓] Use CAD Thickness measurements?
[ ] Find gate start using thickness?
Polarity: Positive wave
Measure point: Rising edge
Calculation type: Maximum Amplit

Figure 7.11 – Screenshot of IntACom software during analysis of the results.

7.3. Through-transmission inspections

The IntACom software was developed to be capable of supporting the acquisition of through-transmission data, besides pulse-echo data. However additional investigations were needed to define the path-planning procedure, necessary to obtain suitable tool-paths. The two robot manipulators of the IntACom system have to maintain a good alignment and their motion has to be synchronized, in order to efficiently perform through-transmission inspections.

Figure 7.12 shows a picture of the setup for the first attempt to set up a through-transmission scan for a small flat surface. Two raster scan tool-paths, one for the master robot and one for the slave robot, were generated through the FastSurf software and synchronization between the two robots was attempted by inserting KUKA Robot Language (KRL) command to manage the status of two digital signals. One digital output of the master robot was connected to one digital input of the slave robot and vice-versa.

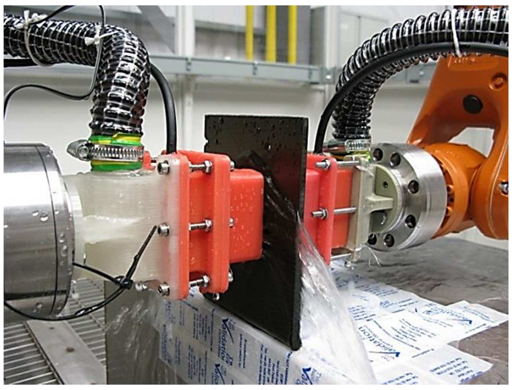


Figure 7.12 – First attempt to set up a through-transmission scan for a small flat surface.

The limited size of the sample in Figure 7.12 did not allow moving the robot end-effector at high speed. The scanning speed achieved between the acceleration and the deceleration ramps was about 30 mm/s. This produced good alignment between the generating and the receiving probe, however poor alignment was discovered when moving to larger surfaces that allow higher speeds.

Figure 7.13 shows the scan of a large curved sample. The sample is an aerospace wing leading edge, made of composite materials; the main layer, made of carbon fibre reinforced plastic (CFRP), is coated by a deposited layer of Inconel[15].

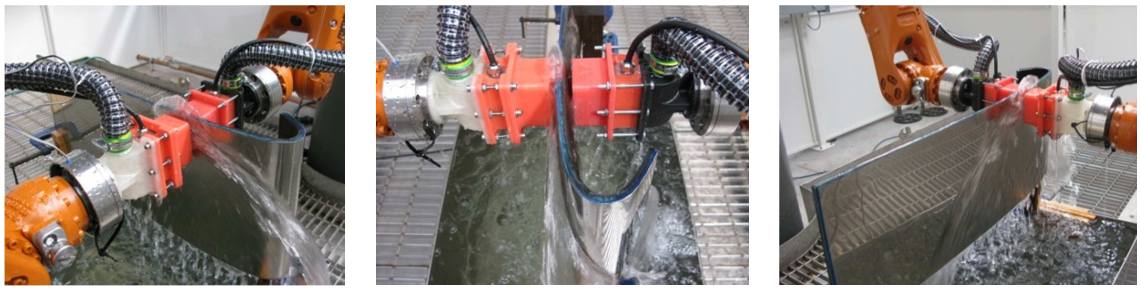


Figure 7.13 – Through-transmission scan of a wing leading edge sample.

Figure 7.14 shows two groups of C-scans, relative to the scan of the central area of the sample. The first column contains the through-transmission results produced by the synchronization method based on digital I/O signals and the pulse-eco scans executed from the Inconel layer (top side) and from the CFRP layer (bottom side) of the sample.

The through-transmission C-scans, obtained synchronizing the robots by digital I/O triggers at the start and at the end of each pass of the raster tool-path, show an unexpected striping effect. The cause of the striping was discovered in the poor synchronization given by the digital I/O flags. It causes a certain misalignment between the generating and the receiving probe, with consequent variability in the sound path and in the signal amplitude.

The through-transmission scans are not influenced by the side of the sample where the generating probe is placed. The through-transmission C-scan obtained after inverting the generating and the receiving probe looks identical to the other.

Very different is the case of the pulse-echo scans. Different results were collected by inverting the side of inspection (everything else was left constant). The visible difference is caused by the different ultrasonic impedance of Inconel and CFRP.

[15] Inconel is a family of austenite nickel-chromium-based alloys. Inconel alloys are oxidation and corrosion resistant materials well suited for service in extreme environments subjected to pressure and heat. When heated, Inconel forms a thick, stable, passivating oxide layer protecting the surface from further attack.

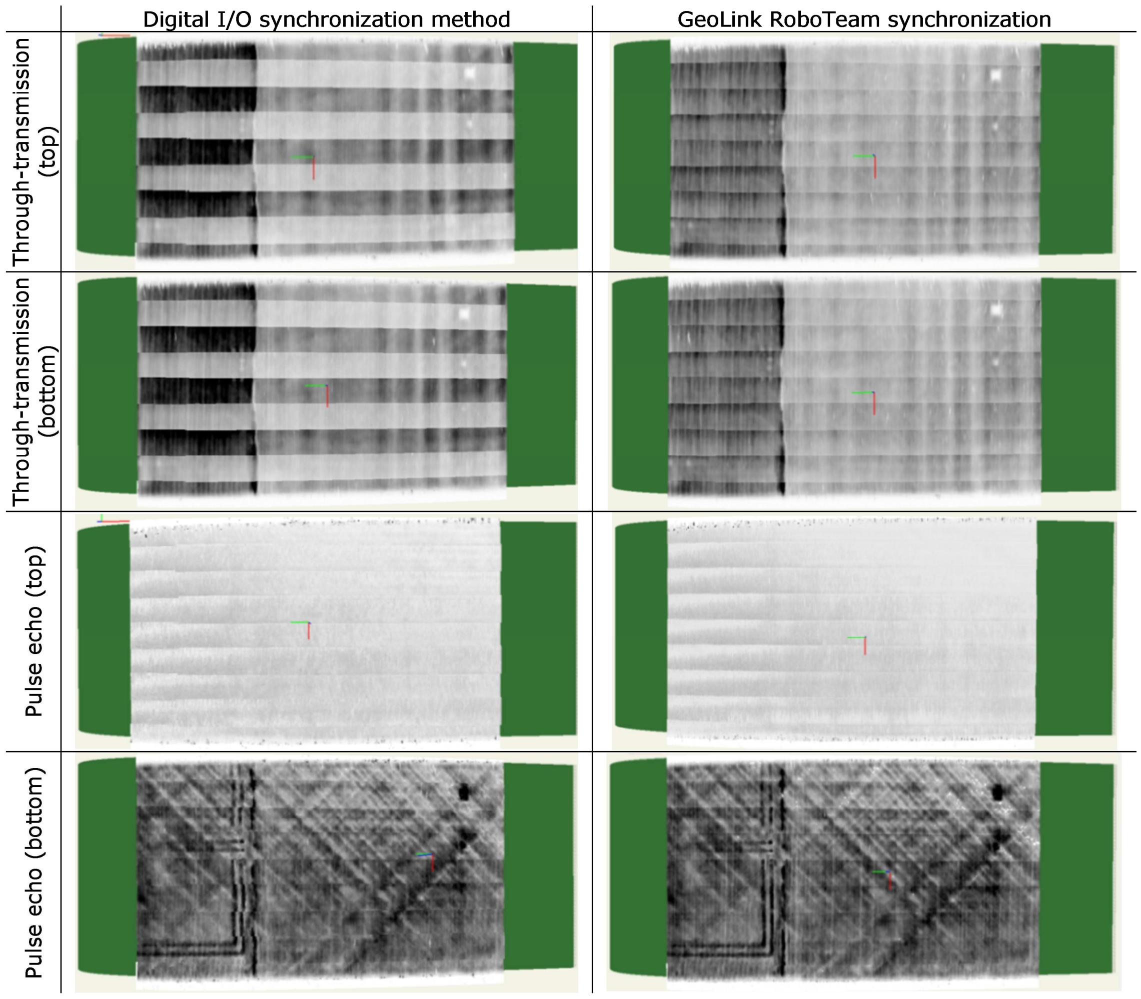

Digital I/O synchronization method
GeoLink RoboTeam synchronization
Through-transmission (top)
Through-transmission (bottom)
Pulse echo (top)
Pulse echo (bottom)

Figure 7.14 – C-scans of the wing leading edge sample.

A solution for the misalignment problem was investigated. The best solution was found in the use of the "*GeoLink approach*". As it was described in Section 3.3.2, the *GeoLink* is a programming method supported by a software add-on (RoboTeam) installed into the TWI KRC4 Robot controllers [2]. The RoboTeam application package enables precisely coordinated teamwork with up to 15 robots through fast synchronization of the path motions. Each robot keeps its standard controller and the RoboTeam software enables the synchronization features. A high-speed local network (Ethernet) is used by the controllers to communicate with one another and synchronize their motion.

The geometric coupling between two robots is one of the program options allowed by RoboTeam, in order to synchronize the motion. Utilising the GeoLink KRL in-line commands, the co-operating software allows geometric coupling of the Slave robot to the Master Robot. Using a unique identifier, a geometric linking command is inserted into each program. From this point forward, the Slave follows the motion of the Master by maintaining the orientation and position of its active tool to that of the Master. Following the linking command, the Master program contains motion instructions to be followed. The Slave program contains no motion commands but follows every movement of the Master. When co-operation is no longer required, geometric tracking is cancelled with a similar command.

Using this approach the end-effectors of the master and slave robot are driven from the Home positions to the start point of the scan through suitable approaching movements. At this point, the GeoLink commands are executed in both robot programmes and the Slave robot starts following the Master robot. At the end of the scan, the Slave robot is decoupled from the Master and both robots return to the Home position. The template of the KRL modules for Master and Slave robot to use the GeoLink approach is given in Appendix D1.

The GeoLink approach allowed reduction of the misalignment between the Master and Slave robot to acceptable levels. This led to better quality results and the C-scans given on the right hand side of Figure 7.14.

The GeoLink method maintains the relative position between the Slave robot tool central point (TCP) and the Master robot TCP constant. Therefore the method is only suitable for through-transmission inspections of parts with constant thickness. This limitation can be removed through exploiting one of the advantages provided by the external control approach described in Chapter 3. As it was mentioned in Section 4.5.4, command coordinates could be sent simultaneously to two robot controllers to achieve synchronized movements.

Future work can comprise the expansion of RoboNDT and its integration with the IntACom platform. The path-planning software could be expanded to enable the generation of tool-paths for through-transmission inspections. The relative command text files, outputted by RoboNDT, would consist in a number of lines comprising two packets of coordinates (the first to be streamed to the Master robot controller and the second to the Slave controller). This would pave the way to the implementation of external control for two or more robots running simultaneously.

7.4. Final performances

Every time new systems are developed, the benchmarking problem arises to compare their performances with current systems. Sometimes, the features of new systems make benchmarking a not trivial problem. It is the case of the IntACom system. This section offers a benchmarking of the new system, comparing its acquisition rates (in terms of squared metres per hour) to the acquisition rates of other systems. The acquisition rates of the IntACom system are the result of the employment of new solutions not available in other systems.

Ultrasonic phased-array probes have been integrated to robot manipulators to take advantage of the wider inspection coverage given by phased-array probes, in comparison to single element probes. More importantly, industrial manipulators allow inspection of complex geometries; a key feature not available in 3-DoF ultrasonic scanners based on gantry manipulators.

7.4.1. Scanning speeds

The theoretical maximum scanning rate can be calculated for a sample of given material by considering the sample thickness and the wave speed in the material. For ultrasonic pulse-echo inspection carried out with water-jet coupling, the theoretical time to capture a single frame (t_f) is given by:

$$t_f = n \cdot (d + e) \qquad \text{Eq. 7.1}$$

and the maximum number of frames per second (FPS), is given by:

$$\text{FPS} = \frac{1}{t_f} = \frac{1}{n \cdot (d + e)} \qquad \text{Eq. 7.2}$$

where n is the number of sub-apertures in a phased array inspection (it is equal to 1 for the traditional single element inspection), d is the acquisition delay from the probe surface and e is the duration of the acquired A-scan.

Assuming $d = 2e$, to allow the acquisition of A-scans without superposition of front-wall waves, it results for a single element probe:

$$t_f = 3e \qquad \text{Eq. 7.3}$$

The duration of an A-scan is given by:

$$e = \frac{2s}{v} \qquad \text{Eq. 7.4}$$

where s is the sample thickness and $\boldsymbol{v}$ is the wave propagation speed in the material. Supposing a composite CFRP sample with propagation speed of 3mm/μs and thickness of 50 mm, it results $e = 33.3$ μs. Therefore the theoretical acquisition time for each frame and the maximum FPS are respectively equal to $t_f \approx 100$ μs and FPS ≈ 10000. The length of the water column between the probe surface and the sample will be equal to 45mm (considering the speed of propagation in water equal to 1.48mm/µs).

Table 7.1 shows the comparison between the infant research systems described in Section 3.4.1 and in Section 3.4.2, respectively based on discrete and continuous robotic motion, and two commercial 3-DoF scanning systems. A USL system and a GE system, based on 3-DoF gantry manipulators were tested at TWI. The USL system runs the *USL Scanner acquisition software (Version 4.51)* and the GE system is equipped with the *GE Nutronik UTxx software*. The acquisition rate of these commercial systems is much higher that the systems presented in Chapter 3 and it is used as a yardstick to evaluate the performances of the IntACom system. The FPS of these systems gets closer to the maximum theoretical value. When they operate traditional pulse-echo single element inspections, they are capable of acquiring 60000 A-scans/min (3.6 million A-scans/hour), providing a scanning rate of 3.6 m^2/hour (assuming a 1mm step between parallel raster scan passes).

	Discrete point measurement system	**Continuous path system**	**USL and GE 3-Dof commercial gantry systems**
Technique	Pulse-echo	Pulse-echo	Pulse-echo
A-scan length (µs)	100	100	100
Sampling freq. (MHz)	50	50	50
Theor. A-scans/sec	10000	10000	10000
A-scans/sec	1.46	17.6	1000
Scan resolution (mm)	1	1	1
Max speed (mm/s)	1.46	17.6	1000
Scanning step (mm)	1	1	1
Scanning rate (m^2/hour)	0.0052	0.062	3.6

Table 7.1 – Performances of ultrasonic single-element scanning systems.

The IntACom system should be compared to similar robotic systems. It is certain that the Tecnatom systems and other commercial systems reviewed in Section 2.4.3 are highly competitive and rival systems opposed to the robotic inspection system prototype developed through the IntACom project.

Like the IntACom system, some of the commercial systems integrate ultrasonic phased array inspection to robotic manipulators; however it is not clear the level of integration between robotic manipulators and the NDT equipment that has been achieved by these competitive systems. The capabilities of their software solutions are not described with enough detail in the related published information, to allow a good comparison with the IntACom system.

Despite the presence of common elements relative to the employed hardware (KUKA robots and water jet coupling system), it is believed that the flexibility of a bespoke robotic NDT system is driven by control software.

First of all, the off-line path generation software is highly critical for the definition of an accurate and suitable toolpath targeted to the chosen inspection techniques. The acquisition software is in charge to exploit the flexibility of motion of industrial robots within their working envelope and the performance of NDT probes to collect consistent data. The collected information must be accurately encoded through the positional data to provide meaningful output. Finally, the way the output is displayed and the freedom left to the user to customize the data display and apply different gates with ease can make one system more attractive than another.

Table 7.2 compares the acquisition rate of the GE 3-DoF scanner based on the *GE Nutronik UTxx software*, deploying phased-array inspection, with the scanning speed of the IntACom system. To ease the comparison, common parameters are kept constant (pulse repetition rate, sub-aperture and scanning resolution).

	GE 3-DoF scanner	IntACom system			
Type of inspection	Phased array	Ph. array	Ph. array	Ph. array	Ph. array
Probe	64el - 1mm pitch	64el - 1mm	64el - 1mm	32el - 1mm	32el - 1mm
Technique	PE	PE	TT	PE	TT
Sub-aperture	10	10	10	10	10
Scanning res. (mm)	1	1	1	1	1
Th. frames per sec (FPS)	192.21	192.21	206.99	459.63	494.98
FPS	44	118	130	272	298
Max scan speed (mm/s)	44	118	130	272	298
Scanning step	54.98	54.98	54.98	22.96	22.96
Scanning rate (m^2/hour)	8.71	23.36	25.73	22.48	24.63

Table 7.2 – Scanning rate of a GE 3-DoF phased-array scanner and of the IntACom system.

The IntACom system shows an average scanning rate of 24m^2/hour. This is 2.8 times faster than the GE 3-DoF scanner deploying phased-array inspection and 6.7 times faster than commercial gantry systems deploying traditional single-element inspection. The ratio between the maximum theoretical FPS and the real FPS can be considered as a quantitative measure of the efficiency of the data acquisition software. This ratio is respectively equal to 0.23 for the GE 3-DoF scanner and 0.61 (average) for the IntACom system, resulting in the IntACom data acquisition being substantially more effective than the GE one. The carefully designed data acquisition and encoding algorithms (see Section 3.8 and 3.9) prove to be effective and able to surpass the performances of the phased array software implemented in the commercial GE system.

7.4.2. Robot feedback rate and data encoding

The collected information must be accurately encoded through the positional data to provide meaningful output.

As reported in Section 2.4.3.4, USL presented a robotic system equipped with fast robot feedback rate. The press released by USL states that, in order to get meaningful ultrasonic imaging applications at scanning speeds, it is required to monitor the probe position in real time [3]. It is mentioned that a rate of 2000 positional readings per second is necessary, so that image pixels can be accurately placed on a C-scan image, with data points every 0.5mm at speed of 1000mm/s. This is necessary to ensure that the size and position of defects and features can be precisely recorded and measured. This result is achieved through using a hardware solution available for Stäubli 6-axis robots. They are equipped with the uniVAL Drive interface. The uniVAL Drive is a "ready to plug" solution to drive Stäubli's range of 4 and 6 axis robots with generic industrial multi-axis controllers and it is designed to facilitate the integration of Stäubli robots with other systems [3].

However it is not clear if, in addition to using the uniVAL Drive, USL considered the possibility of implementing effective time-stamping of robot positions and NDT data, in order to encode NDT information through interpolated positional coordinates.

The modern KUKA robots, used for the IntACom system, allow positional feedback at a rate of 250 Hz (every 4ms) that is much lower than 2000 Hz. The investigations reported in this thesis (see Section 3.7 and Appendix A4) demonstrated that the time taken by the programmed C++ loop of the IntACom software to parse and time-stamp a positional packet coming from the robot controller is approximately equal to 60 ns.

The time at which the position is read by the robot encoders is not equal to the time the position is time-stamped. Considering a speed of 1000 mm/s, the positional uncertainty (P_U) associated to this delay is equal to:

$$P_U = v \cdot t = 1000 \cdot (6 \cdot 10^{-5}) = 0.06\, mm \qquad \text{Eq. 7.5}$$

The KUKA robot controller sends out a feedback position within each cycle of 4ms, however the exact time of streaming is affected by variability that must be resolved to not increase the positional uncertainty. This objective can be fulfilled using a time-stamping unit at least one order of magnitude smaller than milliseconds. The IntACom software uses elapsed "ticks" rather than milliseconds to time-stamp both the NDT data and the positional robot feedback. It was found that a "tick" is the smallest time unit available in C++ language. There is no constant correspondence between real time and "ticks"; this time unit is machine dependent; however, since both robot position and NDT data are time-stamped with the same clock, every time unit (despite being machine dependent) is a congruent unit for measure of time. For the computer used during the software development, the number of ticks per millisecond was equal to 2435.66 ticks/ms (≈ 2.4

million ticks per second); it gives an idea of how "ticks" are a far more accurate unit for accurate measure of elapsed time. The positional uncertainty (P_T) associated to this discrete time-stamping unit, when the probe is manipulated at 1000 mm/s, is equal to:

$$P_T = v \cdot t = 1000 \cdot \left(\frac{1}{2.4 \cdot 10^6}\right) = 0.00042\ mm \qquad \text{Eq. 7.6}$$

Therefore the latter uncertainty is negligible and the positional uncertainty remains equal to P_U. Since it is an order of magnitude smaller than 0.5 mm (the positional resolution reported for the robotic USL system), the use of hardware able to output positional feedback at a rate of 2000 Hz (8 times better than the current rate offered by KUKA controllers) does not constitute a great advantage against the current TWI system, where a very good spatial resolution is achieved through accurate time-stamping and purposely designed fast C++ parsing code.

It might be argued that the time-stamping approach relies on assuming a constant speed of the robot end-effector and in reality the speed may be variable. However 1000 mm/s is already an exaggerated speed, for the usual needs of NDT systems. Real systems move slower, with speeds spanning within a lower range. For this reason, the positional uncertainty is even smaller than the one estimated in Eq. 7.5.

Moreover, the USL system is designed for single-piezoelectric-element ultrasonic probes that require to move fast to increase the extension of surface inspected over time. Phased array probes cannot be moved as fast as single element probes for physical limitations; it is necessary to slow down the speed to allow the collection of all the A-Scans composing each B-Scan in a linear array, without distorting the B-Scan too much in the direction of motion (the passive axis of the transducer). Despite the fact ultrasonic phased array probes are moved at inferior speed, the inspection of a given surface takes less time than the traditional single element inspection. This is because the electronic sweeping (the progressive firing of piezoelectric elements in an array) is faster than the mechanical movement required with single element probes.

Nevertheless, given that the current trend is to use robots able to output the feedback positional information at higher rates, witnessed by the USL case, there is a possible path of investigation for future research. It would consist in getting signals directly from the encoder of each axis in the robot arm. The possibility to use robot manipulators equipped with external encoders can be investigated. A purposely designed piece of hardware with the encoders' data as inputs could provide the Cartesian pose of the robot through the forward kinematics, virtually instantaneously at the time it is required to encode each piece of NDT information.

7.5. Wrapping-up

This chapter presented some applications of the IntACom prototype robotic NDT system. The description of some experimental tests, carried out throughout the different phases of the system integration, explained relevant practicalities to consider during the employment of the IntACom system for the inspection of complex shape high value aerospace parts. Although many tests were carried out during the development of the IntACom system, this chapter reports only those experimental tests that engaged the author of this thesis and were relevant to optimize the performances of the system.

The developed water jet nozzle prototype allowed the achievement of the milestones required by the industrial partners of IntACom. Good performances were obtained with short coupling water columns ($\simeq$ 10mm long). Since narrow water jet gaps can only be used for non-highly curved surfaces, future work will regard the improvement of the design of the water jet nozzle to maximize the length of the free water column.

The possibility to dynamically adjust the length of the inspection gates was implemented to ease the scan of parts with variable thickness. Changeable thickness sometimes produces strongly attenuated back-wall reflections. The length of the gates is adjusted for each A-Scan to follow the position of the back-wall. The position of the back-wall is monitored through the CAD model of the part imported into the IntACom software.

Using the GeoLink approach provided by the KUKA RoboTeam add-on software, effective ultrasonic through-transmission inspections were enabled. The GeoLink method is designed to maintain the relative position between the Slave robot tool central point (TCP) and the Master robot TCP constant. The method proved suitable to enable through-transmission inspections of parts with constant thickness.

Through-transmission inspections for parts with variable thickness can be enabled through exploiting one of key advantages provided by the external control approach described by this thesis in Chapter 3. The expansion of RoboNDT and its integration with the IntACom platform could enable the generation of tool-paths for through-transmission inspections. The relative command text files, outputted by RoboNDT, would consist in a number of lines comprising two packets of coordinates (the first to be streamed to the

Master robot controller and the second to the Slave controller). This would pave the way to the implementation of external control for two or more robots running simultaneously.

The benchmarking of the IntACom system is not trivial. A quantitative evaluation was carried out by the author of this thesis. The chapter offered a benchmarking of the new system, comparing its acquisition rates (in terms of squared metres per hour) to the acquisition rates of other systems. The resulting good performances of the IntACom system are the result of the exploitation of new solutions, not available in other systems. The IntACom system is able to achieve scanning rates up to about 24m^2/hour. This is 2.8 times faster than the GE 3-DoF scanner deploying phased-array inspection and 6.7 times faster than commercial gantry systems deploying traditional single-element inspection. The carefully designed data acquisition and encoding IntACom algorithms proved to be effective and able to surpass the performances of current systems that use traditional approaches.

References

[1] B. Fitzsimons, "Sounds ultra advanced," *Aerospace Testing International,* vol. April, pp. 50-56, 2015.

[2] KUKA, "KUKA Roboteam 1.0 manual," 2012.

[3] USL, "A new NDT development using high speed robots," *NDT.net e-Journal,* vol. 19, 2013.

8. CONCLUSIONS AND FUTURE WORK

8.1. Conclusions

The increasing deployment of composite materials in aerospace applications demands a high integrity and traceability of NDT measurements, combined with a rapid throughput of data. Modern components present challenging shapes and geometries for inspection and traditional manual inspection approaches produce a time-consuming bottleneck in the industrial production environment.

Modern Computer-Aided Design (CAD) is used extensively in composite component design, manufacture and inspection. Whereas it was once necessary to construct large items from many smaller parts, Computer-Aided Manufacturing (CAM) now allows these large items to be produced easily either through traditional subtractive approaches, or with more recent additive manufacturing processes. As a result, large components with complex geometries have become very common in modern structures as exemplified by the aerospace industry. To meet increasing pressures to drive down costs and production time, whilst maintaining or improving quality, it is necessary to overcome the current NDT bottlenecks encountered using either manual or Cartesian axis scanning systems.

This thesis investigated several aspects of the introduction of automation within the inspection process of complex parts. Despite several efforts undertaken in recent years, key areas of research remained unexplored and they had to be investigated. The key challenges, found by the author of the present thesis, include the development of suitable path-planning strategies for robot manipulators interfaced with NDT acquisition instruments, high speed NDT data collection, integration of surface metrology measurements, and overall visualisation of results in a user friendly fashion.

The use of six-axis robots for product inspection and non-destructive testing systems is the central investigation of this thesis. All the relevant aspects of how a 6-axis serial manipulators work, the way they moves and how they are programmed were examined.

A wide range of different robot control strategies for the robotic NDT was discussed, through comparing relevant experimental data. A new method, established for external real-time control of robot manipulators, was presented. The viability of the new control method was proved by experimental testing. Accuracy measurements were obtained through comparing the target positions with the robot feedback positions and through laser tracking of the robot end-effector. The maximum position and orientation errors, observed with the robots moving at 50mm/s, were respectively within 2mm and within 1 degree, over a working envelope of $3m^3$. The main source of error was found in the inherent inaccuracy of the tool central point definition.

The external control paves the way to key inherent advantages to integrate robot manipulators with NDT acquisition systems for automated NDT inspections. The challenges related to the integration of an ultrasonic phased-array instrument were explained. An effective and reliable way to encode the NDT data through the interpolated robot feedback positions was implemented.

Extensive research was embraced to investigate the possibility to overcome the deficiencies of the commercial off-line path-planning software. A MATLAB based path-planning toolbox was developed to enable the exploitation of the externally controlled robot manipulators. Following the promising indications given by the initial experimental results, the path-planning toolbox inspired the commencement of a wider project for the development of a user-friendly interface targeted to robotic NDT usable by technicians.

The RoboNDT software was developed. It enables the possibility of establishing seamless robotic NDT software platforms. RoboNDT is intended to transform the robot path-planning problem into an easy step of the inspection process. The software encapsulates novel path-planning algorithms, the outcomes of the research and it is shaped to overcome practical limitations of current OLP software. Validation experiments were carried out on a large composite winglet sample and on a titanium fan blade. The accuracy of the RoboNDT tool-paths was evaluated through comparing the command coordinates and the feedback coordinates received from the robot controllers. The maximum position error was equal to 2.70mm and the maximum orientation error was equal to 0.29 degrees, for the fastest tested speed of 300mm/s and a scanned surface of 1.6m^2.

The experimental data demonstrated that the path errors achievable through externally controlled robots are lower than the errors given by commercially available path planning software. The variability of the standoff was within 10 mm for the tool-paths created with the commercial software and within 4.5 mm for the RoboNDT tool-paths, over the largest scanned area of 1.6m^2.

The outputs of this research were used to support a 3-year industrial project, called IntACom and led by TWI on behalf of major aerospace sponsors. The objective of the project was to achieve a fourfold increase in the inspection throughput of complex aerospace components. The result is a demonstrator system, currently in use at TWI Technology Centre, which is capable of inspecting complex geometries with high throughput.

The system uses integrated software that intuitively steers the user through the inspection process. A simple-to-use graphical user interface (GUI) was developed to control all aspects of the robotic inspection, from initial loading of part data, through scanning of the part to data analysis. Many unique features, including the real time B-scan for optimization of PAUT settings and the external control of the robotic manipulators to allow returning to points of interest, increase the usefulness of the inspection process.

The employment of the IntACom system for the inspection of complex shape high value aerospace parts was presented. The final benchmarking of the system demonstrated how the system is effective and able to surpass the performances of current systems that use traditional approaches. The IntACom system proved to be 2.8 times faster than the GE 3-DoF scanner deploying phased-array inspection and 6.7 times faster than commercial gantry systems deploying traditional single-element inspection. The ratio between the maximum theoretical FPS and the real FPS was considered as a quantitative measure of the efficiency of the data acquisition software. This ratio is respectively equal to 0.23 for the GE 3-DoF scanner and 0.61 (average) for the IntACom system, resulting in the IntACom data acquisition being substantially more effective.

8.2. Future work

The research inspired several ideas for future developments.

The laser tracking of the robot end-effector highlighted the inherent inaccuracy of the tool central point as main source of error. An algorithm should be developed to try to use the laser tracker data to calibrate the centre of the laser reflector and consequently the TCP of the robot tool. The development of a method based on optical measurements, for correcting the calibration of the tool, would allow a consistent reduction of the error measured by the laser tracker and a good agreement with the positional deviation obtained from the robot feedback coordinates.

Future work should exploit the external control approach for KUKA robots to achieve higher levels of automation and facilitate the integration of metrology for part position assessment and surface mapping.

Future development for the RoboNDT software includes the optimization of the performances of the background code and the implementation of simulation capabilities. Long term objectives comprise commercial exploitations. RoboNDT is also suitable to support research and/or teaching activities in academic institutions. The idea to extend the software to support mobile robot manipulators (6-DoF manipulators mounted on tracks, rotary tables, etc.) and sister platforms, like mobile robots, makes RoboNDT a very promising application.

The IntACom system will see future developments. The main areas of future development regard the implementation of real-time dynamic UT adaptation for probes scanning highly curved surfaces, support for UT through-transmission inspection of samples with variable thickness and employment of metrology.

Additional investigations may concern the automation of defect detection, the development of suitable approaches to manage high data volumes and the enhancement of conventional phased array and FMC data visualization.

APPENDICES

A1 – Implementation of external control for KRC2 robots

The C# code used for the system presented in Section 3.5 is a multi-threading code receiving communications from the robot and mirroring the IPOC time-stamp of the robot feedback packets.

Before developing the code, it was investigated how to make the KUKA robot language (KRL) module able to receive coordinates via the RSI Ethernet interface and move the robotic arm according to them. To do this the KRL module used for the system described in Section 3.4.2 was modified, adding a group of commands suitable to enable the robot controller to receive data from the external computer.

The lines of KRL code concerning the robot movements were replaced with the RSI command "ST_SKIPSENS". The RSI User Manual for KRC2 controllers [1] explains that the "ST_SKIPSENS" command allows a pure external guided movement with coordinates coming from a pre-defined path or from a real-time path correction, using a sensor (e.g. a vision system). The resulting KRL module is given herein and each command is commented.

```
KRL MODULE: EXT_CONTROL.SRC

DEF Ext_control( )
; ============================================
; This KRL module uses RSI in order to establish an Ethernet communication with the
; external PC.
; The task of the program is executing a raster scan of a work piece, receiving the
; motion coordinates from the external computer.
;
; Author name: Carmelo Mineo
; Last update: 5 Sep 2012
; ============================================

;-------------------------
; Declare variables
;-------------------------

DECL BOOL comms  ; Indicate in the connection status
DECL POS safety, safetyone, safetytwo, safetythree, angle
SIGNAL input $in(1) to $in(4)

;FOLD INI
  ;FOLD BASISTECH INI
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
  ;Make your modifications here

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

;-----------------------------------------------
; Set all the digital outputs to FALSE
;-----------------------------------------------
```

```
$out(1)=FALSE
$out(2)=FALSE
$out(3)=FALSE
$out(4)=FALSE
$out(5)=FALSE
$out(6)=FALSE
$out(7)=FALSE
$out(8)=FALSE

;---------------------------------------------
; Go to HOME position
;---------------------------------------------

;FOLD PTP HOME Vel=100 % DEFAULT;%{PE}%R 5.5.32,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 
2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART=FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS(#PTP_PARAMS,100)
$H_POS=XHOME
PTP XHOME
;ENDFOLD

;*******************************************************
;       DEFINE BASE AND TOOL FOR PROGRAM HERE
;*******************************************************

$base = base_data[15] ; 15=blind hole
$tool = tool_data[13]

;-------------------------------------------------------------------
;        CREATING THE RSI OBJECT FOR ETHERNET
;-------------------------------------------------------------------
; This part of the program creates a RSI Object ST_Ethernet, using the object 
configuration located in
; .../INIT/RSI_External_control.xml

;--------------------------------------------------------------------------------
err = ST_ETHERNET(hEthernet,0,"RSI_External_control.xml")
;--------------------------------------------------------------------------------

IF (err <> #RSIOK) THEN
  HALT
ENDIF

err = ST_SETPARAM(hEthernet,eERXmaxLatePackages,65000)   ; After "value" too late
                                                          ; packages the robot stops
; err = ST_SETPARAM(hEthernet,eERXmaxLateInPercent,10)   ; RSI WARNING if the
                                                          ; limit reached
; err = ST_SETPARAM(hEthernet,eERXmaxFieldOfView,1000)   ; Reset every 'value’
                                                          ; statistics.
; err = ST_SETPARAM(hEthernet, eERXFastCycle, 1) ; FALSE: Time to answer 11ms
                                                 ; TRUE: Fast cycle: answer <2ms
; err = ST_SETPARAM(hEthernet, eERXerrorFlag, 1)  ; $FLAG[1] will be set in case
                                                  ; of errors
err = ST_SETPARAM(hEthernet,eERXprecision,2)   ; Sets transmitted values to have
                                               ; 2 decimal places

;----------------------------------------------------------
; Configure the data coming out from the KRC
;----------------------------------------------------------

;FOLD RSI-Objects to link in ST_Ethernet

; read $OUT[5-7]
err = ST_DIGOUT(hDout1,0,5,0,0)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_DIGOUT(hDout2,0,6,0,0)
IF (err <> #RSIOK) THEN
  HALT
```

```
ENDIF
err = ST_DIGOUT(hDout3,0,7,0,0)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

err = ST_NEWLINK(hDout1,1,hEthernet,1)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hDout2,1,hEthernet,2)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hDout3,1,hEthernet,3)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

;ENDFOLD

;---------------------------------------------------------------------------
; Configure the data coming in from the External Computer
;---------------------------------------------------------------------------

;FOLD RSI-Objects to link out of ST_Ethernet

; Link RKorr to correction on path
err = ST_PATHCORR(hPath,0)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

err = ST_SETPARAM(hPath,1,-300)  ; Lower limit of  x-axis
err = ST_SETPARAM(hPath,2,-300)  ; Lower limit of  y-axis
err = ST_SETPARAM(hPath,3,-300)  ; Lower limit of  z-axis
err = ST_SETPARAM(hPath,4,-30)   ; Lower limit of  A-axis
err = ST_SETPARAM(hPath,5,-30)   ; Lower limit of  B-axis
err = ST_SETPARAM(hPath,6,-30)   ; Lower limit of  C-axis
err = ST_SETPARAM(hPath,7,300)   ; Upper limit of  x-axis
err = ST_SETPARAM(hPath,8,300)   ; Upper limit of  y-axis
err = ST_SETPARAM(hPath,9,300)   ; Upper limit of  z-axis
err = ST_SETPARAM(hPath,10,30)   ; Upper limit of  A-axis
err = ST_SETPARAM(hPath,11,30)   ; Upper limit of  B-axis
err = ST_SETPARAM(hPath,12,30)   ; Upper limit of  C-axis

err = ST_NEWLINK(hEthernet,1,hPath,1)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hEthernet,2,hPath,2)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hEthernet,3,hPath,3)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hEthernet,4,hPath,4)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hEthernet,5,hPath,5)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_NEWLINK(hEthernet,6,hPath,6)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

; Show RKorr on $SEN_PREA[1-6]
```

```
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,1,1)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,2,2)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,3,3)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,4,4)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,5,5)
IF (err <> #RSIOK) THEN
  HALT
ENDIF
err = ST_MAP2SEN_PREA(hmap,0,hEthernet,6,6)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

err = ST_BKEAKMOVE(hmap,0,hEthernet,7)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

;ENDFOLD

comms=TRUE      ; Indicate the connection is established

$out[1]=TRUE
$out[5]=TRUE

;------------------    RSI created     -----------------------

safetyone = {POS: X 35, Y 50, Z 100,A 90,B 0,C 180}     ; First approaching position
safetytwo = {POS: X 35, Y 50, Z 50,A 90,B 0,C 180}      ; Second approaching position
angle = {POS: X 19,Y 50,Z 50,A 90,B 0,C -130}           ; Position for tilting the
                                                        ; probe and removing any
                                                        ; trace of air

PTP safetyone

$VEL.CP = 0.2
LIN angle
$VEL.CP = 0.06
LIN safetytwo
WAIT SEC 1

;   ------- DEFINE start position (origin) ------

start_pos = {POS: X 0,Y 0,Z 0,A 90,B 0,C 180}    ; Origin of the base reference
                                                 ; system

;   -------   Start moving   -------

$VEL.CP=0.1
LIN start_pos

WAIT FOR input <> 0                     ; Waiting for the operator to confirm that
                                        ; the scan can start
WAIT SEC 5

;----------- Start motion and data acquisition ---------------

$out[6]=TRUE
$out[2]=TRUE
```

```
;--- Setting the coordinate system for the RSI motion ---

err = ST_ON1(#BASE,0)
IF (err <> #RSIOK) THEN
  HALT
ENDIF

; *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*
ST_SKIPSENS() ;Hold on - until RSI-Break reason occur
; *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

err = ST_OFF()

$out(6)=FALSE    ; The KRC say to the External computer the scan is finished
$out(4)=FALSE
$out(5)=FALSE
$out(1)=FALSE

SHAKE()          ; Calling the shaking sub-program to remove water from the probe
PTP xHOME        ; Return to HOME

END


;*********************************************************************
; SUB-PROGRAM TO SHAKE THE PROBE AND REMOVE THE WATER
;*********************************************************************

DEF shake()

$VEL.CP=0.06
LIN {POS: X 35,Y 50,Z 127.17,A -90,B 0,C 180}
LIN {POS: X 35,Y 50,Z 274,A -100,B -4.5,C 121}
PTP_REL {A5 -10}
PTP_REL {A5 10}
PTP_REL {A5 -20}
PTP_REL {A5 20}
PTP_REL {A5 -30}
PTP_REL {A5 30}
PTP_REL {A5 -50}
PTP_REL {A5 50}
PTP_REL {A5 -70}
PTP_REL {A5 70}
PTP_REL {A5 -70}

END
```

The C# server application sends an XML string to the KRC for each interpolation cycle of the robot controller. The syntax of the string is derived from the following template:

```
<Sen Type="ImFree">
   <EStr>External control activated</EStr>
   <RKorr X="0.00" Y="0.00" Z="0.00" A="0.00" B="0.00" C="0.00" />
   <Break>0</Break>
   <IPOC></IPOC>
</Sen>
```

The XML string includes all the target coordinates for the robot to move to the next point. Basically, what the external computer does is to find the location of these target values inside the XML message, erasing the old values and replacing with new ones before it sends the XML message to the robot controller. The server application has to replace the values on the line that starts with "<RKorr" and ends with " />". If all the coordinates of a scanning path are written in a text file, it is easy to programme the external server application to read a new packet of coordinate from the file and replace the old packet in the XML message, before sending it to the KRC.

In order to have a simple set of coordinate and try the feasibility of this idea, a simple MATLAB script was written to output a text file of coordinates: “coordinates.txt”. An example group of ten consecutive positions is shown below.

```
…
…
X="0.00" Y="109.03" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.00" Y="109.30" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.00" Y="109.57" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.00" Y="109.83" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.27" Y="109.83" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.50" Y="109.83" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.50" Y="109.57" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.50" Y="109.30" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.50" Y="109.03" Z="0.00" A="0.00" B="0.00" C="0.00"
X="0.50" Y="108.77" Z="0.00" A="0.00" B="0.00" C="0.00"
…
…
```

Focusing back on the C# code of the external server application, it was developed through modifying the code used for the system described in Section 3.4.2. The programme thread that manages the mirroring of the IPOC time-stamp back to the robot was put in charge for the additional task of reading one string of coordinates from the text file “coordinates.txt” and copying it in the outgoing XML string

A consecutive string of coordinates is read during each cycle of the thread so that the robotic arm can go forward in the scanning path. When the end of “coordinates.txt” is reached and there are no other lines available to be read, the code continues to mirror the IPOC time-stamp back to the robot and sets to “1” the “Break” value of the XML string. The “<Break>0</Break>” line was intentionally inserted in the XML template in order to create a logic command to trigger the end of the external control. When the “Break” value is changed from “0” (false) to “1” (true) the KRC break the RSI command “ST_SKIPSENS” and the robot continues with the next command. To allow the BREAK signal to terminate the external control, the RSI command “ST_BREAKMOVE” is to be used in the RSI-Ethernet configuration. This command creates a RSI object able to break a RSI movement command like “ST_SKIPSENS” [1]. The object's input changing to TRUE invokes a KRL interrupt routine. The interrupt routine cancels the actual movement and continues immediately with the next command.

Lastly, it was necessary to configure the RSI software to transmit its positional data and digital output states over a TCP/IP connection and expect the new coordinates and the BKEAK signal in return. Data is sent in XML file format and the structure of the packet is dictated by an XML file located in the robot controller. This file is configured, according to a particular template [1]. The configuration file suitable to enable external control has the following structure:

```
<ROOT>
   <CONFIG>
      <IP_NUMBER>192.168.1.30</IP_NUMBER>  <!- IP Number of the socket !->
      <PORT>6008</PORT>                    <!- Port Number of the socket !->
      <PROTOCOL>TCP</PROTOCOL>             <!- TCP or UDP, Protocol of the socket !->
      <SENTYPE>ImFree</SENTYPE>            <!- The name of your external system !->
      <PROTOCOLLENGTH>Off</PROTOCOLLENGTH> <!- On or Off, Send the length in bytes !->
      <ONLYSEND>FALSE</ONLYSEND>           <!- The client does not expect an answer !->

   </CONFIG>

   <SEND>
      <ELEMENTS>
         <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
         <ELEMENT TAG="Digout.out5" TYPE="BOOL" INDX="1" UNIT="0" />
         <ELEMENT TAG="Digout.out6" TYPE="BOOL" INDX="2" UNIT="0" />
         <ELEMENT TAG="Digout.out7" TYPE="BOOL" INDX="3" UNIT="0" />
      </ELEMENTS>
   </SEND>

  <RECEIVE>
      <ELEMENTS>
         <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" UNIT="0" />
         <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" UNIT="1" HOLDON="1" />
         <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" UNIT="1" HOLDON="1" />
         <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" UNIT="1" HOLDON="1" />
         <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" UNIT="0" HOLDON="1" />
         <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" UNIT="0" HOLDON="1" />
         <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" UNIT="0" HOLDON="1" />
         <ELEMENT TAG="Break" TYPE="BOOL" INDX="7" UNIT="0" HOLDON="1" />
      </ELEMENTS>
   </RECEIVE>
</ROOT>
```

In the *<config>* section of the file the IP address of the external PC is specified. The *<port>* line must specify the same port that the host PC is listening on. Under *<protocol>* either TCP or UDP can be defined. The *<sentype>* line defines the name of the server to which you are communicating and is used for identification purposes. This can set to any name however it must match the name given when configuring the external system. The robot will check every data packet it receives for this identification. If *<protocollength>* is set to ON then the robot will transmit the byte length of each packet before it is sent. One- or two-way communication is defined by changing the line *<onlysend>* to TRUE or FALSE. The *<send>* section defines what information is transmitted. In this configuration for this project, only the XYZ coordinates, ABC angles and digital output states are transmitted. However, this section can be extended to include a much wider range of parameters including the angles of each of the six axes of the robot and the currents in each motor. Since the robot is expecting to receive data back, they were specified in the <receive> section.

The line containing the Boolean “Break” signal was originally not present in the XML template provided by KUKA in the RSI User Manual [1]. It was intentionally inserted to enable the stop of the externally controlled motion.

A2 – Ensuring a reliable connection

Although the execution of the tool-path through the external control proved the possibility to use the RSI software add-on to establish a new approach to solve the path-planning problem, the robotic arm occasionally stopped its fluid motion and jumped to another position of the path, some millimetres far from the stop point. This highlights the sudden loss of connection between the KRC and the external computer and the successive recovery of data exchange. The interruption of the connection during small instants has a bad influence on the resultant C-scan image. Fig. 1 shows that the blind hole of the metal block can be seen, but the general quality of the scan is poor due to inaccurate matching between the ultrasound data and the robot's positional data.

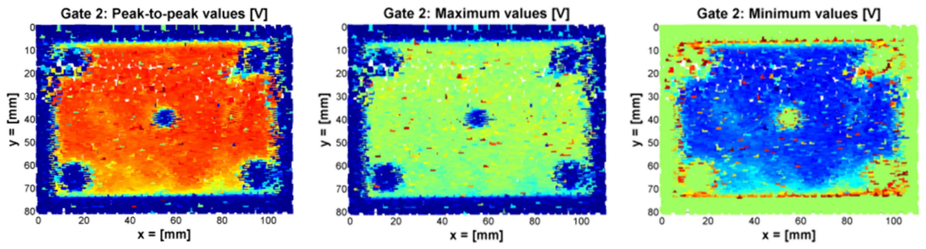
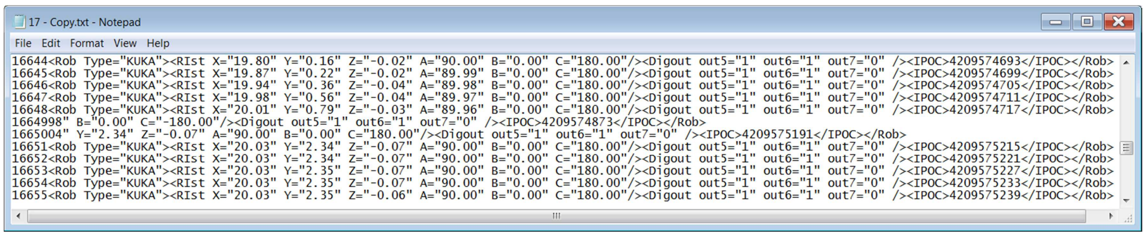


Fig. 1 – C-scan obtained with the externally controlled motion. Peak-to-peak value (left), maximum value (middle) and peak to peak value (right) of the second wave.

In the system described in Section 3.4.2 and in Figure 3.17, it had been already found that occasional errors, regarding the mirroring of the IPOC, occurred during the scan. Since the robot did not receive back its time-stamp, it counted the packet as late and stopped running for some instants. Investigating the problem, it was found that during these times the receive buffer on the PC side – which usually would only be filled with one packet during a read operation – had more than one packet in it. It would overflow, receiving perhaps three and a half packets. This meant that some packets did not have their time-stamp mirrored to the robot and that the next time a read command was executed it would receive the remaining half of the partly read packet. A portion of the received data, where an error an error of this kind is clearly present, is shown in Fig. 2.



```
16644<Rob Type="KUKA"><RIst X="19.80" Y="0.16" Z="-0.02" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574693</IPOC></Rob>
16645<Rob Type="KUKA"><RIst X="19.87" Y="0.22" Z="-0.02" A="89.99" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574699</IPOC></Rob>
16646<Rob Type="KUKA"><RIst X="19.94" Y="0.36" Z="-0.04" A="89.98" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574705</IPOC></Rob>
16647<Rob Type="KUKA"><RIst X="19.98" Y="0.56" Z="-0.04" A="89.97" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574711</IPOC></Rob>
16648<Rob Type="KUKA"><RIst X="20.01" Y="0.79" Z="-0.03" A="89.96" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574717</IPOC></Rob>
1664998" B="0.00" C="-180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209574873</IPOC></Rob>
1665004" Y="2.34" Z="-0.07" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575191</IPOC></Rob>
16651<Rob Type="KUKA"><RIst X="20.03" Y="2.34" Z="-0.07" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575215</IPOC></Rob>
16652<Rob Type="KUKA"><RIst X="20.03" Y="2.34" Z="-0.07" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575221</IPOC></Rob>
16653<Rob Type="KUKA"><RIst X="20.03" Y="2.35" Z="-0.07" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575227</IPOC></Rob>
16654<Rob Type="KUKA"><RIst X="20.03" Y="2.35" Z="-0.07" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575233</IPOC></Rob>
16655<Rob Type="KUKA"><RIst X="20.03" Y="2.35" Z="-0.06" A="90.00" B="0.00" C="180.00"/><Digout out5="1" out6="1" out7="0" /><IPOC>4209575239</IPOC></Rob>
```

Fig. 2 – Example of error reported in the received data flux.

Since the robot motion is now controlled by the external server application rather than by the KRL module, the error in the communication is translated in a sudden discontinuous movement every time the problem occurs. It was initially unclear what was causing this problem. It was judged that it was either due to the robot transmitting more than one packet simultaneously or due to the host PC pausing on a receive command for more than only cycle, therefore receiving several packets.

As a first attempt to stop the robot manipulator controlled via RSI from suddenly jumping between two positions and to avoid loss of ultrasonic data in some areas, a small change was applied to the "listening thread" of the C# code. This was done to make the server application able to halt the robot movement when a connection problem occurs and wait until the connection is restored. This behaviour was introduced using the difference between two consecutive IPOCs as a valid parameter to check the connection status.

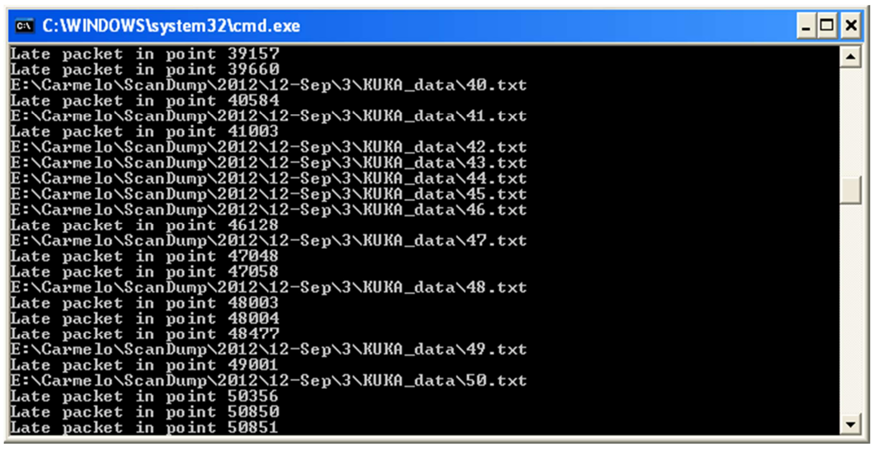
When the connection is established with a server and it is properly working, the robot transmits data packets according to the XML configuration file in a 12ms interpolation cycle. The IPOC is appended to the end of every packet and it is a time-stamp for the packet. An increase of 6 in the IPOC number equates to a 12ms time difference, it is one interpolation cycle of the RSI software. A difference greater than 6 between two consecutive time-stamps means that one or more packets are not being sent by the KRC or not received by the server. A new piece of code was inserted to alternatively store the received IPOC in two variables according to odd or even cycles being processed. This way the external server always knows the current and the previous IPOC and it can calculate the difference.

If the "listening thread" evaluates an IPOC difference not equal to 6 a warning is printed to the application console window. The difference variable *diff* is transferred to the thread responsible for sending coordinates and mirroring the current IPOC time-stamp. This thread was programmed to stop sending new coordinates when *diff* is not equal to 6. As a result, the robot manipulator halts for the short intervals of time with poor connection quality and the movement is resumed as soon as the robust connection is restored.

To prove the performance of the system controlled by the modified code, the metal block in Figure 3.11a was scanned and the resulting C-scans are shown in Figure 3.20. In order to allow a comparison with the C-scan images given in Fig. 1, the same command file ("coordinates.txt") was used. Therefore, the component was scanned with speed of 21 mm/sec and 40923 ultrasonic signals were collected in a total time of 832 seconds (13.8 minutes). The system was able to scan the metal block with an average resolution of 2.3

scans/mm. This means the actual scans/time ratio of the system being equal to 2951 scans/min (around 34 times faster than the first system described in Section 3.4.1 – see Figure 3.7 and Figure 3.10).

During the scan, the C# console window displayed a warning message ("Late packet in *POINT_NUMBER*") every time a difference not equal to 6 was found between two consecutive IPOCs. Fig. 3 shows a screen shot of the console window reporting several warnings among the messages related to the data savings.



```
Late packet in point 39157
Late packet in point 39660
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\40.txt
Late packet in point 40584
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\41.txt
Late packet in point 41003
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\42.txt
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\43.txt
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\44.txt
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\45.txt
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\46.txt
Late packet in point 46128
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\47.txt
Late packet in point 47048
Late packet in point 47058
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\48.txt
Late packet in point 48003
Late packet in point 48004
Late packet in point 48477
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\49.txt
Late packet in point 49001
E:\Carmelo\ScanDump\2012\12-Sep\3\KUKA_data\50.txt
Late packet in point 50356
Late packet in point 50850
Late packet in point 50851
```

Fig. 3 - Example of warning message reported in the C# console.

A total of 72 "Late packet" messages were counted at the end of the scan, meaning that the new C# server application was able to avoid at least the same number of undesirable jumps of the robotic arm.

Even if this seems to sort out the problem, the motion of the robotic arm was not fluent and continuous during the scan, because of the type of fix that was implemented. The several interruptions in the Ethernet communication forced the arm to halt its movement for short but frequent intervals of time.

The problem was further investigated. Several trials in changing the process priority of the C# application and of the MATLAB script in the external computer, using the "Task Manager" of "Windows XP", were attempted and a change in performance was noticed. It was observed that whenever the quality of the movement of the robot improved, the number of ultrasonic signals that MATLAB was able to register during the scan dropped sensibly.

It was decided to create a new C# code step by step from scratch, observing the behaviour of the system and the numbers of communication errors after each step.

- Step 1

A very simple program was generated only to receive the RSI string and to mirror the IPOC number. The code had three threads: the main thread, the listening thread and the mirroring thread. No task for saving data and reading a file of coordinates was implemented in the code. The resulting program was used for maintaining the connection alive for about 13.8 minutes, receiving the incoming XML strings and sending their IPOC to the KRC, without any update of the coordinates so that the robotic arm was firm during the whole observed time. A total of 10 communication errors occurred (0.8 errors/min), as showed in Fig. 4.

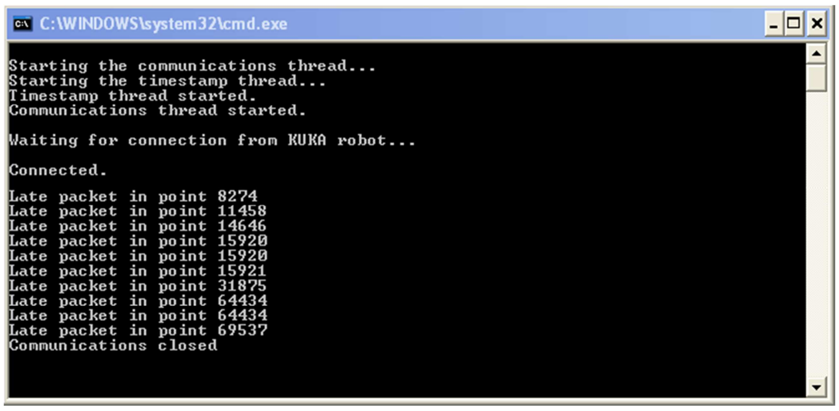

```
C:\WINDOWS\system32\cmd.exe

Starting the communications thread...
Starting the timestamp thread...
Timestamp thread started.
Communications thread started.

Waiting for connection from KUKA robot...

Connected.

Late packet in point 8274
Late packet in point 11458
Late packet in point 14646
Late packet in point 15920
Late packet in point 15920
Late packet in point 15921
Late packet in point 31875
Late packet in point 64434
Late packet in point 64434
Late packet in point 69537
Communications closed
```

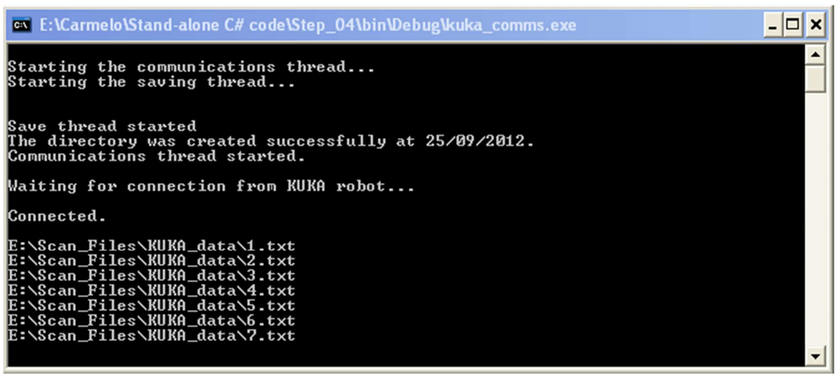
Fig. 4 - C# console at the end of Step 1.

- Step 2

A great improvement was achieved joining together the listening thread and the mirroring thread. Since the RSI works in a way that it is always waiting for receiving back the IPOC after delivering a XML string, it was deemed useful to introduce a relation of consequentiality between the action of listening and the action of mirroring. The simplest way for doing this was the elimination of the *mirroring* thread, appending its task at the end of the listening thread. The system was operated by the updated code for the same time duration (13.8 minutes) and no error occurred. Therefore the ability of reading the external text file of coordinates was inserted in the code, in order to control the motion of the robotic arm, and the performance remained stable without producing any errors of communication. The robot moved very smoothly with a continuous motion, during all the duration of the usual planar scan. Moreover, looking at the CPU usage of the host PC, it was observed that the updated code was much less resource-demanding than the previous one. An XML string is delivered only when the KRC is waiting for receiving back the IPOC from the external computer, otherwise the C# code remains waiting for a new incoming string. The CPU usage dropped down from about 50% to only 2-3%, during the code execution. This confirmed a consistent improvement in the structure of the code. Since a new thread was now dealing with both receiving and sending XML strings, it was called "*communications* thread".

- Step 3

A new thread for saving the robot's positional data was inserted in the code. After the introduction of this new part, the program is able to create a new directory at the beginning of its execution to save the received robot feedback positions. The process of writing the received data to a text file and storing it into a directory takes a few seconds so if this take were not given to a separate thread, it would interrupt the communications. This new thread was programmed in a way that it is executed only when an array of 1000 positional data is filled (by the *communications thread*) and is ready for being saved. Also after this modification in the driving code, the system was operated for the same time duration (13.8 minutes), in order to develop the usual planar raster tool-path, and no error occurred (see Fig. 5).


```
E:\Carmelo\Stand-alone C# code\Step_04\bin\Debug\kuka_comms.exe

Starting the communications thread...
Starting the saving thread...

Save thread started
The directory was created successfully at 25/09/2012.
Communications thread started.

Waiting for connection from KUKA robot...

Connected.

E:\Scan_Files\KUKA_data\1.txt
E:\Scan_Files\KUKA_data\2.txt
E:\Scan_Files\KUKA_data\3.txt
E:\Scan_Files\KUKA_data\4.txt
E:\Scan_Files\KUKA_data\5.txt
E:\Scan_Files\KUKA_data\6.txt
E:\Scan_Files\KUKA_data\7.txt
```
Fig. 5 - C# console at the end of Step 3.

This newly developed code fixed the communication problem. The cause of the issue was discovered being in the structure of the previous code and its solution allows a continuous and fluid motion of the robotic arm.

A3 – Configuration of external control for KRC4 robots

On the KUKA controller side, data exchange via Ethernet is implemented using the RSI object ETHERNET [1, 2]. Up to 64 inputs and outputs can be defined for ETHERNET. The signals at the inputs are sent to the sensor system (our external server computer). The data received from the sensor system are available at the outputs (Fig. 6).

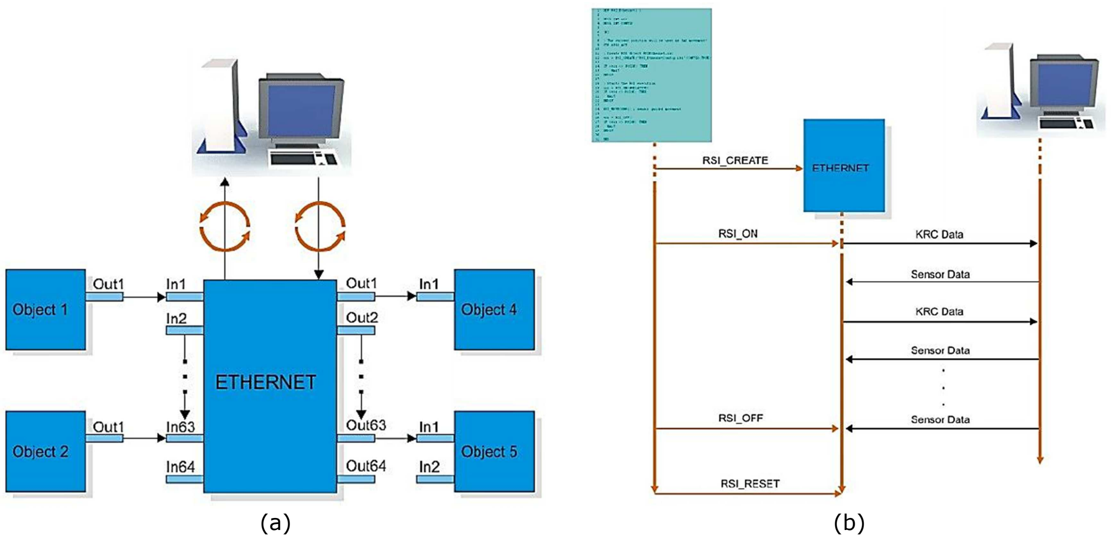


Fig. 6 – a) Data exchange via Ethernet (functional principle); b) Data exchange via Ethernet (sequence) [2].

When signal processing is activated on a KRC4 controller, a channel is prepared for sending data to the sensor system via the UDP/IP protocol. With signal processing activated, the ETHERNET object sends and receives a user-defined data set in XML format at the constant cycle time rate. The data set format must be configured in an XML file and the name of the XML file must be specified in the ETHERNET object. Unlike the old version for KRC2, the new RSI Software for KRC4 controllers facilitates the configuration of the RSI context inside the KRL program. It is done using RSI Visual user interface, an off-line programming tool for generating RSI configuration files. This useful interface generates three files when the signal flow configuration is saved:

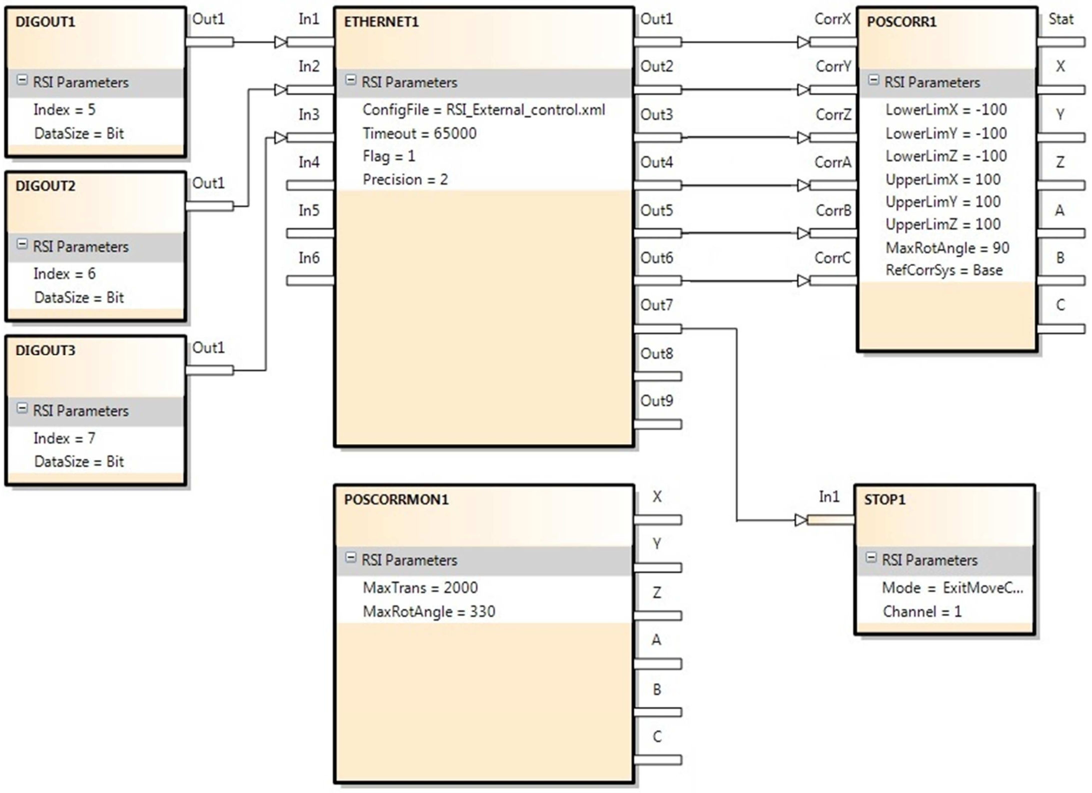
- <File name>.rsi: Signal flow configuration from RSI Visual
- <File name>.rsi.diagram: Signal flow layout from RSI Visual according to XML schema
- <File name>.rsi.xml: XML file for signal processing on the robot controller

The RSI, RSI.DIAGRAM and RSI.XML files form a unit that must be transferred to the robot controller together, to the target directory:

C:\KRC\ROBOTER\Config\User\Common\SensorInterface.

It is now possible to initialize the RSI context from a KRL module just calling the name of the RSI configuration file, without defining a long list of RSI objects and links inside the module itself.

The signal flow for our external control system was configured using RSI Visual user interface and the RSI schema is shown in Fig. 7. RSI objects that were dragged into the signal flow editor and linked together by means of the object-specific signal inputs and outputs.


DIGOUT1
RSI Parameters
Index = 5
DataSize = Bit

DIGOUT2
RSI Parameters
Index = 6
DataSize = Bit

DIGOUT3
RSI Parameters
Index = 7
DataSize = Bit

ETHERNET1
RSI Parameters
ConfigFile = RSI_External_control.xml
Timeout = 65000
Flag = 1
Precision = 2

POSCORR1
RSI Parameters
LowerLimX = -100
LowerLimY = -100
LowerLimZ = -100
UpperLimX = 100
UpperLimY = 100
UpperLimZ = 100
MaxRotAngle = 90
RefCorrSys = Base

POSCORRMON1
RSI Parameters
MaxTrans = 2000
MaxRotAngle = 330

STOP1
RSI Parameters
Mode = ExitMoveC...
Channel = 1

Fig. 7 – RSI signal flow schema for our external control system.

The main RSI object is ETHERNET1; it receives some inputs and provides some outputs. The inputs are variables read from the robot controller and sent to the external computer. The data received from the external computer are redirected to the outputs and become available for the robot controller. The status of three digital outputs of the robot controller is copied into three inputs of ETHERNET1 and they are used to trigger the external motion control operated by the server C# application that starts sending command coordinates. The six Cartesian coordinates received from the external computer are used by the POSCORR1 object, to influence the motion of the robotic arm. RSI_MOVECORR() is the command inside the KRL program that activates the sensor-guided motion. The robot is controlled purely by means of corrections on the basis of correction values of the RSI object POSCORR1. POSCORRMON1 is the object that limits the maximum overall Cartesian correction. The default maximum translation and rotation are equal to 6 mm and 6 degrees; they were set to 2000 mm and 330°. STOP1 is used to abort the sensor-guided motion when the Boolean variable available at Out7 changes from “false” to “true”.

Each RSI object has its own properties. POSCORR1 gives the possibility to specify the lower limit and the upper limit of the positional correction on each Cartesian axis, the maximum rotational angle and the coordinate reference system. ETHERNET1 was set to send robots current coordinates with a precision of 2 decimal places, to switch $FLAG[1] to “TRUE” in case of interrupted communication, to allow a maximum number of 65000 late packets and to send/receive XML strings according to the XML template “RSI_External_control.xml”. This XML format to exchange data via Ethernet, stored in the same directory of the other configuration files, has the following structure:

```
<ROOT>
   <CONFIG>
      <IP_NUMBER>10.1.1.110</IP_NUMBER>    <!-- IP Number of the socket !-->
      <PORT>49202</PORT>                   <!-- Port Number of the socket !-->
      <SENTYPE>ImFree</SENTYPE>            <!-- The name of your system !-->
      <ONLYSEND>FALSE</ONLYSEND>           <!-- TRUE means the client don't expect an answer !--
>
   </CONFIG>
   <SEND>
      <ELEMENTS>
         <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" />
         <ELEMENT TAG="Digout.o5" TYPE="BOOL" INDX="1" />
         <ELEMENT TAG="Digout.o6" TYPE="BOOL" INDX="2" />
         <ELEMENT TAG="Digout.o7" TYPE="BOOL" INDX="3" />
      </ELEMENTS>
   </SEND>
 <RECEIVE>
      <ELEMENTS>
         <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" HOLDON="1" />
         <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" HOLDON="1" />
         <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" HOLDON="1" />
         <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" HOLDON="1" />
         <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" HOLDON="1" />
         <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" HOLDON="1" />
         <ELEMENT TAG="Break" TYPE="BOOL" INDX="7" HOLDON="1" />
      </ELEMENTS>
   </RECEIVE>
</ROOT>
```

<CONFIG and </CONFIG> denote the configuration section of the connection parameters between the sensor system and the RSI interface. <SEND> and </SEND> include the description of the transmission structure. The signals from the RSI context that arrive at the inputs of the ETHERNET object and are sent to the sensor system are defined here.

From the specified XML template, the RSI context automatically creates the XML document that the robot controller transmits. The template was set to transmit the current coordinates of the robotic arm and the status of three digital outputs of the controller (Out 5, 6 and 7). The value of these Boolean variables indicates the external computer when the robot controller is ready to move the robotic arm according to externally commanded positions. The following string gives an example of the XML document transmitted by the robot controller:

```
<Rob Type="KUKA"><RIst X="0.25" Y="78.83" Z="0.00" A="0.00" B="0.00" C="-180.00"/>
<Digout o5="1" o6="1" o7="0"/>
<IPOC>498548</IPOC></Rob>
```

<RECEIVE and </RECEIVE> indicate the configuration of the reception structure. The signals received by the sensor system at the outputs of the ETHERNET object and forwarded to the robot controller in the RSI context are defined here.

The following is an example of the XML document generated by the external computer and received by the robot controller:

```
<Sen Type="ImFree">
   <RKorr X="0.25" Y="79.60" Z="0.00" A="0.00" B="0.00" C="-180.00" />
   <Break>0</Break>
   <IPOC>498548</IPOC>
</Sen>
```

The time-stamp with the keyword IPOC is checked by the robot controller. The data packet is only valid if the time stamp corresponds to the time stamp sent previously.

A4 – Performance optimization of external control code

Chapter 3 presented the development of the C# server application to enable an external computer ("external" to the computer running within the robot controller) to control the movement of a robot arm. The advantages of externally controlling the robot manipulators should be clear by now to the reader of this thesis.

The server application was initially developed for KRC2 robots at the University of Strathclyde, and successively updated to enable the UDP communication with KRC4 robots at TWI. Important reasons were identified that required the conversion of the C# code to a C++ application:

1. In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. The GC management carried out by C# applications is not perfect. The C++ programming language gives the programmer a better control over GC.
2. The implementation of the external control approach into the NDT data acquisition software developed for IntACom, the industrial project led by TWI (see Chapter 6), was deemed useful to address important industrial requirements. The code of the IntACom software, responsible for acquiring data from the NDT equipment, was developed in C++ programming language. This programming language proved the best among a group of other languages, for the obtainment of a reliable acquisition process enhanced by a GUI showing real-time feedback data. The integration of the external control required the use of the same language.
3. An Ethernet communication issue was observed when running the external control in the 4ms interpolation cycle mode. KRC4 robot controllers allow the possibility to run the Robot Sensor Interface in 12ms and 4ms mode; the 12ms mode is the only one available for KRC2 controllers. The external control proved very reliable when the Robot Sensor Interface runs in a 12ms interpolation cycle on both KRC2 and KRC4 robots. However, when running a KRC4 robot in 4ms interpolation mode, the external control was affected by sudden failure and the robot motion could not be resumed,

unless restarting the tool-path execution. It was decided to compare the performances of a C++ server application against the C# code.

Despite being the author's first approach to the C++ language, the conversion itself was not a difficult task. This work provided an opportunity to develop the C++ external control application in a modular way, easier to read and to maintain. Moreover, the resulting application was tested to get the execution time of its loop (an important performance parameter to know for critical pieces of real-time controlling software).

The performance of the code was investigated using diagnostic C++ functions, under the most challenging scenario given by a KRC4 controlling running the RSI context in 4ms interpolation cycles. The elapsed time was measured after each function of the script.

Fig. 8 shows the time it takes to each action to be completed within each loop of the server application. The elapsed times collected in 20,000 cycles were averaged.

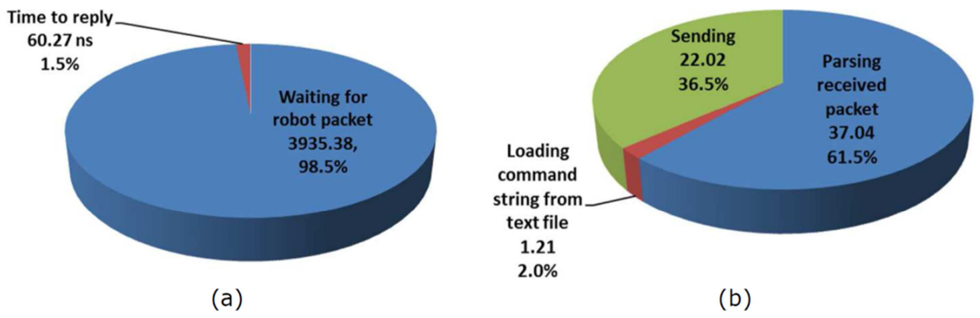


Fig. 8 - Overall loop duration (a) and actions' duration for replying to the robot controller (b).

Fig. 8a shows that the application spends most of the time waiting for the reception of an XML string from the robot. Then the string is parsed and a new packet of command coordinates with the mirrored IPOC number is sent back to the robot controller in 60.3ns.

The application is able to reply very quickly to the robot soon after a string is received, commanding a new target position and mirroring the IPOC number. The mirroring operation consists in reading the IPOC number contained in the received robot XML string and appending it to the outgoing command packet sent out by the external computer.

Despite of the good performance, the converted application did not solve the communication issues with the 4ms interpolation cycle mode. Since the sudden stop of the robot motion was demonstrated not to be caused by the external application, the problem was further investigated.

The reason for the misbehaving external control running in the 4ms interpolation cycle was found in the intrinsic way the RSI context is implemented in robot controllers [1, 2].

Section 3.6.1 showed the general implementation of RSI in the KUKA system architecture (see Figure 3.22). The tool-paths, which are externally commanded through the RSI in 12ms cycles, are executed by the robot drives after travelling through the first interpolation stage that runs. The resulting commands coordinates are filtered before undergoing a fine interpolation. Such interpolation is carried out by the second stage interpolator that produces coordinate packets every 4ms.

RSI with 4ms interpolation cycle, which is only supported by KRC4 robot controllers, allows sending the command positions directly to the fine interpolation stage. This permits a high robot feedback update rate, but the external control command coordinates have to be filtered before sending them to the robot. Failure in doing so results in the robot arm stopping the motion at the very first occurrence of an exceeding acceleration value in the joints.

A5 – Rotational matrices

Given the rotation angles A, B and C respectively around the Z, Y and X axis, the rotational matrices describing the rotations are:

$$\mathrm{R_X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{cosC} & -\mathrm{sinC} \\ 0 & \mathrm{sinC} & \mathrm{cosC} \end{bmatrix}$$

$$\mathrm{R_Y} = \begin{bmatrix} \mathrm{cosB} & 0 & \mathrm{sinB} \\ 0 & 0 & 0 \\ -\mathrm{sinB} & 0 & \mathrm{cosB} \end{bmatrix}$$

$$\mathrm{R_Z} = \begin{bmatrix} \mathrm{cosA} & -\mathrm{sinA} & 0 \\ \mathrm{sinA} & \mathrm{cosA} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where sinA, cosA, sinB, cosB, sinC and cosC are the sine and cosine values of the angles.

B1 – Main program of the path-planning MATLAB toolbox

```
MainProgram.m

close all; clear all; clc;

disp('- LOADING DATA:')
[Sample] = F_LoadSample('NameFile.stl');

Opt=[];
while isempty(Opt);
    Opt=input('Do you want to locate the sample within the cell? (yes (Y) or no (N)): ','s');
    if (Opt=='Y')
        disp('STARTING ALGORITHM ...');
        disp(' ');
[sample_in_MASTER_world,sample_in_SLAVE_world,sample_in_cell,Cell]=F_Int_LocateSample…
…(Sample,1);
    elseif (Opt=='N')
        disp('YOU DECIDED TO NOT LOCATE THE SAMPLE - LATEST POSITION INFORMATION WILL BE
USED.');
disp(' ');
[sample_in_MASTER_world,sample_in_SLAVE_world,sample_in_cell,Cell]=F_LocateSample…(Sample,'rotot
ra_matrices',1);
    else
        disp('ERROR - INPUT NOT RECOGNIZED. GIVE A VALID LETTER (Y for Yes, N for No)')
        Opt=[];
    end
end

Opt=[];
while isempty(Opt);
    disp(' ');
    Opt=input('Was the sample position calculated correctly? (yes (Y) or no (N)):',
    …'s');
    if (Opt=='Y')
        clc;
        disp('CONTINUING ...')
    elseif (Opt=='N')
        disp('PROGRAM ABORTED BY THE USER - SAMPLE POSITION WAS NOT CORRECT');
        Opt=[];
        break;
    else
        disp('ERROR - INPUT NOT RECOGNIZED. GIVE A VALID LETTER (Y for Yes, N for No)')
        Opt=[];
    end
end

if (Opt=='Y')
    clc;
    [Coef_surface,Surface,Edges,axis_lim,z_plane,err_max]=F_FindEntities(sample_in_cell,
    …0,1);
    F_ShowEntities(Surface,Edges,z_plane,3);
    indEdge=F_QueryEdge(Edges,z_plane,4);

    clc;

    [Lines,Step,Edges1,c]=F_FindPathLines(Coef_surface,Edges,indEdge,z_plane,axis_lim);
    [JoinedLines,boundary]=F_JoinLines(Coef_surface,Lines,Step,Edges1,c);
    [N] = F_FindNormals(Coef_surface,JoinedLines);

    F_ShowPath(JoinedLines,N,boundary,sample_in_cell,Cell,z_plane,0,3)
end
```

B2 – Setting up automatic execution of approach and retract motions

In addition to the command coordinates for the generated tool-paths, the output function *F_Output* described in Section 4.5 was developed to output the coordinates of suitable robot poses, to safely approach the starting point of the inspection path and retract to the HOME position at the end of the task. This appendix contains the explanation of the software solution to automatically set approach and retract motions.

The *LogFile* associated to each tool-path and outputted by the output function contains a head section followed by the coordinates of three robot poses. The first point is calculated by the toolbox to be 200 mm above the surface of interest along the perpendicular direction to the surface for the start point of the generated path. The second point coincides with the start point. Similarly to the first point, the third point is located 200 mm above the surface, along the normal direction of the endpoint of the scan. The following reports a copy of the *LogFile* text generated for the scan path shown in Figure 4.22.

```
LogFile_MASTER.txt
--------------------------------------------------------------------------------
MASTER COORDINATE FILE

Suitable for a system with 12ms interpolation cycle

Setting points:
X="1285.19" Y="-874.43" Z="749.96" A="-3.38" B="3.39" C="-33.94"
X="1275.38" Y="-985.89" Z="584.31" A="-3.38" B="3.39" C="-33.94"
X="656.88" Y="692.65" Z="676.15" A="12.98" B="-13.33" C="35.83"
```

The information contained in the *LogFile* was initially used by manually copying the packets of coordinates from it to the KRL module through the robot-teach pendant, in order to manually programme the approaching and the retracting movements. The following shows the KRL module script as it was after transferring the coordinates.

```
RSI_External_control.src
---------------------------------------------------------------------------------------
DEF RSI_External_control( )
; ============================================
; KLR Module for real-time UDP data exchange with C# server application
; Author: Carmelo Mineo
; Date: April 2013
; ============================================

; Declaration of KRL variables
DECL INT ret                    ; Return value for RSI commands
DECL INT CONTID                 ; ContainerID
DECL INT C                      ; Counter
DECL POS ApprStartPoint         ; Approaching point (200 mm above surface)
DECL POS StartPoint             ; Start point
DECL POS LeaveEndPoint          ; Leaving point (200 mm above surface)
DECL AXIS HOMELOCAL             ; Home position
;FOLD INI

;==================================================
; Settings
```

```
ApprStartPoint={POS: X 1285.19,Y -874.43,Z 749.96,A -3.38,B 3.39,C -33.94}
StartPoint={POS: X 1275.38,Y -985.89,Z 584.31,A -3.38,B 3.39,C -33.94}
LeaveEndPoint={POS: X 656.88,Y 692.65,Z 676.15,A 12.98,B -13.33,C 35.83}

HOMELOCAL = {AXIS: A1 0,A2 -135,A3 90,A4 0,A5 90,A6 0}
$base={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
$tool={X 19.711,Y -0.955,Z 81.963,A 0,B -180,C 0}

;====================================================
; Put robot in the HOME position
PTP HOMELOCAL

;====================================================
; Approaching sample
PTP ApprStartPoint                ; Point-to-point motion to ApprStartPoint
LIN StartPoint                    ; Linear motion to StartPoint

;====================================================
; Create and start RSI Context
ret = RSI_CREATE("RSI_External_control.rsi",CONTID,TRUE)

; Start RSI execution
$out[5]=TRUE           ; Set OUT5 and OUT6 to “true” to communicate start of scan
$out[6]=TRUE           ; to the C# server application.

ret = RSI_ON(#ABSOLUTE,#IPO)

;====================================================
; Externally controlled robot motion
RSI_MOVECORR()

;====================================================
; Turn off RSI
$out[5]=FALSE           ; Set OUT5 and OUT6 to “false” to communicate start of scan
$out[6]=FALSE           ; to the C# server application.
ret = RSI_OFF()

;====================================================
; Abandon scanned surface
LIN LeaveEndPoint                     ; Linear motion to LeaveEndPoint

;====================================================
; Return to HOME position by a PTP movement
PTP HOMELOCAL
END
```

Setting the approaching and the abandoning movements by manual typing of all coordinates is time-consuming and not very good in terms of automation. Moreover, it is mainly inconvenient because of the risk of introducing human typing errors and copying wrong coordinates. It can result in the robot following unpredictable trajectories, which are extremely worrying for the health and safety of the working environment.

A new approach was developed to configure the aforementioned movements avoiding the need for the operator to copy coordinates to the KRL module. Each packet of coordinates is sent from the server computer to a global KRC variable, visible from the RSI context. The content of the variable can then be moved to a local positional variable (POS) within the KRL module. This approach was put in place through creating an appropriate RSI configuration, different from the already known configuration to control the robot motion during the scan.

The signal flow for the new RSI context was configured using the RSI Visual user interface. RSI objects were dragged into the signal flow editor and linked together by means of the object-specific signal inputs and outputs. The resulting schematic representation is shown in Fig. 9.

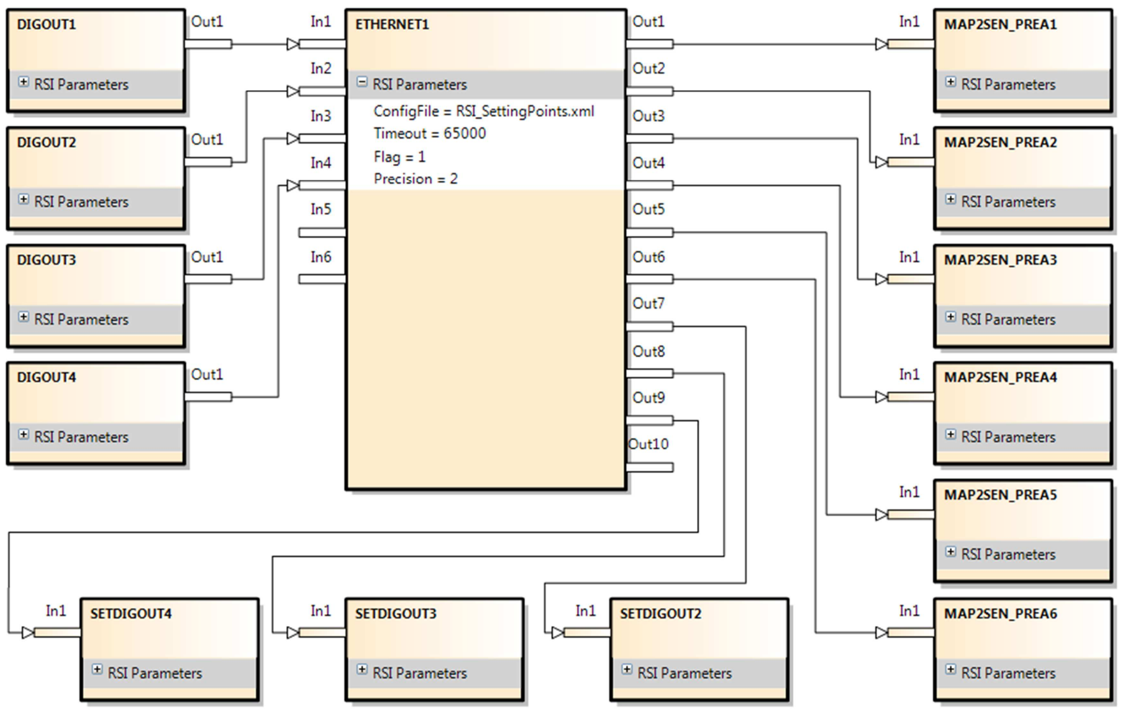


Fig. 9 – RSI signal flow schema for the new configuration.

The main object of the new schema is ETHERNET1. The status of four digital outputs (from OUT1 to OUT4) of the robot controller is copied into four inputs of ETHERNET1 and they are used to trigger the progress of the KRL module execution. They are used like flags to communicate to the KRL module that a packet of coordinates has been sent by the C# application. The six Cartesian coordinates received from the external computer are copied to the KRC global variable *$SEN_PREA*. ETHERNET1 modify the value of three outputs, from OUT2 to OUT4. Each RSI object has its own properties. ETHERNET1 is set to send/receive XML strings according to the XML template "RSI_SettingPoints.xml". This XML format to exchange data via Ethernet, stored in the same directory of the other configuration files, has the following structure:

```
RSI_SettingPoints.xml
-------------------------------------------------------------------------------------
-<ROOT>
   <CONFIG>
      <IP_NUMBER>10.1.1.110</IP_NUMBER>  <!-- IP Number of the socket !-->
      <PORT>49202</PORT>                 <!-- Port Number of the socket !-->
      <SENTYPE>ImFree</SENTYPE>          <!-- The name of the system !-->
      <ONLYSEND>FALSE</ONLYSEND>         <!-- TRUE means the client does not expect
                                              an answer !-->
   </CONFIG>
   <SEND>
```

```
      <ELEMENTS>
         <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" />
         <ELEMENT TAG="Digout.o1" TYPE="BOOL" INDX="1" />
         <ELEMENT TAG="Digout.o2" TYPE="BOOL" INDX="2" />
         <ELEMENT TAG="Digout.o3" TYPE="BOOL" INDX="3" />
         <ELEMENT TAG="Digout.o4" TYPE="BOOL" INDX="3" />
      </ELEMENTS>
   </SEND>
 <RECEIVE>
      <ELEMENTS>
         <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" HOLDON="1" />
         <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" HOLDON="1" />
         <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" HOLDON="1" />
         <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" HOLDON="1" />
         <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" HOLDON="1" />
         <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" HOLDON="1" />
         <ELEMENT TAG="Break1" TYPE="BOOL" INDX="7" HOLDON="1" />
         <ELEMENT TAG="Break2" TYPE="BOOL" INDX="8" HOLDON="1" />
         <ELEMENT TAG="Break3" TYPE="BOOL" INDX="9" HOLDON="1" />
      </ELEMENTS>
   </RECEIVE>
</ROOT>
```

Therefore all three points are sent from the external computer to the KRC and their final reception into KRL local positional variables is communicated back to the C# server application by using the robot digital outputs, whose status is accessible by the RSI context. The following copy of the KRL Module (*KRL_Module.src)* shows how the heading part and the *settings section* were modified to support the aforementioned approach.

```
RSI_Module.src
--------------------------------------------------------------------------------
DEF RSI_Module( )
; ============================================
; KLR Module for real-time UDP data exchange with C# server application
; Author: Carmelo Mineo
; Date: May 2013
; ============================================

; Declaration of KRL variables
DECL INT ret0, ret1              ; Return values for RSI commands
DECL INT CONTID0, CONTID1        ; ContainerID
DECL INT C                       ; Counter
DECL POS ApprStartPoint          ; Approaching point (200 mm above surface)
DECL POS StartPoint              ; Start point
DECL POS LeaveEndPoint           ; Leaving point (200 mm above surface)
DECL AXIS HOMELOCAL              ; Home position

;FOLD INI

;==================================================
; Settings
HOMELOCAL = {AXIS: A1 0,A2 -135,A3 90,A4 0,A5 90,A6 0}
$base={X 0.0,Y 0.0,Z 0.0,A 0.0,B 0.0,C 0.0}
$tool={X 19.711,Y -0.955,Z 81.963,A 0,B -180,C 0}

$OUT(1)=FALSE
$OUT(2)=FALSE
$OUT(3)=FALSE
$OUT(4)=FALSE
$out(5)=FALSE
$out(6)=FALSE

; Create RSI Context
ret0 = RSI_CREATE("RSI_SettingPoints.rsi",CONTID0,TRUE)
ret0=RSI_ON(#ABSOLUTE,#IPO)

WAIT FOR ($OUT(2))
```

```
ApprStartPoint.x=$SEN_PREA(1)
ApprStartPoint.y=$SEN_PREA(2)
ApprStartPoint.z=$SEN_PREA(3)
ApprStartPoint.a=$SEN_PREA(4)
ApprStartPoint.b=$SEN_PREA(5)
ApprStartPoint.c=$SEN_PREA(6)

$OUT(1)=TRUE
WAIT FOR ($OUT(3))

StartPoint.x=$SEN_PREA(1)
StartPoint.y=$SEN_PREA(2)
StartPoint.z=$SEN_PREA(3)
StartPoint.a=$SEN_PREA(4)
StartPoint.b=$SEN_PREA(5)
StartPoint.c=$SEN_PREA(6)

$OUT(1)=FALSE
WAIT FOR ($OUT(4))

LeaveEndPoint.x=$SEN_PREA(1)
LeaveEndPoint.y=$SEN_PREA(2)
LeaveEndPoint.z=$SEN_PREA(3)
LeaveEndPoint.a=$SEN_PREA(4)
LeaveEndPoint.b=$SEN_PREA(5)
LeaveEndPoint.c=$SEN_PREA(6)

$OUT(1)=TRUE

WAIT SEC 0.02

ret0=RSI_OFF()
ret0 = RSI_RESET()

;=====================================================
; THE MODULE CONTINUES AS PREVIOUS VERSION …
```

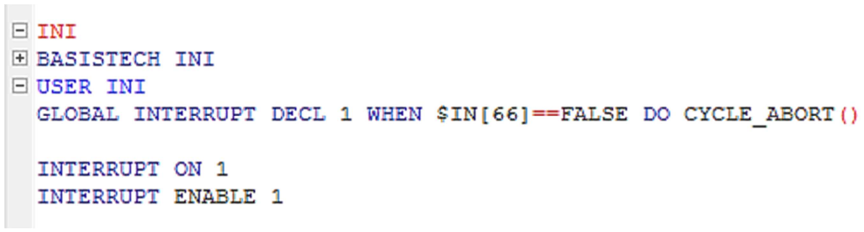
The C# server application was also updated to support the management of the streaming of the packets of coordinates and the logic was matched to the status of the digital flags. This new approach enables the development of seamless software for completely automated inspection processes, from the positioning of the sample within the robot cell to the evaluation of the results.

C1 – Setting a software interrupt to stop the robot motion

The three push button switches of the magnetic holder of the IntACom system where designed to send a digital signal to the robot controller, to stop the motion of the arm every time the Slave side is released for any reasons.

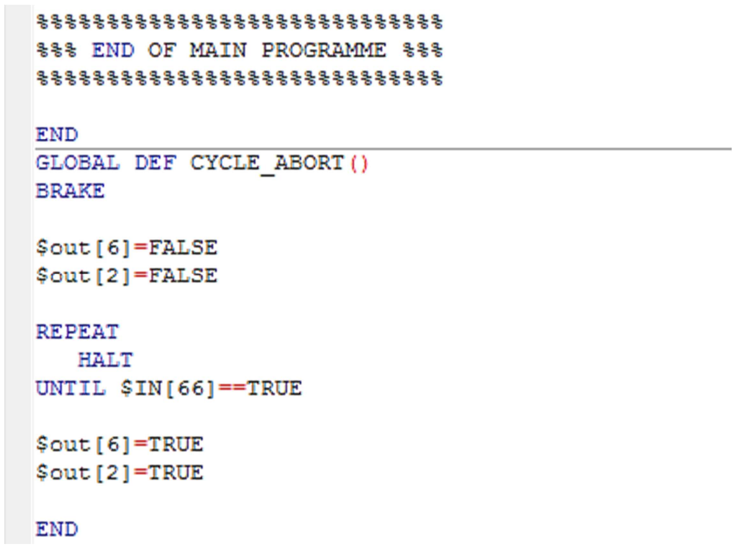
It was investigated how to set an interrupt into a KUKA Robot Language (KRL) programme. The robot controller processes the programme code line by line; the task of ensuring the magnetic holder is constantly locked during the execution of robotic movements has to be defined in a sub-routine running in background.

The aforementioned requirement was fulfilled through the following approach. Within the USER INI fold of the KRL programme it is possible to define an INTERRUPT by calling a sub-program if a certain state of a given input is verified:



```
INI
BASISTECH INI
USER INI
GLOBAL INTERRUPT DECL 1 WHEN $IN[66]==FALSE DO CYCLE_ABORT()

INTERRUPT ON 1
INTERRUPT ENABLE 1
```

The code was written so that when the digital input #66 is false (one or more push button switches are released) the sub-program CYCLE_ABORT() is called, regardless of the current program state. The CYCLE_ABORT() program is defined at the bottom of the KRL programme file. This program can simply execute a HALT command, but it is possible to program whatever functionality is deemed necessary.



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END OF MAIN PROGRAMME %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

END
GLOBAL DEF CYCLE_ABORT()
BRAKE

$out[6]=FALSE
$out[2]=FALSE

REPEAT
   HALT
UNTIL $IN[66]==TRUE

$out[6]=TRUE
$out[2]=TRUE

END
```

The REPEAT loop is used to prevent the program from restarting if someone should press the cycle start button while IN[66] is still FALSE. Without the loop, the CYCLE_ABORT() program completes. Then the robot returns to the program step that was executing at the time of the interrupt. The Robot Sensor Interface (RSI) continues to run during the interruption of the robotic motion. As it was explained in Chapter 3, the RSI is responsible for creating the Ethernet communication with the external computer to send the robot's current coordinates or to receive command coordinates. The change of the state of OUT[2] and OUT[6] is used in a sub-program to communicate to the external computer that the robotic arm is at standstill and the current coordinates do not need to be recorded. The original state of the output signals is restored when the IN[66] becomes TRUE (the magnetic holder is locked again).

This is probably the best method to stop the system when the magnetic holder is unlocked, meaning that a crash or an unwanted disconnection of the tool has happened. It is possible to include this interrupt in each program that uses the magnetic holder. In other words, it must be included in each program that the functionality is required in.

C2 – Water-jet solutions

Some ultrasonic non-destructive testing apparatus utilizing water jet coupling have been investigated by some companies in the last few years.

Even if some air-coupled techniques have been developed, traditional immersion testing, gap coupling and water jet coupling still dominate the industrial use. Ultrasonic couplant is necessary in all contact testing applications because sound energy at the ultrasonic frequencies typically used for non-destructive testing is not effectively transmitted through air. Aside from attenuation effects, air represents a severe acoustic impedance mismatch with respect to both transducer wearplates and typical test materials. Even an extremely thin air gap between the transducer and the test piece will prevent efficient sound energy transmission and make conventional testing impossible. Couplants are typically moderately viscous, nontoxic liquids, gels, or pastes. A number of common substances such as water, motor oil, grease, and even some commercial products like hair gel can be used as ultrasonic couplants in many applications. Water is the most used couplant. This usually governs the design of every testing system. Historically, immersion testing has been a very common method for smaller specimens (Fig. 10a). Usually the entire specimen is immersed in water and a stage carrying the ultrasonic probes is moved along the specimen. The scanning routines can be more or less automated depending on the desired through-put. For an on-line inspection of long profiles, the immersion technique presents the difficulty to keep the water chamber filled while the specimen is entering and leaving the chamber. Other coupling methods are therefore mainly used. Partial immersion means that only the bottom of the specimen is immersed in water (Fig. 10b). This technique is often used for tube and bar inspection. Since only a small portion of the specimen is immersed, only this portion can be inspected at a time. A rotational inspection is commonly used to produce full coverage during the inspection. If specimens have considerable straightness deviations, it is advisable to guide the probes along the specimen profile. Special probe holders have been designed and a commonly used technique is gap coupling (Fig. 10c). The probe holders rest on the specimen surface with a skid which is wear-resistant. A dual-element probe is used and the distance between probe surface and specimen surface is within 2mm. The design of the probe holder is simple but the wear of the skids and the danger of damaging the probe (high temperature or loose scale on specimen) have led to more sophisticated solutions [3].

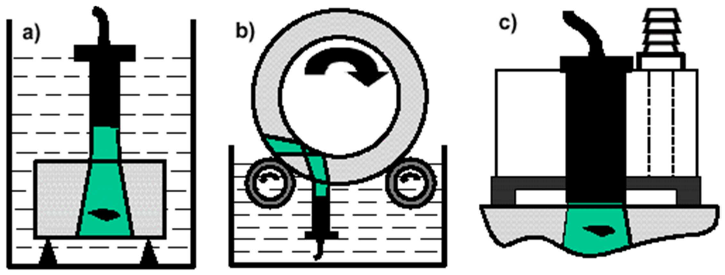


Fig. 10 – Coupling techniques, a) immersion testing, b) partial immersion, c) gap coupling.

The previous methods are not particularly suitable for scanning composite parts with wide and complex geometries. The following two methods show major possibility of integration with industrial robotic arms:

- Bubbler / local immersion technique (guided water jet)
- Squirter / water jet method (free water jet)

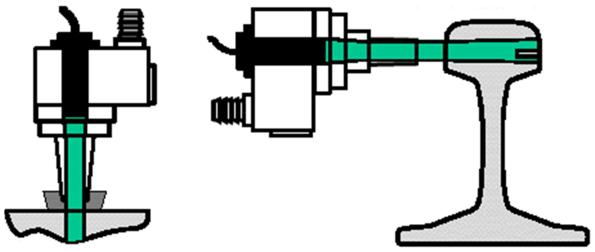


Fig. 11 – Ultrasonic coupling techniques, a) guided water jet, b) free water jet.

Guided water jet coupling offers a higher near-field resolution and better protection of the probes (Fig. 11a). Again, wear-resistant skids or rollers can be used to guide to probe holder along the specimen. Standard immersion type probes are used and a fairly long water column (30 - 50 mm) between probe and specimen guides the ultrasound. Many manufacturers produce an extensive range of bubbler style transducers. These transducers provide a minimal beam (spot) size and other benefits of an immersion style transducer, while providing the convenience and mobility of contact style transducers. Essentially, they are "captive" immersion transducers supporting a captive water path.

Free water jet coupling is wear-less (Fig. 11b) and it is especially suitable for the inspection of complex profiles. The probe holders are fairly complicated; producing a water column which is bubble-free for undisturbed guidance of the ultrasound is a difficult task.

An early ultrasonic non-destructive testing apparatus that used water jet coupling, for ultrasonic through-transmission inspection, was patented in USA in 1979 [4] (Fig. 12). A pair of transducer holders was inserted into the respective nozzle housings, behind electrically isolated and watertight compressible rubber O-rings. It was seen that although a curved path (not a straight line path) was provided for the acoustically modulated stream (between the central axis of a major surface area of the shaped piezo-electric transmitting transducer and the central axis of the piezo-electric receiving transducer) with intermediate through-transmission of acoustical energy provided by direct coupling through sample, such curved path still results in the transmission of sound through a bent water column between transducers.

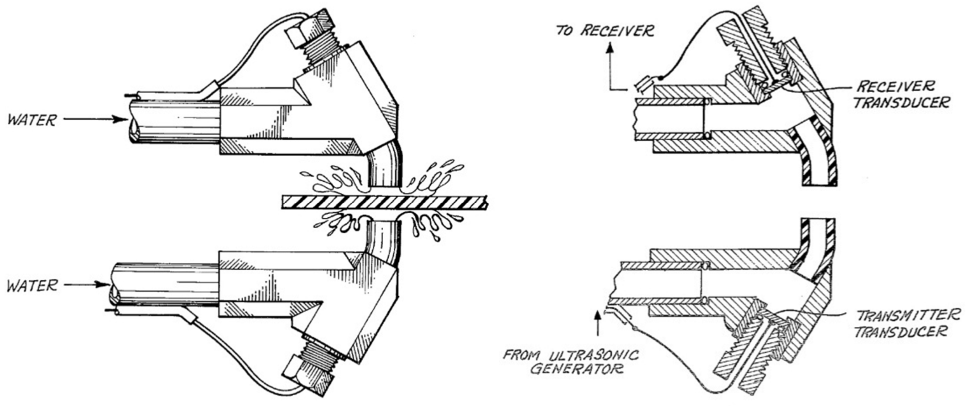
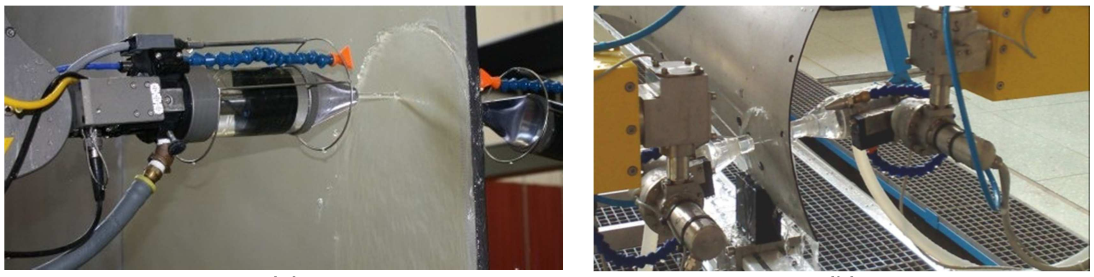


Fig. 12 – Patented water-jet apparatus for ultrasonic through-transmission inspection [4].

Several inspection systems have successfully been installed using the water-jet method, using the previous and similar systems. Machine design manufacturers, like Marietta Non-Destructive Testing (MNDT), working in partnership with drive and control experts from Bosch Rexroth and its automation distributor, Livingston & Haven, needed to develop a system for testing complex shaped geometries [5]. To overcome the challenges of scalability, reliability and speed, MNDT engineers designed the gantry style AG2 Overhead Scanner – a rigid, multi-axis, automated testing machine capable of scanning large, sophisticated parts and intricate shapes without the use of immersion tanks. With a scanning envelope of up to 18 m x 6 m x 5 m, this machine can easily be configured to test a wide range of parts for each customer, instead of being designed for one specific part. The system's ultrasonic scanning system uses two sets of squirter jets that face each other (Fig. 13a).


(a) (b)
Fig. 13 – Automated NDT systems based on water jet coupling.

An order was received by Ultrasonic Sciences Ltd in March 2010 [6] for a 17-axis system for inspection of complex geometry composite parts. This had to be used by a UK manufacturer of components for the F35 Joint Strike Fighter and supplied to European aerospace companies. Users include BAE Systems, Fokker Aerospace and Hafei Aviation. This system was produced with independent, horizontally opposed 5-axis probe manipulators shown in Fig. 13b and the water jet method ensured the coupling of ultrasound for the through-transmission technique. The application of the water jet system is not limited to single element probes. Phased array probes add an increased difficulty in the squirter system, related to the surface extension of the transducers. Fig. 14 shows equipment developed by GE Inspection Technologies with a phased array squirter system. The squirter nozzles are shaped to grant an oval outlet of the water jet. The system has a useful coverage of 12mm and has about 10 times higher throughput compared to a single probe squirter system [7].

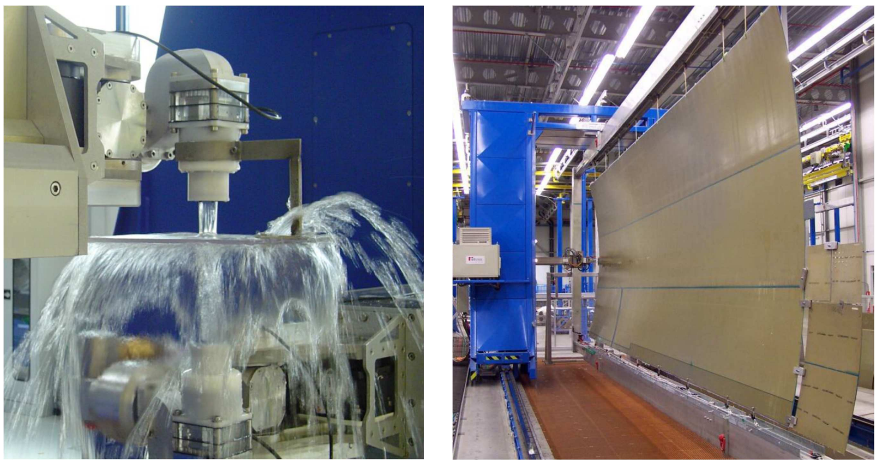

(a) (b)

Fig. 14 – a) Through Transmission Squirter arrangement for phased array probe [7], b) Squirter Fuselage test equipment [7].

Although horizontal distribution of the water can be obtained with the deflecting planes (e.g. see Fig. 12), these are not suitable for producing a laminar flow. Some jet nozzles to perform the squirter technique present a multiplicity of holes, from each of which emerges a free water jet. Each of the jets carries an ultrasonic ray, which is excited by a separate ultrasonic transducer in each instance. The phase relationship between all individual rays is adjustable. Because of turbulence upon impact of each of the individual water jets, the diameter of which is at least one-half wavelength, e.g., about 1 mm, turbulence may be produced which may result in a weakened acoustic signal and hence in an incorrect evaluation. The problem is solved by one or more filtering chambers situated upstream to a Funnel-shape flow space. Each chamber comprises a diving wall having flow channels running perpendicularly to the direction of flow. The number of flow channels in each diving wall increases in the direction of flow and their diameter decreases. To avoid additional reflection and scattering, the jet nozzle preferably is made of a synthetic material having low acoustic velocity, such as acrylic glass. Laminar flow is to be maintained until impact on the surface of the component to be inspected, so that turbulence is virtually ruled out. It makes acoustic coupling and reception of reflected acoustic signal possible.

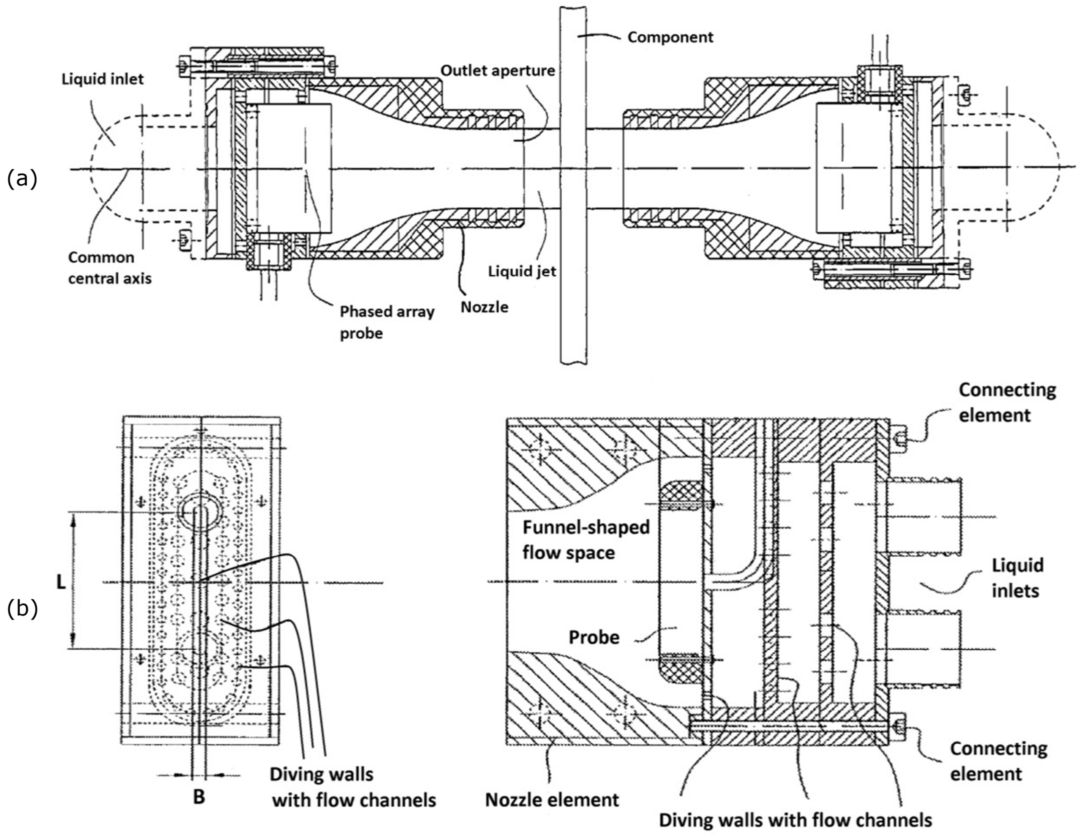


Fig. 15 – a) Arrangement for ultrasonic through-transmission inspection of a component by the squirter method [8], b) Details of the nozzle [8].

Fig. 15 shows the basic inspection arrangement of a component by the squirter method and additional details of the nozzle. The water jet is a rectangular-section jet with width (B) in the range of 4 mm<B<20 mm (preferably B = 8 mm) and length (L) in the range of 40 mm<L<120 mm (preferably L = 90 mm), where B and L have to be greater than one-half the wavelength of the used acoustic signals.

C3 – Software configuration to automate the return to areas of interest

In order to enable the robot controller to support this approach, two KUKA Robot Language (KRL) functions were developed. The functions are general, so that they can be used in any KRL tool path generated via third parts commercial software. The first function is *SendData()*; it sends current *base* and *tool* coordinates to the external computer via strings over the Ethernet socket. These two packets of coordinates are thus received by the external computer that will use them for the successive external control of the sub-scan tool path. The function uses the RSI configuration given in Fig. 16.

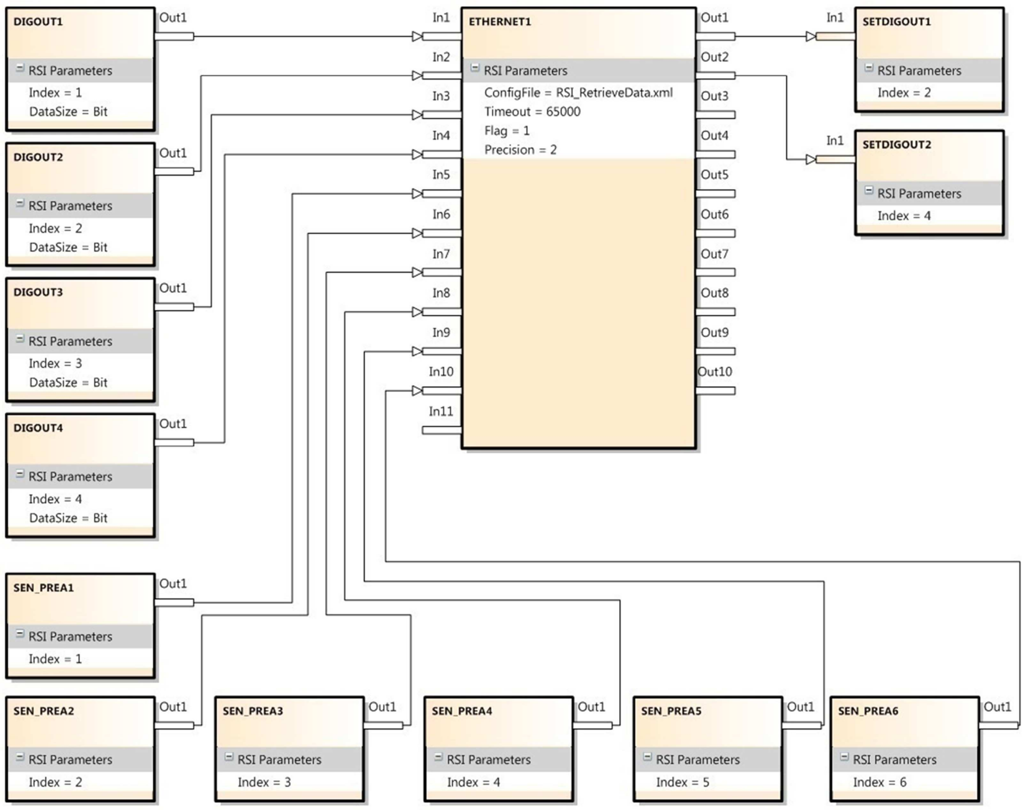


Fig. 16 – RSI configuration scheme for the *SendData()* function.

The six coordinates of base and tool are stored in the array SEN_PREA. This vector array is visible from the RSI context, so the six coordinates are respectively redirected to six inputs of the RSI Ethernet object. The values of four digital output variables (DIGOUT 1-4) are also linked to the Ethernet object and transferred to the external computer. The digital outputs of the robot controller are used to trigger the reception of both coordinate packets for the base and the tool.

The second KRL function is *SubScan().* It sets *base* and *tool* back to the values used during the execution of the initial scan. This makes sure the robot manipulator behaves like in the initial scan, avoiding unexpected actions that can originate from wrong base and tool parameters. Once the base and tool coordinates are properly transmitted from the external computer to the robot controller, *SubScan()* enables the external control to allow the external computer to command the motion of the robot arm through the Ethernet socket. The function *SubScan()* manages two RSI contexts; the first RSI configuration is used during setting base and tool parameters, the second during the execution of the sub-scan tool path. The first RSI configuration is also used to set all approaching and retracting points for the sub-scan path, allowing the robotic arm to reach the starting point at the beginning of the scan and come back to the Home position at the end of the task. The schematic representation of this RSI configuration is given in Fig. 17.

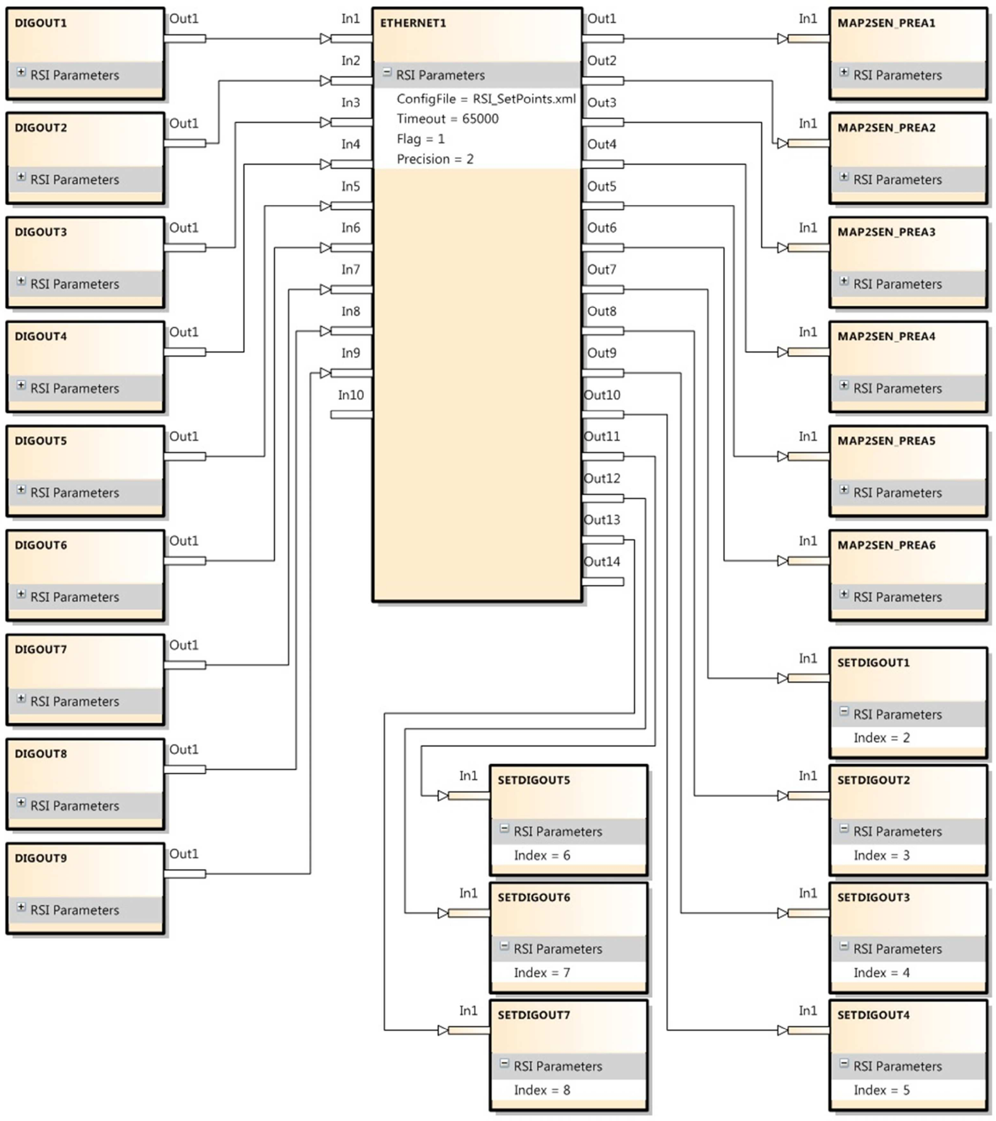


Fig. 17 – RSI configuration scheme for setting base, tool, approaching and retracting points.

Each packet containing six coordinates is mapped into SEN_PREA and the stored values are read during the execution of the KRL module. The progress of the KRL module and the C# code running on the external computer is led by the status of nine digital output variables (DIGOUT 1-9).

The second and last RSI context initialised by the KRL function *SubScan()* is responsible for moving the robotic arm according to the command coordinates received from the external computer via the Ethernet socket. The RSI configuration is shown in Fig. 18.

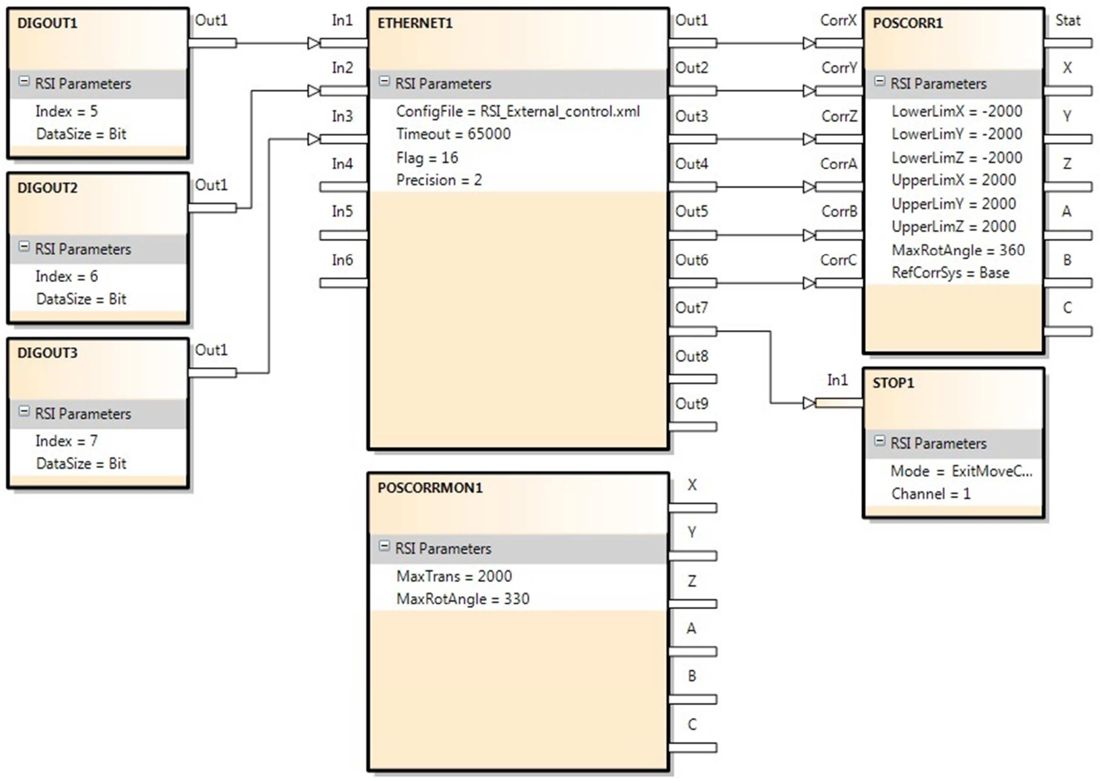
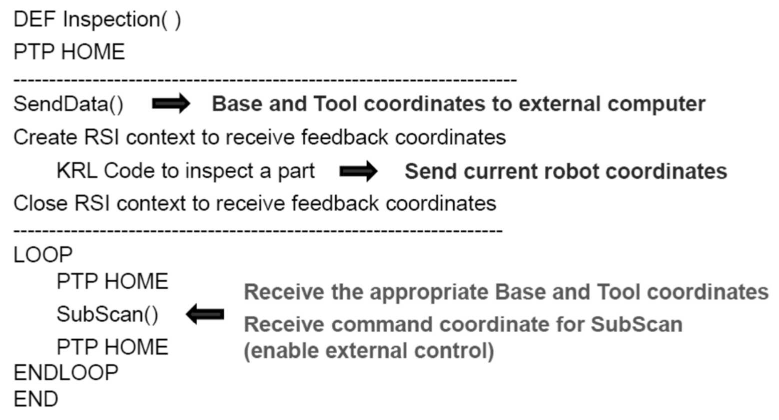


Fig. 18 – RSI configuration scheme for enabling the external control of the robot motion.

The main RSI object is again ETHERNET1; it receives some inputs and provides some outputs. The inputs are variables read from the robot controller and sent to the external computer. The data received from the external computer are redirected to the outputs and become available for the robot controller. The status of three digital output variables (DIGOUT 5-7) of the robot controller is copied in three inputs of ETHERNET1 and they are used to trigger the external motion control operated by the server C# application that starts streaming command coordinates. The six Cartesian coordinates received from the external computer are used by the POSCORR1 object, to influence the motion of the robotic arm. RSI_MOVECORR() is the command inside the KRL program that activates the sensor-guided motion. POSCORRMON1 is the object that limits the maximum overall Cartesian correction. The default maximum translation and rotation are equal to 6 mm and 6 degrees, but they were set to 2000 mm and 330°. STOP1 is used to abort the sensor-guided motion when the Boolean variable available at the Out7 variable changes from “false” to “true”. Each RSI object has its own properties. POSCORR1 gives the possibility to specify the lower limit and the upper limit of the positional correction on each Cartesian axis, the maximum rotational angle and the coordinate reference system. ETHERNET1 was set to send robots current coordinates with a precision of 2 decimal places, to switch $FLAG[1] to “TRUE” in case of interrupted communication, to allow a maximum number of 65000 late packets and to send/receive XML strings according to the XML template “RSI_External_control.xml”.

The general template of the KRL programme is given below:



```
DEF Inspection( )
PTP HOME
-----------------------------------------------------------------------
SendData()    ➡  Base and Tool coordinates to external computer
Create RSI context to receive feedback coordinates
    KRL Code to inspect a part  ➡  Send current robot coordinates
Close RSI context to receive feedback coordinates
---------------------------------------------------------------------
LOOP
    PTP HOME       Receive the appropriate Base and Tool coordinates
    SubScan()  ⬅   Receive command coordinate for SubScan
    PTP HOME       (enable external control)
ENDLOOP
END
```

D1 – Template of KRL modules for Master-Slave synchronization

The following shows an example of the KRL modules (for Master and Slave robot) to use the GeoLink synchronization approach.

```
Master Robot KRL Module

PTP HOME

;====== Approach motion ======
- Reach start point

;==== Geometric coupling ====
PROGSYNC ScanStart -> R1_R2 WAIT
GEOLINK MASTER Scan_Start -> R1_R2

;====== Scan tool-path ======
- Reach point
- Reach point
- Reach point
- Reach point
- ...

;====== Decoupling ======
PROGSYNC ScanEnd -> R1_R2 WAIT
GEOLINK MASTER Scan_End -> R1

;====== Retracting motion ======
- Reach retracting point

PTP HOME

END
```

```
Slave Robot KRL Module

PTP HOME

;====== Approach motion ======
- Reach start point

;==== Geometric coupling ====
PROGSYNC ScanStart -> R1_R2 WAIT
GEOLINK SLAVE Scan_Start -> R1_R2
Base[1]:LK_BASE(Robot 1)

;==== Follow master tool-path ====




;====== Decoupling ======
PROGSYNC ScanEnd -> R1_R2 WAIT
GEOLINK SLAVE Scan_End -> R1 Base[8]:

;====== Retracting motion ======
- Reach retracting point

PTP HOME

END
```

References

[1] KUKA, *KUKA.RobotSensorInterface 2.3 Documentation - Version: KST_RSI_2.3_V1_en*, 2009.

[2] KUKA, *KUKA.RobotSensorInterface 3.1 Documentation - Version: KST_RSI_3.1_V1_en*, 2010.

[3] W. A. K. Deutsch, "Automated ultrasonic inspection," in *Conference Proceedings of 15th WCNDT Conference. Rome*, 2000.

[4] L. W. George, "Water coupled ultrasonic through transmission apparatus," ed: Google Patents, 1979.

[5] Tecnatom, "Tecnatom passes the acceptance tests for two new aeronautical systems," *NDT.net e-Journal,* vol. 19, 2013.

[6] USL, "A new NDT development using high speed robots," *NDT.net e-Journal,* vol. 19, 2013.

[7] A. Maurer, W. Haase, and W. De Odorico, "Phased array application in industrial scanning systems," *ECNDT Proceeding, September,* 2006.

[8] A. Maurer, M. Strauss, W. D. Oodorico, W. Haase, and R. Kock, "Ultrasound probe arrangement," United States Patent, 2010.