Department of Bioengineering
University of Strathclyde

A Virtual Reality Electric Powered Wheelchair Simulator:

a Research Platform for
Brain Computer Interface Experimentation

a Doctor of Engineering thesis by

Bartłomiej Grychtol

Supervisors:
Prof. Bernard A. Conway
Dr Heba Lakany

Glasgow, 2011

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.

The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Signed: Date:

21 September 2011

To Niki, the light of my days

Abstract

For patients with high-level spinal cord injury (SCI), autonomous control of an electric powered wheelchair (EPW) is very difficult, if not impossible. Recent developments in brain-computer interfaces (BCI) suggest an alternative pathway for wheelchair control. To develop a BCI-controlled wheelchair, a safe test-bed on which to conduct repeatable trials and a training platform for prospective users are required. The thesis presents a virtual environment wheelchair driving simulator to satisfy this need.

Unlike previously reported approaches, the simulator is based on a commercial game engine which significantly lowers the hardware requirements. Incorporating research software from the field of robotics and a dome display system, the system constitutes a powerful and extendible wheelchair simulation at a relatively low cost, addressing a gap in the literature and a need in the clinic.

To accommodate for the small command set afforded by most BCIs, a novel method to control an EPW has been developed. Healthy volunteers navigating a bespoke virtual obstacle course obtained similar results when using a conventional joystick and the new method on a number of performance measures such as time to complete or average distance from the prescribed path. A questionnaire administered after the experiment revealed subjects experienced immersion, flow and positive affect while using the simulator.

The simulator was integrated with previous work on detecting the direction of intended motion from EEG signals, leading to the first complete BCI system at the University of Strathclyde - the Strathclyde BCI. The system is used to study the impact of feedback on the performance of BCI in healthy volunteers, revealing a learning effect when accurate real-time feedback is provided.

The simulator presented herein addresses a key requirement for developing an intuitive BCI for wheelchair control and constitutes a tool for further investigation into basic and applied neuroscience.

Contents

Acknowledgements

First and foremost, I am grateful to my parents and my wife, without whose enduring love and support throughout the years I would have never got to where I am now.

None of the work presented here would be possible without the invaluable guidance, support and advice of my supervisors Prof. Bernard Conway and Dr Heba Lakany for which I am immensely grateful.

I remain indebted to Dr Gopal Valsan for his help in the lab and his knowledge in neurophysiology which he shared with me as well as his previous work on which I was able to build. I am also grateful to Bahman, Ange and Pauline for their company, advice and stimulating discussions in the lab.

I am thankful to Mr. David Smith who helped me design the new projector frame and Mr. Stephen Murray who built it.

Thanks are also due to the Department of Architecture, University of Strathclyde, for lending me their theodolite and Mr. Derek McNee for showing me how to use the device.

Dr Heba Lakany contributed ideas and Dr Gopal Valsan also data to the study presented in chapter 6. The results were published in a jointly authored paper (Grychtol et al., 2010).

Last but not least, I am grateful to the Department of Bioengineering and the Engineering and Physical Sciences Research Council for funding my studies. I am thankful to all my subjects who offered their time to participate in my experiments.

List of Figures

List of Tables

List of Symbols and Abbreviations

Abbreviation	Description	Definition on p.
AOV	Angular field of vision	78
BCI	Brain Computer Interface	1
BSP	Binary Space Partition	36
CoM	Centre of mass	40
DOF	Degree of freedom	14
ECoG	Electrocorticography	2
EEG	Electroencephalography	3
EMG	Electromayography	3
EPW	Electric Powered Wheelchair	1
ERD	Event Related Desynchronisation	3
ERP	Event Related Potential	3
ERS	Event Related Synchronisation	3
EY	Eye position along the y axis	65
EZ	Eye position along the z axis	65
FFT	Fast Fourier Transform	14
fMRI	Functional Magnetic Resonance Imaging	2
FPS	Frames Per Second	22
FSM	Finite State Machine	22
GEQ	Game Experience Questionnaire	28
GUI	Graphical User Interface	9
HMD	Head-Mounted Display	12
LY	Lens position along the y axis	65
LZ	Lens position along the z axis	65
MEG	Magnetoencephalography	2
MFC	Microsoft Foundation Class library	19
OS	Operating System	9
PSD	Power Spectral Density	97
SCI	Spinal Cord Injury	1
SPI	Spherical Projection of Images	19

Abbreviation	Description	Definition on p.
SSVEP	Steady State Visual Evoked Potential	3
UDN	Unreal Developer Network	23
UT2004	Unreal Tournament 2004	15
VE	Virtual Environment	4
VR	Virtual Reality	7

Chapter 1

Introduction

1.1 Background

Spinal cord injury (SCI) is a traumatic life-changing event, almost always leading to some degree of paralysis, depending on the level of the lesion. There are about 20 cases per million population (666 in 2005) in the UK alone (Department of Health, 2005), while in the USA the incidence is twice as high with about 12000 new cases every year (National Spinal Cord Injury Statistical Centre, 2010). People with SCI experience a low quality of life, largely as a result of lost independence in terms of mobility and self-care (Putzke et al., 2002; van Koppenhagen et al., 2008). One way in which their independence can be increased is through provision of an appropriate electric powered wheelchair (EPW). However, for many controlling a conventional wheelchair by mechanical means is difficult or impossible due to the extent of their paralysis or other conditions, such as tremor or spastisity.

Over the past 30 years, advances in mobile robot technology allowed the development of several "smart wheelchairs" that possess a degree of autonomy and relieve the user of some of the burden of operating the chair. They are equipped with various sensors and implement behaviours such as wall-following, obstacle avoidance or path planning and new functions are continually being developed (see Simpson (2005) for a review). Still, for some quadriplegics controlling even a smart wheelchair by mechanical means is impossible.

Brain-computer interfaces (BCI) are systems that translate information extracted from recorded brain signals into commands to control and/or communicate with external devices offering a new brain output pathway, see Figure 1.1. They have developed significantly over the past decade to a point where multiple command states can be identified and utilised for navigation (see Wolpaw et al. (2002) for a review). Many BCI paradigms have been developed and a significant research effort is directed towards developing a BCI-controlled (smart) wheelchair, with some prototypes already

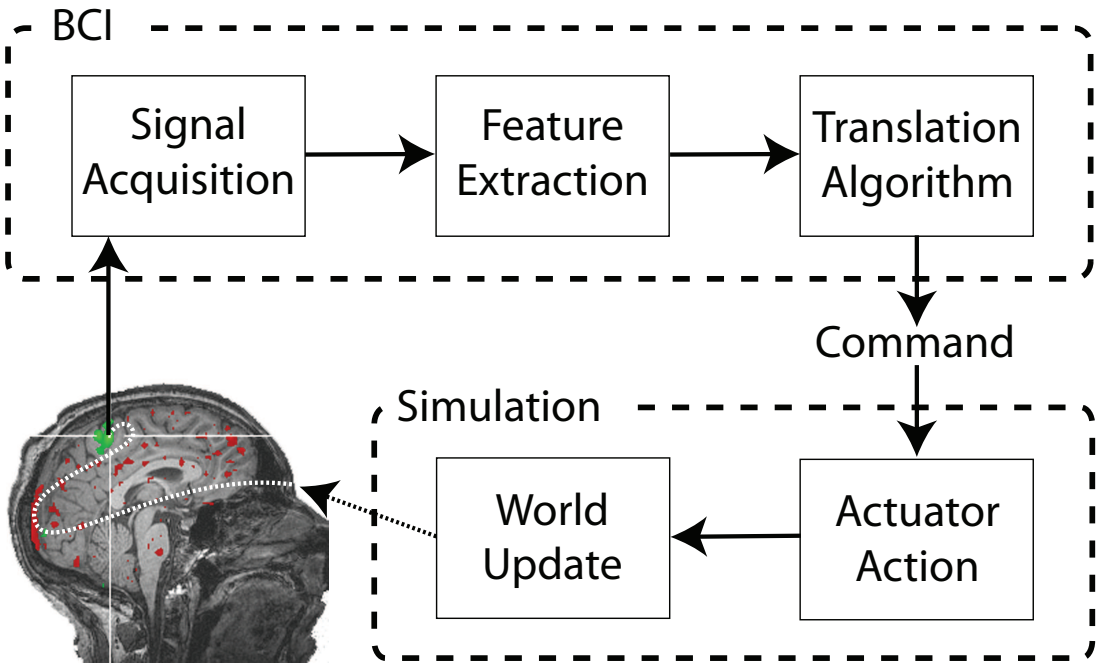


Figure 1.1: A brain-computer interface (BCI) as part of a bio-feedback loop as pertaining to the thesis. Electrical signals acquired over the subject's motor cortex are analysed to extract the intention of movement and translate it into a machine-usable command. The command is used to control an actuator within a simulation. The information loop is closed by visual feedback provided to the subject.

in existence (e.g. Rebsamen et al., 2007; Vanacker et al., 2007).

In recent years, virtual reality (VR) has attracted much interest in the field of motor rehabilitation engineering. It has been used for training and assessment of wheelchair users (Erren-Wolters et al., 2007), where a transfer of skills to real life was demonstrated (e.g. Adelola et al., 2005), rehabilitation (Holden, 2005) and testing new wheelchair systems (Stott and Sanders, 2000), among others. Also in the field of robotics, VR simulators help decrease the cost of developing and testing new systems (e.g. Carpin et al., 2007). Naturally, in BCI research where both algorithm development and user training is required, virtual reality emerges as the method of choice (e.g. Bayliss and Ballard, 2000; Lalor et al., 2005; Leeb, Friedman, Müller-Putz, Scherer, Slater and Pfurtscheller, 2007; Galán et al., 2008; Edlinger et al., 2009; Grychtol et al., 2010).

1.2 Motivation and Objectives

Many approaches to building a brain computer interface exist, differing in the type and source of the recorded brain signal and the underlying physiological mechanism exploited (see Vallabhaneni et al. (2005) or Allison et al. (2007) for a recent review). Monitoring neuronal potential from individual neurons or groups of neurons with implanted electrode micro-arrays has shown great potential in primate research (e.g.

Velliste et al., 2008) but few such studies have been conducted on humans (Vallabhaneni et al., 2005). More frequent are partially invasive approaches using electrodes placed just beneath the dura mater (electrocorticography, ECoG), see e.g. Leuthardt et al. (2006). However, despite their comparatively poor spatial resolution and signal quality, non-invasive approaches are by far the most popular. While magnetoencephalography (MEG) (e.g. Chen and Bai, 2009) and functional magnetic resonance imaging (fMRI) (e.g. Yoo et al., 2004) are sometimes used, most frequently BCI systems rely on electroencephalography (EEG) signals, which are best understood and least expensive to acquire. EEG exhibits several features that can be endogenously or exogenously controlled and hence exploited for a BCI application (Vallabhaneni et al., 2005).

Among the most successful is the P300 paradigm, whereby a positive event related potential (ERP) is elicited approximately 300 ms after the presentation of an infrequent relevant stimulus within a sequence of other stimuli. In the context of wheelchair control, the stimuli are most often implemented as a matrix of symbols indicating possible steering commands, flashing in random order (e.g Rebsamen et al., 2007; Pires et al., 2008; Gentiletti et al., 2009; Iturrate et al., 2009). Since P300 is a natural response, it is reliable and does not require subject training. It is also well suited to support a large number of commands. However, this mode of control is necessarily slow as the user's intention cannot be detected until the desired command flashes. It also requires the user's constant attention to the flashing commands, undoubtedly inducing fatigue.

Another reliable method to detect the user's intent relies on the steady state visual evoked potential (SSVEP) observed when a subject focuses on a flickering stimulus as an increase in the EEG activity over occipital areas at the stimulus frequency. Several successful BCIs have been developed based on the SSVEP (see Vialatte et al. (2010) for a review) with applications including movement control in VR (e.g. Lalor et al., 2005), but so far no SSVEP-based BCI system to control a wheelchair has been proposed.[1] Like the P300 paradigm, the SSVEP allows for a potentially large number of different commands and does not require training, but again requires the user's continuous attention.

Changes in different EEG rhythms can also be induced voluntarily without external aids (Wolpaw et al., 2002) e.g. through the execution of different mental tasks. Popular choices include non-trivial arithmetic, visual counting, 3D object rotation, and, most frequently, sustained motor imagery tasks, such as left vs. right hand, foot or tongue movement (Vallabhaneni et al., 2005). Here, the classification of the user's intention from the EEG signal often requires sophisticated machine learning methods (see Lotte et al. (2010) for a review) and increases in difficulty with the number of commands to be distinguished. Thus, only recently four-command BCI systems of that type have been reported (e.g Wolpaw and McFarland, 2004; Morash et al., 2007), and some

[1] To the best of the author's knowledge

studies explicitly concentrate on developing useful systems utilising only one command (e.g. Velasco-Álvarez and Ron-Angevin, 2009). Additionally, users require substantial training before they can control their EEG effectively, and not all find it possible (Nijholt and Tan, 2008).

At the neurophysiology lab at the University of Strathclyde, we believe that this need for user training will be significantly smaller if the motor imagery task is intuitive and the imagined movement is fast. Conway et al. (2004) demonstrated a relationship between mu rhythms (8–12 Hz) in the EEG recorded over the motor cortex and muscle activity (electromyography, EMG) during rapid point-to-point wrist joint movements in different directions. Given that mu rhythm event related synchronisation (ERS) and desynchronisation (ERD) occur also during hand movement imagination (Pfurtscheller et al., 2006), rapid wrist movements emerge as an intuitive multi-dimensional motor imagery task with potential for use in BCI. Indeed, previous studies succeeded in classifying first 2 (Lakany and Conway, 2007) and later 4 (Valsan, 2007) directions of imagined wrist movement, and up to 20 directions of actual movements (Lakany and Conway, 2005; Worrajian, 2009) based on the readiness (Bereitschaft) potential (Deecke et al., 1969) occurring during the movement-planning phase starting about one second before the actual movement.

Building upon these successes, the objective of this thesis is to build a realistic immersive virtual reality wheelchair driving simulator to be utilised for user training and system development and integrate it with ongoing work on movement intention classification thus closing the feedback loop between the subject and the system (Figure 1.1) and creating a true bi-directional brain-computer interface.

1.3 A Versatile Virtual Reality Wheelchair Simulator

To date, the development of virtual reality wheelchair simulators has been largely motivated by two applications: (i) *clinical* such as training and assessment of patients' abilities where the stress is frequently on engagement and immersion, and (ii) *technical* where motion and control realism as well as sensor simulation are important, as in testing and development of smart wheelchairs. The clinical simulators offer little flexibility to modify the wheelchair model or include sensors, while the testing simulators are frequently graphically too simplistic to provide the degree of immersion necessary for training. For the purpose of BCI control development, however, a more versatile simulator able to fulfil both roles is required. First, a test-bed is necessary to choose the optimal paradigm as well as develop and test the BCI algorithms, including both intention detection and wheelchair control. Hence, the simulation must provide accurate physical behavior and include sensors which the control algorithms may use, as in smart wheelchairs. Then, a training platform is required to fine-tune the algorithms for each patient in order to maximise accuracy and reliability, and provide them with ample time to develop proficiency and security in using the technology.

By marrying a general purpose robotics simulator with a relatively inexpensive immersive display technology, the simulator presented in this thesis satisfies these requirements. Based on a commercial game engine, it runs on a standard PC and provides high-fidelity graphics as well as a software tool to create additional virtual environments (VE). A 3D model of a popular wheelchair with realistic motion and control characteristics has been developed and added to the robotics simulator. An optional large spherical screen provides the user with a feeling of being immersed in the VE, while the underlying robotics software allows for simulation of various standard sensors. The simulator provides a network interface and can be controlled by external applications such as, for example, an EEG-based classifier. A control algorithm based on a finite state machine was developed to allow smooth and intuitive control of the wheelchair using few and infrequent commands, as can be provided by a BCI. Additionally, the simulator can be controlled with a joystick and thus serve as a training aid for prospective wheelchair users. A large VE representing an obstacle course of varied difficulty has been built to test users' performance.

Details of hardware and software implementation of the simulator as well as a review of the state of the art are provided in chapter 2. Chapter 3 describes the wheelchair model and its mechanics. Chapter 4 presents a study of the immersive display system leading to the design of an automated procedure for optimisation of its settings for individual subjects. A novel wheelchair control mechanism for BCIs is proposed and tested in chapter 5. In chapter 6, the simulator is integrated with ongoing work to create the Strathclyde BCI which is then applied to the study of subject learning. Closing the thesis are concluding remarks and suggestions for future work in chapter 7.

1.4 Contributions

The thesis tackles several topics and addresses engineering challenges across different disciplines. Novel **engineering achievements** include:

- Integration, for the first time,[2] of USARSim, a robotics simulator, and a library for spherical projection of images into a single application with an intuitive graphical user interface.
- Modification of VisionStation®, a commercial immersive display system, to allow accurate positioning of its different elements and use by a person seated in a wheelchair.
- Development of a reusable 3D geometric model of a common electric wheelchair and its incorporation as a robot within USARSim, accounting for physical properties including centre of mass, inertia tensor and motor torque.

[2]To the best of the author's knowledge

- Implementation of independent freely rotating, self-aligning castor wheels and a functional kerb climber, features not available in USARSim[3], as part of the wheelchair model.
- Development of a virtual environment for training and testing, including features such as turns of varied degree, doors, tight passages, a kerb and a ramp.

While major **contributions to knowledge** include:

- Integration of human body segment data from different studies to provide a full set of measurements.
- Development and implementation of a novel method to calibrate a wide-angle projector lens using a theodolite and image registration techniques.
- Development of a measure to quantify the correctness of the geometry displayed on the immersive screen and implementation of a procedure to optimise it for individual subjects.
- Development and implementation of a novel algorithm to control a wheelchair with sparse discrete commands, tested by a group of 10 healthy subjects.
- Integration of the simulator with ongoing work on EEG-based motion intention classification, resulting in a complete bi-directional brain computer interface system, the Strathclyde BCI.
- Demonstration and quantification of the impact of continuous performance feedback on learning and classification rates using the aforementioned VR simulator.
- Development of a novel algorithm for detection of movement onset from EMG recordings.

The work presented here has so far led to two peer-reviewed publications describing the Strathclyde BCI (Valsan et al., 2009) and demonstrating the learning effect afforded by continuous performance feedback provision as measured by BCI classification rates (Grychtol et al., 2010).

[3]USARSim version 3.1.3

Chapter 2

Immersive Powered Wheelchair Simulator

2.1 Introduction

The need for virtual reality (VR) electric powered wheelchair (EPW) simulators has been identified already in the 1980s (e.g. Crocker and Turner, 1988) when the civilian applications of the technology were picking up. Chiefly, it is motivated by two main applications or scenarios of usage. On the one hand, due to their high cost, public funding is frequently sought for power wheelchairs. A demonstration of wheelchair driving proficiency is often required for a successful application. However, training to acquire the requisite skills is difficult and costly, as the only available method is to explore the behaviour of a basic training chair under the supervision of a specialist in a vast uncluttered space at first, and progressively increasing the complexity of the environment. This approach is very time consuming while the learning process is often hindered by the inadequate seat and controls positioning and the training chair's lack of access adaptations. Here, a VR simulator could provide additional training time, speeding up the procurement of power wheelchairs and lowering the cost of the process. On the other hand, advances in robotics and sensor technology find their obvious application in the so-called "smart" wheelchairs. The purpose of these semi-autonomous vehicles is to take over the more low-level tasks of controlling the wheelchair from patients who find conventional control methods difficult or impossible, thus increasing their safety and independence. However, many tests are required to find the optimal sensor arrangement and to prove the safety of the algorithms, which can be very costly and time consuming. Here, again, a VR simulator could lower the costs and accelerate the development by offering a safe and flexible test-bed for new technologies.

In the past, simulators were specifically designed to address either of these needs,

but rarely both. The training simulators were not flexible enough to easily modify the wheelchair model or include sensors, while the testing simulators were frequently graphically too simplistic to provide the degree of immersion necessary for training. For the purpose of BCI control development, however, both scenarios are applicable, and hence a versatile simulator is required, able to act as a testbed for the developing technology and a training platform for its prospective users.

This chapter describes the immersive VR powered wheelchair simulator developed at the University of Strathclyde for research and training applications alike, which is the main contribution of the thesis. In the following section, a set of requirements identified for both applications is given. This is followed by an overview of the various simulators found in the literature. Subsequent sections detail the design and implementation of the system, including a bespoke virtual environment. Section 2.5 is devoted to performance considerations, while section 2.6 presents results of a questionnaire evaluation by 11 able-bodied first time users of the system. Detailed account of how each design requirement was addressed is offered in section 2.7 together with a discussion of some limitations. Closing the chapter is a short summary.

2.2 Design Requirements

Following are the design requirements identified for a VR wheelchair simulator to be used both in research of smart wheelchairs and as a training tool for prospective wheelchair users. In developing the requirements four *roles* have been taken into account:

- **patient** who uses the simulator for training or testing
- **clinical user** who administers the training or testing session to the patient
- **researcher** using the simulator for development and testing of new technologies such as smart wheelchairs or brain computer interfaces
- **developer** charged with maintaining and expanding the system in future work at the Neurophysiology Lab at University of Strathclyde.

1. **User Requirements**
   Users should be able to:
   a) use the simulator for training (patient, clinical user)
   b) use the simulator for testing new technologies (researcher)
   c) use the simulator in the clinical environment (patient, clinical user)
   d) use the simulator with multiple patients (clinical user)
   e) analyse patients' performance (clinical user)
   f) use the simulator at home (patient)

g) use third party control devices (patient, clinical user)

h) use with own virtual environments (clinical user, researcher)

i) change aspects of the simulated wheelchair (researcher, clinical user)

j) control the wheelchair through external applications (researcher)

k) repeat the same test (clinical user, patient, researcher)

l) analyse performance of third party control algorithms (researcher)

m) maintain the simulator and extend its capability (developer)

2. **Quality Attributes:**

a) **Portability** The portability requirements of the two applications are as different as the understanding of the term by the two target communities. In the clinical setting, portability means ease of transport and deployment. In the (robotics) research community it more often refers to the compatibility of software with different operating systems. Since the Microsoft® Windows® family of operating systems (OS) is vastly predominant in the clinical setting, at times the requirements of the two applications are in contradiction, in which cases the clinical application, as the less flexible, should take precedence.

b) **Ease of use** Again, this requirement is satisfied differently for each application. For the clinical user, the simulator has to provide a simple graphical user interface (GUI) where most common options can be selected and training promptly started. To satisfy the requirements of the researcher, a simple (preferably text-based) interface must be provided to all the advanced features of the simulator.

c) **Degree of immersion** The degree of immersion is of great importance for the training outcome. Although no study has fully investigated the relationship between the degree of immersion and the training outcome, it is intuitive that a wealthier environment with more cues and closer resemblance to reality should lead to better training outcomes. LU et al. (2007) found, for instance, that including sound effects in the simulation improves the training outcome. However, a balance must be achieved, as prolonged usage of an immersive simulator may cause cyber sickness.

d) **Realism** Equally critical for the simulator's utility both as a training tool and as a development aid is realistic simulation of the wheelchair's control dynamics and physical behavior as well as data input from various sensors (including noise).

e) **Maintainability** For the benefit of the future developer, the simulator software must be implemented in a popular object oriented programming lan-

guage allowing modular architecture. The source code must be properly structured and well documented.

3. **Functional Requirements**
   The simulator shall perform the following (references to the corresponding user requirements (1a–m) and quality attributes (2a–e) are given in parenthesis):

   a) simulate wheelchair movement in a physically realistic fashion (2d)

   b) simulate real-time sensory input (1b, 2d)

   c) simulate sonar, laser, touch-sensitive and camera sensors (1b,1i)

   d) log simulation parameters for offline analysis:: wheelchair position at 1 Hz and all sensor output (1b, 1l)

   e) log subject performance: path, collisions, and all subject input (1a, 1e)

   f) display VE using immersive display technology (1c, 1a, 2c)

   g) display geometrically correct images when using immersive display technology (2c, 2d)

   h) display VE using standard displays (1b, 1f)

   i) update the displayed image at least 30 times per second (1a, 2c, 2d)

   j) accept and interpret input from third party wheelchair controllers (1g)

   k) expose an interface for third party applications (1j, 2a)

   l) accept new VEs developed externally (1h)

   m) simulate a number of typical wheelchair models (1a, 1b, 1d, 1i)

   n) allow modification of simulation parameters through text-based configuration files (1b, 1i, 2b)

   o) provide a graphical user interface (GUI) (1a, 2b)

   p) allow modification of clinically-relevant parameters through the GUI (1a, 1d, 2b)

   q) allow modifying display settings to maximise the immersive effect for different viewing positions (1d, 2c)

   r) run on Windows XP®[1] (1c, 2b)

[1] In 2007, when these requirements where drawn up, Windows XP was the most widely used Windows operating system

4. **Non-functional Requirements**
   The hardware shall be:

   a) fitting in a van (e.g. VW Transporter) (2a)

   b) light enough to carry by two people (2a)

   The software shall be:

   c) extendible (1i, 1m)

   d) modular (1i, 1m)

   e) well documented (1m,2e)

5. **Constraints**

   a) The simulator shall use mainstream computer hardware, i.e. no purpose-built pieces (1c, 1f)

   b) The design must minimise cost. In particular, the simulator, or a limited version of it, should be inexpensive enough (less than £200) to be purchased by the patients so that they can train at home (1c, 1f)

Ideally, in drawing these design requirement, representatives of the four user groups should have been consulted, but lack of ethical approval prevented contacting patients and no prospective developers were available. However, once the simulator was built, its design requirements and their fulfilment were assessed by Dr. Sujay Galen, a practising physiotherapist with research expertise working with SCI patients, and Dr. Mariel Purcell, Consultant in Spinal Cord Injury, Queen Elizabeth National Spinal Injuries Unit, Southern General Hospital, Glasgow. Dr. Galen believes that "a VR wheelchair simulator has multiple benefits in a clinical environment, both from a patient training and research perspective." He concluded that "the design requirements presented have addressed the clinical challenges effectively and therefore the VR wheelchair simulator has the potential to be used in different clinical conditions and in patient populations with severe and mobility problems." His full report as received on 13 July 2011 is included in Appendix D. However, Dr. Purcell expressed doubts about the ability of SCI patients with autonomic dysfunction to cope with the potential cyber sickness and suggested that "users with postural hypotension may experience nausea and dizziness and those with dysreflexia, headaches" (pers. comm., 5 August 2011). She proposed to incorporate a blood pressure/pulse monitor in the design to monitor the patients.

2.3 State of the Art

Broadly, a VR simulator has three main components: an input device to control the actor(s) in the virtual environment (VE), software that maintains the state of the

actors and the VE, and a number of output devices. The latter can include visual displays, speakers, force feedback systems, motion platforms or haptic interfaces. Based on the complexity of the VE maintained by the software component, simulators are divided into two- and three-dimensional ones. Within each class, the training simulators differ mainly in the degree of immersion afforded by the output devices, while research simulators distinguish themselves by their ability to simulate sensors and details of implementation.

2.3.1 Training Simulators

The first wheelchair simulators appeared in the 1980s (e.g. Pronk et al., 1980; Crocker and Turner, 1988). They were simple 2D animations offering a birds-eye planar view of a wheelchair-like shape. Crocker and Turner's (1988) simulator was slightly more advanced of the two and featured a fixed wheelchair against a moving background, limited sound effects and two training scenarios that exercised basic joystick control ability. Although the authors reported unanimous positive feedback from users (school pupils), no study into the efficacy of the training was conducted. This was later done by, among others, Hasdai et al. (1998), who reported significant improvements in wheelchair driving performance of inexperienced wheelchair users following simulator training, as measured with a specifically designed functional evaluation rating scale. However, despite the training, inexperienced users did not achieve the performance of experienced users, which the authors attributed to the over-simplistic nature of the 2D simulation. Nonetheless, some groups (e.g. Cooper et al., 2005) still use 2D simulators that, despite the technological advances, are not substantially different from the one of Crocker and Turner (1988), save for larger displays. As discussed in their original study, although definitely less realistic than their 3D counterparts, 2D simulators have the advantage of visualising at all times the outline of the wheelchair and any obstacles behind and on the sides of it. This, they argue, is unattainable in a 3D simulator, unless a multiple screen system or a head-mounted display (HMD) is used.

The first 3D virtual reality wheelchair simulators appeared in the early 1990s, as the one constructed by a group at the Ohio State University (Swan II et al., 1994; Carlson et al., 1994). They have identified four uses of a simulator, namely (1) the evaluation of a patient's ability to drive a wheelchair, (2) the choice and fitting of custom control systems, (3) the training of inexperienced users, and (4) the evaluation of wheelchair accessibility of architectural designs. Their "instrumented chair" was interfaced with a PC through a serial cable and supported a wide range of control systems, such as joystick, chest muscle actuator, sip and puff device etc., that used the same interface. The 3D effect was enhanced by stereoscopy implemented on a dual-scan stereo monitor and accompanying synchronised stereo viewing glasses. The authors acknowledged the advantages of HMDs in their ability to track head movements and provide peripheral vision but preferred the dual-scan solution due to its lower cost

and lesser complexity. Wheelchair movement was derived from the turning speed of the wheels as *intended* by the controller on-board the chair. Hence, although turning radius and wheelchair velocity were accurately reflected in the simulation, it did not account for effects of inertia or friction. Since only very limited user trials (one subject) are reported (Swan II et al., 1994), no conclusions can be drawn about the efficacy of the system. However, although the level of graphical detail was very high, the small screen and lack of peripheral vision surely limited the applicability of the system.

A concurrent simulator developed at Oregon Research Institute (Inman et al., 1994; Inman and Loge, 1995; Inman et al., 1997) addressed these shortcomings. The wheelchair's rear wheels were placed on rollers which allowed the system to continuously monitor the real speed of the wheels. To introduce inertial effects, the rollers were filled with ballast, resulting in realistic acceleration and breaking. Friction on different surfaces was modelled algorithmically. An advantage of this solution is that the system can be used with any wheelchair, including those with custom control mechanisms and manual wheelchairs, but it makes realistic simulation of a crash difficult, since motors cannot be stopped. The simulator was constructed as a more motivating alternative to conventional wheelchair training for children and featured three VEs of increasing complexity. The use of a HMD equipped with headphones featuring 3D sound provided very high degree of immersion, even by current standards. However, the realism of the VEs was limited by performance considerations, especially in the more complex environments. Importantly, the authors found that the children's driving skills increased as a function of time spent using the simulator. This finding was confirmed by Adelola et al. (2005), who reported lasting transfer of skills from VR to real life in some of the 28 participants in their study. The outcome of training on their simulator (Desbonnet et al., 1998) was measured using Conjoint Analysis (Adelola et al., 2003). Inman et al. (1997) also reported that children generally preferred to look at a large screen on which the 3D environment was displayed, rather than use the HMD. This preference was also expressed by Niniss and Nadif (2000), who were using a HMD to simulate a smart wheelchair (Bourhis and Agostini, 1998), where they wanted to "get free of the helmet by projecting the simulation on a giant screen".

Following this trend, Harrison et al. (2000; 2002), a group based in London, used a 17 inch monitor to display their 3D virtual models of two hospital areas. Their simulator was a training aid for patients with neurological impairments, and concentrated as much on route finding as on the ability to operate the chair. A wheelchair joystick connected to the PC running the simulation was used as an input device. An interesting feature of this VE was the attempt to include simulated people. A number of avatars moved around the environment on predesignated paths and could be stopped by the participants. The results of their multi-stage study on both experienced and novice wheelchair users, who performed the same tasks in both VR and real life, revealed that navigating in VR was consistently more challenging, as measured by the number of collisions and the time to complete the task, and as reported by most users. Experienced

users identified the lack of peripheral vision as one source of the difficulty. But, as no attempt at physically correct movement modelling is reported, limited motion realism afforded by the simulator VR could have also been a factor.

Building upon the many years of collective experience of the research community in the field, Niniss and Inoue (2006) in Japan constructed what the author believes to be the most advanced in terms of its realism electric power wheelchair simulator to date. The 3D virtual environment is a high fidelity model of a real hospital area. It is displayed using four separate projectors onto a hemispherical screen with 110° horizontal field of view (FOV), providing a highly immersive effect. The display system was designed following an experiment in which the minimal FOV required to drive a wheelchair was estimated by tracking the head movements of a healthy volunteer driving a wheelchair through various environments. This was found to be $\pm 70°$ horizontally and $+65°;-70°$ vertically (Niniss and Inoue, 2006). The system includes an actual wheelchair seating system and supports a range of input devices from commercial joysticks through control pads to specialised control systems interfaced via an external input. The screen and the seat are both part of one structure resting on a 6 degrees of freedom (DOF) motion platform. The latter provides added motion realism and simulates vibrations and crashes. Since the main purpose of the simulator is to evaluate a person's ability to drive a wheelchair, it is also equipped with several cameras recording the participant and a data logging capability. Applying fast Fourier transform (FFT) to the recording of joystick position throughout the experiment, the authors revealed a difference in usage patterns between skilled and unskilled users (Niniss and Inoue, 2005), where the skilled users made slower joystick movements.

Although the simulator of Niniss and Inoue (2006) is extremely realistic in terms of graphical detail and provides a high degree of immersion, it has several practical disadvantages. First of all, it cannot be used with an actual wheelchair as a seat is already mounted on the motion platform on which the rest of the system rests and it is unclear how a disabled person is to climb the motion platform. Second, there is no evidence that the model of wheelchair movement is physically accurate, although it can be presumed that some inertial effects were calculated to control the motion platform. The authors also concede that vertical FOV of the display may be insufficient. They also reported a problem with cyber sickness. Last but not least, the four-projectors system is undoubtedly rather costly, and, together with the motion platform, difficult to transport. Overall, the system's utility as a training simulator is limited.

Finally, a simulator similar in aims to the one described in this thesis in that it addresses both the need for training and for simulating sensors is the ISIDORE system developed at Toulon University (Randria et al., 2009). It seeks to assess the users' driving ability and need for assistance from an autonomous wheelchair equipped with up to 6 distance sensors by comparing their travelled trajectories to optimal paths derived with various algorithms. User input is obtained indirectly through incremental encoders in touch with the wheels and thus the system can cater for all types of control method.

However, with the rather simplistic virtual environment being projected on a flat screen above the user's eye level, it is questionable how well the measured parameters reflect the users' true driving skills and ability to judge distances.

Thus, the simulator proposed here will use an approach similar to that of Niniss and Inoue (2006), in that the VE will be projected on a hemispherical screen, but using only one projector. This is preferred over the more immersive cave-like virtual environments that project the VE on all walls of a room (e.g. Browning et al., 1994), due to their high cost and complicated setup.

2.3.2 Smart Wheelchair Simulators

There are far too many groups developing some kind of smart wheelchairs to mention them all. Surprisingly few, however, employ a simulator for testing or development. Those that do, usually develop their own simulators, almost always designed specifically to model a particular wheelchair prototype or algorithm. An example is the simulator developed by Niniss and Nadif (2000), for the VAHM wheelchair (Bourhis and Agostini, 1998), where the user is immersed into a 3D virtual environment displayed on a HMD and simulated proximity sensor readings are passed to the VAHM robot. Many of the other simulators are 2D (e.g. Sgouros, 2002; Hamagami and Hirata, 2004; Špacapan et al., 2004) or do not include a user in the simulation (e.g Ojala et al., 1991). There is, therefore, a clear need for a general-purpose high-fidelity wheelchair simulator for the robotics community.

A natural way to build such a wheelchair simulator is to extend one of the freely available general robot simulators, such as SimRobot (Laue et al., 2006) or USARSim (Carpin et al., 2007) (see Laue et al. (2006) for more examples). SimRobot has been developed at the University of Bremen, Germany, and has previously been used to simulate the Bremen Autonomous Wheelchair (Röfer, 1998). It is a 3D "general physical robot simulator" (Laue et al., 2006) and includes models of several actuators and a few general classes of sensors. Specific sensors still have to be written by the user. SimRobot has several advantages, including a user interface to interact with the running simulation and the ability to specify new robots and environments completely externally to the main application (using an XML-based script file). However, there seem to be no straightforward way of extending its graphics capabilities without modifying the source code. Moreover, manually describing complex environments as text files seems cumbersome. Due to these limitations of SimRobot, USARSim was preferred as the base of the simulator presented here.

USARSim is unique among robotics simulators in that it is based on a commercial video game engine. Game engines are designed to separate the underlying technology from a game's content and allow the implementation of multiple games re-using the same engine, thus justifying the huge development costs necessary to reach the level of realism today's video games possess. Their relatively low hardware requirements,

client–server architecture and flexibility, combined with high fidelity graphics and physical simulation, make game engines particularly well suited for scientific applications (Lewis and Jacobson, 2002). USARSim is based on Unreal Engine 2.0 produced by Epic Games and supplied with the *Unreal Tournament 2004* (UT2004) game. It is an inexpensive game engine implementing *Unreal Virtual Machine*, similar in concept to the Java Virtual Machine. It provides a level editor (*UnrealEd*) and a scripting language (*UnrealScript*) similar to C++ and Java, in which custom simulation elements can be written. "USARSim in itself is a set of models and classes defining the simulations of robots, sensors and actuators" (Carpin et al., 2007). USARSim was originally intended to simulate Urban Search and Rescue, hence the name. Thanks to wide community support,[2] it is now re-branded as a general-purpose simulator (Unified System for Automation and Robot Simulation), featuring numerous sensors and actuators, and a range of complete robot models. The hierarchical structure of USARSim classes makes it relatively easy to develop new models (Wang and Balakirsky, 2007). Another advantage of USARSim is its ability to interface with *Player*, a popular control middleware for real and simulated robots (Gerkey et al., 2003), and Mobility Open Architecture Simulation and Tools (MOAST) framework, a fully-functional hierarchical control system (Balakirsky et al., 2005; Scrapper et al., 2006). Most importantly for the application intended here, USARSim implements multiple robot cameras. This gives the basis for providing an immersive VE for the wheelchair simulator, taking advantage of the high performance graphics afforded by the game engine.

2.3.3 Simulators for BCIs

As mentioned in chapter 1, virtual reality is extensively used in BCI research. While many systems have been built to simulate wheelchair movement for the purpose of BCI experiments (e.g. Bayliss and Ballard, 2000; Leeb, Lee, Keinrath, Scherer, Bischof and Pfurtscheller, 2007), few actually simulated any physical characteristic of the wheelchair, its motion or control. One exception is the simulator built within the MAIA project (Galán et al., 2008), which simulates the MAIA wheelchair and its sensors in a simple virtual environment displayed on a computer screen. A more versatile system has recently been described by Gentiletti et al. (2009). Like that described in this thesis, it is based on a commercial robotics simulator allowing for accurate simulation of motion and sensors. However, its use as a training simulator is limited by the lack of an immersive display system.

2.3.4 Other Applications

Although outside the scope of this thesis, for the sake of completeness it should be mentioned that VR is also used to simulate manual wheelchairs (see Pithon et al.

[2] http://sourceforge.net/projects/usarsim/

(2009) for a review). Such simulators often find their use in motor rehabilitation (see Holden (2005) for a review). They can serve as a motivation to exercise (e.g. Cooper et al., 2005) or, less frequently, as training platforms (e.g. Secco et al., 2007). In general, though, their motion realism is limited as a manual wheelchair's motion depends on the changing position of the user more than that of its electric powered counterpart.

As outlined by Carlson et al. (1994), manual wheelchair simulators are also useful for assessing the accessibility of buildings and vehicles, and as an aid in their design. One such simulator is that constructed by Yamada et al. (2003). It is a 3D manual wheelchair simulator featuring a HMD and a motion platform. It has been employed to study barrier-free transport aboard a ship. Another noteworthy simulator is that constructed previously at the University of Strathclyde (Grant et al., 2004; Harrison et al., 2004), featuring a giant screen and wheel-supporting rollers with motors and breaks that realistically simulated driving a manual wheelchair on inclines and carpets. The simulator was used to assess the accessibility of buildings.

2.4 Implementation

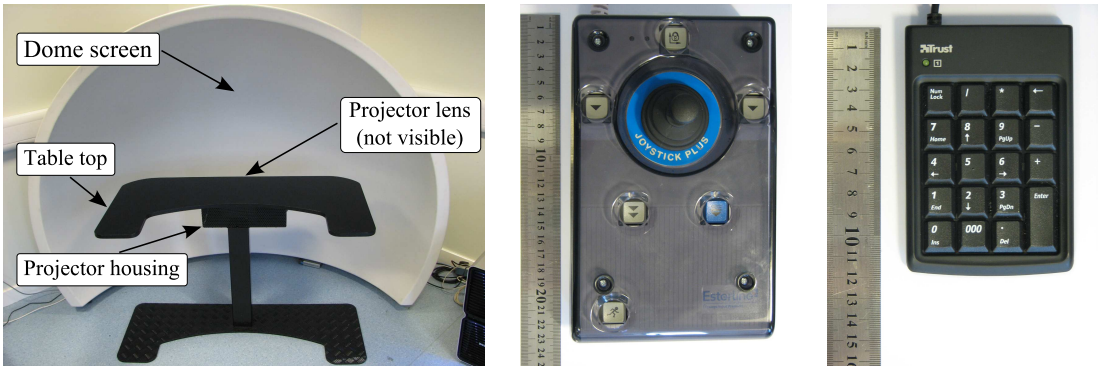
The simulator presented in this thesis provides both a flexible training system for EPW users and a versatile platform for the development and testing of smart wheelchairs. This is afforded by combining the advantages of USARSim and an immersive display system respectively for robot simulation and user training.

The following subsections describe in detail the implementation of the simulator. The design choices are discussed in light of the requirements outlined in section 2.2 at the end of the chapter.

2.4.1 Hardware

The simulator setup currently includes a workstation PC with a gaming-standard graphics card, an immersive display system, and a joystick similar to those installed on wheelchairs.

While considering different display systems, the use of head-mounted displays was excluded due to their high cost, patient preference (Harrison et al., 2002), and possible interference with BCI. Similarly, the use of multi-projector systems was excluded due to both their high cost and difficult maintenance. Thus, the simulator features an Elumens® Vision Station® — a 2m in diameter dome-shaped screen illuminated by a single media projector fitted with a wide-angle lens (figure 2.1a) — providing a 160° field of vision (horizontally). The maximum resolution supported by the system is 1024×768 pixels. The simulator can be controlled with any PC-compatible joystick, a mouse (or a mouse replacement device) or the keyboard. 3D joysticks with throttle control are supported. The throttle slider controls the sensitivity of the joystick and the maximum speed. To provide a familiar interface to wheelchair users, the simulator



(a) VisionStation® (b) Traxsys Joystick (c) Keypad

Figure 2.1: Hardware components of the simulator.

also features the Traxsys Joystick Plus that closely resembles wheelchair joysticks in feel and appearance (figure 2.1b). The keyboard mode of control was included to test a novel wheelchair control algorithm, described in chapter 5, designed to work with the type of discrete commands that could be available in a BCI system. The keypad used for these tests is depicted on figure 2.1c

The workstation runs Windows® XP Service Pack 2 and includes the following hardware:

- Intel® Core™ 2 Duo 6600 (2.40Hz) processor
- nVidia® GeForce® 8800 GTX graphics card
- Ageia® PhysX™ AG1011 Physics Processor

The physics processor is not currently utilised, but will become useful once USARSim is fully updated to use the latest Unreal Engine.

Projector Mount

As is evident from figure 2.1a, one disadvantage of the VisionStation® is that, because of the column supporting the projector, wheelchair users would not be able to drive up close enough to the dome to experience the full immersive effect. Moreover, it is difficult to maintain optimal positioning of the picture on the screen, as the projector housing can easily move with respect to the dome screen, and so can the projector within the housing.

To address these problems, a new projector mount was designed and built, as depicted on Figure 2.2. It consists of a large aluminium platform to which two poles are affixed. The projector is mounted on a horizontal bar between the poles such that it can tilt up and down, with the amount of tilt indicated on the protractor affixed on the bar below the projector. The height of the horizontal bar can be adjusted with the help of the rack and pinion system on the left pole. The projector can be levelled using the spirit levels affixed to its top. Two rails on the platform guide the wheels

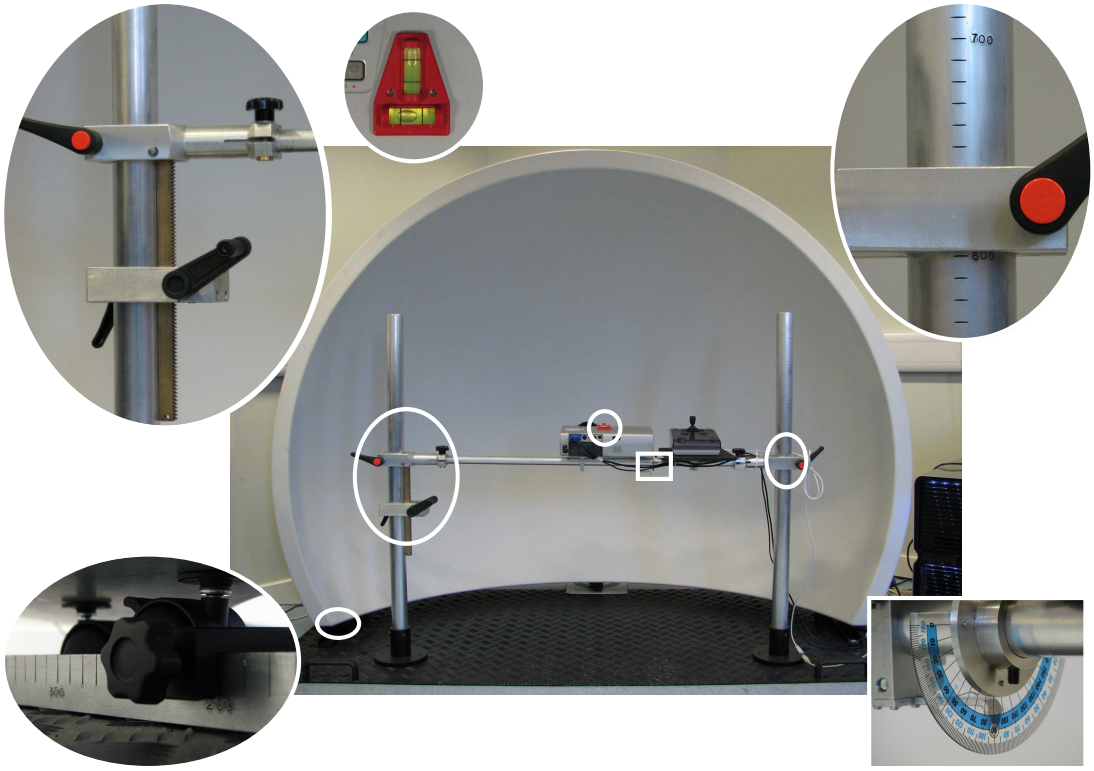


Figure 2.2: New projector mount allowing precise positioning of the projector with respect to the dome and its features: a rack and pinion system for projector height adjustment (top left), etches on the right pole to read out the current height (top right), a protractor to read out the projector tilt (bottom right), etched rails to guide the dome in place (bottom left) and two spirit levels on top of the projector (top centre).

of the dome, such that it can only be moved back and forth, and stoppers are placed along the rails to control the distance from the projector to the screen. A plywood plate on which control devices can be placed is also affixed to the horizontal bar next to the projector. All parts beside the poles were painted black in order to minimise reflections. The entire structure can be easily disassembled for transport.

The current position of the dome and the height of the horizontal bar can be read from the etches on respectively the guide rails and the right pole, as indicated on figure 2.2. However, more important for correct spherical projection of images on the dome screen are the height of the lens and the distance between the lens and centre of the sphere of which the screen is a section. Those measurements have been calibrated using a theodolite (described in chapter 4) and are presented in Table 2.1.

2.4.2 Software

The simulator software involves three main elements: the USARSim simulator, a library for spherical projection of images (SPI) provided by Elumens® (SPIClops), and a user interface developed by the author. These components are integrated into a single

Table 2.1: Calibration of projector mount distance readings.

Measured distance d	Desired distance	Correction [m]
Dome wheels to poles	Lens to dome edge	$d + 0.054$
	Lens to dome centre	$0.496 - d$
Bar height (read above)	Lens to floor	$d + 0.091$
	Lens to dome centre	$d - 0.479$

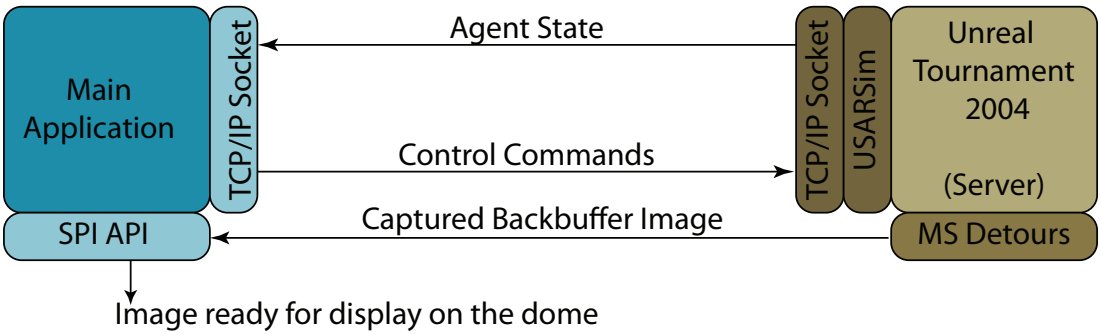


Figure 2.3: Application logic. UT2004 with USARSim act as a server communicating with main application through a TCP/IP socket. Microsoft® Detours technology is used to capture the backbuffer images from UT2004. These are passed to SPIClops for stitching (see Figure 2.4).

application implemented as a Visual C++ project developed in Microsoft® Visual Studio® 2005, using the Microsoft Foundation Class library (MFC). The restriction to the Windows® operating system that this choice of development environment imposes on the resulting application was not considered problematic, due to the popularity of the OS in the clinical environment and a limitation to the same effect in the USARSim simulator (discussed below).

The application logic is depicted on Figure 2.3. The virtual environment is maintained by the Unreal Engine, an integral part of UT2004, and includes USARSim models. UT2004 acts as a TCP/IP server to which the main application connects. The TCP/IP channel is used for exchange of control commands and agent state(s) between the client application and the server. This architecture allows for multiple users to interact in the same VE. Four simulated robot cameras supply orthogonal views from the wheelchair (Figure 2.4A). These are captured from the backbuffer of UT2004 using Microsoft® Detours technology (Hunt and Brubacher, 1999) to be stitched using the SPIClops API provided by Elumens® to provide a spherically correct image ready for display on the dome (Figure 2.4B). The use of Detours technology is the factor that limits the use of USARSim simulated cameras to the Windows® OS. However, only the UT2004 server needs to run on Windows®, as captured frames can be transmitted over TCP/IP.

The user interface (Figure 2.5) was designed with the clinical user in mind, since

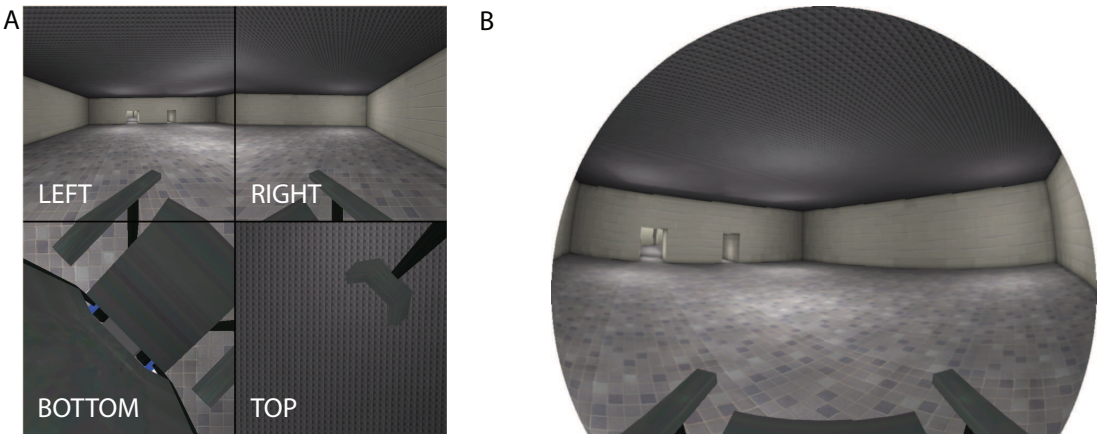
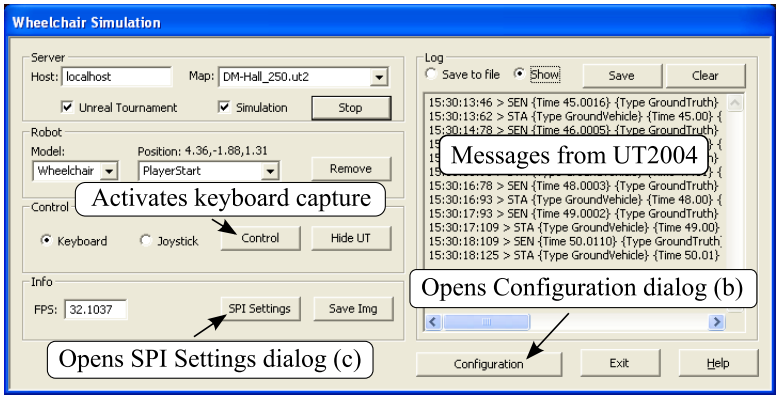
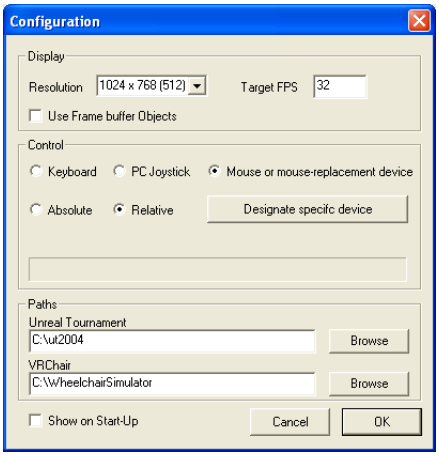


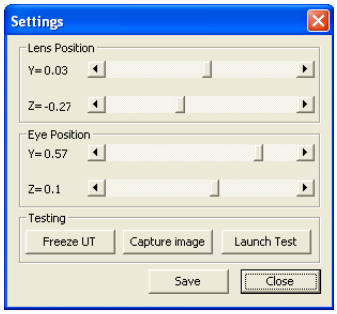
Figure 2.4: *A*: Multiple views from the wheelchair; *B*: Spherically correct image ready for projection on the dome.



(a) Main window



(b) Configuration dialog



(c) SPI Settings dialog

Figure 2.5: Graphical User Interface.

researchers in the field of robotics are expected to build their own controllers or use Player or MOAST to interact with the simulation. Therefore, it provides only basic functionality like choosing the virtual environment (map) and the robot model, starting and stopping the simulator, choosing a control device or adjusting spherical correction parameters. Default values for the choices can be set in an XML configuration file read by the main application.

In the spirit of object-oriented programming, the simulator application was developed as several independent classes, with all functionality concentrated in a single class independent of the GUI. Brief description of the most important classes is given below. Full documentation and source code of the software are presented in Appendix A.

CWheelchairApp (derived from CWinApp) is the main class of the application. It interacts with objects of all the other classes and provides all the application's functionality. Notably, it implements keyboard and joystick control, and provides interface for interacting with Unreal Tournament and USARSim. It also implements a standard Win32 timer, which is responsible for the continuous update of the images displayed to the user. This is achieved by periodically calling the associated `TimerProc()` function which retrieves the backbuffer image from UT2004 and calls the `Draw()` function of the CSimWnd class, where the image is stitched and subsequently displayed. The target update frequency can by set by changing the FPS (frames per second) parameter in the configuration file.

CWheelchairDlg (derived from CDialog) implements the main dialog of the graphical user interface (Figure 2.5). It is the parent window of all other dialogs and windows of the application.

CSimWnd (derived from CWnd) implements the window that displays the Virtual Environment. It initialises an OpenGL rendering environment and implements the `Draw()` function which uses the OpenGL version of SPIClops to stitch the frames and subsequently refresh the display.

CStateControl implements the finite state machine (FSM) behind the keyboard mode of control, described in full in chapter 4.

CSettingsDlg (derived from CDialog) implements a small dialog box that pops up when the "Settings" button is pressed on the main GUI. It allows to modify in real time the lens and viewpoint settings that affect the internal geometry SPIClops uses for computing the final image. Once optimal settings have been found, these can be stored in the application's configuration file.

CAppConfig implements an interface for accessing the application's configuration file. It is the only non-native (managed) class of the application — it employs elements of the .NET framework and hence uses C++/CLI.

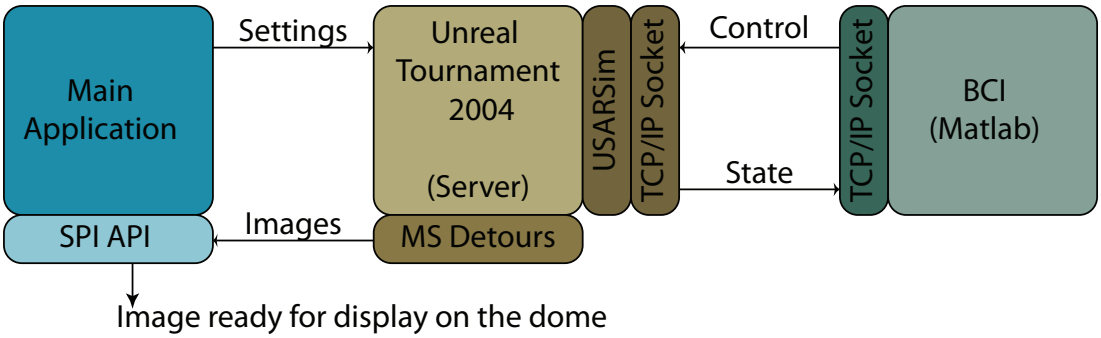


Figure 2.6: Appication logic when controlled from Matlab®. The wheelchair is controlled through a bespoke Matlab® toolbox which also receives all sensor input from USARSim. The main application maintains the UT2004 server and provides a video stream to display on the dome (c.f. Figure 2.3).

Additionally, to integrate the simulator into ongoing work on motion intention detection from EEG signals, a modular class-based toolbox has been developed allowing control of the wheelchair from Matlab®. The toolbox communicates directly with UT2004 via a TCP/IP socket as implemented in the Instrument Control toolbox and provides a logging capability. Thus, when using the toolbox, the role of the main application is reduced to only maintaining the UT2004 and processing its video stream for projection on the dome. This usage scenario is depicted on Figure 2.6. In future, thanks to the modular nature of the toolbox, the default FSM-based discreet command controller (described in chapter 4) can be replaced with one taking into account the sensor input and thus implementing autonomous behaviour.

2.4.3 3D Wheelchair Model

To ensure maximum realism, a 3D model of an actual wheelchair has been created and added to USARSim (see Figure 2.7). The model is based on the SpectraPlus wheelchair from INVACARE® (Figure 2.7a) which was chosen as one of the more commonly prescribed indoor/outdoor EPWs in Scotland for which relatively good documentation was available. The 3D model was built in Maya 7.0 PLE based on images available on the manufacturer's website[3] and exported to UnrealEd as separate static meshes using the *unEditor* plugin available on Unreal Developer Network[4] (UDN). The model was then added to USARSim by writing appropriate UnrealScript classes and modifying the `USARBot.ini` configuration file. The model features fully autonomous castor wheels, a functional kerb climber and four orthogonal simulated cameras. Although the masses of the different parts of the wheelchair are not known precisely, due to the lack of an actual wheelchair to disassemble and weigh, rough estimates have been made in order to calculate inertia tensors. For that purpose, each part of the model was saved in

[3] http://invacare.co.uk/
[4] http://udn.epicgames.com/Two/MayaCompletePlugins.html

(a) INVACARE® SpectraPlus

(b) 3D model as seen in UT2004

Figure 2.7: Wheelchair model and the real wheelchair (image from INVACARE® product brochure).

Maya as a separate DWG file. These files were then analysed with AutoVue™ software where the volume and inertia tensor of each part was calculated. The inertia tensors were then incorporated into the script files describing each part.

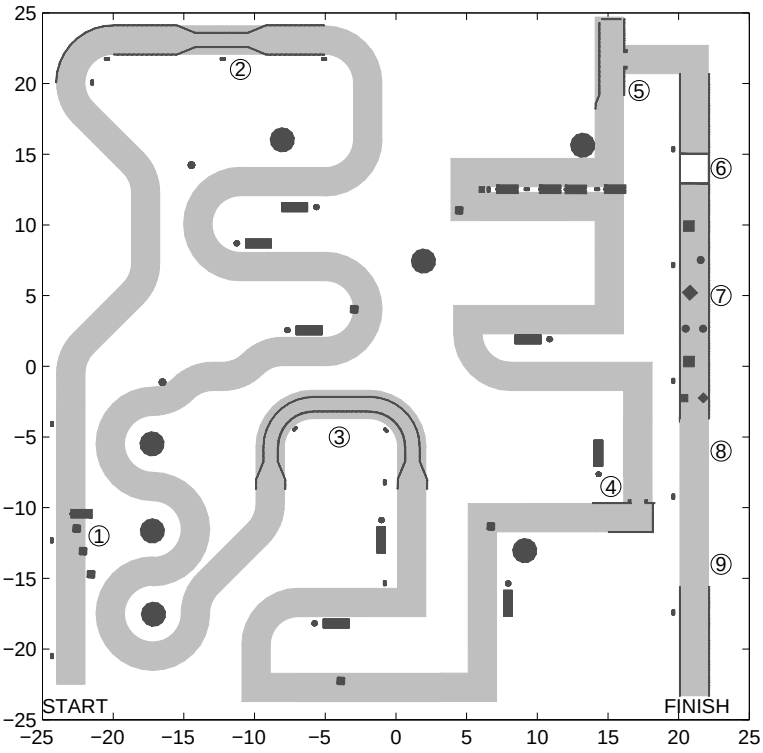
The wheelchair model also features a number of invisible touch detectors developed by the author[5] which allow the simulator to provide audio feedback when collisions occur. All collisions are logged, as are position and velocity provided every second by the USARSim GroundTruth sensor also installed on the wheelchair. Thanks to these sensors, the log file allows analysing the trajectory of the wheelchair, number and location of touches and the frequency of steering commands after an experiment is complete. Several Matlab® scripts have been developed to process and visualise this information and are included in Appendix A.

The design and implementation of the wheelchair model are detailed in chapter 3, including the derivation of inertia tensors, description of each class, as well polygon counts for each part of the model.

2.4.4 Virtual Environment

Two training environments have been built for use with the simulator. The first is a large ($10 \times 10 \times 3$ m) empty hall intended for subjects to get acquainted with the wheelchair's control and motion or for experiments that require the absence of visual stimuli. The second VE is a track with turns of varied curvature, narrow passages and other obstacles, see Figure 2.8. The track is 2 m wide and approximately 360 m long, contained within a 25×25 m hall. The course includes two narrow (1 m) doors, a 7 cm

[5] Although USARSim provides simulated touch sensors, they do not detect collisions with all objects



1. Traffic cones
2. Narrow (1 m) passage
3. Narrow (1 m) passage (with turns)
4. Door (1 m wide)
5. Door (1 m wide) with less space
6. Kerb (7 cm high)
7. Obstacle course
8. Ramp up (10°)
9. Ramp down (10°)

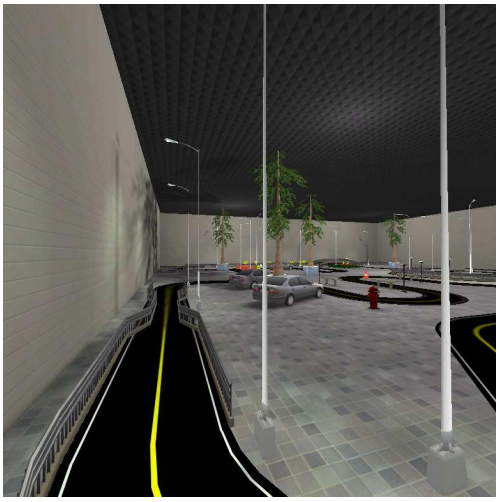
Figure 2.8: Map of the training track.

kerb and a 10° ramp, as well as several traffic cones placed on the track and a difficult obstacle course. Several common outdoors objects (street lamps, cars, benches, trees, etc.) have been placed in the VE to enhance the perception of motion and provide a size reference. Figure 2.9 depicts some of the more interesting features of the VE. In general, the difficulty of the track increases progressively from start to finish. However, the wheelchair can be placed in various places on the map in order to allow practising individual segments of the course.

2.5 Performance

The simulator comfortably achieves the target frame rate of 30 fps and the resolution of 1024×768. However, an important distinction is that between resolution and definition. The latter refers to the quality of the resulting image, which increases with the resolution (size) of the textures from which the final image is stitched for spherical projection. Figure 2.10 presents three images stitched using textures of different sizes. The maximum texture size is limited by the size of the backbuffer provided by the operating system and the graphics card. Most computers will support using 512×512 pixel textures (so a UT2004 window containing the four views slightly in excess of 1024×1024). Powerful multi-display graphics cards could support higher resolutions. By arranging two screens in a diagonal configuration (Figure 2.11), a large enough backbuffer to support 1024×1024 textures was successfully obtained. However, this

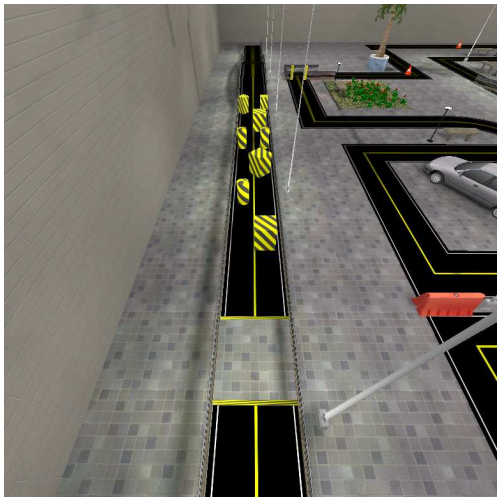
(a) View from the starting point



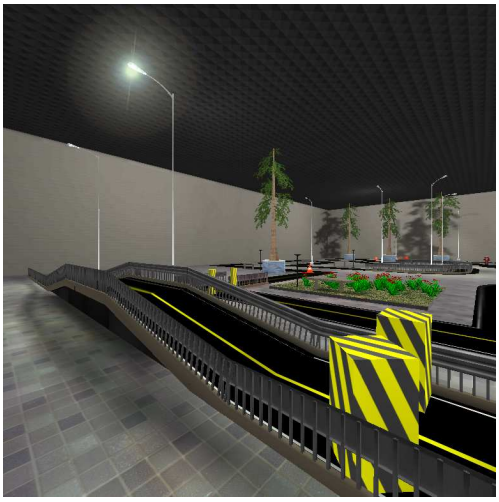
(b) Narrow passage

(c) First door

(d) Kerb

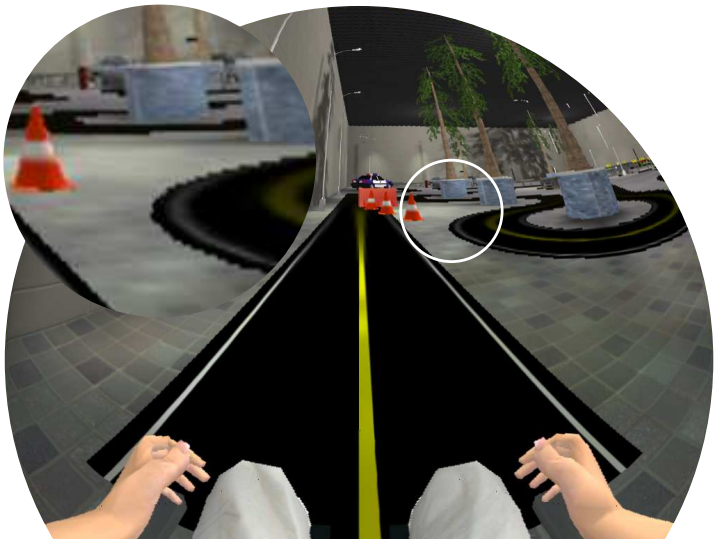


(e) Obstacle course

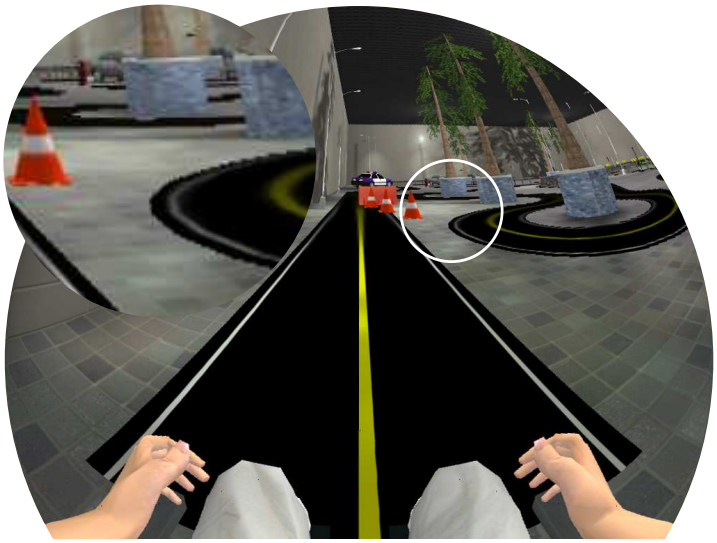


(f) Ramp

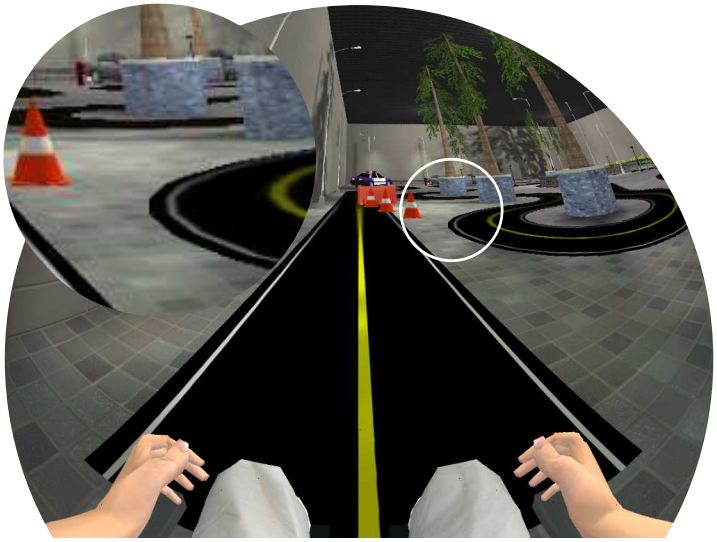
Figure 2.9: Selected features of the training track.



(a) 512 × 512



(b) 768 × 768



(c) 1024 × 1024

Figure 2.10: Impact of texture size on the quality of stitched frames. Each frame has the same resolution 1024 × 768. Lines are visibly more jagged when the texture resolution is small.

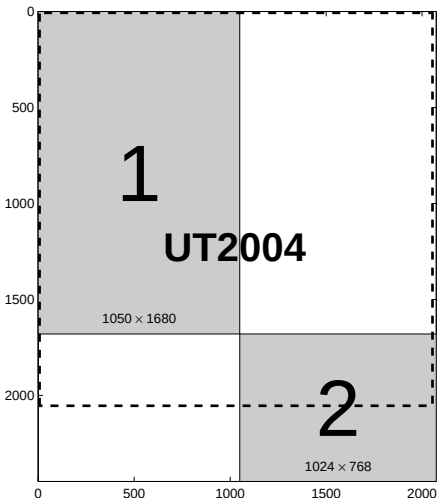


Figure 2.11: Screen positioning for optimal backbuffer extension. Screen 1 is a rotated high resolution widescreen display; screen 2 is the VisionDome® projector. The backbuffer is 2448 × 2074 pixels and just fits a UT2004 window big enough to contain four 1024 × 1024 textures.

increased definition came at a cost of performance — a drop to about 21 fps, which further decreased (to about 16) for complicated VEs like the track presented above when the wheelchair was moving fast. The decrease was most pronounced when the wheelchair was turning, due to the rapid changes in the scene, and when operated by the mouse replacement device, due to its overflowing the application's message queue with cursor movement update messages.

2.6 Testing and Evaluation

To evaluate the overall appeal and usability of the simulator, $n = 11$ able-bodied volunteers (six females; age between 23 and 40 — mean 29.27, standard deviation 6.40) were asked to complete a questionnaire after a short experience using the simulator. They completed the training course described above twice, each time using a different mode of control — the keypad or the Joystick Plus (see Figure 2.1 on 17). The subjects' performance is analysed in detail in chapter 5.

Since the target group for the training simulator is novice and prospective wheelchair users, the simulator was tested by healthy volunteers rather than experienced wheelchair users. Experienced users used to their particular wheelchair's handling and control devices would likely be too disturbed by the simulated wheelchair's different handling to appreciate all aspects of the simulator.

The questionnaire consisted of four items asking subjects to rate different aspects of the simulator, questions about each mode of control, some background questions, and the Game Experience Questionnaire (GEQ) developed at the Eindhoven University of Technology as part of the EU FP6 FUGA (Fun of Gaming) project (IJsselsteijn

et al., n.d.). Using a five options Likert-type scale where respondents indicate their level of agreement, ranging from "not at all" (0) to "extremely" (4), with 33 statements, the GEQ measures seven independent dimensions of a gaming experience: *challenge*, *competence*, *flow*, *immersion*, *positive affect*, *negative affect* and *tension*. Consensus is still emerging on the definitions of these factors (Wirth et al., 2007). Usually, flow is understood as the state of engagement where skills match the difficulty of the task, one performs at their best and is in effortless control (Csikszentmihalyi and Csikszentmihalyi, 1988). In the GEQ, challenge and competence are measured separately, and *flow* is a measure of engagement (Gajadhar et al., 2009). Example questions for these concepts are, respectively, "I thought it was hard", "I was good at it", "I forgot everything around me". *Immersion* refers to the feeling of *being* in the game or simulation rather than playing or using it and combines sensory (audio, visual, etc.) immersion and imaginative immersion in the game's story. Example questions include "It was aesthetically pleasing" and "I felt I could explore things". *Positive affect* measures enjoyment ("I enjoyed it") and *negative affect* measures feelings like boredom and annoyance ("I thought about other things", "It gave me bad mood"). Finally, *tension* measures frustration ("I felt annoyed").

The results of the GEQ are presented in Figure 2.12, while subjects' ratings of the different aspects of the simulator are shown in Figure 2.13. More details on answers to individual items are available in Appendix A. Overall, the response was very positive. Subjects enjoyed their experience of using the simulator (median *positive affect* 2.60 out of 4) and felt immersed (2.33). Although *flow* was one of the higher scoring factors (2.40), *challenge* scored relatively low (1.40) and in particular lower than *competence* (2.0). However, 7 out of 11 subjects found the task "somewhat difficult". Importantly, the negative factors *tension* and *negative affect* scored very low — 0.33 and 0.50 out of 4, respectively — and the level of graphical detail as well as the speed and steering response of the wheelchair were all rated "about right" by the majority of subjects (9, 7 and 8 out of 11, respectively).

2.7 Discussion

Through a unique combination of video game technology, a robotic simulation platform and an immersive display system, the simulator presented in this chapter provides a versatile research and training tool. It addresses both the clinical need for accurate and immersive visual feedback and realistic control dynamics, and the requirements of smart wheelchair system developers in terms of sensor simulation. Importantly, through the provided Matlab® interface, the simulator integrates into ongoing work on EEG analysis at the neurophysiology lab at the University of Strathclyde. By allowing the subjects to observe the effect of commands extracted from their EEG, the simulator closes the feedback loop between the subject and the system and thus becomes a key element of the Strathclyde Brain Computer Interface (Valsan et al., 2009).

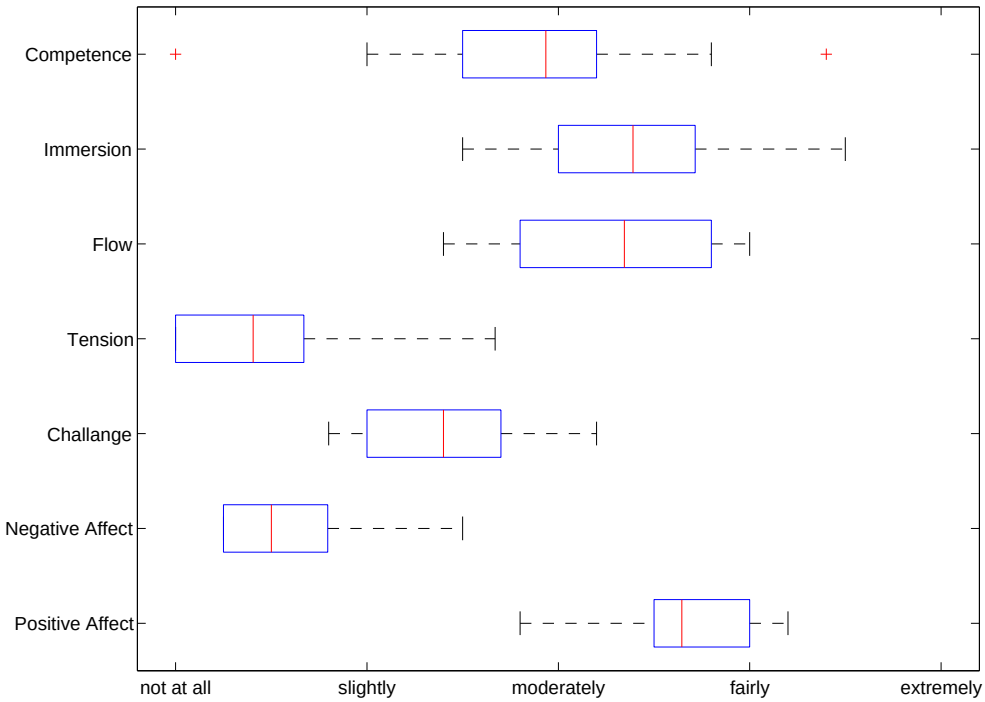


Figure 2.12: Game Experience Questionnaire results. Boxes represent the interquartile range, red lines — the median, and whiskers — the range of data. Outliers are indicated by red crosses. High flow and immersion and low negative affect and tension indicate suitability for prolonged training.

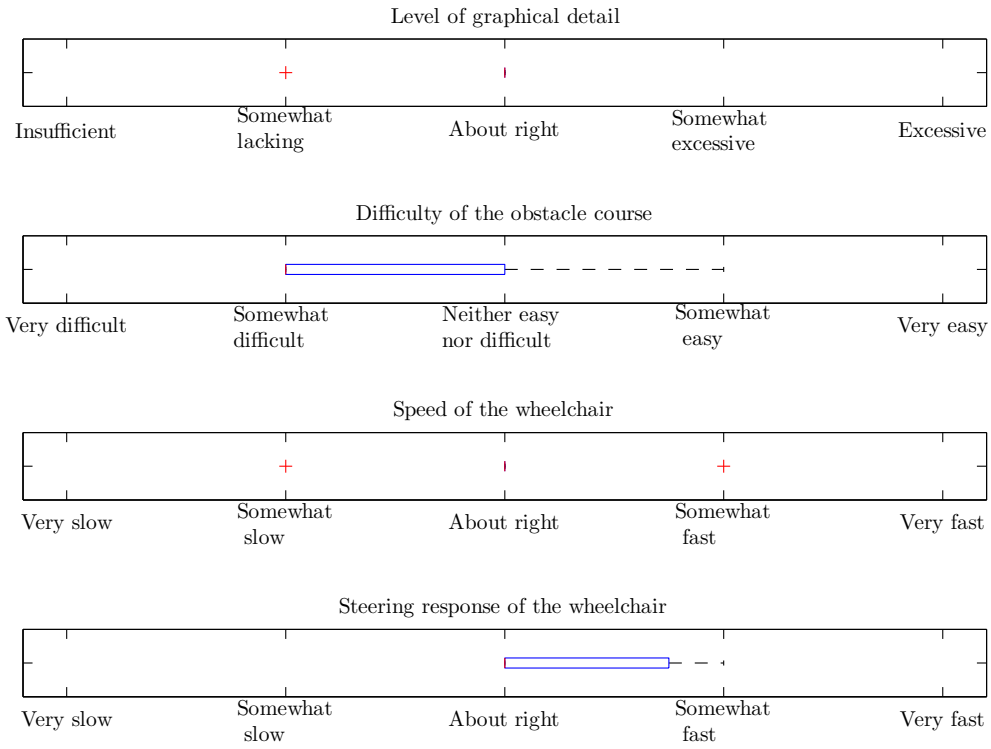


Figure 2.13: Responses to general questions about the simulator. Most aspects were rated “about right” by the majority of users.

The simulator has been positively received by users, as expressed by the overall scores from the Game Experience Questionnaire. In particular, the high score for immersion and flow are encouraging, as they support the use of the simulator for patient training. Moreover, subjects did not report any negative feelings such as boredom, annoyance or tiredness. However, three of the eleven subjects reported some degree of nausea during or after using the simulator, which, albeit unpleasant, could be seen as further evidence of the strength of the illusion of motion provided by the simulator.

As is often the case with projects of that size, external factors precluded the best design choices and unexpected incompatibilities between components extended the development time. Nonetheless, the final design of the simulator fulfils its major goal of serving both as a training and a development simulator, and meets most of the requirements described in section 2.2. An account of how each requirement is addressed is presented in the next section followed by a short discussion of some technical limitations.

2.7.1 Addressing the Requirements

Because fulfilment of the user requirements and realisation of quality attributes is a product of the functional and non-functional requirements, the latter are discussed first.

Functional Requirements (16/18 satisfied)

a) (Achieved) Simulate wheelchair movement in a physically realistic fashion (2d)
Achieved by using the Karma Physics Engine with USARSim settings and the purpose-built wheelchair model with carefully chosen simulated physical properties as described in chapter 3.

b) (Achieved) Simulate real-time sensory input (1b, 2d)
Capability inherited from USARSim.

c) (Achieved) Simulate sonar, laser, touch-sensitive and camera sensors (1b,1i)
Capability inherited from USARSim. Currently (version 3.1.3) USARSim supports infra-red and sonar range sensors, an odometer, a GPS receiver, an internal navigation system (INS), an encoder for joint positions, a touch sensor, a radio frequency identification (RFID) reader, a victim sensor (detects people), a sound sensor, a pyroelectric (human motion) sensor, and a normal as well as omnidirectional cameras.

d) (Achieved) Log simulation parameters for offline analysis: wheelchair position at 1 Hz and all sensor output (1b, 1l)
All sensor readings, including the wheelchair model's position and velocity vectors (every 1 second), and commands sent to the simulator are recorded in text-based log files.

e) (Achieved) Log subject performance: path, collisions and all subject input(1a, 1e)
The path of the wheelchair can be reconstructed from the log file where any collisions and all control commands issued are also recorded. These can be analysed, as in chapter 5, with bespoke Matlab scripts included in Appendix A

f) (Achieved) Display VE using immersive display technology (1c, 1a, 2c)
The system supports the VisionStation immersive display system.

g) (Achieved) Display geometrically correct images when using immersive display technology (2c, 2d)
The display settings can be optimised for individual subjects as discussed in chapter 4. Typical error in distance and size perception is below 5 cm.

h) (Achieved) Display VE using standard displays (1b, 1f)
Native Unreal Tournament 2004 display is available.

i) (Partially achieved) Update the displayed image at least 30 times per second (1a, 2c, 2d)
As discussed above, 30 fps are easily achieved. However, complex features, fast movement and high definition cause a decrease in fps below the target when using the immersive display.

j) (Not achieved) Accept and interpret input from third party wheelchair controllers (1g)
While the simulator can operate with any device that emulates a PC keyboard or mouse, wheelchair controllers are not supported. The optimal solution would be for the simulator to interface with a modular wheelchair controller system like DX (from Dynamic) or R-net (from PG Drives Technology). However, these options were not available due to phased out products (DX-key for the DX System) or prohibitive costs (R-net). A mouse-replacement device similar in look and feel to a wheelchair joystick has been installed instead.

k) (Achieved) Expose an interface for third party applications (1j, 2a)
Capability inherited from USARSim which communicates through a TCP/IP socket and offers full control options. A purpose built Matlab toolbox using this interface is included in Appendix A.

l) (Achieved) Accept new VEs developed externally (1h)
New virtual environments can be built with the *UnrealEd* tool sold with Unreal Tournament 2004.

m) (Partially achieved) Simulate a number of typical wheelchair models (1a, 1b, 1d, 1i)

While the system is capable of simulating any number of wheelchair models, only one actual model was developed, due to the extensive time required to build the geometric model. However, given a new geometric model, a fully functional model can be quickly developed re-using the solutions developed in this thesis and described in detail in chapter 3.

n) (Achieved) Allow modification of simulation parameters through text-based configuration files (1b, 1i, 2b)
Several parameters of the wheelchair model can be modified in the relevant section of the `USARBot.ini` file, including maximum velocity, mass, inertial parameters as well as number and configuration of different sensors. The XML-based `Wheelchair.exe.config` file allows text-mode access to settings of the GUI application including default screen size, SPI settings, path to UT2004 and the command used to start the UT2004 server.

o) (Achieved) Provide a graphical user interface (GUI) (1a, 2b)
An intuitive GUI was developed by the author. All functions necessary to run a training session are available within the GUI.

p) (Achieved) Allow modification of clinically-relevant parameters through the GUI (1a, 1d, 2b)
The GUI allows to choose the mode of control, a VE and a starting position as well as the wheelchair model to be simulated.

q) (Achieved) Allow modifying display settings to maximise the immersive effect for different viewing positions (1d, 2c)
SPI settings which affect the geometry displayed with the immersive display system can easily be adjusted within the GUI. A Matlab script to find the optimal settings for a given viewing position is included in Appendix B.

r) (Achieved) Run on Windows XP® (1c, 2b)

Non-functional Requirements (5/5 satisfied)

The simulator hardware consists of the dome screen, the new projector mount which can be easily dismounted into several pieces, and a PC. It is therefore:

a) (Achieved) fitting in a van (2a)

b) (Achieved) light enough to carry by two people (2a)

The simulator software is:

c) (Achieved) extendible (1i, 1m)
Because the software was implemented as several classes in an object-oriented

language, it's functionality can be easily extended with only minor modifications to the existing source code.

d) (Achieved) modular (1i, 1m)
Because each class has distinct functionality, the software is modular. In particular, the CSimWnd and CStateControl classes can be replaced to support a different immersive display system or implement alternative discrete command controller, respectively.

e) (Achieved) well documented (1m,2e)
The source code is well commented and detailed cross-referenced documentation in HTML format including class inheritance diagrams and links to specific lines of code is included in Appendix A. This will facilitate future modification of the software by another developer.

Constraints (2/2 satisfied)

a) (Achieved) Use mainstream computer hardware (1c, 1f)
The minimal requirements to use the simulator without the immersive display technology are those of Unreal Tournament 2004 itself (1.0 GHz processor, 256 MB RAM, 5.5 GB HDD space, DirectX 9.0b-compatible graphics and audio cards) which, it being a six year old game, are easily met but nowadays commonplace PCs. For use with immersive displays, a gaming-grade graphics card and a dual-core processor are recommended. The only non-standard element of the present system is the Ageia® PhysX™ physics processor, which will only be used when the simulator and USARSim are updated to use Unreal Tournament 3. However, NVidia now offers PhysX support on its GeForce graphics card series 8 and higher (including the one installed on the present system).

b) (Achieved) Minimise cost (1c, 1f)
For the home user, the only cost associated with the simulator is that of Unreal Tournament 2004 (about £10), provided they have a PC capable of running the game. For the clinic, the additional cost of a supported immersive display system in the order of £10000 is probably justified given the expected reduction in staff cost of patient training and an increase in its availability. Given its low maintenance and installation cost, the single projector immersive display is more cost effective than multi-projector or cave-like systems, although they do not require expensive bespoke lenses. Moreover, the simulation can be used with a large computer screen (or an HD TV) instead of the dome, as most games are intended to be played.

User Requirements (11.5/13 satisfied)

As a consequence of meeting almost all functional and all non-functional requirements, all but one user requirements have been met. The users can:

a) (Achieved) use the simulator for training (patient, clinical user)

b) (Achieved) use the simulator for testing new technologies (researcher)

c) (Achieved) use the simulator in the clinical environment (patient, clinical user)

d) (Achieved) use the simulator with multiple patients (clinical user)

e) (Achieved) analyse patients' performance (clinical user)

f) (Achieved) use the simulator at home (patient)

g) (Not achieved) use third party control devices (patient, clinical user)

h) (Achieved) use with own virtual environments (clinical user, researcher)

i) (Partially achieved) change aspects of the simulated wheelchair (researcher, clinical user)
Limited by the availability of only one geometric model. Hence, front-wheel drive wheelchairs cannot be simulated.

j) (Achieved) control the wheelchair through external applications (researcher)

k) (Achieved) repeat the same test (clinical user, patient, researcher)
The GUI allows returning the wheelchair to a predefined spot in the VE and restarting the simulation.

l) (Achieved) analyse performance of third party control algorithms (researcher)
Facilitated by the text-based log files.

m) (Achieved) maintain the simulator and extend its capability (developer)
Facilitated by modular design and extensive documentation.

Quality Attributes (2.5/3 satisfied)

a) (Partially achieved) **Portability**
In its basic version (without the VisionStation®), the system only requires a joystick and a PC to run on. Although using the VisionStation® demands significant space commitment, the size of the dome is not prohibitive of transport. Hence the simulator can be considered portable in the clinical environment. However, since the ability to display a virtual environment is dependent on the Windows® operating system, it cannot be considered portable in the software sense. But if

no immersive display is required, as will most often be the case for smart wheelchair developers, the simulation remains as cross-platform as USARSim itself (i.e. requiring only a Windows® server), since the wheelchair model is developed as an integral part of USARSim.

b) (Achieved) **Ease of use**
The GUI provides in a clear and intuitive manner all the functionality required to use the simulator for training purposes in the clinical setting. The development of new maps in *UnrealEd*, however, has its learning curve, as does more advanced use of USARSim's capabilities. However, the application can be modified through configuration files without the need to recompile.

c) (Achieved) **Degree of immersion**
The large spherical screen with correct perspective and the accurate movement simulation provide a high degree of immersion. However, because the users cannot look around in the virtual world other than by steering the wheelchair, the degree of immersion provided by the present system is less than that offered by e.g. a head-mounted display.

2.7.2 Limitations

Software Limitations

Over the course of the project, several limitations of the design emerged, most of which could easily be addressed in future work. Firstly, it was noted when building the training track VE that *UnrealEd* becomes unstable when the number of binary space partitions (BSP) becomes large. Since the road is composed of many double-sided surface segments separate from the floor, this factor limited the smoothness of the turns which cause many BSPs. Future VEs should avoid BSPs of the floor and instead use one large texture for the whole surface with the appropriate elements drawn on it.

Secondly, it seems that only three of the four wheelchair cameras can correctly display an effect called fake backdrop which allows creating the illusion of an outdoors open space, and hence VEs are limited to (possibly very large) indoor environments. One way to address this problem would be to give up the top looking camera whose texture usually occupies the smallest and least important area of the dome screen. Another visual effect that is not supported by the wheelchair's cameras is textured light corona which simulates the effect of looking straight at a light source.

Finally, perhaps the most important limitation is the use of Windows Timers to redraw the stitched scene. Windows Timers execute their TimerProc function by posting an appropriate message to the applications message queue. A better solution would be to run the function in a separate thread, using e.g. the Multimedia Timers. However, attempts at using any other type of timer have failed, possibly because the Microsoft

Foundation Classes (MFC) are not thread-safe at object level[6]. If that is the case, one solution would be to implement the CSimWindow class without using MFC.

Hardware Limitations

Although the VisionStation® provides wide FOV, it is crucially limited in that nothing is projected on the floor or around the wheelchair — areas that demand most attention when manoeuvring in tight passages. The system also does not allow looking around or behind, making driving backward particularly challenging. Although a head-mounted display can be incorporated into the simulator to address these shortcomings without affecting the basic software architecture, a screen-based system was preferred due to the possible interference with BCI applications and the additional burden it would place on the users.

Another limitation of the system is that no motion simulation is provided by either a motion platform or a force-feedback control device, although sufficient information can be obtained from the game engine. Motion platform is obsolete in robotics simulation and not practical in the clinical setting, as it would require transferring the patients from their wheelchairs. Force-feedback control devices, on the other hand, are object of research (Fattouh et al., 2004), and may constitute a future extension of the simulator. They are not supported at the moment as they offer no benefit for the development or testing of BCI-based wheelchair control.

Another hardware limitation is that of the projector's resolution (1024×768) ultimately limiting the quality of the displayed images. Furthermore, achieving good definition is in effect conditional upon having two high resolution monitors and a dual head graphics card (see section 2.5).

Model Limitations

A further limitation of the current simulator is that only one wheelchair model was created, albeit of a popular wheelchair. There are significant differences in the handling of front-wheel drive, rear-wheel drive and centre-wheel drive wheelchairs, as well as between those intended primarily for indoor or outdoor usage. Consequently, some users will find the simulator not suiting their needs. However, since the method has been described in detail (see chapter 3), adding new models and re-using existing elements should not be too difficult.

2.8 Summary

This chapter provides an overview of the virtual reality wheelchair simulator developed by the author as the main contribution of this thesis. First, in section 2.1 two

[6] http://msdn.microsoft.com/en-us/library/h14y172e%28VS.71%29.aspx (retrieved on 21.08.2010)

applications were identified for the simulator in the context of BCI research and beyond: 1) (prospective) wheelchair user training and 2) development and testing of semi-autonomous and BCI wheelchair technology. This led to the development, in section 2.2, of 41 detailed design requirements of which 35 were satisfied and 4 partially satisfied, as accounted for in subsection 2.7.1. A review of the state of the art offered in section 2.3 demonstrated that no simulator developed to date could satisfactorily fulfil the requirements of both applications.

Based on the design requirements and the review of previous efforts, a single-projector dome display system was identified as the immersive technology of choice, while an extendible robotics simulator using a commercial game engine was chosen as a base for the implementation of the present system. As detailed in section 2.4, the hardware comprises a PC workstation with a gaming-standard graphics card, a keypad and a mouse replacement joystick, in addition to the aforementioned dome screen and a purpose-built projector mount. The software includes the USARSim robotics simulator, the Unreal game engine, and a library for spherical projection of images, all integrated into a GUI application developed by the author. The simulator also includes a model of a common wheelchair to be elaborated on in the following chapter, and a bespoke virtual obstacle course built by the author.

The simulator was tested by eleven able-bodied novice users who navigated the obstacle course, all of whom completed it successfully. As reported in section 2.6, a questionnaire administered after the experiment revealed the subjects enjoyed the task and experienced *flow* and *immersion*, but little *tension* or *negative affect*. Additionally, the majority of subjects rated the level of graphical detail as well as the speed and steering response of the virtual wheelchair as "about right". All these findings are encouraging for using the simulator as a training aid and research tool, despite a few limitations identified in subsection 2.7.2.

Chapter 3

Wheelchair Model

3.1 Introduction

This chapter presents in detail the procedure and tools used to develop a model of an electric powered wheelchair, integrate it in UnrealSim and ensure correct physical simulation. First, in section 3.2 the 3D geometrical model of the wheelchair is described. Subsequently, the mass-inertial properties of the wheelchair and the occupant are calculated in section 3.3. In section 3.4, the calculated centre of mass is validated and a motor torque value is chosen for the model through dynamic analysis of the forces acting on the wheelchair in two situations. The steps taken to include the new wheelchair model in USARSim and ensure its correct simulation are detailed in section 3.5. Closing the chapter are a discussion of limitations of the presented model and ideas for future work in section 3.8 and a short summary offered in section 3.9.

3.2 Geometry

The geometrical model of the INVACARE® SpectraPlus wheelchair was created in Maya 7.0 PLE (see Figure 3.1) in scale 1:100, i.e. one Maya unit corresponding to 1cm. The model consists of 19 separate objects, with a total of 2311 polygons (prior to triangulation). Ten of the objects were grouped together to form the chassis of the wheelchair. The remaining nine objects are the six wheels (including the tiny wheels at the back that prevent falling over backwards), the castor forks, and the kerb climber. These objects were exported from Maya to UnrealEd using the *unEditor* plug-in available on the Unreal Developer Network[1] (UDN).

Before the wheelchair model was exported to UnrealEd, it had been realigned in Maya such that the origin of the coordinate system roughly coincided with where the centre of mass (CoM) could be expected. This entailed a shift by 0.1 m forward

[1] http://udn.epicgames.com/Two/MayaCompletePlugins.html

(10 Maya units) and 0.46 m down (46 Maya units) with respect to where the model was originally built in Maya space. While this was approximate and later required adjustment, this step was necessary to conduct initial tests of the model in UT2004. From now on, this point will be referred to as the origin of the Unreal coordinate system. Unfortunately, the Maya and Unreal coordinates systems differ further in that their axes are not equivalent, see Figure 3.2. While the *unEditor* plug-in correctly transforms the exported geometry, other parameters of the simulation have to be transformed manually. The transformation method is described in section 3.5. Until then, the Maya coordinate system will be used with the x axis pointing left, y axis pointing up, and z axis pointing forward, as on Figure 3.1 (c.f. Figure 3.2a).

3.3 Mass-Inertial Properties

Correct physical simulation using the Karma engine requires definition of the mass, centre of mass (CoM) and inertia tensor of each object, all of which influence its reaction to external forces. Hence, to ensure physical correctness, the mass-inertial properties of the wheelchair-occupant system were required. These were calculated separately for the wheelchair — using the geometric model — and the occupant — using literature body segment data — and then combined to find a single centre of mass and inertia tensor of the whole system.

The inertia tensor summarises all inertial properties of a rigid body with one quantity and can be used to calculate the moment of inertia for rotation around an arbitrary axis (Tenenbaum, 2004). The inertia tensor is expressed as:

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \tag{3.1}$$

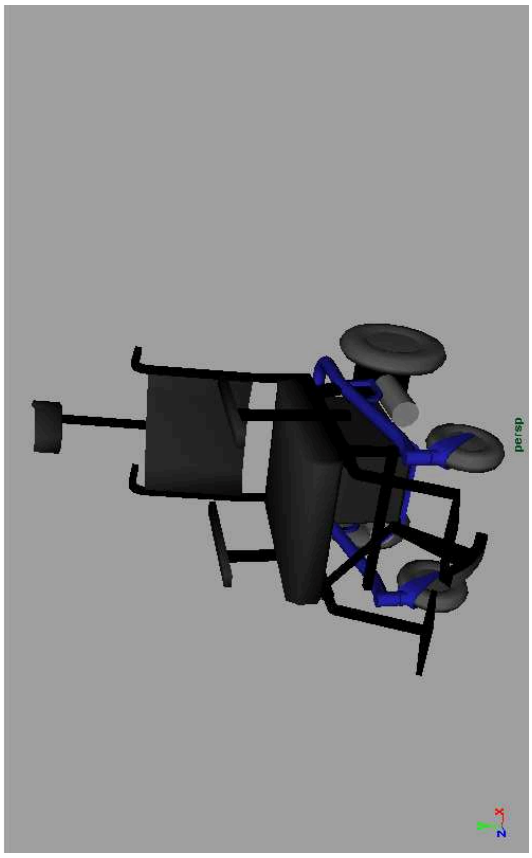
$$I_{ij} = \sum_{k=1}^{N} m_k (r_k^2 \delta_{ij} - r_{ki} r_{kj}) \tag{3.2}$$

where i and j range from 1 to 3 and correspond to the x, y and z axes respectively, r_k is the distance from the point about which the tensor is calculated to the given mass element m_k, N is the number of elements, and δ_{ij} is the Kronecker delta. The diagonal elements can also be expressed as:

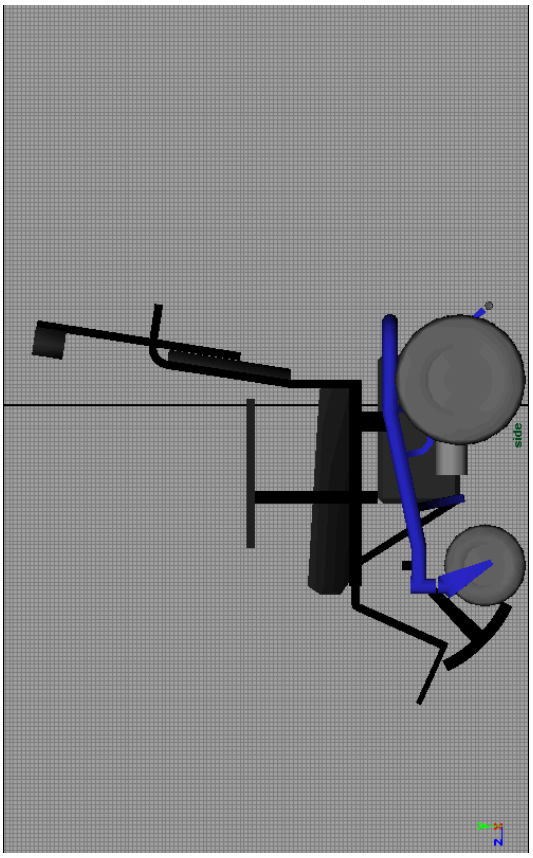
$$I_{xx} = \sum_{k=1}^{N} m_k (y_k^2 + z_k^2) \tag{3.3a}$$

$$I_{yy} = \sum_{k=1}^{N} m_k (x_k^2 + z_k^2) \tag{3.3b}$$

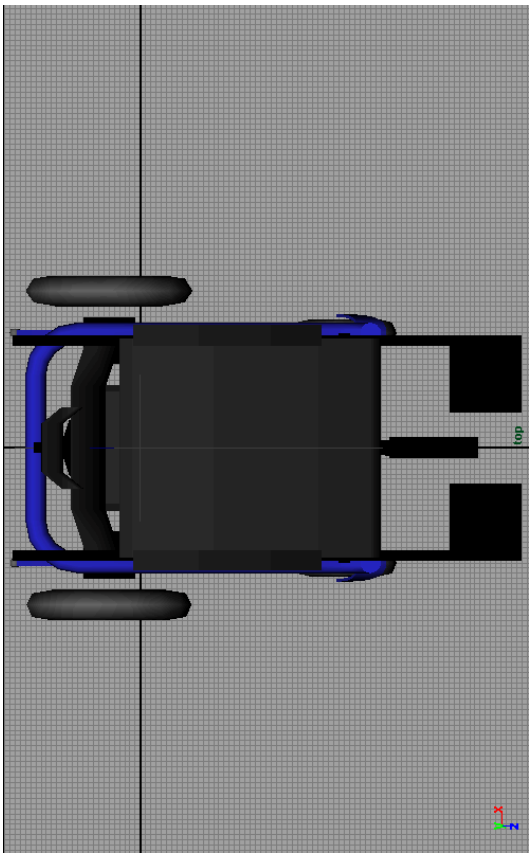
$$I_{zz} = \sum_{k=1}^{N} m_k (x_k^2 + y_k^2) \tag{3.3c}$$



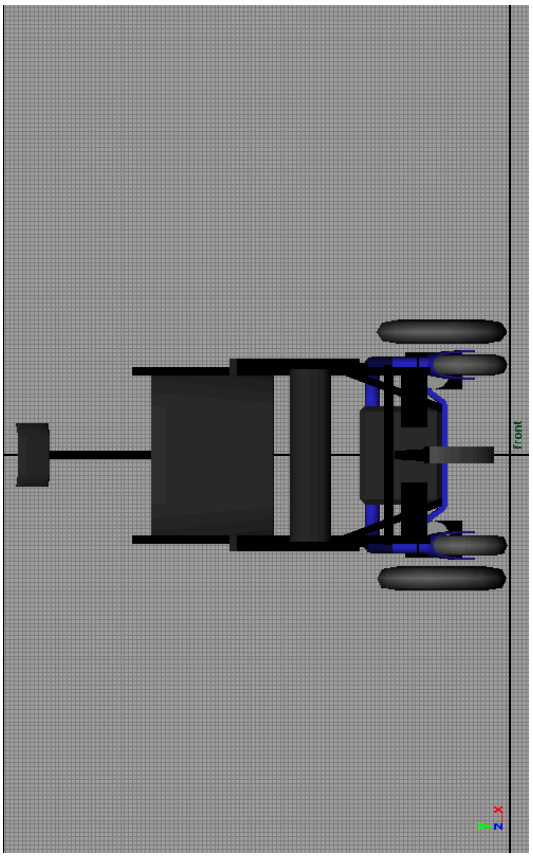
(b) Perspective view



(d) Side view

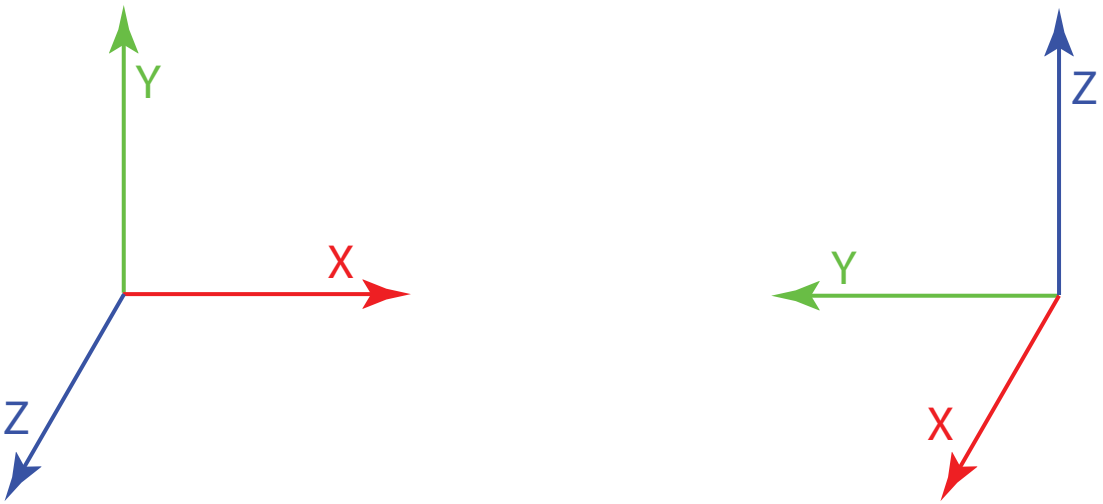


(a) Top view



(c) Front view

Figure 3.1: 3D Wheelchair model as seen in Maya®.



(a) Maya coordinates (b) Unreal coordinates

Figure 3.2: The different coordinates systems in Maya and Unreal.

and the off-diagonal elements as:

$$I_{ij} = -\sum_{k=1}^{N} m_k r_{ki} r_{kj} \tag{3.4}$$

3.3.1 Wheelchair

Each of the wheelchair parts was separately exported to a DWG file and analysed with AutoVue™ software where volume, centre of mass and inertia tensor were calculated assuming uniform density and roughly estimating the mass. Since the coordinate system was preserved, this allowed for the calculation of the CoM of the chassis and the inertia tensors of each part of the chassis with respect to that CoM (see Table 3.1). The inertia tensor of the entire chassis was then evaluated by summing the corresponding values of all its parts.

3.3.2 Occupant

Critically affecting the inertial properties of the whole system, and hence also its safety and stability, is the distribution of mass of the patient. To include this factor in the simulation, literature values for average human body segment parameters have been used to calculate the centre of mass and inertia tensor of a person in a sitting position.

First, average body segment lengths have been obtained from de Leva's (1996) correction of data published by Zatsiorsky et al. (1990) who used gamma-ray scanning to determine relative masses, centre of mass locations and radii of gyration of human body segments in young Caucasian population. De Leva (1996) translated Zatsiorsky et al.'s (1990) segment data from the original bony landmarks definitions to more convenient ones based on joint centres. The values used are reproduced in Table 3.2. The

Table 3.1: Wheelchair model parts: physical properties. All values reported in Maya coordinates.

Part	Mass [kg]	Volume [dm^3]	Density [kg dm^{-3}]	CoM [cm]	Inertia Tensor [kg m^2]
Chassis	63			$\begin{pmatrix} 0.0 \\ -37.4 \\ 8.7 \end{pmatrix}$	$\begin{pmatrix} 13.03 & 0.00 & 0.00 \\ 0.00 & 5.54 & -3.62 \\ 0.00 & -3.62 & 12.21 \end{pmatrix}$
Base frame	10	2.38	4.20	$\begin{pmatrix} 0.0 \\ -47.0 \\ 7.7 \end{pmatrix}$	$\begin{pmatrix} 3.11 & 0.00 & 0.00 \\ 0.00 & 0.90 & -3.83 \\ 0.00 & -3.83 & 3.00 \end{pmatrix}^{\dagger}$
Side frame, armrest and footrest					
Left	5	1.59	3.14	$\begin{pmatrix} 19.9 \\ -29.4 \\ 25.9 \end{pmatrix}$	$\begin{pmatrix} 0.72 & -1.39 & 0.27 \\ -1.39 & 0.94 & -0.27 \\ 0.27 & -0.27 & 0.58 \end{pmatrix}^{\dagger}$
Right	5	1.59	3.14	$\begin{pmatrix} -19.9 \\ -29.4 \\ 25.9 \end{pmatrix}$	$\begin{pmatrix} 0.72 & 1.39 & -0.27 \\ 1.39 & 0.94 & -0.27 \\ -0.27 & -0.27 & 0.58 \end{pmatrix}^{\dagger}$
Motor and gearbox					
Left	5	2.11	2.37	$\begin{pmatrix} 20.3 \\ -58.0 \\ -0.4 \end{pmatrix}$	$\begin{pmatrix} 1.74 & -0.80 & -0.24 \\ -0.80 & 0.58 & 0.46 \\ -0.24 & 0.46 & 1.99 \end{pmatrix}^{\dagger}$
Right	5	2.11	2.37	$\begin{pmatrix} -20.3 \\ -58.0 \\ -0.4 \end{pmatrix}$	$\begin{pmatrix} 1.74 & 0.80 & 0.24 \\ 0.80 & 0.58 & 0.46 \\ 0.24 & 0.46 & 1.99 \end{pmatrix}^{\dagger}$
Seat	10	21.00	0.48	$\begin{pmatrix} 0.0 \\ -28.7 \\ 21.5 \end{pmatrix}$	$\begin{pmatrix} 0.41 & 0.00 & 0.00 \\ 0.00 & 0.46 & -0.10 \\ 0.00 & -0.10 & 0.25 \end{pmatrix}^{\dagger}$
Backrest	8	8.40	0.95	$\begin{pmatrix} 0.0 \\ 2.8 \\ -10.7 \end{pmatrix}$	$\begin{pmatrix} 1.30 & 0.00 & 0.00 \\ 0.00 & 0.82 & -0.59 \\ 0.00 & -0.59 & 0.71 \end{pmatrix}^{\dagger}$
Battery	15	17.03	0.88	$\begin{pmatrix} 0.0 \\ -50.0 \\ 6.0 \end{pmatrix}$	$\begin{pmatrix} 3.29 & 0.00 & 0.00 \\ 0.00 & 0.31 & 0.50 \\ 0.00 & 0.50 & 3.12 \end{pmatrix}^{\dagger}$
Castor fork	0.5	1.50	0.33	$\begin{pmatrix} -3.0 \\ -9.4 \\ 0.1 \end{pmatrix}$	$\begin{pmatrix} 9.77 & 3.11 & 0.00 \\ 3.11 & 3.33 & 0.00 \\ 0.00 & 0.00 & 10.00 \end{pmatrix} 10^{-3}$
Castor wheel	1	1.22	0.82	$\begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} 2.47 & 0.00 & 0.00 \\ 0.00 & 2.50 & 0.00 \\ 0.00 & 0.00 & 4.66 \end{pmatrix} 10^{-3}$
Rear wheel	2	3.63	0.55	$\begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$	$\begin{pmatrix} 12.21 & 0.01 & 0.00 \\ 0.01 & 12.38 & 0.00 \\ 0.00 & 0.00 & 23.76 \end{pmatrix} 10^{-3}$
Kerb climber	0.8	0.45	1.78	$\begin{pmatrix} 15.8 \\ -15.6 \\ -0.2 \end{pmatrix}$	$\begin{pmatrix} 0.04 & -0.30 & 0.00 \\ -0.30 & 0.04 & 0.18 \\ 0.00 & 0.18 & 0.09 \end{pmatrix} 10^{-3}$

† Reported with respect to the CoM of the entire Chassis.

Table 3.2: Mass-inertial body segment parameters according to de Leva (1996, Table 3). Mass is relative to body mass; centre of mass (CoM) and radii of gyration (r_S, r_T and r_L – in saggital, transverse and longitudinal planes, respectively) are relative to respective segment lengths.

Segment	Length [mm]	Origin	End	Mass [%]	CoM [%]	r_S [%]	r_T [%]	r_L [%]
Trunk	614.8	CERV	MIDH	42.57	49.64	30.7	29.2	14.7
Head	243.7	VERT	CERV	6.68	48.41	27.1	29.5	26.1
Upper arm	275.1	SJC	EJC	2.55	57.54	27.8	26	14.8
Forearm	264.3	EJC	WJC	1.38	45.59	26.1	9.4	25.7
Hand	170.1	WJC	MET3	0.56	34.27	24.4	15.4	20.8
Thigh	368.5	HJC	KJC	14.78	36.12	36.9	36.4	16.2
Shank	438.6	KJC	AJC	4.81	43.52	26.7	26.3	9.2
Foot	228.3	HEEL	TTIP	1.29	40.14	29.9	27.9	13.9
Foot*	69.4	AJC	floor	1.29	50.00	29.9	13.9	27.9

AJC, EJC, HJC, KJC, SJC, WJC – the joint centres of ankle, elbow, hip, shoulder and wrist, respectively; CERV – cervicale; HEEL – pternion; TTIP –acropodion; MET3 – 3rd metacarpale; MIDH – the point midway between hip joint centres; VERT – vertex. Refer to de Leva's (1996) study for full descriptions.
*Ankle—floor distance was estimated by subtracting the length of head, trunk, thigh and shank from the total height. The result is 4% of total height, compared to 3.9% reported by Drillis and Contini (1966). The CoM position was assumed.

Table 3.3: Additional body parameters required to calculate the inertia tensor.

Distance	Length [mm]	Source	Description
Shoulder width	447.6	D	25.8% of total body height
Pelvis width	333.12	D	19.2% of total body height
Trunk	497.9	L	alternative definition MIDH—MIDS

MIDH,MIDS – the point midway between, respectively, hip and shoulder joint centres.
D – Drillis and Contini (1966); L – de Leva (1996).

two missing horizontal distances between the arm joint centres and the hip joint centres (shoulder width and pelvis width) have been calculated using the relative segment lengths from an older study by Drillis and Contini (1966), see Table 3.3. Second, an avatar female figure was fitted into the virtual wheelchair in order to find the joint angles of a person sitting in the wheelchair Figure 3.3. The female avatar available in USARSim was at first scaled to the average height reported by de Leva (1996) — 1.735 m — but it proved difficult to aesthetically fit in the wheelchair model, so instead 1.65 m was used. Through trial and error the joint angles of the avatar's skeleton and its position relative to the wheelchair model were modified until it assumed a seemingly comfortable position. Thus obtained joint angles were then used in conjunction with the body segment data to calculate the mass-inertial properties of a person sitting in the wheelchair. The original height of 1.735 m was used for these calculations on the grounds that for a taller person the wheelchair's seating and footrests would be adjusted such as to obtain the same position in terms of joint angles.

Table 3.4: Geometric and inertial parameters of a person in a sitting position. Where applicable, only left side of the body is reported. Right side is symmetric about the YZ plane. Angles ϕ_x and ϕ_z pertain to segment rotations about the respective axis from the initial vertical position. Coordinates are in Maya coordinate system, c.f. Figure 3.2. Segments are as defined in Table 3.2.

Segment	ϕ_z	ϕ_x	Origin [m]	End [m]	CoM [m]	Inertia Tensor† [kg m^2]
Trunk	0°	0°	$\begin{pmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0000 \\ 0.6148 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0000 \\ 0.3052 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 1.544 & 0.000 & 0.000 \\ 0.000 & 0.471 & 0.299 \\ 0.000 & 0.299 & 1.199 \end{pmatrix}$
Head	0°	0°	$\begin{pmatrix} 0.0000 \\ 0.6148 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0000 \\ 0.8585 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0000 \\ 0.7328 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 1.277 & 0.000 & 0.000 \\ 0.000 & 0.057 & 0.221 \\ 0.000 & 0.221 & 1.240 \end{pmatrix}$
Upper arm	-10°	10°	$\begin{pmatrix} 0.2238 \\ 0.4979 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.0470 \end{pmatrix}$	$\begin{pmatrix} 0.2513 \\ 0.3444 \\ 0.0271 \end{pmatrix}$	$\begin{pmatrix} 0.055 & -0.060 & 0.028 \\ -0.060 & 0.111 & 0.018 \\ 0.028 & 0.018 & 0.145 \end{pmatrix}$
Forearm	0°	0°	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.0470 \end{pmatrix}$	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.3113 \end{pmatrix}$	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.1675 \end{pmatrix}$	$\begin{pmatrix} 0.010 & -0.010 & -0.016 \\ -0.010 & 0.071 & -0.002 \\ -0.016 & -0.002 & 0.065 \end{pmatrix}$
Hand	0°	0°	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.3113 \end{pmatrix}$	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.4814 \end{pmatrix}$	$\begin{pmatrix} 0.2716 \\ 0.2311 \\ 0.3696 \end{pmatrix}$	$\begin{pmatrix} 0.027 & -0.004 & -0.026 \\ -0.004 & 0.051 & -0.004 \\ -0.026 & -0.004 & 0.026 \end{pmatrix}$
Thigh	0°	100°	$\begin{pmatrix} 0.1666 \\ 0.0000 \\ 0.0000 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ 0.0640 \\ 0.3629 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ 0.0231 \\ 0.1311 \end{pmatrix}$	$\begin{pmatrix} 0.433 & 0.254 & -0.050 \\ 0.254 & 0.424 & 0.217 \\ -0.050 & 0.217 & 0.705 \end{pmatrix}$
Shank	0°	10°	$\begin{pmatrix} 0.1666 \\ 0.0640 \\ 0.3629 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ -0.3679 \\ 0.4391 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ -0.1240 \\ 0.3960 \end{pmatrix}$	$\begin{pmatrix} 0.598 & 0.156 & -0.148 \\ 0.156 & 0.352 & 0.284 \\ -0.148 & 0.284 & 0.416 \end{pmatrix}$
Foot	0°	10°	$\begin{pmatrix} 0.1666 \\ -0.3679 \\ 0.4391 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ -0.4363 \\ 0.4511 \end{pmatrix}$	$\begin{pmatrix} 0.1666 \\ -0.4021 \\ 0.4451 \end{pmatrix}$	$\begin{pmatrix} 0.380 & 0.079 & -0.046 \\ 0.079 & 0.121 & 0.163 \\ -0.046 & 0.163 & 0.303 \end{pmatrix}$

† Reported with respect to the CoM of the body.

Figure 3.3: Human avatar placed in the wheelchair while negotiating a 10 cm kerb.

To calculate the mass-inertial properties of a seated person, each segment was treated as a vector, with the middle point between the hip joint centres treated as the origin of the coordinate system. The Cartesian coordinates of each joint centre were obtained using the joint angles and the segment lengths. The centre of mass of each segment was then located along the respective segment vector. For calculation of the centre of mass of the entire body, the mass of each segment was assumed to be concentrated at its centre of mass. The total body mass used was 61.9 kg, the average reported by de Leva (1996, Table 4), corresponding to the average height of 1.735 m for females. The diagonal elements of the inertia tensor of each segment were calculated using the radii of gyration reported by de Leva (1996) with the formula $I = m_s r_s^2$, where m_s is the segment's mass, and r_s is its radius of gyration about the same axis as the moment of inertia I being calculated. The resultant inertia tensor for each segment was multiplied by a rotation matrix to reflect the respective segment's position in space. Using the parallel axis theorem, each segment's inertia tensor with respect to the CoM of the occupant could then be calculated. The final results are reported in Table 3.4 and presented graphically in Figure 3.4.

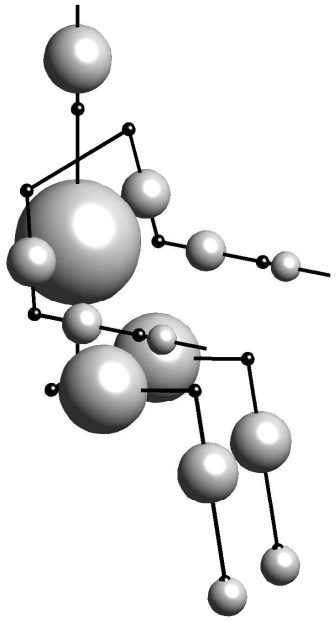


Figure 3.4: Distribution of mass of an average female sitting in a wheelchair (data combined from de Leva (1996) and Drillis and Contini (1966)). The volume of each ball is proportional to the mass of the respective segment while its centre indicates the location of the CoM along that segment.

3.3.3 The Entire System

Once the mass-inertial properties of both the wheelchair and the occupant were obtained, the entire system could be described with one set of values. First, using the relative position of the occupant relative to the wheelchair, found as described in the previous subsection, the CoM of the occupant was described in the wheelchair coordinate system. Second, the CoM of the entire system was found. Last, the inertia tensors of both the wheelchair and the occupant about that CoM were found using the parallel axis theorem (Muvdi et al., 1997, section 11.2) and subsequently added resulting in a single value representing the entire wheelchair-occupant system. The final values are:

$$CoM = \begin{pmatrix} 0.0000 \\ -0.1838 \\ 0.1128 \end{pmatrix} \quad [\text{m}] \tag{3.5a}$$

$$\mathbf{I} = \begin{pmatrix} 27.513 & 0.000 & 0.000 \\ 0.000 & 8.345 & -1.414 \\ 0.000 & -1.414 & 26.618 \end{pmatrix} \quad [\text{kg m}^2] \tag{3.5b}$$

Table 3.5: Combined centre of mass and inertia tensors for the wheelchair chassis and the occupant. Inertia tensors reported with respect to the common CoM.

	Mass [kg]	CoM Position [m]	Inertia Tensor [kg m^2]
Wheelchair Chassis	63. 0	$\begin{pmatrix} 0.0000 \\ -0.3742 \\ 0.0875 \end{pmatrix}$	$\begin{pmatrix} 13.028 & 0.001 & 0.000 \\ 0.001 & 5.544 & -3.619 \\ 0.000 & -3.619 & 12.212 \end{pmatrix}$
Occupant	61. 9	$\begin{pmatrix} 0.0000 \\ 0.0099 \\ 0.1385 \end{pmatrix}$	$\begin{pmatrix} 14.485 & -0.001 & 0.000 \\ -0.001 & 2.801 & 2.204 \\ 0.000 & 2.204 & 14.406 \end{pmatrix}$
Chassis + Occupant	124. 9	$\begin{pmatrix} 0.0000 \\ -0.1838 \\ 0.1128 \end{pmatrix}$	$\begin{pmatrix} 27.513 & 0.000 & 0.000 \\ 0.000 & 8.345 & -1.414 \\ 0.000 & -1.414 & 26.618 \end{pmatrix}$

The intermediate values are shown in Table 3.5.

3.4 Motor Torque

Once the combined CoM was calculated, dynamic analysis was performed to verify feasibility of the CoM value and find the required motor torque. To this end, two situations were considered: 1) climbing up an incline and 2) negotiating a kerb. In the following analysis ground reaction forces have been purposefully omitted for the sake of simplicity. Vectors are represented by symbols with arrows over them, while their magnitudes are denoted with the same symbols without the arrows.

3.4.1 Climbing up an Incline

The situation under consideration is depicted on Figure 3.5. The wheelchair is at rest (or moving with constant velocity) on an incline with angle α. Assuming the wheelchair is perfectly symmetric about the YZ plane (in Maya coordinates, c.f. Figure 3.1), the problem can be reduced to two dimensions.

Mechanical equilibrium requires the sum of forces and torques about any point to vanish:

$$\sum_i \vec{F}_i = \vec{0} \tag{3.6}$$

$$\sum_i \vec{\tau}_i = \vec{0} \tag{3.7}$$

In particular, the force $\vec{F}_{\mathrm{M}}$ generated by the motors must balance $\vec{W}_{\mathrm{x}}$, the component of weight $\vec{W}$ along the incline. Thus we obtain:

$$\vec{F}_{\mathrm{M}} + \vec{W}_{\mathrm{x}} = \vec{0} \tag{3.8a}$$

$$F_{\mathrm{M}} = W_{\mathrm{x}} \tag{3.8b}$$

$$F_{\mathrm{M}} = W \sin\alpha \tag{3.8c}$$

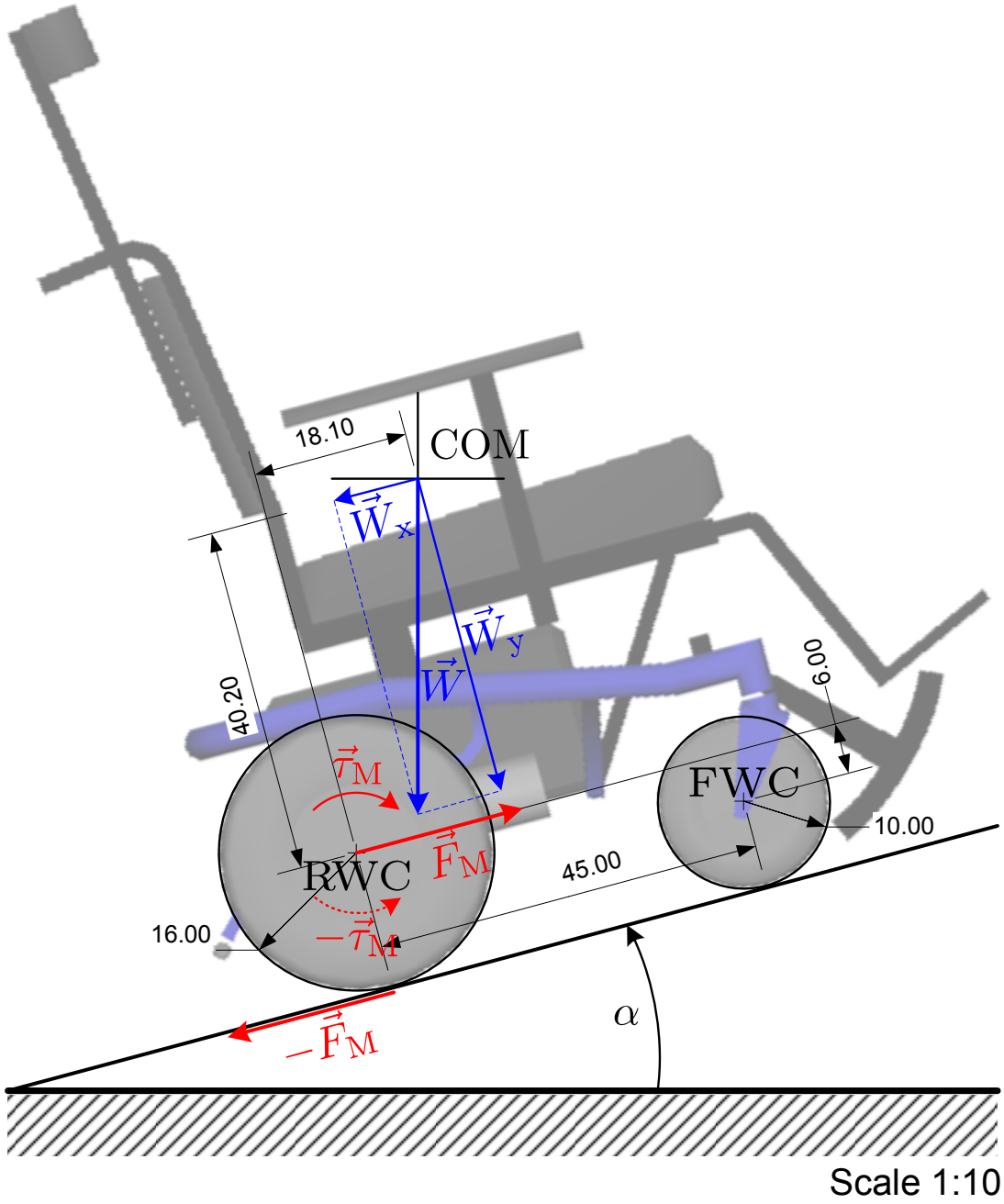


Figure 3.5: Wheelchair on an incline.

And the required motor torque is

$$\vec{\tau}_{\mathrm{M}} = \vec{r}_{\mathrm{RW}} \times \vec{F}_{\mathrm{M}} \quad , \tag{3.9}$$

where $\vec{r}_{\mathrm{RW}}$ is a vector connecting the rear wheel centre (RWC) and the point of contact between the rear wheel and the ground. Since $\vec{r}_{\mathrm{RW}}$ and $\vec{F}_{\mathrm{M}}$ are necessarily perpendicular,

$$\tau_{\mathrm{M\,min}} = r_{\mathrm{RW}} F_{\mathrm{M}} = W r_{\mathrm{RW}} \sin\alpha \tag{3.10}$$

is the minimal motor torque required to allow the wheelchair to climb up an incline with angle α.

As the incline angle α increases, the fraction of weight supported by the front wheels decreases, such that at a critical angle Equation 3.7 becomes:

$$\vec{\tau}_{\mathrm{M}} + \vec{\tau}_{\mathrm{W}} = \vec{0} \quad , \tag{3.11}$$

where $\vec{\tau}_{\mathrm{W}}$ is the moment of weight and can be expressed as:

$$\vec{\tau}_{\mathrm{W}} = \vec{r}_{\mathrm{CoM}} \times \vec{W} \tag{3.12a}$$

$$\tau_{\mathrm{W}} = dW_{\mathrm{y}} - hW_{\mathrm{x}} = W\left(d\cos\alpha - h\sin\alpha\right) \tag{3.12b}$$

where $\vec{r}_{\mathrm{CoM}}$ is the vector connecting RWC and CoM, and d and h are its components along the wheelchair's horizontal (x) and vertical (y) axes, respectively. From Equations 3.11 and 3.12b, we obtain the maximum motor torque allowed on an incline α before the wheelchair falls over backwards:

$$\tau_{\mathrm{M\,max}} = W\left(d\cos\alpha - h\sin\alpha\right) \quad . \tag{3.13}$$

Using the calculated CoM position (Equation 3.5) and the total mass of 128.7 kg (occupant and chassis plus all parts but the rear wheels), the conditions of Equations 3.9 and 3.13 are plotted in Figure 3.6. The green area indicates the motor torque values that satisfy both conditions ($T_{\mathrm{M\,min}} \leq T_{\mathrm{M}} \leq T_{\mathrm{M\,max}}$). As the incline angle increases, the margin between $\tau_{\mathrm{M\,min}}$ and $\tau_{\mathrm{M\,max}}$ narrows and no motor torque value will allow the wheelchair to climb inclines in excess of 17.86°. The real wheelchair's User Manual (INVACARE, 2006) indicates that the maximum incline the wheelchair can safely negotiate is 12° which would imply a motor torque between 42 and 118 Nm.

3.4.2 Negotiating a Kerb

In this scenario, the wheelchair is negotiating a kerb of height h_{K}, as depicted on Figure 3.7. The castors (front wheels) are already on the kerb, facilitated by the kerb climber. The rear wheels are resting against the kerb. Assuming sufficient friction between the rear tires and the edge of the kerb, the wheelchair will be able to climb the kerb if the motor torque generates a force exceeding the fraction of weight supported by the rear axle of the chair. The first step in estimating the required motor torque

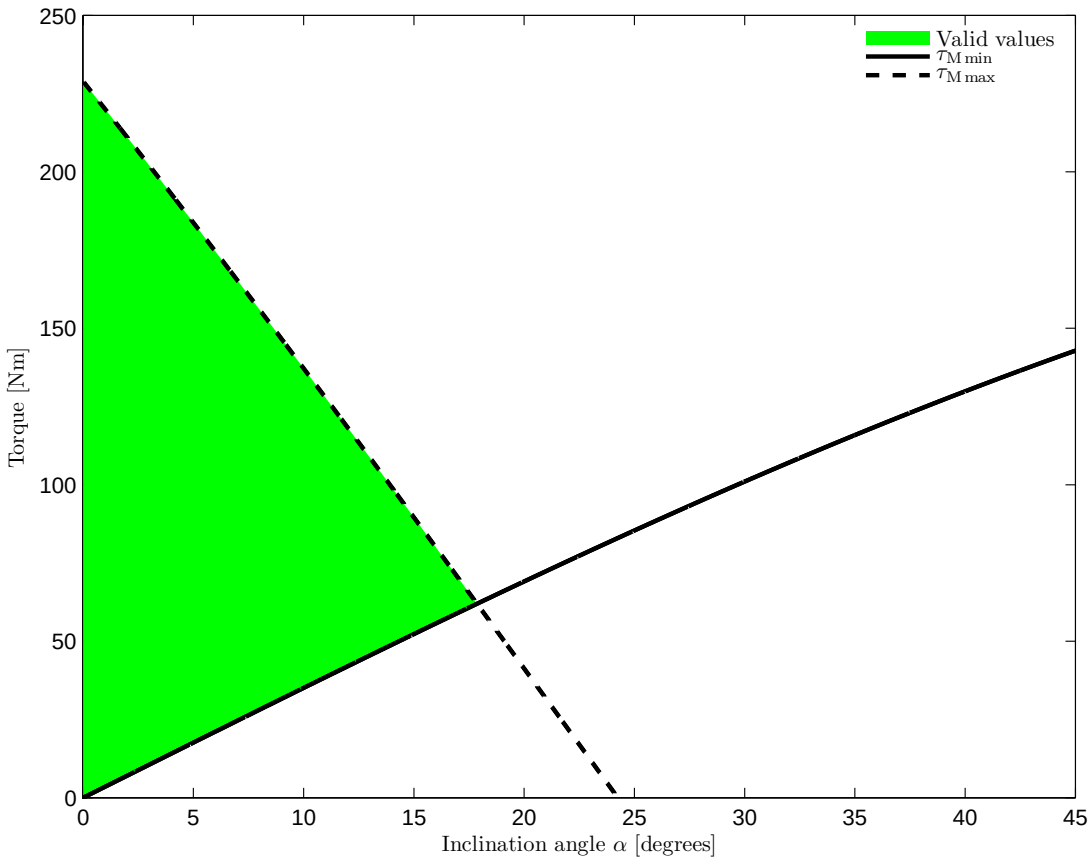


Figure 3.6: Maximum torque permissible and minimum torque required when climbing an incline as functions of the incline angle α. The intercept at 17.86° indicates the maximum incline the wheelchair can safely climb, provided the motor torque does not exceed 61.97 Nm which would cause the wheelchair to fall over backwards.

is hence the calculation of the distribution of weight of the wheelchair (including the occupant and all parts but not including rear wheels) between the rear axles and the point of contact between the castors and the kerb surface.

Assuming the x and z components of its components $\vec{W}_{\mathrm{FW}}$ and $\vec{W}_{\mathrm{RW}}$ are zero, the distribution of weight $\vec{W}$ can be determined by the following system of equations:

$$\begin{cases} \vec{W}_{\mathrm{FW}} + \vec{W}_{\mathrm{RW}} = \vec{W} \\ \vec{r}_{\mathrm{FWC}} \times \vec{W}_{\mathrm{FW}} + \vec{r}_{\mathrm{RWC}} \times \vec{W}_{\mathrm{RW}} = \vec{0} \end{cases} \tag{3.14}$$

where $\vec{r}_{\mathrm{RWC}}$ and $\vec{r}_{\mathrm{FWC}}$ are the vectors connecting the CoM and, respectively, the rear wheel centre and the point of contact between the front wheels and the kerb. Both of these vectors rotate around the rear axle when the castors are on the kerb by an angle

$$\alpha = \alpha_1 + \alpha_2 = \arcsin\left(\frac{r_{\mathrm{RW}} - r_{\mathrm{FW}}}{d_{\mathrm{RF}}}\right) + \arcsin\left(\frac{h_{\mathrm{K}} + r_{\mathrm{FW}} - r_{\mathrm{RW}}}{d_{\mathrm{RF}}}\right) \tag{3.15}$$

where

$$\vec{d}_{\mathrm{RF}} = (0.45, -0.06, 0.00)^T \quad [\mathrm{m}] \tag{3.16}$$

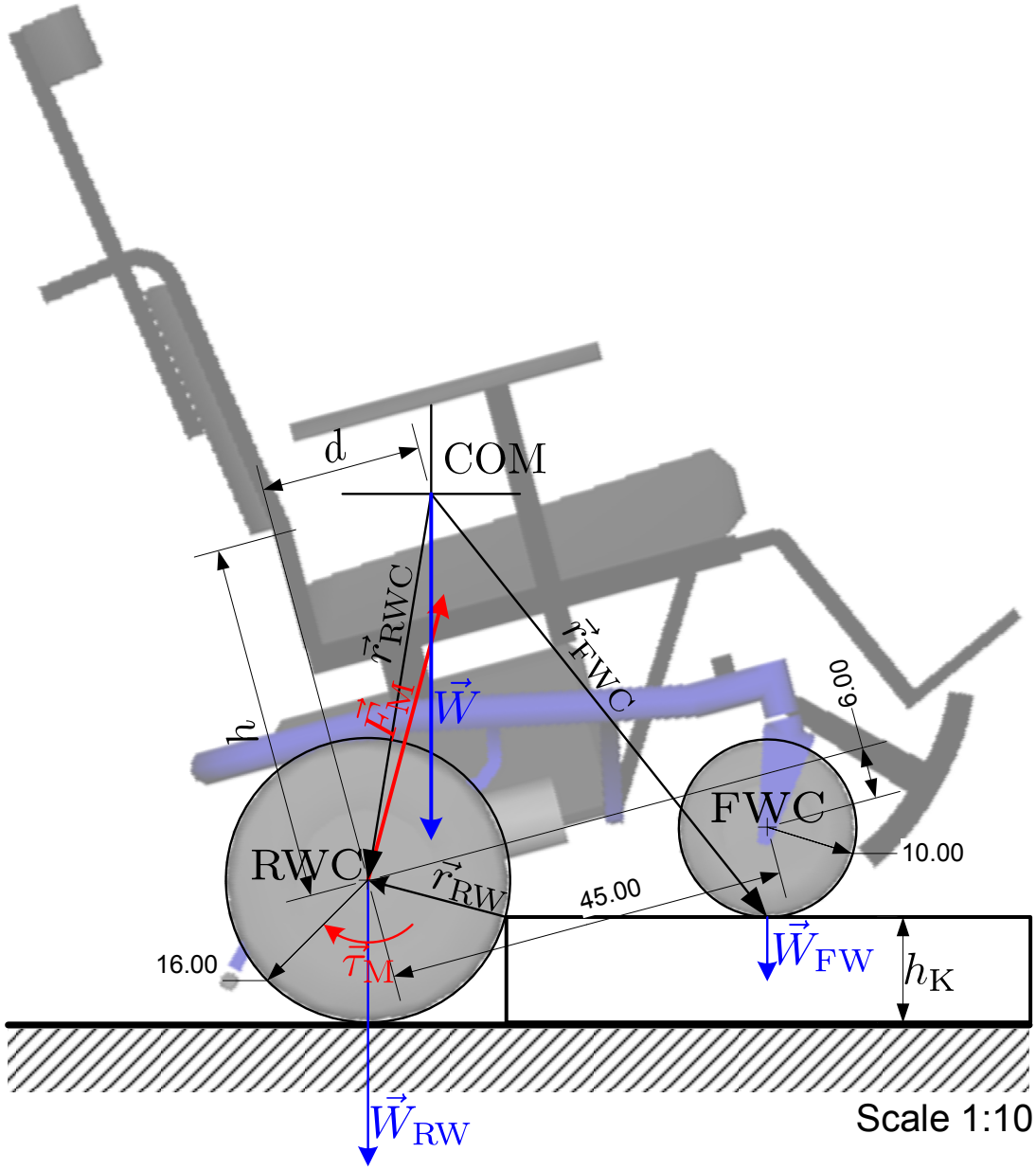


Figure 3.7: Wheelchair negotiating a kerb.

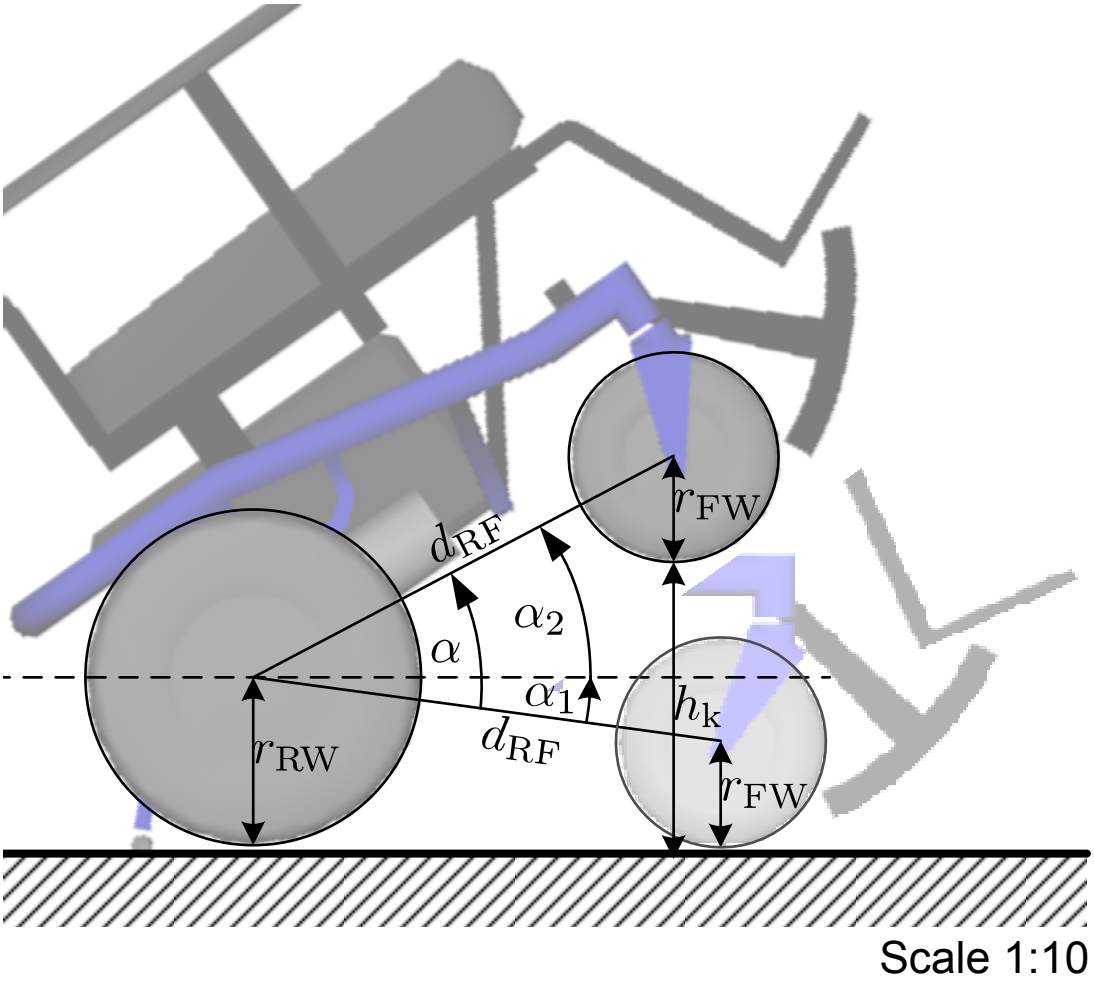


Figure 3.8: Wheelchair with front wheels on a kerb of height h_{K}. The rotation angle α can be calculated as a sum of angles α_1 and α_2 (c.f. Equation 3.15), where $\sin(\alpha_1) = (r_{\mathrm{RW}} - r_{\mathrm{FW}})/d_{\mathrm{RF}}$ and $\sin(\alpha_2) = (h_k + r_{\mathrm{FW}} - r_{\mathrm{RW}})/d_{RF}$.

connects the rear and front wheel centres (RWC and FWC, respectively) and r_{RW} and r_{FW} are the radii of the two wheels (c.f. Figure 3.8). Substituting the various geometrical parameters, Equation 3.15 becomes:

$$\alpha = 0.132 + \arcsin\left(2.203\,(h_{\mathrm{K}} - 0.06)\right) \tag{3.17}$$

The two moment arms $\vec{r}_{\mathrm{RWC}}$ and $\vec{r}_{\mathrm{FWC}}$ can then be expressed as:

$$\vec{r}_{\mathrm{RWC}} = \mathbf{R}_\alpha \cdot (-\vec{r}_{\mathrm{CoM}}) \tag{3.18a}$$

$$\vec{r}_{\mathrm{FWC}} = \mathbf{R}_\alpha \cdot \left(\vec{d}_{\mathrm{RF}} - \vec{r}_{\mathrm{CoM}}\right) \tag{3.18b}$$

where $\mathbf{R}_\alpha$ is the rotation matrix:

$$\mathbf{R}_\alpha = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

The solution to the system in Equation 3.14 can then be expressed in terms of the angle α as follows:

$$\vec{W}_{\mathrm{RW}} = \vec{W}\left(1 - \frac{d\cos\alpha - h\sin\alpha}{0.45\cos\alpha + 0.06\sin\alpha}\right) \tag{3.20a}$$

$$\vec{W}_{\mathrm{FW}} = \vec{W}\frac{d\cos\alpha - h\sin\alpha}{0.45\cos\alpha + 0.06\sin\alpha} \tag{3.20b}$$

Thus the minimum motor torque required to overcome the weight support by the rear axle and lift it by rotating the rear wheels (assuming sufficient friction to prevent sliding) is

$$\vec{\tau}_{\mathrm{M\,min}} = -\vec{\tau}_{\mathrm{W_{RW}}} = -\vec{r}_{\mathrm{RW}} \times \vec{W}_{\mathrm{RW}} \tag{3.21}$$

where

$$\vec{r}_{\mathrm{RW}} = \left(\sqrt{{r_{\mathrm{RW}}}^2 - (h_{\mathrm{K}} - r_{\mathrm{RW}})^2}, h_{\mathrm{K}} - r_{\mathrm{RW}}, 0\right)^T \tag{3.22}$$

is the vector (of length r_{RW}) from the point of contact between the rear tire and the kerb and the RWC (by Pythagorean theorem).

Analogously to the incline situation, the maximum permissible motor torque (and, hence, kerb height) is limited by stability considerations, as motor torque cannot exceed the moment of weight around the rear axle (Equation 3.11), which decreases with angle α, as per Equation 3.12b. Thus, the maximum motor torque is still expressed by Equation 3.13, with the added twist that the angle α is a function of kerb height (Equation 3.17).

The maximum and minimum motor torque conditions (Equations 3.21 and 3.13) are plotted in Figure 3.9 as a function of kerb height. It is clear from the graph that, whatever the motor torque, the wheelchair will not be able to climb kerbs in excess of 8.12 cm (value obtained by equating Equations 3.21 and 3.13). Although no guarantees are made regarding the wheelchair's ability to negotiate kerbs, the value of 10 cm is mentioned in the User Manual (INVACARE, 2006), and this value shall be regarded as the minimum requirement for the wheelchair model. This requirement cannot be met with the CoM at the previously calculated position (Equation 3.5).

Since the CoM calculation was based on estimated, rather than measured, masses of the different wheelchair parts, it is likely to be slightly offset with respect to real wheelchairs, especially having in mind that seating is adjusted individually to each patient. However, it is more likely that the longitudinal (along the x axis) rather than the vertical position of the CoM is wrong (measurement d on Figure 3.7). Since it is also the more critical of the two measurements, the following analysis seeking to find a plausible position of the CoM is limited to finding the smallest value of d that satisfies the requirements.

Fixing the kerb height $k_{\mathrm{K}} = 10\,\mathrm{cm}$ and the corresponding rotation angle $\alpha = 12.65°$ (as per Equation 3.16), the minimum required and maximum permissible motor torque conditions (Equations 3.21 and 3.13) become functions of the CoM position expressed by the variables d and h, c.f. Figure 3.7. These are displayed in Figure 3.10 as functions

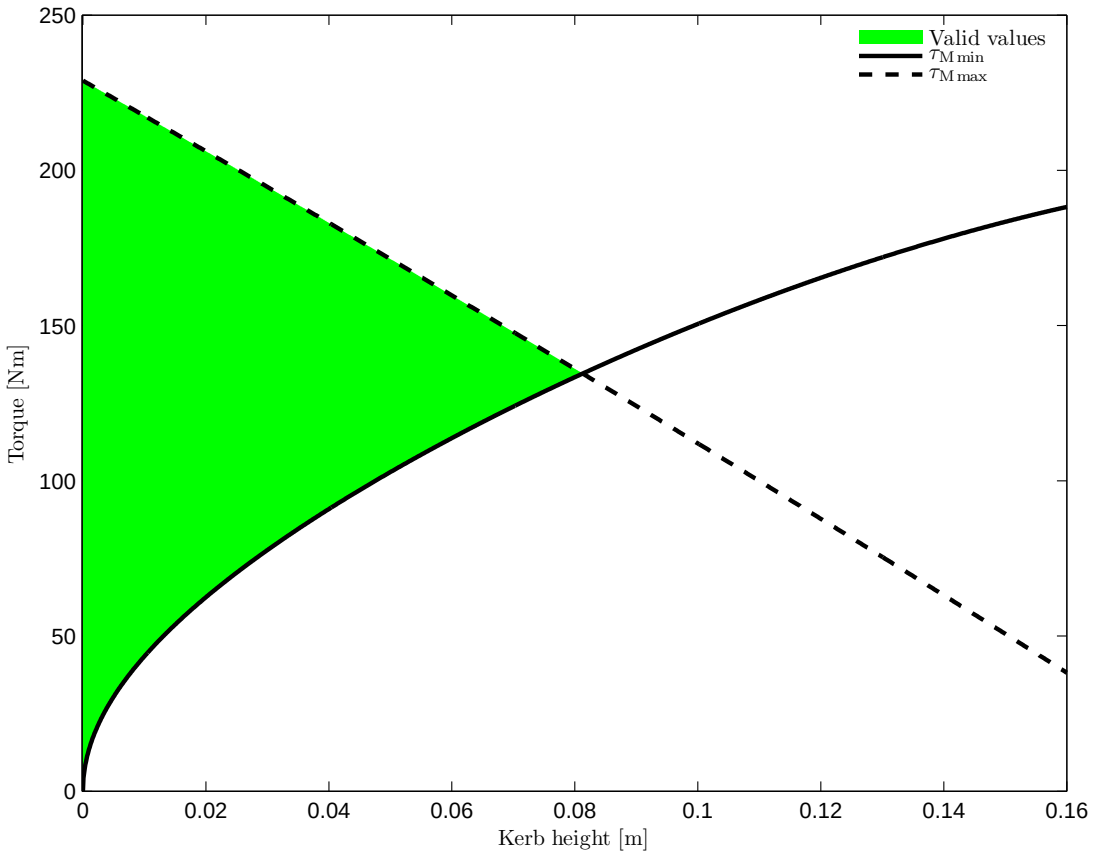


Figure 3.9: Maximum torque permissible and minimum torque required when negotiating a kerb as functions of the kerb height h_{K}. The two lines intercept at $h_{\mathrm{K}} = 8.12\,\mathrm{cm}$. No higher kerb can be climbed.

of d alone, keeping h at its previously calculated value of 40.20 cm. A shift of the CoM forward by just over 2 cm, from the calculated value of 18.20 cm to 20.49 cm (the intercept in Figure 3.10), suffices to meet all the conditions to allow the wheelchair to negotiate a kerb of $h_{\mathrm{K}} = 10\,\mathrm{cm}$.

Since the required amount of shift in CoM position is relatively small, it can serve as evidence that the assumed mass distribution of the whole system is approximately correct.

3.4.3 Adjusted Values

Following the mechanical analysis of the forces and torques acting on a wheelchair when climbing up an incline and when negotiating a kerb, the total motor torque (for both wheels together, without gearing) for the wheelchair model was chosen to be

$$\tau_{\mathrm{M}} = 140\ [\mathrm{Nm}] \tag{3.23}$$

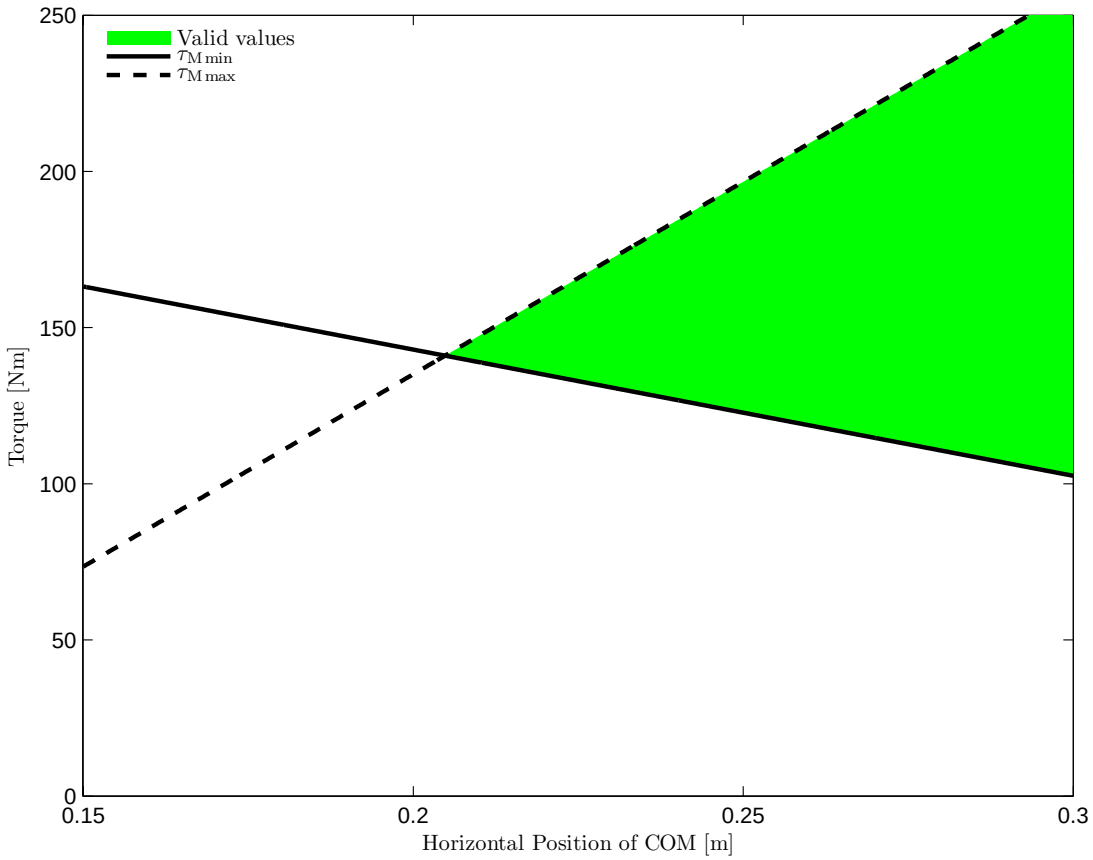


Figure 3.10: Maximum torque permissible and minimum torque required to negotiate a kerb of height $h_K = 10\,\text{cm}$ as a function of horizontal CoM position. The two lines intercept at $d = 20.49\,\text{cm}$.

and the modelled CoM position was shifted forward by 5 cm, such that the new position of the CoM, in Maya reference frame, is

$$CoM = \begin{pmatrix} 0.0000 \\ -0.1838 \\ 0.1628 \end{pmatrix} \text{[m]} \tag{3.24}$$

These values are shown on Figures 3.11 and 3.12 against updated versions of the graphs in Figures 3.6 and 3.9. The maximum incline the model wheelchair should be able to climb is just above 16°, allowing the wheelchair to muster some acceleration when on lesser inclines. Attempting to climb steeper inclines will cause the wheelchair to fall over backwards if full motor torque is applied at any time. To increase safety when negotiating kerbs, the motor torque value was chosen below the optimal value. Thus, with the new values, the model wheelchair should be able to negotiate kerbs up to 11.3 cm. While with stronger motors the wheelchair could theoretically climb higher kerbs, this would increase the risk of the wheelchair falling over backwards during the attempt.

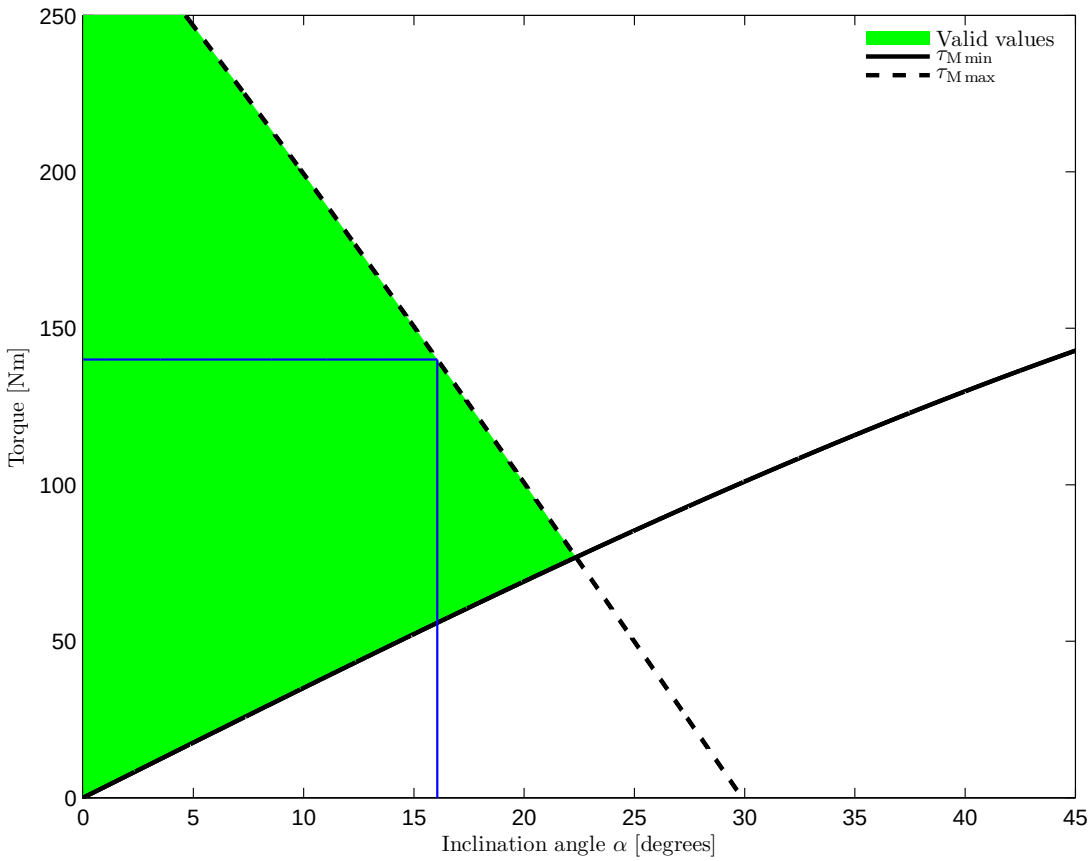


Figure 3.11: Maximum torque permissible and minimum torque required to negotiate an incline angle α with adjusted CoM position. With total motor torque of 140 Nm, the maximum incline angle that can be safely climbed is 16° (blue lines).

3.5 Simulation

In order to simulate a vehicle in USARSim, an UnrealScript class has to be written for the vehicle itself and each part of it that is separate from the chassis. The relative position of the parts and the joints between them are then defined in the UsarBot.ini configuration file. For each part also a static mesh must exist in the appropriate Unreal static mesh package (here, USARSim_Vehicle_Meshes.usx). All static meshes in Unreal need three bits of information for correct simulation:

- **The geometrical model**, which defines the appearance of the object, including its shape, materials and textures
- **Collision hull**, a much simpler geometrical representation that is used for detecting collisions
- **Karma mass properties**, a set of properties characterising the physical behaviour of the object, which is used by the Karma physics engine for simulation.

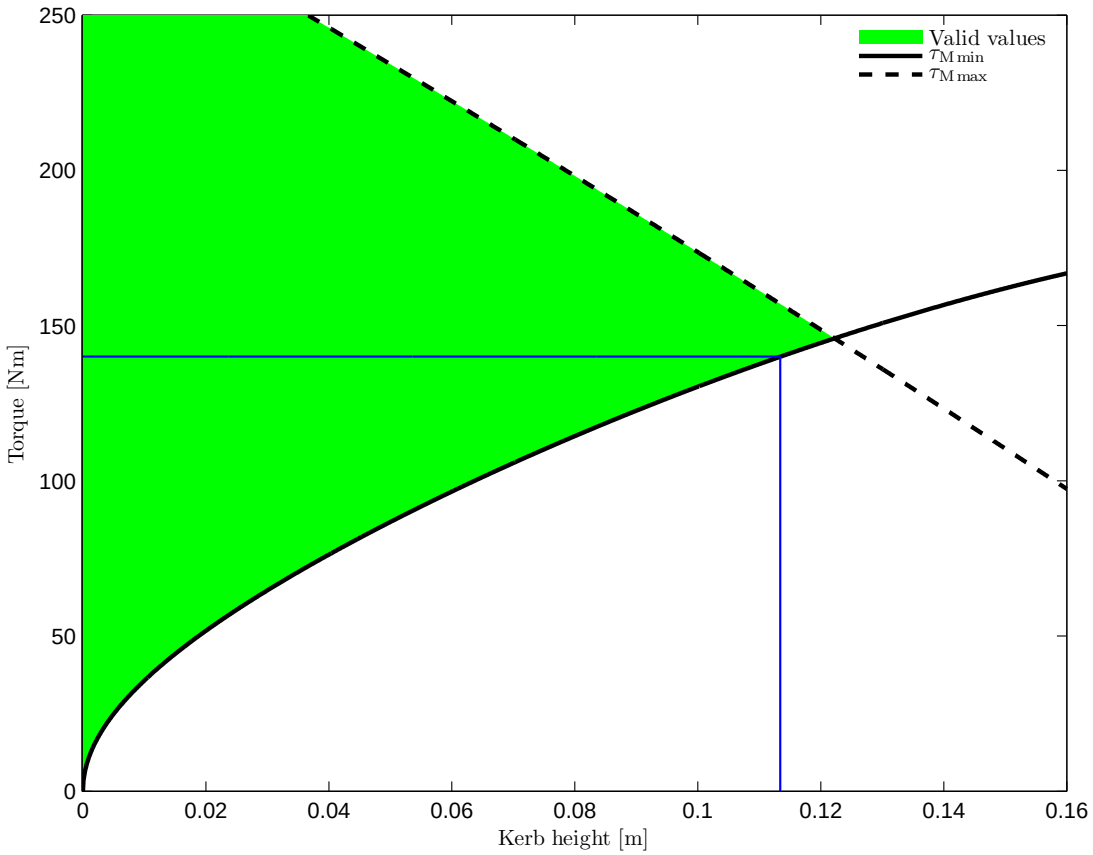


Figure 3.12: Maximum torque permissible and minimum torque required to negotiate a kerb of height h_K with adjusted CoM position. With total motor torque of 140 Nm, the maximum kerb height the wheelchair is able to negotiate is 11.3 cm (blue lines).

With the exception of the main chassis model, the collision models for all parts of the wheelchair were fitted automatically in UnrealEd. For the more complicated chassis model, a separate polyhedron was created in Maya and exported together with the object.

Unfortunately, the coordinates systems used by Maya and Unreal are different, see Figure 3.2 on p. 42. While the UnrealEd plug-in assures correct transformation of the geometry when exporting, the inertia tensors have to be transformed manually. Using equations 3.3 and 3.4 and the transformation rules

$$x' = z \tag{3.25a}$$
$$y' = -x \tag{3.25b}$$
$$z' = y \tag{3.25c}$$

it is easy to derive that

$$\mathbf{I}' = \begin{pmatrix} I'_{xx} & I'_{xy} & I'_{xz} \\ I'_{yx} & I'_{yy} & I'_{yz} \\ I'_{zx} & I'_{zy} & I'_{zz} \end{pmatrix} = \begin{pmatrix} I_{zz} & -I_{zx} & I_{zy} \\ -I_{xz} & I_{xx} & -I_{xy} \\ I_{yz} & -I_{yx} & I_{yy} \end{pmatrix} \tag{3.26}$$

where the primed symbols refer to the Unreal coordinate system.

The Karma properties are defined separately for each object in their respective Unreal Script class definition files using the `KInertiaTensor` array. Since the inertia tensors are always symmetric about the diagonal, `KInertiaTensor` has only six elements (numbered 0 to 5), arranged as follows:

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{pmatrix}$$

However, it was discovered that using the off-diagonal elements (1, 2 and 4) destabilises the physical simulation. Following the advice of relevant manuals (MathEngine, 2002; Busby et al., 2004; Wang and Balakirsky, 2007), only the diagonal elements were set in the class definition files.

The wheelchair model consists of five classes (*.uc files), one for each part. The Wheelchair class defining the chassis and behavior of the wheelchair is described below. The remaining classes are much less complex. The relevant sections of the Wheelchair.uc file are presented in Appendix A.

DefaultProperties

The `DefaultProperties` section of the Wheelchair class file (Listing A.1 in Appendix A) defines variables affecting the appearance and behaviour of the model, some of which can be overwritten in the USARBot.ini configuration file. Of particular importance is the `KarmaParamsRBFull` object which defines all parameters pertaining to physical simulation, including the `KIneriaTensor` discussed above and the `KCOMOffset` which defines the position of the centre of mass with respect to the centre of the static mesh. As mentioned earlier, the static mesh of the chassis was exported with the centre at $(0, -46, 10)$ cm (in the Maya coordinate system), as in Table 3.1. Since the correct value is $(0, -18.38, 16.28)$ cm (Equation 3.24), the necessary correction is $(0, 27.62, 6.28)$ cm or, in Unreal units and coordinate system, $(0.1255, 0, 0.5523)$ UU.

Other noteworthy parameters in this section include the `maxSpinSpeed` which limits the angular velocity of the powered wheels and thus the speed of the wheelchair, and the `MountRider` boolean variable which determines whether a human avatar is shown in the wheelchair. With the current `maxSpinSpeed` value, the wheelchair's maximal velocity is 6 km/h in accordance with the manufacturer's specification (INVACARE, 2006).

Additional parameters of the individual joints (connection between different parts) are set in the `PostNetBeginPlay` function where also internal class variables are initialised.

The Tick function

Each Unreal class has the opportunity to override the `Tick` function which is executed periodically by the game engine. Often, this is used to maintain internal status and implement position-dependent behaviour. In the Wheelchair class, the `Tick` function (displayed in Listing A.2) is used to maintain the orientation of the cue arrow such that it always points in the same direction (like a compass). This could be useful in some BCI experiments where subjects are required to perform a sequence of turns without regard for the actual path. An alternative behaviour would be for the arrow to point at a specific object or point in space. This would be useful for objective-driven experiments.

Another functionality implemented in the `Tick` function is checking whether the wheelchair is touching any objects and sending a log message if that is the case. The actual checking for touching objects is implemented in the `isTouch` function shown in Listing A.3, and relies on a number of `Beam` objects placed in strategic positions around the wheelchair. The `Beam` objects work similar to a laser burglar alarm — they detect obstacles in their path (`Trace` function).

3.6 Castoring

An important feature affecting the stability of electric powered wheelchairs is the self-alignment of castor wheels. Particularly when the castors are trailing, e.g. when reversing in a wheelchair with powered rear wheels and front castors, the castoring effect can cause a significant deviation from the intended trajectory (Ding et al., 2004). It is therefore imperative that wheelchair users are trained to anticipate and compensate for the castoring effect. Yet, due to their complicated dynamic and kinematic behaviour, the castoring effect is often ignored in wheelchair simulators (e.g. Harrison et al., 2004). In fact, there is no evidence[2] to suggest that any of the simulators reviewed in section 2.3 accounts for the castoring effect.

In the present simulator, the castoring effect is fully accounted for and can be experienced when driving the virtual wheelchair. Moreover, the freely rotating, self-aligning behaviour of the castor wheels is simulated with minimal effort from the developer. It suffices to specify the wheel size and the relative orientation and position of the two free joints that compose a castor for the Karma physics engine to correctly simulate its behaviour. All of these parameters can easily be modified in the USARSim.ini file, allowing to customise the virtual wheelchair model to more closely reflect a particular user's wheelchair.

[2] to the best of the author's knowledge

(a) Mild turn right

(b) Pivotal turn left

(c) Driving in reverse

(d) Right castor on an obstacle, left castor freely rotating in the air

Figure 3.13: Castor self-alignment in various situations. Note the different angle of the two castors when turning.

3.7 Validation

To validate the mass-inertial properties and motor torque value chosen for the simulated wheelchair, a series of tests was conducted by the author using USARSim's bespoke validation VE (`DM-ValidationTest_250.ut2`). This VE features, among others, ramps with different slopes (10 to 45 degrees in 5° intervals) and kerbs of different height (5 to 30 cm in 5 cm intervals). By attempting to negotiate these obstacles, the objective of the tests was to establish if the simulation is internally consistent, i.e. whether the virtual wheelchair's interaction with the virtual world corresponds to the real wheelchair's specifications.

On the 10° and 15° ramps, the wheelchair was stable, able to accelerate and travel up and down the ramp. However, when attempting the 20°ramp, the model would invariably fall over backwards. This behaviour is consistent with the dynamic analysis of subsection 3.4.1 and the calculated maximum negotiable incline angle of just above 16° (c.f. Figure 3.11).

Similarly, the wheelchair model performed as expected when negotiating kerbs. After a few trials, the 10 cm kerb could easily be negotiated at first attempt, while the 5 cm obstacle did not present any challenge at all. However, the 20 cm kerb proved insurmountable. This finding is in agreement with the predicted maximum negotiable kerb height of just above 11 cm (c.f. Figure 3.12).

While these results extend support for the choice of the simulated wheelchair's physical properties, they do not offer a complete validation. To this end, the model's parameters should be compared against the specific wheelchair (and user) being modelled. Unfortunately, due to the lack of a real SpectraPlus wheelchair, such validation could not be performed within the scope of this thesis. Should such chair become available, the validation could be carried out as follows:

- The position of the centre of mass of an occupied wheelchair (measurements h and d could be obtained through Equation 3.20a by measuring with a force plate the ground reaction forces acting on the rear wheels of the chair when its front wheels rest 1) on the floor and 2) on a kerb of known height.

- The maximum torque applied to the wheels could be found by immobilising a single wheel by means of a clamp equipped with a force transducer. An alternative method would be to collect measurements using an accelerometer mounted on the chair during maximum acceleration, but that would not guarantee measuring the maximum motor torque as acceleration could be capped by the wheelchair's controller.

These measurements would offer a real world reference, currently missing, against which the simulation's quality could be empirically validated.

3.8 Discussion and Future Work

The wheelchair model developed by the author is presented in this chapter with a view to justifying the parameters chosen and providing sufficient detail of the tools and methods used to allow future extension. In absence of detailed information about the wheelchair being modelled, such as masses of its individual parts or the motor torque, the simulated parameters where derived through calculation to conform to the known parameters — maximal speed and maximal incline angle and kerb height that can be safely negotiated. Included in the calculations was the average distribution of mass of the human body in the sitting position derived from data in the literature. Thus, the developed model not only looks like a wheelchair but also, through careful choice of simulation parameters and modelling of freely moving castor wheels and a functional kerb climber, behaves similar to a wheelchair.

One limitation of the present model is unrealistic motor characteristics. The motors do not exhibit a typical torque curve or inertia. Instead, the torque can change instantaneously and reach its maximum value to provide the desired wheel rotation speed. This results in an unrealistically high acceleration, especially at slow speeds or from standstill. Future work could look into achieving a more realistic response by artificially smoothing the control signal.

Another area of future work is equipping the simulated wheelchair with additional sensors, such as laser scanners and sonars, and implementing autonomous behaviour, e.g. obstacle avoidance or wall following. While such algorithms could be implemented directly in UnrealScript, it is recommended that they are developed externally to UT2004 to allow easy translation into a real system. To test such algorithms, functional sensors already available in USARSim can be added to the current wheelchair model simply by adding appropriate lines to the USARBot.ini file. However, to integrate them into the geometrical model will require first modifying the model in 3D modelling software and then recomputing its inertial parameters to ensure correct simulation.

3.9 Summary

A geometric model of the INVACARE® SpectraPlus wheelchair was first developed in 3D modelling software and then exported to UT2004 as a static mesh, as described in section 3.2. Its inertia tensor was calculated in section 3.3 by assuming a mass for each part and analysing its volume and position. Also there, the contribution of the occupant's distribution of mass to the inertial parameters of the system was evaluated using human body segment data collated from several sources. Subsequently, in section 3.4, the position of the centre of mass and the maximal torque of the simulated motors were adjusted after considering the wheelchair's dynamics in two specific situations so as to conform to the real wheelchair's specification. The centre of mass was

shifted forward from previously calculated position by 5 cm to ensure stability. An appropriate UnrealScript class was written for each object defining its behavior and implementing additional functionality, and these were presented in section 3.5. The resultant simulated wheelchair resembles its real counterpart in both appearance and behavior. As outlined in section 3.8, future effort should be directed towards implementing more realistic motor characteristics and equipping the model with simulated sensors as required.

Chapter 4

Geometry Validation

4.1 Introduction

To maximise the feeling of immersion and the translation of skills from the simulator to the real world, it is crucial that the projection of the virtual environment is as realistic as possible. This entails ensuring that vertical and horizontal lines appear as such, distances and sizes can be accurately judged and that perspective is preserved. Importantly, without stereo vision it is impossible to judge distance or size unless objects of known dimensions are available as reference. Additionally, if the image projected on the screen is static, changes in viewing position will affect the geometrical correctness of the perceived scene and there is at most one position in space, called a "sweet spot", from which the scene looks exactly correct.

The SPIClops API (ELUMENS, 2001) exposes settings that affect the position of the sweet spot: the projector lens position and the viewing position, both measured relative to the centre of the dome and its radius. However, neither estimates of the two quantities nor the recommended values provided satisfactory results when viewed by a person sitting upright in a chair — vertical lines appeared pronouncedly curved. Moreover, different settings resulted in projections of the same subjective quality but with different perceived viewer position, such that the floor and obstacles appeared too close or too far away. Yet, for the simulator to be used for training, it is imperative that the perception of distance (and hence velocity) is correct.

This prompted the study, described in this chapter, of the impact of the SPIClops settings on the stitched image and the relationship between pixel positions and lens projection angles with a view to optimising the projected geometry for individual subjects. For this purpose, bespoke image processing algorithms and a novel method to calibrate a wide-angle projector lens have been developed by the author.

The chapter is structured as follows. In section 4.2, a theoretical model of SPIClops is formulated. Its predictions are compared with data extracted with bespoke

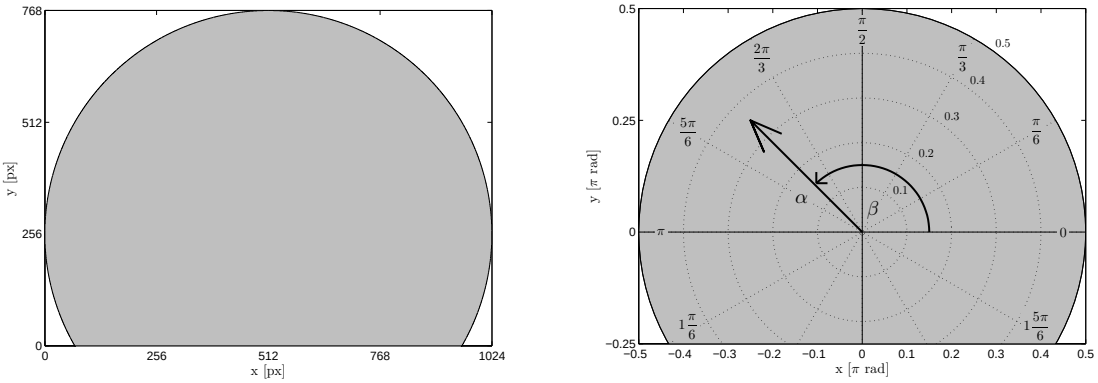
algorithms from images stitched by SPIClops. Section 4.3 describes a mathematical model of the projector lens and the experimental method used to obtain it, while section 4.4 details how the two models are used to optimise the SPIClops settings for an arbitrary viewing position. In section 4.5, the quality of the image projected on the dome is evaluated in terms of the accuracy of size and distance representation. Concluding remarks are offered in section 4.6 and a short summary of the chapter in section 4.7.

4.2 SPIClops Settings

The SPIClops API exposes six input parameters to control the sweet spot's location — three for each lens and eye 3D position. These are expressed in the dome coordinate system and normalised to the radius of the dome and thus varying from -1 to 1. The dome coordinate system has its origin at the centre of the dome, the y axis pointing up, the x axis to the right and the z axis away from the dome towards the observer. As there is no reason to move either the lens or the viewing position off the centre along the x-axis (left-right), four settings are relevant: LY and LZ for the lens — its position along the y (down-up) and z (forward-backward) axes, respectively — and the corresponding parameters EY and EZ for the viewing (eye) position.

In this section, the impact of these four parameters on the resulting stitched image produced by SPIClops is investigated with a view to obtaining the governing equations. The resulting model of the stitching process, allowing application of an arbitrary parameter combination to a reference image without re-stitching the scene, will form an integral part of a procedure searching for the optimal settings. The alternative to using such a model would be to collect the actual stitched images for a number of parameter combinations. This, however, is impractical. Uniform sampling of the four-dimensional parameter space with 0.01 precision (as allowed by the GUI) would require over 1.6 billion (201^4) images as each parameter can assume 201 different values between -1 and 1. With each 1024×768 pixels bitmap image occupying over 2.25 MB, the total space requirement would be almost 4 petabytes (4000 terabytes). Additionally, the discretization of the parameter space would complicate the search for the optimal combination.

In the remainder of the section, first, hypotheses about the impact of each parameter are developed based on the information in the SPIClops API manual (ELUMENS, 2001) and, second, methods for obtaining and analysing a dataset of sample images are described. The section concludes with the results of a comparison between simulated and actual images, which show good agreement.



(a) Frame as sent to the projector.

(b) Frame coordinate systems

Figure 4.1: True (a) and normalised (b) frame coordinates. The displayed image is contained within the greyed area. The polar coordinate system is defined by the radial distance α and the angle β.

4.2.1 Hypothesis

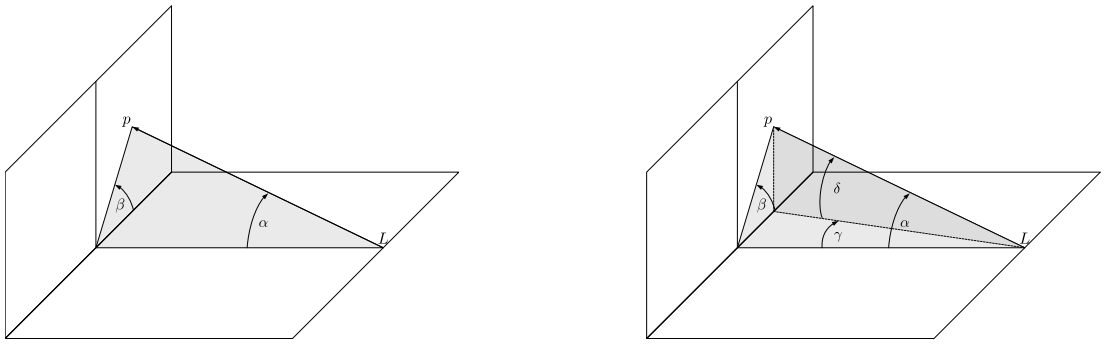
As stated earlier, the maximum resolution supported by the VisionStation display system is 1024×768 pixels. Within each frame, the image area is limited to a circle with a radius of 512 pixels and centred at (512, 256) as depicted on Figure 4.1a, with the rest of the frame being uniform background colour and not displayed on the dome. The SPIClops manual states that the "Elumens VisionStation [...] system use[s] a single projector and lens combination to provide the full 180-degree projection" and that "[the] lens is specially designed such that the projection angle of every pixel is linearly proportional to its relative distance from the centre of the frame buffer" (ELUMENS, 2001, p. 2). Under these assumptions, coinciding polar and Cartesian coordinate systems are defined for the frames to be displayed such that the horizontal axis varies from $-\pi$ to π and the vertical axis from $-\frac{\pi}{2}$ to π and the origin of both coordinate systems coincides with the centre of the image-containing circle, as shown on Figure 4.1b. The polar coordinates (β, α) of a pixel then describe its projection direction, as shown on Figure 4.2a. This odd coordinate system can be readily transformed into a more usual one where azimuth γ and elevation δ are defined by the following equations:

$$\sin\delta = \sin\alpha \sin\beta \tag{4.1a}$$

$$\sin\gamma = \sin\alpha \cos\beta / \cos\delta \tag{4.1b}$$

Importantly, the distance from the lens L to the projected point p (c.f. Figure 4.2b) and the geometry of the screen do not influence the conversion. In particular, the screen can be spherical, as is the case in the VisionStation.

Assuming that in a reference frame obtained with SPIClops without correction for lens or eye position (i.e. both positions set to coincide with the origin of the dome coordinate system) some feature of the scene being stitched appears at coordinates



(a) (b)

Figure 4.2: A pixel with polar coordinates (β, α) is projected from the lens L onto a point p on a screen (a). The projection direction can be expressed in terms of azimuth γ and elevation δ (b).

(β_0, α_0) (and so (γ_0, δ_0)), the problem is to calculate its coordinates when the same scene is stitched with different (non-zero) settings for lens and eye position. The problem is tackled by considering changes in viewing position first and changes in lens position second.

If the projector and the eye were both placed at the centre of the sphere of which the dome screen is a section, the scene projected on the dome would appear correct and no sweet spot correction would be necessary (assuming a perfect lens). The objective of correcting for the viewing position as it moves away from the centre must then be to change the projected image such as to preserve the angles at which its different features appear to the viewer. Thus, the problem comes down to finding the three dimensional coordinates (x, y, z) on the dome screen where a point must appear in order to be observed at angles γ_0 and δ_0 from a viewing position $(0, \mathrm{EY}, \mathrm{EZ})$. As can be seen from Figure 4.3 (dark blue geometry), the angles γ_0 and δ_0 can be expressed as:

$$\tan\gamma_0 = \frac{x}{z - \mathrm{EZ}} \tag{4.2a}$$

$$\tan\delta_0 = (y - \mathrm{EY})\frac{\sin\gamma_0}{x} \tag{4.2b}$$

Additionally, all points on the screen satisfy

$$x^2 + y^2 + z^2 = R^2 \tag{4.3}$$

where R is the radius of the spherical screen. Substituting Equation 4.2a and Equation 4.2b for y and z respectively, a quadratic equation is obtained:

$$x^2 + \left(\frac{x\tan\delta_0}{\sin\gamma_0} + \mathrm{EY}\right)^2 + \left(\frac{x}{\tan\gamma_0} + \mathrm{EZ}\right)^2 - R^2 = 0 \tag{4.4}$$

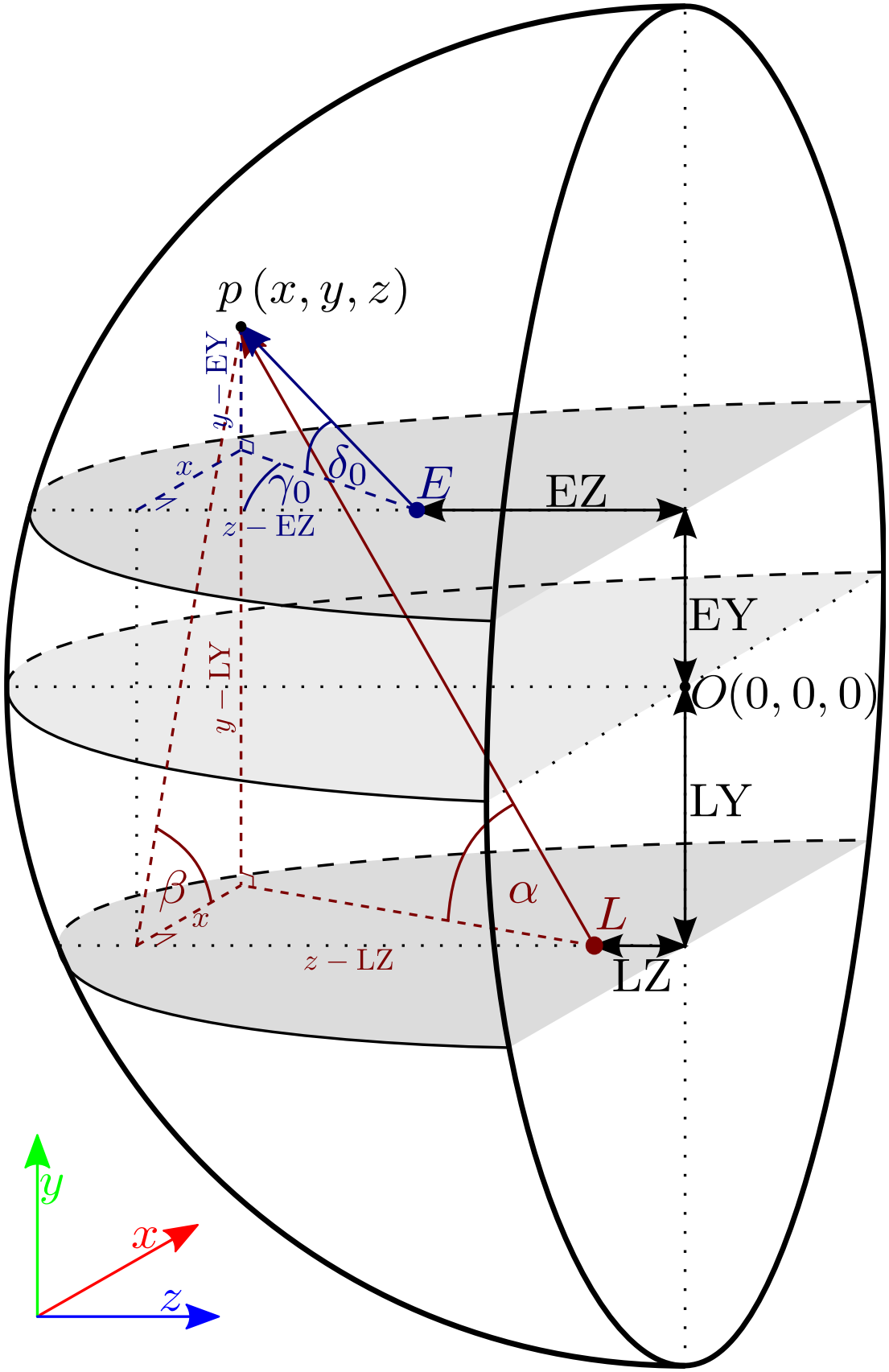


Figure 4.3: Relationship between the observation angles γ_0 and δ_0 (in dark blue) and the projections angles β and α (in dark red) of a point p projected by lens L at $(0, \mathrm{LY}, \mathrm{LZ})$ and observed from $E(0, \mathrm{EY}, \mathrm{EZ})$

which, after expanding, grouping terms together and simplifying, gives

$$\left(1+\frac{\tan^2\delta_0}{\sin^2\gamma_0}+\frac{\cos^2\gamma_0}{\sin^2\gamma_0}\right)x^2+2\left(\frac{\tan\delta_0}{\sin\gamma_0}\mathrm{EY}+\frac{\cos\gamma_0}{\sin\gamma_0}\mathrm{EZ}\right)x+\mathrm{EY}^2+\mathrm{EZ}^2-R^2=$$
$$=\frac{1}{\cos^2\delta_0\sin^2\gamma_0}x^2+2\frac{\mathrm{EY}\tan\delta_0+\mathrm{EZ}\cos\gamma_0}{\sin\gamma_0}x+\mathrm{EY}^2+\mathrm{EZ}^2-R^2=0 \tag{4.5}$$

The roots are expressed as

$$x=-\cos^2\delta_0\sin^2\gamma_0\left(\mathrm{EZ}\cot\gamma_0+\mathrm{EY}\csc\gamma_0\tan\delta_0\right. \tag{4.6}$$
$$\left.\pm\sqrt{\csc^2\gamma_0\left(-\left(\mathrm{EY}^2+\mathrm{EZ}^2-R^2\right)\sec^2\delta_0+\left(\mathrm{EZ}\cos\gamma_0+\mathrm{EY}\tan\delta_0\right)^2\right)}\right)$$

and correspond to the two points where the line defined by angles γ_0 and δ_0 and passing through the point $(0,\mathrm{EY},\mathrm{EZ})$ intersects the sphere defined in Equation 4.3 (if they exist) — one in front and one behind the observer. Which of the two roots is in front of the observer depends on the sign of the γ_0 angle. Hence, x is given by

$$x=-\cos^2\delta_0\sin^2\gamma_0\left(\mathrm{EZ}\cot\gamma_0+\mathrm{EY}\csc\gamma_0\tan\delta_0\right. \tag{4.7a}$$
$$\left.-\operatorname{sgn}(\gamma_0)\sqrt{\csc^2\gamma_0\left(-\left(\mathrm{EY}^2+\mathrm{EZ}^2-R^2\right)\sec^2\delta_0+\left(\mathrm{EZ}\cos\gamma_0+\mathrm{EY}\tan\delta_0\right)^2\right)}\right)$$

from which the other coordinates can finally be calculated using Equations 4.2a and 4.2b:

$$z = x\cot\gamma_0+\mathrm{EZ} \tag{4.7b}$$
$$y = x\tan\delta_0\csc\gamma_0+\mathrm{EY} \tag{4.7c}$$

The corresponding frame coordinates (β,α) depend on the position of the lens $(0,\mathrm{LY},\mathrm{LZ})$ relative to the centre of the sphere. The relationship is given by

$$\beta = \arctan\left(\frac{y-\mathrm{LY}}{x}\right) \tag{4.8a}$$
$$\alpha = \arctan\left(\frac{x}{(z-LZ)\cos\beta}\right) \tag{4.8b}$$

Thus, through the Cartesian coordinates of Equation 4.7, Equation 4.8 defines the frame coordinates (β,α) of a feature in the imaged scene in terms of the lens and eye positions and its reference coordinates (β_0,α_0) (obtained without sweet spot correction). It relies on assumptions based on the text of SPIClops API manual (ELUMENS, 2001) and constitutes the hypothesis to be tested.

4.2.2 Methods

Data

To test the hypothesis, a set of frames stitched with different settings from the same images of a bespoke virtual environment was obtained and analysed in Matlab®. The

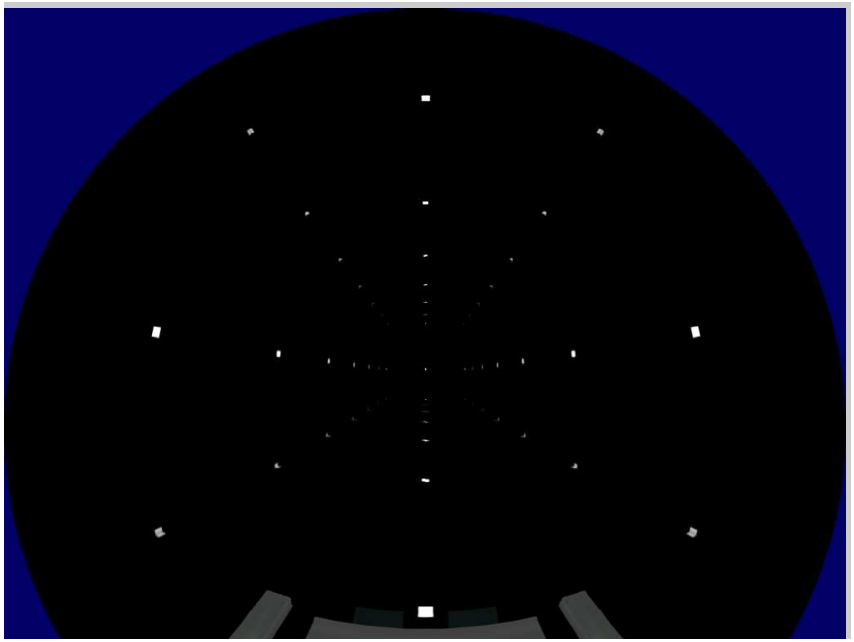
VE was a simple corridor — 12.064 m in length, 3.064 m tall and equally wide (the additional 6.4 cm resulted from an initial misunderstanding about the "wall thickness" setting in UnrealEd) — with regularly placed white marks on otherwise perfectly black walls, ceiling and floor. The marks were produced with the textures presented in Appendix A and appeared every 1.5 m along the length of the corridor at the borders between the walls and the floor/ceiling, along the centre line of the floor and the ceiling, and at a height of 1.2 m on the walls. The wheelchair model was placed 2 m from one end of the corridor facing directly the other end. The images from its four cameras were acquired once and subsequently used to stitch multiple frames with SPIClops. Each of the four settings (EY, EZ, LY, LZ) were varied by ± 0.5 from a starting point in steps of 0.1 while all other settings were kept constant, giving 11 frames per variable. Two sets of images were obtained: one where all variables were set to zero as a starting point, and one where the starting point was a combination of settings found to produce qualitatively correct looking scenes when projected on the dome: $(0.43, 0.28, 0, -0.6)$. In total, 82 frames were acquired and saved in a bitmap format to prevent any loss of data due to compression.

Analysis

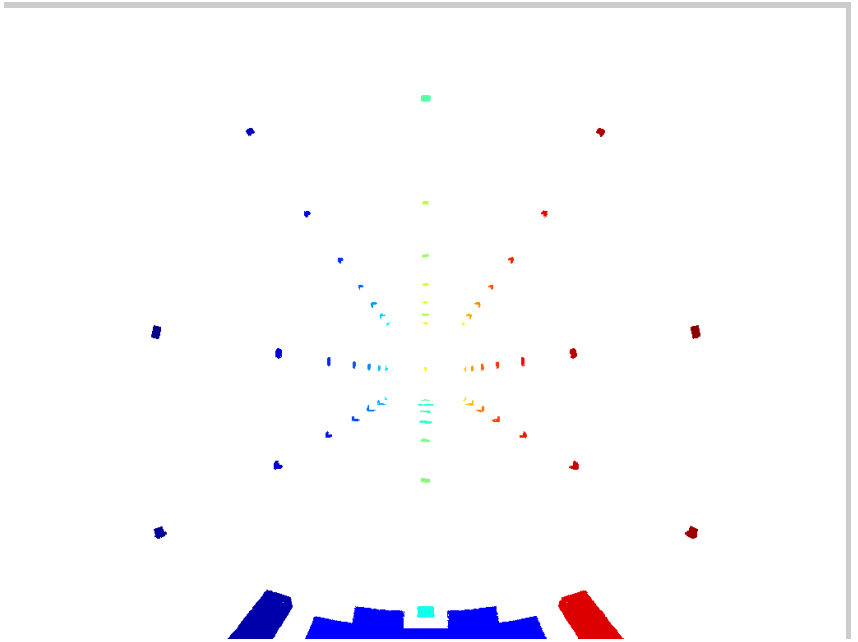
Analysis of the acquired frames presented two main challenges — to extract the positions of the marks (and only the marks) and to automatically match the marks in different frames. This subsection presents the algorithms developed to address these problems. They rely on several functions from the Matlab® Image Processing Toolbox.

First, each image was converted from 8-bit RGB (Figure 4.4a) to grayscale and then, using a threshold of $1/255$ (the smallest non-zero value in an 8-bit grayscale image), to a binary image. The low threshold ensured that none of the distant marks which are small and faint were lost. Using the `bwlabel` function, each 8-connected region of the binary image was assigned a unique label, producing a label image, as in Figure 4.4b. Subsequently, the labels representing parts of the wheelchair were recognised as those in contact with the bottom border of the image and removed from the label set. Similarly, small artefacts at the border of the image-containing circle were removed. At times, a single mark would appear on the image in two or more parts and not as a single 8-connected region. Those were merged (i.e. assigned the same label) with Algorithm 4.1 relying on the distance transform, as shown in Figure 4.4c. Finally, the centroid of each mark region was calculated (Figure 4.4d) resulting in a list of point coordinates.

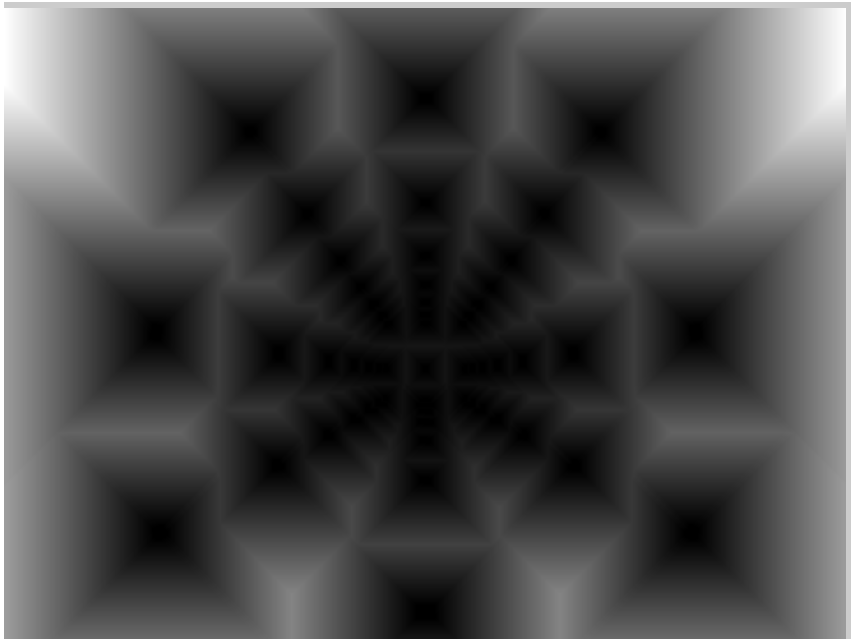
Corresponding to the number of marks detected, this list had in general different length and ordering of marks for each frame. Thus, an algorithm was developed to synchronise the lists between two frames. Consider the template frame I, another frame I' stitched with different settings than I, and the corresponding lists of labels $\mathcal{L}$ and $\mathcal{L}'$ indexed with variables i and j, respectively. The problem, then, is to find a mapping M such that $\mathcal{L}'[\mathrm{M}(i)]$ and $\mathcal{L}[i]$ correspond to the same mark. This problem is



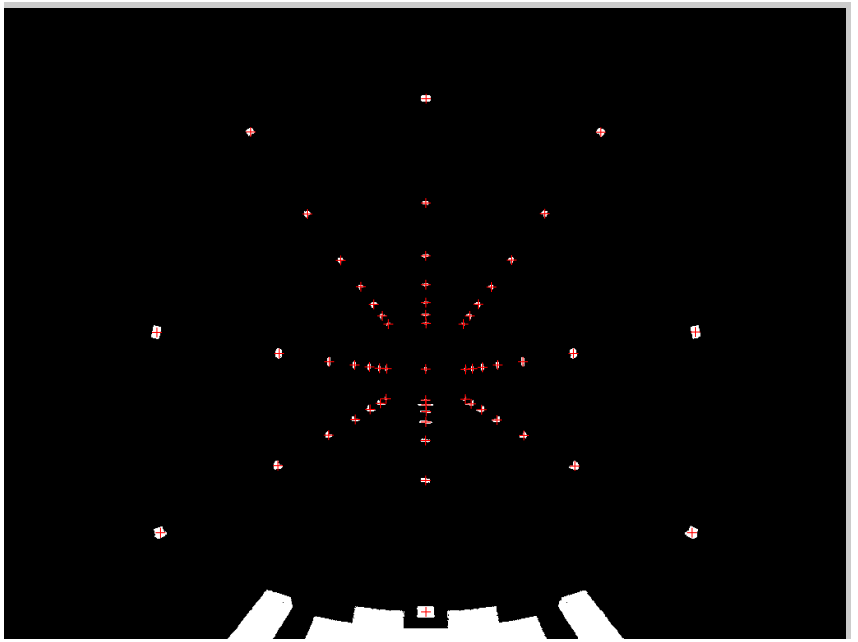
(a) Original image



(b) Unprocessed label image. Each label is assigned a different colour.



(c) Distance transform. Brightness increases with distance.



(d) Final result

Figure 4.4: Marker position extraction. After labelling, the distance transform is used to detect and merge disjoint marks.

Algorithm 4.1 Pseudocode for connecting disjoint marks

Require: L — set of all labels
Require: I_{L} — label image
Require: $S \subseteq L$ — set of labels whose regions are smaller than 7 pixels
for all $s \in S$ **do**
  $D \leftarrow$ distance transform of $I_{L/s}$
  $C \leftarrow \left\{ p \in I : \big(D(p) = 2\big) \wedge \big(I_{\mathrm{L}}(p) = s\big) \right\}$ {set of pixels labeled s whose distance to a region with another label is 2}
  if $C \neq \emptyset$ **then**
    $N \leftarrow \emptyset$ {set of neighbouring regions}
    for all $c \in C$ **do**
      $L_{\mathrm{N}} \leftarrow \{l \in L : \exists_{t:\|c-t\|\leq 2} \quad I_{L/s}(t) = l\}$
      $N \leftarrow N \cup L_N$
    end for
  end if
  for all $n \in N$ **do** {assign all neighbours the same label}
    $\forall_{p:I_{\mathrm{L}}(p)=n} \quad I_{\mathrm{L}}(p) = s$
  end for
end for

not trivial even for frames acquired with settings differing only in one variable and by a single 0.1 step due primarily to the close packing of marks in the centre of the scene. There, a mark k with position p_k on I could move to a new position p'_k on I' such that, for some other mark x on I, $\|p'_k - p_x\| < \|p'_k - p_k\|$ which means that matching each mark on I to the closest mark on I' would in general not produce the correct mapping M. However, such trivial mapping $\mathrm{M_T}$ works well for marks on the periphery of the image where they are more sparse. Thus an iterative algorithm was developed using $\mathrm{M_T}$ calculated from an increasing set of peripheral marks to transform I' into I in a process called image registration. This brings p'_k closer to p_k for all k at each iteration. The process is repeated until $\mathrm{M_T}$ produces a unique mapping exhausting all marks in either $\mathcal{L}$ or $\mathcal{L}'$, or no improvement in the number of non-unique matches is made, as detailed in Algorithm 4.2. This algorithm was applied sequentially to all frames starting with a reference frame that contained the most marks (60) such that each frame was compared to its "neighbour" — a frame acquired with stitching settings differing by 0.1 in one variable only — thus propagating the same mark labels to all frames in a set. The procedure was first completed on the set of frames stitched with settings centred on $(0, 0, 0, 0)$. To ensure the same labels are applied in the second set, a frame with settings $(0.43, -0.28, 0, -0.30)$ from the second set was matched with one with settings $(0.4, 0, 0, 0)$ from the first set and then used as a starting point for the propagation of labels.

At the end of the process, each visible mark in the VE had a unique label and its positions on the frame image under different stitching settings were assembled in an array indexed with the label. To test the hypothesis, using the mark positions on the

Algorithm 4.2 Pseudocode for the `MatchLabels` routine.

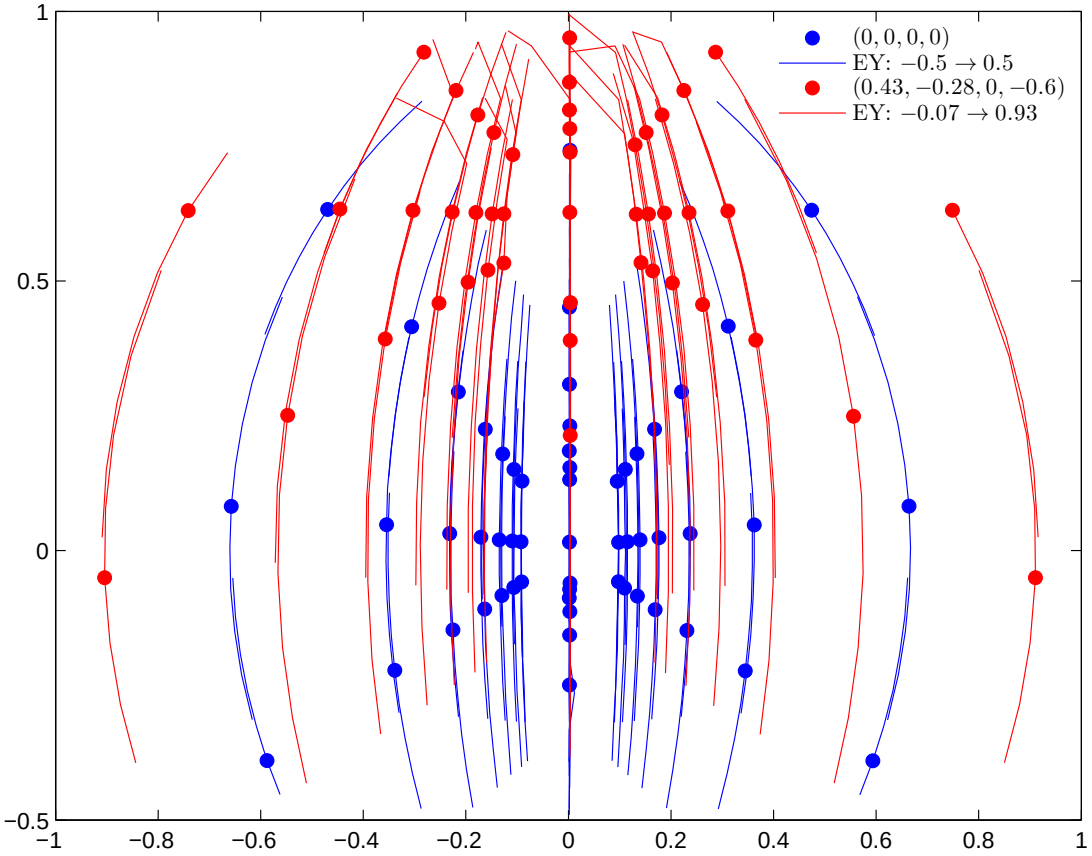
Require: list $\mathcal{L}$ of marks in image I
Require: list $\mathcal{L}'$ of marks in image I'
Require: function `DistMatch`$(\mathcal{L}, \mathcal{L}')$ implementing $\mathrm{M_T}$ and returning a list containing for every mark in $\mathcal{L}'$ the closest mark in $\mathcal{L}$

$\mathcal{T} \leftarrow \mathcal{L}'$ {local copy for modification}
$n \leftarrow \#\mathcal{L}'$
$n_\mathrm{U} \leftarrow \lceil 0.7\, n \rceil$ {initialise number of unmatched marks}
$n_\mathrm{I} \leftarrow n$ {initialise improvement counter}
repeat
  $n_\mathrm{C} \leftarrow \lceil \frac{1}{2}(n - n_\mathrm{U}) \rceil$ {number of marks to consider}
  $\mathcal{N}_\mathrm{C} \leftarrow$ list of n_C points furthest from the centroid of $\mathcal{T}$
  $\mathcal{M} \leftarrow$ `DistMatch`$(\mathcal{L}, \mathcal{N}_\mathrm{C})$
  $\mathrm{T} \leftarrow$ projective transform mapping $\mathcal{N}_\mathrm{C}$ to M
  $\mathcal{T} \leftarrow \mathrm{T}(\mathcal{T})$ {image registration}
  $\mathcal{M} \leftarrow$ `DistMatch`$(\mathcal{L}, \mathcal{T})$
  $tmp \leftarrow 2\big(n - \#\mathrm{Unique}(\mathcal{M})\big)$
  $n_\mathrm{I} = n_\mathrm{U} - tmp$
  $n_\mathrm{U} = tmp$
until $n_\mathrm{U} \leq 0$ or $n_\mathrm{I} \leq 0$
if $n_\mathrm{U} > 0$ **then**
  `RemoveDuplicates`$(\mathcal{M})$
end if
$\forall_{0<i\leq n}\mathcal{R}[i] \leftarrow \begin{cases} \mathcal{L}'[m] & \text{if } \exists_m \mathcal{M}[m] = \mathcal{L}[i]\,, \\ 0 & otherwise \end{cases}$
return $\mathcal{R}$

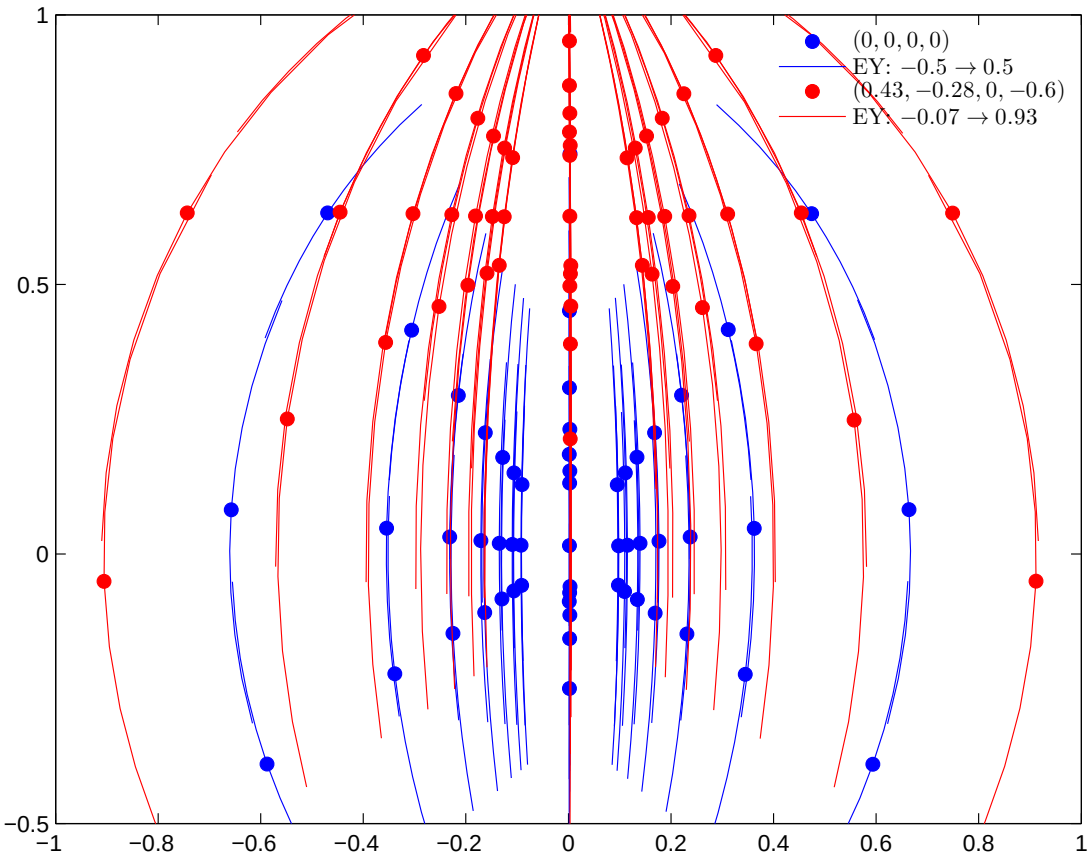
frame acquired with all variables kept at zero as a reference and the equations developed in subsection 4.2.1, predicted positions for each mark were calculated for all settings combinations at which frames were acquired. The measured and predicted positions were than compared in terms of distance, as described in the following subsection.

4.2.3 Results and Discussion

Figures 4.5 to 4.8 present the results of the study of the effect of the stitching settings EY, EZ, LY and LZ exposed by the SPIClops API on the resulting stitched images. The top part of each figure presents the positions of the different marks in normalised image coordinates (c.f. Figure 4.1b) measured as described above. The positions of each mark across the different settings are connected with a line and the starting frame of each data set is shown with dots. For each setting, a clear and smooth pattern emerges. With a few exceptions at extreme settings, visible as broken lines e.g. at the top of Figure 4.5a, the algorithms for mark position extraction and labelling were successful, although some marks that have been correctly extracted were lost in the label synchronisation process at more extreme settings. The bottom graph in each figure presents the corresponding predicted positions of the marks. The excellent agreement between

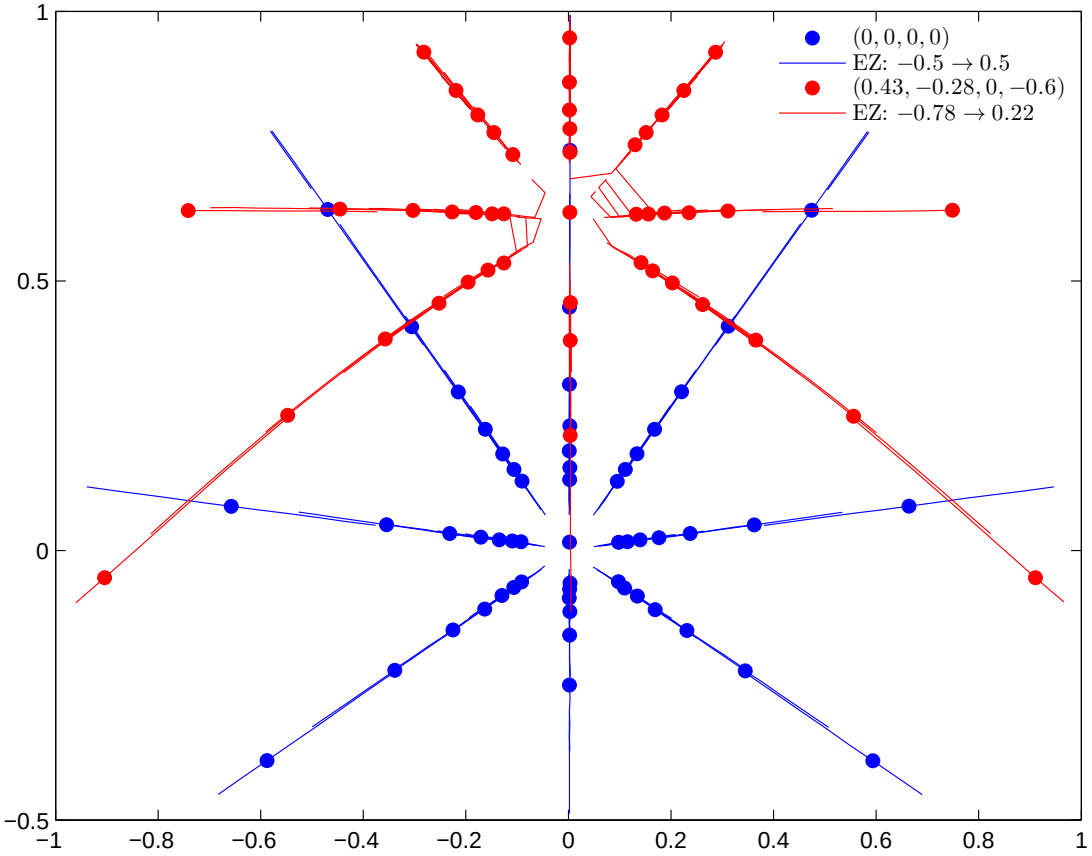


(a) Measured mark positions

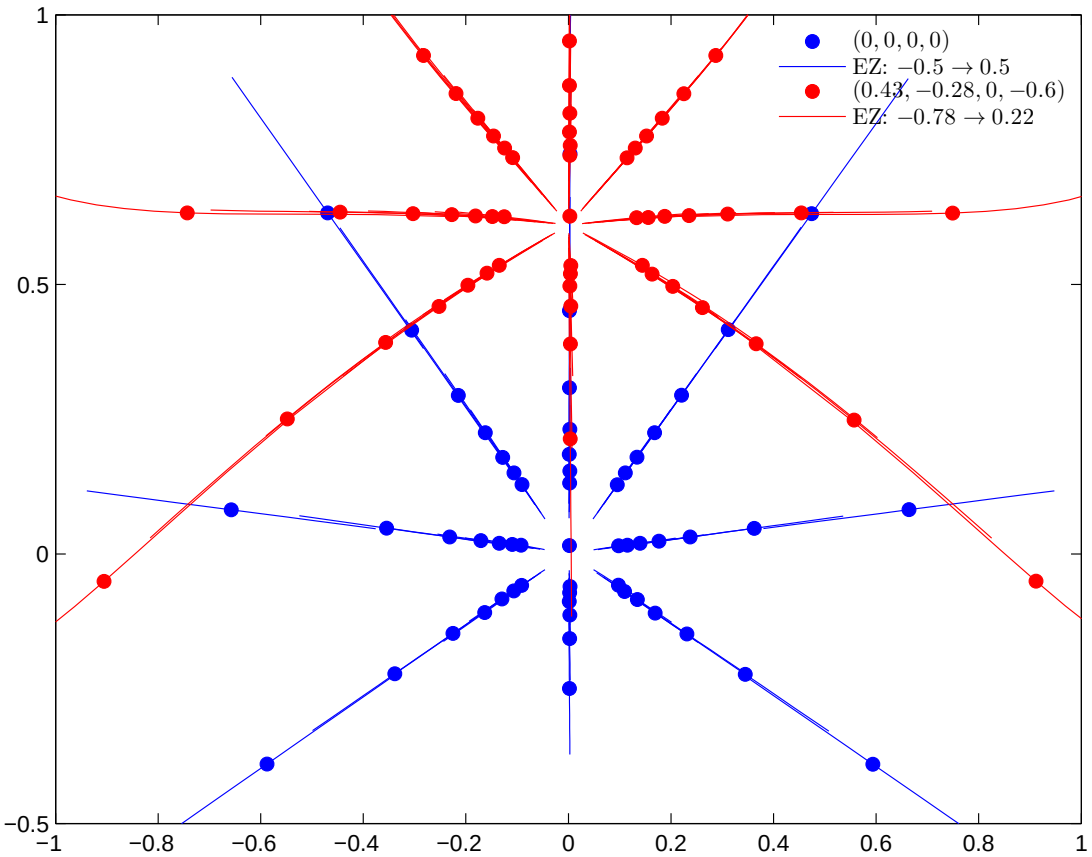


(b) Predicted mark positions

Figure 4.5: Measured and predicted mark positions as the viewing position used for stitching is moved up and down from two starting positions: $(0, 0)$ in red and $(0.43, -0.28)$ in blue. Marks move upwards with increasing EY.

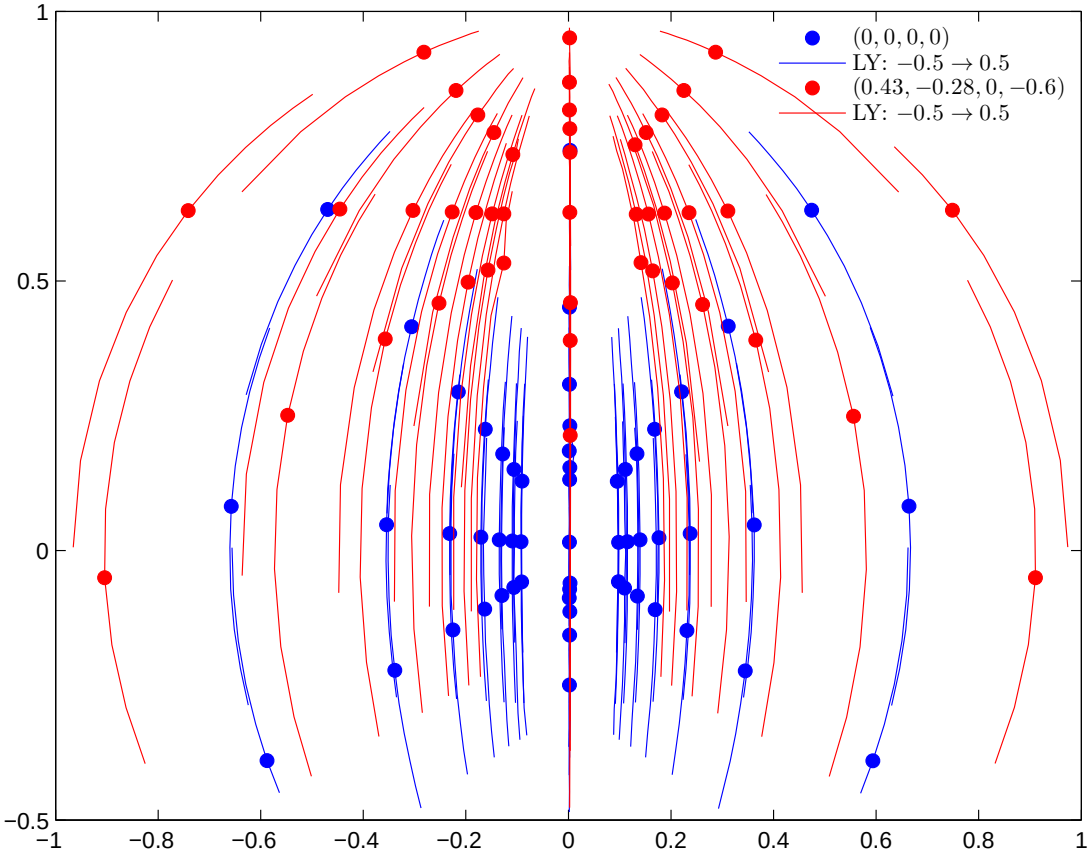


(a) Measured mark positions

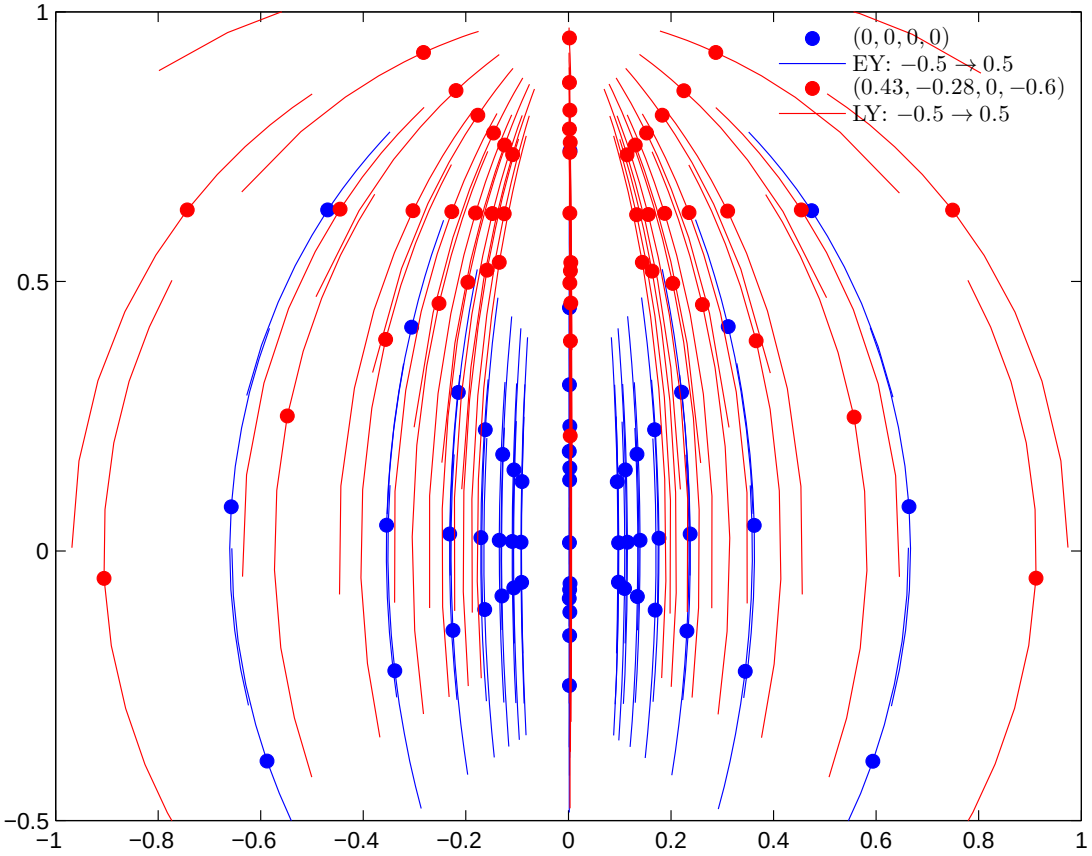


(b) Predicted mark positions

Figure 4.6: Measured and predicted mark positions as the viewing position used for stitching is back and forth from two starting positions: $(0, 0)$ in red and $(0.43, -0.28)$ in blue. Marks move outwards with increasing EZ.

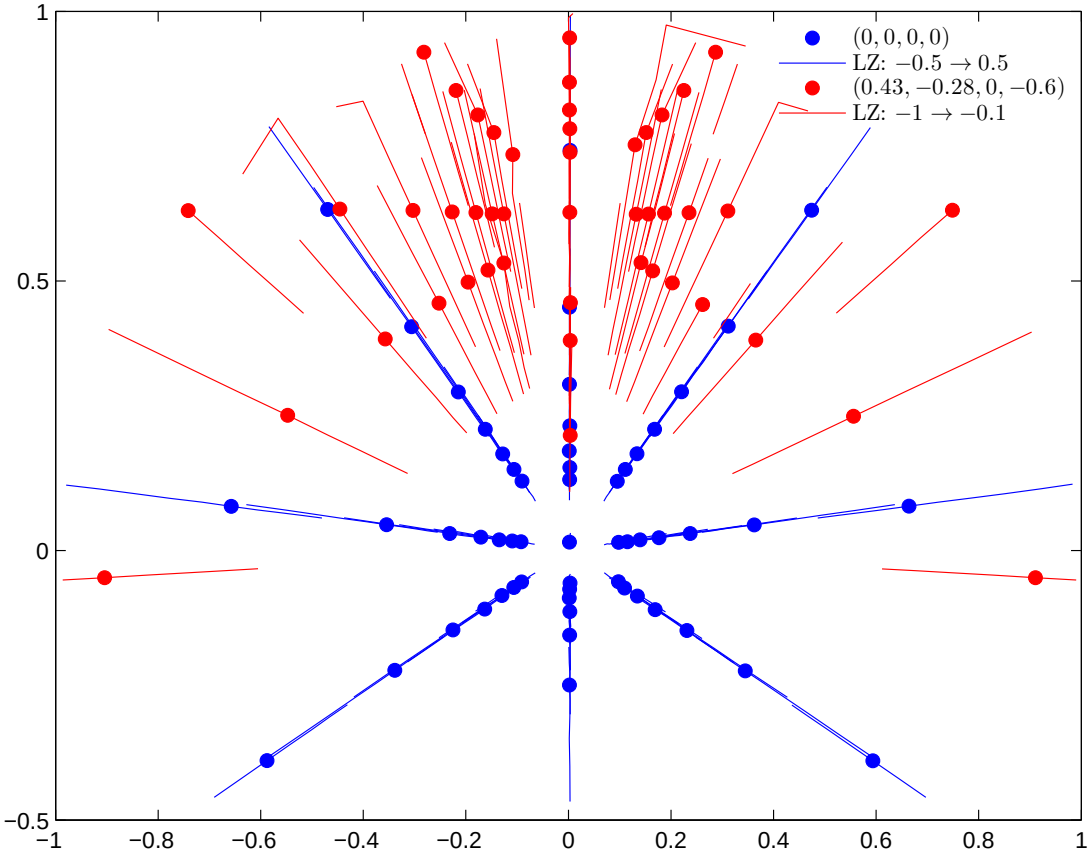


(a) Measured mark positions

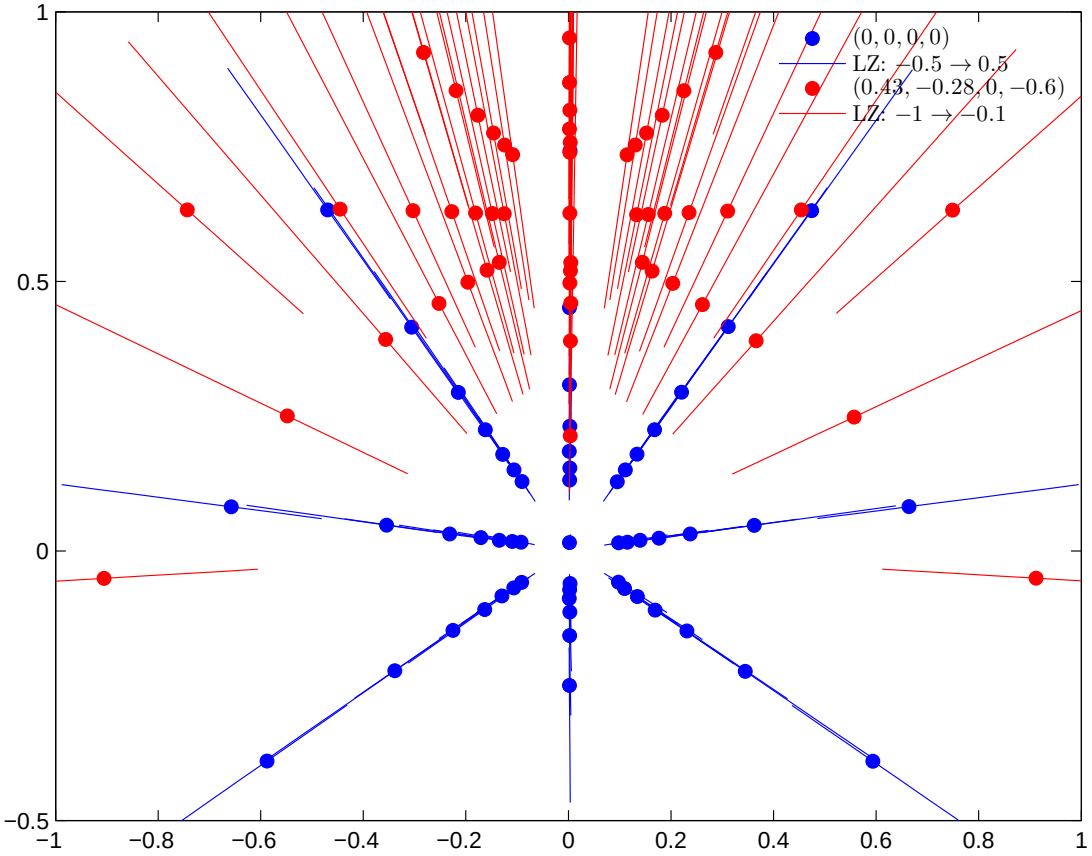


(b) Predicted mark positions

Figure 4.7: Measured and predicted mark positions as the lens position used for stitching is moved up and down from two starting positions: $(0, 0)$ in red and $(0, -0.6)$ in blue. Marks move downwards with increasing LY.



(a) Measured mark positions



(b) Predicted mark positions

Figure 4.8: Measured and predicted mark positions as the viewing position used for stitching is moved up and down from two starting positions: $(0, 0)$ in red and $(0, -0.6)$ in blue. Marks move inwards with increasing LZ.

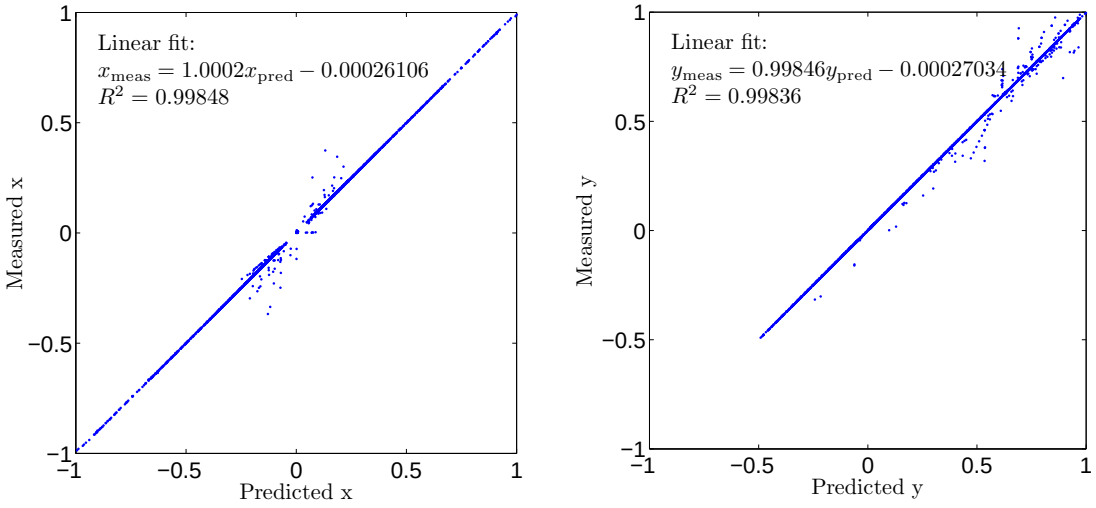


Figure 4.9: Comparison of predicted and measured mark positions, in terms of the x and y normalised coordinates on the left and right respectively, in the sample of 3988 extracted positions.

predicted and measured mark positions is further evidenced by the regression analysis presented in Figure 4.9 for each coordinate. Among the 3988 measured mark positions, the regression coefficient was almost exactly one (with precision to 2 significant digits) with $R^2 > 0.998$. The average distance between measured and predicted mark positions (including the few wrongly matched marks) was 1.875 pixels (with standard deviation of 8.99).

Overall, sufficient evidence was found to accept the model developed in subsection 4.2.1 as a true representation of the way the different sweet spot settings influence the frames stitched by SPIClops. The model can be used to construct a frame with any combination of settings given a reference image. This, in turn, allows for programmatic optimisation of the settings with respect to some quantitative criterion based on the produced frames, which is one of the aims of this chapter.

4.3 Lens Model

To optimise the stitching settings such as to produce a desired image on the dome, it is necessary to model the relationship between a pixel's position on the stitched frame and its projection angle from the lens onto the dome, i.e. to calibrate the lens. As stated in the manual (ELUMENS, 2001) and confirmed in the preceding section, the SPIClops library operates under the assumption of a lens with 180° angular field of vision (AOV) and a one to one relationship between a pixel's distance from the image centre (normalised, c.f. Figure 4.1b) and the lens projection angle (between 0 and 90°). However, even superficial inspection of an image projected by the lens suffices to falsify these assumptions — the dome must be a few centimetres in front of the lens for the

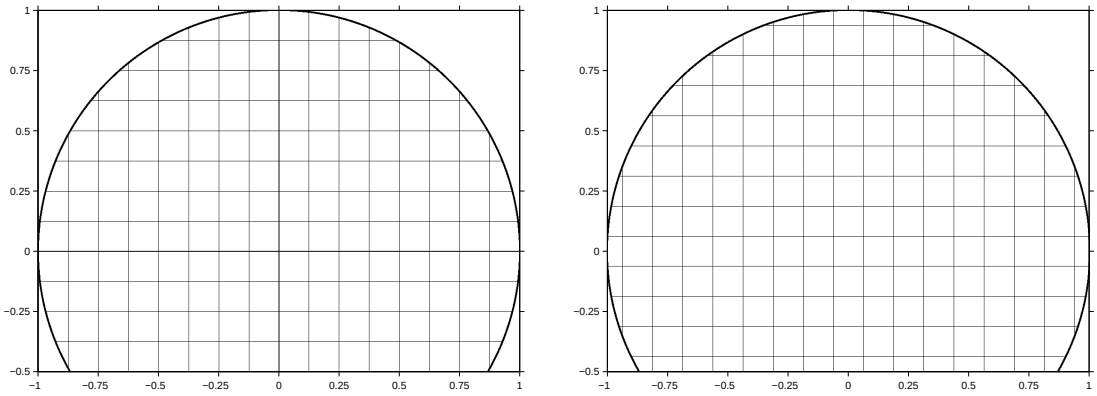
projected image to fill the screen. Since no method to calibrate a wide-angle projector lens was available in the literature[1], the required relationship between pixels' positions and their projection angles was obtained through a novel approach.

The process of projector lens calibration is most often carried out in the context of multi-projector systems and augmented reality, as in e.g. Raskar et al. (1999) or Brown et al. (2005), and employs an already calibrated camera to obtain the required measurements. However, a camera-based method would not be practical for calibrating the VisionStation® system. To capture the entire scene, the camera would either have to be positioned several meters behind the sweet spot, likely introducing inaccuracies at the periphery of the dome screen, or be equipped with a fish-eye lens, which are expensive and whose calibration is itself an active area of research (e.g. Kannala and Brandt, 2006; Dunne et al., 2010). Thus, the novel method presented here uses a theodolite equipped with an electro-optical distance measuring device and a laser pointing vertically down to measure the requisite angles. An additional benefit of this approach is that the relative positions of the theodolite, the lens and the dome can be measured with great accuracy using the theodolite.

4.3.1 Methods

The projector was held by the newly constructed projector mount, as described in section 2.4.1, allowing for accurate and repeatable positioning with respect to the dome. The theodolite was carefully levelled and positioned over the centre line of the dome screen (marked on the floor) and, with its help, so was the projector lens. Subsequently, the height of the theodolite, its distance from the dome screen and the position of the lens were measured. Two frames with rectangular grids covering the image area were generated programmatically, as shown on Figure 4.10. The grids were originally prepared in white on black background, but due to visible chromatic aberration the white was replaced with green as the middle wavelength component in the RGB system. The positions of the grid intersections were measured with the theodolite as azimuth and elevation angles with accuracy to 10 seconds. This process was carried out twice, once for each grid, with the lens in different position each time, as depicted on Figure 4.11 — first in a position in which the projected image maximally filled the dome screen and second with the projector placed higher leaving sufficient clearance for the legs of a seated person. Through calculations similar to those presented in subsection 4.2.1, the projection angles α_p and β_p for every intersection point were calculated from the measured azimuth and elevation angles, taking into account the relative positions of the theodolite, the lens and the dome screen. Bringing the results from both sets of measurements — one for each grid and lens position — into the same coordinate system verified the accuracy of the measurements and the method. A model of the lens was acquired as a local weighted mean transformation (Goshtasby, 1988) from the image co-

[1] to the best of the author's knowledge



(a) (b)

Figure 4.10: Programmatically generated frames for projection on the dome to obtain a relationship between pixel position and projection angle. Each grid contains a different set of intersection points, but combined the two frames provide a uniform distribution of points. Grids were projected green on black background.

ordinates $(\beta_\mathrm{i}, \alpha_\mathrm{i})$ of the grid intersections to their projection angles $(\beta_\mathrm{p}, \alpha_\mathrm{p})$ calculated with the Matlab® Image Processing Toolbox. Although less readily interpretable, this image registration method was preferred over global transformations such as affine, projective or polynomial, which did not produce sufficiently good alignments, and the piecewise linear mapping (Goshtasby, 1986) which did not generalise well.

4.3.2 Results

The observation angles of the intersection points on the projected grids as measured with the theodolite are presented in Figure 4.12. The corresponding lens projection angles are contrasted with their expected positions (true frame coordinates) in Figure 4.13a. Measurements acquired in both positions align in a clear pattern validating the calculations. The AOV of the lens is clearly smaller than 180°. The result of registering the grid frames with their projections is shown in Figure 4.13b and the transformation is visualised in Figure 4.14. The transformation is approximately symmetric. The projected image appears lower than expected and is compressed in a non-uniform fashion with greater degree of compression at the periphery. Additionally, there is a slight tilt in the projected image.

Reversing the calculations carried out earlier, the observation angles of the projected points can be predicted from their projection angles calculated, in turn, by applying the transformation obtained above to their frame coordinates. Figure 4.15 presents the result of such a process, successfully recovering the measured angles for both grids and positions and validating the lens model obtained. The average error is 0°8’50” in azimuth and 0°5’42” in elevation.

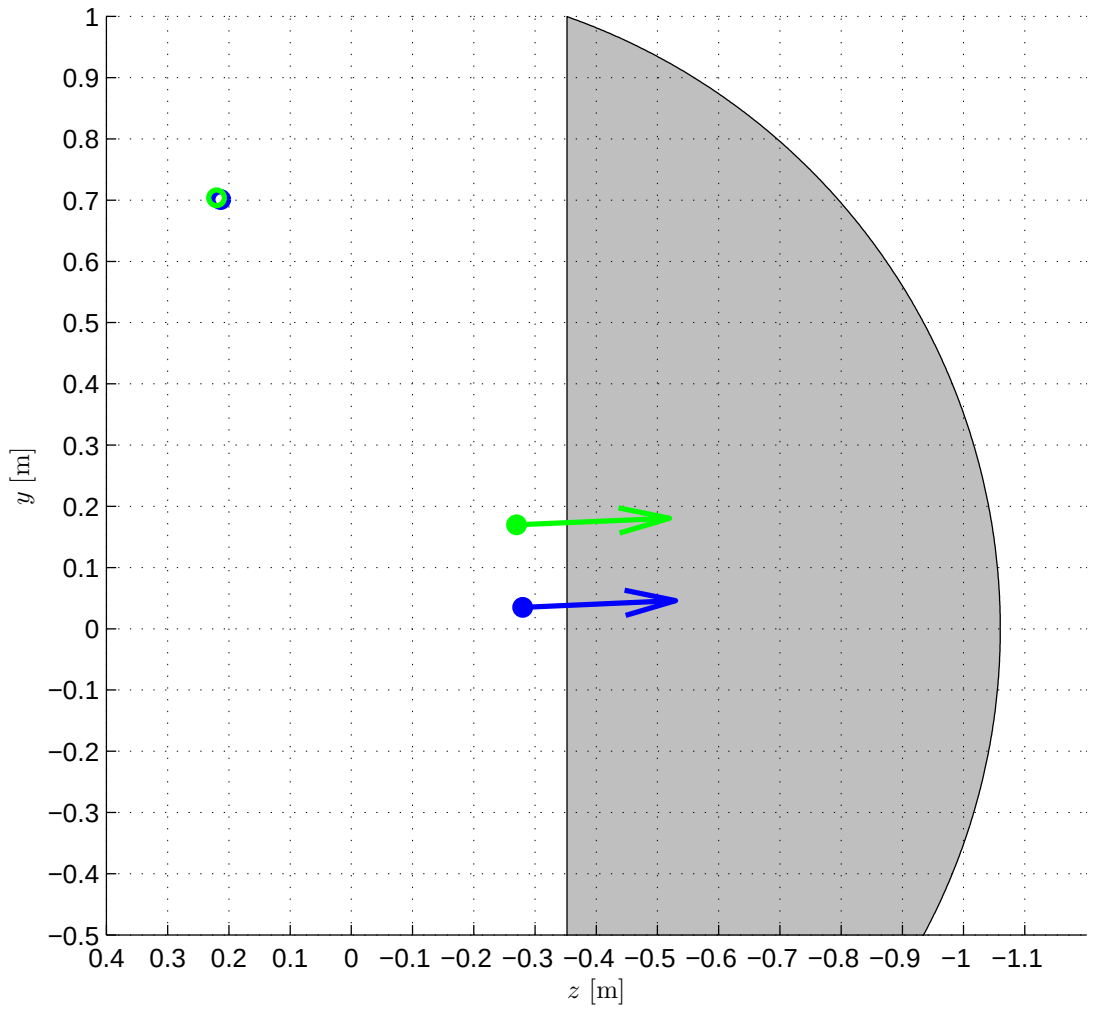
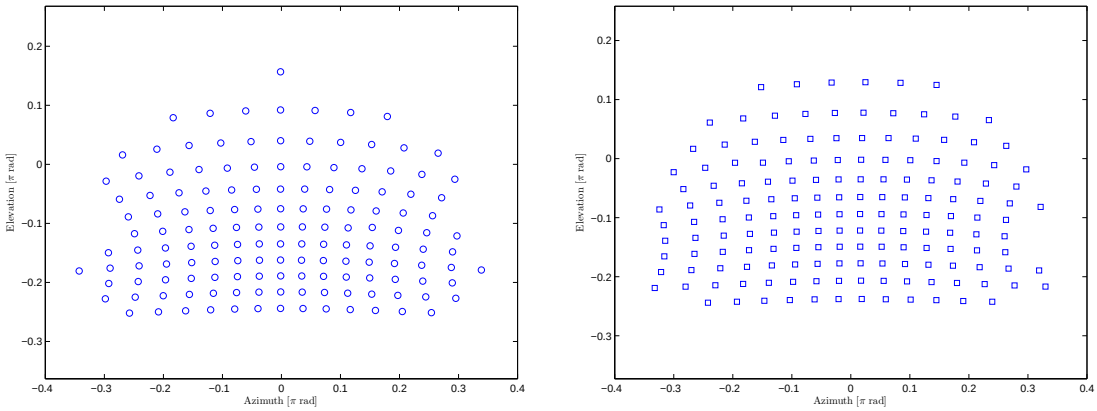


Figure 4.11: Lens (disks) and theodolite (circles, almost overlapping) positions in the dome coordinates system during measurements of the intersection point positions on projections of the grids in Figure 4.10a and 4.10b in blue and green, respectively. View from the right. Arrows indicate the tilt of the projector necessary for the projected image to maximally fill the dome.

4.4 Optimisation

Having both a model of the SPIClops library and the lens allows to predict the observation angle of any feature on the projected frame (given a reference frame). If an objective function can be constructed to quantify the quality of the (predicted) observed scene, a vector of optimal SPIClops settings (EY, EZ, LY, LZ) can be found for any viewer position. Such an objective function could measure, for instance, how straight different lines appear or, for a set of landmarks, the deviation between their desired and actual observation angles. Here, the latter approach is chosen. To this end, the VE described in subsection 4.2.2 was modelled in Matlab® taking into account both the observer's position E and the point P from which the VE was viewed when the refer-

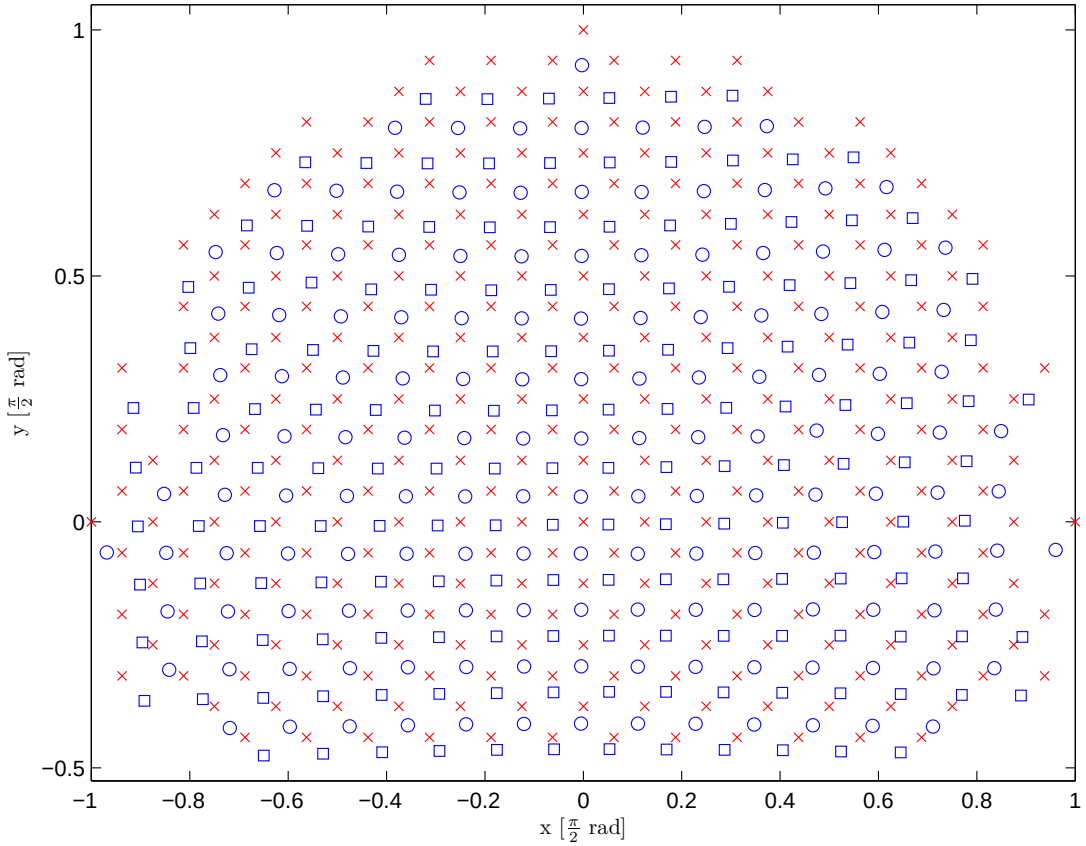


(a) (b)

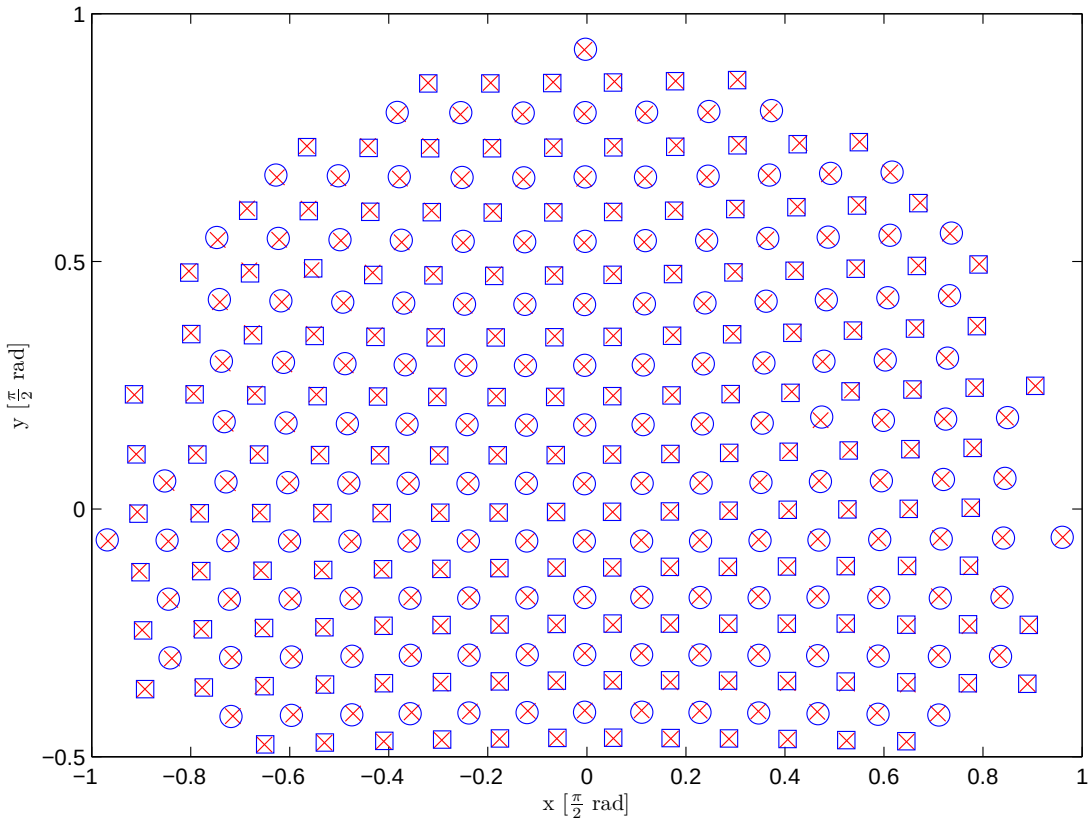
Figure 4.12: Theodolite measurements of the intersection points of the two grids presented in Figure 4.10 when projected on the dome from the lens in positions indicated in Figure 4.11.

ence frame was acquired. The model is presented in Figure 4.16. Using the model, the observation angles at which the different landmarks would appear if the observer were at point P within the VE were calculated, as shown on Figure 4.17 (dots). Treating the azimuth-elevation coordinates as a Cartesian plane, the objective function was defined as the sum of Euclidean distances between the desired (i.e. obtained from the model) and simulated (by applying different SPI settings to a reference image) mark locations. An additional additive term was introduced to penalise combinations of settings that resulted in a smaller number of projected marks. The optimisation was carried out using the Matlab® built-in `fminsearch` function, which implements the simplex search method of Nelder and Mead (1965). Other methods from the Optimization Toolbox were also explored. In particular, to exploit the constraints of the search space (observer cannot be in front of the lens), the `fmincon` function implementing constrained minimum search for non-linear functions (Coleman and Li, 1996) was tried, but in a number of trials it did not always converge and when it did, the results were identical to those of `fminsearch`. The predicted location of the marks under optimal SPI settings (rounded to 2 decimal places) for the lens and observer positions as modelled in Figure 4.16 is presented in Figure 4.17 (circles). In this case, the average distance between desired and simulated mark positions is 53.1×10^{-3} rad. Figure 4.17 also presents the actual measured positions of the marks under the optimal SPI settings (red crosses), showing good agreement. The average distance from their predicted locations is 6.51×10^{-3} rad and the errors increase visibly towards the periphery where, due to the curvature of the dome, even small deviations have a large effect.

A Matlab® script implementing the optimisation procedure is included in Appendix A. It allows calculating the optimal SPI settings given a user's viewing position relative to the dome (eye level and horizontal distance). In general, such calibration

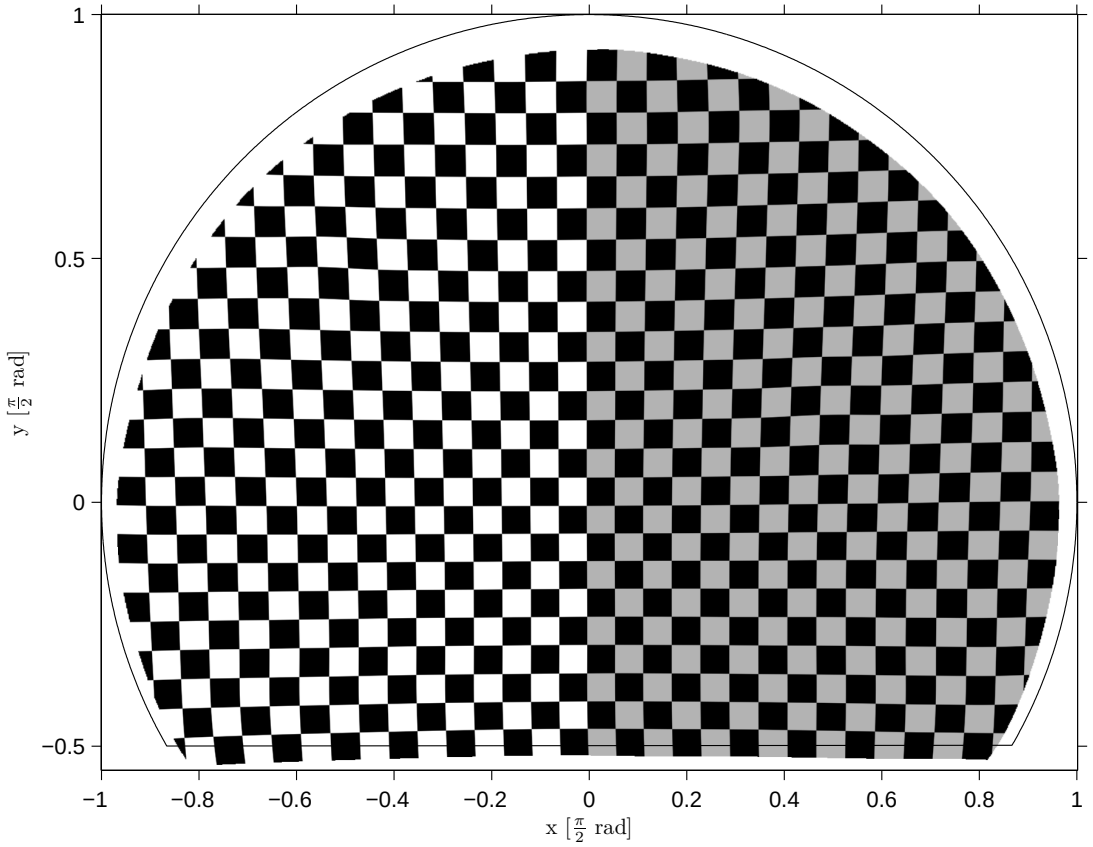


(a) Before registration. Average error 0.1104 rad (std 0.0231 rad).

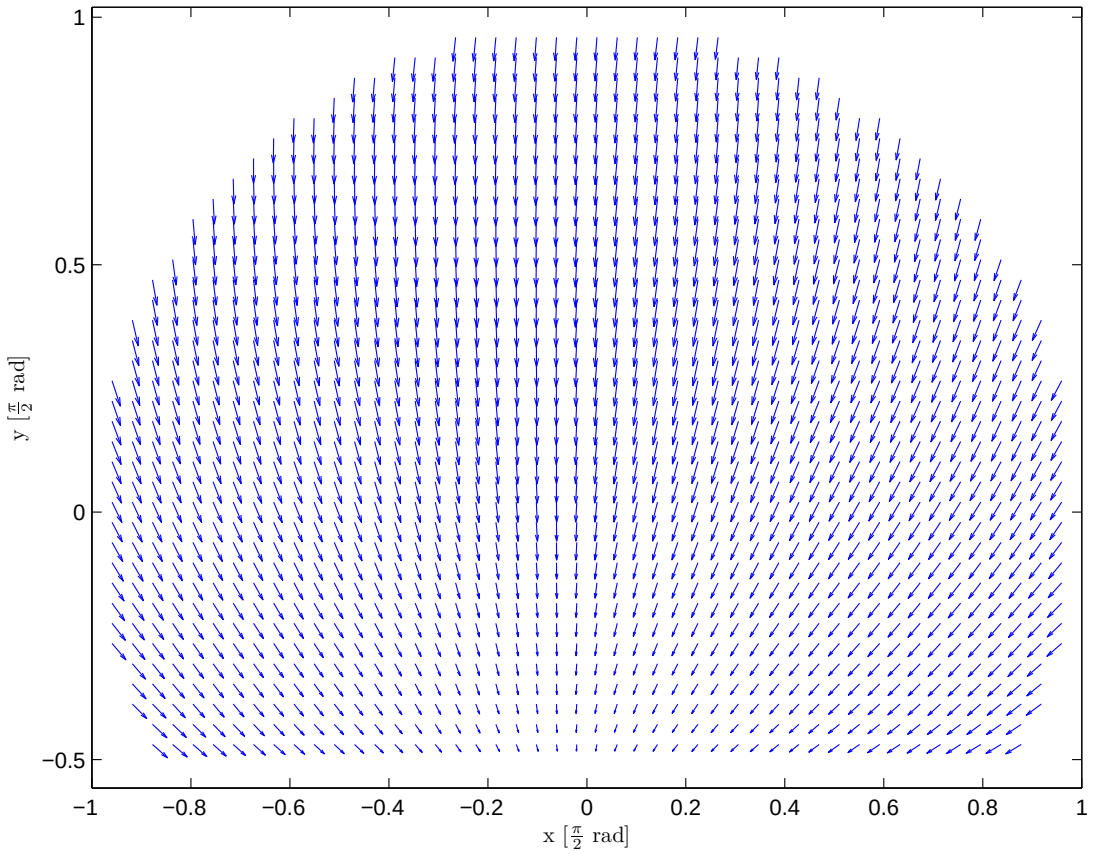


(b) After registration. Average error 0.0031 rad (std 0.0010 rad)

Figure 4.13: (a) The measured positions expressed in the normalised image coordinates in blue against their locations on the displayed frames in red (c.f. Figure 4.10) and (b) the result of the registration process. Error calculated as Euclidean distance.



(a)



(b)

Figure 4.14: Results of applying the image-to-lens transformation (a) to a checker board image (original shape outlined) and (b) to a set of points distributed uniformly over the image area, visualised as a vector field.

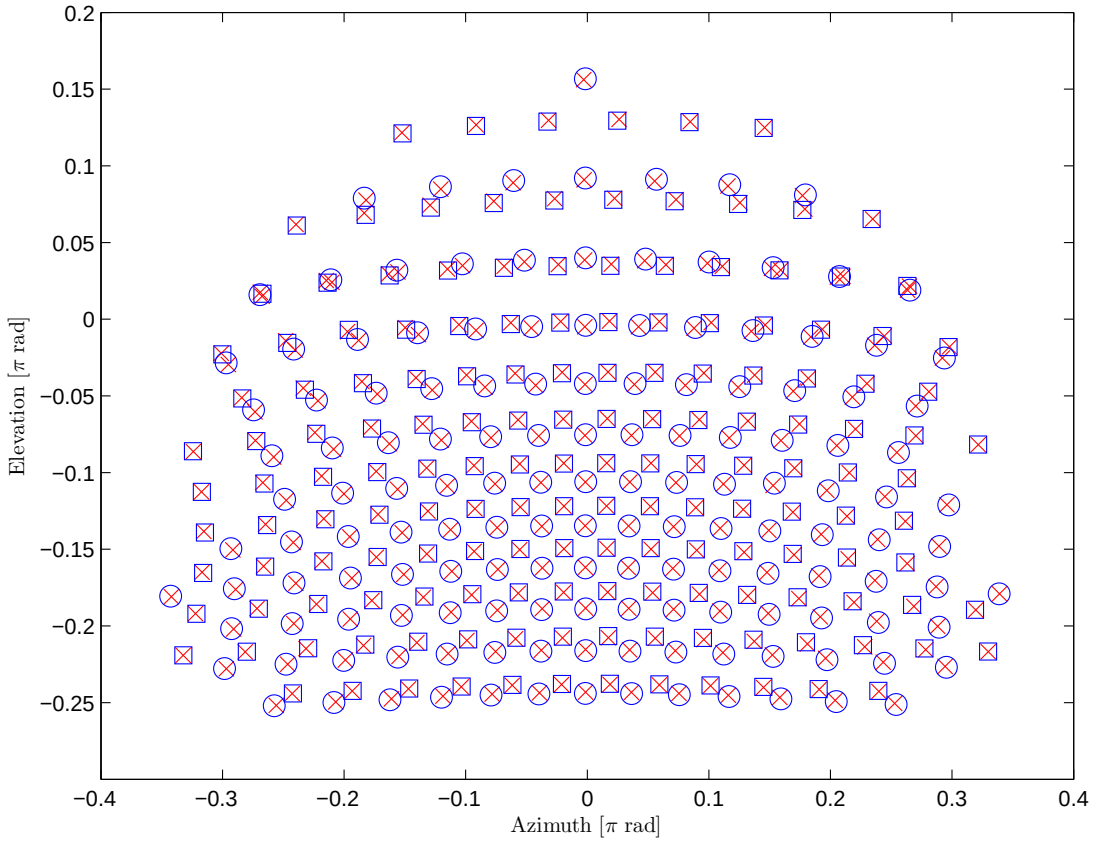


Figure 4.15: Measured observation angles in the two data sets in blue against the predictions based on the lens model in red.

will be required for each new user.

4.5 Size and Distance

To further investigate the perception of size and distance, two objects have been placed in the virtual environment, as shown on Figure 4.18 representing the projected frame. The azimuth and elevation angles of all visible corners of the two boxes have been measured with the theodolite. As mentioned earlier, without stereo vision, it is impossible to estimate the size and distance of an object simultaneously. Thus, the two are analysed separately — first the size of the objects is estimated assuming the distance is known and, second, the distance is estimated assuming the size is known.

For any point (x, y, z), the following relations hold for the azimuth γ and elevation δ in the dome coordinate system:

$$\tan\gamma = \frac{x}{z} \tag{4.9a}$$

$$\tan\delta = y\frac{\cos\gamma}{z} \tag{4.9b}$$

Assigning each corner a three letter identifier as depicted on Figure 4.19, where the

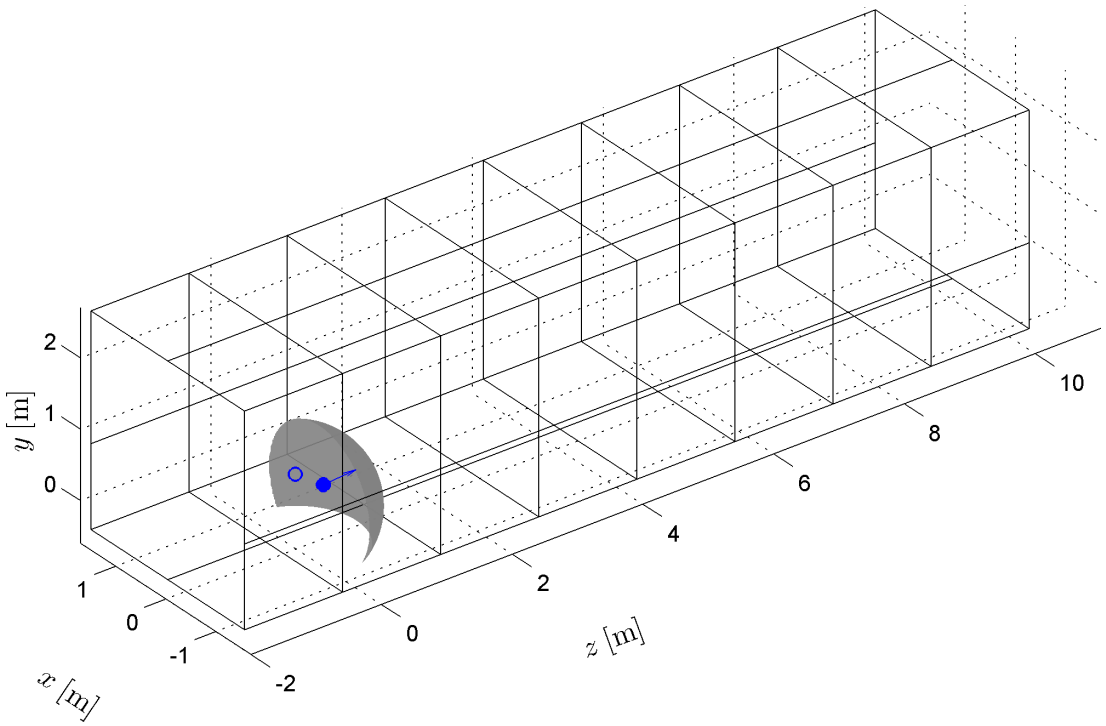


Figure 4.16: Matlab® model of the VE coinciding the viewpoint in VE with the observer's position relative to dome centre. Wireframe intersections indicate landmark positions.

first letter is either F or R for the front or rear face, the second is one of L and R for left and right, respectively, and the last is T or B for the top and bottom face, and calling d the distance to the front face along the z axis, the height h, width w and length l of the box can be expressed as:

$$h = y_{\mathrm{FLT}} - y_{\mathrm{FLB}} = d\left(\frac{\tan\delta_{\mathrm{FLT}}}{\cos\gamma_{\mathrm{FLT}}} - \frac{\tan\delta_{\mathrm{FLB}}}{\cos\gamma_{\mathrm{FLB}}}\right) \tag{4.10a}$$

$$= y_{\mathrm{FRT}} - y_{\mathrm{FRB}} = d\left(\frac{\tan\delta_{\mathrm{FRT}}}{\cos\gamma_{\mathrm{FRT}}} - \frac{\tan\delta_{\mathrm{FRB}}}{\cos\gamma_{\mathrm{FRB}}}\right) \tag{4.10b}$$

$$= y_{\mathrm{RLT}} - y_{\mathrm{RLB}} = d\left(\frac{\tan\delta_{\mathrm{RLT}}}{\cos\gamma_{\mathrm{RLT}}} - \frac{\tan\delta_{\mathrm{RLB}}}{\cos\gamma_{\mathrm{RLB}}}\right) \tag{4.10c}$$

$$= y_{\mathrm{RRT}} - y_{\mathrm{RRB}} = d\left(\frac{\tan\delta_{\mathrm{RRT}}}{\cos\gamma_{\mathrm{RRT}}} - \frac{\tan\delta_{\mathrm{RRB}}}{\cos\gamma_{\mathrm{RRB}}}\right) \tag{4.10d}$$

$$l = z_{\mathrm{RLT}} - z_{\mathrm{FLT}} = d\left(\frac{\tan\delta_{\mathrm{FLT}}}{\cos\gamma_{\mathrm{FLT}}}\frac{\cos\gamma_{\mathrm{RLT}}}{\tan\delta_{\mathrm{RLT}}} - 1\right) \tag{4.11a}$$

$$= z_{\mathrm{RRT}} - z_{\mathrm{FRT}} = d\left(\frac{\tan\delta_{\mathrm{FRT}}}{\cos\gamma_{\mathrm{FRT}}}\frac{\cos\gamma_{\mathrm{RRT}}}{\tan\delta_{\mathrm{RRT}}} - 1\right) \tag{4.11b}$$

$$= z_{\mathrm{RLB}} - z_{\mathrm{FLB}} = d\left(\frac{\tan\delta_{\mathrm{FLB}}}{\cos\gamma_{\mathrm{FLB}}}\frac{\cos\gamma_{\mathrm{RLB}}}{\tan\delta_{\mathrm{RLB}}} - 1\right) \tag{4.11c}$$

$$= z_{\mathrm{RRB}} - z_{\mathrm{FRB}} = d\left(\frac{\tan\delta_{\mathrm{FRB}}}{\cos\gamma_{\mathrm{FRB}}}\frac{\cos\gamma_{\mathrm{RRB}}}{\tan\delta_{\mathrm{RRB}}} - 1\right) \tag{4.11d}$$

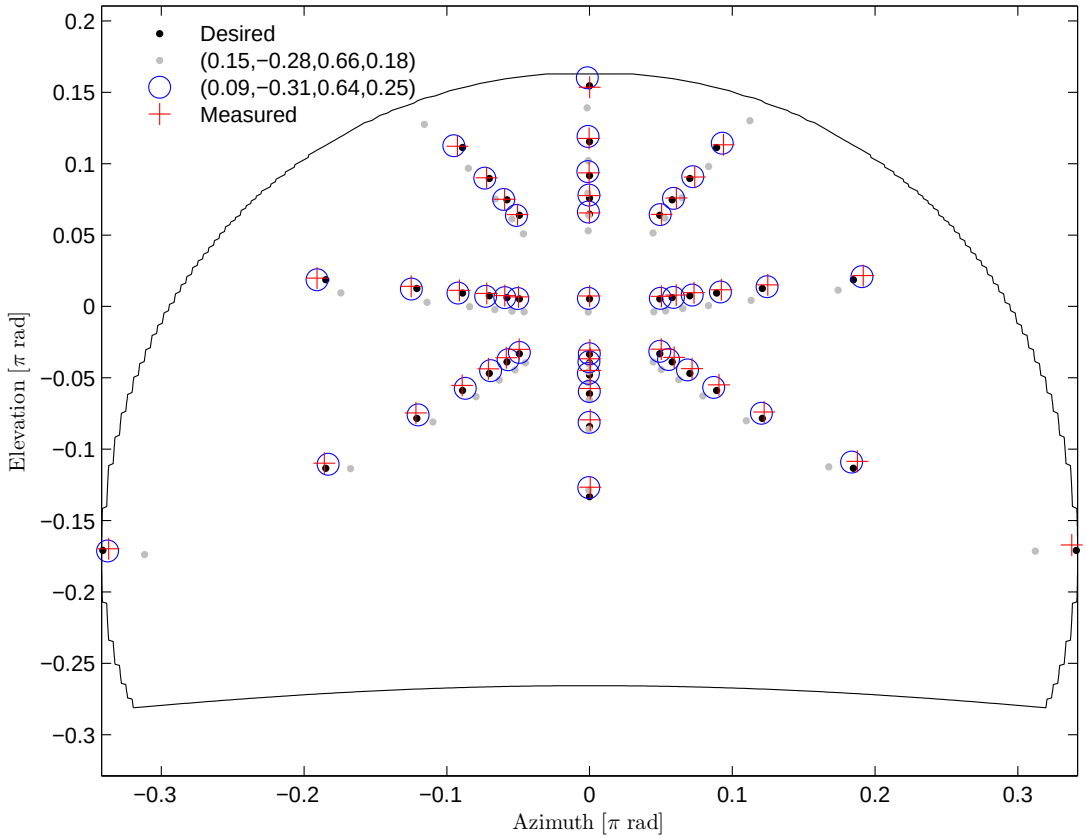


Figure 4.17: Desired observation angles of VE features (based on the Matlab® model) as black dots against their predicted (circles) and measured (crosses) values for optimal choice of sweet spot settings. The values predicted for SPI settings based on geometry alone (without optimisation) are shown in gray.

$$w = x_{\mathrm{FRT}} - x_{\mathrm{FLT}} = d\,(\tan\gamma_{\mathrm{FRT}} - \tan\gamma_{\mathrm{FLT}}) \tag{4.12a}$$
$$= x_{\mathrm{FRB}} - x_{\mathrm{FLB}} = d\,(\tan\gamma_{\mathrm{FRB}} - \tan\gamma_{\mathrm{FLB}}) \tag{4.12b}$$
$$= x_{\mathrm{RRT}} - x_{\mathrm{RLT}} = (d+l)\,(\tan\gamma_{\mathrm{RRT}} - \tan\gamma_{\mathrm{RLT}}) \tag{4.12c}$$
$$= x_{\mathrm{RRB}} - x_{\mathrm{RLB}} = (d+l)\,(\tan\gamma_{\mathrm{RRB}} - \tan\gamma_{\mathrm{RLB}}) \tag{4.12d}$$

Thus, every edge of the box provides an opportunity to measure a dimension, provided that both corners are visible. These measurements are reported in Table 4.1. In most cases the error is less than 3.5 cm, and in the one case where the error is large one of the corners involved in the calculation is almost exactly at the height of the observer, making it very sensitive to measurement error.

Analogically, through simple transformation of the above equations, the distance to the front or rear face of each box can be calculated using the corners of each visible edge if its length is known. These measurements are presented in Table 4.2 and demonstrate accuracy of a few centimetres for both boxes. Again, the top right edge of the further

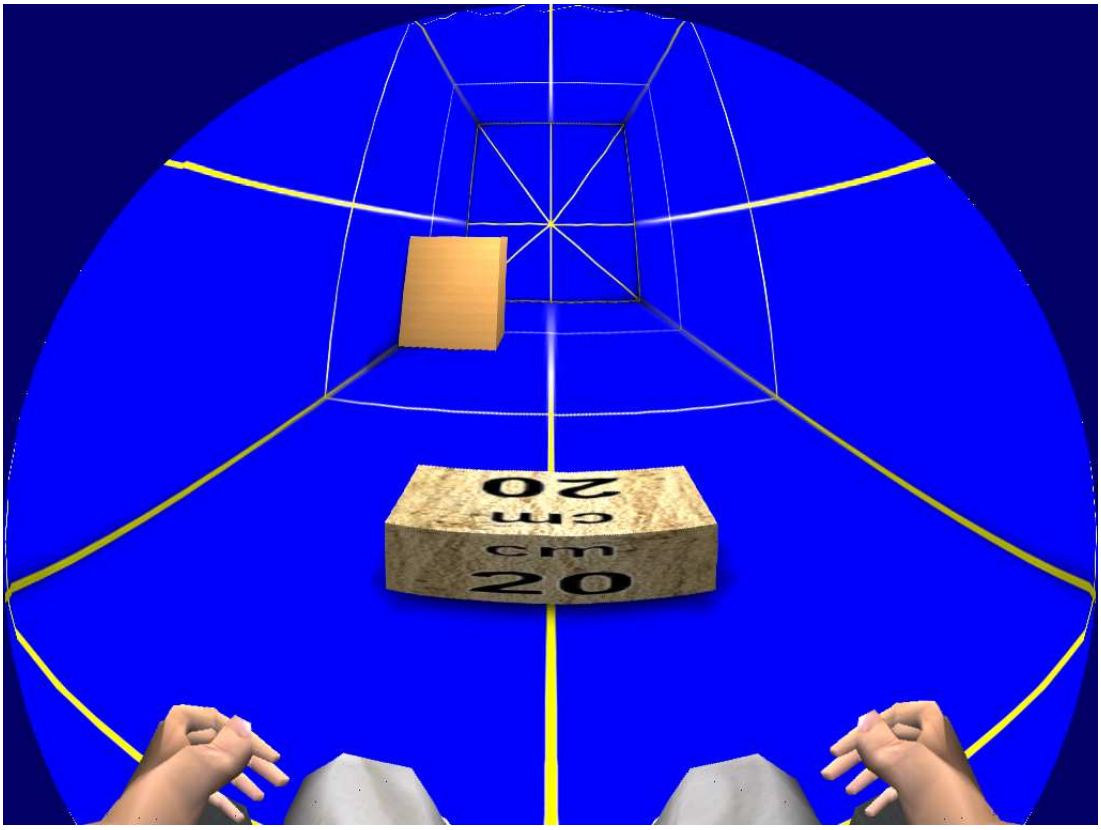


Figure 4.18: The frame used for the study of size and distance perception. The intersection points on the walls correspond to the marker locations in the VE used for SPI settings analysis.

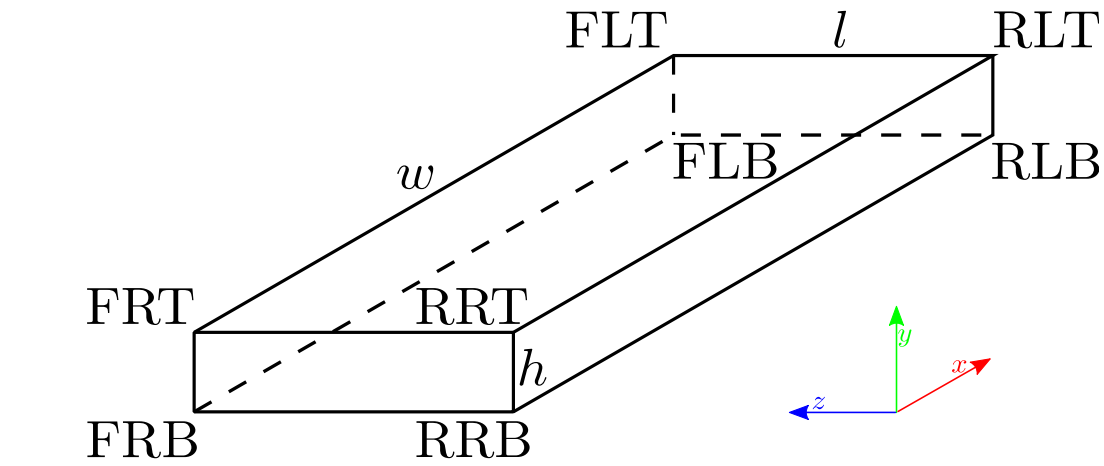


Figure 4.19: Symbols used in analysis of size and distance of objects placed in the VE. E indicates the position of the observer. The corners of the box are coded with three letter identifiers: the first letter is F or R for the front or rear face, the second — L or R for left and right, and third — T or B for the top and bottom.

Table 4.1: Estimated dimensions of objects at known distances.

Dimension	Measure	True Value [m]	Measured Value [m]	Error [m]
Box 1 ($d = 1.874$)				
height h	$y_{\mathrm{FLT}} - y_{\mathrm{FLB}}$	0.200	0.198	0.002
	$y_{\mathrm{FRT}} - y_{\mathrm{FRB}}$		0.198	0.002
length l	$z_{\mathrm{RLT}} - z_{\mathrm{FLT}}$	0.512	0.512	0.000
	$z_{\mathrm{RRT}} - z_{\mathrm{FRT}}$		0.515	0.003
width w	$x_{\mathrm{FRT}} - x_{\mathrm{FLT}}$	1.024	1.055	0.031
	$x_{\mathrm{FRB}} - x_{\mathrm{FLB}}$		1.052	0.028
	$x_{\mathrm{RRT}} - x_{\mathrm{RLT}}$		1.057	0.033
Box 2 ($d = 5.880$)				
height h	$y_{\mathrm{FLT}} - y_{\mathrm{FLB}}$	1.000	0.995	0.005
	$y_{\mathrm{FRT}} - y_{\mathrm{FRB}}$		0.992	0.008
	$y_{\mathrm{RRT}} - y_{\mathrm{RRB}}$		0.851	0.149
length l	$z_{\mathrm{RRB}} - z_{\mathrm{FRB}}$	1.000	0.968	0.032
width w	$x_{\mathrm{FRT}} - x_{\mathrm{FLT}}$	1.000	1.033	0.033
	$x_{\mathrm{FRB}} - x_{\mathrm{FLB}}$		1.026	0.026

Table 4.2: Estimated distance to objects of known dimensions.

Edge	Length [m]	True Distance [m]	Measured Distance [m]	Error [m]
Box 1				
FL*	0.200	1.874	1.896	0.022
FR*			1.893	0.019
*LT	0.512		1.873	0.001
*RT			1.862	0.012
F*T	1.024		1.818	0.056
F*B			1.824	0.050
R*T		2.386	2.314	0.072
Box 2				
FL*	1.000	5.880	5.909	0.029
FR*			5.925	0.045
RR*		6.880	6.910	0.030
*RB		5.880	6.072	0.192
F*T			5.827	0.053
F*B			5.866	0.014

box is clearly the outlier with an error of almost 20 cm.

4.6 Conclusion

This chapter presented a body of work aiming to maximise the immersive effect of the simulator by ensuring the projected geometry is as close to reality as possible. To this end, the equations governing the SPI settings that control the displayed geometry were derived theoretically and verified experimentally. Additionally, in the first[2] attempt to calibrate a wide-angle projector lens without a camera, based on a number of pre-

[2]to the best of the author's knowledge

cise measurements with a theodolite, a model of the lens was developed whereby the position on the dome screen where each pixel of the projected frame will appear can be calculated. This allowed for optimisation of the SPI settings such as to counteract the distortion introduced by the lens and account for the observer and lens positions. Finally, under the optimal settings, the perception of size and distance was found to be accurate to within a few centimetres. In subjective terms, this means subjects perceive the virtual environment to look correct (not distorted) and themselves to be at the "right" spot within it.

Altogether, these developments ensure that the simulator can cater for different subjects' needs in terms of eye level (practically limited by the size of the screen to between 0.5 and about 1.4 m) and clearance under the projector (limited by the range of the horizontal bar to 1 m) without compromising the quality of the simulated wheelchair driving experience. Its high accuracy in terms of producing a correct perception of distance, and therefore also velocity, is a critical factor increasing the simulator's value as a training tool, helping to ensure that skills learnt in VR are readily transferable to real life. This could not be afforded without an immersive display and the optimisation of settings presented here.

Nonetheless, there are some limitations to the quality of the projected environment. As can be seen on Figure 4.17, small deviations between desired and observed positions at the periphery of the dome could not be eliminated. These are possibly not disturbing, as most of the time a user's attention would be in the centre of the screen. Another issue is that of a slight counter-clockwise tilt in the projected image, which is a feature of the projector. The projector mount was designed to keep the projector exactly horizontal and, although it allows for slight deviation from that position, that margin is small and the tilt cannot be quantified or kept constant. Thus, although attempts were made to adjust the position of the projector such as to eliminate the tilt in the image, residual tilt affected the lens model as can be seen in Figure 4.14 and Figure 4.17 where the rightmost mark was predicted to project outside the dome (or not at all) while it was measured on the dome (at the desired location). However, it is unlikely that this small error influences the optimal combination of SPI settings.

4.7 Summary

As set out in section 4.1, the objective of this chapter was to evaluate and maximise the geometrical correctness of scenes displayed by the virtual reality simulator. First, in section 4.2, the equations governing the settings exposed by SPIClops to control the displayed geometry were derived theoretically and validated against a set of 82 images from which 3988 marker points were extracted with a method developed by the author. The agreement between measured and calculated positions of markers was excellent (linear regression coefficient 1.00, $R^2 > 0.998$ for both coordinates). Second, as described in section 4.3, a theodolite was used to measure the positions of markers

projected on the screen. These data served to obtain a mathematical model of the lens through image registration, representing the first attempt to calibrate a wide-angle projector lens without a camera. To ensure correctness of the procedure, two different sets of markers were projected, each from a different lens position. The model resulted in a marked improvement in accuracy of predicted location of projected image features, from 0.1104 to 0.0031 rad, over the perfect lens assumption. Finally, these models allowed developing a procedure, described in section 4.4, to optimise the SPIClops parameters for any viewing position such that the displayed scene appears as correct as possible. When using the optimised settings, simulated distances and sizes are accurate to within 3.5 cm for most areas on the screen, as found in section 4.5. As a result from this work, the simulator can cater for a range of subjects' needs in terms of eye level (between 0.5 and 1.4 m) and clearance under the projector (up to 1 m) providing consistent and geometrically realistic projections of virtual environments. However, as mentioned in section 4.6, perfect geometrical accuracy could not be achieved and objects at the periphery of the dome screen will appear slightly distorted.

Chapter 5

Discrete Command Control for Powered Wheelchairs

5.1 Introduction

Brain computer interfaces (BCI) offer great potential to allow locked-in patients to control external devices, such as spellers or computer cursors. However, most BCI paradigms only afford a small set of discrete commands, and are limited in command frequency. This poses a challenge in terms of operating devices such as powered wheelchairs which conventionally require a control signal continuous both in time and over its domain. One approach to address the issue is to delegate the continuous low level control of a wheelchair to an autonomous control system, leaving the user to choose between a few high level control options. Such shared control systems are used in smart wheelchairs generally (e.g. Bourhis and Agostini, 1998; Levine et al., 1999; Sgouros, 2002; Špacapan et al., 2004), and specifically in the context of BCI (e.g Millán et al., 2004; Rebsamen et al., 2007; Vanacker et al., 2007; Gentiletti et al., 2009). Ultimately, though, we want to provide the patients with the feeling of empowerment and independence that comes with driving the wheelchair rather than being driven by one. Thus, this chapter presents a finite state machine (FSM) controller allowing to operate the wheelchair with flexibility comparable to that afforded by conventional joystick controllers but using only 5 discrete commands.

A number of studies employed an FSM-based approach to control a wheelchair with biological signals such as EMG, EOG or EEG. Most of them use three or four commands and assign each a fixed action (one of "forward", "reverse", "rotate right" and "rotate left") (e.g. Barea et al., 2002; Tsui et al., 2007; Galán et al., 2008). In the field of BCI, Teymourian et al. (2008) compared two controllers: one using only two commands and four fixed actions and another one using four commands and nine fixed actions. They found that when using the latter controller, which allows not only

rotating in place but also travelling in a curve, subjects could follow a smoother path and thus completed a navigation task faster and with fewer collisions. However, the mapping from commands to actions in their study was complex and unintuitive — initiating a pivot turn (rotation in place) from standstill required the wheelchair to move forward first and took in total three commands.

The FSM presented here allows thirty five different actions (motor control outputs), including seven different turn radii (excluding pivot turns), all accessible with five commands, each of which has a unique meaning. A similar controller has recently been used by Hashimoto et al. (2009) to implement EOG and EMG gesture-based control of a wheelchair. While their system also uses five commands, it only allows two different turn radii.

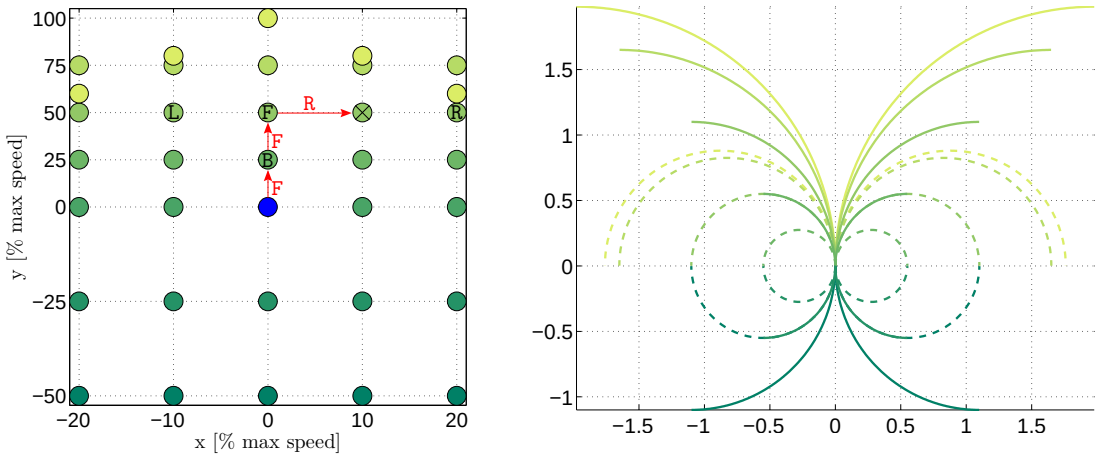
To assess the feasibility of the proposed system to control powered wheelchairs, able-bodied volunteers completed a virtual training track using both a joystick and the proposed FSM-based controller and their performance when using the two modes of control was compared. Additionally, subjects filled in a short background survey and rated their experience. To eliminate the impact of BCI errors, a small keypad was used to provide input to the FSM controller in place of BCI commands. Results show that subjects were able to successfully navigate the course using either input device, although they performed slightly worse using the FSM-based keyboard control.

The remainder of the chapter is organised as follows. First, the methods are presented in section 5.2 including details of the proposed FSM controller, the experimental setup and data analysis. Second, subjects' performance when using the two modes of control is compared in section 5.3, followed by a short analysis of their ratings and a presentation of some common trends observed in the subject population. Closing the chapter is a discussion, in section 5.4, of the results in light of the proposed wheelchair controller's application for BCIs. A short summary of the chapter is offered in section 5.6.

5.2 Methods

5.2.1 FSM controller

A typical wheelchair is a differentially steered vehicle where two of its wheels are powered independently and steering is achieved solely by varying their rotational speed. The wheelchair control input is an ordered pair of voltages applied to the two motors, normally derived from the joystick position (x, y) as $(y + x, y - x)$, where y and x are deflections along the front–back and left–right axes respectively. If $(100, 100)$ represents the maximum forward speed, $(-50, -50)$ is the maximum reverse speed, and $(-20, 20)$ and $(20, -20)$ are the extrema of turning speed, respectively right and left (P Davies, PG Drives Technology 2008, pers. comm., 8 May).



(a) (b)

Figure 5.1: Control states achievable with the proposed controller. (a) Joystick position equivalents: The position marked with a cross can be reached from standstill (blue circle) with three commands – `FORWARD`, `FORWARD`, `RIGHT` (marked `F`, `F`, `R` in red). The positions reachable from there by the commands `LEFT`, `RIGHT`, `FORWARD` and `REVERSE` are marked `L`, `R`, `F` and `R`, respectively. The `STOP` command always brings the wheelchair to standstill (blue circle). (b) Possible turning trajectories of the modelled wheelchair (2.1 seconds of movement, axes in metres). Dashed lines represent trajectories available through repeating the relevant turning command. Both figures use the same colour coding.

The proposed FSM controller allows thirty five different control states in thus defined joystick input space as illustrated on Figure 5.1a using five commands: `LEFT`, `RIGHT`, `FORWARD`, `REVERSE` and `STOP`. The `FORWARD` and `REVERSE` commands increase and decrease the forward velocity (to the point of going in reverse), while the `LEFT` and `RIGHT` commands initiate turning in the respective direction by slowing down one wheel and speeding up the other. Repeating either command will produce a tighter turn. Overall, the proposed controller allows seven different turn trajectories, not including the pivot turn, as depicted on Figure 5.1b where the trajectories available through repeated commands are indicated with dashed lines. The `STOP` command does not work as a break, but instead always brings the wheelchair to a standstill. Thus, each command retains its unique function. For example, if the wheelchair is currently turning to the right, the `LEFT` command will cause it to turn to the left, as opposed to going straight as in some computer games. Hence, the `FORWARD` and `LEFT` commands retain their unique meanings.

The controller can most compactly be represented as a three-state Mealy machine — a type of FSM that produces output on transitions — whose outputs are functions acting on the current control signal (L, R) as depicted on Figure 5.2. Thus, the `FORWARD` command produces in all states the function $(L, R) \mapsto (M + 25, M + 25)$ where M is the mean $(L+R)/2$, while `LEFT` produces $(L, R) \mapsto (L-10, R+10)$ in state q1 (turning

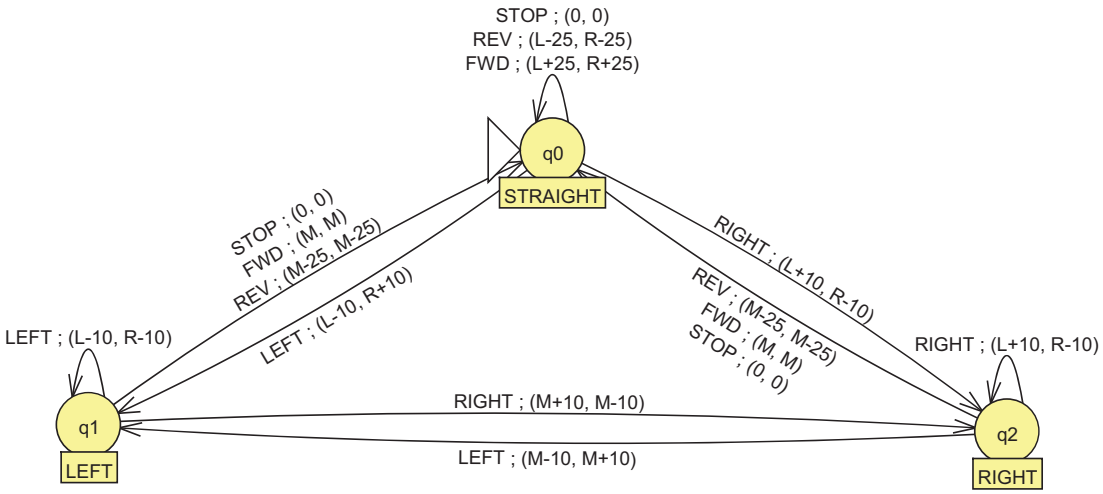


Figure 5.2: The proposed controller represented as a Mealy machine. Arrows indicate transitions between states (q0, q1,q2) and are annotated with the causing action (`LEFT`, `RIGHT`, `FWD`, `REV` or `STOP`) and the resultant output — functions acting on the current control signal (L, R). $M = (L+R)/2$. Not captured by the diagram are the constraints: $\|L - R\| \leq 40$ and $L,\ R \in [-50, 100]$.

left) but $(L, R) \mapsto (M - 10, M + 10)$ in state q2 (turning right), etc. Not captured by this representation are the constraints

$$\|L - R\| \leq 40 \tag{5.1a}$$

$$L,\ R \in [-50, 100]\,. \tag{5.1b}$$

These are implemented in two layers: first that blocks changes to the control signal that would violate Equation 5.1a and $M \in [-50, 100]$ and second that modifies the control signal such that

$$\max(L, R) > 100 \Rightarrow \big((L, R) \mapsto (L - c, R - c)\big) \tag{5.2}$$

where $c = \max(L, R) - 100$, transforming the possible Mealy machine output (110,90) into (100,80). Necessarily, the Mealy machine is oblivious to the impact the second layer has on the ultimate control signal output.

5.2.2 Experimental Procedure

The proposed mode of control was tested by $n = 11$ able-bodied volunteers (six females; age between 23 and 40 — mean 29.27, standard deviation 6.40) using the simulator described in chapter 2 under local ethical approval.[1] The subjects were asked to navigate the training course virtual environment (VE) presented in subsection 2.4.4 such as to 1) avoid collisions, 2) stay in the centre of the track whenever possible and 3) complete the task as fast as possible, in that order of priority, using the Joystick Plus

[1] Departmental Ethics Committee, Project# BAC2010_1

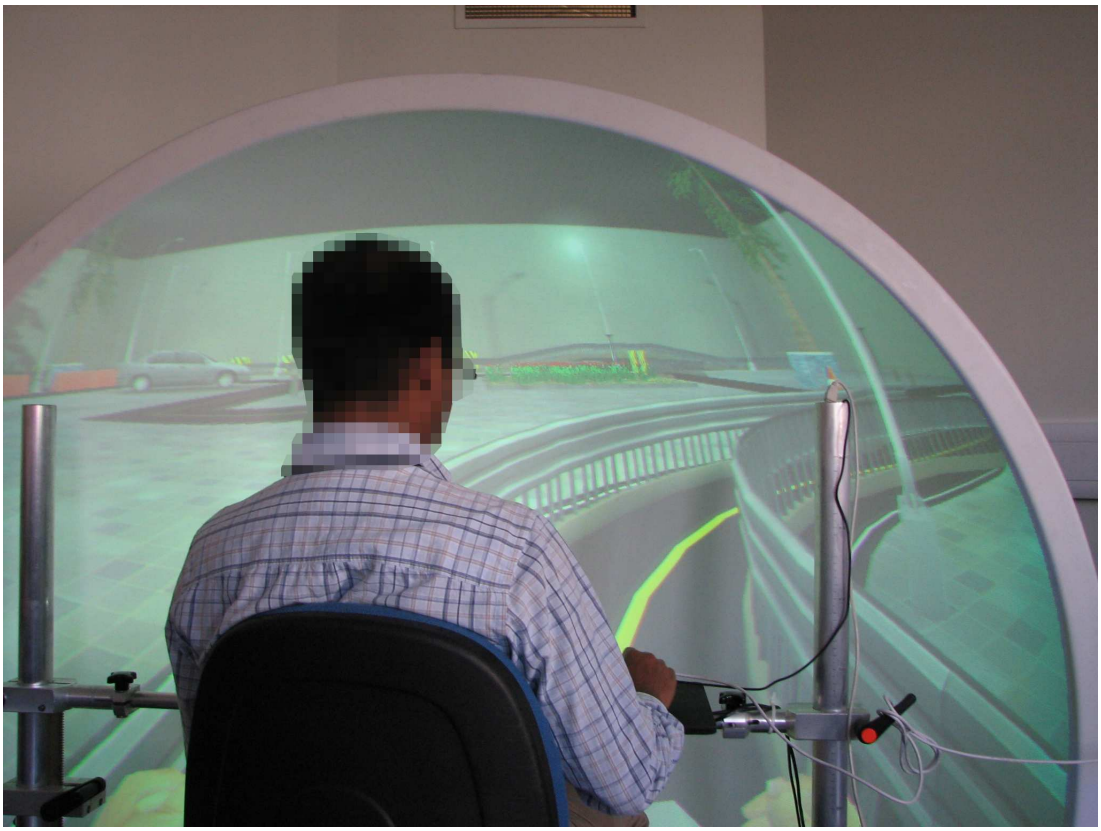


Figure 5.3: A subject navigating a narrow part of the training course using the keypad control.

and a numerical keypad. Five of the keys on the keypad arranged in the shape of a plus served to simulate discrete BCI commands (four arrow keys: 8, 6, 2, 4 for the directions and the central 5 key for the `STOP` command, see Figure 2.1c on p. 17). Six of the subjects used the keypad first. One subject did not complete the experiment due to mild cyber-sickness. Before attempting the training course, subjects were given a virtual fly through the training course and verbal introduction to the mode of control they were about to use and allowed sufficient time to become comfortable with its behavior in the empty hall VE. During the attempts, audio feedback was given to the subjects when the virtual wheelchair came in contact with other objects. In cases when the wheelchair fell over backwards, e.g. when attempting to negotiate the kerb, the trial was resumed close to the incident's position.

During the experiment, data was collected on the wheelchair's position (about every 1 second), velocity and collisions (if any) as well as the control signal generated by the subjects.

After completing the exercise, subjects where asked to fill in a questionnaire including a number of background questions (age, gender, experience with various types of games and daily computer use), the Game Experience Questionnaire (see section 2.6), and an opportunity to rate and comment on the different modes of control.

5.2.3 Data Processing and Analysis

For each subject and mode of control, several parameters were extracted from the records gathered during the experiment. Where completing the course required multiple attempts due to the wheelchair falling, the attempts were stitched together such that no part of the course was traversed twice.

Time to complete, number of collisions, average and maximal velocity (excluding the downhill ramp) as well as the total path length were calculated directly using the wheelchair's status messages (about 1 every second). To calculate the average distance from the centre line, the trajectory of the wheelchair was interpolated using piecewise cubic Hermite polynomials and the area between the trajectory and the centre line was calculated through numerical integration and subsequently divided by the ideal path length. Additionally, continuous control signal was reconstructed from the unevenly spaced record of actual command issue through zero-order hold and subsequently re-sampled at 100 Hz. The power spectral density (PSD) of the left-right (x) component of thus obtained control signal was calculated using Welch's method with a Hamming window and 1024-point FFT. This served to calculate the spectrum-based performance measures Niniss and Inoue (2005) proposed to distinguish skilled and unskilled wheelchair users: R_{bfhf} and I_{x}. R_{bfhf} is the ratio of the sum of low frequency (up to 5 Hz, excluding DC) components to the sum of high frequency components (5–50 Hz) while I_{x} is defined as

$$I_{\mathrm{x}} = N^{-1} \sum_{k=1}^{N} x^2(k) \tag{5.3}$$

where N is the length of the resampled control sequence x (normalised).

The results were explored statistically to identify common trends in performance and compare the proposed discrete mode of control with the joystick. Due to the small sample size (10 excluding the subject who did not complete the experiment), correlation and t-test were the only tools employed. p values up to 0.05 were considered significant.

Aggregate data from all subjects were used to map the locations on the training course where subjects spent the most time, had the most collisions and issued the most direction-changing (left-right) commands.

5.3 Results

All subjects performed the task with reasonable success, although differences in strategy and performance could be observed between them. However, the subjects' performance was correlated with neither their experience with different types of games (arcade, first-person shooter, simulators), nor the mode of game control (joystick or keyboard) or hours a day spent using the computer (once one outlier was excluded). However, age was strongly correlated with several outcome variables, see Table 5.1. Time taken to complete the task and average distance from the middle of the track increase with age

Table 5.1: Correlations between age (in years) and performance measures.

	Keypad	Joystick
Time [s]	0.7376*	0.8587**
Max speed [m/s]	−0.3618	0.0124
Avg. speed [m/s]	−0.7365*	−0.8302**
Path length [m]	0.9211**	−0.0831
Avg. dist. [m]	0.8316**	0.6523*
Collisions	0.5819	0.2735
I_x	−0.4167	−0.2022
R_{bfhf}	0.2678	−0.6098

* $p \leq 0.05$; ** $p \leq 0.01$

Table 5.2: Gender differences in performance measures assessed with two-sample t-test.

	Joystick			Keypad		
	male	female	diff.	male	female	diff.
Time [s]	590.1045	422.8428	167.2618*	534.8724	412.3571	122.5143
Max speed [m/s]	1.6769	1.7476	−0.0707	1.5486	1.7762	−0.2276*
Avg. speed [m/s]	0.6589	0.8804	−0.2215*	0.6826	0.9462	−0.2636*
Path length [m]	364.67	361.15	3.52	354.32	360.75	−6.43
Avg. dist. [m]	0.3501	0.3539	−0.0038	0.2657	0.3019	−0.0363
Collisions	25.00	34.50	−9.50	5.50	19.50	−14.00*
I_x	0.1048	0.1687	−0.6380**	0.0722	0.1552	−0.0830**
R_{bfhf}	39.1588	39.4500	−0.2920	35.0584	38.4755	−3.4172

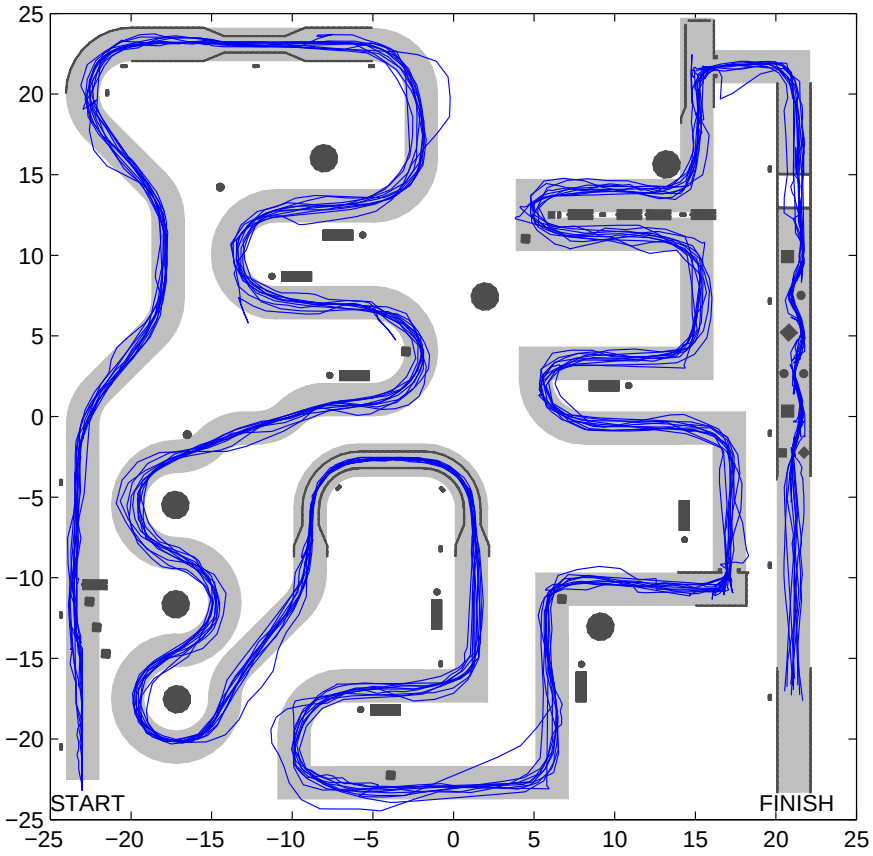
* $p \leq 0.05$; ** $p \leq 0.01$ (one sided)

while average speed falls, for both modes of control. Additionally, for keypad only, age was positively correlated with total path length.

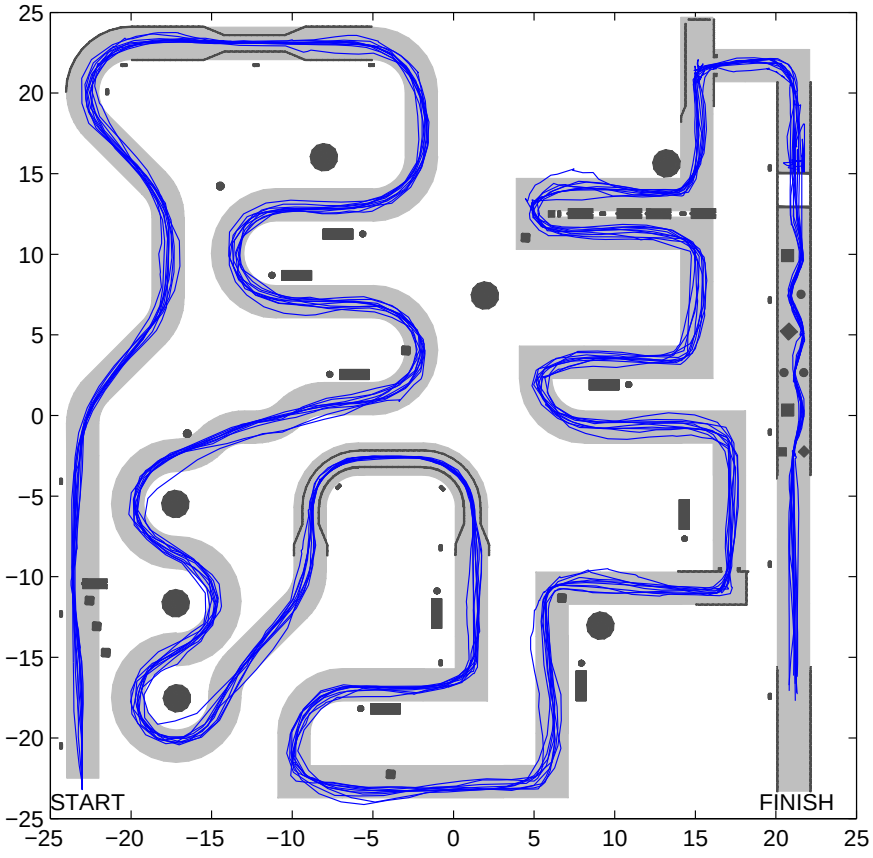
Using two-sample t-tests, significant differences were also found between genders (Table 5.2), with women achieving higher average and maximum speeds and thus completing the task much faster. However, they collided more than men and used more and/or more extreme turning commands as reflected by their higher score on the I_x measure.

Comparing the subjects' performance when using the two modes of control revealed almost no statistically significant differences (Table 5.3), other than in the number of collisions which was more than twice as high when using the keypad. On average, however, when using the joystick subjects completed the course faster and drove closer to the centre of the track as is clear from Figure 5.4, although these differences were not significant.

Pooling together data from both modes of control, comparison between the first and second round revealed some surprising patterns (Table 5.3). While subjects generally drove faster the second time round, they had more collisions (although the effect was not significant) and their total path length increased with no significant change in distance from the centre line. This was associated with a smoother control signal as



(a) Keypad



(b) Joystick

Figure 5.4: Paths followed by the subjects using the different modes of control.

Table 5.3: Mean differences in performance measures between the two modes of control and the first and second attempts assessed with two-sample t-test.

	Control mode			Round		
	Joystick	Keypad	diff.	1st	2nd	diff.
Time [s]	461.3628	489.7475	−28.3847	478.2560	472.8543	5.4017
Max speed [m/s]	1.6852	1.7193	−0.0341	1.6686	1.7358	−0.0671
Avg. speed [m/s]	0.8408	0.7918	0.0490	0.7810	0.8516	−0.0706
Path length [m]	358.18	362.56	−4.38	356.47	364.27	−7.80*
Avg. dist [m]	0.2874	0.3524	−0.0649	0.3106	0.3292	−0.0186
Collisions	13.90	30.70	−16.80**	20.60	24.00	−3.40
I_x	0.1220	0.1431	−0.0210	0.1266	0.1385	−0.0118
R_{bfhf}	37.1087	39.3340	−2.2253	32.2213	44.3831	−12.3200**

* $p \leq 0.05$; ** $p \leq 0.01$ (one sided)

manifested by the higher R_{bfhf} value.

The range of responses to questionnaire items about the two modes of control are presented in Figure 5.5. Although the joystick was perceived as more intuitive, quicker to learn and easier to use (all significant at $p \leq 0.05$ level, one sided two-sample t-test), the median responses for the keypad were not negative. The remaining questionnaire items did not show significant differences. In the open-ended questions, subjects commented mainly on the keypad mode of control stating that turning is too fast at low speeds (2 comments), but "perfect" at high speeds (1 comment), and that the reverse key is hard too reach (1 comment). They also suggested that the non-functional keys be removed from the keypad (1 comment) and that the current speed level be indicated on the screen (1 comment).

Maps of time expenditure in different locations of the track (Figure 5.6) show a predictable pattern — most time was spent in the obstacle course, the kerb and the two doors. The maps show no obvious differences between the two modes of control, other than more time spent passing the kerb when using the joystick. Maps of collisions (Figure 5.7) for both modes of control show very few collisions with free-standing obstacles, but a large number with barriers in narrow passages.

The parameters proposed by Niniss and Inoue (2005) — the amplitude measure I_x and the frequency measure R_{bfhf} — did not produce the expected clustering of subjects into skilled and unskilled. Indeed, none of the subjects produced in the PSD of their left-right control signal a peak at 13–14 Hz as described by these authors for unskilled subjects. However, some qualitative differences in the way subjects used the controls emerged (see Figure 5.8), where subjects with poorer performance (in terms of time to complete) did more small radius or pivotal turns and stopped more often while subjects who achieved high performance drove faster and performed milder turns. This usage pattern seemed consistent between the two modes of control.

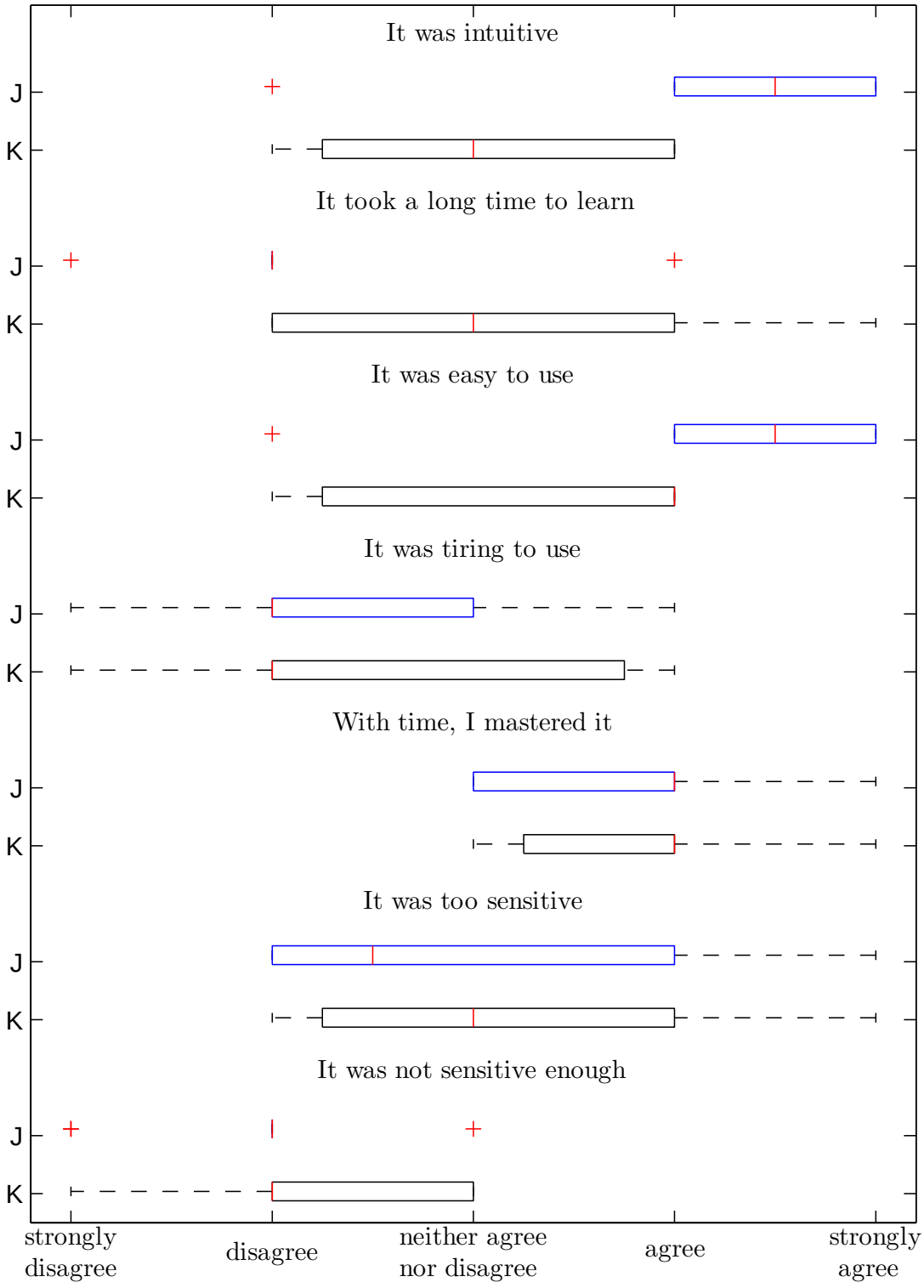
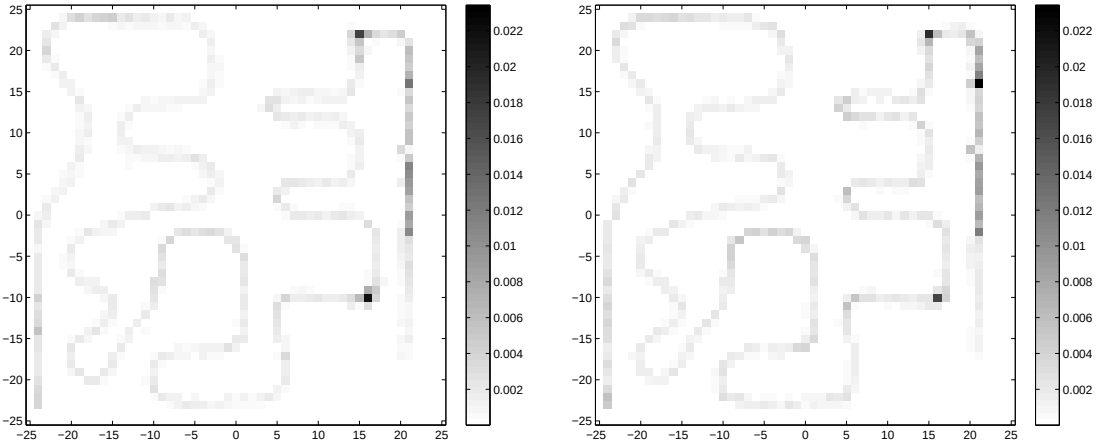
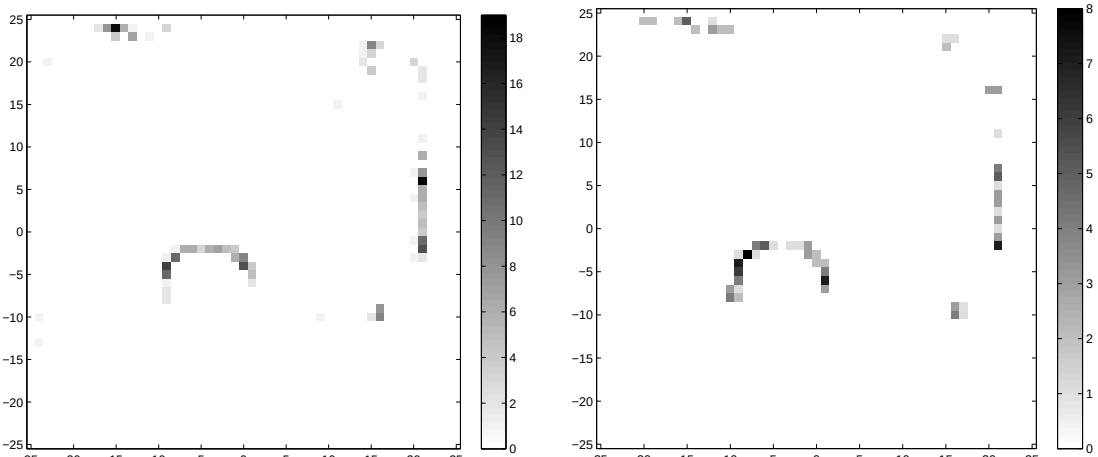


Figure 5.5: Responses related to mode of control (K — Keypad, J — Joystick).



(a) (b)

Figure 5.6: Average fraction of time spent at different locations of the training course when using (a) the keypad and (b) the joystick mode of control. Axes in metres.



(a) (b)

Figure 5.7: Locations of collisions when using (a) the keypad and (b) the joystick mode of control. Aggregate data from all subjects. Axes in metres.

5.4 Discussion

The results presented in the previous section are encouraging in that few significant differences were found in virtual wheelchair navigation task between the proposed FSM-based controller and conventional joystick. A number of interesting trends emerged relating the subjects' performance to their age and gender. However, a larger study would be required to confirm these findings. In particular, larger n would allow assessing through appropriate regression analysis the separate effects of age and gender. Moreover, assessment of the proposed controller by experienced wheelchair users will be necessary in the future, preferably on a real rather than a virtual wheelchair. At the

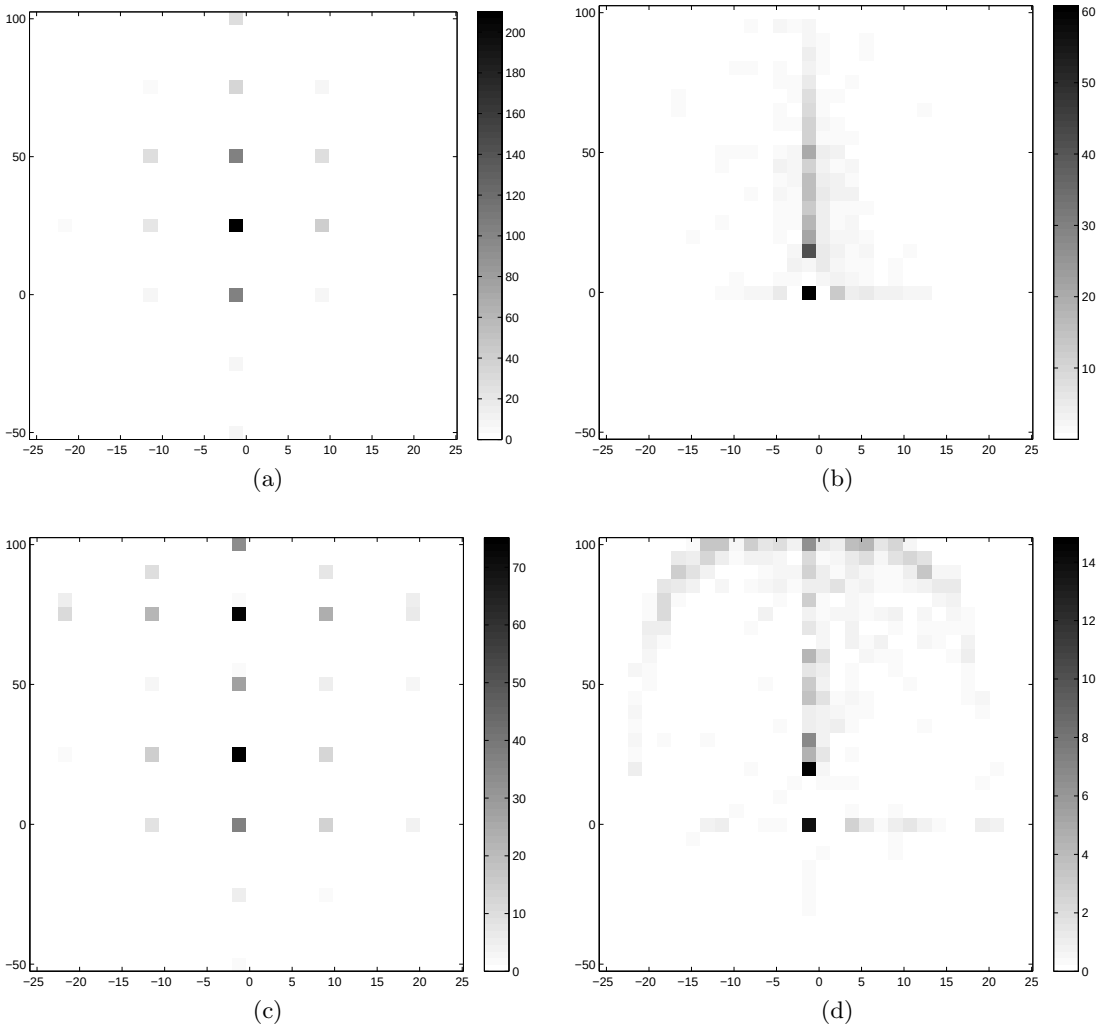


Figure 5.8: Typical control use patterns for poor (top) and high (bottom) performing subjects using the keypad (left) and the joystick (right) mode of control. Data from two individuals. The color scale indicates time in seconds driven with the particular control setting.

present stage of development, such assessment would be neither practical nor ethical.

The one statistically significant difference between the joystick and keypad mode of control is the much higher number of collisions when using the keypad. One factor that certainly contributed to that difference is the difficulty some subjects had with the keypad itself (hitting wrong or non-functional keys), independent of the FSM-based controller. This will not affect the use of the proposed controller in BCI-controlled wheelchairs. Another reason is perhaps related to the high steering sensitivity at low speeds pointed out by the subjects which made it difficult to operate the wheelchair in narrow passages. However, the subjects were allowed to drive at full speed using both modes of control while, for practical and safety reasons, any BCI-controlled wheelchair will need to drive a lot slower. For instance, the EOG/EMG controlled wheelchair of

Hashimoto et al. (2009) develops a maximum speed of 0.72 km/h, compared with just over 6 km/h in the experiment presented here. It is likely that the steering sensitivity problem will disappear when the control signal is scaled.

An interesting finding is the improvement in speed but deterioration of precision between the first and second attempt, independent of the mode of control. It would appear that subjects gained confidence after the first completion of the training course and drove in perhaps a more enjoyable way the second time round, committing more mistakes but completing the task faster. This sense of security afforded by virtual reality may be detrimental to the learning process when the simulator is used for training prospective wheelchair users. To remedy, some penalty for collisions could be incorporated into the simulator, e.g. a simple indicator of the total number of collisions thus far or a louder crash sound.

There were large differences in subjects' performance that were not explained by their experience with games, simulators or control devices. The qualitative differences found in control device usage patterns support the general idea proposed by Niniss and Inoue (2005) to use this information to evaluate wheelchair driving ability. However, their original measures performed poorly in the current study and further research into robust indicators is required.

Overall, the lack of significant differences in terms of completion time, total path length and distance from the centre line between the two modes of control as well as the neutral-to-positive user response to the proposed controller are encouraging, especially considering the problems some subjects had with the keypad. The full speed used in the tests and the excessively difficult obstacle course (completed successfully by all subjects) provides additional confidence in the suitability of the FSM-based controller for wheelchair navigation using discrete commands.

5.4.1 Use with BCI

For the specific application of brain-computer interfaces, further evaluation is necessary before tests with real wheelchairs can commence. So far, no explicit requirements regarding the user input have been made, implicitly assuming it to be error-free. However, BCIs are characterised by a considerable error rate (up to 30%), limited bandwidth (frequency of commands) and, depending on design, a fixed or variable delay between user's intention and the system's recognition of it. The interplay between these variables, together with the wheelchair's top speed, determines the ability of the proposed controller to allow the user to safely reach their destination. Higher error rates can be tolerated if erroneous commands can be promptly corrected (high command frequency). Alternatively, effective error rate can be decreased by giving the user a chance to confirm a command before execution, but this would increase the delay and force the top speed down.

While the simulator described in this thesis allows for safe tests of BCI wheelchair

control, the overall performance will always be a product of the user's abilities, the BCI performance and the characteristics of the controller. A separate study would be required to evaluate the input requirements of the controller alone, i.e. the maximum error rate and delay at which the controller is usable. One way in which this could be achieved is by using a simulator of BCI input characteristic like that recently presented by Quek et al. (2011), where the user keyboard-generated commands are modified in accordance to a prescribed error-rate and delay. Such a simulator could be used in conjunction with the keypad mode of control in experiments similar to those presented in this chapter. More effectively, a synthetic control signal could be generated by an objective-seeking algorithm, a sort of auto-pilot for wheelchairs, fully aware of the virtual environment and the wheelchair's position and velocity within it. This would allow not only to run many more trails then would be possible with users, but also to investigate the user's reaction time among the variables determining the usability of the proposed controller.

5.5 Conclusion and Future Work

Presented here is a method of power wheelchair control using a small set of commands as can be derived from physiological signals that affords comparable performance to the conventional joystick. Like other approaches in the literature, the proposed method is based on a finite state machine (FSM). Yet it distinguishes itself with its simplicity (5 commands with intuitive command to action mapping) and unprecedented[2] flexibility in terms of the number of possible control outputs (trajectories). As manifested by the successful completion of a challenging virtual training course by first-time users, the proposed system is intuitive and can quickly be explained and learnt. This represents an important milestone towards building a BCI-controlled wheelchair at the University of Strathclyde.

The proposed system relies on the availability of well timed commands and will thus require an asynchronous (i.e. not cued) BCI system, or one with high potential command frequency. Once that is developed, future work on the controller should involve thorough testing using the BCI input instead of the keypad. This will require scaling down the control signal, thus slowing down the wheelchair, and accounting for the possibility of erroneous (mis-classified) commands. The maximum BCI error rate and delay beyond which the controller is inappropriate should be found through simulation. Additionally, a visual indicator of the current control state should be developed, as suggested by one of the subjects.

[2]to the best of the author's knowledge

5.6 Summary

Described in this chapter is a novel wheelchair controller algorithm to act as an interface between BCI-derived commands and wheelchair motor control. Unlike similar approaches based on finite state machines, the proposed controller allows the user to steer the wheelchair along smooth paths without conceding control to a semi-autonomous system. Using only five discrete commands, the controller allows seven different turn trajectories (and pivot turns), while maintaining unique and intuitive meaning of each command. The controller was tested by eleven able-bodied subjects using a keypad to emulate BCI commands who navigated a virtual obstacle course. Their performance in terms of time to complete, number of collisions, distance from the desired path as well as maximum and average speed was contrasted with that achieved when using a joystick instead of the proposed controller.

Statistical analysis revealed few significant differences between the two modes of control, other than a higher number of collisions when using the keypad which can be attributed, at least partially, to limitations of the keypad itself and the fact that wheelchair speed was not limited as it would be for BCI experiments. These results suggest the proposed controller is suitable for wheelchair navigation, a conclusion further enforced by feedback from users, most of whom felt they were able to master it over the course of the experiment (about 10 min). Additionally, intriguing relations were found between the subjects' performance and their age and gender with women, for example, completing the task faster but committing more collisions. However, a larger study would be required to confirm these findings.

The controller is an important milestone towards the development of a BCI-controlled wheelchair at the University of Strathclyde. However, further work on asynchronous BCI is required before such system can be realised.

Chapter 6

Application: A Study of Learning

6.1 Introduction

After the first success of integrating existing work on wrist movement intention classification from EEG and the virtual reality powered wheelchair simulator into the Strathclyde Brain Computer Interface (S-BCI), reported in Valsan et al. (2009), the focus of this chapter is on the effect real-time provision of performance feedback in virtual reality has on a subject's classification rates. In particular, the study seeks to determine whether learning occurs while feedback is being provided, whether that feedback is necessary to sustain any improvements, and whether motivation alone could explain them.

Previous studies (Marr, 1969; Kawato, 1990; Wolpert et al., 1995) have shown that motor skill learning in humans is error-driven, whereby the next repetition of a motor task is corrected by a fraction of the error of the previous one. It has also been shown that artificially increasing the error (e.g. through visual error augmentation) can improve the speed and amount of learning (Patton and Mussa-Ivaldi, 2004; Wei et al., 2005; Grafton et al., 2008). However, results of a recent study with a BCI (Barbero and Grosse-Wentrup, 2010) suggest that while feedback distortion is beneficial for poor performing subjects, it may be detrimental to the performance of the more capable ones. The authors hypothesise that the improvement is mediated through increased motivation. A number of studies utilising virtual reality for BCI feedback provision have also identified motivation as a major factor improving performance (Friedmann et al., 2004; Leeb, Lee, Keinrath, Scherer, Bischof and Pfurtscheller, 2007; Ron-Angevin and Daz-Estrella, 2009), noting higher performance in motivational virtual environment than in experiments using non-immersive feedback.

The study provides experimental evidence that subject learning occurs while real-time performance feedback is provided with the S-BCI. This is supported by both an improvement in successful classification rates and the existence of an error potential

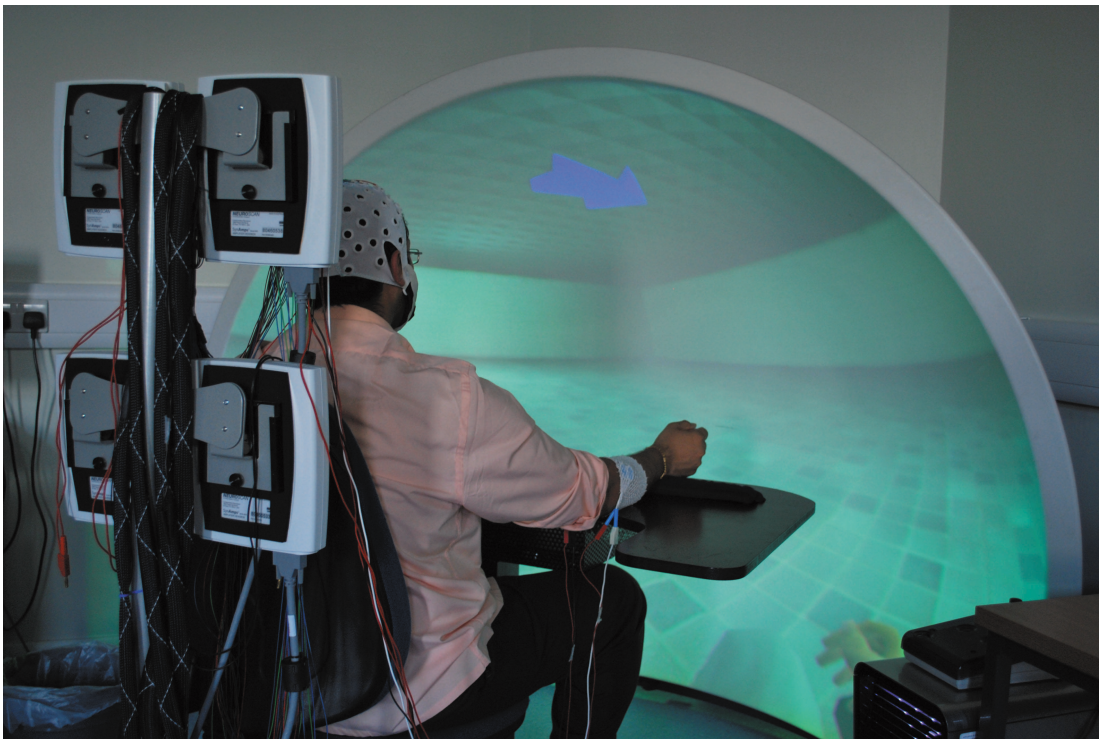


Figure 6.1: The experimental setup. The subject is seated in front of the immersive VR system providing a first-person view from the perspective of a person sitting in the virtual wheelchair. ©2010 IEEE

detectable in the frontal electrodes. The positive impact of motivation on classification rates is also confirmed.

The remainder of the chapter is organised as follows. Details of the experimental procedure and the acquired data as well as the algorithms used to analyse them are described in section 6.2. The results in terms of the subjects' classification rate and reaction time under different feedback conditions as well as a typical recording of an error potential in one subject are provided in section 6.3. The results are discussed in light of previous findings and directions for future work are offered in section 6.4. Closing the chapter is a short summary in section 6.5.

Figures and tables in this chapter are reprinted, with permission, from: Grychtol, B., Lakany, H., Valsan, G. and Conway, B. A. (2010). Human behavior integration improves classification rates in real-time BCI, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* **18**(4):362–368. ©2010 IEEE.

6.2 Methods

6.2.1 Experimental Procedure

The study was conducted at the Neurophysiology Lab at the Department of Bioengineering, University of Strathclyde under local ethical approval. Eight healthy volun-

teers ($n = 8$) participated in the study, six of whom have never used a BCI system before.

The experiments were conducted using the Strathclyde BCI (S-BCI) system (Valsan et al., 2009), consisting of the virtual reality wheelchair simulator described in preceding chapters, an EEG/EMG signal acquisition system and a dedicated BCI workstation where online classification was performed as described in subsection 6.2.3. The subjects were seated in front of the dome screen, as depicted on Figure 6.1, where a first person view from the perspective of a person driving a wheelchair across the diagonal of a large hall was displayed. The subjects were asked to flex (palmar flexion) or extend (dorsiflexion) their right wrist according to the direction indicated by the cue arrow at the top of the screen. The arrow which initially pointed forward deflected either to the left or the right (randomly, with equal probability) every 9 seconds and remained in this position for 6 seconds. During the last second of the arrow's deflection, feedback was provided to the subjects by means of temporarily steering the otherwise forward travelling wheelchair to either the left or the right. The 9 second trials were conducted in runs of between 15 and 37 (mean 29.25), depending on how fast the wheelchair would reach a wall.

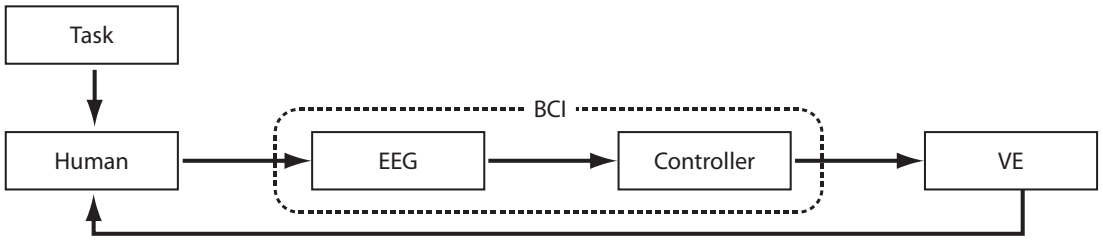
Following the collection of about 60 trials (usually 2 runs) of data and training the classifier, the experiment was divided into three sessions, about 120 trails each, differing in the type of feedback provided to the subject:

Closed-loop feedback In the first session, the virtual wheelchair was steered according to the result of motion intention classification from EEG signal, correct or otherwise. Thus, the subjects received accurate feedback of their performance and were in closed loop with the BCI system (Figure 6.2a). This session served to investigate any learning effects.

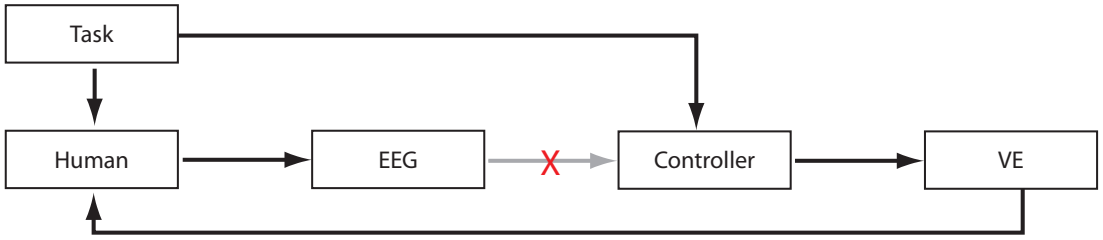
Open-loop feedback In the following session, the wheelchair was steered according to the direction of the cue and was in no way related to the subjects' actions or the classification results which were stored for later processing. Thus, the subjects perceived a 100% correct performance and the feedback loop was open (Figure 6.2b).

Artificial errors feedback In the last session, random 20% of the trials were presented as incorrect i.e. the wheelchair was steered in the direction opposite of the cue. The subjects still had no control over the wheelchair (Figure 6.2c), but now perceived a diminished performance. This session was designed to match the subjects' motivation level from the first session without providing any information content.

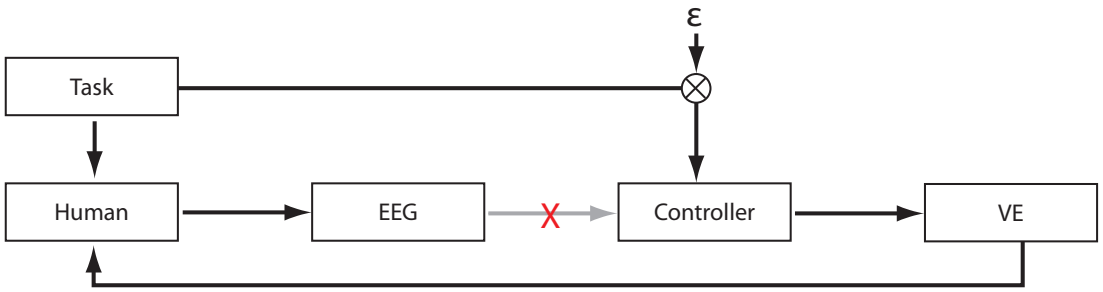
Only one subject (EH) was aware of the nature of the experiment, and thus did not participate in the last session as no changes would be expected.



(a) Session 1: Human-in-the-loop setup. Subject receives accurate feedback.



(b) Session 2: The loop is broken. Subject has no influence on the controller.



(c) Session 3: Introduction of error. Subject perceives diminished performance.

Figure 6.2: Schematic representation of the experiment. ©2010 IEEE

The three sessions were always conducted in the order indicated above. It was feared that if the true feedback session was not the first, subjects would be demotivated by the lower performance in this session, while the open-loop session was required to separate the other two to avoid a spill-over effect that could obscure some patterns in the data.

6.2.2 Data Acquisition

EEG signals were acquired using twenty three Ag/AgCl sintered ring electrodes placed over the sensorimotor and planning areas of the left (contra-lateral) cortex as indicated on Figure 6.3. Electrodes AFz and Cz were used as the ground and the reference, respectively. Two pairs of unipolar surface gel electrodes (Blue Sensor N, Ambu, Denmark) were placed on the right forearm such as to capture the EMG activity of the Flexor Carpi Radialis and the Extensor Carpii Radialis Longus muscles responsible, respectively, for the flexion and extension movements. The EMG signal was recorded to enable task performance analysis, but was not used for classification. The EEG and EMG signals were simultaneously acquired with the Synamps2 data acquisition system (Compumedics Germany GmbH) at a sampling frequency of 2 kHz in 9 second blocks

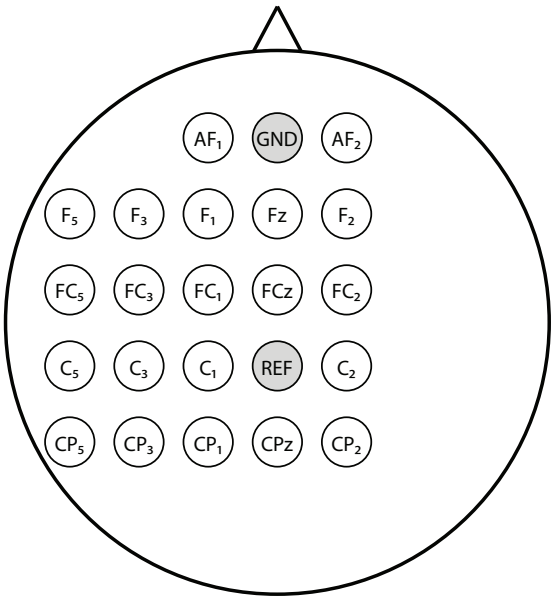
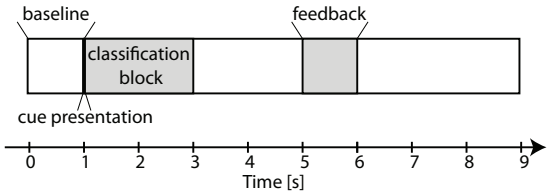


Figure 6.3: EEG electrode montage. ©2010 IEEE



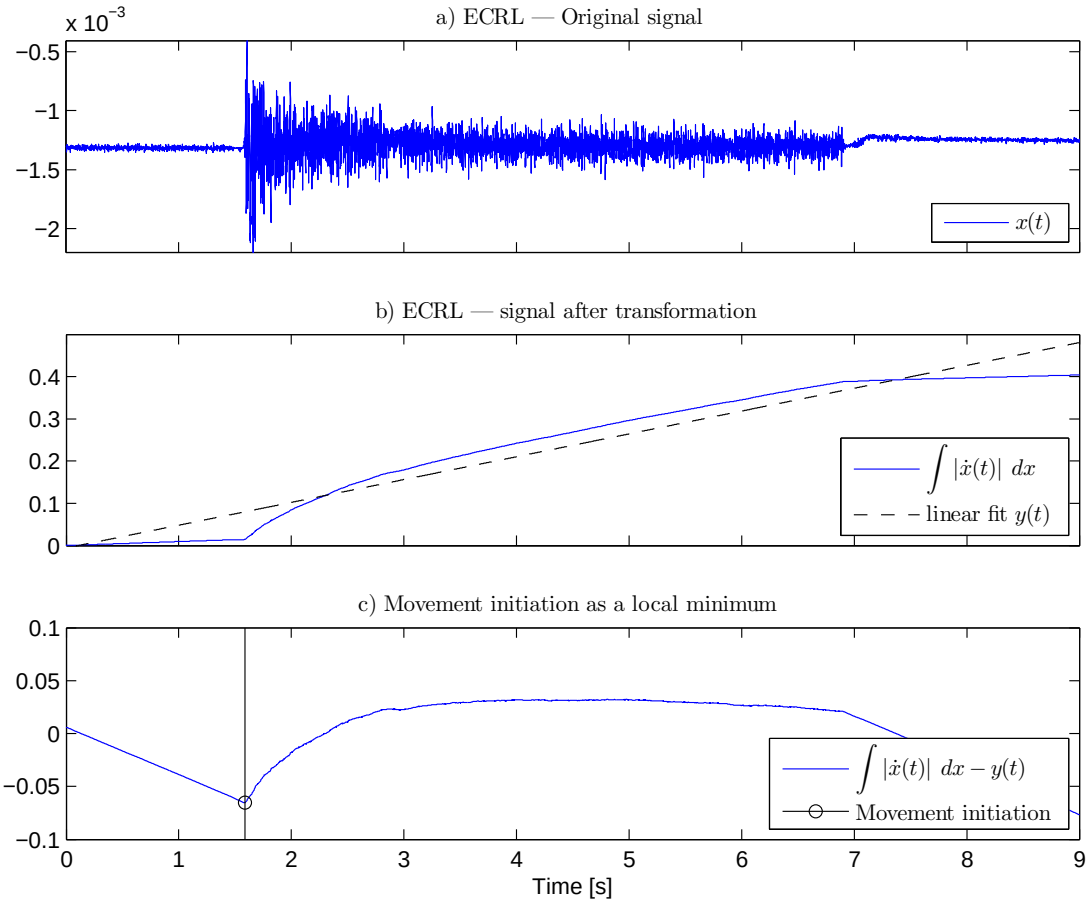
©2010 IEEE

Figure 6.4: Timeline of a single trial. 1 second corresponds to 2000 samples.

(18000 samples), as depicted on Figure 6.4.

6.2.3 Classification

The algorithm for detection of movement direction from EEG is described in full detail in earlier work (Valsan et al., 2009; Grychtol et al., 2010). Shortly, the classification is based on time-domain features extracted through principal component analysis of 2 seconds of data from a single EEG channel immediately following cue presentation. Mean feature vectors for the two directions of wrist movement are computed from the training data. New trials are assigned to the closer of the two classes as determined by weighted Euclidean distance, where the eigenvalues obtained from the training set are used as weights. The channel used for classification was selected manually based on performance on the training data. Where two channels achieved similarly high true positive rate, the one with more even performance between the two classes was preferred.



©2010 IEEE

Figure 6.5: Movement initiation detection algorithm: a) the original EMG recording $x(t)$ from Extensor Carpi Radialis Longus; b) the integral of the first derivative of $x(t)$ and the best-fit trendline $y(t)$; c) the moment of movement initiation is found as the local minimum after subtracting $y(t)$ from the transformed signal.

6.2.4 Data Processing and Analysis

Performance Measures

The subjects' performance in terms of correctly classified trials was analysed with a view to capturing differences between sessions and changes occurring during them, if any. Mean success rates in the three sessions were compared pair-wise using a two-sample t-test, as were the first and last ten trials in each direction (twenty in total) of each session. The two directions were analysed separately to remove any bias resulting from the (random) order of cue presentation. For each direction, the true positive rate can be interpreted as sensitivity of the classifier for that direction and specificity for the other.

Response time

The standard deviation of the subjects' response time was calculated as a measure of consistency in performing the task. The response time is defined as the interval between cue presentation and movement initiation. The timing of cue presentation was recorded by the Synamps2 device in an additional channel connected to the audio output of the wheelchair simulator which emitted a spike every time the cue arrow rotated. No actual sound was produced.

Movement initiation was detected through analysis of the recorded EMG activity of the subject's forearm muscles which increases drastically during contraction (c.f. Figure 6.5a). This increase was detected by 1) numerically integrating the absolute value of the first derivative of the signal (Figure 6.5b); 2) subtracting the best linear approximation; and 3) finding the minimum of the resulting function (Figure 6.5c). This process was carried out for both muscles and the moment of movement initiation was recognised as the smaller of the two results. Although detection of movement initiation in EMG is an active area of research (see Morey-Klapsing et al. (2004) for a review or Vaisman et al. (2010) for a recent example), due to well specified signal characteristics and low signal to noise ratio, the novel method presented here was found to be robust and sufficiently accurate.

Error Potentials

Error-related changes in EEG activity are well described in the context of motor control (e.g. Gehring et al., 1995; Falkenstein et al., 2000). Recently, several studies (e.g. Schalk et al., 2000; Ferrez and del R. Millán, 2008) demonstrated the presence of error potentials also in the context of BCI, where the error is committed by the interface and not the subject. Detecting error potentials in real time could help improve the accuracy and speed of BCI systems.

In this study, the EEG signal from the fronto-central electrodes AF_1 and AF_2 (c.f. Figure 6.3) from correct and erroneous trials during the first session of the experiment were compared. The signal acquired during each trial was smoothened with a third order Savitzky-Golay filter using a window of 101 sample points. Ensemble averages for correct and erroneous trials were computed based on a three second long segment of data immediately following feedback presentation and subsequently subtracted to identify the error potential. Statistical significance of the difference was evaluated on a point-by-point basis with a two-sample t-test.

Table 6.1: Average classification rates for each session in %. ©2010 IEEE

Subject	Class	Session 1	Session 2	Session 3
BZ	L	53.61	46.81	62.75
	R	27.78	48.48	37.50
EH	L	48.21	42.37	N/A
	R	52.17	58.33	N/A
EO	L	49.06	57.81	64.06
	R	47.14	37.70	27.27
GN	L	100.00	100.00	95.38
	R	85.25	98.31	100.00
RB	L	65.62	53.85	45.31
	R	51.32	48.15	63.79
ON	L	68.33	47.54	98.33
	R	79.69	69.33	96.83
HN	L	79.31	76.79	79.66
	R	68.42	61.76	50.75
UI	L	85.07	98.11	100.00
	R	54.55	92.86	100.00
Mean		63.47	64.89	72.97

6.3 Results

Classification Rate

Average classification results for all three sessions are reported in Table 6.1. Overall, the performance increased from session to session, although no difference between two sessions was significant at 0.05 level (difference between session 1 and session 3 reached $p = 0.077$ in single-tailed paired-samples t-test). In general, the classifier performed similarly for the two directions (with the notable exception of subjects BZ and EO).

Comparing the beginning and end of each session, significant difference was found only in session 1 (closed loop feedback) — a 20% improvement ($p = 0.002$, single-tailed paired samples t-test for both directions together). This was followed by an 8.75% decrease during session 2 (open loop) which narrowly escaped statistical significance ($p = 0.055$). However, the mean performance throughout session 2 was almost 7% lower than at the end of session 1 ($p = 0.017$). The introduction of artificial errors resulted in a slight improvement in performance over the duration of session 3 (2.86%), but the difference was not significant.

This pattern is best exemplified by the performance of subject RB whose performance in each trial is documented in Figure 6.6 as a cumulative sum: every correct classification is assigned a reward of 1 while erroneous classifications incur a penalty of -1. Thus, the gradient of the line reflects performance: a gradient of 1 signifies 100% correct classification, a gradient of 0 — 50% and a gradient of -1 — 0%. RB's performance improved during session 1 from about chance level at the beginning to about

Table 6.2: Classification results at the beginning and end of each session.
©2010 IEEE

		Session 1		Session 2		Session 3	
Subject	Class	Beg.	End	Beg.	End	Beg.	End
BZ	L	20	70	10	50	100	60
	R	50	40	50	30	30	20
EH	L	30	50	50	30	N/A	N/A
	R	50	60	50	40	N/A	N/A
EO	L	40	70	50	50	50	80
	R	30	40	40	40	40	40
GN	L	100	100	100	100	90	100
	R	60	100	100	100	100	100
RB	L	40	70	80	70	40	60
	R	40	60	80	40	60	60
ON	L	50	80	40	50	90	100
	R	50	90	80	60	100	100
HN	L	90	70	100	80	90	80
	R	70	50	90	40	40	70
UI	L	50	100	100	100	100	100
	R	60	100	90	90	100	100
Mean		51.87	71.88*	69.38	60.63	73.57	76.43

All numbers represent percentage success rate over 10 trials in one direction.
* Difference between the beginning and end of the session is significant at 0.01 level.

70% at the end and that trend continued into the beginning of session 2 achieving close to 100% correct classification at times. However, this was followed by a period of worse than chance performance throughout the remainder of session 2. In session 3, performance improved again, dramatically at first, but did not match the high performance from the beginning of session 2 settling instead at about 60%.

An interesting case is presented by the performance of subject UI which improved dramatically half-way through session 1. Questioned about it after session 2, UI admitted to having found a successful strategy that involved concentrating his gaze at the part of the screen corresponding to the desired direction of movement. Because UI did not use this strategy during training, it was considered unlikely that the classification would be based on eye saccades. UI was allowed to continue the experiment as normal and advised not to change this strategy. To further investigate this case, a series of 3 short recordings were performed after the end of experiment. In these, accurate feedback was provided as in session 1 and UI was asked to (1) continue concentrating his gaze and moving his wrist as he had done so far, (2) only use his gaze and (3) only use wrist movement. The results for these sessions were respectively 100% (16 trials), 100% (19 trials) and 86.11% (36 trials). This demonstrates that although the eye movements were helpful for classification, very high classification rates could be achieved without them as well. UI claimed that moving his gaze helped him concentrate on the space and the direction of movement, which looking at the arrow did not. In the last trial, he was advised to keep his gaze concentrated below the arrow into the distance, which

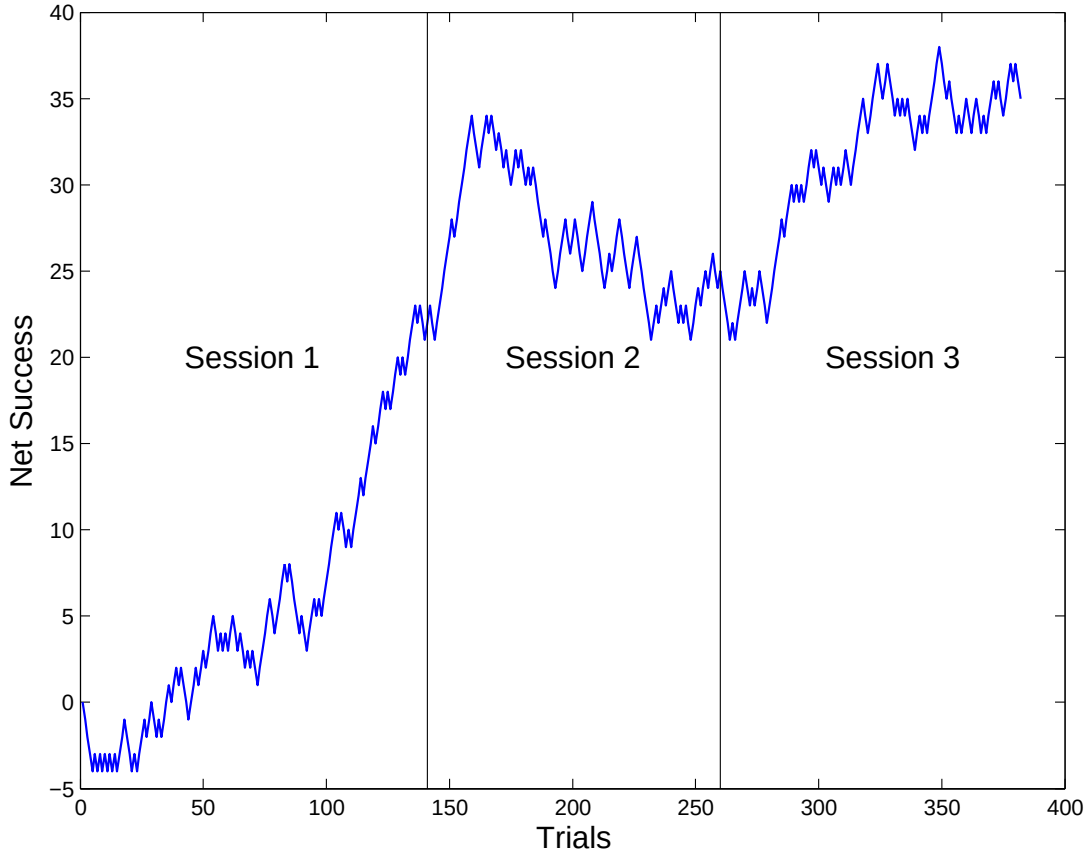


Figure 6.6: Cumulative success rate for subject RB over the three different feedback modalities. ©2010 IEEE

he claimed "also worked". It is interesting to note that the standard deviation of UI's reaction time increased as the experiment progressed without affecting his high classification rates. Excluding UI's data from the analysis does not affect the significance of the results reported earlier.

Response Time

Mean and standard deviation of the subjects' response times in each session are presented in Table 6.3. Average response times range between 353.3 and 580 ms. Subjects also varied greatly in consistency of task performance as reflected by the standard deviations of their response times varying between 48.4 and 166.9 ms. In general, the more successful subjects (e.g. GN, RB or HN) were also the more consistent, but the (negative) correlation between the mean classification rate in each session (as reported in Table 6.1) and the standard deviation of the response times was weak and not significant ($r = -0.185$, $p = 0.199$).

Table 6.3: Average response times in milliseconds for all subjects. ©2010 IEEE

Subject		Training	Session 1	Session 2	Session 3
BZ	μ	479.2	531.4	483.1	458.1
	σ	90.1	105.0	107.2	93.3
EH	μ	372.9	469.8	405.2	N/A
	σ	75.7	109.6	73.4	N/A
EO	μ	549.6	581.0	576.8	568.6
	σ	102.0	166.9	140.1	143.8
GN	μ	408.6	449.1	455.6	477.6
	σ	99.6	56.0	84.8	85.0
RB	μ	466.6	440.5	408.3	395.9
	σ	58.8	49.6	53.3	49.5
ON	μ	509.0	507.4	460.5	442.6
	σ	89.1	113.5	89.7	97.6
HN	μ	370.6	358.8	353.5	373.1
	σ	56.9	52.9	48.4	66.4
UI	μ	398.0	376.0	524.7	580.1
	σ	83.7	67.5	106.9	112.4

All values reported in milliseconds; μ — mean; σ — standard deviation.

Error Potentials

A typical average difference in the EEG recorded at electrode AF_2 between correct and erroneous trials during and after feedback provision in session 1 is presented in Figure 6.7 (data from a single subject). Characteristic features of the waveform are a trough 800 ms after the start of classification feedback, a peak with latency of 1800 ms and another trough at 2300 ms, all of which have reached statistical significance ($p \leq 0.05$) in a two-sample two-tailed t-test as indicated by the gray areas of the graph. Similar results were obtained at electrode AF_1.

6.4 Discussion and Future Work

The large improvement in classification rates during the first closed-loop session and the low initial classification results could be explained by a decrease in the subjects' attention and motivation following the longer (5–10 minutes) break during which the classifier was being trained (after the initial 60 trials session). However, this is not likely as subjects were aware that the "real" experiment was about to begin, were quite excited about it and, as they had only performed some 60 trails so far, not yet tired. A more probable explanation is that in the presence of accurate feedback of their (and the system's) performance, the subjects found and learnt to induce EEG features that the classifier was able to recognise with greater success but that were potentially different from those generated during the initial training session with no feedback. An extreme example of this behavior is the strategy found consciously by subject UI.

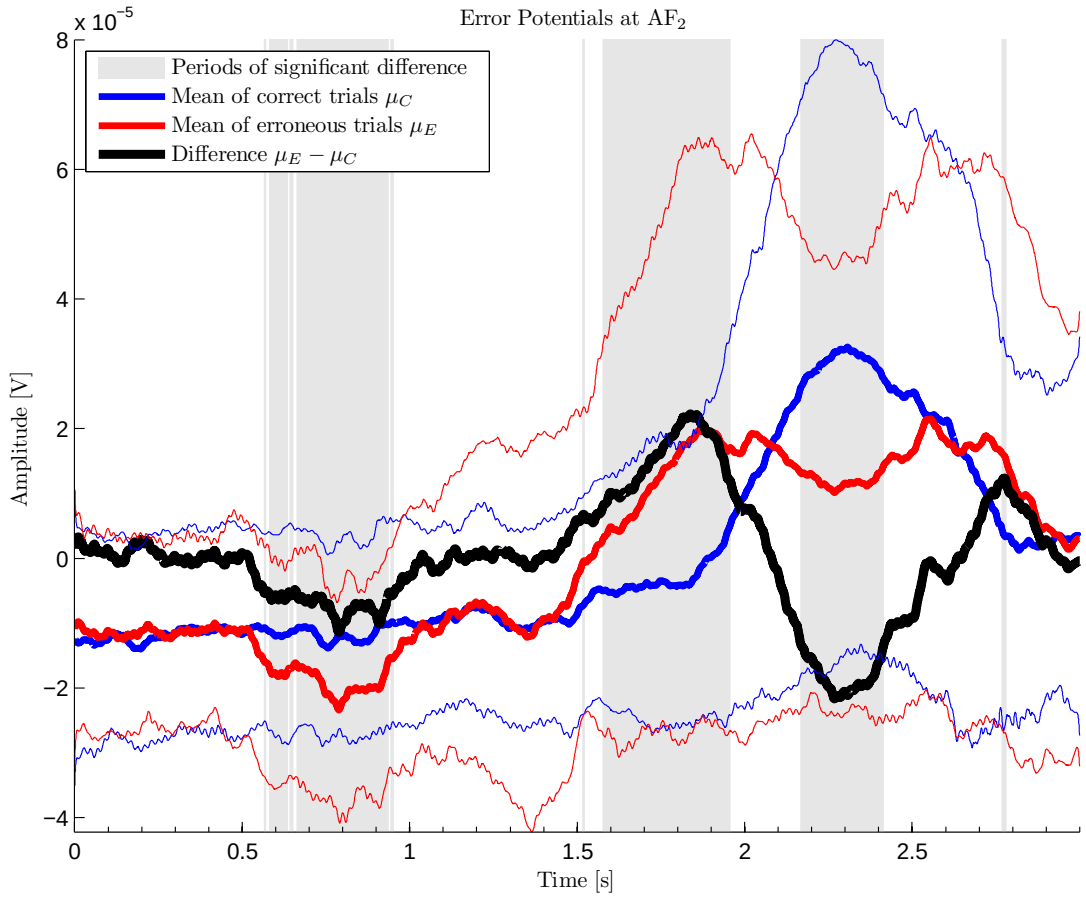


Figure 6.7: Error potentials recorded at electrode AF_2 during (first 2s) and after feedback presentation in subject RB. Feedback begins at t = 0s.The gray areas indicate periods where means of correct and erroneous trials (bold blue and red lines, respectively) differ significantly ($p < 0.05$, two-sample t-test). The thinner lines represent the range of plus/minus one standard deviation from the means in corresponding colors. ©2010 IEEE

The error potentials found in fronto-central electrodes offer further evidence of the subjects' reaction to erroneously classified trails and confirm findings of Schalk et al. (2000) and Ferrez and del R. Millán (2008) that errors committed by the system and not the subject also elicit error potentials. However, the exact features of the error potentials found in this study appear later and are less pronounced than those reported by Ferrez and del R. Millán (2008), possibly reflecting the time required to recognise the change in the direction of the virtual wheelchair's movement. Automatic detection of error potentials could improve the safety and speed of BCIs as well as allow for adaptive classifiers.

The poor classification results obtained in the open-loop feedback session as compared to the closed-loop session indicate that in absence of accurate feedback, the subjects' performance deteriorates. However, the effect seems less pronounced for better performing subjects with classification rates close to 100%, which is not surprising —

the better are the true results, the closer to reality is the simulated 100% classification rate.

Better results in the last session with open-loop feedback and artificial errors confirm the positive impact of motivation on BCI performance (Friedmann et al., 2004; Leeb, Lee, Keinrath, Scherer, Bischof and Pfurtscheller, 2007; Ron-Angevin and Daz-Estrella, 2009). However, unlike in the closed-loop session, no significant improvement throughout the session was observed, and the average results were not significantly different from those attained at the end of the first session. Both of these findings corroborate subject learning in the closed-loop session, but further research would be required to confirm if motivation alone is sufficient to maintain the high performance achieved through learning, as seems to be the case in this study. No evidence of different effect of feedback distortion on well and poorly performing subjects was found, contrary to the findings of Barbero and Grosse-Wentrup (2010).

In the present study, the subjects had no chance to practice the wrist movement task prior to the experiment other than the 60 trials that served to train the classifier. Analysis of the subjects' response times seems to indicate a relation between consistency in task performance and classification accuracy, although further experiments would be required to confirm this finding. Nonetheless, the large difference between different subjects' consistency attest the need for additional training for some subjects. Task performance practice before the experiment would also improve the quality of the data used to train the classifier.

The results presented in this chapter also point to the need for a more extensive training session. Future work with S-BCI could incorporate an iterative training and testing scenario whereby data from testing sessions are used to retrain the classifier during the experiment. This approach would allow to compensate for changes in the EEG features caused by the subjects' response to the feedback and thus facilitate learning. Additionally, subjects could benefit from a simple practice session at the very beginning of the experiment that would allow them to find the most comfortable way of performing the tasks assigned and develop consistency in doing so.

In the experiments presented here, unlike most BCI studies, in order to allow subject task performance analysis actual movements were performed by the subjects. While previous studies of our BCI system found that classification is based exclusively on the subjects' brain activity (Valsan et al., 2006), once lessons learnt from this study are taken into account, future work should nevertheless use imagined rather than actual movements, as required for the potential use of BCI by paralysed patients.

Future effort should also be directed at developing an adaptive system utilising the error-potentials and using more sophisticated feature extraction and classification algorithms. Improved classification accuracy will then facilitate incorporating more directions of wrist movements.

6.5 Summary

The chapter presented an application of the virtual reality wheelchair developed in this thesis in tandem with previous work on motion intention classification from EEG signals to the study of subject learning during BCI operation. The simulator forms an integral part of the Strathclyde BCI (S-BCI) and provides a user-cooperative environment.

The study demonstrates how the human-in-the-loop design enhances the classification performance of the S-BCI through voluntary behaviour modification brought about by accurate feedback. The importance of motivation and need for additional training are also discussed. The chapter provides experimental evidence that artificial introduction of random errors improves subjects' ability to control the BCI and confirms the presence of error potentials associated with the system's misinterpretation of the subject's intentions.

Chapter 7

Conclusion

The objective of the project was to develop an immersive virtual reality electric powered wheelchair simulator for brain computer interface research at the Neurophysiology Lab, University of Strathclyde. Literature research revealed a need in academia and clinical practice for a wheelchair driving simulator that can serve for patient training and evaluation as well as testing and development of semi-autonomous wheelchair systems. Previously reported simulators only fulfilled either one of these roles — training simulators did not offer the tools and flexibility required for smart wheelchair development, while the robotics simulators used in the latter field were too graphically simplistic for use in patient training.

Based on an analysis of usage scenarios of four user groups — clinicians, researchers, patients and developers — a set of 41 detailed requirements were identified for the design of the simulator. Of those, 35 were satisfied and 4 were partially satisfied. Two requirements could not be met due to a lack of interest from potential industrial partners.

The simulator takes advantage of a commercial game engine to provide high fidelity graphics and physics simulation, and a robotics simulator to provide sensors. A model of a common commercial wheelchair has been developed, including such details as free castor wheels and a functional kerb climber. Its simulated physical properties have been derived theoretically to mirror the abilities of its real counterpart, taking into account the mass distribution of the occupant as well as the wheelchair.

The simulator can run on a standard PC with a typical monitor or, for an enhanced immersive effect, a spherical display system, achieving sufficient frame rate (above 30 fps) in both cases. The definition of the picture for immersive display can be adjusted depending on the power of the computer. Settings that control the geometry displayed on the spherical display can be optimised to a subject's eye level (0.5–1.4 m) through an automated procedure. After optimisation, positions of displayed objects as seen by the subject are in most cases accurate to within a few centimetres.

The simulator has been evaluated by a group of eleven able-bodied volunteers, who filled in a questionnaire assessing such factors as immersion, flow, boredom and annoyance. Main findings include a high score for both flow and immersion (2.40 and 2.33 out of 4 respectively), and a very low score for boredom and annoyance (negative affect, 0.50 out of 4). These results suggest the simulator is suitable for prolonged use in training. Additionally, most respondents judged the speed and handling of the wheelchair as "about right", although further testing will be necessary in the future as they were not actual wheelchair users.

A novel method of wheelchair control using five discrete commands was developed to allow flexibility and smoothness similar to that afforded by the conventional joystick. This method of control is suitable for use with BCIs which usually have a limited number of command states (up to four were reported for the motor imagery paradigm). Using a key-pad to emulate BCI commands, eleven first time users of the simulator navigated a bespoke virtual obstacle course. Their performance was evaluated in terms of average and top speed, time to complete, distance from ideal path and number of collisions. The results were compared with those achieved when using a small joystick similar to those installed on wheelchairs to navigate the course. Encouragingly for the future use of the proposed algorithm for BCI-based wheelchair control, little difference was found in subjects' performance between the two devices. Only the number of collisions was significantly higher when using the keypad, but a large part of the difference can be attributed to problems with the keypad itself and not the underlying control algorithm. Overall, the results indicate that the proposed control algorithm is intuitive and can quickly be learnt.

Additionally, interesting trends were found relating some performance measures and the subjects' age and gender, whereby age was generally correlated with worse performance, while female subjects completed the course faster but with more collisions. These findings indicate that different training schemes may be most effective for different subjects groups, although a higher n study is required to confirm the trends. Moreover, a slight decrease in performance from session to session was found, independent of the control device. If confirmed, this trend, likely caused by over-confidence, will require counter-measures in a training scenario. The study also revealed distinct control usage patterns for poor and high performing subjects that may be useful in assessing profficiency of wheelchair users.

The simulator was successfully integrated into ongoing work on movement intention detection from EEG signals through a bespoke Matlab® toolbox. Thus, the University of Strathclyde's first BCI system complete with visual biofeedback was created. The system was used for a study of the impact feedback provision has on BCI classification rates in a cued wrist movement experiment. Average results from eight subjects show quick (over 120 trials in 18 minutes) and pronounced (20%) improvement in classification results when accurate real-time performance feedback was provided. The effect largely disappeared when, unknown to the subjects, the feedback was removed,

although artificially introduced errors seem to improve performance by maintaining motivation. These results indicate that future BCI systems need to be adaptive to benefit from and account for the users' ability to rapidly find and learn to repeatedly induce well classifiable features in their EEG signal. The study also confirmed the presence of a detectable error potential associated with mis-classification that could be used for continues online performance monitoring and adaptation in BCI systems.

7.1 Contributions

The project has led to a number of significant engineering achievements:

- Integrated, for the first time[1], a robotics simulator (USARSim) with spherical display technology (SPIClops). The unique advantage of this solution over other wheelchair simulators is its ability to accurately simulate physical interactions and sensor input while providing high fidelity immersive graphics.
- Developed a graphical user interface allowing non-specialist personnel to use the simulator for both patient training (including self-administered training) and smart wheelchair prototyping.
- Built a reusable 3D geometric model of a common electric powered wheelchair and integrated it into USARSim. Its physical properties including distribution of mass and motor torque have been carefully calculated such as to accurately simulate the behaviour of a real occupied wheelchair.
- Implemented freely rotating, self-aligning castor wheels and a functional kerb climber as part of the wheelchair model. It is the first[1] simulated wheelchair with these features.
- Developed a Matlab® interface allowing remote control of the simulation.
- Built a bespoke virtual environment for user training and system testing featuring turns of varied difficulty, tight passages between obstacles, a kerb and a ramp.
- Validated the settings used to control the geometry of scenes displayed on the spherical screen.

Facilitated by these developments, the major contributions to knowledge of the thesis are:

- Developed and implemented a novel method to derive a quantitative model of the wide-angle lens used to project images on the spherical screen using precise measurements with a theodolite and image registration techniques.

[1]To the best of the author's knowledge

- Developed a procedure to optimise the geometry displayed on the spherical screen for individual subject's eye level within the range of 0.5–1.4 m. Optimised geometry allows accurate perception of distance, increasing the immersive effect and the utility of the system for training prospective wheelchair users.
- Developed a novel control method facilitating smooth operation of a wheelchair with a set of only five discrete commands. The proposed method offers more possible trajectories than any comparable system[1], while offering an intuitive and fixed mapping between commands and actions.
- Integrated the wheelchair simulator with ongoing work on motion intention detection from EEG signals, resulting in the Strathclyde BCI — the first BCI system at the University of Strathclyde with real-time visual biofeedback capability.
- Demonstrated the existence of a learning effect and an error potential associated with system errors during BCI use by healthy volunteers in presence of real-time visual feedback.

7.2 Future Work

Notwithstanding the contributions listed above, the development of the Strathclyde Brain Computer Interface and the associated virtual powered wheelchair simulator is an ongoing effort. Offered below are suggestions for future work, some of which stem from limitations that due to constraints on time and resources could not be addressed in this thesis, while others represent an evolution of the design requirements identified at the outset (chapter 2).

Software

To offer greater flexibility for both patient training and smart wheelchair testing, the simulator should include at least one more wheelchair model of a front-wheel drive wheelchair, and preferably another one of a centre-drive wheelchair, although the latter are less common. This will address the partially fulfilled functional requirement 3m. The main effort required is that of developing a 3D geometric model which can then be imported as a static mesh into UnrealEd. Depending on the chosen wheelchair to model, one may be able to re-use the wheels and castors, so only the frame will have to be modelled. Once that is done, it should be possible to add the new wheelchair to UT2004, and hence USARSim, by writing a new section in the `USARBot.ini` file and sub-classing the existing `Wheelchair` class. By overriding the default properties only, one will be able to re-use the methods developed by the author to control the cue arrow and detect collisions.

A further enhancement to the wheelchair model, falling under functional requirement 3a, would be a simulation of the motors' torque curve. At the moment, the torque

can change instantaneously resulting in unrealistically high acceleration, especially at low speeds. The virtual wheelchair's handling could be improved, for example, by artificially smoothing the control signal.

As an additional functional requirement for the benefit of the patient, an on-screen indicator of the current state of the discrete command controller presented in chapter 5 should be developed. This could be implemented as an additional static mesh with changing textures attached to the armrest of the wheelchair model where the controller is usually mounted. Additionally, as work towards building a BCI controlled wheelchair progresses, the need for autonomous behaviour of the vehicle will likely grow, adding another functional requirement. The required sensors can readily be simulated by USARSim, but care must be taken to account for the possibly changed distribution of mass and collision geometry of the model. Moreover, specialised virtual environments will likely be required to test the new functions.

Graphics

The performance of the simulator in terms of the number of displayed frames per second could be improved by moving away from the Microsoft Foundation Classes which are not well suited for multi-threaded real time applications. In particular, the stitching process would benefit from running in a separate thread started regularly by a high-resolution timer as opposed to one that posts a message to the application's message queue as is the case now. Further improvements in performance can be gained by using a proper joystick instead of a mouse-replacement device which sends an excessive amount of messages to the application, crowding the message queue. The challenge is to find (or build) a PC joystick with the look and feel of those used on wheelchairs.

The quality of the simulator's graphics (and presumably performance) would also improve with a move from the UT2004 engine to the newer UT3 version that became available recently. This might require a major overhaul of the UnrealScript classes implementing the behaviour of the wheelchair model and depends on an update to USARSim itself, currently an alpha release.

Hardware

The major effort in terms of hardware enhancements should be directed at developing support for a modular wheelchair control system, as was the original intention of this project (functional requirement 3j). This would allow patients to use their own hardware with the simulator, thus catering for their individual needs in terms of controller type and placement. Perhaps now that the simulator has been built, described and tested, an industrial partner could be found.

Additionally, replacing the current projector in the VisionStation system should be considered. This could be done at a relatively low cost as long as the current lens will fit the new projector. The current device offers resolution of only 1024×768 and produces

a tilted image, a defect that cannot be serviced in the UK, while modern devices often offer the HD standard (1400×1050). However, this change will only make sense if the software bottlenecks are addressed first, as mentioned in the previous section.

As suggested by Dr. Mariel Purcell (pers. comm. 5 August 2011) and mentioned in chapter 2, a blood pressure/pulse monitor should be included in the system when used by patients with spinal cord injury. This will allow to monitor the patients' well-being while using the simulator.

Brain-Computer Interface

As an integral part of the Strathclyde BCI, the simulator's development depends on that of the other components. Future work should be targeted towards developing an asynchronous multi-class motion intention detection algorithm. As concluded in chapter 6, the algorithm should be adaptive to best utilise the subjects' ability to adjust their performance in response to feedback. As a step towards that goal, an iterative testing and training procedure should be developed and tested. Ultimately, however, an online algorithm must be developed that will use the error potentials described in chapter 6 to detect errors in intention detection. Moreover, future work must move away from actual movements towards imagined ones.

Experimental Work

Further experimental work is also required with the simulator itself. Primarily, an assessment by experienced electric powered wheelchair users is needed before the system can be used for training prospective ones. This should concentrate on the realism of the simulated wheelchair's handling and the quality of the immersive effect, and serve to identify areas that need further improvements. Additionally, as discussed in chapter 5, the tentative findings relating subjects' age and gender to their performance using the simulator may have important implications for customising training scenarios for prospective wheelchair users and, as such, merit further investigation.

We have designed, developed and implemented an electric powered wheelchair simulator offering an immersive virtual environment. The simulator's design was mathematically modelled and its operation was experimentally verified. The simulated wheelchair was successfully interfaced with a real-time brain computer interface.

Bibliography

Adelola, I. A., Cox, S. L. and Rahman, A. (2005). *VEMS - Training Wheelchair Drivers*, Vol. 16 of Pruski and Knops (2005), pp. 757–761.

Adelola, I. A., Rahman, A. and Cox, S. L. (2003). Conjoint analysis in virtual reality based powered wheelchair rehabilitation of children with disabilities, *Technology and Persons with Disabilities Conference*, California State University, Northridge Center on Disabilities. http://www.csun.edu/cod/conf/2003/proceedings/263.htm

Allison, B. Z., Wolpaw, E. W. and Wolpaw, J. R. (2007). Braincomputer interface systems: progress and prospects, *Expert Review of Medical Devices* **4**(4): 463–474. http://www.expert-reviews.com/doi/abs/10.1586/17434440.4.4.463

Balakirsky, S., Scrapper, C. and Messina, E. (2005). Mobility open architecture simulation and tools environment, *Proceedings of the 2005 Knowledge Intensive Multi-Agent Systems (KIMAS) Conference*, Waltham, MA, USA.

Barbero, A. and Grosse-Wentrup, M. (2010). Biased feedback in brain-computer interfaces, *Journal of NeuroEngineering and Rehabilitation* **7**(1): 34. http://www.jneuroengrehab.com/content/7/1/34

Barea, R., Boquete, L., Mazo, M. and Lopez, E. (2002). System for assisted mobility using eye movements based on electrooculography, *IEEE Trans. Neural Syst. Rehabil. Eng.* **10**(4): 209–218.

Bayliss, J. and Ballard, D. (2000). A virtual reality testbed for brain computer interface research, *IEEE Trans Rehabil Eng* **8**(2): 188–190.

Bourhis, G. and Agostini, Y. (1998). The VAHM robotized wheelchair: System architecture and human-machine interaction, *J Intell Rob Syst Theor App* **22**(1): 39–50. http://www.springerlink.com/content/m1217095583791q2

Brown, M., Majumder, A. and Yang, R. (2005). Camera-based calibration techniques for seamless multiprojector displays, *IEEE Trans. Vis. Comput. Graphics* **11**(2): 193–206.

Browning, D. R., Cruz-Neira, C., Sandin, D. J., DeFanti, T. A. and Edel, J. G. (1994). Input interfacing to the CAVE by persons with disabilities, *Conference on Virtual Reality and Persons with Disabilities (Proceedings)*, California State University (CSUN), Center on Disabilities. http://www.csun.edu/cod/conf/1994/proceedings/Licd~1.htm

Busby, J., Parrish, Z. and Van Eenwyk, J. (2004). *Mastering Unreal Technology: The Art of Level Design*, Sams Publishing.

Carlson, W., Swan II, J. E., Stredney, D. and Blostein, B. (1994). The application of virtual wheelchair simulation to the determination of environmental accessibility and standards compliance in architectural design, *Proceedings of the Symposium on Computers & Innovative Architectural Design, The 7th International Conference on Systems Research, Informatics & Cybernetics*, Baden-Baden, Germany.

Carpin, S., Lewis, M., Wang, J., Balakirsky, S. and Scrapper, C. (2007). USARSim: a robot simulator for research and education, *2007 IEEE International Conference on Robotics and Automation*, pp. 1400–1405.

Chen, X. and Bai, O. (2009). Towards multi-dimensional robotic control via noninvasive brain-computer interface, *Proc. CME Complex Medical Engineering ICME Int. Conf*, pp. 1–5.

Coleman, T. F. and Li, Y. (1996). An interior trust region approach for nonlinear minimization subject to bounds, *SIAM Journal on Optimization* **6**(2): 418–445.
http://link.aip.org/link/?SJE/6/418/1

Conway, B. A., Reid, C. and Halliday, D. M. (2004). Low frequency cortico-muscular coherence during voluntary rapid movements of the wrist joint, *Brain Topography* **16**: 221–224.

Cooper, R. A., Ding, D., Simpson, R., Fitzgerald, S. G., Spaeth, D. M., Guo, S., Koontz, A. M., Cooper, R., Kim, J. and Boninger, M. L. (2005). Virtual reality and computer-enhanced training applied to wheeled mobility: An overview of work in pittsburgh, *Assist Technol* **17**(2): 159–170.
http://www.ncbi.nlm.nih.gov/pubmed/16392719

Crocker, D. M. E. and Turner, J. D. (1988). A microcomputer training system for electric wheelchair users, *Comput Educ* **12**(2): 301–309.

Csikszentmihalyi, M. and Csikszentmihalyi, I. (1988). *Optimal experience. Psychological studies of flow in consciousness*, Cambridge University Press, Cambridge, UK.

de Leva, P. (1996). Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters, *J Biomech* **29**(9): 1223–1230.
http://www.sciencedirect.com/science/article/B6T82-3WF7DVD-H/2/7d02d6a65ae8ef6492126785e6ee4d3e

Deecke, L., Scheid, P. and Kornhuber, H. (1969). Distribution of readiness potential, pre-motion positivity, and motor potential of the human cerebral cortex preceding voluntary finger movements, *Experimental Brain Research* **7**: 158–168.
http://dx.doi.org/10.1007/BF00235441

Department of Health (2005). The National Service Framework for long term conditions. annex 4 - incidence and prevalence of some neurological conditions in the uk.

Desbonnet, M., Cox, S. L. and Rahman, A. (1998). Development and evaluation of a virtual reality based training system for disabled children, *in* R. Sharkeand and Lindström (eds), *The Second European Conference on Disability, Virtual Reality and Associated Technologies*, Mount Billingen, Skvde, Sweden, pp. 177–182.
http://www.icdvrat.reading.ac.uk/1998/papers/1998_22.pdf

Ding, D., Cooper, R. A., Guo, S. and Corfman, T. A. (2004). Analysis of driving backward in an electric-powered wheelchair, *IEEE Trans. Control Syst. Technol.* **12**(6): 934–943.

Drillis, R. and Contini, R. (1966). Body segment parameters, *Technical Report 1166-03*, Office of Vocational Rehabilitation, New York.

Dunne, A. K., Mallon, J. and Whelan, P. F. (2010). Efficient generic calibration method for general cameras with single centre of projection, *Computer Vision and Image Understanding* **114**(2): 220–233. Special issue on Omnidirectional Vision, Camera Networks and Non-conventional Cameras.
http://www.sciencedirect.com/science/article/B6WCX-4WS9BX1-1/2/2acad75f4b4776b640ca3c86c99ba556

Edlinger, G., Holzner, C., Guger, C., Groenegress, C. and Slater, M. (2009). Brain-computer interfaces for goal orientated control of a virtual smart home environment, *Neural Engineering, 2009. NER '09. 4th International IEEE/EMBS Conference on*, pp. 463–465.

ELUMENS (2001). *The SPIClops API*.

Erren-Wolters, C. V., van Dijk, H., IJzerman, M. J. and Jannink, M. J. (2007). Virtual reality for mobility devices: training applications and clinical results: a review, *Int J Rehabil Res* **30**(2): 91–96.
http://www.intjrehabilres.com/pt/re/intjrr/abstract.00004356-200706000-00001.htm

Falkenstein, M., Hoormann, J., Christ, S. and Hohnsbein, J. (2000). Erp components on reaction errors and their functional significance: a tutorial., *Biol Psychol* **51**(2-3): 87–107.

Fattouh, A., Sahnoun, M. and Bourhis, G. (2004). Force feedback joystick control of a powered wheelchair: preliminary study, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 3, pp. 2640–2645.

Ferrez, P. and del R. Millán, J. (2008). Error-related EEG potentials generated during simulated braincomputer interaction, *Biomedical Engineering, IEEE Transactions on* **55**(3): 923–929.

Friedmann, D., Leeb, R., Keinrath, C., Guger, C., Garau, M., Pfurtscheller, G. and Slater, M. (2004). Navigating virtual reality by thought: First steps, *7th Annual International Workshop on Presence, PRESENCE 2004*, Valencia, Spain, pp. 160–167.

Gajadhar, B. J., de Kort, Y. A. W. and IJsselsteijn, W. A. (2009). Rules of engagement: Influence of co-player presence on player involvement in digital games, *International Journal of Gaming and Computer-Mediated Simulations* **1**(3): 14–27.

Galán, F., Nuttin, M., Lew, E., Ferrez, P. W., Vanacker, G., Philips, J. and Millán, J. D. R. (2008). A brain-actuated wheelchair: asynchronous and non-invasive brain-computer interfaces for continuous control of robots., *Clin Neurophysiol* **119**(9): 2159–2169.
http://dx.doi.org/10.1016/j.clinph.2008.06.001

Gehring, W. J., Coles, M. G., Meyer, D. E. and Donchin, E. (1995). A brain potential manifestation of error-related processing., *Electroencephalogr Clin Neurophysiol Suppl* **44**: 261–272.

Gentiletti, G., Gebhart, J., Acevedo, R., nez Suárez, O. Y. and nuelos, V. M.-B. (2009). Command of a simulated wheelchair on a virtual environment using a brain-computer interface, *IRBM* **30**(5-6): 218–225.
http://www.sciencedirect.com/science/article/B94S6-4XMTJXJ-1/2/b803e383a264e1982262e5ffad181e6a

Gerkey, B. P., Vaughan, R. T. and Howard, A. (2003). The Player/Stage Project: Tools for multi-robot and distributed sensor systems, *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, pp. 317–323.

Goshtasby, A. (1986). Piecewise linear mapping functions for image registration, *Pattern Recognition* **19**(6): 459 – 466.
http://www.sciencedirect.com/science/article/B6V14-48MPKB6-PV/2/66a50326acb67a6a0e854f12a4381905

Goshtasby, A. (1988). Image registration by local approximation methods, *Image Vision Computing* **6**(4): 255–261.

Grafton, S. T., Schmitt, P., Horn, J. V. and Diedrichsen, J. (2008). Neural substrates of visuomotor learning based on improved feedback control and prediction, *NeuroImage* **39**(3): 1383 – 1395.
http://www.sciencedirect.com/science/article/B6WNP-4PWF0MN-6/2/8a1146b9522aca559c38808b0a23c647

Grant, M., Harrison, C. and Conway, B. (2004). Wheelchair simulation, *Proceedings of the 2nd Cambridge Workshop on Universal Access and Assistive Technology*, Cambridge, UK, pp. 101–109.

Grychtol, B., Lakany, H., Valsan, G. and Conway, B. A. (2010). Human behavior integration improves classification rates in real-time BCI, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* **18**(4): 362–368.

Hamagami, T. and Hirata, H. (2004). Development of intelligent wheelchair acquiring autonomous, cooperative, and collaborative behavior, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, pp. 3525–3530.

Harrison, A., Derwent, G., Enticknap, A., Rose, F. and Attree, E. (2002). The role of virtual reality technology in the assessment and training of inexperienced powered wheelchair users, *Disabil Rehabil* **24**(8): 599–606.
http://www.ingentaconnect.com/content/apl/tids/2002/00000024/F0020011/art00005

Harrison, A., Derwent, G., Enticknap, A., Rose, F. D. and Attree, E. A. (2000). Application of virtual reality technology to the assessment and training of powered wheelchair users, *Proc. 3rd Intl Conf. Disability, Virtual Reality & Assoc. Tech.*, Alghero, Italy.

Harrison, C. S., Grant, M. and Conway, B. A. (2004). Haptic interfaces for wheelchair navigation in the built environment, *Presence* **13**(5): 520–534.

Hasdai, A., Jessel, A. S. and Weiss, P. L. (1998). Use of a computer simulator for training children with disabilities in the operation of a powered wheelchair., *Am J Occup Ther*

52(3): 215–220.
http://www.ajnonline.com/pt/re/ajn/abstract.00000446-199809000-00028.htm

Hashimoto, M., Takahashi, K. and Shimada, M. (2009). Wheelchair control using an EOG- and EMG-based gesture interface, *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics AIM 2009*, pp. 1212–1217.

Holden, M. K. (2005). Virtual environments for motor rehabilitation: review., *Cyberpsychol Behav* **8**(3): 187–211; discussion 212–219.
http://dx.doi.org/10.1089/cpb.2005.8.187

Hunt, G. and Brubacher, D. (1999). Detours: Binary interception of Win32 functions, *Proceedings of the 3rd USENIX Windows NT Symposium*, USENIX, Seattle, WA, pp. 135–143.
http://research.microsoft.com/sn/detours

IJsselsteijn, W. A., de Kort, Y. A. W. and Poels, K. (n.d.). The Game Experience Questionnaire: Development of a self-report measure to assess the psychological impact of digital games. Manuscript in preparation.

Inman, D. P. and Loge, K. (1995). Teaching motorized wheelchair operation in virtual reality, *Virtual Reality and Persons with Disabilities*, California State University, Northridge Center on Disabilities.
http://www.csun.edu/cod/conf/1995/proceedings/1001.htm

Inman, D. P., Loge, K. and Leavens, J. (1997). VR education and rehabilitation, *Commun ACM* **40**(8): 53–58.

Inman, D. P., Peaks, J., Loge, K. and Chen, V. (1994). Teaching orthopedically impaired children to drive motorized wheelchairs in virtual reality, *Virtual Reality and Persons with Disabilities*.

INVACARE (2006). *Spectra Plus: User Manual*. [Consulted: March 2008].
http://doclibrary.invacare.co.uk/Office\Europe\Marketing\MktDocUK.nsf/MAcceuil?openForm

Iturrate, I., Antelis, J. M., Kubler, A. and Minguez, J. (2009). A noninvasive brain-actuated wheelchair based on a P300 neurophysiological protocol and automated navigation, *IEEE Trans. Robot.* **25**(3): 614–627.

Kannala, J. and Brandt, S. S. (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**: 1335–1340.

Kawato, M. (1990). Feedback error learning neural network for supervised motor learning, *in* R. Eckmiller (ed.), *Advanced neural computers*, Elsevier Science Publishers B.V. (North-Holland), pp. 365–72.

Lakany, H. and Conway, B. (2005). Classification of wrist movements using EEG-based wavelets features, *27th Annual International Conference of the Engineering in Medicine and Biology Society IEEE-EMBS 2005*, pp. 5404–5407.
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1615704

Lakany, H. and Conway, B. (2007). Understanding intention of movement from electroencephalograms, *Expert Systems* **24**(5): 295–304.

Lalor, E. C., Kelly, S. P., Finucane, C., Burke, R., Smith, R., Reilly, R. B. and McDarby, G. (2005). Steady-state vep-based brain-computer interface control in an immersive 3D gaming environment, *EURASIP J. Appl. Signal Process.* **2005**: 3156–3164.

Laue, T., Spiess, K. and Röfer, T. (2006). SimRobot — A general physical robot simulator and its application in RoboCup, *in* A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi (eds), *RoboCup 2005: Robot Soccer World Cup IX*, Vol. 4020 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 173–183.
http://www.springerlink.com/content/wh81484n700q3q3h

Leeb, R., Friedman, D., Müller-Putz, G. R., Scherer, R., Slater, M. and Pfurtscheller, G. (2007). Self-paced (asynchronous) bci control of a wheelchair in virtual environments: A case study with a tetraplegic, *Comput Intell Neurosci* **2007**: Article ID 79642, 8 pages.

Leeb, R., Lee, F., Keinrath, C., Scherer, R., Bischof, H. and Pfurtscheller, G. (2007). Brain-computer communication: Motivation, aim, and impact of exploring a virtual apartment, *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* **15**(4): 473–482.

Leuthardt, E. C., Miller, K. J., Schalk, G., Rao, R. P. N. and Ojemann, J. G. (2006). Electrocorticography-based brain computer interface-the seattle experience, *IEEE Trans. Neural Syst. Rehabil. Eng.* **14**(2): 194–198.

Levine, S., Bell, D., Jaros, L., Simpson, R., Koren, Y. and Borenstein, J. (1999). The navchair assistive wheelchair navigation system, *IEEE Trans Rehabil Eng* **7**(4): 443–451.

Lewis, M. and Jacobson, J. (2002). Introduction, *Commun ACM* **45**(1): 27–31. Special Issue: Game engines in scientific research.

Lotte, F., van Langhenhove, A., Lamarche, F., Ernest, T., Renard, Y., Arnaldi, B. and Lcuyer, A. (2010). Exploring large virtual environments by thoughts using a braincomputer interface based on motor imagery and high-level commands, *Presence: Teleoperators and Virtual Environments* **19**(1): 54–70.
http://www.mitpressjournals.org/doi/abs/10.1162/pres.19.1.54

LU, C., Tsai, W., Sung, W. and Chen, J. (2007). Effects of powered wheelchair virtual-reality system on powered-wheelchair driving skills in people with SCI, *J Biomech* **40**(Suppl. 2: Program and Abstracts of the XXI Congress, International Society of Biomechanics): S716.
http://www.sciencedirect.com/science/article/B6T82-4PD6NSN-W4/2/2038ac5ccad406de606ed306097a6a1f

Marr, D. (1969). A theory of cerebellar cortex, *The Journal of Physiology* **202**(2): 437–470.
http://jp.physoc.org/content/202/2/437.abstract

MathEngine (2002). *Karma™ User Guide*.

Millán, J., Renkens, F., Mourino, J. and Gerstner, W. (2004). Noninvasive brain-actuated control of a mobile robot by human EEG, *IEEE Trans Biomed Eng* **51**(6): 1026–1033.

Morash, V., Bai, O., Furlani, S., Lin, P. and Hallett, M. (2007). Prediction of multiple movement intentions from cnv signal for multi-dimensional bci, *Proc. IEEE/ICME Int. Conf. Complex Medical Engineering CME 2007*, pp. 1946–1949.

Morey-Klapsing, G., Arampatzis, A. and Brüggemann, G. P. (2004). Choosing EMG parameters: comparison of different onset determination algorithms and EMG integrals in a joint stability study, *Clinical Biomechanics* **19**(2): 196–201.
http://www.sciencedirect.com/science/article/B6T59-4B3NMBC-2/2/7a42367abd1bcd0a4e46ddddd8916133

Muvdi, B. B., Al-Khafaji, A. W. and McNabb, J. W. (1997). *Statics for Engineers*, Springer.

National Spinal Cord Injury Statistical Centre (2010). Spinal cord injury facts and figures at a glance.
http://www.nscisc.uab.edu

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization, *The Computer Journal* **7**(4): 308–313.
http://comjnl.oxfordjournals.org/cgi/content/abstract/7/4/308

Nijholt, A. and Tan, D. (2008). Brain-computer interfacing for intelligent systems, *IEEE Intell. Syst.* **23**(3): 72–79.

Niniss, H. and Inoue, T. (2005). *Assessment of Driving Skills Using Virtual Reality: Comparative Survey on Experts and Unskilled Users of Electric Wheelchairs*, Vol. 16 of Pruski and Knops (2005), pp. 413–417.

Niniss, H. and Inoue, T. (2006). Electric wheelchair simulator for rehabilitation of persons with motor disability, *Symposium on Virtual Reality VIII (Proceedings)*, Brazilian Computer Society (BSC), Belém (PA).
http://www.sbc.org.br/bibliotecadigital/download.php?paper=424

Niniss, H. and Nadif, A. (2000). Simulation of the behaviour of a powered wheelchair using virtual reality, *Proc. 3rd Intl Conf. Disability, Virtual Reality & Assoc. Tech.*, Alghero, Italy, pp. 9–14.

Ojala, J., Inoue, K., Sasaki, K. and Takano, M. (1991). Development of an intelligent wheelchair using computer graphics animation and simulation, *Comput Graph Forum* **10**(4): 285–295.
http://www.blackwell-synergy.com/doi/abs/10.1111/1467-8659.1040285

Patton, J. and Mussa-Ivaldi, F. (2004). Robot-assisted adaptive training: custom force fields for teaching movement patterns, *Biomedical Engineering, IEEE Transactions on* **51**(4): 636–46.

Pfurtscheller, G., Brunner, C., Schlögl, A. and da Silva, F. L. (2006). Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks, *NeuroImage* **31**(1): 153 – 159.
http://www.sciencedirect.com/science/article/B6WNP-4J4HK9W-3/2/e2ea7ba6446cd2f634604f1cc42b6080

Pires, G., Castelo-Branco, M. and Nunes, U. (2008). Visual P300-based BCI to steer a wheelchair: A bayesian approach, *Proc. 30th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBS 2008*, pp. 658–661.

Pithon, T., Weiss, T., Richir, S. and Klinger, E. (2009). Wheelchair simulators: A review, *Technology and Disability* **21**(1): 1–10.

Pronk, C. N. A., de Klerk, P. C., Shouten, A., Grashuis, J., Niesing, R. and Bangma, B. D. (1980). Electric wheelchair simulator as a man-machine system, *Scand J Rehabil Med* **12**(3): 129–135.

Pruski, A. and Knops, H. (eds) (2005). *From Virtuality to Reality*, Vol. 16 of *Assistive Technology*, IOS Press.

Putzke, J. D., Richards, J. S., Hicken, B. L. and DeVivo, M. J. (2002). Predictors of life satisfaction: a spinal cord injury cohort study., *Arch Phys Med Rehabil* **83**(4): 555–561.

Quek, M., Boland, D., Williamson, J., Murray-Smith, R., Tavella, M., Perdikis, S., Schreuder, M. and Tangermann, M. (2011). Simulating the feel of brain-computer interfaces for design, development and social interaction, *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, ACM, New York, NY, USA, pp. 25–28. http://doi.acm.org/10.1145/1978942.1978947

Randria, I., Romero, F. C., Abellard, P., Ben Khelifa, M. M., Abellard, A. and Ramanantsizehena, P. (2009). A performant low cost wheelchair simulator for rehabilitation planning, *Proc. 35th Annual Conf. of IEEE Industrial Electronics IECON '09*, pp. 2169–2174.

Raskar, R., Brown, M. S., Yang, R., Chen, W.-C., Welch, G., Towles, H., Scales, B. and Fuchs, H. (1999). Multi-projector displays using camera-based registration, *Proc. Visualization '99*, pp. 161–522.

Rebsamen, B., Burdet, E., Guan, C., Zhang, H., Teo, C. L., Zeng, Q., Laugier, C. and Ang Jr., M. H. (2007). Controlling a wheelchair indoors using thought, *IEEE Intell Syst* **22**(2): 18–24.

Röfer, T. (1998). Strategies for using a simulation in the development of the Bremen Autonomous Wheelchair, *Simulation: Past, Present and Future. 12th European Simulation Multiconference 1998. ESM'98*, Manchester, UK, pp. 460–464.

Ron-Angevin, R. and Daz-Estrella, A. (2009). Brain-computer interface: Changes in performance using virtual reality techniques, *Neuroscience Letters* **449**(2): 123–127. http://www.sciencedirect.com/science/article/B6T0G-4TVHSTS-2/2/348693b2efd5c364ad3d810810ad6559

Schalk, G., Wolpaw, J. R., McFarland, D. J. and Pfurtscheller, G. (2000). EEG-based communication: presence of an error potential, *Clinical Neurophysiology* **111**(12): 2138–44. http://www.sciencedirect.com/science/article/B6VNP-41PP15D-6/2/e58e98a7c9e9eb4b96a660062fe05dd2

Scrapper, C., Balakirsky, S. and Messina, E. (2006). MOAST and USARSim - a combined framework for the development and testing of autonomous systems, *Proceedings of the SPIE Defense and Security Symposium*, Orlando, FL, USA.

Secco, E. L., Sottile, R., Davalli, A., Calori, L., Cappello, A. and Chiari, L. (2007). VR-Wheel: a rehabilitation platform for motor recovery, *Virtual Rehabilitation*, pp. 39–43.

Sgouros, N. M. (2002). Qualitative navigation for autonomous wheelchair robots in indoor environments, *Auton Robot* **12**(3): 257–266.
http://www.springerlink.com/content/ldm5ln8a2fyhh1cb

Simpson, R. C. (2005). Smart wheelchairs: A literature review., *J Rehabil Res Dev* **42**(4): 423–436.

Stott, I. and Sanders, D. (2000). The use of virtual reality to train powered wheelchair users and test new wheelchair systems., *Int J Rehabil Res* **23**(4): 321–326.

Swan II, J. E., Stredney, D., Carlson, W. and Blostein, B. (1994). The determination of wheelchair user proficiency and environmental accessibility through virtual simulation, *Proceedings of 2nd Annual International Conference: Virtual Reality and Persons with Disabilities*, California State University, Northward, CA, pp. 156–161.

Tenenbaum, R. A. (2004). *Fundamentals of Applied Dynamics*, Advanced Texts in Physics, Springer.

Teymourian, A., Lueth, T., Graeser, A., Felzer, T. and Nordmann, R. (2008). Brain-controlled finite state machine for wheelchair navigation, *Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, ACM, New York, NY, USA, pp. 257–258.

Tsui, C. S. L., Jia, P., Gan, J. Q., Hu, H. and Yuan, K. (2007). EMG-based hands-free wheelchair control with EOG attention shift detection, *Proc. IEEE Int. Conf. Robotics and Biomimetics ROBIO 2007*, pp. 1266–1271.

Vaisman, L., Zariffa, J. and Popovic, M. R. (2010). Application of singular spectrum-based change-point analysis to EMG-onset detection, *Journal of Electromyography and Kinesiology* **20**(40): 750–760.
http://www.sciencedirect.com/science/article/B6T89-4YMXXXY-1/2/27424008db13d19130088439a6efb97a

Vallabhaneni, A., Wang, T. and He, B. (2005). Braincomputer interface, *in* B. He (ed.), *Neural Engineering*, Bioelectric Engineering, Springer US, pp. 85–121.

Valsan, G. (2007). *Brain computer interface using detection of movement intention*, PhD thesis, University of Strathclyde. Bioengineering Unit.

Valsan, G., Grychtol, B., Lakany, H. and Conway, B. A. (2009). The Strathclyde brain computer interface, *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 606–609.

Valsan, G., Worrajiran, P., Lakany, H. and Conway, B. (2006). Predicting intention and direction of wrist movement from eeg, *Advances in Medical, Signal and Information Processing, 2006. MEDSIP 2006. IET 3rd International Conference On* pp. 1–4.

van Koppenhagen, C. F., Post, M. W., van der Woude, L. H., de Witte, L. P., van Asbeck, F. W., de Groot, S., van den Heuvel, W. and Lindeman, E. (2008). Changes and determinants of life satisfaction after spinal cord injury: a cohort study in the netherlands., *Arch Phys Med Rehabil* **89**(9): 1733–1740.
http://dx.doi.org/10.1016/j.apmr.2007.12.042

Vanacker, G., del R. Millán, J., Lew, E., Ferrez, P. W., Moles, F. G., Philips, J., Brussel, H. V. and Nuttin, M. (2007). Context-based filtering for assisted brain-actuated wheelchair driving, *Computational Intelligence and Neuroscience* **2007**(119): Article ID 25130, 12 pages.

Velasco-Álvarez, F. and Ron-Angevin, R. (2009). Asynchronous brain-computer interface to navigate in virtual environments using one motor imagery, *in* J. Cabestany, F. Sandoval, A. Prieto and J. Corchado (eds), *Bio-Inspired Systems: Computational and Ambient Intelligence*, Vol. 5517 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 698–705.

Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S. and Schwartz, A. B. (2008). Cortical control of a prosthetic arm for self-feeding, *Nature* **453**(7198): 1098–1101.
http://dx.doi.org/10.1038/nature06996

Vialatte, F.-B., Maurice, M., Dauwels, J. and Cichocki, A. (2010). Steady-state visually evoked potentials: Focus on essential paradigms and future perspectives, *Progress in Neurobiology* **90**(4): 418 – 438.
http://www.sciencedirect.com/science/article/B6T0R-4XVK435-1/2/35b2e1a00c0200d98addacaca2d0c4c4

Špacapan, I., Kocijan, J. and Bajd, T. (2004). Simulation of fuzzy-logic-based intelligent wheelchair control system, *J Intell Rob Syst Theor App* **39**(2): 227–241.
http://www.springerlink.com/content/m6r4432v3531x426

Wang, J. and Balakirsky, S. (2007). *USARSim Manual V3.1.1.*
http://sourceforge.net/projects/usarsim

Wei, Y., Bajaj, P., Scheidt, R. and Patton, J. (2005). Visual error augmentation for enhancing motor learning and rehabilitative relearning, *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pp. 505 – 510.

Wirth, W., Hartmann, T., Böcking, S., Vorderer, P., Klimmt, C., Schramm, H., Saari, T., Laarni, J., Ravaja, N., Gouveia, F. R., Biocca, F., Sacau, A., Jäncke, L., Baumgartner, T. and Jäncke, P. (2007). A process model of the formation of spatial presence experiences, *Media Psychology* **9**(3): 493–525.

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G. and Vaughan, T. M. (2002). Brain-computer interfaces for communication and control, *Clin Neurophysiol* **113**(6): 767–791.
http://www.sciencedirect.com/science/article/B6VNP-45HFKTC-2/2/070472147433d00168e8d54909b982d2

Wolpaw, J. R. and McFarland, D. J. (2004). Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans, *Proc Natl Acad Sci U S A* **101**(51): 17849–17854.
http://www.pnas.org/cgi/content/abstract/101/51/17849

Wolpert, D., Ghahramani, Z. and Jordan, M. (1995). An internal model for sensorimotor integration, *Science* **269**(5232): 1880–1882.

Worrajian, P. (2009). *Towards development of brain-computer interface based on point-to-point movements*, PhD thesis, University of Strathclyde.

Yamada, H., Bando, N. and Muto, T. (2003). Hands-on wheelchair simulator using hydraulic motion base, *Transactions of the Japan Fluid Power System Society* **34**(5): 112–117. [In Japaneese].

Yoo, S., Fairneny, T., Chen, N., Choo, S., Panych, L. P., Park, H., Lee, S. and Jolesz, F. A. (2004). Brain-computer interface using fMRI: spatial navigation by thoughts, *NeuroReport* **15**(10): 1591–1595.
http://journals.lww.com/neuroreport/Fulltext/2004/07190/Brain_computer_interface_using_fMRI__spatial.12.aspx

Zatsiorsky, V. M., Seluyanov, V. N. and Chugunova, L. G. (1990). Methods of determining mass-inertial characteristics of human body segments, *in* G. G. Chernyi and S. A. Regirer (eds), *Contemporary Problems of Biomechanics*, CRC Press, Massachusetts, pp. 272–291.

Appendices

Appendix A

Electronic Resources

Electronic resources produced by the author throughout the project are included on the enclosed CD-ROM, which constitutes an integral part of the thesis. The CD-ROM contains the following files and directories:

[Matlab]	[UT2004]
[Mathematica]	[Wheelchair]
[Model]	COPYRIGHT
[TestImg]	Documentation.bat
[Textures]	EngD Thesis.pdf

where folders are indicated with square brackets. The contents of each folder is described below.

A.1 Model

The `Model` folder contains files pertaining to the 3D geometric model of a wheelchair created in Maya. The main file is `wheelchair_whole_recentre.mb`, which contains the entire model. All remaining `*.mb` files describe individual parts and were created for the purpose of exporting to UnrealEd. Additionally, for each part, a `*.dwg` file is included. Those were used for analysing the inertial properties with AutoVue™, as described in chapter 3.

A.2 Simulator source code and documentation

The `Wheelchair` directory contains the entire source code and documentation of the simulator application — the main software product of the thesis. The debug and release version of the executable `Wheelchair.exe` can be found in the `debug` and `release` folders, respectively. Browser-viewable documentation of the source code,

including all relevant files, classes and members, is in the `doc/html` folder and is also available through the `Documentation.bat` shortcut in the top level directory of the CD-ROM. The `Wheelchair` directory also contains VisualStudio project files (`Wheelchair.vcproj`, `Wheelchair.sln`), which define the application's compiler and linker commands, among others.

A.3 UT2004 files

The `UT2004` directory mimics the tree of Unreal Tournament's root directory. Included here are all the files needed to use the wheelchair model and the maps developed by the author. Of main interest are:

- `USARBot.ini` in the `System` folder — the main configuration USARSim configuration file. The "[USARBot.Wheelchair]" section defines the properties of the wheelchair model and its constituent parts. Additional sensors can be added here. When working with newer versions of USARSim, only that one section should be copied.
- the `USARBot/Classes` directory contains source files for Unreal classes. Three of those are modified USARSim files: `GroundTruth.uc` (added additional information to the GroundTruth message), `USARBotConnection.uc` and `USARRemoteBot.uc` (implemented communication with the arrow object). Thus, care must be taken when working with newer versions of USARSim as overwriting may cause loss of other functionality. New files in this folder are:
  - `Wheelchair.uc` — the main file defining the wheelchair's functionality.
  - `WheelchairRider.uc` — a class providing interface to the human avatar sitting in the wheelchair.

  Excerpts from the 'Wheelchair.uc' file as discussed in section 3.5 are reproduced below.
- `USARModels/Classes` directory contains source code for the classes defining the smaller objects attached to the main wheelchair model — the wheels, forks, curb climber and the arrow.
- `Textures` directory contains three UT2004 texture packages (`*.utx`), two of which are modified USARSim files:
  - `USARSim_Vehicles_Textures.utx` where a new texture was added in the Wheelchair group.
  - `ArdaArenaB.utx` — a modified version of USARSim's ArdaArena package, where useful street and outside environment textures were defined already and new ones were added by the author.

  The new package `EngD.utx` includes textures used in the VEs used for geometry validation, as described in chapter 4.

- `USARSim_Vehicles_Meshes.utx` in the `StaticMeshes` directory adds a "Wheelchair" group to the set of USARSim's static meshes. When working with newer versions of USARSim, care must be taken when overwriting this file as new features might be lost. To facilitate, a separate `W_USARSim_Vehicles_Meshes.utx` file includes only the new meshes added by the author.
- the `Maps` directory contains four new VEs built by the author:
  - `DM-Track_250_r50.ut2` is the obstacle course described in subsection 2.4.4 (revision 50)
  - `Hall_250.ut2` is the empty hall VE used for practice runs
  - `DM-GeometryBW_250.ut2` and `DM-Geometry_250_objects.ut2` define the virtual environments built for the validation measurements detailed in chapter 4.

A.4 Matlab files

The `Matlab` folder contains three Matlab toolboxes developed during the project. In the `WheelchairSimulator` subfolder a TCP/IP interface to the simulator is provided by the `SimControl` class. The functionality of translating user-defined control commands into normalised angular velocities of the wheelchair's two wheels is separated into a virtual `Conroller` class. The `StateMachine` class, derived from `Controller`, implements the control mechanism described in chapter 5. Further details and instructions for use are provided in the `README.m` file.

The `DomeToolbox` folder provides a number of functions and scripts used in chapter 4. Each file is documented separately. The `main.m` implements the optimisation procedure described in section 4.4. To find the optimal settings for a new subject, the lens and eye positions have to be defined in the first cell of the script. To obtain the correct values, the following readings must be taken:

- projector height read from the etched scale on the poles
- dome-pole distance read from the etched scale along the rails guiding the dome's wheels
- the user's eye level
- the distance between the user's eyes and the line connecting the poles.

A Microsoft Excel file `Template` is provided where these readings can be entered (on the "Settings" tab). The lens and eye position to be entered in the `main.m` will then be automatically calculated. Once these values are entered, the script can run without further interaction and will print the optimal settings on the screen.

The `PerformanceAnalysis` folder contains functions used to analyse subjects' performance (see subsection 5.2.3. Each function is documented separately, while the `README.txt` file provides instructions for using of the toolbox and a working example.

Listing A.1: File Wheelchair.uc – `DefaultProperties`

```
1  defaultproperties
2  {
3   UseTouchSound=False;
4   TouchSound = Sound 'NewWeaponSounds.NewGrenadeShoot'
5   // cue sound played when changing arrow direction
6   TheSound = Sound '2K4MenuSounds.Generic.msfxMouseOver'
7   RiderOffset=(X=15,Y=0,Z=0);
8   CurbClimberAngle=0;
9   CurbClimberTorque=50;
10  MountRider=True;
11  MassScale = 1.0;
12  bCollideWorld=True;
13  bBlockPlayers=True;
14  BumpRelativeLocation=False;
15
16  bDebug=false
17  StaticMesh=StaticMesh'USARSim_Vehicles_Meshes.Wheelchair.WheelchairFrame'
18  DrawScale=2.5                    //compensate for the USARSim scale 250 UU = 1m
19  DrawScale3D=(X=1.0,Y=1.0,Z=1.0)
20  AmbientGlow=128                  // how bright the chassis appears
21
22  // Number=0 indicates that this wheel is the first JointPart defined in
23  // USARBot.ini (ORDER MATTERS!)
24  // Power=Right_Powered tells USARSim to spin this wheel using the right throttle
25  Wheels(0)=(Number=0,PowerType=Right_Powered);
26  Wheels(1)=(Number=1,PowerType=Left_Powered);
27
28  // USARSim Configuration Parameters:
29  ChassisMass=63.0
30  SteerTorque=0.0100 //default: 50
31  WheelRadius=0.16    // in meters
32  Dimensions=(X=1.20,Y=0.64,Z=1.00)// X=Length, Y=Width, Z=Height in meters.
33  maxSpinSpeed=10.4167              // 10.4167 rad/sec, 6.00 km/h
34  MaxTorque=4000                    // default value, overwritten in USARBot.ini
35
36  //Tire configuration:
37  TireRollFriction=15.0
38  TireLateralFriction=15.0
39  TireRollSlip=0.06
40  TireLateralSlip=0.06
41  TireMinSlip=0.001
42  TireSlipRate=0.0005
43  TireSoftness=0.0//0.000005
44  TireAdhesion=0.0//1
45  TireRestitution=0.000000
46
47  // Karma properties:
48  Begin Object Class=KarmaParamsRBFull Name=KParams0
49     KActorGravScale=2.58              //USARSim compensation
50     bKNonSphericalInertia=True
51     KMass=124.9 //kg    (includes driver's weight)
52     //Remember to divide the tensor by mass !!
53     KInertiaTensor(0)=0.210138691
54     KInertiaTensor(1)=0
55     KInertiaTensor(2)=-0.011323948
56     KInertiaTensor(3)=0.220280388
57     KInertiaTensor(4)=0
58     KInertiaTensor(5)=0.063836029
59     // 1 KU = 50 UU = 50 cm (DrawScale applies later)
60     KCOMOffset=(X=0.125510254,Y=0.0000,Z=0.552306384)
61     KLinearDamping=0
62     KAngularDamping=0
63     KMaxAngularSpeed=100
64     KMaxSpeed=25000
65     KStartEnabled=True
66     bHighDetailOnly=False
67     bClientOnly=False
68     bKDoubleTickRate=True
69     KFriction=0.9
70     Name="KParams0"
71  End Object
72  KParams=KarmaParamsRBFull'USARBot.Wheelchair.KParams0'
73 }
```

Listing A.2: File Wheelchair.uc – `Tick` function

```
simulated function Tick(float Delta)
{
    local int ind;
    local vector ChassisX;
    local bool Touching;
    local string outstring;
    local float ChassisAngle, ArrowHorAngle, ArrowHorAngleDiff, AngleDiff;
    local float ArrowVerAngle, ArrowVerAngleDiff, ChairAngleDiff;

    Super.Tick(Delta);
    // Maintain arrow orientation
    // 32768 UU = pi rad, 65536 UU = pi/2 rad
    if (Role == ROLE_Authority) {
       ind = FindJointPartId("Dummy");
       // "Dummy" is the hidden object inside the arrow
       if ( ind > -1 ) {
          ChassisX = QuatRotateVector(KGetRBQuaternion(), vect(1, 0, 0));
          ChassisAngle = (ACos(vect(1, 0, 0) Dot ChassisX)) * 32768 / 3.14;
          if (ChassisX.y > 0)
          {
             ChassisAngle = 65536 - ChassisAngle;
          }
          ArrowHorAngle = USARRemoteBot(Controller).ArrowHorAngle;
          ArrowHorAngleDiff = ArrowHorAngle - lastArrowHorAngle;
          if (ArrowHorAngleDiff != 0) {
             lastArrowHorAngle = ArrowHorAngle;
             PlaySound(TheSound);
          }
          ChairAngleDiff = (ChassisAngle - lastChassisAngle);

          AngleDiff = ArrowHorAngleDiff - ChairAngleDiff;
          if (abs(AngleDiff) > 364)
          {
             if (AngleDiff > 32768) {
                AngleDiff -= 65536;
             } else if (AngleDiff < - 32768) {
                AngleDiff += 65536;
             }
             lastChassisAngle = ChassisAngle;
             KDHinge(Joints[ind]).KDesiredAngle +=  AngleDiff;
             KDHinge(Joints[ind]).Update();
          }
       }

       ind = FindJointPartId("Arrow");
       if ( ind > -1 ) {
          ArrowVerAngle = USARRemoteBot(Controller).ArrowVerAngle;
          ArrowVerAngleDiff = ArrowVerAngle - lastArrowVerAngle;
          if (ArrowVerAngleDiff != 0) {
             lastArrowVerAngle = ArrowVerAngle;
             KDHinge(Joints[ind]).KDesiredAngle =  ArrowVerAngle;
             KDHinge(Joints[ind]).Update();
             PlaySound(TheSound);
             log(ArrowVerAngle);
          }
       }

    }

    // Detect collisions
    SetCollision(true, true, true);
    bCollideWorld = true;
    Touching = isTouch();
    if (alreadyTouching && !Touching)    {
       alreadyTouching = false;
    } else if (!alreadyTouching && Touching) {
       outstring = "TOUCH {Time " $Level.TimeSeconds$"}{Location "
                   $converter.Str_LengthVectorFromUU(HitLocation)$"}";
       USARRemoteBot(Controller).myConnection.SendLine(outstring);
       log(outstring);
       alreadyTouching = true;
       if(UseTouchSound){
          PlaySound(TouchSound);
       }
    }
}
```

Listing A.3: File Wheelchair.uc – `isTouch` function

```
function bool isTouch()
{
    local vector HitNormal;
    local actor Bumper;
    local vector RotX, RotY, RotZ;
    local rotator curRot;
    local vector startVec, stopVec, temp;
    local int i;

    curRot = Rotation;     // current wheelchair rotation
    GetAxes(Rotation, RotX, RotY, RotZ);

    for (i=0;i<Beams.length;i++) {
        // Obtain beam object parameters for Trace function call
        startVec = Location + Beams[i].Location.X * RotX
                 + Beams[i].Location.Y * RotY + Beams[i].Location.Z * RotZ;
        temp = Normal(Beams[i].Direction.X * RotX
                 + Beams[i].Direction.Y * RotY + Beams[i].Direction.Z * RotZ);
        stopVec = startVec + Beams[i].Length * temp ;
        // Trace returns the ID of the first object on the path
        Bumper = Trace(HitLocation, HitNormal, stopVec, startVec, true);
        if (Bumper != NONE) {
            // Prepare variables for log message
            if (BumpRelativeLocation) {
               HitLocation = HitLocation - Location;
               HitLocation.X = HitLocation Dot RotX;
               HitLocation.Y = HitLocation Dot RotY;
               HitLocation.Z = HitLocation Dot RotZ;
            }
            HitName = string(Bumper);
            return true;
        }
    }
```

A.5 Mathematica files

The `Mathematica` folder contains the Mathematica® files `Kerb_analysis.nb` and `Parameters.nb` used to solve the system of algebraic equations in subsection 3.4.2 (Equation 3.14) and the quadratic equation in subsection 4.2.1 (Equation 4.5), respectively. The files are presented in Appendix C.

A.6 Textures

The `Textures` folder in the top level directory of the CD-ROM contains the original artwork files created by the author. These were used to texture objects in the different VEs built throughout the project. Each texture is provided in two formats — a vector graphics format (either Adobe Illustrator `*.ai` or Scalable Vector Graphics `*.svg` developed with Inkscape) and the Truevision TGA (TARGA) file format that can be imported into UnrealEd. For some files, also a Portable Network Graphics `*.png` is included which was used as an intermediate format between SVG and TGA (converted with GIMP). Overall, fifteen different textures are provided.

Appendix B

User Evaluation Questionnaire

B.1 Questions

The following pages contain the questionnaire administered to the subjects upon completing the training track VE. Details of the experimental procedure and subject population are discussed in section 2.6 and chapter 5.

Simulator Experience Questionnaire

Name of department:
Department of Bioengineering

Title of the study:
User Testing of an immersive virtual reality wheelchair driving simulator.

University of Strathclyde Engineering

This questionnaire consists of two parts.

In **Part A**, you will be asked about your experience using the simulator. Please complete this part in full.

In **Part B**, you will be asked some questions about your background. You can skip any question you feel uncomfortable with.

Note:
Part A of this questionnaire is the Core Module of the Game Experience Questionnaire developed by IJsselsteijn, W.A., de Kort, Y.A.W. & Poels, K in a project founded by the European Commission:

> IJsselsteijn, W.A., de Kort, Y.A.W. & Poels, K. (in preparation). The Game Experience Questionnaire: Development of a self-report measure to assess the psychological impact of digital games. Manuscript in preparation.

Please do not copy without permission from the authors.

Part A Please indicate how you felt while playing the game for each of the items, on the following scale:

not at all	slightly	moderately	fairly	extremely
0	1	2	3	4

		0	1	2	3	4
1	I felt content					
2	I felt skilful					
3	I was interested in the game's story					
4	I thought it was fun					
5	I was fully occupied with the game					
6	I felt happy					
7	It gave me a bad mood					
8	I thought about other things					
9	I found it tiresome					
10	I felt competent					
11	I thought it was hard					
12	It was aesthetically pleasing					
13	I forgot everything around me					
14	I felt good					
15	I was good at it					
16	I felt bored					
17	I felt successful					
18	I felt imaginative					
19	I felt that I could explore things					
20	I enjoyed it					
21	I was fast at reaching the game's targets					
22	I felt annoyed					
23	I felt pressured					
24	I felt irritable					
25	I lost track of time					
26	I felt challenged					
27	I found it impressive					
28	I was deeply concentrated in the game					
29	I felt frustrated					
30	It felt like a rich experience					
31	I lost connection with the outside world					
32	I felt time pressure					
33	I had to put a lot of effort into it					

Part A cont.

Do you agree with the following statements about the **keyboard** mode of control?

	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree
It was intuitive	□	□	□	□	□
It took a long time to learn	□	□	□	□	□
It was easy to use	□	□	□	□	□
It was tiring to use	□	□	□	□	□
With time, I mastered it	□	□	□	□	□
It was too sensitive	□	□	□	□	□
It was not sensitive enough	□	□	□	□	□

Other comments:

Do you agree with the following statements about the **joystick** mode of control?

	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree
It was intuitive	□	□	□	□	□
It took a long time to learn	□	□	□	□	□
It was easy to use	□	□	□	□	□
It was tiring to use	□	□	□	□	□
With time, I mastered it	□	□	□	□	□
It was too sensitive	□	□	□	□	□
It was not sensitive enough	□	□	□	□	□

Other comments:

Part A cont.

Please rate the **level of graphical** detail in the simulation:

Insufficient	Somewhat lacking	About right	Somewhat excessive	Excessive
☐	☐	☐	☐	☐

Please rate the **difficulty of the obstacle course**:

Very difficult	Somewhat difficult	Neither easy nor difficult	Somewhat easy	Very easy
☐	☐	☐	☐	☐

Please rate the **speed of the wheelchair**:

Very slow	Somewhat slow	About right	Somewhat fast	Very fast
☐	☐	☐	☐	☐

Please rate the **steering response of the wheelchair**:

Very slow	Somewhat slow	About right	Somewhat fast	Very fast
☐	☐	☐	☐	☐

Part B Background questions:

1. **YEAR** of birth :

2. Gender:

3. How would you describe your level of expertise with arcade, action and action-adventure video games (those based on quick reflexes, timing and precision):

No experience – Beginner – Intermediate – Advanced – Guru

4. How would you describe your level of experience with first-person (shooter) games – those where you “see” through the eyes of the character you control:

No experience – Beginner – Intermediate – Advanced – Guru

5. How would you describe your level of expertise with vehicle simulation video games (flight simulators, racing games, etc):

No experience – Beginner – Intermediate – Advanced – Guru

6. How would you describe you level of experience using the following devices for game/simulation control:

a) Keyboard

No experience – Beginner – Intermediate – Advanced – Guru

b) Joystick

No experience – Beginner – Intermediate – Advanced – Guru

7. How many hours a day do you use a computer?

Thank you for your time!

B.2 Responses

Figure B.1: Game Experience Questionnaire responses. Boxes represent the interquantile range, red lines — the median, and whiskers — the range of data. Outliers are indicated by red crosses.

Appendix C

Equation Derivation

The following pages contain the Mathematica® files used to aid deriving and solving algebraic equations in chapter 3 and chapter 4. The file `Kerb_analysis.nb` , entitled addresses equations 3.14, 3.15 and 3.20, while the file `Parameters.nb` offers solution to Equation 4.5. Both files are also included on the accompanying CD-ROM.

Negotiating a Kerb

Dynamic Analysis

Initial data:

```
COM = {d, h, 0};
RWC = {0, 0, 0};
rRW = 0.16;
FWC = {0.45, -0.06, 0};
```

Gravity and weight vectors :

```
g⃗ = {0, -g, 0};
W = m g⃗;
```

Prepare a rotation matrix by rotating a unit vector in z (i.e [0, 0, 1]) by an as yet undefined angle α :

```
Rotα = RotationMatrix[α, UnitVector[3, 3]]
```

$$\{\{\text{Cos}[\alpha], -\text{Sin}[\alpha], 0\}, \{\text{Sin}[\alpha], \text{Cos}[\alpha], 0\}, \{0, 0, 1\}\}$$

Calculate positions of the front wheel centre (FWC) and the centre of mass (COM) as funct

```
rFWC = Simplify[Rotα.{0.45, -0.06, 0}]
```

$$\{0.45\,\text{Cos}[\alpha] + 0.06\,\text{Sin}[\alpha],\ -0.06\,\text{Cos}[\alpha] + 0.45\,\text{Sin}[\alpha],\ 0\}$$

```
rCOM = Simplify[Rotα.{d, h, 0}]
```

$$\{d\,\text{Cos}[\alpha] - h\,\text{Sin}[\alpha],\ h\,\text{Cos}[\alpha] + d\,\text{Sin}[\alpha],\ 0\}$$

Solve the equilibrium (zero net force, zero net torque) - Equation 3.14

```
sol = FullSimplify[Solve[{{0, WFy, 0} + {0, WRy, 0} == W,
    -rCOM × {0, WRy, 0} == -(rFWC - rCOM) × {0, WFy, 0}}, {WRy, WFy}]]
```

$$\left\{\left\{\text{WRy} \to -1.\,g\,m + \frac{0. + 1.\,d\,g\,m\,\text{Cos}[\alpha] - 1.\,g\,h\,m\,\text{Sin}[\alpha]}{0.45\,\text{Cos}[\alpha] + 0.06\,\text{Sin}[\alpha]},\ \text{WFy} \to \frac{0. - 1.\,d\,g\,m\,\text{Cos}[\alpha] + 1.\,g\,h\,m\,\text{Sin}[\alpha]}{0.45\,\text{Cos}[\alpha] + 0.06\,\text{Sin}[\alpha]}\right\}\right\}$$

Which, substituting W for -mg, can be re-written as (Equation 3.20):

$$\left\{\left\{\text{WRy} \to W - \frac{d\,W\,\text{Cos}[\alpha] - W\,h\,\text{Sin}[\alpha]}{0.45`\,\text{Cos}[\alpha] + 0.06`\,\text{Sin}[\alpha]},\ \text{WFy} \to \frac{d\,W\,\text{Cos}[\alpha] - W\,h\,\text{Sin}[\alpha]}{0.45`\,\text{Cos}[\alpha] + 0.06`\,\text{Sin}[\alpha]}\right\}\right\}$$

$$\left\{\left\{\text{WRy} \to W\left(1 - \frac{d\,\text{Cos}[\alpha] - h\,\text{Sin}[\alpha]}{0.45`\,\text{Cos}[\alpha] + 0.06`\,\text{Sin}[\alpha]}\right),\ \text{WFy} \to W\,\frac{d\,\text{Cos}[\alpha] - h\,\text{Sin}[\alpha]}{0.45`\,\text{Cos}[\alpha] + 0.06`\,\text{Sin}[\alpha]}\right\}\right\}$$

To solve for the actual values, we need first to calculate the angle α as a function of kerb height hk. These can be done on vectors or from basic geometry. Both are presented below in turn.

Distance between FWC and RWC as vector and scalar:

```
dRF = FWC - RWC
```

$$\{0.45, -0.06, 0\}$$

```
ndRF = Norm[dRF]
```

```
0.453982
```

Calculate position of the FWC as function of kerb height hk :

```
rFW = {Sqrt[ndRF^2 - (dRF[[2]] + hk)^2], dRF[[2]] + hk, 0}
```

```
{Sqrt[0.2061 - (-0.06 + hk)^2], -0.06 + hk, 0}
```

To show that the two calculations lead to the same answer, let' s assume a value for hk :

```
hk = 0.1;
```

Angle α calculated from the dot product definition :

```
α = ArcCos[rFW.dRF / (Norm[rFW] Norm[dRF])]
```

```
0.220775
```

Angle α calculated from basic geometry (Equation 3.15):

```
α = ArcSin[(hk + 0.10 - 0.16) / Norm[dRF]] + ArcSin[0.06 / Norm[dRF]]
```

```
0.220775
```

Substitute the rest of the data:

```
g = 9.81;
m = 128.7;
h = 0.4025;
d = 0.1813;
```

```
Solve[{{0, WFy, 0} + {0, WRy, 0} == W,
  -rCOM × {0, WRy, 0} == -(rFWC - rCOM) × {0, WFy, 0}}, {WRy, WFy}]
```

```
{{WRy → -1014.74, WFy → -247.803}}
```

Point projected on the dome

Position exressed as a function of desired observation angles γ and δ

Define the coefficients (Equation 4.5):

```
a := 1 / (Cos[δ] Sin[γ])^2
b := 2 (EY * Tan[δ] + EZ * Cos[γ]) / Sin[γ]
c := EY^2 + EZ^2 - R^2
```

Solve the equation :

```
Solve[a * x^2 + b * x + c == 0, x]
```

$$\left\{\left\{x \to \frac{1}{2}\,\text{Cos}[\delta]^2\,\text{Sin}[\gamma]^2\left(-2\,\text{Csc}[\gamma]\,(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])-\sqrt{-4\left(\text{EY}^2+\text{EZ}^2-\text{R}^2\right)\text{Csc}[\gamma]^2\,\text{Sec}[\delta]^2+4\,\text{Csc}[\gamma]^2\,(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])^2}\right)\right\},\right.$$
$$\left.\left\{x \to \frac{1}{2}\,\text{Cos}[\delta]^2\,\text{Sin}[\gamma]^2\left(-2\,\text{Csc}[\gamma]\,(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])+\sqrt{-4\left(\text{EY}^2+\text{EZ}^2-\text{R}^2\right)\text{Csc}[\gamma]^2\,\text{Sec}[\delta]^2+4\,\text{Csc}[\gamma]^2\,(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])^2}\right)\right\}\right\}$$

Simplify first solution :

```
FullSimplify[1/2 Cos[δ]^2 Sin[γ]^2 (-2 Csc[γ] (EZ Cos[γ] + EY Tan[δ]) -
  Sqrt[-4 (EY^2 + EZ^2 - R^2) Csc[γ]^2 Sec[δ]^2 + 4 Csc[γ]^2 (EZ Cos[γ] + EY Tan[δ])^2])]
```

$$-\text{Cos}[\delta]^2\,\text{Sin}[\gamma]^2\left(\text{EZ}\,\text{Cot}[\gamma]+\text{EY}\,\text{Csc}[\gamma]\,\text{Tan}[\delta]+\sqrt{\text{Csc}[\gamma]^2\left(-\left(\text{EY}^2+\text{EZ}^2-\text{R}^2\right)\text{Sec}[\delta]^2+(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])^2\right)}\right)$$

Simplify second solution :

```
FullSimplify[1/2 Cos[δ]^2 Sin[γ]^2 (-2 Csc[γ] (EZ Cos[γ] + EY Tan[δ]) +
  Sqrt[-4 (EY^2 + EZ^2 - R^2) Csc[γ]^2 Sec[δ]^2 + 4 Csc[γ]^2 (EZ Cos[γ] + EY Tan[δ])^2])]
```

$$\text{Cos}[\delta]^2\,\text{Sin}[\gamma]^2\left(-\text{EZ}\,\text{Cot}[\gamma]-\text{EY}\,\text{Csc}[\gamma]\,\text{Tan}[\delta]+\sqrt{\text{Csc}[\gamma]^2\left(-\left(\text{EY}^2+\text{EZ}^2-\text{R}^2\right)\text{Sec}[\delta]^2+(\text{EZ}\,\text{Cos}[\gamma]+\text{EY}\,\text{Tan}[\delta])^2\right)}\right)$$

The solutions differ by a single + /- sign, which depends on the angle γ, hence (Equation 4.7 a):

```
x := Cos[δ]^2 Sin[γ]^2 (-EZ Cot[γ] - EY Csc[γ] Tan[δ] -
  Sign[γ] Sqrt[Csc[γ]^2 (-(EY^2 + EZ^2 - R^2) Sec[δ]^2 + (EZ Cos[γ] + EY Tan[δ])^2)])
```

Appendix D

Clinical Opinion

The following page presents the opinion of Dr. Sujay Galen, a practising physiotherapist with research expertise, about the virtual reality wheelchair simulator and its design requirements as received on 13 July 2011.

Design requirements for a VR wheel chair simulator

Designing a VR wheelchair simulator has multiple benefits in a clinical environment, both from a patient training and research perspective. The design requirements presented has clearly identified and defined the stakeholders, who will either be training (patients and clinical users) using he VR wheel chair simulator or developing (researchers and developers) the VR wheelchair simulator. The main end user for this VR application will be patients with severe disability and mobility issues. The user and functional requirements has clearly identified and addressed some of the clinical challenges that may be encountered during the use of this VR simulator. The design has also taken into consideration that patients may have motor and/or cognitive impairments. The provision of a graphic user interface that is simple has directly addressed this issue, making the VR simulator accessible to patients with a variety of clinical diagnoses.

The design also has the provision for the patient's performance to be monitored, while they use the simulator. This provides an excellent tool for the clinician to objectively monitor and measure the progress, patients are making in learning their wheelchair skills. The design also considers providing the patients a rich visual and auditory feedback, and the immersion aspect of the simulator provides the patient an almost 'real world' experience. This sensory input is very important in rehabilitation and in learning the wheelchair skills. The portability aspect of the VR wheelchair simulator once again from a clinical perspective is important, because if the VR environment has the capability to be used in any PC with a good spec, this opens up the possibility of home based training and also remote training by the clinician.

In conclusion the design requirements presented have addressed the clinical challenges effectively and therefore the VR wheelchair simulator has the potential to be used in different clinical conditions and in patient populations with severe disability and mobility problems.