

University of Strathclyde

**Department of Design, Manufacture &
Engineering Management**

**Information Hiding in Boundary
Representation Geometric Models**

by

Csaba Salamon

**A thesis represented in fulfilment of the
requirements for the degree of Doctor of
Philosophy**

2011

Declaration of Authenticity and Author's Rights

'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 lead 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 Regulations 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.'

Signed: *Cs. Salaman*

Date: 19/05/2011

Abstract

The increasing mobility of computer aided design (CAD) data between subcontractors and distributed manufacturing facilities is creating a need to verify provenance and protect the copyright of three-dimensional CAD models. Although watermarking technologies are well established for many types of digital media (e.g. music, image and video) no viable techniques have yet emerged for the high value solid boundary representation (B-rep) models used in 3D mechanical CAD systems.

This thesis reviews some of the established approaches to watermarking 3D CAD objects, before describing the particular challenges inherent in the nature and use of B-rep data. After discussing some of the possible methods for incorporating watermarks into three dimensional data structures, an approach is described that subtly alters a B-rep model's geometric representation to incorporate a "transparent" signature. Furthermore, this thesis introduces a watermark assessment scheme and adequate test objects to enable objective benchmarking of the proposed method. In addition, it demonstrates the embedding and retrieval of text information, and investigates the robustness of the embedded watermark after a variety of transformation and modifications have been carried out.

Test results show that the introduced parametric orientation (PO) watermark does not change the model's shape or file size, nor has any influence on the CAD model's capabilities. Considering the overall performance, PO-watermarking has shown to be a viable method for marking 3D B-rep CAD models. It works particularly well within a single CAD environment (i.e. company network) where no data is being translated into different formats, and is robust for everyday use.

Thesis Related Publications

Salamon, C., Corney, J.R. and Ritchie, J.M. (2008). Information Hiding through Variance of the Parametric Orientation Underlying a B-rep Face. In Solanki, K., Sullivan, K. and Madhow, U. (Eds.) *Information Hiding*. Lecture Notes in Computer Science 5284. Berlin: Springer, pp. 268-282. (ISBN 978-3 540-88960-1)

Salamon, C., Corney, J.R., Rea, H.J., Ritchie, J.M. and Sung, R.C.W. (2007). *Watermarking 3D-CAD Data*. Paper presented at Digital Enterprise Technology 2007 (DET '07), Bath, UK, 19-21 September. (Best Paper Award)
(ISBN 978-0-86197-141-1)

Acknowledgement

This thesis is the result of five years of work during which I have been accompanied and supported by many people. I would like to take this opportunity to thank them. First and foremost, I would like to thank my supervisor Professor Jonathan Corney for all the support and encouragement he gave me during these five years. Without his guidance and feedback this PhD would not have been possible.

I gratefully acknowledge the studentship and funding received towards my PhD research from the Engineering and Physical Sciences Research Council (EPSRC) and the Knowledge and Information Management (KIM) Grand Challenge project.

Last but not least, this work would not have been achieved without the support and understanding of my family. They have always supported me in every way I have chosen in life. Special thanks go to my wife Sandra who has supported and motivated me throughout my PhD.

Table of Contents

Declaration of Authenticity and Author's Rights	ii
Abstract	iii
Thesis Related Publications	iv
Acknowledgement.....	v
Table of Contents	vi
List of Figures	ix
List of Tables.....	xi
List of Tables.....	xi
List of Abbreviations.....	xii
1 Introduction	2
1.1 Background	2
1.2 Research Aim and Objectives	4
1.3 Thesis Structure.....	4
2 Introduction to Data Protection and Digital Watermarking	7
2.1 Digital Data Protection.....	7
2.1.1 Message Coding Techniques.....	9
2.1.1.1 Cryptography	9
2.1.1.2 Message Digest	11
2.1.2 Information Hiding Techniques	12
2.1.2.1 Steganography.....	13
2.1.2.2 Digital Watermarking.....	14
2.1.3 Other Techniques	16
2.1.3.1 Digital Fingerprinting	16
2.1.3.2 Digital Rights Management (DRM)	17
2.1.3.3 Digital Signature	18
2.1.4 Comparison of Data Protection Techniques	19
2.2 Digital Watermarking.....	22
2.2.1 History of Watermarking	22
2.2.2 Industrial Application of Digital Watermarking.....	29
2.2.3 The Process of Digital Watermarking.....	30
2.2.4 Visibility of Digital Watermarks.....	33
2.2.5 Robustness of Digital Watermarks.....	35
2.2.6 Attacks on Watermarked Data	36
2.2.6.1 General Classification of Watermark Attacks.....	37
2.2.6.2 Attacks in the CAD Environment	39
2.2.7 Requirements for CAD Data Watermarking Schemes.....	40
3 Representation of 3D-CAD Data	44
3.1 Line or Wire-frame Modelling for 3D Data Representation.....	47
3.2 Surface Modelling for 3D Data Representation.....	49
3.2.1 Polygonal Mesh Surface	52
3.2.2 Bézier Surface	56
3.2.3 Non-Uniform Rational Basis-Spline (NURBS).....	59
3.3 Solid Modelling for 3D Data representation.....	61
3.3.1 Constructive Solid Geometry.....	64
3.3.2 Boundary Representation (B-rep)	67
3.4 Other Techniques for 3D Data Representation	75

3.4.1	Voxels	75
3.4.2	Point Clouds	79
3.5	Discussion	81
4	Work Related to Digital Watermarking	88
4.1	Watermarking Approaches for Non-Three Dimensional Data	88
4.1.1	Text-Document Watermarking	88
4.1.2	Still Image Watermarking	92
4.1.2.1	Watermark Signal Design	94
4.1.2.2	Watermark Embedding	95
4.1.2.3	Watermark Recovery	96
4.1.2.4	Discussion	96
4.1.3	Video Watermarking.....	98
4.1.3.1	Space	98
4.1.3.2	Frame Rates.....	99
4.1.3.3	Data Processing.....	99
4.1.3.4	Live Editing.....	101
4.1.3.5	Video Watermarking Techniques	101
4.1.3.6	Discussion	102
4.1.4	Audio Watermarking.....	104
4.1.4.1	Time Domain	106
4.1.4.2	Compressed Domain	106
4.1.4.3	Frequency Domain	107
4.1.4.4	Other Audio Watermarking Techniques	109
4.1.4.5	Discussion	109
4.2	Watermarking Approaches for Three-Dimensional Data	111
4.2.1	Watermarking Techniques for 3D Models Represented as Surface	112
4.2.1.1	Approaches for Polygonal Mesh-type Data	112
4.2.1.2	Approaches for Bézier-type Data.....	124
4.2.1.3	Approaches for NURBS-type Data.....	125
4.2.2	Watermarking Techniques for 3D Models Represented as Solids...	126
4.2.3	Alternative 3D Watermarking Techniques	128
4.2.4	Discussion	132
4.2.5	Gap in the Literature	139
5	Methodology	142
5.1	Research Approach	142
5.1.1	Research Philosophy	142
5.1.2	Research Process	143
5.2	Knowledge & Information Management Project Survey.....	145
5.3	Requirements and Mechanisms for 3D Watermarking Systems.....	146
5.4	Method Overview of the PO-watermarking Scheme	153
5.4.1	Watermark Embedding	154
5.4.1.1	Task One: Preparation of Carrier CAD Data	155
5.4.1.2	Task Two: Text Message Encoding.....	161
5.4.1.3	Task Three: Tweaking or Embedding Process	164
5.4.2	Watermark Retrieval Process.....	168
6	Testing Procedures for the PO-watermarking Scheme	172
6.1	Lack of Common Standards for Testing Watermarked 3D Data.....	172
6.2	Possible Benchmarking Scheme for Watermarked 3D CAD Data.....	174

6.2.1	Classification of Attacks on CAD Data	174
6.2.2	Appropriate CAD Test Data	178
6.3	Modifications for Testing the PO-watermark's Robustness	180
7	Evaluation and Discussion of Test Results	185
7.1	Mathematical Considerations of the Embedded Watermark Message ...	185
7.2	Produced Test Data	189
7.3	Discussion of Test-Result Data.....	193
7.4	Discussion on the Requirements for a Watermarking Scheme.....	197
7.5	Comparison of the PO-watermarking Scheme against Existing Watermarking Methods and Techniques	202
7.6	Possible Applications of the PO-watermarking Approach	204
7.6.1	Fragile Watermarking	204
7.6.2	Robustness-enhanced Watermarking.....	206
8	Discussion and Conclusion	213
8.1	Summary	213
8.2	Contribution to Knowledge.....	216
8.3	Limitations and Directions for Future Work.....	217
	References	221
	Appendices.....	235

List of Figures

Figure 2.1 Data protection techniques	8
Figure 2.2 The process of cryptography	10
Figure 2.3 The Message Digest process.....	12
Figure 2.4 The steganography process.....	14
Figure 2.5 Example of use of digital watermarking.....	15
Figure 2.6 Fingerprinting as a unique identifying pattern	16
Figure 2.7 The process of digital signature.....	19
Figure 2.8 Egyptian symbols as information	22
Figure 2.9 Different marking patterns: thin wire patterns, stamping of paper, coloured fibre adding (from left to right).....	24
Figure 2.10 Number of publications on digital watermarking.....	27
Figure 2.11 The distribution of digital watermarking applications.....	28
Figure 2.12 The process of digital watermarking	31
Figure 2.13 Example of visible watermark for images	34
Figure 3.1 Classification of geometric modelling.....	45
Figure 3.2 The wire-frame model of a mechanical component	48
Figure 3.3 An object defined by complex surfaces.....	50
Figure 3.4 A dolphin represented by polygonal mesh	53
Figure 3.5 Mesh modifications	56
Figure 3.6 A Bézier curve (C) with its control points (1-4) and its control polygon (P)	56
Figure 3.7 Example of a Bézier surface	58
Figure 3.8 An object represented as a solid model	62
Figure 3.9 Boolean functions: union \cup , intersection \cap , difference \ominus	65
Figure 3.10 Boundary representation of a 3D model.....	67
Figure 3.11 Definition of a B-rep model's face	68
Figure 3.12 ACIS hierarchy representation of a B-rep data.....	69
Figure 3.13 Hierarchical representation of a B-rep model.....	70
Figure 3.14 Concept of parametric surface definition based on basic geometrical objects	72
Figure 3.15 Voxels can be regarded as a stack of two-dimensional images.....	76
Figure 3.16 A macromolecule in biochemistry represented by Voxels.....	77
Figure 4.1 Word-shift coding by moving words horizontally.....	90
Figure 4.2 Feature coding by stretching the upper part of the letter 'h'	90
Figure 4.3 Image watermarking for data authentication	93
Figure 4.4 Information embedding with TSPS	114
Figure 4.5 Carrier objects for information hiding.....	131
Figure 5.1 Research process adopted for this study.....	144
Figure 5.2 Flow chart of the watermark embedding process.....	155
Figure 5.3 Problems within CAD data: gap (left) and overlap (right).....	158
Figure 5.4 Creation of reorientation/tweaking parameter for two faces	163
Figure 5.5 Tweaking parameter creation process within the PO-watermarking scheme based on the word "Strathclyde".....	164
Figure 5.6 Example of surface tweaking.....	165

Figure 5.7 Iso-parameter lines showing surface orientations before (left) and after (right) tweaking.....	166
Figure 5.8 Cube containing the message “Strath” embedded by the PO-watermarking scheme.....	167
Figure 5.9 Flow chart of watermark retrieval process	170
Figure 6.1 Level 1 test objects with low geometrical complexity	179
Figure 6.2 Level 2 test objects with advanced geometrical complexity	179
Figure 6.3 Level 3 test objects with sophisticated geometrical complexity	180
Figure 6.4 Testing procedure of the PO-watermarking scheme	183
Figure 7.1 Parametric orientation of a planar face before (left) and after (after) tweaking	186
Figure 7.2 Schematic triangle representing parametric reorientation	187
Figure 7.3 Output of the PO-watermark reader	188
Figure 7.4 Watermark recovery success rate after various attacks	197
Figure 7.5 Ways of cutting objects	200
Figure 7.6 Entry of information for fragile watermarking.....	205
Figure 7.7 Information conversion and encoding for fragile watermarking.....	206
Figure 7.8 Entry of information for robust watermarking	208
Figure 7.9 Information conversion and encoding for robust watermarking	208
Figure 7.10 Message recovery matrix.....	210

List of Tables

Table 3.1 Representational ACIS ENTITIES	70
Table 3.2 Lowercase geometry classes of B-rep data (source: Corney, 1997)	73
Table 4.1 Digital watermarking techniques and methods for three dimensional data	136
Table 5.1 Comparison of potential B-rep watermark embedding mechanisms	151
Table 6.1 Classification of threats and attacks in the CAD field	175
Table 7.1 Output of the PO-watermark extractor based on Cube object	185
Table 7.2 Watermarking success rate for Level 1 test objects	190
Table 7.3 Level 1 test objects after data handling (or unintentional) attacks	191
Table 7.4 Level 1 test objects after geometrical modification (or intentional) attacks	191
Table 7.5 Level 1 watermark survival rate after data handling (or unintentional) attacks	192
Table 7.6 Level 1 watermark survival rate after geometrical modification (or intentional) attacks	192
Table 7.7 Data structure before and after surface tweaking	198
Table C0.1 Watermarking success rate for Level 2 test objects	264
Table C0.2 Level 2 test objects after data handling (or unintentional) attacks	265
Table C0.3 Level 2 test objects after geometrical modification (or intentional) attacks	265
Table C0.4 Level 2 watermark survival rate after data handling (or unintentional) attacks	266
Table C0.5 Level 2 watermark survival rate after geometrical modification (or intentional) attacks	266
Table C0.6 Watermarking success rate for Level 3 test objects	267
Table C0.7 Results for Level 3 test objects after data handling (or unintentional) attacks	267
Table C0.8 Results of Level 3 test objects after geometrical modification (or intentional) attacks	268
Table C0.9 Level 3 watermark survival rate after data handling (or unintentional) attacks	268
Table C0.10 Level 3 watermark survival rate after geometrical modification (or intentional) attacks	269

List of Abbreviations

3D	Three Dimensional
AD	Analogue/Digital
AIE	Affine Invariant Embedding
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AVI	Audio Video Interleave
BC	Before Christ
B-rep	Boundary representation
CAD	Computer Aided Design
CAM	Computer Aided manufacturing
CAT	Computer Axial Tomography
CATIA	Computer Aided Three-dimensional Interactive Application
CGI	Computer Generated Imagery
CIT	Cylindrical Integration Transform
CNC	Computer Numerical Control
CSG	Constructive Solid Geometry
DA	Digital/Analogue
DAB	Digital Audio Broadcasting
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
DNC	Direct Numerical Control
DRM	Digital Rights Management
DVD	Digital Video Disc
DWG	Drawing
DWF	Design Web Format
DWS	Drawing Standards
ECDR	Enhanced Centroid Distance Ratio
FAP	Facial Animation Parameter
FEM	Finite Element Method
FFT	Fast Fourier Transform
GPS	Global Positioning System
HAS	Human Audio Sense
HD	High Definition
HVS	Human Visual Sense
IGES	Initial Graphics Exchange Specification
IP	Intellectual Property
ISO	International Organization for Standardisation
JPEG	Joint Photographic Experts Group
KIM	Knowledge and Information Management
LSB	Last Significant Bit
MAP	Maximum a-posterior
MATLAB	Matrix Laboratory
MD	Message Digest
MLS	Moving Least Squares
MPEG4	Moving Picture Experts Group 4

MRI	Magnetic Resonance Imaging
NBE	Normal Bin Encoding
NURBS	Non-Uniform Rational Basis Spline
NTSC	National Television System Committee
OCR	Optical Character Recognition
OpenGL	Open Graphics Library
PAL	Phase Alternate Line
PC	Personal Computer
PDF	Portable Document Format
PDM	Product Data Management
PLM	Product Life-cycle Management
PN	Pseudo Noise
PO	Parametric Orientation
POA	Principal Object Axis
SECAM	Séquentiel couleur à mémoire (French for "Sequential Colour with Memory)
SPOA	Sectional Principal Object Axis
SSA	Singular Spectrum Analysis
STEP	Standard for the Exchange of Product Data
STL	Stereo Lithographic
Tiff	Tagged Image File Format
TSPS	Triangle Strip Peeling Symbol
TSQ	Triangle Similarity Quadrule
TV	Television
TVR	Tetrahedral Volume Ratio
US	United States
VFA	Vertex Flood Algorithm
Voxel	Volumetric Pixel
VRML	Virtual Reality Modelling Language
W.H.Smith	William Henry Smith

Chapter 1

1 Introduction

1.1 Background

When the Internet became wide spread in the 1990s, it quickly emerged that people wanted to advertise, buy, sell, and exchange many kinds of digital information such as text, images, music, videos, and engineering design data (Podilchuk and Delp, 2001). It also became apparent that the Internet was an excellent distribution system for digital media because it is inexpensive, eliminates the need for warehousing, and delivery is almost instantaneous.

The 1990s also saw personal computer (PC) based three-dimensional (3D) computer aided design (CAD), computer aided manufacturing (CAM) systems, data analysis (e.g. Finite Element Method) and simulation programmes becoming widely used in the engineering industry. Eventually, these separate systems were integrated to form product life-cycle management (PLM) and product data management (PDM) systems that enable efficient management of engineering data.

Many companies today have security policies and processes implemented in their PLM systems to support global collaboration. However, a PLM system can only protect information while it is inside the system. In other words, once the data is outside the physical protection ring of the intellectual property (IP) owner or the secure PLM system, there is usually nothing to prevent its intended or unintended disclosure to third parties (e.g. the press, competitors, black-market manufacturers, and other non-authorised parties).

For these reasons companies and copyright owners who exchange, or sell their products in virtual environments, face copyright-related problems and a risk of

piracy or theft. Consequently, the protection and enforcement of intellectual property rights for digital media has become an important issue. As one commentator states, "... there is typically more security around a \$0.99 iTunes song than the detailed engineering models of a discrete manufacturer's next generation product when they are outside a secure PLM system!" (PTC, 2008, p. 4). Therefore, the interest in steganography, information hiding, fingerprinting, digital rights management (DRM), and other digital watermarking technologies for three-dimensional CAD data has increased continuously since 1997.

However, 3D CAD models pose distinctly different challenges from other types of computer generated image (CGI) or music data, and to date only a small amount of academic work has been reported that addresses their unique challenges. While the spread of networks and digital multimedia materials, such as MPEG4 and VRML has also led to work on the development of digital watermarking techniques, most of them focus on the embedding of information into images, audio and video data. Only few algorithms have been developed to hide security and/or copyright relevant data within the data structures of 3D CAD models (Corsini et al., 2003a; Hartung and Kutter, 1999a). Despite the rapid evolution of dedicated hardware, and software systems to display, process, and protect three dimensional CAD models effectively, no viable watermarking technique has yet emerged for the high value Boundary representation (B-rep) models used in most mechanical CAD systems.

1.2 Research Aim and Objectives

Attempting to fill this gap, the overall aim of this thesis is to develop a digital watermarking scheme that allows the embedding of textual information into solid three-dimensional Boundary representation CAD models.

The objectives of this thesis can be summarised as follows:

1. Investigate the feasibility of B-rep data watermarking for information embedding while preserving both the topology and the exact geometrical shape of the model.
2. Develop testing procedures that are appropriate to assess watermarked data under real-world conditions (i.e. in a CAD environment)
3. Investigate the robustness and limitations of the proposed parametric orientation watermarking (PO-watermarking) approach using the newly developed testing procedures.

1.3 Thesis Structure

This thesis is presented in eight chapters. The contents of the chapters following this introductory part are as follows:

Chapter 2 provides an introduction to data protection and digital watermarking. It first presents different intellectual property protection techniques and explains why digital watermarking is the best suited approach for marking 3D CAD data. The chapter continues with an introduction to digital watermarking which includes the historical development, the watermarking process and relevant terminology. Finally, the requirements for a digital watermarking scheme for CAD related data are identified.

Chapter 3 reviews the different ways/methods of representing CAD data, their ability to store watermark information, and their use in mechanical CAD systems.

The first part of Chapter 4 describes different established technologies and methods for digital watermarking for various types of data. In the second part of this chapter, the various watermarking techniques for three dimensional data are identified, classified and examined regarding their strength and weaknesses. Finally, the gap in the literature is identified.

Chapter 5 discusses the research methodology and proposes an original watermarking approach (known as “PO-watermarking”) for three dimensional B-rep CAD data that enables the encoding and embedding of text information.

Chapter 6 analyses the need for an appropriate evaluation scheme for marked CAD data, and introduces a watermark assessment scheme and adequate test objects to enable objective benchmarking of the proposed method under real-world conditions.

In Chapter 7 the test results of the PO-watermarking method are analysed and discussed, and compared to the requirements for CAD data watermarking. Before the capabilities of the PO-watermarking approach are discussed in the context of various forms of watermarking methods, the PO-watermarking scheme is compared to other 3D watermarking schemes reported in the literature review section.

Chapter 8 summarises the key findings of this research and illustrates the study’s contribution to knowledge. It ends with a selection on the PO-watermarking scheme’s limitations and directions for future research.

Chapter 2

2 Introduction to Data Protection and Digital Watermarking

This chapter provides a general overview of data protection and digital watermarking. The initial sections review the different data protection techniques. Next, to get a better understanding of the term digital watermarking, a brief review of its historical development and terminology is provided. After that, the industrial application and the process of digital watermarking are explained and the characteristics of digital watermarks described. The chapter continues with the classification of different attack types on digital data and possible attacks on watermarked data in the CAD environment. It ends by detailing the functional requirements for a digital watermarking scheme in the context of CAD.

2.1 Digital Data Protection

The need to communicate secretly (or to transmit trustworthy information) is as old as communication itself. The first stories which can be interpreted as early records of covert, or secret, communication appeared in ancient Greek literature (*The Histories*¹) where the word “steganography”, which is still in use for copyright protection today, derives from the Greek language and means “covert communication” or “covered writing” (Katzenbeisser and Petitcolas, 2000). Today, covert communication is, of course, concerned with digital communication and the protection of digital content. As in classical tales of covert communication (e.g. invisible ink, or secret symbols) there are many different methods that allow the

¹ *The Histories* of Herodotus of Halicarnassus is considered the first work of history in Western literature. Written about 440 BC in the Ionic dialect of classical Greek, *The Histories* tells the story of the Greco-Persian Wars between the Achaemenid Empire and the Greek city-states in the 5th century BC. Herodotus tells how around 440 B.C. Histiaeus shaved the head of his most trusted slave and tattooed it with a message which disappeared after the hair had re-grown. The purpose was to investigate a revolt against the Persians (Katzenbeisser and Petitcolas, 2000).

transmission of digital information to be hidden, or embedded, into digital carrier data.

While modern digital techniques often share similar principles and methods, there are also important distinguishing features, mainly in terms of robustness against intrusions, or attacks, that are unique to the media. However, the meaning of the terminology is often confusing, and therefore it is necessary to clarify the differences and similarities between the various information hiding and data protection techniques.

The most frequently cited techniques are cryptography, message digest, steganography, digital watermarking, digital fingerprinting, digital rights management (DRM), and digital signature. Based on their functionality the various techniques can be grouped into different categories as shown in Figure 2.1.

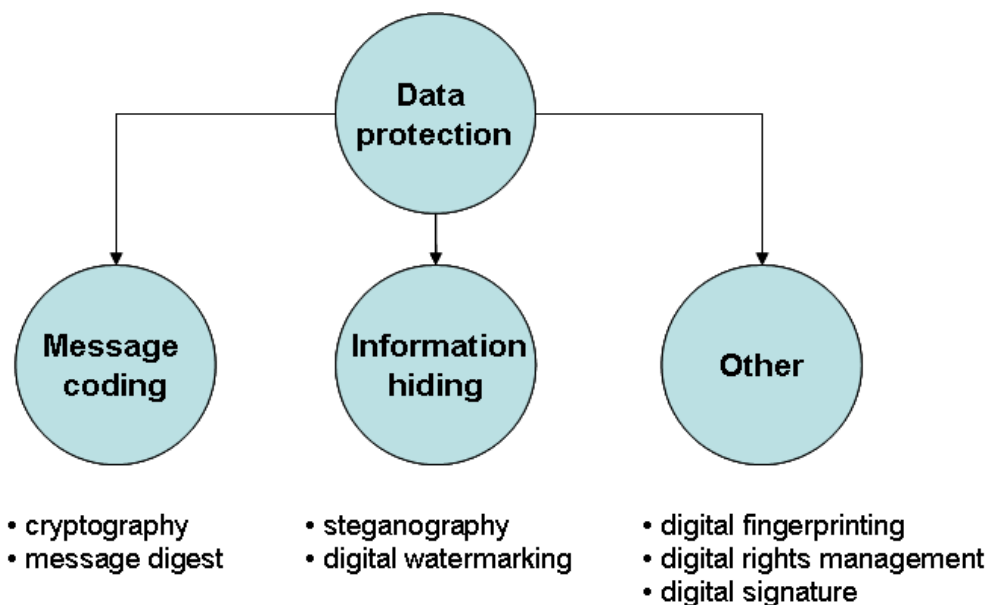


Figure 2.1 Data protection techniques

Steganography and digital watermarking are related to hiding information in the carrier data; other techniques like cryptography and message digest are concerned

with the protection of data or messages through various encoding schemes. In contrast, technologies like digital fingerprinting, digital rights management, and digital signature focus on identifying content ownership, access control and data authentication, respectively (Cox et al., 2002b).

To a certain extent, some of these techniques overlap and share many similar technical approaches. However, there are other fundamental philosophical differences between them that affect the details of their functionality, their outcome, and thus the design of technical solutions.

In general, it is appropriate to categorise these techniques since all of them have different characteristics. Therefore, in the following sections each of these seven approaches are described and their differences, or similarities, highlighted, while in the last section their applications in the context of CAD discussed.

2.1.1 Message Coding Techniques

Message coding techniques are concerned with the encoding or cipher of the signal or data that is going to be transmitted, instead of putting any form of copyright or ownership related information into the carriers signal (Light et al., 1990). Therefore, the application areas include for instance encryption of system signals (e.g. for Sky TV or DAB radio) or secure (i.e. shielded) telecommunication (e.g. police or military radio signals).

2.1.1.1 Cryptography

Cryptography (in Greek: *crypto* for “hidden” and *graphia* for “writing”) is probably the oldest and one of the most common methods for secret communication by obscuring the meaning of a message (Coyle, 2003; Kahn, 1996). Usually, the original

message that is to be encoded is called “*plaintext*”, and the information after encoding is called “*cyphertext*”. The process of transforming information is called “*encryption*” which uses an algorithm (called a cipher) to make it unreadable to anyone, except those possessing the right decoder key. *Decryption* is the process of converting encrypted data back into its original, readable format. The process of cryptography is shown in Figure 2.2 and is used as follows:

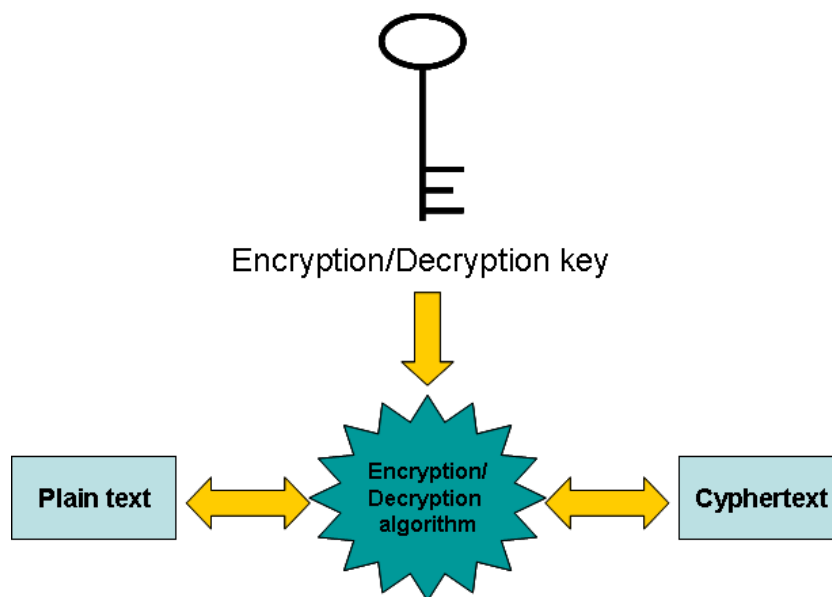


Figure 2.2 The process of cryptography
(adapted from: (Menezes et al., 2001))

The valuable digital content is encrypted prior to delivery, and a decoder decryption key (e.g. a password or device) is provided only to those who have purchased legitimate copies of the content, or are authorised to use or see the data. Then, the encoded file can be transmitted or distributed, for instance through the Internet, but would be useless to others without an appropriate decryption key.

Cryptography has long been used by armies and governments to facilitate secret communication. Nowadays, it is also used for protecting information exchange

within many kinds of civilian systems, such as computer networks, satellite telephones, wireless intercom systems, or internet banking (Bauchle et al., 2008).

Unfortunately, cryptography cannot help the seller or data owner monitor how a legitimate customer handles the content after decryption. For instance, a “pirate” can legally purchase the information and a decryption key, use the key to obtain an unprotected copy of the content, and then distribute illegal copies (Arnold et al., 2003; Kessler, 1998). In summary, cryptography can protect content in transit, but once decoded the content offers no further protection.

2.1.1.2 Message Digest

Another scheme associated with the transmission of information is the *Message Digest* (or short MD). Similar to cryptography, a message digest algorithm (also known as a hash function) is a software application that when given input data or file (for example a text message), produces an output number (or a combination of numbers and letters) that is much shorter than the original input message (Arnoud, 2005). These input messages can be very large – sometimes a complete disk file. However, the corresponding output is a much shorter code number, typically with a size of 125 or 512 bits (Menezes et al., 2001). Crucially the output number is ‘characteristic’ of the input, but does not ‘contain’ it. Because of this a message digest can verify that data has not changed, but cannot say anything about the content of the message or information. A message digest conversion algorithm has the following properties:

- 1) Given the same input file it will always generate the same output number.
- 2) The algorithm is very difficult to predict or reverse, which means that given an output number it is very difficult to derive the input that created it.

- 3) Finally, and perhaps most importantly, a slight change in the input file, say adding a comma, results in a dramatically different output number.

The process of message digest is illustrated in Figure 2.3.



Figure 2.3 The Message Digest process
(adapted from: (Menezes et al., 2001))

A text document being transmitted (e.g. emailed) to a sub-contractor could be the input message to a message digest algorithm. The output number would be sent separately to the sub-contractor who could compare it to a message digest number generated from the received file to confirm nothing has changed during transmission.

2.1.2 Information Hiding Techniques

Information hiding (or data hiding) is an umbrella term for a number of methods and technologies with various application areas to embed copyright or security related information into digital media (e.g. audio, video, and images) or CAD data (Petitcolas et al., 1999). However, in terms of data embedding into carrier objects, information hiding is closely related to digital watermarking and steganography. As Cox et al. (2002b, p. 3) state “The term hiding can refer to either making the information imperceptible (as in watermarking) or keeping the existence of the information secret”. Therefore, information hiding techniques typically apply either steganography or watermarking methods, or applications from both techniques.

2.1.2.1 Steganography

Steganography is another term derived from the Greek words *steganos* equivalent of “covered”, and *graphia*, meaning “writing” (Drake, 2003). Steganography stands for a technique that allows secret point-to-point communication between trusted parties, usually by embedding, or hiding, secret information into unsuspected carrier objects or data, in such a way that does not disturb the carrier data’s appearance or functions (Arnold et al., 2003; Cox et al., 2002b; Katzenbeisser and Petitcolas, 2000). Apart from the sender and the intended recipient (who expects to receive a message), no one should realise that there is a hidden message embedded in the carrier data. The detection of steganographically encoded information is called *steganalysis*. The simplest method to detect the secret information is to compare the carrier data to the unmarked originals. The differences (assuming the carrier is unchanged) will be the secret message. Consequently, steganographic methods, in general, have only limited ability to protect the embedded information against carrier data modifications, or other technical transformations and modifications that may occur during handling, transmission and storage (Arnold et al., 2003). In other words, the hidden information cannot be recovered after the carrier data has been manipulated, modified or attacked.

The advantage of steganography over cryptography is that the embedded message does not attract attention to itself since it is invisible or is not detectable without appropriate software. Often, steganography and cryptography are used together to ensure, and increase the security of the covert message. The process of steganography is illustrated in Figure 2.4.

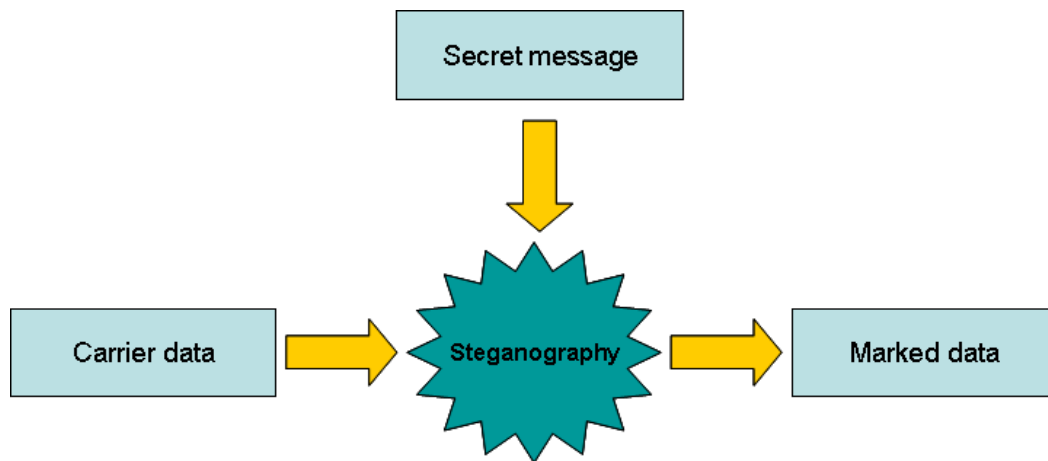


Figure 2.4 The steganography process
 (adapter from: (Menezes et al., 2001))

2.1.2.2 Digital Watermarking

Digital watermarking, similar to steganography, describes a technique that is used to imperceptibly convey information by embedding it into carrier data, whereas steganography typically relates to covert point-to-point communication between two trusted parties, with limited robustness against technical modifications. Digital watermarking, on the other hand, has the additional notion of resilience against attempts to remove the hidden information (Cox et al., 2002b). Thus, watermarking is used whenever the carrier data is exposed to third parties who know about the existence of the hidden data and may have an interest removing it. A popular application of digital watermarking is to give prove of ownership of digital data by embedding copyright statements (Katzenbeisser and Petitcolas, 2000).

Digital watermarking can be distinguished from other techniques in two important ways. First, watermarks are “inseparable” from the carrier object in which they are embedded. In other words, they should not get removed or destroyed when the carrier objects are displayed, attacked or modified. Second, watermarks undergo the same transformations as the carrier data. This means that it is sometimes possible to

learn about the transformation procedures by simply looking at the resulting watermarks.

Digital watermarking could, for example, be used to mark digital intellectual property which is meant for storage, placed on a website for visualisation, or for sale (see illustration in Figure 2.5). However, it could also be used to embed a signal that is used by a DRM system to identify an object and prove its authenticity, or to embed a copyright notice. Further applications of digital watermarking are discussed in section 2.2.2.

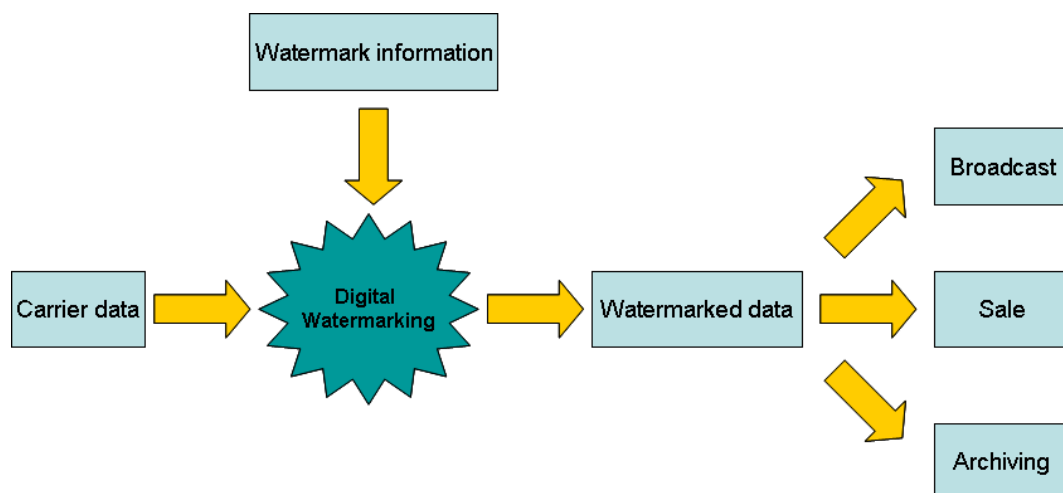


Figure 2.5 Example of use of digital watermarking

A negative aspect of digital watermarking is that typically watermarking methods embed much less information in the host data than steganographic methods because of the desire to keep robustness levels high (Arnold et al., 2003; Hartung and Kutter, 1999a). In other words, to make a watermark more robust (i.e. detectable after modifications) the same information will be embedded multiple times and distributed over the carrier object. However, this procedure takes up more storage space within the carrier data. Therefore, steganography and digital watermarking are complementary rather than competitive approaches.

2.1.3 Other Techniques

2.1.3.1 Digital Fingerprinting

The figurative term *fingerprinting* has acquired two completely different interpretations in the field of digital content protection. However, only one refers to the process of information embedding or watermarking (Hartung and Kutter, 1999a). In the watermarking interpretation a unique digital mark, consisting of information about the content owner, or the end user for instance, that identifies and authorises him to use the data, is embedded into the carrier object. These fingerprint marks are usually embedded at the time of distribution to the customer.

The second interpretation of fingerprinting involves the derivation of unique characteristics of the protected data for transaction tracking (Arnold et al., 2003). In this context, digital fingerprinting uses unique characteristics of the content as an identifying pattern to verify the data (see fingerprinting process in Figure 2.6).

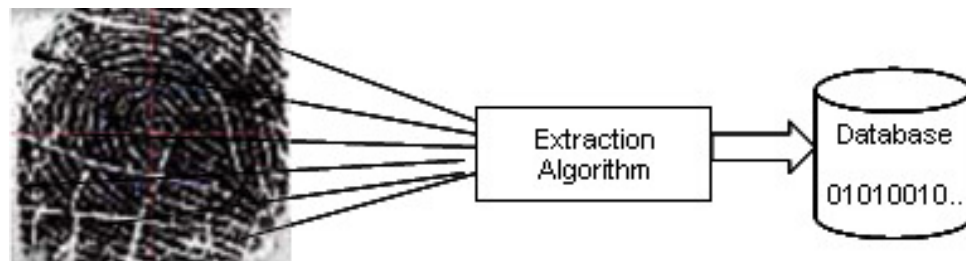


Figure 2.6 Fingerprinting as a unique identifying pattern

Similar to a message digest, these ‘unique’ fingerprints are sent separately to the recipient to verify that he/she received the correct data and the content has not been altered during transmission. This technique can offer ways of data tracking or authentication, but does not offer any protection against attacks or copying.

2.1.3.2 Digital Rights Management (DRM)

Digital Rights Management - a term commonly reduced to the acronym DRM - is a generic term that refers to access control technologies used to limit usage of digital media or devices (Arnold et al., 2003). Technologies which are involved are encryption, data embedding, watermarking, fingerprinting, steganography, transmission control, and tracking devices. Its aim is to control use of digital media by preventing access (e.g. viewing) and restricting copying, printing, sharing or conversion to other formats by end users (Slowinski, 2003; Godwin, 2006; Ceusters and Smith, 2006). Furthermore, digital rights management also covers the description, identification, trading, protection, monitoring and tracking of all forms of rights over both tangible and intangible assets, including management of relationships between rights holders in a digital environment (Felten, 2003).

While DRM is commonly used in the entertainment industry (e.g. for film and music recording and distribution), it has also found application in other situations such as online purchasing, music downloading, or office file handling. An example of office file handling is where DRM regulates the access to a PDF file and allows only viewing, but not printing, copying or saving.

However, the watermark's robustness is particularly important to data management systems (e.g. PDM and DRM) since it helps to identify the 'correct' data. In other words, should a CAD model lose the embedded watermark information after a modification attack, even the most sophisticated IP system would not be able to determine its authenticity. As a consequence of this, the purpose of intellectual property protection would fail.

2.1.3.3 *Digital Signature*

A *Digital Signature* is essentially a combination of two data protection techniques: message digest and digital watermarking. The term digital signature is used for marking, or signing, electronic documents with the help of an encryption-key such as message digest. In other words, an encrypted code (e.g. a message digest code) is attached to, or embedded into, carrier data that uniquely identifies the sender and/or the content of the message, and insures that the document has not been altered during transmission (Youd, 1996; Cox et al., 2002b).

The process of creating a digital signature has two main steps: first, creation and embedding of the digital signature before sending out the data; and second, verification of the digital content by the receiver (see process in Figure 2.7). In step one, a private-key hash function (a mapping of a variable-length string into a fixed-length string) creates a digital signature (e.g. messages digest) which is the encrypted summary of the original document and details about the content owner or sender. The resulting computed value, representing the digital signature, is then embedded into the carrier data with the help of a watermarking scheme.

In step two, the verification stage, a function reconverts the digital signature and checks its content against the original-carrier message. If the calculated message and sender details do not match the original information, the document loses its authenticity.

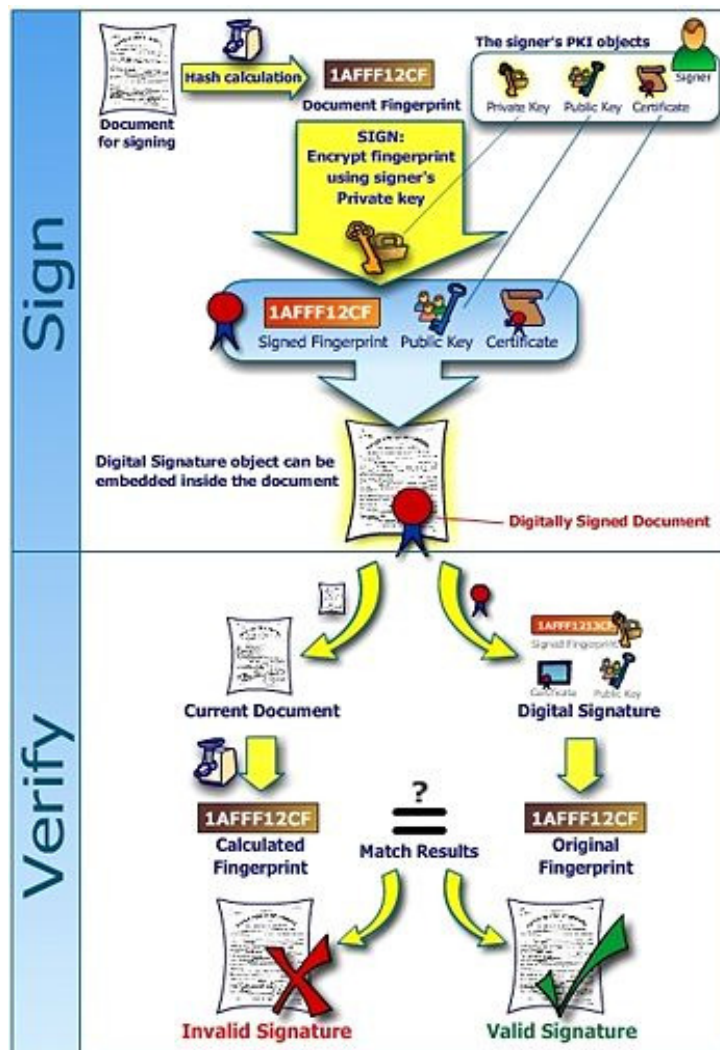


Figure 2.7 The process of digital signature (adapted from: (Merkle and Brassard, 1990))

2.1.4 Comparison of Data Protection Techniques

Most of the listed techniques for information hiding and data protection are suitable for message encoding, information embedding, secure data transmission, data tracking or access regulation. While cryptography and message digest are rather useful for signal or text message encoding, other technologies like steganography and digital watermarking are concerned with the embedding of information into carrier data. Digital rights management is more suitable for access regulation and

restriction, whereas digital fingerprinting and digital signature are more appropriate for data authentication (Arnold et al., 2003; Katzenbeisser and Petitcolas, 2000).

It should be mentioned that these technologies are not restricted to one application area only, since their capabilities and characteristics overlap. Take for example digital fingerprinting: on one hand, it can add identifying features about the owner to the carrier data and is thus similar to digital watermarking; on the other hand it is also able to extract characteristics of an object as identifying features for data verification, a process which is similar to digital signature. The digital signature itself also demonstrates multiple characteristics: the encryption of text document content for verification makes it similar to message digest, and the embedding of this message digest into the carrier object relates it to digital watermarking.

It is also worth looking at the advantages and disadvantages of these techniques. Most encryption techniques are dealing more or less with text messages for secret communication or secure information exchange. However, the biggest disadvantage of cryptography and message digest is that, once the message has been decoded, the content has no further protection against misuse, and does not carry any information about the copyright or content owner. Therefore, we could say that cryptography and message digest can protect the confidentiality of messages, or data, during transmission, but other techniques are still needed to protect the integrity and authenticity of the decrypted message.

In the case of digital signatures and digital fingerprinting, the added message to digital data can only prove the data's originality but cannot do anything about improper use (e.g. copying, modification, etc.).

Digital rights management, on the other hand, can unify some of these protecting characteristics (for example secure data transmission, access regulation, data authentication), but is useless when the embedded security features within the carrier data are lost after heavy data modifications or attacks (e.g. data processing or editing).

Consequently, there is a need for alternative, or complementary, technologies which enable robust embedding of information into valuable digital content (e.g. precise CAD data). Crucially, the information that is hidden inside the carrier data has to be able to undergo all the modifications, transformations and attacks, commonly applied to the media, without getting destroyed. Steganography and digital watermarking have the potential to fulfil this need by embedding information invisibly inside a carrier CAD model, which is not removed or destroyed during normal data usage, transmission or editing. Since the embedded information is invisible, the strength of these techniques is their inconspicuous nature. In other words, third parties who can neither see nor have any knowledge about the embedded 'features' have therefore no incentive to remove them.

A positive feature of digital watermarking, compared to steganography, is that watermarks do not get removed or destroyed during data modification and processing since they are robustly connected to the carrier object. This is crucial, because CAD models are constantly edited, modified or transformed in the engineering world.

Therefore, the most important aspect in the field of data protection and information hiding is the ability to embed a watermark message into the carrier data robustly and invisible so that it can travel with the carrier object even if it has been modified or transformed.

Since only digital watermarking meets all these requirements, this thesis focuses on the development of a digital watermarking scheme suitable for hiding text information into three dimensional CAD data.

2.2 Digital Watermarking

2.2.1 History of Watermarking

The principles of watermarking are not a modern invention. The roots of information hiding technology can be traced some 2500 years back to Greece (Katzenbeisser and Petitcolas, 2000). At that time, specific arrangements of symbols and drawings were used to forward information, for instance, at war times (see example in Figure 2.8).



Figure 2.8 Egyptian symbols as information
(source: Wikipedia, 2008c)

At around 440 BC the ancient Greek used different ways of carrying or sending combat related messages during the Greco-Persian wars. For instance, the information was scratched into a drawing board and covered by a thin wax layer, or even tattooed onto the messengers scalp (Katzenbeisser and Petitcolas, 2000). By

removing the wax layer from the board, or shaving the messengers head, the information was made visible again.

However, the basic principles of watermarking as we understand it today started with the papermaking in Europe in the 13th and 14th century. Although the method of papermaking was invented in China over a thousand years earlier, paper watermarks did not appear until the end of the 13th century (Cox et al., 2002b). The oldest watermarked paper found in archives dates back to 1282 and has its origin in the town of Fabriano in Italy (Katzenbeisser and Petitcolas, 2000). This town has played a major role in the evolution of the papermaking industry. At the end of the 13th century, about 40 paper mills shared the paper market in Fabriano and were producing paper with different format, quality, and price. Competition was not only between the 40 paper mills but also between artisans and merchants. It was difficult for any party to keep track of a paper's provenance and thus identification of format and quality was problematic. The introduction of identifying patterns or marks was the perfect method to eliminate any possibility of confusion.

The marks were made by adding thin wire patterns to the paper moulds, or by stamping the paper. When the water drained from the paper pulp, the layer of fibres over the raised wire pattern in the mould was thinner than the rest of the sheet. When the sheet was pressed and dried, the thinner areas resulted in a pattern that only shows clearly when held up to light. In the case of the stamping method, the opacity of the paper was altered by physically stamping with an identifying pattern. This created an uneven surface whose distinctive mark or logo differentiated the paper type. The same effect was also achieved by adding thin coloured textile fibres to the paper pulp to create a unique identification pattern.

The marks have been used as trademarks to identify the paper maker, or for practical functions such as identifying the moulds on which sheets of papers were made. Beyond their purely functional use watermarks may also have served as decoration or paper design. See for example the three different paper marks in Figure 2.9.

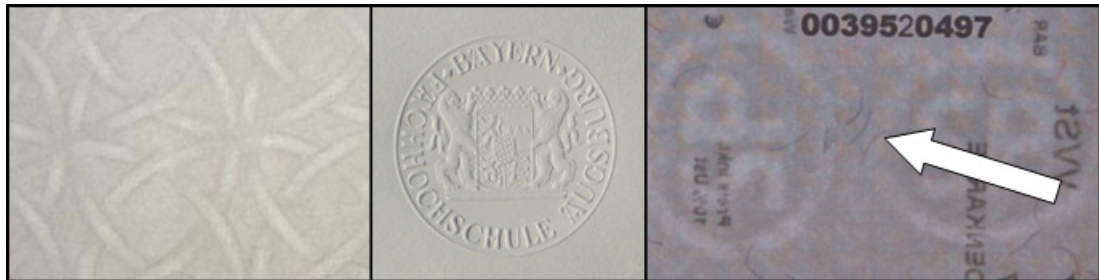


Figure 2.9 Different marking patterns: thin wire patterns, stamping of paper, coloured fibre adding (from left to right)

By the eighteenth century, these so called watermarks on paper had become more clearly functional and practical in Europe and America. They were used as trademarks to record the date the paper was manufactured and to indicate the sizes of original sheets. It was also about this time that watermarks began to be used as anti-counterfeiting measures on bank notes and other important documents (Cox et al., 2002b).

In 1818, an English inventor named Sir William Congreve devised a technique for making coloured watermarks by inserting a thin tinted material sheet between two white paper sheets during papermaking. The resulting marks were so difficult to forge, that even the Bank of England itself declined to use his method for money printing with the explanation that they were too difficult and costly to manufacture (Hunter, 1978).

In 1848, a more practical technology was invented by a fellow Englishman named William Henry Smith (W. H. Smith). His device for light-and-shade watermarking

could produce any degree of intensity or light, and made it possible to vary the thickness of a sheet of paper to show any form, regardless of how complicated the shading might be (Drake, 2003). This is actually the basic technique used today to incorporate historic faces, buildings and figures as watermarks in currency notes.

The term “watermark” seems to have appeared towards the end of the eighteenth century and may have been derived from the German term ‘*wassermarke*’ (Simpson and Weiner, 2000). However, the term is actually a misnomer, because water is not important or necessary in the creation of the mark, so the name was probably adopted because the mark looked like the effects of water on paper.

The analogy between paper watermarks and digital watermarking is obvious: paper watermarks on bank notes, stamps or important government documents for anti-counterfeiting measures inspired the first use of the term watermark in the context of protecting digital data.

It is difficult to determine when “digital watermarking” was first discussed. Electronic watermarking or digital watermarking can be traced back as far as 1954. The first example of a technology similar to digital watermarking was protected by patent in 1954 by an American researcher named Emil Frank Hembrooke of the Muzac Corporation (Cox and Miller, 2002a). In his patent he described a method for imperceptibly embedding an identification code (e.g. sound or noise) into music for the purpose of proving ownership. In 1979, another researcher named W. Szepanski described a machine-detectable pattern that could be placed on electronic documents for anti-counterfeiting purposes. About nine years later in 1988, a group of researchers around Lori L. Holt described a method for embedding an identification code (signal) into audio data. However, Komatsu and Tominaga were the first to use

the term “digital watermarking” in 1988, while Tirkel et al. (1993) were one of the first to publish on the subject with their paper “Electronic Water Mark” in the field of electrical content protection.

The emergence of the Internet around 1990, led to an increased interest in digital watermarking. This was motivated by copyright concerns that became acute with advances in computer technology and the development of eCommerces. These technologies enabled the perfect copying and instant distribution of copyrighted material to anywhere in the world at almost no cost. Since then, digital watermarking has gained a lot of attention and has experienced rapid growth. The rising interest in this topic can be seen in Figure 2.10, which shows the number of publications for digital watermarking each year since 1990.

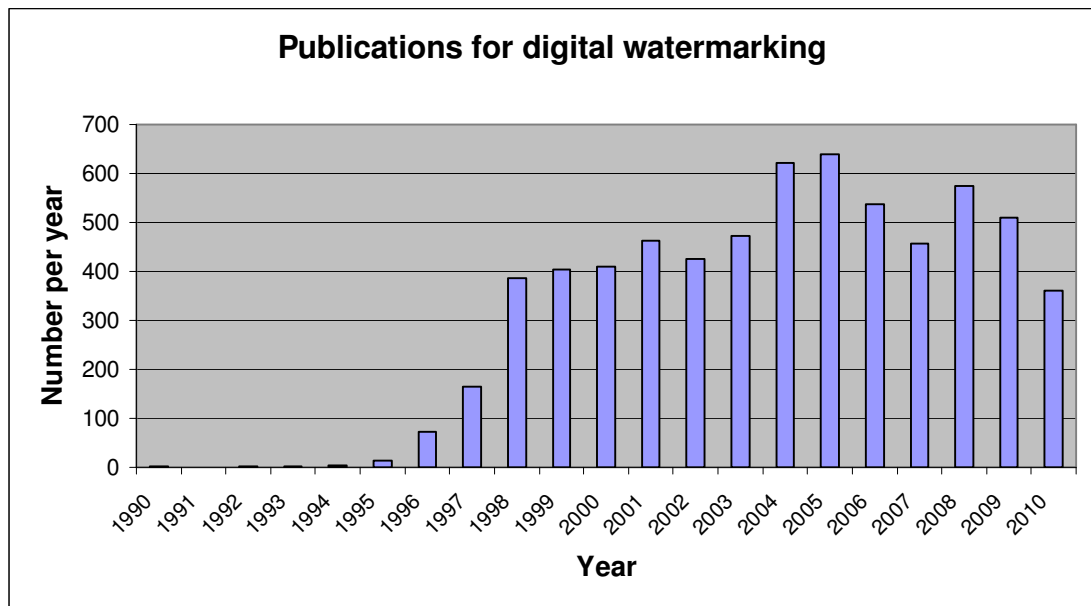


Figure 2.10 Number of publications on digital watermarking
(source: Achilles and Ortyl, 2011; Roche and Dugelay, 1998)

Compared to the early 1990s, the figures exploded in the past decade. Further evidence of the high demand of digital content protection systems was provided on the MultiMedia Intelligence website in January 2008, which reported: “applications leveraging content identification technologies such as digital watermarking and fingerprinting are growing rapidly and could surpass US\$500 million worldwide by 2012” (MultiMedia Intelligence, 2008). Key applications include Internet and broadcast content monitoring, metadata association, copyright control, content protection and forensics, and interactive advertising” (1888, 2008).

Since its origin, most of the research in digital watermarking has focused on digital media such as text documents, audio, still images (bitmaps), or video data. As a result, data embedding techniques for these ‘traditional’ digital content data types has been widely studied. The most effort, however, has been made in the development of

image watermarking followed by video, audio and finally text document watermarking (see distribution for different media types in Figure 2.11).

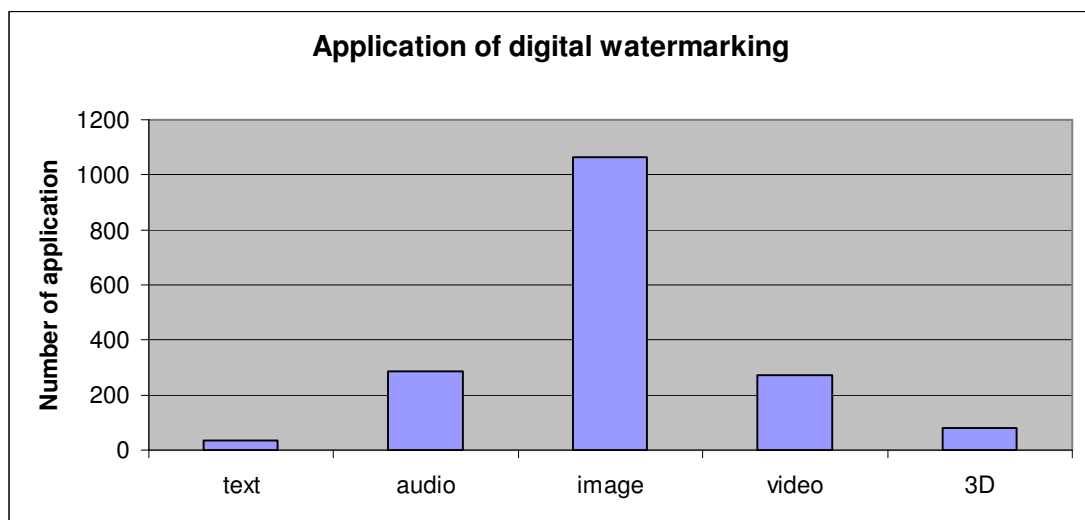


Figure 2.11 The distribution of digital watermarking applications
(source: Achilles and Ortyl, 2011)

In the mid 1990s, when cheap 3D-CAD systems appeared on the market and interest in protection of three dimensional design data started to grow, research began to appear in both academia and industry. However, the field of 3D watermarking is still relatively new, and only a few digital watermarking methods have been developed and published to date. Nevertheless, the number of watermarking techniques for 3D geometric data is steadily increasing, as people realise the importance of protecting this kind of data.

Since this thesis focuses on watermarking three dimensional CAD data, chapter 4 first provides a brief overview of traditional watermarking techniques, and then presents in more detail proposed and current 3D watermarking schemes.

2.2.2 Industrial Application of Digital Watermarking

Digital Watermarking has been considered for many verification, prevention and protection applications. For instance, in copy prevention the watermark may be used to inform hardware, or software, devices that copying should be restricted or forbidden. In copyright protection applications a digital watermark may be used to identify the copyright holder, or might ask the user to apply for proper payment of charges/fees before using the acquired product.

Although copy prevention and copyright protection have been major driving forces behind research in the watermarking field, there are a number of other applications for which digital watermarking has been suggested or could be used (Bender et al., 1996; Berghel and O'Gorman, 1997; Ohbuchi et al., 1997a; Matheson et al., 1998). These applications include (i) proof of ownership, (ii) content authentication, (iii) data tracking, and (iv) authorisation for viewing and access.

- i. *Proof of Ownership:* It is tempting to use watermarks not only for copyright protection but also to prove content ownership. Nowadays, this is usually done in form of a textual notice (e.g. notification of ownership). However, the document can be forged or might get lost during transportation, delivery or transmission. If the data is provided with a digital watermark, the recipient of marked data supplied with watermark detector might be able to identify the owner of the work, even after the data has been modified.
- ii. *Content Authentication:* Advances in computer technologies (e.g. CAD translation and modification tools) have made it easier to tamper with digital works in ways that are difficult to detect. The use of digital watermarks makes sure that the authentication-signature stays with the carrier data at all times. Of

course, care must be taken to ensure that through distribution and transmission the mark will not be removed or destroyed, or the watermarked data does not change the carrier object's shape so that it appears invalid or corrupt when compared with the original data.

- iii. *Data tracking*: Digital watermarking techniques could also be used for tracking design data within a homogeneous environment (e.g. a CAD environment within a company), or to search a corporate Intranet for "lost" design data. In this context an embedded watermark containing specific information about an object, or company/department related information (e.g. details about design engineer), might be used to track down or to locate data, even if its original identifying name, part number or optical appearance has been erased or changed, retrospectively.
- iv. *Access regulation and/or additional information provider*: Embedded digital watermarks might also be used as access regulation tool. They could permit the operator at the machine to open a file, but not to modify anything. For instance, an operator at a CNC machine is allowed to view drawings or get dimensions of it, but is not allowed to modify the model and save changes. Alternatively, a watermark (carrying a text message) could give the machine operator additional information about which fixtures and tools to use for machining the part.

2.2.3 The Process of Digital Watermarking

The term digital watermarking can be described as "the process by which identifying, copyright or security related information, called a watermark, is embedded into digital contents in such a way that they do not interfere with the content's intended

use (e.g. viewing or simulation)” (Ohbuchi et al., 1997a, p. 8). A digital watermark that is “imperceptible to the human senses yet easily recognized by special software detectors, gives content a unique identity that remains constant even after data modification or manipulation without affecting the quality or the intended use of the digital work” (Matheson et al., 1998, p. 229).

Nearly all digital watermarking methods share the same generic building elements: a watermark *embedding* system and a watermark *extraction* or *retrieval* system (often just called watermark recovery or watermark detection and decoding). The object or carrier data in which the information is embedded is called the *cover*, and the object carrying the watermark is called *stego*. This terminology follows the description recommendations of Pfitzmann (Pfitzmann, 1996). Figure 2.12 schematically illustrates the concept of the general digital watermarking process.

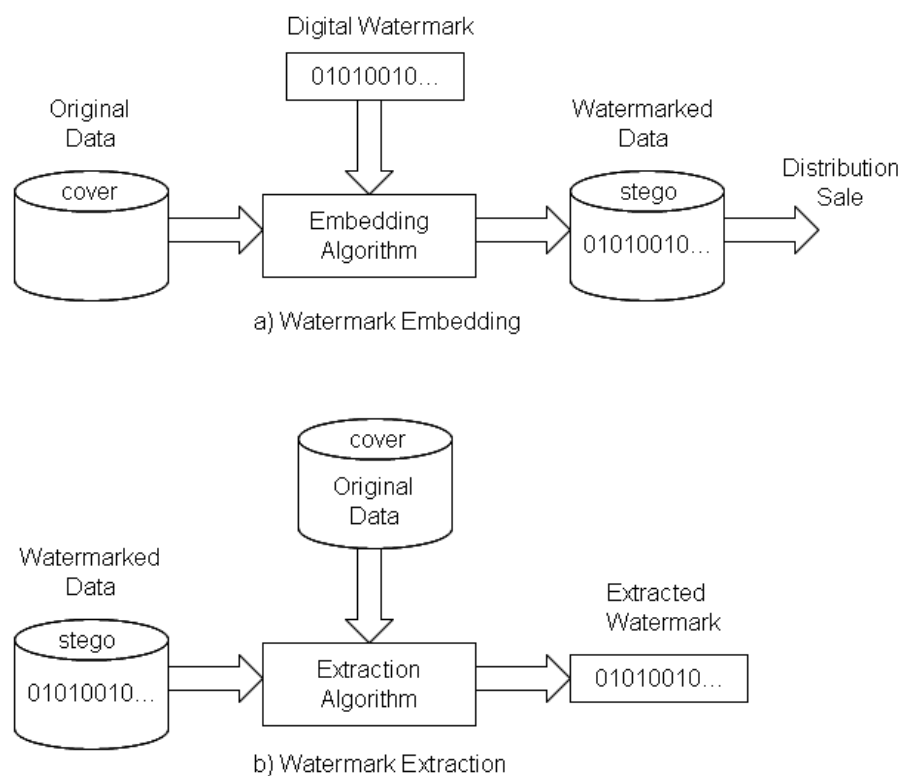


Figure 2.12 The process of digital watermarking

The input into the watermarking scheme is the original, or cover-data, and the watermark information. To embed, or hide, the watermark an embedding algorithm is used. The watermark message/information can be of any nature such as numbers, text, symbols, or simply random binary information. The output of the watermarking scheme is the watermarked data, or the, so called, stego.

Watermarking techniques are usually defined in terms of the method by which the embedded information is extracted from the carrier stego data. Therefore, the different watermarking schemes can be described as follows:

- *Blind* watermarking scheme: The detection of watermarks happens without any knowledge of the original, or unmarked data and/or the watermark message itself. In other words, neither the embedded watermark information, nor the cover data is necessary to locate and to retrieve the embedded message.
- *Non-blind* watermarking scheme: A non-blind watermarking system requires the original unmarked data in the retrieval stage. This type of system extracts the watermark information from the carrier stego data by using the original cover data to find (i.e. to locate) where the watermark is embedded.

Since both watermarking schemes use different approaches in the detection stage, they can be used for different applications. Non-blind watermarking methods are usually applied within a secure, or controlled, environment (e.g. within a company) where the original unmarked data is immediately available for verification purposes. Digital data containing blind watermarks, however, can be used everywhere (even on the Internet) since they carry all necessary information to locate and to retrieve the embedded message.

In the watermarking literature, systems that use informed or non-blind detection are often called *private watermarking* systems, whereas those that use blind detection are described as *public watermarking* systems. This terminology refers to the general usefulness of the systems in applications in which only a selected group of people are allowed to detect the watermark (private watermarking), or applications in which everyone is allowed to detect the watermark (public watermarking) (Cox et al., 2002b). In general, the original unmarked data is only available in private application, and therefore informed detectors might not be used in public applications.

2.2.4 Visibility of Digital Watermarks

Usually, the general principle of a digital watermark is to be invisible to human senses but there is an exception found in the protection of digital images. Such visible watermarks are used in quite the same way as their bond paper ancestors, with the intention to be seen, so that the source, or the authenticity, of the object can be recognised immediately on inspection. The mark is designed to be easily read or seen by the viewer and may spoil the visibility/appearance of the protected data. An example of a visible watermark is shown in Figure 2.13 below.



Figure 2.13 Example of visible watermark for images
(source: Autobil, 2009)

For invisible watermarking, it is important that the hidden message cannot be detected by human senses or by any kind of casual inspection, so that it cannot be attacked and removed. At the same time however, it should still be possible to detect and recover the watermark but only using appropriate software/tooling.

Visible and invisible watermarks are suitable for several purposes. Visible watermarks, such as company logos or copyright notice, can be used for an immediate claim of ownership, to uncover faked items, or to obtain information about the copyright owner (Berghel and O'Gorman, 1997; Ohbuchi et al., 1997b).

Invisible watermarks can have the same purpose as visible watermarks (e.g. carrying copyright related information, storing, or archiving product related data), but can also be used for preventing illegal use, or catching a thief, rather than discouraging the theft in the first place. For instance, security related information embedded into multimedia data (e.g. a computer game) could be used to track down unauthorised users, when they use the illegally acquired programme/game for Internet-gaming.

2.2.5 Robustness of Digital Watermarks

Watermarks can also be classified as robust and fragile. Robustness is the quality of being able to withstand modifications of any type or changes in procedure or circumstance. In other words, it is a capability to cope with variations (sometimes unpredictable) in its operating environment with minimal damage, alteration or loss of functionality. In the context of digital watermarking, robustness refers to the ability to detect and recover the embedded information after modification operations (so called attacks) have been carried out on the carrier stego data (Cox et al., 2002b).

Fragile watermarks are watermarks that are mainly imperceptible and have only a limited robustness to attacks. According to Chou and Tseng (2006) they are applied to detect unauthorised modifications on the protected object, or to verify data authenticity, rather than conveying un-erasable information. In other words, by modifying or changing the stego data, a fragile watermark breaks (i.e. it is not possible to recover/recreate the watermark message) and the carrier data loses its authenticity. Ideally, a fragile watermark might even help to reveal how the original data has been distorted or changed, or what processing it has undergone.

Independent of the application purpose, the robustness of watermarks is also based on the following aspects (Katzenbeisser and Petitcolas, 2000):

- *The amount of embedded information:* This is an important parameter since it directly influences the watermark robustness. The more information (i.e. bytes) one wants to embed, the lower the watermark's robustness. In other words, it is more likely after an attack to recover and restore a number of small information fragments that have been embedded several times and distributed all over the

carrier object than one single large block of information, which has been embedded only once.

- *The size of the cover data:* The size and complexity of the cover data usually have a direct impact on the robustness of the embedded watermark. For example, a simple three dimensional cube (with six faces) does not offer many possibilities to hide a watermark, whereas a complex structure with hundreds of faces does. In other words, the more complex a CAD model is (i.e. the more faces it has), the more places are available to hide the information, and consequently the likelihood of the watermark being detected is similarly smaller.

However, the requirements for watermarks to be both robust and invisible at the same time often contradict each other. Making a watermark more robust to attacks tends to make it more visible and detectable.

2.2.6 Attacks on Watermarked Data

“An attack can be described as any kind of processing that circumvents the intended purpose of the watermarking technique for a given application” (Arnold et al., 2003, p. 149). According to this definition, watermark attacks include normal data processing operations like zip compression, data editing, or file format conversion, which may happen in day-to-day operations, and could also unintentionally destroy the watermark. On the other hand, there are also some other types of attacks that are specifically aimed to disturb or even to destroy the mark’s robustness, which in turn disables the watermark’s purpose.

The following section describes different attack types and where they might occur. The subsequent section identifies possible instances of these attack types in the context of CAD.

2.2.6.1 General Classification of Watermark Attacks

In order to easily identify attacks, a classification of attack types into several groups helps both a watermark developer and the potential user of the watermarking system identify the security requirements, as well as judge the usability of the watermarking technology. The different types of attacks can be summarised as follows (Hartung and Kutter, 1999a; Cox et al., 2002b; Arnold et al., 2003; Katzenbeisser and Petitcolas, 2000):

- *Simple attacks:* These types of attacks are conceptually simple in nature and attempt to impair the embedded watermark by manipulations of the watermarked carrier data. Examples include data compression and decompression for creating small file sizes, cropping, addition of an offset, or translation into other file formats (e.g. JPEG into Tiff).
- *Detection-disabling attacks:* Detection disabling attacks aim to break the correlation between the carrier object and the watermark to make a recovery of the watermark impossible for the watermark detector in the retrieval stage. Actions could include modification or editing of the carrier data.
- *Ambiguity attacks:* These types of attacks are also called ‘over-marking’, or ‘fake watermark attacks’. In this case, a second or a third watermark is embedded into an already marked data to confuse the authority of the first watermark. In the detection stage several marks are recovered but it cannot be determined which watermark is the ‘original’.

- *Removal attacks:* Removal attacks are attacks that attempt to erase the watermark from the marked objects with ‘brute force’ instead of the correct retrieval algorithm.
- *Collusion attacks:* Having access to several watermarked copies of the same data but with different watermarks, or different data but with the same watermark, offers the possibility to check, compare and to estimate the watermarks position to deploy removal attacks to obtain an unrestricted version of the carrier data.
- *Oracle attacks:* An oracle attack is where an attacker abuses a facility (e.g. Adobe Photoshop tool) provided to gain access to protected information. Since the Adobe Photoshop tool acts as a watermark detector (i.e. to visualise the incorporated watermark logo like in Figure 2.13) one can apply small changes to the image (e.g. repeated blurring, optical whitening, or replacing progressively pixels by gray) until the detector cannot find or clearly identify the watermark anymore. In this way the attacker can get a slightly changed but unprotected version of the image (Murdoch, 2006).
- *Copy attack:* The aim of the copy attack is to copy the watermark information from an approved data set to an unauthenticated data set. This attack is basically performed in two steps. First, an estimation of the watermark from the marked data is calculated. Second, the estimated watermark information is copied from the marked data set to the target carrier data set to obtain an ‘approved’ version.

It should be noted that the different types of attacks are not always distinct and they sometimes overlap. In other words, some attacks do not clearly belong to only one group. For instance, collusion attacks could be argued to be a group of their own, since they require, unlike other attack types, more than one differently watermarked

copy of the data. However, since they attempt to reconstruct a copy of an un-watermarked original host data set, and thus remove the watermark, this type of attack could be also classified as a ‘removal attack’.

Another example is when a design engineer revises a CAD model: this can be considered as a detection disabling attack, but also as a simple attack, since geometric modelling procedures (such as Boolean operations) are part of the daily routine.

2.2.6.2 Attacks in the CAD Environment

Attacks in the CAD environment can be similarly classified into different groups.

The general attacks on 3D CAD data can be described as follows:

- *File format conversion*: CAD models can be easily imported, exported, stored or converted into various formats such as DWF, DWS, IGES, STEP or STL by using various CAD tools. Within these conversion processes not only do parts of the CAD model get “lost”, but so does any attached watermark information. Therefore, a digital watermark needs to be robust against various format conversions.
- *Geometric modifications*: Within the various CAD systems 3D objects can be geometrically edited (e.g. cropping, rotation, scaling, trimming, extending, chamfering, joining) to modify, or update, a model’s shape. A digital watermark must be robust enough to withstand geometric attacks in order to preserve the quality of the original design.
- *Surface modification*: In particular for 3D surface objects, represented as a polygonal mesh, there are a number of functions (e.g. mesh simplification, remeshing, retriangulation, or mesh smoothing) that change the models

appearance slightly to either simplify the objects internal structure or the overall file size. These changes would massively complicate (or even prevent) the recovery of the embedded watermark.

- *Boolean operations*: Shear or cutting a model in half, removing or inserting of faces from/into a CAD object would not only destroy the model's original shape (and with it the embedded watermark) but would also make the detection of the (remaining) embedded watermark very difficult (if it still exists). Therefore, watermarks need to be evenly distributed throughout the model to make a recovery possible.
- *CAD data repair*: Object or surface healing and data repair functions are meant to eliminate overlaps and close "gaps" in the CAD model's structure. These 'healing' functions would also eliminate distortions that were specifically created to "host" the watermark information when embedding the watermark message.

All of these 'attack' types might have some effects on the robustness of the embedded watermarks in CAD data. Therefore, it is important to identify these possible threats and should be considered when creating new watermarking systems or defining watermark requirements.

2.2.7 Requirements for CAD Data Watermarking Schemes

The previous sections have shown several points to be relevant when determining the requirements for a watermarking scheme and the watermark itself. One of the most important requirements of a digital watermark is transparency. As Ohbuchi and Masuda (2000, p. 105) state, "Watermarks must be transparent to the intended applications". However, there are two kinds of transparency: functional and

perceptual transparency. For most of the traditional data types, such as image, audio and video, transparency of a watermark is only judged by human senses (human visual sense (HVS) and human audio sense (HAS)). If the original and watermarked data are indistinguishable to human observers, then the watermark is perceptually transparent.

For other data types, such as 3D geometric CAD data, transparency of the watermark is judged by considering if the functionality (and of course appearance) of the model is altered or not. For example, a perceptually transparent watermark incorporated into the CAD data of an engine cylinder may alter the shape of the cylinder enough to interfere with the simulated functionality of the engine (e.g. overlapping clashing of faces), or might distort the results in FEM simulation.

To be effective in the protection of the ownership of CAD data, digital watermarking schemes have to fulfil further requirements as stated by numerous authors (Matheson et al., 1998; Berghel and O'Gorman, 1997; Ohbuchi et al., 1997a):

- A watermark must be difficult or impossible to remove; at least it should cause a degradation of the content when trying to erase the watermark.
- A watermark must survive modification processes that are common to content processing (in the case of CAD: scaling, translation, editing, transformation, compression, copying, etc.).
- The actual bits representing the watermark should be spread evenly throughout the CAD file and be difficult to identify them as the watermark.
- Watermarks should be readily detectable by proper authorities, even if imperceptible to the average observer/viewer.

- Watermarks should have a robust connection to the CAD model and adaptability to different systems (e.g. CAM, simulation, analysis, calculation) at the same time.
- And finally, the embedded watermark should not influence, or interfere, with the capabilities and functions of the CAD data (i.e. no change in shape).

Establishing how closely any given scheme meets these requirements is an “open” question. No standards or benchmarking exist, and consequently, the development of credible assessment method is just as important as the “invention” of new watermarking methods.

Chapter 3

3 Representation of 3D-CAD Data

The following chapter describes the nature of CAD data by discussing the different types of 3D geometric shape representation and their usability for CAD applications. In the discussion the various object presentation formats are compared with each other, and the most appropriate presentation type for solid CAD data watermarking is determined.

In designing products such as structures, mechanical components or electrical devices, shape is one of the most important pieces of information needed to represent the product. Knowledge of an object's shape allows systems to determine possible interferences of related components, and calculate or simulate the models capabilities (Rodriguez, 1992). So engineers use geometric modelling to describe and visualise the physical shape of components created in CAD systems.

Since geometric models have the potential to completely define the exterior and interior of a physical object, ambiguities in viewing and interpreting objects are minimised (McMahon and Browne, 1998). Engineers use geometric models because they are easy to alter and to adjust, and so they represent an economic and convenient substitute for real-world objects in the development stage of a product. These geometric models can also provide important information (such as dimensions and tolerances) needed to manufacture the part or to assemble the final product (Tizzard, 1995; Mottram and Shawn, 1996). Depending on the capabilities of the CAD system used, geometric models also have the potential to demonstrate the capabilities of the designed product in a simulation video, or to deliver relevant attributes needed to analyse the product's capabilities in FEM.

Historically, three main types of geometric models were distinguished:

- Line or wireframe models
- Surface models
- Solid models

Today all three approaches are combined and used in most modern CAD systems. In addition to these, there are other techniques and ways such as Voxels or Point clouds to visualise three dimensional data (see Figure 3.1), but they are not widely used in the CAD environment. Each technique has its own particular application in the design of data, which is dependent on the ability of the method to model certain geometric structures effectively or to generate data for analysis or manufacture.

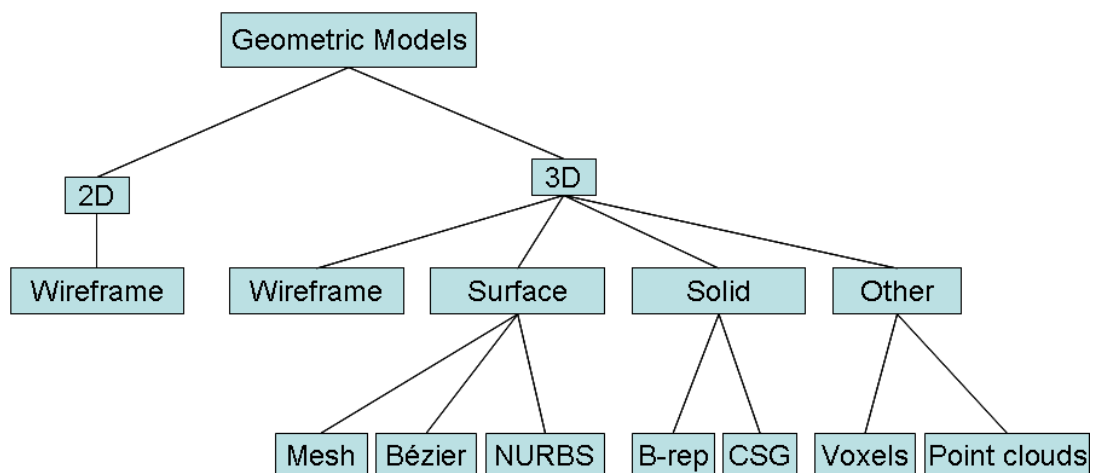


Figure 3.1 Classification of geometric modelling
(adapted from: Rodriguez, 1992, p. 179)

According to Rodriguez (1992) geometric models can also be classified according to the object's unique characteristics and features. The three predominant classifications are: *analogue*, *iconic-descriptive*, and *symbolic-descriptive*. These three characteristics can be described as follows:

- *Analogue*: Analogue models substitute one object for another and thus allow the problem/task to be solved in a substitute manner. In other words, analogue models do not have to look like the original objects as long as they follow the same physical principles or simulate the original system's capabilities. Graphs, charts, and force diagrams, for example, can be classified as analogue models because they can present magnitudes of physical quantities like distance or force (Dieter, 1983).
- *Iconic-descriptive*: An iconic model is a scaled prototype of the real object that looks similar to the object being represented, but much smaller. They can be represented either as 2D (e.g. scaled drawings) or 3D (e.g. as a scaled model) on a computer screen, or physically produced in rapid prototyping. Iconic models have the advantage that their manufacturing costs are much lower than for the original-sized objects since it is a scaled version. In terms of 3D representation, iconic models are often used in the automotive and aerospace industry for object visualisation, airflow, crash and stress simulation during the development processes.
- *Symbolic-descriptive*: Symbolic models, also called mathematical models, are abstract representations of the object or product being designed (e.g. wireframe model). Since geometric models are defined by mathematical functions, they are usually classified as symbolic. This type of data is often used in FEM analysis.

Since geometric modelling describes the object being designed both mathematically (symbolic) and visually (iconic), it is the most useful and comprehensive modelling technique available for designing, developing and analysing of new physical objects. Both 2D and 3D geometric modelling are extensively used in computer graphics,

computer-aided design and manufacturing, but also in many other areas such as medical or geological imaging. While two dimensional modelling is mostly important in computer typography and technical drawing, three dimensional modelling is central to CAD and CAM including data analysis and simulation.

Since this thesis focuses on watermarking of three-dimensional CAD data, only the characteristics and capabilities of three-dimensional representation types will be evaluated and discussed. Although today's commercial systems no longer make the distinction between wire, surface and solid models these categories still serve as a useful introduction to geometric representation.

3.1 Line or Wire-frame Modelling for 3D Data Representation

Line or wire-frame models are probably the earliest type of geometric modelling to be implemented and date back to 1960s because they are the simplest form of geometrical representation of three-dimensional objects (Rodriguez, 1992). In this approach the geometry of a three dimensional model is defined as a series of lines and curves representing the edges of, and perhaps sections through, the object (McMahon and Browne, 1998). The name of the scheme arises from the wire-like appearance of the modes when viewed on a computer screen or hard copy (Figure 3.2 shows the wire-frame representation of a simple 3D mechanical component). Line or wire-frame representations may be regarded as an extension into a third dimension of the techniques used for visualising 2D draughting.

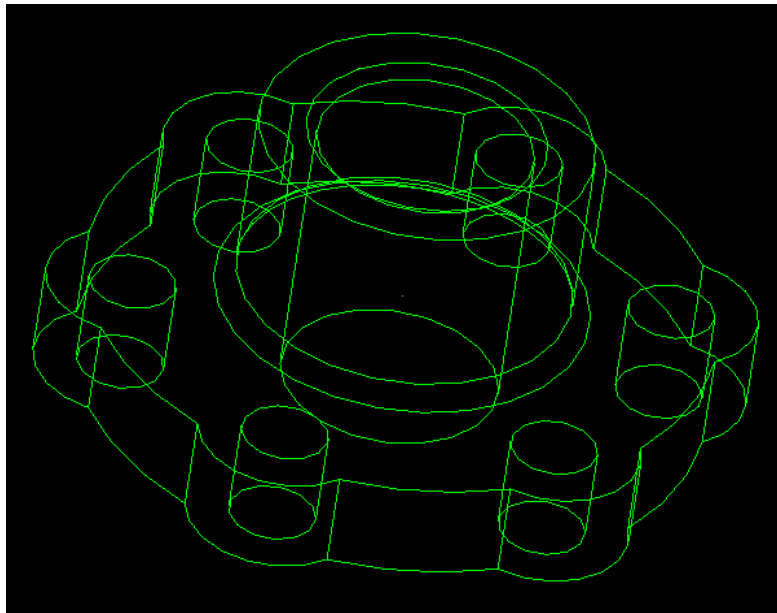


Figure 3.2 The wire-frame model of a mechanical component

Advantages:

Line or wire-frame representations have a number of advantages. For instance, wireframe models can be displayed very quickly and easily (no formatting or surface creation processing required) since their data size is much smaller than for surface or solid models. Since wire-frame renderings are relatively simple and fast to calculate, they are often used in cases where a high screen frame rate is needed (for instance, when working with a particularly complex 3D model, or in real-time systems that models exterior phenomena) (Rodriguez, 1992). Using wireframe models can even simplify the visualisation of the underlying design structures of complex 3D models (McMahon and Browne, 1998). Since wire-frame representation is a basic form of graphic visualisation, nearly all modern 3D CAD systems are able to display them.

Another positive feature is that a component modelled using wire-frame modelling is very simple to convert into 2D orthogonal drawings (or working/blueprint drawings for shop floor use), although it will require some manual processing. Normally, this

involves displaying the object from different angles to produce desired views for manufacturing, then positioning them onto the drawing sheet and removing or editing hidden or unnecessary lines in each view. The resulting views might therefore be 'inaccurate' copies of the actual object to a certain extent because edges and vertices will have been removed or hidden behind other lines. However, they are geometrically accurate (at least in two dimensions) and are suited for hardcopy presentation purposes (Tizzard, 1995). Even today the wire-frame format is still used for programming tool paths for DNC (Direct Numerical Control) and CNC (Computer Numerical Control) machine tools.

Disadvantages:

Probably the biggest negative aspect of 3D wire-frame representation is its visual presentation on a computer screen. At first glance it is not easy to determine the shape of the object because it is difficult to distinguish which lines are in the foreground and which in the background (see example in Figure 3.2).

Furthermore, since wireframe models do not contain any information about the surfaces between the edges, this scheme has limited ability to determine the geometric intersections of two objects (such as a collision in an object assembly simulation) or to calculate mechanical properties of the object (Rodriguez, 1992).

3.2 Surface Modelling for 3D Data Representation

As the name implies, surface modelling involves representing an object by specifying some or all of its surfaces. Basic objects such as a cube or pipe are described by simple mathematical surfaces like plane or cylinder, and can be easily represented by wireframe models. However, other structures such as a car body

panel, aircraft fuselage, wings, or a turbine fan blade have much more complex surface forms which are more complicated to visualise. In these cases, wireframe models would be inappropriate not only for proper data representation, but also for object simulation and precise data analysis (Mottram and Shawn, 1996). Therefore, surface modelling is used to represent free form shells and surfaces of complex 3D objects. Figure 3.3 illustrates an example object for which surface modelling is ideally suited.

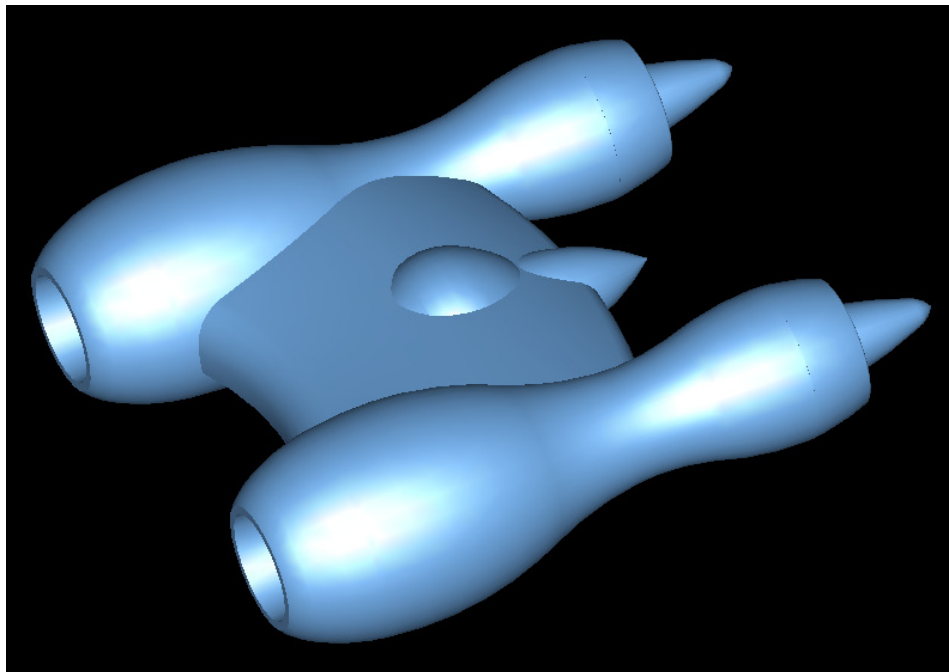


Figure 3.3 An object defined by complex surfaces

Surface models do not necessarily have only “ruled or defined” surfaces (i.e. mathematically or geometrically defined) such as planes, cylinders and spherical surfaces, but also some free-form surfaces like spline² or wiggle. Therefore, surface models are not stored or defined in terms of simple analytical equations, but through complex mathematical definitions requiring poles/knots, degrees, and number of

² A spline is an approximation curve that is described by mathematical equations to match a complex shape.

patches of the surface being represented. The degree of surface complexity determines its mathematical properties and can be seen as the representation of shapes by a polynomial with different variables and degree values.

According to Tizzard (1995) surface modelling provides a far better visual and mathematical description of an object's surfaces than wireframe modelling, because it places flat and curved patches together to form the actual surfaces that surround the object. The term patches is used in CAD modelling to designate a limited region on a larger surface and represents essentially the 'faces' of a modern B-rep modeller. Patches are mathematically defined by a curve-bounded collection of points whose coordinates are given by continuous two-parameter (u, v) or three-parameter (u, v, z) functions (Rodriguez, 1992).

CAD software packages use two basic methods for the creation of surface models. The first begins with the construction of curves (splines) from which the 3D surface is then swept or lofted. The second method is direct creation of the object surface by manipulating the surface knots or control points. From these initially created surfaces, other surfaces are constructed by using either derived methods such as offsetting or angled extensions from existing surfaces, or via bridging and blending between groups of surfaces (Wikipedia, 2009b).

An advantage of surface models is that they are easy to construct by simply manipulating surfaces with operations such as sweeping, revolving, extruding, or tweaking entities. In addition, a designer can use patches to create a transition between adjacent surface edges. Furthermore, surface modelling is very appropriate for finding the intersection of complex surfaces in space and creating models for shaded renderings (Mottram and Shawn, 1996). Surface models also provide an

excellent basis for the generation of manufacturing information (e.g. for the machining of press-tooling for sheet metal presswork) and for analysis data (e.g. finite element models for object stiffness calculations).

However, surface representations also have several drawbacks. In general, they are more computationally demanding than wireframe modelling, and they also require more skills in their construction and use (specifically for free-form surface types). The main disadvantage of surface modelling is, however, that it is ‘only’ an approximation of complex surfaces, albeit if a close one. In other words, a surface patch is only a relative approximation to the original surface (since interpolation is used for surface calculation, and tolerances are inevitable).

For the representation of surface models most mechanical CAD systems use Bézier surfaces or non-uniform rational B-spline (NURBS) surfaces, although there is a belief that both will be ultimately replaced by polygonal meshes (Rodriguez, 1992). The following sections will demonstrate each of these techniques and will discuss their theoretical aspects and characteristics.

3.2.1 Polygonal Mesh Surface

The surface of three dimensional objects can be approximated and represented by a mesh of planar polygonal facets, mostly triangles (Arnold et al., 2003). Therefore, a three dimensional polygonal mesh is a collection of vertices, edges, faces, polygons and surfaces in 3D space joined together to form and represent the shape of a 3D polyhedral object commonly used in computer graphics imagery (CGI) (Uccheddu et al., 2004). The faces usually consist of triangles, quadrilaterals or other simple convex polygons with n edges, since this simplifies rendering, but may also be

composed of more general concave polygons, or polygons with holes. Such polygonal meshes are used for representing many kinds of three dimensional data. An example of a 3D object represented as polygonal mesh is shown in Figure 3.4.

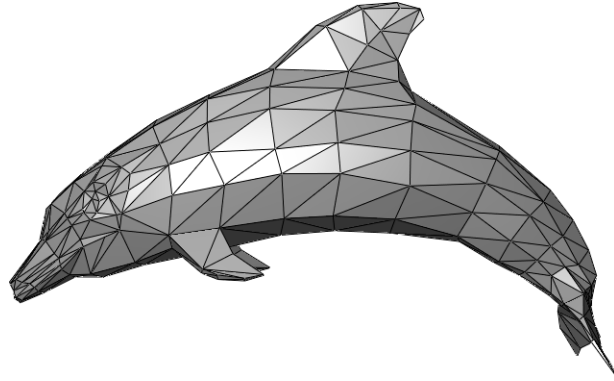


Figure 3.4 A dolphin represented by polygonal mesh
(source: Wikipedia, 2009c)

Polygon meshes may be represented in a variety of ways using different methods to store the vertex, edge and face data(Wikipedia, 2008d). These include:

- *Face-Vertex Meshes*: A simple list of vertices and a set of polygons that point to the vertices it uses.
- *Winged-Edge Meshes*: In which each edge points to two vertices, two faces, and the two edges that it touches.
- *Half-Edge Meshes*: Similar to Winged-Edge meshes except that only half the edge traversal information is used.
- *Quad-Edge Meshes*: A Quad-Edge mesh stores edges, half-edges, and vertices without any reference to polygons. The polygons are implicit in the representation, and may be found by traversing the structure.
- *Corner-Table*: A Corner-Table stores vertices in a pre-defined table, such that traversing the table implicitly defines polygons. This is in essence the ‘triangle

fan' used in hardware graphics rendering. The representation is more compact, and more efficient to retrieve polygons, but operations to change polygons are slow. Furthermore, Corner-Tables do not represent meshes completely. Multiple corner-tables (triangle fans) are needed to represent most meshes.

- *Vertex-Vertex Meshes:* A Vertex-Vertex mesh represents only vertices, which point to other vertices. Both the edge and face information is implicit in the representation's ordering. However, the simplicity of the representation allows for many efficient operations to be performed on meshes.

The choice of the data structure is governed by the application, the size of data, the performance required, and the operations to be performed. For example, it is easier to deal with triangles rather than general polygons when displaying three dimensional objects, especially when computing surface geometry. For certain operations it is necessary to have a fast access to topological information such as edges or neighbouring faces. This requires a more complex structure such as the winged-edge representation. For hardware rendering, compact and simple structures are needed; thus the corner-table (triangle fan) is commonly incorporated into low-level rendering APIs such as DirectX and OpenGL (Polygonal Mesh, 2008; Wikipedia, 2008d).

Advantages:

One of the main advantages of polygonal meshes is that they are the common denominator of other representations. In other words, nearly all CAD systems are capable of viewing or processing mesh data (e.g. STL or VRML formats), and nearly all 3D formats can be converted (i.e. exported) into polygonal meshes.

In addition, a positive aspect of polygonal mesh is that modern graphics hardware is able to render millions of triangles every second, which makes data handling and representation very quick. This means, that data animation or representation (e.g. in pocket computer game consoles, car satellite navigation systems) can be made possible without complicated and expensive three dimensional graphics accelerators.

Disadvantages:

Polygonal meshes have several limitations and disadvantages. First of all, since triangles, quadrilaterals and other polygons used in mesh representation are predefined shapes, complex curved surfaces (e.g. the surface of a turbine blade) can only be approximated for representation. Therefore, accuracy is inherently relative; more polygons would mean more accuracy. In fact, each triangle can be seen as an estimated section of the actual shape, a discrete sample of the curved surface. In other words, polygons cannot accurately represent curved objects, since they are surface-modules to approximate the object's surface. As a result, using this 'approximation' for data analysis or simulation might distort calculation and analysis results.

Another disadvantage of polygonal mesh representation is that they are very sensitive to transformation processes like rotation, translation, scaling, mesh simplification, re-triangulation or re-meshing (Kurowski, 2004). So, although the data size of a 3D object could be reduced by mesh simplification, the accuracy and precision of the model's surface would suffer (see example in Figure 3.5b). Moreover, where re-triangulation and re-meshing is aimed to re-distribute or to smoothen the model's surfaces (see example in Figure 3.5a), it could destroy or change the object's shape.

In other words, both methods would modify vertices' coordinates slightly, which again would slightly disfigure or alter the model's shape and appearance.

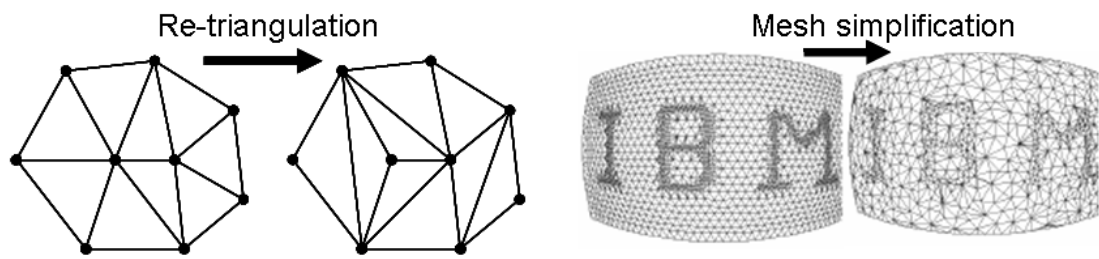


Figure 3.5 Mesh modifications
a) re-triangulation (left) and b) mesh simplification (right)

3.2.2 Bézier Surface

Bézier surfaces³ are an extension of the idea of Bézier curves, and share many of their properties (Mottram and Shawn, 1996). The following figure (Figure 3.6) shows a simple 4th order Bézier curve (C), its end and control points (1 - 4), and its control polygon (P). With the help of these four control points and the control polygon, all other points on the parametric curve can be interpolated. In general, a Bézier curve depends on the degree of the represented curve and has therefore not a fixed number of end and control points.

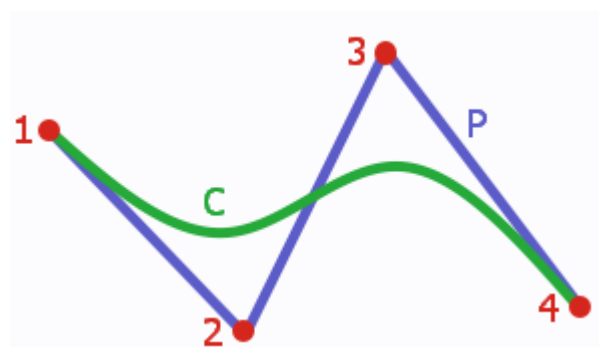


Figure 3.6 A Bézier curve (C) with its control points (1-4) and its control polygon (P)

³ Bézier surfaces are mathematically defined splines used in computer graphics, computer-aided design, and finite element modeling. As with the Bézier curve, a Bézier surface is defined by a set of control points and control polygons; however, a key difference is that the surface does not pass through the central control points.

(source: Wikipedia, 2008a)

Each point on a Bézier curve is computed as a weighted sum of all control points. This means that each control point influences the final curve according to an assigned blending function (Davies et al., 1991). The x and y coordinates of each point on the curve segment (C) can be determined as follows:

$$X(t) = (1-t)^3X_1 + 3t(1-t)^2X_2 + 3t^2(1-t)X_3 + t^3X_4 \dots$$

$$Y(t) = (1-t)^3Y_1 + 3t(1-t)^2Y_2 + 3t^2(1-t)Y_3 + t^3Y_4 \dots$$

where t varies between 0 and 1 ($0 \leq t \leq 1$). In each curve the first and last points (in the example above points 1 and 4) define the position of the start and end points of the curve segment. At these positions the parameter t has the values $t = 0$ and $t = 1$ respectively.

Similarly, three dimensional Bézier surfaces (or Bézier patches) can be defined by a grid of sixteen control points. These can be thought of as four rows, with each row being a 2D Bézier curve. In this way, points are interpolated bilinearly to generate the surface patch. The Bézier curves which form the boundaries of the patch will also be points in the 3D surface. An example of a Bézier surface with 36 control points is shown in Figure 3.7.

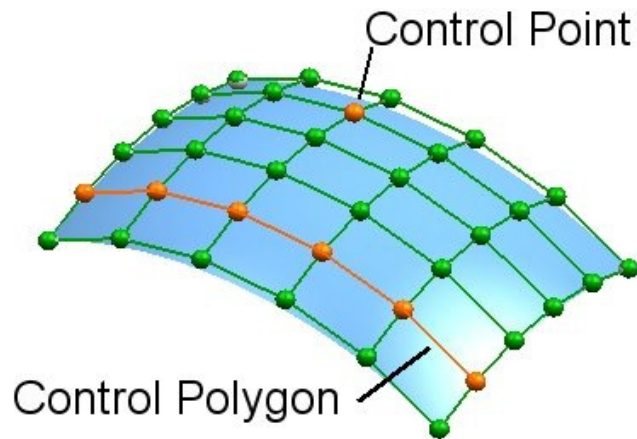


Figure 3.7 Example of a Bézier surface
(source: Wikipedia, 2008b)

Bézier surfaces give an improved description of a surface when compared to wireframe and mesh representation, as information from within the boundaries is used to define the surfaces of an object. This can also help to define a surface in space and can help to manipulate or to design unique surface shapes.

Advantages:

This method has several advantages over the polygon mesh method. Clearly, one of the advantages of Bézier surfaces is that they allow a much more concise and smooth representation of a surface than vertex-polygons in mesh representation. Other advantages are that they provide nearly exact analytical descriptions of surfaces, and permit easy deformations of those surfaces. Finally, if texture mapping is being performed, Bézier surfaces are convenient because the parameters (u, v) used to generate the surface can easily be reused as texture parameters (u, v) .

Disadvantages:

There are, however, some disadvantages when using Bézier patches for complex surface representation. For instance, when rendering a parametrically defined patch

with graphics hardware, it is usually necessary to generate a vertex-polygon description of the object first. Also, given a set of 3D vertices, it may be difficult to exactly extrapolate (precision might suffer) the control points for a Bézier surface. Furthermore, if multiple patches are combined to form a description of a complex shape, there may be problems with normal interpolation across patch boundaries, and with cracks forming at patch boundaries (Bentley, 2003).

However, the major disadvantage of Bézier curves emerges when representing complex curved surfaces. Here, the degree of the curve rises with the number of control points, and this again makes it numerically unstable.

3.2.3 Non-Uniform Rational Basis-Spline (NURBS)

A Non-Uniform Rational B-Spline⁴ (NURBS) is commonly used in computer graphics for generating and representing curves and surfaces of three-dimensional objects. Like Bézier surfaces, NURBS can provide mathematically precise representations of standard analytical shapes, such as cones, as well as complex free-form shapes, such as car body panels.

A NURBS surface is a set of smoothly connected curves enclosed by an angular control polygon defined by its order, a set of weighted control points (vertices), and a knot vector. The order of a NURBS surface defines the number of adjacent control points that influence any given point on the curve. In other words, each point of the curve is computed by taking a weighted sum of a number of the control points. The knot vector is a sequence of parameter values that determine where and how the

⁴ B-spline is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition.

control points affect the NURBS curve. The number of knots is always equal to the number of control points plus the curve's degree plus one.

Essentially, a B-Spline can be considered to consist of multiple Bézier arcs and provides a unified mechanism to define continuity in the joints. B-splines use external conditions to put multiple pieces together while keeping the original concept of control points. The neighbouring curves share some control points. External conditions are either implicit (uniform curves) or explicitly given by a knot vector. A knot vector defines how much information should be used by neighbouring curves in defining the slope of adjacent segments at the point they meet.

NURBS curves and surfaces are generalizations of both B-splines and Bézier curves and surfaces, while the primary difference between B-spline and Bézier is the weighting of the control points which makes NURBS curves rational (non-rational B-splines are a special case of rational B-splines) (Davies et al., 1991). Therefore, it can be said that the shape of a NURBS surface is determined by degree, control points, weights, and knot vectors. Since NURBS and Bézier surfaces are closely related, they share some characteristics and features.

Advantages:

Probably the biggest advantage of NURBS is its ability to represent complex free-form shapes (e.g. a wing design) and arbitrary topologies and degree. Compared to polygonal mesh representation NURBS objects have fewer surfaces, are easier to edit (because control points only have a local effect) and stay accurate during data transfer. Therefore, NURBS are the preferred toolset used for presenting free form surfaces amongst most industrial engineers in the aerospace, motorsport, and car

industry, and can be considered standard in the CAD environment. Another positive aspect of NURBS is that its curves and surfaces can be re-parameterised without changing its geometric shape.

Disadvantages:

The biggest drawback with NURBS might come from differences in the algorithms used to represent them in commercial CAD systems. For instance, an algorithm to reproduce NURBS faces in, say, a CNC mill-machine may not match the algorithm used by the CAD system that created the model and could cause tolerance and accuracy problems (Hayden, 2004). Furthermore, since control points (corner points are fixed and define the vertex of the control polygon) of NURBS surfaces are ‘only’ approximations of the characteristic polygon surface, simulation and calculation results in a FEM analysis for instance might be distorted (McMahon and Browne, 1998).

Another disadvantage might come from the fact that NURBS is lacking one to one correspondence between control points and the actual shape of the polygon surface. For example, moving a control point by 10mm does not result in a change the object’s shape of exactly 10mm, since the position of each control point on the control polygon is an interpolation of all control points.

3.3 Solid Modelling for 3D Data representation

Surface does not represent or define 3D volume. Consequently, solid, or volume, models were developed in the early 1970’s with the aim of providing a complete and unambiguous representation of three-dimensional models in computer graphics (McMahon and Browne, 1998). For many engineering purposes two-dimensional

representations are satisfactory, but for enhanced visualisation and increasing application of computer engineering analysis (e.g. FEM, MATLAB, ANSYS), or the direct generation of manufacturing information (e.g. CNC cutter paths), means that an ideal representation should be as complete as possible (Rodriguez, 1992; Tizzard, 1995; Kalay, 1989). Today, the once separated surface and wire frame models are now incorporated within the solid model's data structure, and can be created or displayed, if required, by selecting a subset of the overall model.

Solid models consist of two major components: geometry and topology. Geometrical data is the shape, size, and the location of the geometric elements, while topological data describes the connectivity and associativity of these geometric elements as non-graphical, relational information. In other words, a solid model defines the volume of an object and not just its surfaces or edges. An example of a 3D model is shown in Figure 3.8.

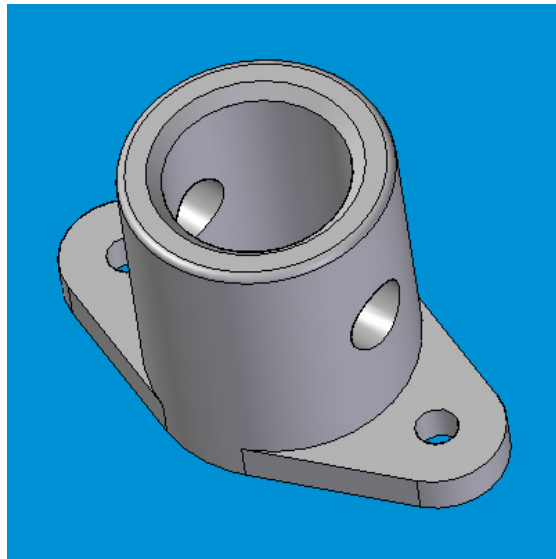


Figure 3.8 An object represented as a solid model

Solid modelling has three major attributes which describe the geometric representation of a physical object; *complete*, *valid* and *unambiguous*. They can be summarised as follows:

- *Complete*: points in space that can be classified as inside/outside of an object.
- *Valid*: vertices, edges, faces are connected properly with each other (and do not self intersect).
- *Unambiguous*: there can only be one interpretation of an object.

Solid models offer a number of advantages over pure wireframe and surface models. Since they are able to describe and distinguish between exterior and interior volumes and can have properties such as material density, solid models also offer the potential to analyse and calculate the object's abilities and properties (i.e. mass, strength, centre of gravity, moment of inertia, deformation under stress, etc.) in simulations without requiring the actual physical object. Therefore, the primary use of solid models are for CAD, engineering analysis, computer graphics and animation, rapid prototyping, medical testing, product visualisation and visualisation of scientific research.

Because of this wide field of application, solid models are used in many industries like entertainment, healthcare and engineering where unambiguous and precise representations of objects, parts and assemblies are required. Since solid models feature more accurate information about the closure and connectivity of shapes than wireframe and surface models, they have become the most important type of models used in modern mechanical CAD systems (McMahon and Browne, 1998).

Many methods (e.g. parameterized primitive instancing, spatial occupancy enumeration, cell decomposition sweeping, etc.) have been proposed for implementing solid modelling, of which none are entirely complete/unambiguous. However two techniques have been partially successful and have successively come to dominate the development of solid modelling. These two methods belong to the constructive approach. The most widely applied variant is the constructive solid geometry (CSG) method which achieved early prominence in CAD systems. The second method, the boundary representation (B-rep), dominates in today's applications.

3.3.1 Constructive Solid Geometry

Constructive Solid Geometry is a specific way of representing three-dimensional solid objects. In this method, 3D models are constructed by using combinations of simple solid 'primitives' and Boolean operators (Arnold et al., 2003; McMahon and Browne, 1998). Standard CSG primitives include cuboid, cylinder, sphere, cone, wedge, torus, prism and the like. Typical Boolean operators are *union*, *intersection* and *difference* (Constructive Solid Geometry, 2008; Zeijl J. van, 1994). The *union* of two objects results in an object that encloses the space occupied by the two given objects. *Intersection* results in an object that encloses the space where the two given objects overlap. *Difference* is an order dependent operator; it results in the first given object minus the space where the second intersected the first (Mitchell and McCullough, 1995).

Examples of CSG primitives and operators are visualised in Figure 3.9. Some primitives may require other kind of transformations such as scaling, rotation or translation before they can be positioned at the desired place. The CSG method

enables users to develop complex models relatively quickly, however, within the limitations of the set of primitives available within the system.

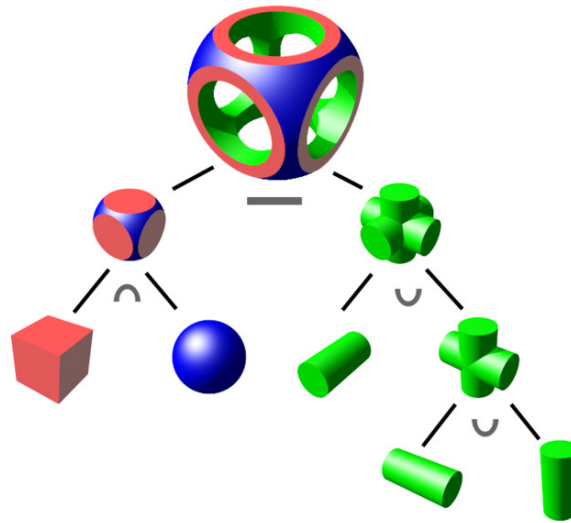


Figure 3.9 Boolean functions: union \cup , intersection \cap , difference \ominus
(source: Wikipedia, 2009a)

CSG models have a number of practical uses. Particularly, they are used in cases where simple and/or symmetric geometric objects are desired (e.g. for construction engineering), or where mathematical accuracy is important (e.g. for data simulation or analysis). CSG modelling is also popular for designing computer games, because the modeller can use a set of relatively simple geometric objects (primitives) to create more complex design geometry or background environment (e.g. buildings and houses for an urban battlefield in a war game).

Advantages:

A major advantage of CSG is that it can easily assure that objects and their surfaces are closed solids since the primitive shapes are ‘complete’ (McMahon and Browne, 1998). This quality may be important for some manufacturing or engineering

computation applications where overlapping of solids or gaps between surfaces are not tolerated.

Another advantage is that, since CSG primitives have relative simple descriptive functions for shapes, they have a much simpler and smaller file size. This characteristic can be crucial for systems (e.g. in hand-held computer or mobile satnav systems) where a quick calculation time is required and/or the storage space is limited.

Disadvantages:

Problems with CSG occur when surfaces of objects being operated upon coincide. For example, when subtracting a box from another box to make a square cup, the result will frequently be wrong or create error messages if, in the design phase, the tops of the two boxes coincide. To correct this, one box should be made slightly larger than the other one. However, this 'correction' could slightly change the overall shape of the object since it removes small parts of the box.

Another negative aspect of CSG is that it is not able to reproduce complex spline or curved surfaces since it uses simple primitive shapes. In other words, it cannot represent the surface of a fan blade or the hull of a ship, for instance, which makes CSG unsuitable for more complex 3D CAD designs.

Furthermore, while providing an extremely compact representation, CSG-represented solids are inefficient sources of geometric data for applications that require explicit information about the composite shape or its boundary (Kalay, 1989). For example, if an application requires the identification of an edge for blending difficulties can arise because edges are not explicitly represented in the CSG data structure.

3.3.2 Boundary Representation (B-rep)

In the field of computer-aided design applications, **Boundary representation (B-rep)** is the dominating model for representing 3D shapes. As the name suggests, a three-dimensional object is represented as a collection of connected surface elements that define the boundary between solids and non-solids. In other words, Boundary representation is essentially a local representation connecting faces, edges and vertices of a 3D object using the limits. Figure 3.10 shows an example of a simple cube-model represented as B-rep.

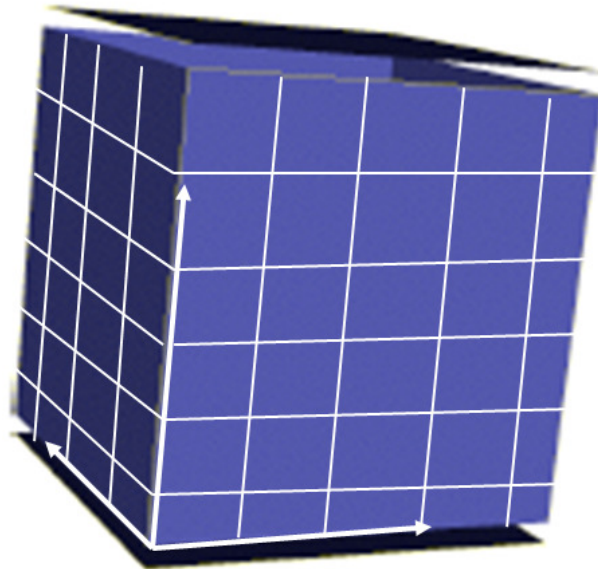


Figure 3.10 Boundary representation of a 3D model

A B-rep modeller, where a solid is represented by an evaluated data structure, contains elements which describe an object by its boundaries (Rooney and Steadman, 1987). These elements can be divided into two categories: topology and geometry. These two concepts are explained in the following sections using the terminology of the ACIS modeller (Corney and Lim, 2001).

Topology of B-rep Data

The main topological items of a 3D B-rep object are: FACES, EDGES and VERTICES. The shape (i.e. geometry) of a FACE is defined by a ‘surface’ whose boundary is represented by a collection of EDGES associated with it. An EDGE shape (i.e. geometry) is defined by a ‘curve’ whose boundary is represented by a pair of VERTICES, and the location of a VERTEX (i.e. its geometry) is defined by a ‘point’ (with x, y and z coordinates). EDGES can also be defined to lie between two or more FACES regardless of their shapes or sizes (Mitchell and McCullough, 1995).

FACE, EDGE and VERTEX items are also known as topological entities or composite entities, because they define how things are interconnected or refer to collections of other elements. FACES have also an underlying parametric orientation associated with their defining surface geometry expressed in term of (u, v). Figure 3.11 illustrates a spline face where its topological entities and elements, and its underlying parametric orientation are highlighted.

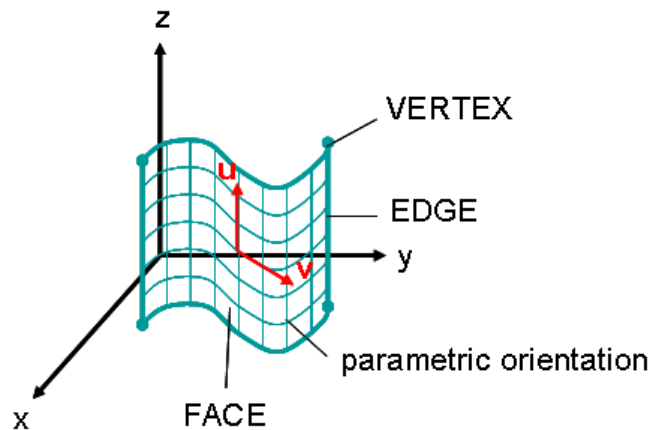


Figure 3.11 Definition of a B-rep model's face

There are many ways of viewing the B-rep data structure: for instance, it can be thought of as being a tree, or a hierarchy, with BODY at its root. Figure 3.12 below showing the ACIS hierarchy of a B-rep data.

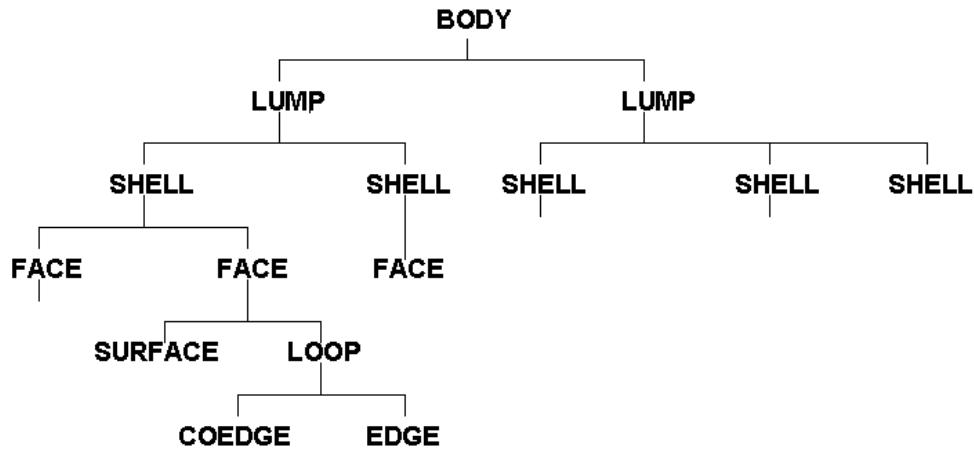


Figure 3.12 ACIS hierarchy representation of a B-rep data
(source: Corney, 1997)

A BODY can have a collection of LUMPs, each of which is comprised of one or more SHELLs. These SHELLs are formed from connected FACES. Likewise, FACES are composed of LOOPS, formed from circuits of bounding EDGES and COEDGES (Corney and Lim, 2001). Most EDGES on a 3D model also lie between two FACES. In this sense, it can be said that boundary representation is essentially a local representation connecting FACES, EDGES and VERTICES with each other.

An example of the hierarchical structure is also shown in Figure 3.13 where a three dimensional B-rep model is subdivided into its entities. The BODY on the left is defined by two LUMPs (one cylinder and one quadratic plane with a hole). LUMPs are a collection of SHELLs, which again are a collection of FACES. So, basically, the cylinder is represented by three, and the plane with a hole by seven differently shaped FACES. As a result, the ‘one’ BODY is represented by ten individual FACES.

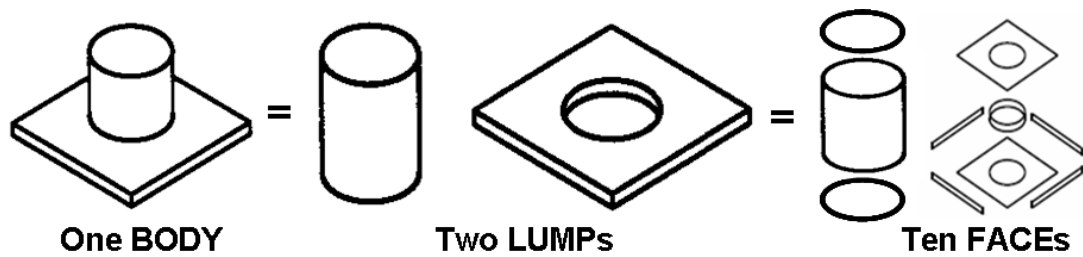


Figure 3.13 Hierarchical representation of a B-rep model
(source: Corney, 1997)

Although, theoretically, only three topological entities (i.e. FACE, EDGE and VERTEX) are needed to describe three-dimensional shapes, in practice a number of other entities are incorporated into the ACIS data structure. The reason for this is either to increase the speed of modelling operations or to provide high-level handles (i.e. means of reference) (Corney and Lim, 2001). Table 3.1 gives an informal summary of the various ACIS entities used to represent shapes by describing what they comprise and what they physically represent.

Table 3.1 Representational ACIS ENTITIES
(source: Corney, 1997)

Entity	Comprises	Represents physically
BODY	Collection of LUMPs or WIRES.	Highest level ENTITY in an ACIS model can be a 1D, 2D or 3D shape with a complete or incomplete boundary.
LUMP	Collection of SHELLs.	A region of a BODY disjoint from any other LUMP.
SHELL	Collection of FACEs and/or WIREs.	A continuous portion of a LUMP.
FACE	One SURFACE and zero or more LOOPs.	A portion of an individual surface.
LOOP	Circuit (or list) of COEDGEs.	Connected portion of a FACEs boundary which may be open or closed.

Entity	Comprises	Represents physically
COEDGE	EDGE and (on spline surfaces) a PCURVE.	Records the occurrence of an EDGE in a FACE boundary.
VERTEX	APOINT	Boundary of an EDGE.
EDGE	Collection of COEDGES, two VERTICES and a CURVE.	Holds the model together with adjacency information.
WIRE	Collection of EDGES.	A continuous collection of EDGES not attached to a FACE or enclosing any volume.
SURFACE	Geometric definition.	The shape of a FACE.
CURVE	Geometric definition.	The shape of an EDGE.
APOINT	(x, y, z) position.	The location of a VERTEX.
PCURVE	Spline curve in (u, v) space.	Parametric curve on a spline defining a COEDGE's shape.
SPHERE	Analytic surface.	Refinement of SURFACE.
PLANE	Analytic surface.	Refinement of SURFACE.
CONE	Analytic surface.	Refinement of SURFACE.
TORUS	Analytic surface.	Refinement of SURFACE.
SPLINE	Spline surface.	Refinement of SURFACE.

Geometry of B-rep Data

The geometry of B-rep objects is defined by underlying simple analytical equations (e.g. plane, cone, sphere, torus, spline, etc.), which are similar to those used in CSG representation. These geometric shapes again have an underlying parametric orientation associated with their defining surface geometry. Figure 3.14 shows a few basic geometric shapes with their parametric orientations illustrated.

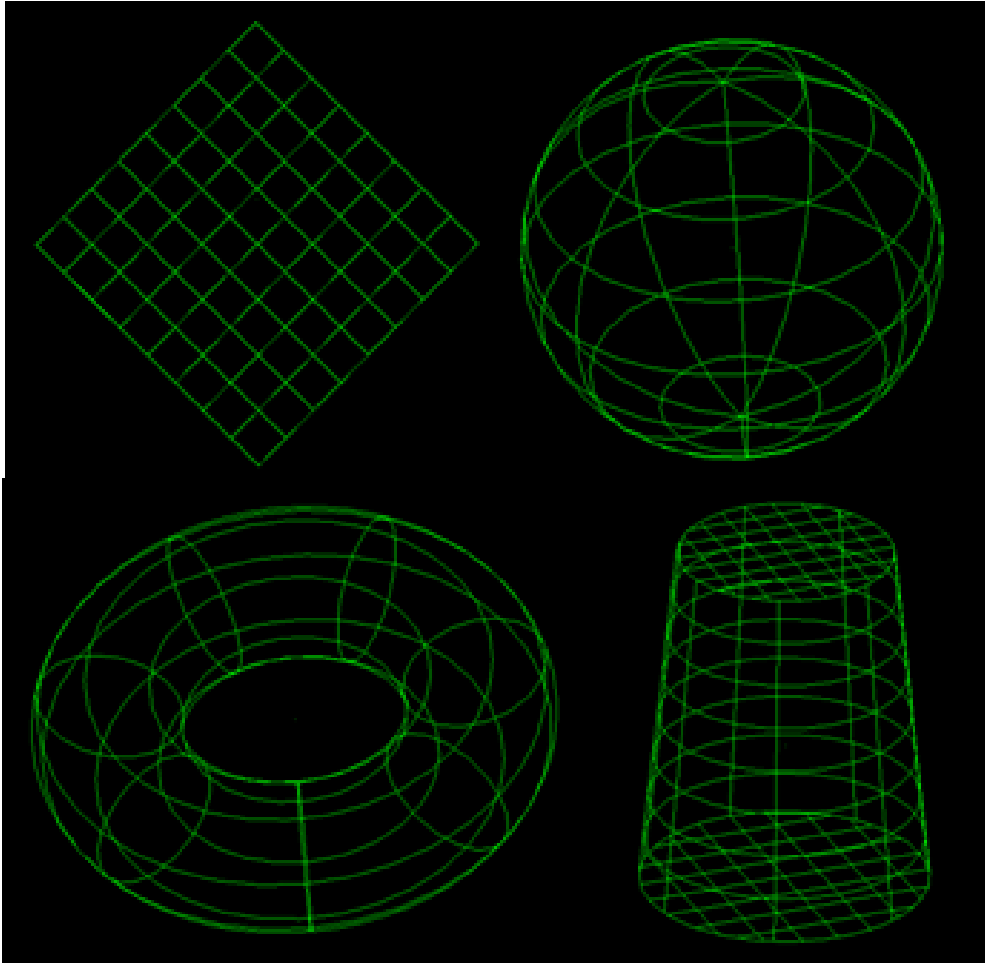


Figure 3.14 Concept of parametric surface definition based on basic geometrical objects

To allow geometry to be shared by different ENTITIES and yet maintain a clear separation between the representation of geometry and topology, ACIS defines two classes for each type of supported geometry. The two classes are distinguished by capitalisation of their names (Corney, 1997):

- **Uppercase:** Geometry classes (such as CONE, SPLINE and PCURVE) keep a record of how many other ENTITIES refer to the surface, or curve, and supply member functions for memory management and save/restore utilities.

- **Lowercase:** Geometry classes (such as cone, spline and pcurve) define the constructors and methods required to display and manipulate the shape by various modelling operations.

Therefore, the equations which represent a type of curve or surface, reside in the lowercase class, while the uppercase class supplies all the administrative functions. The following table (Table 3.2) lists the lowercase geometry classes supported by the ACIS Kernel.

Table 3.2 Lowercase geometry classes of B-rep data (source: Corney, 1997)

Name	Description
curve	Defines functions common to all curves such as position and parameter based inquiries (i.e. Base Class).
ellipse	Defines a circle (or ellipse) in any plane.
straight	Defines an infinite straight line.
int curve	Defines a curve formed by the intersection of two surfaces or an exact spline.
pcurve	Defines a curve on a spline, or blend surface in terms of (u, v) parameters.
surface	Defines functions common to all surfaces such as position and parameter based inquiries.
plane	Defines an infinite planar surface.
cone	Defines conical surfaces and cylinders.
torus	Defines a toroidal surface.
sphere	Defines a spherical surface.
spline	Defines procedural surfaces such as exact splines and blends.
meshsurf	Defines a composite mesh surface.

Considering these topological and geometrical features the B-rep is a much richer and more complex data structure than CSG or the meshes to define three dimensional shapes. Furthermore, the high degree of interconnections within the data structure means that even small changes have to be made with care. The interdependence

inherent in the B-rep data structure is well described by Stroud (1992, p. 89) who says:

“... the shape of an edge cannot be changed without changing the position of the surface, unless the new curve also lies in the surfaces of the faces adjacent to the edge. Similarly the position of vertices can not be changed unless they lie on the curves of all edges meeting at the vertex and hence the vertex position lies on the surfaces of all the faces meeting at the vertex.”

Advantages:

The B-rep scheme has a strong history in CAD because it is closely related to traditional drafting. Its main advantage is that it is possible to construct solid models of many shapes that are difficult to build by just using primitives.

Another major advantage of boundary representation is that it is relatively simple to convert into a mesh or wireframe model because the model's boundary definition is similar to the wireframe definition. Furthermore, the B-rep data structure makes it easy to extract important features (e.g. holes, slots, etc.) for advanced manufacturing reasons..

Last but not least, unlike the CSG representation, which represents objects as a collection of primitives and uses Boolean operations to combine them, boundary representation is far more flexible and easily modified. Also, while in CSG edges are implicitly defined by intersection, in B-rep the rule applies - what you see is what you get - with every visible entity having an explicit representation. This allows a richer operation set and makes boundary representation a more appropriate choice for modern mechanical CAD systems than CSG or any other form of representation.

Disadvantages:

On one hand, boundary representation models facilitate a very high object resolution, but on the other hand, the shapes represented by them are much more difficult to manipulate than those represented by means of the CSG model (because of the more complex data structure) and are, in particular, inefficient for spatial addressing (Kalay, 1989).

Moreover, related to the complex data structure, boundary representation data requires slightly more storage space than CSG because they store the explicit definitions of the more complex face types and the models boundaries. In other words, the defining functions and data for its faces, edges, and vertices tend to grow fairly fast for complex models.

3.4 Other Techniques for 3D Data Representation

Besides wireframe, surface and solid modelling there are some other methods of visualising three-dimensional data. For example, Voxels and point clouds are popular techniques to visualise 3D-design objects. These methods, basically derived from 3D scanning, will be discussed briefly in the following two sections.

3.4.1 Voxels

A Voxel (word combination of **v**olumetric and **pix**el) is a volume element representing a value on a regular image-grid in a three-dimensional space (Arnold et al., 2003). It is analogous to picture images where individual pixels (picture elements) represent a two dimensional image data (e.g. a photograph). The pixels for volume elements, however, lie on many individual layers, so called Voxel-layer, piled-up one behind the other.

Since a pixel point in a two dimensional space is defined by its x and y coordinates, a third z coordinate is necessary to define the position of a Voxel in a three-dimensional space. In many implementations of Voxel modelling the elements do not have an explicit third coordinate, but rather their relative position (i.e. distance) to each other. In other words, these volume elements sit behind each other or between other neighbouring Voxels, and their relative position to each other is used to create the required third coordinate to define a three-dimensional objects. Therefore, a stack of volume elements, as shown in Figure 3.15, can be considered a three-dimensional object.

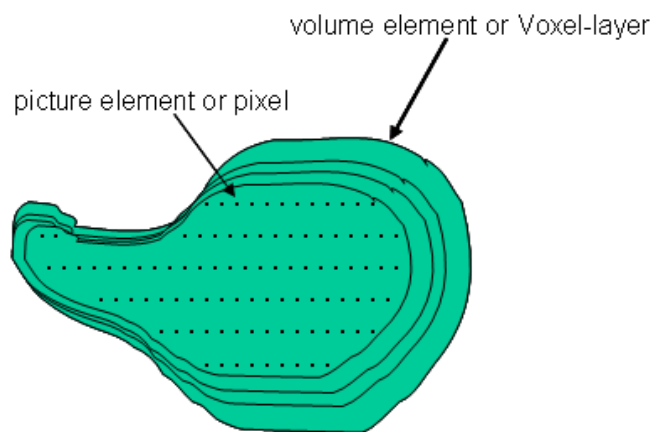


Figure 3.15 Voxels can be regarded as a stack of two-dimensional images

With the two pixel coordinates, the relative distance between Voxel-layers and an appropriate 3D rendering software views of three-dimensional objects can be created and viewed on a computer screen.

Voxels have a relatively wide scope of application. Most commonly, they are used in the field of healthcare to create visual-medical content for education and examination, and are applied to X-rays, CAT (Computer Axial Tomography) scans, MRI (Magnetic Resonance Imaging), cathode tube scans, and ultrasound. With these

technologies physicians and medical practitioners have the opportunity to obtain a ‘live’ and accurate three-dimensional model of the human body for diagnosis and examination. Furthermore, they are even able to visualise moving parts/bodies (e.g. a beating human heart) or represent smallest shapes and objects such as internally hidden tumours or macromolecules in biochemistry (see example in Figure 3.16).

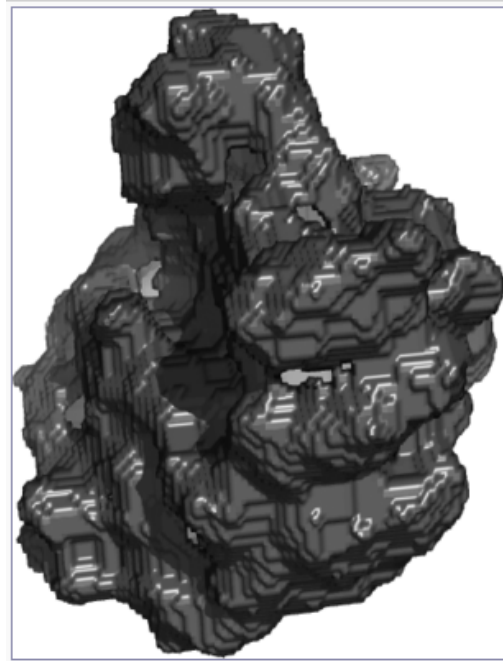


Figure 3.16 A macromolecule in biochemistry represented by Voxels
(source: Wikipedia, 2008e)

Another application of Voxels is the so called ‘Voxel terrain’ which is used to create height maps for GPS (Global Positioning System) navigation systems due to its ability to represent overhangs, caves, arches, and other 3D terrain features. Geologists even create three dimensional views, maps and profiles of our planet or accomplish ocean floor scanning (based on sound echoes).

Since Voxels are easily reproducible, they can be used to create 3D background for a car navigation system, or for a computer game console where a 3D accelerator is not available.

Last but not least, Voxels are often used in rapid prototyping where the individual volume elements (i.e. layers) are printed out one on top of the other to create a three-dimensional prototype object layer by layer (Mitchell and McCullough, 1995).

Advantages:

Voxel based modelling techniques extend the reach of 3D modelling systems to very complex arbitrary free-form shapes where interior information (e.g. hollow structure like honey comb) is important, and which cannot be easily handled by some other conventional surface based modelling tools.

Voxels can also model and represent different forms of inhomogenities in the same 3D-model, for example: use of different materials, or the use of different densities of the same material across the cross section of a solid object.

Another advantage of Voxel models is that they have very good local editability and permit the easy implementation of certain global operations that can alter the entire model (Chandru et al., 1999).

Disadvantages:

Voxel based modelling techniques are known for their robustness, flexibility and insensitivity to surface and object complexity. However, they have three major shortcomings. They are: 1) memory intensive since a large number of Voxels are needed to represent high-resolution models (e.g. a human body), 2) time intensive since a large number of Voxels need to be created and examined, 3) computationally intensive and expensive since surface extraction is needed to visualise the results (e.g. in a medical diagnostic).

Because volume elements have a limited resolution, they are not particularly suited for many CAD applications such as mass property, interferences, collision detection (e.g. for tool path planning) or FEM analysis and simulation. Furthermore, they have also limitations in accurate surface representation since a single surface has to be represented by a single Voxel-layer, and that again might not deliver enough precision.

3.4.2 Point Clouds

Another way of representation three dimensional models is by using point clouds. Point clouds consist of a set of vertices in a three dimensional coordinate system. These vertices are usually defined by x, y and z coordinates in the three-dimensional space (Arnold et al., 2003).

Point clouds are generally created by 3D scanners and laser scanners. These devices measure a large number of points on the surface of an object and assign each point x, y and z coordinates. These points are then computed to create a visible surface on the computer screen and the data saved to a file. In other words, point clouds represent only the visible surface of an object that has been scanned and digitised.

Point clouds themselves are not directly usable in most 3D CAD applications, and usually need to be converted to polygonal (triangle) mesh surface models, NURBS surface models, or other types of CAD data through a process commonly referred to as reverse engineering. Therefore, point clouds are related to the converted data types' characteristics (e.g. polygonal mesh surface), which have already been discussed earlier.

Point clouds are used for many purposes, including creating 3D CAD models for manufactured parts, industrial metrology or quality inspection, and a multitude of visualisation, animation, rendering and mass customisation applications (Point Cloud, 2009). For instance, this technique is often used in the designing process of automobiles where the outer shape of a prototype clay model is scanned with a 3D scanner to create a digitised data set.

Similar to other representation techniques, scanned point clouds of manufactured parts can be aligned to reference CAD models (or even another point clouds) to check for any differences or manufacturing mistakes. These differences can be displayed as coloured maps that give a visual indicator of the deviation between the manufactured part and the reference CAD model.

Last but not least, point clouds can also be used to re-create archaeological figures and shapes (e.g. a statue or vase) after scanning in the puzzle or jigsaw pieces, or to represent volumetric data used in medical imaging similar to MRI or visual scans (e.g. when creating tooth fillings).

Advantages:

Probably the biggest advantage of point clouds is that they can be used to visualise and digitize the surface of any complex three-dimensional object or shape that can be scanned. This is especially useful when no 3D design data about an object is available, or data for archiving reasons needs to be created.

Also, since each point in the cloud is defined by an individual x, y and z coordinate the values of each coordinate can be changed or adjusted if necessary.

Finally, point cloud data can be converted into any other 3D surface data type and therefore viewed in all CAD systems.

Disadvantages:

Despite the listed advantages there are also some negative aspects of point clouds. For instance, the resolution and accuracy of an object's surface is based on the capabilities of the scanning system. If a scanner has a coarse resolution, the scanned results (in this case the point cloud of the object) will only be a close approximation of the original model with a poor surface accuracy.

However, the biggest disadvantage of point clouds is that when object simplification is carried out or point cloud data is rendered into other formats such as polygonal mesh or Voxels, large amounts of data get lost during transformation (because not all data/coordinates can be converted or used), which again affects data precision.

3.5 Discussion

Various applications and techniques can be identified which represent three-dimensional CAD data. Each of these methods has a different approach with varying degrees of suitability for CAD application.

Wireframe Modelling is probably the simplest approach to visualise CAD data. The models created can effortlessly be displayed by all CAD systems and can easily be converted into 2D data for blue print purposes. However, the 3D wireframe structure makes it not easy to determine the object's actual shape (difficult to distinguish between lines in foreground and background), and intersections between surfaces cannot be calculated because faces are not displayed. Furthermore, since it is not able to represent complex spline surfaces and cannot provide object features (e.g. wall

thickness or material density) for numerical analysis, this modelling technique is not appropriate for sophisticated mechanical CAD and other data analysis applications.

Surface Modelling is a more advanced representation concept and is more appropriate for the visualisation of 3D objects in the CAD environment. Methods like polygonal mesh surfaces, Bézier surfaces and NURBS provide a much more accurate visual and mathematical description of physical objects than the wireframe approach and are relatively simple to establish. Since these techniques reproduce an object by its surfaces, they can be applied to various engineering analysis functions like collision detection, surface calculation or product simulation.

Polygonal mesh surfaces can be considered as the common denominator of all other surface representation techniques since it is easy to convert any other CAD format type into mesh type data, which is supported (i.e. readable) by nearly all CAD systems. On the other hand, planar face primitives like triangles or more general polygons provide a poor approximation of the real shape (especially curvatures), unless used in large quantities to define fine tessellations of highly curved geometries. Although algorithms for dealing with large numbers of individual triangles are simple and relatively robust, the structure of the data itself is sensitive to rendering and transformation actions (e.g. cropping, file compression or data simplification) which would alter the shape and appearance of the CAD object (Rossignac and Requicha, 1999).

Bézier surfaces offer a much more concise and smoother representation of a model's surfaces but their degree is directly proportional to the number of control points. NURBS on the other hand, produce a similar smooth and accurate description of an object's surfaces than the Bézier technique, and are less affected by data

transformation and transfer actions. Therefore, NURBS is often considered as the standard toolset for surface modelling in CAD applications. However, detecting and computing intersections of NURBS free-form surfaces involves elaborate mathematical calculation techniques and algorithms that are significantly slower and less reliable than their counterparts for triangular geometries (Rossignac and Requicha, 1999).

All surface modelling techniques have a few negative aspects in common. They are 'only' able to provide a rough digital approximation of complex surface structures and have relatively complex mathematical definitions for them. Furthermore, they often provide only an incomplete and ambiguous representation of a model because they do not distinguish between interior and exterior points of an object, which again might prevent a clear interpretation of the object. Also, some transformation and transfer actions could influence or harm the robustness and precision of models represented by surfaces.

In comparison to wireframe and surface modelling, Solid Modelling offers the most complete, valid and unambiguous representation and description of three-dimensional CAD data that is required for sophisticated CAD applications. Solid models carry all the necessary information to distinguish between interior, exterior and material density, have properly connected vertices, edges and faces, and possess a very precise and mathematical stable description of 3D objects (Rossignac and Requicha, 1999).

The main difference between CSG and B-rep modelling systems is the way in which a model is represented and stored in the geometric database and how that data is processed for further graphical output. A CSG model is defined as an ordered binary

tree in which the primitives are defined by their position, size and orientation and objects are represented as Boolean expressions between the primitives. The representation of a B-rep model within the geometric database is based on the notion that a solid is bounded by a number of faces, and these faces are effectively bounded by intersecting surfaces. By establishing the side of the surface where there is material, a solid can be defined. However, such a scheme is far more memory-intensive to store than CSG or triangle meshes because the interactions between edges, vertices, and surfaces are explicitly defined (i.e. more topological and geometrical information). The procedures for a B-rep's presentation onto a graphic display are far simpler than CSG (McMahon and Browne, 1998; Tizzard, 1995). Additionally, B-rep is capable to represent more complex free-form surfaces, such as those found in automotive body panels, and it has a richer operation set (i.e. blending, tweaking, etc.). At the same time, B-rep is also able to deliver important object features (e.g. material strength and density) for advanced engineering analysis (McMahon and Browne, 1998). In addition, B-rep data can easily be converted without resolution-loss into other formats such as polygonal mesh, NURBS or wireframe data. However, this wealth of data and complexity makes it more sensitive and fragile to some model modifications and alterations.

Other approaches such as Voxels and point clouds are also capable of representing complex shapes of 3D objects. While Voxels are able to deliver data about an objects exterior, interior and material or substance density, point clouds can only reproduce the object's outer surface. However, since Voxels contain a lot of information, they are more time and computationally intensive than point clouds and some other modelling methods.

Another negative aspect, which both techniques have in common, is that they can only be used to a certain extent in CAD applications because they have poor surface and object resolution and data accuracy (depending on the capabilities of the scanning system). Furthermore, before the data can be visualised on a regular CAD system, both data types need to be converted into common file formats such as polygonal meshes, but then they possess the characteristics of mesh data. Lastly, some data transformation and modification actions could alter the model's shape, appearance and accuracy, and are therefore an indication of low data robustness.

Many computer-aided-manufacturing processes use geometric models to generate instructions, and finite element methods for structural, numerical analysis and physical simulation of products. The results are often expensive engineering products, where even the slightest modification of data can lead to unwanted side-effects or could even destroy the intended object's functionality. Therefore, accuracy and high geometrical and topological precision are necessary to guarantee a functioning product.

As the previous sections have shown, each technique has its own particular strength in the representation and visualisation of CAD data, which is dependent on the ability of the method to model certain geometric structures effectively. Considering the needs for CAD applications and comparing the characteristics of the mentioned three-dimensional representation methods, only B-reps appear to be able to deliver and support the required attributes and features for a complete, valid and unambiguous representation of 3D objects. The data within the B-rep definition classifies the internal and external points of an object, properly connects vertices, edges and faces with each other (no gaps in the structure), and thus delivers an

unambiguous interpretation of a 3D object. Also, it is capable of representing free-form spline surfaces of more complex objects while describing geometrical objects with the highest precision. Besides, it has the potential to deliver object features like material density and other properties that enable the calculation or simulation of a component's mechanical capabilities. Last but not least, since B-rep has become a core representation technique for three dimensional data in the CAD environment, it can be converted into any other data type (i.e. STL, IGES, STEP) with a controlled reduction of its accuracy and robustness, and still offers possibilities to embed watermark information robustly and invisibly.

Chapter 4

4 Work Related to Digital Watermarking

In order to provide a broad overview of 3D watermarking techniques, chapter four presents different algorithms, schemes, and concepts applied to three dimensional digital data. In later sections, the different 3D watermarking schemes and approaches are grouped according to how three-dimensional objects are represented, and are classified and analysed within the scope of CAD applications. Additionally, the techniques are assessed regarding their robustness against various forms of data modifications and attacks.

However, before this is accomplished, other digital watermarking approaches that have been proposed for one and two-dimensional digital data, such as text document, digital image, video and audio data are briefly explained and discussed.

4.1 Watermarking Approaches for Non-Three Dimensional Data

4.1.1 Text-Document Watermarking

Methods for embedding information into text-based documents have been frequently reported (Brassil et al., 1995a; Low et al., 1995; Low and Maxemchuk, 1998; Brassil et al., 1995b) and have been used for many years to enable document authentication and secure information transmission.

Text watermarking methods fall into two main categories: first, techniques that hide information in the semantics, (i.e. in the meaning and ordering of the words), and second, techniques that hide information in the format (i.e. the layout and appearance of the text, words or letters).

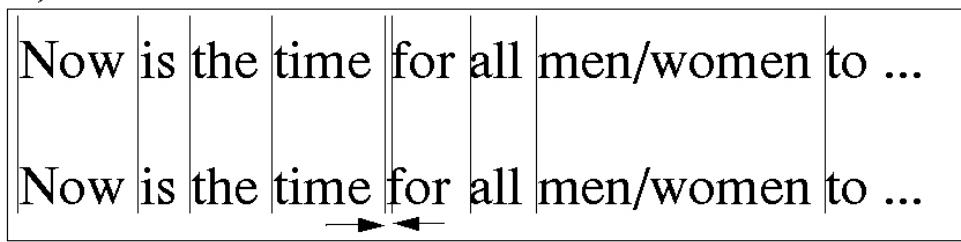
In the first case the message is not really ‘embedded’ in the text document, but rather covered by misleading words, letters, symbols within the text, or a combination of

these. Consequently, this first category is outside the scope of digital watermarking and will therefore not be considered here.

The second category, which is more in the sense of digital watermarking, uses information embedding techniques to hide secret messages into text documents. This category can be separated into three different methods: *line-shift coding*, *word-shift coding*, and *feature coding*.

For *line-shift coding* single lines of the text document are shifted (i.e. moved) imperceptibly for the naked eyes upwards or downwards by very small amounts. The information, that is meant to be hidden, is encoded in the distance the lines are shifted. In other words, the amount of line movement defines the message or signal to be transmitted. A 40-line text page, for instance, could hold up to 40 message bits (Berghel and O'Gorman, 1997; Low et al., 1995; Low and Maxemchuk, 1998).

Similar to line-shift coding, *word-shift coding* moves the words horizontally in order to modify the space between consecutive words. An example of word-shift coding can be seen in Figure 4.1, where the word “for” has been slightly shifted to the left (in second text line). In this sense, one secret message bit is encoded in one “word movement”; if a word is moved left, a 1 is encoded, if to the right, a 0. When decoding the message, *centroid detection* method (and of course the original text document) can be used to determine the movement of the “modified” words. For this technique the more words in a line and lines on a page, the larger the quantity of information that can be embedded.



**Figure 4.1 Word-shift coding by moving words horizontally
(adapted from: Katzenbeisser and Petitcolas, 2000)**

The third technique, called *feature or character coding*, imperceptibly extends the length of the upper or lower end-lines in characters and letters, like b, d, h, t, p, etc. (Berghel and O'Gorman, 1997; Thiemert et al., 2006). The variance in extensions (similar to line/word shift coding) can represent a secret message code that can be re-converted into readable or meaningful information. An example of feature coding is shown in Figure 4.2.

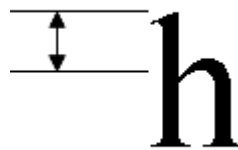


Figure 4.2 Feature coding by stretching the upper part of the letter 'h'

In the watermark retrieval stage the line-shift coding does not necessarily require the original text document (blind detection), since it relies on the assumption that lines are uniformly spaced and the standard line spacing (e.g. in Microsoft Word) could be used for referencing. Whereas for word-shift coding the original text document is required (non-blind detection), since spaces between words are usually variable (e.g. tab format left, right, justified, etc.).

Decoding the embedded message from feature or character coding is very time intensive, since all characters need to be checked and measured to re-gain the secret message. However, a reference text is not required since letters and characters have standard/defined font sizes (e.g. Times New Roman or Arial font size 12).

Data hidden in text documents has a variety of applications, including copyright verification, document authentication, and content annotation. Making copyright information inseparable from the text is one way for publishers to protect their intellectual contents before distribution. Annotation, however, can be used for tamper protection. For example, if a cryptographic hash (a message digest) of the paper is encoded into the paper, it is a simple matter to determine whether or not the text has been altered during transmission by re-converting the message digest into text and compare the content to the 'carrier' text.

The verification, for instance, could be easily performed by an e-mail server, which would reject or verify a message-text depending on whether or not any hidden data is found.

The main goal of text-document watermarking is to make the watermark removal more expensive (in cost and time) or complicated than obtaining the rights from the copyright owner to copy the document (Hartung and Kutter, 1999a). If this goal is achieved, text-document watermarking makes sense.

Advantages:

A major advantage of these text-based watermarking methods is that they can be combined with each other, and can be applied to nearly any kind of text document. Because they use the gaps and free spaces between words and lines, they will go

unnoticed by the reader, since ‘additional white space’ is peripheral to the text. Also, aligning two different watermarked documents to extract the watermark would be useless, since the number of words in lines and the number of lines on a page differ from each other (Berghel and O’Gorman, 1997). Consequently, there is no easy re-calculation of the watermark message possible.

Disadvantages:

However, there are a number of threats and problems associated with these watermarking techniques. A major disadvantage of text document watermarking is that the hidden data cannot be retrieved once the data has been printed (exact measuring of letters is not possible any more since printers have different printing quality which might effect precise measuring). The biggest threat, however, is that the ‘security feature’ can be easily destroyed by manually typewrite the text (or by Optical Character Recognition – OCR), changing the text layout, or using a new character font, line spacing or page setup.

4.1.2 Still Image Watermarking

The majority of reported watermarking research has focused on digital still images as illustrated in Table 2.2. Motivated by the fact that there are so many images available on the Internet which need to be protected, there are a large number of commercial and free image watermarking products and tools available.

The watermarking of still images presents a variety of challenges that arise due to the way the human visual system (HVS) works and the typical modifications and attacks that images are exposed to. Additionally, still images provide a relative small host (i.e. carrier data) in which watermark information can be hidden. For instance, a

fairly typical picture of 200 x 200 pixels offers theoretically up to 40 kilobytes (kB) of data space to hide the watermark (assuming that each pixel of the image could be used to carry one bit of the watermark information) (Bender et al., 1996). Also, it is reasonable to expect that still images will be subject to some modification operations ranging from simple affine transformation (i.e. stretching) to non-linear transformations such as cropping, blurring, filtering, cutting, and compression. Therefore, practical watermarking techniques need to embed as much watermark information as possible and be resistant to as many modifications or attacks as possible.

To demonstrate image watermarking practices, Figure 4.3 below shows a fishing boat where authentication watermarking has been applied. In the original picture (left) the light house, the ships name and its owner have been erased (see modified image in the middle). The right image highlights the areas with bright pixels that have been modified. In this case the image would lose its authenticity because of these modifications. On the other hand, the watermark helps to identify the modified areas.



Figure 4.3 Image watermarking for data authentication
(source: Salzburg, 2009)

The number of still image watermarking publications is far too large to give a complete overview of all proposed techniques in this section. However, most image watermarking techniques share common principles and differ only in parts or single aspects of the three major areas: watermark signal design, watermark embedding, and watermark recovery.

4.1.2.1 Watermark Signal Design

The watermark signal for still images is typically a pseudorandom signal with low amplitude (compared to the image amplitude which is high) and usually with spatial distribution of one information bit (i.e. 1 or 0) over many image pixels (Hartung and Kutter, 1999a).

The watermark signal is often designed as a white (Tirkel et al., 1993; Schyndel et al., 1994) or coloured pseudorandom signal with e.g. Gaussian (Cox et al., 1995), uniform, or bipolar (Darmstaedter et al., 1998; Kundur and Hatzinakos, 1998; Kutter, 1998a; Nikolaidis and Pitas, 1996; Tirkel et al., 1993) probability density function. In order to avoid the embedded watermark being visible, an implicit or explicit spatial (Benham et al., 1997; Kankanhalli et al., 1998; Swanson et al., 1997; Wolfgang et al., 1999) or spectral (Kankanhalli et al., 1998; Piva et al., 1997; Podilchuk and Zeng, 1997; Swanson et al., 1997; Wolfgang et al., 1999; Swanson et al., 1996) shaping is often applied with the goal to attenuate the watermark in areas of the image where it would otherwise become visible.

The resulting watermark signal leaves image pixels unchanged (Darmstaedter et al., 1998; Kutter et al., 1998b), but mostly they are more dense after processing and the pixels of the image to be watermarked are slightly altered (i.e. the picture is getting blurred). The watermarking signal is often designed in the spatial domain, but

sometimes also in transform domain like the full-image discrete cosine transform (DCT) domain (Cox et al., 1995) or block-wise DCT domain (Koch and Zhao, 1995).

4.1.2.2 Watermark Embedding

Watermark embedding is done by adding or signal-adaptive (i.e. scaled) addition of the watermark signal (Barni et al., 1998a; Langelaar et al., 1997a; Nikolaidis and Pitas, 1996) mostly to the luminance channel alone, but sometimes to colour channels (Kutter et al., 1997) of the image as well. The watermark addition takes place in several domains like the spatial or transform domain using discrete Fourier transform (DFT) (Ruanaidh et al., 1996), the full-image DCT, the full-frame DCT (Barni et al., 1998b; Cox et al., 1995; Piva et al., 1997), the block-wise DCT (Benham et al., 1997; Hartung and Girod, 1996; Koch and Zhao, 1995; Langelaar et al., 1997a; Podilchuk and Zeng, 1997; Xia et al., 1997), the wavelet domain (Kundur and Hatzinakos, 1997, 1998; Wang and Kuo, 1998), the fractal domain (Davern and Scott, 1996; Chassery et al., 1998; Puate and Jordan, 1996), the Hadamard domain (Johnson and Biggar, 1997; Ramkumar and Akansu, 1999), the Fourier-Mellin domain (Ruanaidh and Pun, 1997, 1998), or the Radon domain (Hartung and Kutter, 1999a; Wu et al., 1999).

For some proposed methods (Barni et al., 1998b; Chen and Wornell, 1998, 1999; Langelaar et al., 1997a; Wolfgang and Delp, 1997) the watermark generation and embedding are executed at the same time and cannot be regarded separately, especially if the watermark is signal adaptive.

It is often claimed that embedding a watermark in the transform domain (mostly DCT or wavelet) is advantageous in terms of visibility and security (Barni et al., 1998b). While some argue that the watermarks should be embedded into the low

frequencies (Cox et al., 1995; Ruanaidh and Pun, 1997), others argue they should rather be embedded into the medium or high frequencies of an image so that they are not noticed by the viewer (Barni et al., 1998b; Deguillaume et al., 1999; Hsu and Wu, 1997a). For maximum robustness, however, watermarks should be embedded into the same spectral components where the carrier data already populates. For images these spectral components are typically in the low frequencies (Hartung and Kutter, 1999a).

4.1.2.3 Watermark Recovery

The watermark recovery is usually done by some sort of correlation method, like a correlation receiver or a matched filter between the watermarked and original image. Since watermark signals are often designed without knowledge of the carrier data, cross-talks between the watermark and the carrier data is a common problem in image watermarking (Hartung and Kutter, 1999a). In order to suppress cross-talk issues, many proposed schemes require the original unmarked image data in order to subtract necessary information (e.g. frame size) before watermark extraction.

Some proposed methods (Depovere et al., 1998; Kutter et al., 1997; Langelaar, 1996; Schyndel et al., 1994) apply a pre-filter principle instead of subtracting the original signal, while other (Piva et al., 1997) do not suppress cross-talks at all. Yet some other schemes (Barni et al., 1998b) use a more sophisticated detector like a ‘maximum a-posteriori’ (MAP) instead of simple correlation detectors to retrieve the mark.

4.1.2.4 Discussion

Most image watermarking methods are based more or less on the same basic principles: small, pseudorandom changes are applied to selected image pixel

coefficients in the spatial or transform domain. These changes are identified in the extraction stage by some correlation or correlation-like similarity measures. Usually, the number of modified coefficients is much larger than the number of information bits to be encoded. This can be considered as redundant embedding and leads to implicit robustness.

When embedding a mark, the different domains may have substantial influence on the watermark's robustness. Although the spatial watermarking domain is the most common proposed method for still images, it is generally less robust towards noise or lossy JPEG compression. However, a big advantage of it is that the watermark may easily be recovered if the image has been cropped or translated.

Spatial watermarking can also be applied using colour separation. In this way, the watermark appears in only one of the colour bands. This renders the watermark visibly unobtrusive so that it is difficult to detect under regular viewing. However, the watermark appears immediately when the colours are separated especially for offset printing or xerography.

This is less obvious if the frequency domain has been used. Image cropping in the frequency domain results in a substantially large distortion which usually destroys the embedded watermark (Berghel and O'Gorman, 1997). The same is true for the full-frame DCT domain. If a frame block is DCT watermarked, it is important to know the block's exact position for successful watermark extraction.

The wavelet domain has very similar drawbacks as the frequency domain, because the wavelet transform is neither shift nor rotation invariant. This is probably due to the simplicity and efficiency of such methods.

4.1.3 Video Watermarking

A video sequence consists of a series of consecutive and equally time-spaced still images. Thus, the general challenge for watermarking video data should be similar to image watermarking. Therefore, the idea that image watermarking techniques might be directly applied or extended to video data is obvious. However, this is only partly true, although there are a lot of publications on image watermarking which conclude with the remark that the proposed approaches could be also applied to video data (Cox et al., 1996, 1995; Hsu and Wu, 1997a; Busch et al., 1999).

Even though some say that watermarking still images and video data is similar, they are not identical. There are some important differences (including data characteristics and handling) between image and video data which ask for specific methodology for video watermarking (Doërr and Dugelay, 2003). These differences can be narrowed down to four dissimilarities which are: *space*, *frame rates*, *data processing* and *live editing*.

4.1.3.1 Space

The first difference between images and video data is the availability of space for hiding a watermark. Within an image the space to hide a watermark is limited due to the fact that it is 'only one' image with a certain resolution (i.e. number of pixels). In other words, the higher the quality of an image is, the more pixels it has, and therefore more space is available to hide/embed a watermark.

A video sequence, however, can contain several thousand or more images and thus is capable of carrying more watermark information. Even if the resolution of each individual image is poor, the watermark information can be distributed onto several

consecutive images. Therefore, a video sequence offers a much larger ‘hiding space’ for watermark information than a single image.

4.1.3.2 Frame Rates

Another point to consider when watermarking is applied to video is the structure of the video data as a sequence of still images. In contrast to images, video sequences might be more susceptible to particular attacks such as frame averaging, frame dropping, and frame swapping where the watermark could be destroyed (Hartung and Girod, 1996; Swanson et al., 1997). A good watermarking scheme, however, should be able to resist these kinds of attacks, for example, by distributing watermark information onto several consecutive frames. However, it might be desirable to retrieve the full watermark information from a short part of the video sequence.

4.1.3.3 Data Processing

In the context of video the possibilities of attacking or altering watermarked data are multiplied compared to still images. Besides intentional attacks (similar to image attacks), many different non-intentional or non-hostile video methods exist which might harm the embedded watermark. Non-hostile refers here to the fact that even content owners are likely to change or attack the watermarked data unintentionally in order to manage or handle their own intellectual property efficiently. These non-hostile modifications can be due to a wide range of video processing and data transmission procedures that are summarised as follows (Hartung and Kutter, 1999a; Busch et al., 1999):

- Data transmission is very likely to add some kind of noise or disturbance that might change the photometric nature of the video which might have some influence on the embedded watermark.

- Similarly, digital to analogue and analogue to digital conversion (DA/AD conversion) can also introduce some distortions to the video signal when changing the format.
- Another common process to video is to perform a gamma correction in order to increase the contrast.
- In order to reduce video file size (for storage or transmission reasons) content owners often compress their video data.
- Also, spatial filtering inside each frame is often used to restore a low-quality video, while inter-frames filtering (i.e. filtering between adjacent frames of a video) has to be considered too.
- The most common examples of non-hostile attacks on video are changes across display formats (such as 4/3, 16/9 or 2.11/1) and changes of spatial resolution (such as NTSC, PAL, SECAM and other video standards).
- Similarly, customers are also likely to convert their videos from a standard video format such as MPEG-1, MPEG-2 or MPEG-4 to other popular formats such as DivX or AVI.
- Another kind of non-hostile attack might be cut-and-modify operations that a video editor may perform. Cut-and-splice and cut-insert-splice are two very common processing methods used during video editing. Cut-insert-splice is basically what happens when a commercial is inserted in the middle of a movie. Such kind of editing can be seen as temporal editing in contrast to spatial editing.
- Spatial editing refers to the addition of a visual content in each frame of the video stream. This includes, for example, graphic overlay (e.g. logos or subtitle

insertion) and video stream superimposition like in the Picture-in-Picture (multi-screen) technology.

4.1.3.4 Live Editing

Real-time marking of video data can be an additional requirement when creating a video watermarking scheme, but is not necessary for image watermarking. Video frames are sent at a typical rate of 25 frames per second (or 50-60 for High Definition) to obtain a smooth video stream. Therefore, the watermark embedder should be able to handle such high rates. However, in the context of real-time broadcast monitoring, the detector should also be able to detect and decode the hidden watermark in real time.

In order to meet real-time watermarking and detection requirements (i.e. speed, frame rates per second, etc.) the complexity of video watermarking algorithms should be as low as possible (to be able to keep up with the frame rates), but still thorough enough to mark every frame in the video sequence, while being able to withstand different types of attacks (Kalker et al., 1999).

4.1.3.5 Video Watermarking Techniques

It can be observed from the literature that about half of the video watermarking techniques and methods are proposed for compressed video data, while the other half is conducted on un-compressed video data. Watermarking on compressed video makes sense, since video data is stored in a compressed stage most of the time in order to save some storage space. As a result of this, some watermarking schemes have been specifically designed to embed the watermark information directly into the compressed video stream.

Regardless of video format (i.e. uncompressed or compressed), published watermarking techniques are based on one of the following concepts to embed information by modifying one or more of the video's characteristics:

- Spatial domain: Watermark embedding and detection are performed on spatial pixels values (i.e. luminance, chrominance, colour space) on video images, or on the overall video frame characteristics (Linnartz, 1998; Darmstaedter et al., 1998; Bush et al., 1999; Kalker et al., 1999).
- Feature or salient point watermarking by modifying geometric properties of the video frames (Dittmann et al., 1998b).
- Frequency domain: Spatial values of the video data are transformed with techniques like Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT), and Wavelet Transform to embed the watermark information (Hartung and Girod, 1996, 1997; Hartung et al., 1998; Hsu and Wu, 1997a; Langelaar et al., 1997b; Deguillaume et al., 1999; Swanson et al., 1997, 1998a; Langelaar and Lagendijk, 2001; Langelaar et al., 1998).
- Format specific approaches: The structure elements of MPEG-4 data or motion vectors, like Facial Animation Parameter (FAP), are also used to anchor a watermark signal (Jordan et al., 1997; Kim et al., 1999; Swanson et al., 1998a).
- For fragile video watermarking, the watermark signal can be anchored in the video data by manipulating sensitive video elements like the Last Significant Bits (LSB) to detect changed and manipulated regions of the video sequence.

4.1.3.6 Discussion

The proposed methods span a wide complexity range from simple to extremely complex including, e.g. wavelet transforms and models based on the HVS. In

general, the more complex the methods and techniques used to embed the watermark message, the higher the watermark's robustness.

Also, since image and video watermarking share similar techniques for watermark signal design, embedding and recovery, the characteristics of the embedded watermarks are also similar. Since the qualities and behaviour of image watermarking are already mentioned in section 4.1.2, there is no need to discuss the features that are common to still image and video watermarking again. Therefore, only unique characteristics and threats to video watermarking are discussed here.

For instance, in video watermarking some schemes apply extended image watermarking technology to hide information in the individual video frames. Unfortunately, this means that each frame has a distinct watermark unrelated to the preceding and following frames, which may be visually very similar. A sophisticated attacker, however, can take advantage of this by averaging video frames to detect the watermark signal and remove it or duplicate it (basic principles of collusion attack).

In the case of real-time marking, watermark data rates for video data are between a few bits per second and a few hundred bits per second. To achieve robust video watermarking, rates do not need to be higher than a few bits per second to a few dozen bits per second, which is sufficient for most video applications including DVDs (Hartung and Kutter, 1999a).

Last but not least, the biggest influence or threat to video watermarking might be the numerous different data processing and modification applications that are performed on video data (see sections 4.1.3.3 and 4.1.3.4). When comparing modification processes of video data to other media, one can see that they are more diverse and

sophisticated for video. Therefore, these modifications do not only represent a higher threat to video data, but also need to be considered when developing a new video watermarking scheme.

4.1.4 Audio Watermarking

It is often stated that audio watermarking is similar to image and video watermarking (Hartung and Kutter, 1999a; Gomes et al., 2003; Wang et al., 2004). However, audio, video and still image data differ significantly from each other. First of all, image and video data are two dimensional data which not only provide more ‘space’ to hide watermark information, but also more possibilities for distortions and attacks (see sections 4.1.3.3 and 4.1.3.4) that might affect the watermark’s integrity. Audio data, on the other hand, do not need to deal with these types of attack, since audio signals are ‘only’ one-dimensional. Second, audio signals are represented by far fewer samples per time interval than video sequences (Hartung and Kutter, 1999a). Consequently, the amount of information that can be embedded robustly and inaudibly into audio data is much lower than for visual media.

Probably the biggest challenge for audio watermarking is the human audible system (HAS) which is much more sensitive to changes than the human visual system (HVS) (Arnold et al., 2003; Cox et al., 2002b). Therefore, inaudible watermarks are much more difficult to achieve than invisible watermarks.

The core requirements that an audio watermarking system must fulfil are application-dependent and can sometimes conflict with each other (Gomes et al., 2003). These can be summarised as follows:

- *Inaudibility*: watermarking should not degrade the quality of the music/ sound.

- *Robustness*: the watermark should resist any transformation applied to the audio data (e.g. to wind, cut-in-splice, duplicate, compression, filtering, etc.).
- *Capacity*: the watermark should be large enough to store all necessary information. This can however conflict with *Inaudibility* – a trade-off must be found.
- *Reliability*: watermark data should be embedded and extracted with low error rates to avoid false positive detection.
- *Low complexity*: watermarking and extraction algorithms should not be excessively time-consuming and complex to enable real-time applications.

The robustness of watermarks to signal-processing operations such as filtering, re-sampling or coding, and A/D – D/A conversion is a basic necessity for audio watermarking. In the case of copyright protection, robustness to malicious attacks is also required, especially where a part of the audio signal might be deleted. In this case the watermark should still be detectable within a fraction of the carrier signal and still carry the whole watermark information. However, for integrity-verification applications (e.g. testimonies recorded before a court), the watermark must no longer be recognised when audio content is modified in any way. In that case, robustness is no longer required; on the contrary, the watermark must be fragile.

Similar to video watermarking, digital audio watermarking techniques and approaches can be classified according to the domain where the actual watermarking takes place. The common techniques applied to audio watermarking techniques can be described as *Time Domain*, *Compressed Domain*, and *Frequency Domain* (Hartung and Kutter, 1999a). Also, some other audio watermarking schemes are described and discussed in the following sections.

4.1.4.1 Time Domain

Embedding a watermark within the time domain involves challenges related to fidelity and robustness. Shaping the watermark before embedding enables the system to maintain the original audio signal fidelity and renders the watermark inaudible.

Bassia and Pitas (1998; 2001) proposed a method in which the watermark signal is modulated and embedded into the original audio signal by using a low-pass filter to reduce distortions. The carrier audio data is divided into segments and each segment is watermarked separately by embedding the same watermark. Although the original data is not required for detection, this method shows robustness against audio compression, filtering, time-shifting, and cropping.

The different audio watermarking systems by Boney et al. (1996) and Swanson et al. (1998b) use the HAS masking effects to shape the watermark signal. Here, the watermark shaping is performed in the frequency domain to create a noise-like sequence, but the actual watermark insertion is executed in the time domain. Both schemes are robust to time-shifting, cropping, filtering, re-sampling, or data compression, although the original audio data is required in the watermark detection process.

4.1.4.2 Compressed Domain

Another way of embedding the watermark is to use the compressed domain. This approach is applied directly on compressed data (i.e. MPEG audio bit stream).

For instance, Qiao and Nahrstedt (1999) developed a technique which uses a data encryption standard (DES) algorithm to generate non-invertible watermarks, and then hides the watermark message with the help of ‘level changes of scale factors’

into MPEG audio frames. To detect and extract the watermark message the original audio signal is required.

Other approaches have been developed by Lacy et al. (1998), Neubauer and Herre (1998; 2000a; 2000b), and Swanson et al. (1998b). They apply a lossy compression algorithm that uses the special nature of the HAS to hide information into MPEG audio data. In other words, this mechanism removes perceptually irrelevant parts of the audio signal (that are inaudible to the human ear) and creates specific replacement signals which carry the watermark message. Then, the newly created replacement-watermark-signals are inserted directly in the MPEG bit stream generated by an audio coder. The embedded watermark can be detected both in the uncompressed and the compressed version of the carrier audio data. To locate and extract the watermark, the original unmarked audio data is required where the similarities between the two versions are measured and converted back.

4.1.4.3 Frequency Domain

Instead of adding the watermark signal into the original signal domain (i.e. to the actual carrier audio signal), it may be also added in some transform domain (i.e. amendment of the original audio signal). Almost any transform domain could be used, but most work has concentrated on Fourier, Wavelet, and Spread-spectrum domain which are thought to be either readily computable or offer beneficial mappings to human perception systems, or both.

Arnold (2000, 2001) established an audio watermarking technique that uses a statistical algorithm in the Fourier domain. The audio carrier signal is broken into frames, and in each frame one bit of the watermark message is embedded. For the embedding itself, each frame is transformed by using Discrete Fourier Transform

(DFT). His method is based on the patchwork algorithm proposed by Bender et al. (1996), however, it does not require the original audio data in the detection process.

Another method for audio signal watermarking in the wavelet domain, using a “patchwork” algorithm, was proposed by Kim et al. (2002). In this scheme a binary watermark is embedded bit by bit into splitted audio data blocks. Then, the watermark bits are locally repeated for the purpose of robustness. Also a number of supporting bits are added in front of watermark bits to locate the point where the watermark is embedded into the audio signal. These supporting bits are called synchronisation bits. Since this scheme carries necessary information about the watermark location, it can be detected and extracted without the knowledge of the original audio signal. The authors claim that this watermarking system has a high performance in data synchronisation and is able to resist time shifting attacks.

Other proposed watermarking schemes use ideas borrowed from spread-spectrum radio communications. For instance, Boney et al. (1996) embedded a watermark by adding a pseudo-noise (PN) signal with a low amplitude to the host audio data. This PN signal has been filtered in several stages in order to exploit long-term and short-term masking effects of the HAS. The PN signal can later be detected by using a correlation receiver or matched filter.

Cox et al. (1997) and Garcia (1999) reported a useful audio watermarking technique by simply spreading the watermark signal over many frequency components of the host audio data so that the trace of the mark is very small or barely detectable. They claim that, in order for the watermark to be robust, it must be placed in perceptually significant regions of the audio data despite the risk of potential fidelity distortion.

Bender et al. (1996) also proposed a watermarking technique applicable to audio data in the spread-spectrum domain. Their watermarking scheme uses bi-phase shift keying and a pseudorandom noise. Their scheme introduces a perceptible noise to the original audio signal by using adaptive and redundant coding where the perceptible noise can be reduced.

4.1.4.4 Other Audio Watermarking Techniques

A number of audio watermarks have been proposed that do not fit into any of the present categories. For example, rather than adding the watermark to the audio signal in the time domain, Garcia (1999) performed the embedding operation in the frequency domain by directly replacing spectral components of the audio signal.

The schemes created by Gomes et al. (2000) and Furon et al. (2000) use the same domains as Garcia (1999) but with different coding keys for watermark insertion and detection.

Tilki and Beex (1996) have developed a watermarking system for interactive television where the watermark is hidden inside the audio signal. In this application they embed the watermark information directly into the audio component of a television signal, where the mark is partitioned in blocks of 35 bits. Each block of bits is then modulated and added to the host signal by using a sinusoidal carrier of a specific frequency and amplitude (frequency domain). Their research shows that this scheme is robust against room noise and video tape recording.

4.1.4.5 Discussion

Audio watermarking systems have a number of differences. These differences can be demonstrated when evaluating the performance of the various systems or their suitability for a specific application.

- 1) Some audio watermarking systems require the original audio data, or information derived from it, to be able to detect and decode the watermark message. This, however, has the effect that a large amount of original data needs to be stored and searched, which will consequently slow down the detection process.
- 2) In order to keep watermark security high, the mark is embedded into selected regions of the audio signal using a watermarking key. This watermarking key can be used either to create the watermark itself (a random sequence of bits generated by some sort of algorithm) or to embed and detect the watermark, or for both (dependent on the transform domain). The watermarking key is either provided by the copyright owner or is created both from information provided by him/her and information derived from the original audio data. In this case the original audio data will be required in the detection process for key generation purposes.
- 3) During the embedding processes the original audio signal is divided into several frames, where each frame is watermarked separately. Some watermarking systems embed the same watermark into a number of frames to enhance watermark robustness, while others embed various marks into the frames (for fragile marking). However, a major difficulty in audio watermark detection and extraction is the need for precise synchronisation (Kim et al., 2002). To be successful, the detector must know the exact starting and finishing point of each hidden message in order to perform a precise and accurate extraction.
- 4) However, probably the biggest challenge of creating and embedding a watermark into an audio signal is the sensitivity of the HAS (i.e. the watermark must be inaudible to the human ear). It is usually convenient to use a domain in which the properties of the human perceptual system can best be bypassed (Walker, 2004).

However, this is not a necessary limitation when developing a watermarking system, although the choice may have a significant impact on the implementation complexity of the system. Masking characteristics of an audio signal can be used for this purpose. Psychoacoustic MPEG model is commonly used to calculate a masking threshold that is appropriate for “weighing” the watermark. This technique, as well as other techniques, uses the original audio signal in modulating the watermark where the amplitude of the watermark signal is controlled by the amplitude of the audio signal. However, this watermark shaping process might affect the existence of the watermark in the cover work. In other words, the possibility of a false negative detection will be increased.

4.2 Watermarking Approaches for Three-Dimensional Data

The nature of shapes and graphical data representing three-dimensional objects is completely different from that of any other media data type (e.g. image or audio data). For instance, a mesh based representation of a 3D object consists of a graph having vertices as its nodes, joined by edges that define the boundary of polygonal surfaces in a three-dimensional space, while a still image is an accumulation of pixels on a regular image-grid on a two-dimensional plain level (Chang et al., 2003; Bors, 2004). Therefore, simply applying image watermarking technologies to three dimensional graphical data is not going to work.

To be able to watermark three dimensional shapes new/different approaches and methodologies have to be adopted to achieve effective 3D watermarking. In the following sections a number of watermarking schemes and approaches for 3D models are presented and discussed. The literature is organised by grouping the

methods together according to the three dimensional object's representation. These groups are: *surface* and *solid representations*. Following this, watermarking techniques for different types of three dimensional data are described and discussed. Finally, all presented techniques are compared with each other in the discussion section at the end of this chapter, and a gap in the literature is identified.

4.2.1 Watermarking Techniques for 3D Models Represented as Surface

In this section watermarking schemes and approaches for 3D objects represented as polygonal meshes, Bézier, and NURBS surfaces are described and discussed in the context of reported watermarking methods.

4.2.1.1 Approaches for Polygonal Mesh-type Data

Some of the earliest work in the field of three dimensional digital watermarking was reported by Ohbuchi et al. (1997a, 1997b, 1998a, 1998b, 2001). They proposed a large variety of techniques and methods for embedding data into three dimensional polygonal mesh models. Some of their watermark embedding algorithms are based on topological and some on geometrical modifications. For instance, their Triangle Similarity Quadruple (TSQ) embedding algorithm modifies vertex coordinates of four adjacent triangles to encode a watermark (in form of bits) by setting the value of ratios between edge lengths of the triangle group. The TSQ algorithm can be classified as a blind watermarking scheme because it does not require the original unmarked 3D model in the extraction stage. The embedded watermark can only be used to prove the presence of a mark since it does not carry any text message. The authors report that watermarks produced by the TSQ algorithm withstand translation, rotation, and uniform-scaling transformations of the host polygonal meshes.

However, the embedded message can be destroyed by randomisation of coordinates, or by topological modification of the surface structure such as re-meshing.

Their second method, called Tetrahedral Volume Ratio (TVR), randomly selects specific tetrahedron formations from the mesh model's surface and embeds the watermark information (converted to bits) in the volume ratio of consecutive tetrahedrons by modifying their vertices. In other words, the size of modification for a vertex depends on the bit-message content. Compared to TSQ, the message embedded with the TVR can be recovered and converted back into text during the detection stage. The TVR algorithm is also a blind watermarking scheme, but with the added benefit that the watermark withstands local deformation and cropping. However, the watermark can be destroyed by attacks which cause topological modifications such as re-meshing, polygon simplification, vertex coordinate randomisation, and geometrical transformations.

The third blind watermarking scheme proposed by Ohbuchi et al., called Triangle Strip Peeling Symbol (TSPS), is based on topological embedding principles. This algorithm "peels" off a triangle-strip from a given mesh model into which the watermark message is embedded as a bit-string. In other words, the watermark bit-string determines the shape of the triangle-strip. The embedding principle uses the adjacency relationship between a pair of triangles in the strip, and each pair encodes a single information bit (i.e. 1 = left adjacent triangle, 0 = right adjacent triangle; see example in Figure 4.4). During the detection process it is possible to recover the embedded watermark message without the unmarked original (although the start of the strip needs to be specified/known). The embedded information is resistant to affine transformation or arbitrary geometrical transformation, but can be destroyed

by topological manipulations such as polygon simplification, re-meshing or noise addition.

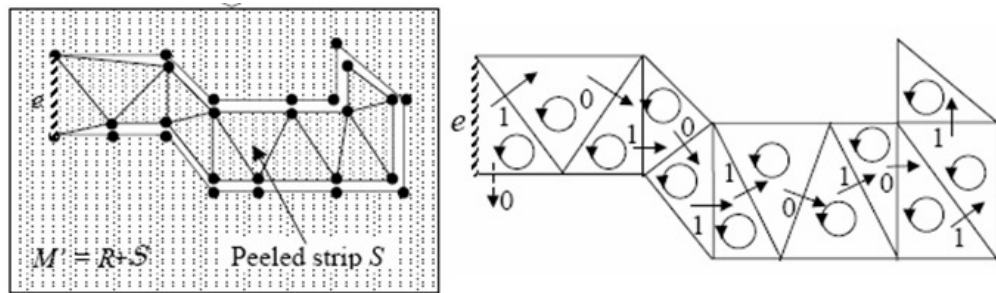


Figure 4.4 Information embedding with TSPS
(adapted from: Ohbuchi et al., 1998c)

Wavelet transformation techniques, which belong to the transform domain, have also been applied to the digital watermarking of three dimensional shapes. Kanai et al. (1998) are the first to report the use of wavelet transform as a watermarking method on 3D polygonal mesh models. Their robust but non-blind detection algorithm first decomposes a 3D polygonal mesh by using lazy wavelets induced on 3D polygonal meshes. Then a binary image watermark is embedded in the wavelet coefficient vectors (or vertex coordinates) at various resolution levels of the multiresolution representation. This makes the embedded watermark imperceptible and resistant against affine transformation, partial resection, and random noise added to vertex coordinates, but needs the original cover data in the watermark retrieval stage. The embedded message can only be used to prove the presence of a watermark (i.e. the watermark does not carry any meaningful text information).

Similar to Kanai's technique, Praun et al. (1999) also proposed a non-blind detection, robust mesh watermarking algorithm that works in the transform domain (i.e. multiresolution analysis) and is applicable to polygonal meshes that have arbitrary vertex connectivity. This method modifies the shape of the mesh by using a spatial

kernel to embed information in the low-frequency components (i.e. visually the most important components) of the shape by imperceptibly displacing the vertices. This is achieved by recreating the connectivity of the object's original mesh by means of mesh alignment followed by re-sampling. The embedded watermark message is a cryptographic random number created by a hash function (like a MD5) which is extracted in the watermark recovery stage and can be converted back to readable text. Tests by the authors show that the resulting watermarks are resistant against similarity transformation, smoothing, additive random noise, cropping, mesh simplification, vertex reordering, insertion of a second watermark, and other operations that preserve shape but modify vertex connectivity.

Benedens et al. (1999a, 1999b, 1999c, 2000b) also developed a number of digital watermarking methods for the embedding of information into 3D models, most of which are based on geometrical transformations. Their system, often referred to as the GEOMARK system, is based on three different algorithms for watermarking three dimensional polygonal meshes: Vertex Flood Algorithm (VFA) – suitable for fragile public watermarking, the Affine Invariant Embedding (AIE) – realising affine invariant watermarks, and the Normal Bin Encoding (NBE) – realising robust watermarks against polygon reduction.

To encode arbitrary watermark bits, the VFA algorithm modifies the vertices of a mesh model so that their distances to the centre of the surface have a specific value. This algorithm solely operates on mesh-vertices and does not take further topological relationship into account. Neither does it require connectivity of the faces from the carrier mesh model. A watermark created by the VFA method is suitable for blind detection and can survive uniform scaling, rotation, cropping and translation into

different formats. On the other hand, they are not robust against complex geometry modifications, non-uniform scaling and topology alterations, and can only be used to prove that a watermark is present (i.e. for verification purposes).

The AIE is also based on the alteration of surface normals similar to the VFA method. However, the embedding algorithm is based on the local application of the Nielson Foley norm and deals only with infrequently appearing non-triangular faces to embed affine invariant watermarks. The arbitrary bit watermark (used for data authentication) embedded with this technique is robust against cropping and general affine transformations, but are lost in re-meshing operations and after mesh simplification. Unfortunately, this technique is mainly of academic value because watermarks do not survive quantisation of vertex coordinates to less than 6 decimal digits (Benedens, 2002).

The NBE approach, which is based on similar alteration of surface normals like the AIE technique, modifies the models vertex normals to store the information. This scheme embeds a 1-bit random watermark into each vertex coordinate and does not require the location of the mark, except the secret key in the recovery stage (i.e. blind watermarking to recalculate the watermark bit). The watermarks created by this approach show robustness against mesh simplification methods. One drawback of the algorithm, however, is the amount of pre-processing needed before the watermarking algorithm can be applied.

The novelty of the GEOMARK system is the combination of all three algorithms to obtain a watermarking scheme that is robust against randomisation of vertices, mesh altering, and polygon simplification operations.

Yin et al. (2001) reported a non-blind robust mesh watermarking algorithm that works in the transform domain. It is based on the multiresolution decomposition of polygonal mesh shapes that separates a mesh into its elements (i.e. coarse feature sequences) by repeatedly applying local smoothing operations. Arbitrary binary watermark information is then embedded into the least significant bits (LSB) of vertex coordinates using additional order bits in the pre-processing phase. Experimental results show that their watermarking algorithm is robust to various attacks, including vertex reordering, noise addition, mesh-simplification, filtering and enhancement, and cropping. This type of watermark is mainly used to show that a watermark is present, since the watermark message is meaningless.

Harte and Bors (2002a, 2002b) proposed a blind watermarking scheme for 3D mesh models and objects through altering the location of certain vertices. Their watermarking technique is essentially of a geometrical nature and therefore belongs to the transform domain. The two main steps of the watermarking algorithm can be described as follows: In step one a chain of vertices and their neighbouring vertices are selected from the mesh object and are ordered according to a specific distance criterion. In the second step, bounding volumes are modelled from the selected neighbouring vertices and the corresponding vertices are moved inside the bounding volume when embedding a '1' bit, or outside for embedding a '0' bit. In the detection stage, the embedded bits are retrieved by checking their relative location with respect to the bounding volumes. Due to the geometrical nature of the watermarking algorithm, the watermark can be recovered after scaling, rotation or combinations of geometrical transformations, and is used to confirm that the data has been watermarked.

Similar to Beneden's approach Aspert et al. (2002) developed a method to embed information into 3D polygonal mesh models. Their algorithm slightly changes the position of vertices of the mesh composing the model and embeds the information in the length of the normal vectors of the defining vertices. The watermark message to be embedded is created by a MD5 hash function and can be fully recovered and reconverted in the extraction stage. This method is robust to simple geometric transformations such as scaling, translation, and rotation. However, this scheme will not resist stronger attacks such as re-sampling, compression, and model simplification. Furthermore, to extract the hidden information successfully, the original unmarked mesh-model is needed in the detection stage, i.e. this is a non-blind watermarking method.

In 2004 Daras et al. presented a robust blind watermarking technique for 3D models using a cylindrical integration transform (CIT) method in which an arbitrary bit-watermark is embedded into the nodes by modifying the location of a subset of the 3D model's vertices. The embedded watermark mainly serves as prove of authentication, but can also be used as a link to an identifier for 3D model search. The extraction of the watermark can be done without the original unmarked model. Since the process is so fast and accurate, it can be effectively used in real-time 3D model content-based search and retrieval applications. Experiments by the authors show that their watermarking technique is not only robust to geometric distortions such as translation, rotation and uniform scaling, but also to other point and vertex rendering and reordering attacks.

Uccheddu et al. (2004) also demonstrated a multiresolution watermarking algorithm designed to work on semi-regular meshes with subdivision connectivity. Their

proposed blind algorithm embeds the watermark information (random numerical values) by modifying existing vertex coordinates of a 3D model and introducing new vertices to the model in the wavelet domain. The watermark is recovered by means of correlation detection (i.e. computing the correlation between the watermark signal and the inspected mesh) and does not require the original mesh model in the extraction stage. The embedded watermark, used to prove the presence of a watermark, is robust to geometric transformations such as rotation, translation, uniform scaling, and other attacks such as noise addition, smoothing, and cropping. However, this method is only applicable to sub-divisions of surfaces.

Maret and Ebrahimi (2004) proposed a blind data hiding tool to embed information into the geometry of 3D polygonal mesh objects that is suitable for content annotation and secret communication applications. Their algorithm consists of three stages: In the first stage, the model geometry is transformed into a similarity-invariant⁵ space resulting in a non-uniformly sampled function of a sphere unit curve. Stage two modifies the resulting samples in order to embed the message. The final and third stage computes the necessary vertex modifications to the model geometry, and the information is embedded as a bit stream into the projection curve of a sphere unit. While the message can be extracted after basic geometric transformations such as translation, scaling, and rotation, it does not resist more invasive modification such as surface subdivision, simplification and compression. This scheme is capable to recover and recreate the embedded message in the extraction stage.

⁵ In mathematics, similarity invariance is a property exhibited by a function whose value is unchanged under similarities of its domain. For instance, f is invariant under similarities if $f(A) = f(B^{-1}AB)$ where $B^{-1}AB$ is a similarity of A .

Wu and Cheung (2005a; 2005b) developed a fragile watermarking scheme to authenticate 3D mesh models that is similar to Benedens & Busch's (2000b) approach. Within this method, the watermark information is adaptively embedded into the mesh geometry by slightly adjusting vertex positions while the mesh topology remains unchanged. By applying their so called Enhanced Centroid-Distance-Ratio (ECCR) algorithm, the proposed method can be enhanced to detect unauthorised modifications on the carrier mesh model. The watermarking process is conducted in the spatial domain and applies to the whole mesh surface on the model without any restrictions. The embedded blind watermark (arbitrary bit data) applied by this approach is invariant to translation, affine transformations, rotation and uniform scaling, but sensitive to geometrical or topological modifications on the watermarked model. Also, its main purpose is to prove the presence of a watermark and carries no text information.

Denis et al. (2005) proposed a robust watermarking algorithm for polygonal mesh data applied on compressed 3D objects. Similar to Ohbuchi's (2002) method, they modulate the object's vertex coordinates in the transform domain to insert the bit message. For each coordinate, one bit is embedded, so for each vertex, 3 bits can be concealed (i.e. one bit for each x, y, and z coordinate). The extraction operates in a non-blind manner (i.e. the original unmarked object is necessary to locate and extract the message), and is based on the same principle as the insertion process: the spectral decomposition is applied to the initial mesh and to the watermarked mesh simultaneously. Robustness has been achieved against noise addition, subdivision and re-meshing attacks. By the nature of this approach, however, it introduces small changes to the model's overall shape.

Alface and Macq (2005) presented a robust blind watermarking scheme for 3D mesh objects based on automatic selection of feature points through a multi-scale estimation of the curvature tensor field. These feature points are used to build a partition of the mesh shape that is resistant to common 3D watermarking attacks. Their spectral domain watermarking scheme proceeds by first partitioning the mesh shape using a geodesic Delaunay triangulation of the detected feature points, then the concerned vertices are re-parameterised to contain the arbitrary bit message. This method shows good preliminary result as it resists affine transformations, noise addition, smoothing, crop and sampling changes, although its robustness is doubtful against re-meshing attacks.

Murotani and Sugihara (2005) demonstrated a robust watermarking method that adds watermarks to 3D polygonal meshes, based on the generalised singular spectrum analysis (SSA). The structure of the 3D polygonal mesh is computed by the singular decomposition of the trajectory matrix, and the arbitrary binary watermark message is then embedded into the vertex coordinates as a multiplying factor (i.e. the x, y, and z coordinates are multiplied by the binary watermark message values). The watermarks embedded by this algorithm are robust against similarity transformations such as rotation, translation and uniform scaling. They are also resistant against random noises added to vertex coordinates. To identify and retract the embedded message (used to confirm the watermark's presence), no unmarked original data is necessary. Therefore, this scheme can be considered a blind watermarking method which modifies the overall shape of a 3D model.

Zafeiriou et al. (2005) published two methods suitable for blind 3D mesh object watermarking applications. In both methods pseudo-random numbers are embedded

as watermarking signals in the 3D mesh object by deforming its vertices geometrically (altering vertex components/coordinates), but without altering the vertex topology. Prior to watermark embedding and detection, the object is rotated and translated so that its centre of mass and its principal component coincide with the origin and the z-axis of the Cartesian coordinate system. Their first method, the so-called Principal Object Axis watermarking (POA) scheme, is robust against rotation, translation, and uniform scaling. The second one, the so-called Sectional Principal Object Axis watermarking (SPOA) scheme, is additionally robust against mesh simplification attacks. Both algorithms, however, are based on principal component analysis, and thus both algorithms will fail against cropping attacks, which can cause severe alteration to an object's principal axis.

Bors (2006) reported another technique for watermarking three dimensional graphical objects, represented as 3D meshes. The watermark information, provided as binary code, is embedded by means of small local geometrical perturbations while maintaining the local connectivity and without affecting the mesh topology. The scheme proposed here is blind, in the sense that in the detection stage the original stego model is not required for the recovering of the encoded message. Watermarks embedded using this methodology do not affect the mesh topology or the vertex interconnectivity, and are unaffected by rotation, translation and scaling of the objects. However, certain changes in the mesh topology (e.g. re-meshing or mesh simplification) can affect the watermark's robustness.

Chou and Tseng (2006) proposed a blind fragile watermarking scheme based on the sensitivity of vertex geometry of a 3D polygonal mesh model. To overcome casualty

and convergence problems⁶ they use a multi-function vertex embedding and adjusting-vertex method to create a watermark signal. In the first step, they select a set of vertices from the original mesh model, where these selected vertices and their neighbouring vertices cover the whole CAD model (vertex strip). Then the three coordinates (x, y, and z) of each marked vertex are assigned and modified (relocated) to hide the bit-watermark in the embedding stage. In the watermark recovery process the carrying vertices are identified and matched, and then the watermark signal is extracted by “following” the neighbouring vertices (i.e. the vertex strip). This proposed scheme can detect, and locate, unauthorised modifications even if several changes are made to a model at the same time. Watermarks embedded with this algorithm are robust to cropping, vertex decimation, and Boolean modification.

Numerous other authors such as Corsini et al. (2003b), Yeo and Yeung (1999), Lin and Liao (2005), Wu and Cheung (2005a), Quan and Xiaomei (2006), Rušinović and Mihajlović (2005), Li et al. (2004), Garcia and Dugelay (2003), Koller et al. (2004), Koller and Levoy (2005), Funk (2004), Ohbuchi et al. (2002), or Benedens (2002) have also proposed 3D watermarking methods, techniques and schemes with only minor differences or variations to the above described mesh based schemes. However, since the fundamentals of mesh watermarking have been described they will not be discussed further here.

⁶ Two problems frequently arise during the embedding process for mesh data: the causality problem and the convergence problem. The causality problem arises while the neighbouring relationship of a former processed vertex is influenced by the perturbing of its latter processed neighbouring vertices. The convergence problem means that the original model has been heavily distorted before some vertices reach the predefined relationship.

4.2.1.2 *Approaches for Bézier-type Data*

Chadwick et al. (2003) were amongst the first who developed a method for watermarking Bézier surfaces by making small changes to the control points of the Bézier curve. The small changes slightly adjust the geometry of a surface without visually degrading or altering the quality of the model. The scheme is capable to insert 16 bits of random watermark message in a surface-grid of 4 x 4 for watermark authentication purposes. To detect the watermark's presence the original model is required for the statistical correlation evaluation. Therefore, this scheme has non-blind characteristics. Tests by the authors show that watermarked surfaces that underwent standard transformations like rotation and translation still contained the watermark bits.

Nagahashi et al. (2004) presented a non-blind method for embedding a text message in the form of ASCII characters into Bézier polynomial patches. This watermarking method is model-altering since the watermark is embedded by subdividing the original Bézier patches, i.e. new data points are added to the boundary curves of the original patch which carry the message. Moreover, this method is also shape-altering because a boundary modulation was introduced to the sub-patches to increase robustness. The reported test results show that this method can provide a shape-preserving watermark with low robustness or shape-altering watermark with increased robustness to translation, rotation and model transformation attacks.

Hu et al. (2008) proposed a digital map watermarking scheme for data authentication purposes where parts of the map were reconstructed using Bézier curve segments as the watermark carrier. To embed the arbitrary watermark bits robustly, they apply spread spectrum technique that alters the coordinates of selected sample points on the

Bézier B-spline curves. Test results by the authors show that their scheme is robust to scaling, rotation, affine transformation and collusion attacks. However, to detect and extract the mark from the curves the original unmarked Bézier curves are required.

4.2.1.3 Approaches for NURBS-type Data

In the field of 3D NURBS watermarking Ohbuchi et al. (1999, 2000) were again amongst the first to develop an appropriate method for this type of data. Their scheme focuses on providing data authentication, tamper-detection, IP protection and other security related operations for geometric 3D CAD models. The approach introduced is called the “*rational-linear re-parameterisation*” algorithm, where random bit-watermark values are embedded by re-parameterisation of NURBS knots and control points. The method preserves the exact geometric shape of NURBS curves and surfaces, as well as the CAD data size. However, the watermark embedded can easily be destroyed or removed by re-parameterisation and re-approximation actions of the curves and surfaces. The original unmarked model is required in the detections stage to extract the embedded watermark.

Oliver Benedens (2000a) also presented a scheme for embedding publicly readable watermarks (blind detection) into 3D models consisting of NURBS surfaces. His scheme hides affine invariant watermarks (in the form of a bit-stream) by displacing vertices (i.e. control points of the curves) of NURBS surfaces. The algorithm uses the volume of two tetrahedrons as an embedding feature, and is based on applying mesh simplification processes. The scheme described by Benedens, which is mainly designed for authentication purposes, enhances robustness with respect to affine transformations and, in particular, vertex randomisation attacks.

Lee et al. (2004) proposed a non-blind watermarking algorithm for 3D non-uniform rational B-spline (NURBS) graphic data. Instead of directly embedding bit-data into the parameters of NURBS, the scheme hides the watermark information into 2D virtual images extracted from parameter sampling of 3D NURBS model (the pixels of which represent the coordinates of surfaces for the given parameters) by using an existing 2D watermarking algorithm for images. As a result, the proposed watermarking approach can embed information into more places of the surface than other conventional 3D watermarking algorithms, while preserving the data size of the model. According to their experiments, this scheme is robust to attacks on knot vectors such as control points modification, knot vector modification and surface approximation.

4.2.2 Watermarking Techniques for 3D Models Represented as Solids

In the field of watermarking solid objects only few approaches have been reported. These mainly focus on constructive solid geometry (CSG) models, which is surprising given its rarity in commercial CAD systems.

For instance, Fornaro and Sanna (2000) proposed a method that is intended for verification purposes and is based on digital signature techniques. Their scheme computes the watermark message from selected attributes of the model (e.g. topology, colours, textures, etc.) using a standard encryption operation (a hash function) for encoding the information. Then the watermark-digest is added to the CAD object's CSG tree. Since this is a blind watermarking technique neither the watermark information nor the original model are required at the verification stage. However, a verification key is necessary to verify the signature and the author's certificate. The advantage of the method is that, because it does not alter the model's

structure or appearance, it is compatible with copyright protection techniques and can be used several times to have a multiple watermark. Moreover, no pre-processing of the model is required before this method can be applied, and all the attributes and the structure of a model can be protected. However, type, shape and position of the watermark have to be known in the detection stage, which makes this approach not entirely a blind scheme. Furthermore, the watermark can be removed by a computer programme, and without the watermark message the authentication is no longer possible. Finally, neither the watermark nor the model can be modified or altered without avoiding an inconsistency with the watermark itself since they are generated from each other.

Another watermarking technique on 3D solid models was presented by Djaghloul (2007). Similar to Fornaro and Sanna's technique, this blind watermarking scheme uses CSG modelled objects as carrier data. The whole watermarking and retrieval process can be achieved in three major steps: in step one, empty bounded volumes (four spheres) are created and united with the CSG tree (described as the 'right leaf') which do not affect the original object's appearance and will not be deformed by any topological or set operators. Then the binary-watermark is placed into this newly created bounded volume by replacing nominated spheres coordinate vectors by 1 and 0 using multiresolution techniques based on wavelets. In the watermark detection process the CSG tree is searched for added branches and leaves and only the 'right leaf' is checked for hidden watermarks. Since the location of the watermark is obvious, no original model is needed for watermark extraction. However, everyone with appropriate tooling, knowledge and skills would be able to find and remove the "protective" feature by removing the concerned leaf. According to the author's test

results this approach is robust against rotation and transformation attacks and is not affected by scaling operations. However, given these characteristics the method is very fragile.

To the best of the author's knowledge, no existing work dealing with digital watermarking of solid 3D boundary representation (B-rep) models have been reported in the literature.

4.2.3 Alternative 3D Watermarking Techniques

As already highlighted in section 3.4 there are different ways of representing the three dimensional objects that could potentially act as watermark carriers. At the same time, there are also some other ways for hiding watermark information. Beyond the applied watermarking schemes described above, other alternative watermarking methods and techniques for less common 3D model types have been reported. This section presents some of the alternative schemes and techniques that have been applied to embed watermark information into three dimensional data.

In 1998 Hartung et al. reported a non-blind watermarking method for embedding information into MPEG-4 facial animation parameter (FAP) sequences by using a spread-spectrum technique. This algorithm exploits the shape, texture, and motion information from a 3D triangular B-spline human-head model whose facial expressions are modelled according to the MPEG-4 standard. The arbitrary watermark (in binary form) is embedded by displacing control points of the spline surface of the model. The watermark can be extracted from a rendered movie sequence of 2D images or from the watermarked parameters of the model by using the watermarked animation parameters for head animation. To extract the watermark

successfully, they apply a facial feature tracking system which generates FAP sequences from video data. Their test results show that the embedded watermarks survive compression/decompression attacks but cannot be extracted from the video sequence without knowledge of the facial animation parameters.

Dugelay et al. (2002) described a non-blind watermarking framework for 3D objects via texture information. Unlike other methods, which hide information in the polygonal mesh structure with the help of some form of transform domain, they embed and retrieve the arbitrary binary watermark data directly from the texture image of a 3D model with the help of a common image watermarking algorithm. The bit-watermarks are embedded by changing pixel point characteristics (e.g. change of pixel colour). The watermark created by this algorithm is robust to transformation attacks such as cropping, geometric attacks, rotation affine transformation, and low level filtering, and can be used to confirm watermark presence.

Cotting et al. (2004) presented a non-blind technique for digital watermarking of point-sampled geometry based on spectral analysis (but can also be applied to triangle meshes or spline patches). This scheme is a more advanced extension of Ohbuchi et al. (2001; 2002) work however, tuned for scanned objects in the form of point clouds. In this scheme the watermark is directly embedded into the raw data (point cloud) obtained from a 3D scanning. To achieve this, they apply a fast hierarchical clustering algorithm that partitions the model into a set of patches. Then, each patch is mapped onto the space of eigenfunctions of an approximate Laplacian operator to obtain a decomposition of the patch surface into discrete frequency bands. The bit-watermark is then embedded into the patch-point coordinates (i.e. one bit for each x, y, and z) to minimise visual artefacts in the model geometry. The

authors state that this procedure can also be applied to achieve hierarchical or multiple marking. In order to extract the watermark successfully, the original and the marked object have to be aligned and the patch-point differences computed. During extraction, the watermarked model is re-sampled at optimal resolution using a moving least-squares (MLS) projection. After extracting the watermark, the corresponding bit stream is analysed using statistical methods based on correlation. According to their test results a successful watermark extraction can be achieved as long as one patch contains the necessary encoding. This watermarking scheme is robust against numerous attacks including low-pass filtering, smoothing, re-sampling, affine transformation, cropping, additive random noise, and combinations of them. However, embedded watermark information in raw point clouds would be lost in a rendering process that converts point cloud data into some sort of surface or solid based data.

A different way to encode information into a 3D object is to fill the model with carrier objects that carry the watermark data (so-called illustration watermark), as presented by Sonnet and Lange (2005). Their technique comprises two main aspects: first, carrier objects are generated according to the amount of data to be encoded; and second, the carrier objects are placed inside cross-sections of a 3D polygonal mesh model (see example in Figure 4.5). These carrier objects offer several possibilities to carry the information in the form of binary codes. Amongst these possibilities are: object transformation such as rotation and scaling (i.e. the watermark bit determines the magnitude of rotation or scaling), the topology of the carrier object, and the material's colour (i.e. colour channel bits are replaced by watermark bits), which can all be used for information encoding. The encoded data is visually imperceptible

since the polygonal mesh of the original model remains unchanged, which encloses the carrier objects. Since the carrier objects carry all the information necessary for watermark detection and decoding this scheme can be described as a blind method.

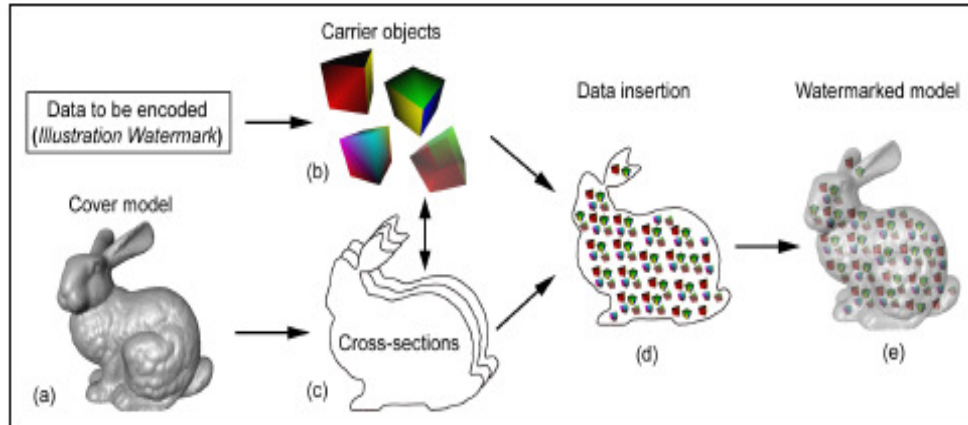


Figure 4.5 Carrier objects for information hiding
(adapted from: Sonnet and Lange, 2005)

Depending on the complexity of the carrier objects and the available space inside the model, several megabytes of data can be inserted. The watermarked 3D model is robust against common geometric transformations such as translation, rotation and scaling. File format conversions do not affect the embedded data as long as the model's overall structure is not altered or violated. However, model compression or partial cutting of model components may destroy the embedded data. Also, decreasing the accuracy of the model's geometry (e.g. cutting decimal places of vertex positions) has an effect on the embedded watermark. The method also significantly increases the "size" of models making the presence of a watermark obvious.

In 2006 Bennour and Dugelay proposed a unique non-blind method for watermarking 3D objects. Unlike most 'conventional' 3D watermarking techniques where both watermark insertion and extraction are performed on a 3D object (3D/3D

approach), they developed an asymmetric 3D/2D watermarking procedure. In other words, the bit-watermark embedding takes place on the 3D object contour, but the watermark retrieval is regained from specific 2D views of the model. In the first step, the contour of the 3D object is extracted and defined as the watermark carrier. Then, the watermark is embedded into these contour vertex coordinates by applying an extended 3D version of a Fourier transform descriptor. For a successful watermark extraction a set of secret parameters (such as viewing angle or magnifying factor) are used to create a specific 2D projection of the model, where the contour is created. Then, a 2D Fourier descriptor is used to compare silhouette properties of the original and watermarked contours. This method, which is basically used for authentication purposes only, is robust against translation and rotation actions and scaling attacks. However, it requires the modification of a 3D object's shape.

4.2.4 Discussion

All watermarking systems are designed to achieve one goal, i.e. embedding a hidden, robust watermark into 3D objects. These systems have to satisfy two conflicting requirements; first, the watermark must be immune against intentional and unintentional attacks; and second, the watermark should be perceptually undetectable.

In the field of 3D digital watermarking, a variety of different watermarking methods and schemes have been proposed. The majority, however, focuses on three dimensional polygonal mesh models (~ 57%). Only a few techniques focus on other model and surfaces representation types like Bézier, NURBS or CSG data (see Table 4.1). There are equally few researchers who watermark 3D objects (represented in less common data structures) by using different techniques to hide watermark

information. No one has attempted to watermark solid three dimensional B-rep CAD data.

Most of the techniques described above can be considered shape-altering methods (e.g. spatial, wavelet, frequency, Fourier, or geometrical transformations, etc.), since they displace control point vertices or alter an object's geometry and/or topology. The reason why so many use the transform domain is that this domain offers various advantages. For instance, by modifying the spatial frequency band, where human beings are not very sensitive, a watermark embedded in an object can be made less visible. Or, by targeting the coarse shape feature, the embedded watermarks are less susceptible to low-pass filtering or additive random noise. In this case, watermarks become harder to remove since coarse shape features are often essential to the target content data (Ohbuchi et al., 2002).

About 90% of the watermarks embedded with the reviewed techniques and schemes are robust in nature. In other words, they are intended to remain 'anchored' in the host data even if someone tries to destroy them, or if the carrier data has been modified. However, most watermarks can withstand only certain types of attacks. Usually, topology-based watermarks are not robust to affine transformations, while vertex displacement marks are not robust to mesh simplification. Robust watermarks are those based on geometry, since topological modifications are destroyed by elementary simplification or re-meshing operations. Despite the varying methods and techniques, re-meshing is still the "worst" type of attack for polygonal mesh data since it changes the whole mesh constellation of an object and with it the embedded data. The remaining methods (around 10%) have fragile characteristics, which serve to prove data authenticity, even when someone changes or violates the protected

data. In this sense, its main purpose is to break at the slightest data modification or attack.

In 87% of the reviewed approaches, as listed in Table 4.1, the watermark has a binary format when it is embedded into the carrier object. In the remaining cases the watermark is represented by random numbers (10%) or by ASCII code (3%). For the embedding of the watermark information, 96 per cent of all approaches use some sort of vertex/control-point/pixel modification/displacement techniques to anchor the values. However, some approaches hide the watermark message into a newly created CSG tree, which is added to the CSG model; other methods use binary watermark message to create a triangle strip on the 3D CAD model (i.e. the message content determines the course/shape of the strip). In most cases (80%) the watermark information is used to confirm the presence of a watermark, while only 17% of the presented techniques are aimed to “transport” a text message. In one case only, the watermark message is used for content verification (i.e. the mark carries information about the characteristics of the model).

To recover the embedded information 53% of the presented techniques do not need a hint about how and where to find the watermarks and/or how to encrypt the information. For the remaining 47% the hidden watermarks cannot be recovered or decoded without the support of the original cover file.

Existing watermarking techniques, mostly focusing on 3D polygonal mesh shapes, are not suitable for the precise CAD data used in most mechanical design systems. There are two reasons for this: First, current mechanical CAD systems use parametric curves and surfaces such as Bézier, non-uniform rational B-spline (NURBS) curves and surfaces, or boundary representation (B-rep) as standard for

precise shape definition. Thus, watermarking algorithms that target polygonal meshes cannot be applied or extended to common solid mechanical CAD data. Second, most polygonal mesh watermarking algorithms for 3D objects alter shape geometry and/or topology for watermark insertion by modifying either model vertex coordinates, model vertex topology (i.e. connectivity), or both. However, precise CAD models rarely tolerate any changes in geometry or topology. Vertices have to lie on edges which, in turn, must be located on surfaces. Consequently, changing the location of any element impacts the integrity of the entire data structure. Therefore, watermarking algorithms that exactly preserve shape and geometry are required for most mechanical CAD data application systems (Ohbuchi and Masuda, 2000). A summary of the watermarking systems discussed in section 4.2 is presented in Table 4.1.

Table 4.1 Digital watermarking techniques and methods for three dimensional data

Author	CAD data Format	Operating domain	Type of modification	Characteristic of watermark	Format of watermark	Type of detection	Purpose of watermarking
Alface and Macq (2005)	polygonal mesh	spectral	displacement of vertices	robust	arbitrary bit value	blind	confirm presence of watermark
Aspert et al. (2002)	polygonal mesh	transform	displacement of vertices	robust	bytes (created by hash function)	non-blind	carry text message
Bendens et al. (1999a, 1999b, 1999c, 2006)	polygonal mesh	transform	displacement of vertices	robust	arbitrary bit value	blind	confirm presence of watermark
Benedens (2000a)	NURBS	transform	displacement of vertices	robust	bit-string	blind	confirm presence of watermark
Bennour and Dugelay (2006)	2D/3D contour	transform	modification of vertex coordinates	robust	arbitrary bit value	non-blind	confirm presence of watermark
Bors (2006)	polygonal mesh	transform	geometrical perturbation of vertex location	robust	arbitrary bit value	blind	confirm presence of watermark
Chadwick et al. (2003)	Bézier surface	transform	control point modifications	robust	arbitrary bit value	non-blind	confirm presence of watermark
Chou and Tseng (2006)	polygonal mesh	transform	relocate/modify vertex coordinates	fragile	arbitrary bit value	blind	confirm presence of watermark
Cotting et al. (2004)	point cloud	frequency	patch-point coordinate modification	robust	arbitrary bit value	non-blind	confirm presence of watermark
Daras et al. (2004)	polygonal mesh	transform	geometrical perturbation of vertex location	robust	arbitrary bit value	blind	confirm presence of watermark
Denis et al. (2005)	polygonal mesh	transform	modulation of object coordinates	robust	arbitrary bit value	non-blind	confirm presence of watermark

Author	CAD data Format	Operating domain	Type of modification	Characteristic of watermark	Format of watermark	Type of detection	Purpose of watermarking
Djaghloul (2007)	constructive solid geometry	data extension	coordinate vector replacement	robust	arbitrary bit value	blind	confirm presence of watermark
Dugelay et al. (2002)	image texture	transform	pixel coordinate modification	fragile	arbitrary bit value	non-blind	confirm presence of watermark
Farnaro and Sanna (2000)	constrictive solid geometry	data extension	added to CSG tree	fragile	bytes (created by hash function)	blind	content verification
Harte and Bors (2002a, 2002b)	polygonal mesh	transform	displacement of vetices	robust	arbitrary bit value	blind	confirm presence of watermark
Hartung et al. (1998)	facial animation parameter	spread-spectrum	displacement of control points	robust	arbitrary binary data	non-blind	confirm presence of watermark
Hu et al. (2008)	Bézier curves	Spread-spectrum	control point modifications	robust	arbitrary bit value	non-blind	confirm presence of watermark
Kanai et al. (1998)	polygonal mesh	wavelet transform	wavelet vector coefficient modification	robust	arbitrary bit value	non-blind	confirm presence of watermark
Lee (2004)	NURBS	transform	control point modification	robust	arbitrary bit value	non-blind	confirm presence of watermark
Maret and Ebrahimi (2004)	polygonal mesh	transform	vertex position modification	robust	text converted to bit values	blind	carry text message
Murotani and Sugihara (2005)	polygonal mesh	transform	vertex position modification	robust	arbitrary bit value	blind	confirm presence of watermark
Nagahashi et al. (2004)	Bézier polynomial patches	transform	patch replacement	robust	ASCII code	non-blind	carry text message

Author	CAD data Format	Operating domain	Type of modification	Characteristic of watermark	Format of watermark	Type of detection	Purpose of watermarking
Ohbuchi et al. (1997a, 1997b, 1998a, 1998b, 2001)	polygonal mesh	transform	vertex displacement, triangle-strip creation	robust	arbitrary bit values	blind	confirm presence of watermark, carry text message
Ohbuchi et al. (1999, 2000)	NURBS	transform	reparameterisation of knot vectors	robust	arbitrary bit values	non-blind	confirm presence of watermark
Praun et al. (1999)	polygonal mesh	transform	displacement of vertices	robust	cryptographic random number created by hash function	non-blind	carry text message
Uccheddu et al. (2004)	polygonal mesh	transform	displacement of vertices	robust	numerical values	blind	confirm presence of watermark
Sonnet and Lange (2005)	carrier objects	transform	vertex displacement of carrier objects	robust	arbitrary bit values	blind	carry text message
Yin et al. (2001)	polygonal mesh	transform	least significant bit (LSB) replacement	robust	arbitrary bit values	non-blind	confirm presence of watermark
Wu and Cheung (2005a, 2005b)	polygonal mesh	spatial	vertex coordinate modification	fragile	arbitrary bit values	blind	confirm presence of watermark
Zafeiriou et al. (2005)	polygonal mesh	transform	vertex coordinate modification	robust	pseudorandom number	blind	confirm presence of watermark

4.2.5 Gap in the Literature

As highlighted above, there has been only limited prior research on watermarking three-dimensional solid CAD data, and was mainly restricted to modification of Bézier, NURBS and mesh surfaces. Also, since most watermarking schemes alter geometry and/or topology of the intended CAD file, a technique is required that does not change the object's shape (a crucial requirement for many CAD applications, e.g. assembly simulation, FEM analysis, etc.), and is capable to carry text message information for various purposes, and not only for data authentication.

In addition to designing a digital watermarking method suitable for solid B-rep CAD data, another important issue needs to be addressed: the CAD specific evaluation and benchmarking of watermarked data. This requirement includes evaluation for robustness and subjective and/or quantitative evaluation of the distortion introduced to the watermarked data. Currently, it is difficult to compare the reported 3D watermarking techniques with each other for the following reasons:

- No standard shape representation (polygonal meshes, NURBS patches, Voxels, B-rep, etc.) for benchmarking 3D watermarks.
- No standard 3D CAD models for benchmarking.
- No standard evaluation and testing procedures.
- Varying amount and nature of information to be embedded.
- No standard sets of attacks (both intentional and unintentional) on CAD data.
- No effective shape distortion measure, both geometric and human-perception based.

It is clear that for fair benchmarking and performance evaluation, watermarking methods need to be tested on a common basis.

To fill these gaps this thesis will introduce a potential watermarking scheme for solid three-dimensional B-rep CAD data that enables text message embedding and retrieval. Unlike existing techniques it does not change the CAD model's shape or increase the file size. In addition, a CAD relevant testing procedure including appropriate solid 3D CAD objects will be developed.

Chapter 5

5 Methodology

This chapter presents the methodology adopted in this research study. The chapter begins with outlining and discussing the general philosophical and methodological approach of the study. Then an overview of the results of the Knowledge & Information Management (KIM) project survey is presented. This is followed by the requirements for a 3D watermarking system and the process of the PO-watermarking scheme is discussed.

5.1 Research Approach

A research strategy may be thought of as providing the overall direction of the research including the process by which the research is conducted. At a strategic level the research process is defined in broad terms that take into account the general philosophical approach adopted by the researcher. The philosophical approach describes how the researcher views the world and it thus influences the chosen research methodology.

5.1.1 Research Philosophy

Two main research philosophies can be distinguished that view the world in different ways: positivism and phenomenology. The phenomenologic paradigm views the world as socially constructed and subjective. Therefore, phenomenology stresses the subjective aspects of phenomena by focusing on the meaning, rather than the measurement, of them (Gill and Johnson, 2002). The phenomenological research paradigm is sometimes described as the descriptive/interpretative approach and implies that every event studied is a unique incident in its own right. In this school of

thought there is nothing other than phenomena and the essence of a phenomenon is understood intuitively (Easterby-Smith and Thorpe, 1991).

The positivistic approach, on the other hand, seeks the facts or causes of phenomena, with little regard to the subjective state. Thus, logical reasoning is applied to the research so that precision, objectivity and rigour replace hunches, experience and intuitions as the means of investigating research problems. According to positivists, laws provide the basis of explanation, permit the anticipation of phenomena, predict their occurrence and therefore allow them to be controlled. Explanation consists of establishing causal relationships between the variables by establishing causal laws and linking them to a deductive or integrated theory (Collis and Hussey, 2003).

The present study clearly falls under the positivistic paradigm. As discussed in Chapter 1, the aim of this study is to develop a digital watermarking algorithm for three dimensional CAD data and appropriate testing procedures. These aims require a highly scientific and objective process based on facts and logical reasoning.

5.1.2 Research Process

This study follows the process suggested by Duffy and O'Donnell (1998). Figure 5.1 provides a schematic overview of the adopted process.

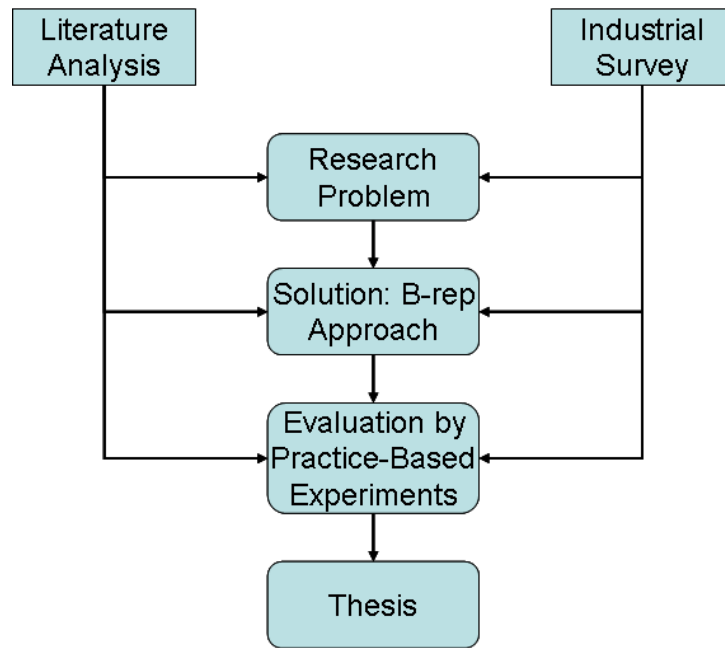


Figure 5.1 Research process adopted for this study

The research started with the definition of the research problem based on a literature analysis and the industry requirements identified in a survey conducted for the KIM project. A need for a watermarking scheme for high value 3D B-rep CAD data was identified that enables the encoding and embedding of text information without affecting the overall shape and the geometrical and topological structure of the 3D model.

Then the proposed solution was implemented, i.e. the PO-watermarking scheme including the text message encoding algorithm and the tweaking algorithm were developed.

Afterwards the proposed solution was evaluated against the set objectives of the research study. The most appropriate way to evaluate the PO-watermarking scheme's capabilities was to apply an experimental method, since it can be used in a controlled manner (e.g. assignment and manipulation of specific variables) (Wood, 1974). As the PO watermark has to interact with commercial systems whose internal algorithms

are not public, the only way to conclusively determine how the watermark behaves is to conduct experiments. Thus, this method gave the opportunity to investigate the PO-watermarking behaviour under real-world conditions.

In the final phase of the research methodology the results of the research were documented, primarily in this thesis but also in some published papers.

5.2 Knowledge & Information Management Project Survey

In April 2007 a survey was carried out as part of the KIM project to investigate industrial needs, requirements and expectations towards watermarking of three dimensional CAD data. All industrial partners (large UK based engineering companies) of the Grand Challenge project were surveyed (N = 10). Three completed questionnaires were received resulting in a response rate of 33 percent. The questionnaire and a summary of the responses can be found in Appendix D. According to responses watermarking is required for identifying data source and owner, copyright notice, restrict and regulate access, monitor data distribution, and carry project or product related information. In order to meet their needs the embedded watermark needs to withstand various translation procedures (e.g. into neutral file formats), carry text information, should visually inform users about product status (e.g. for internal use only), withstand geometrical data modifications, and should not interfere with the CAD model capabilities. The survey results were used to inform the requirements of a CAD watermarking system described in the next section and the choice of attack types are discussed in section 6.2.1.

5.3 Requirements and Mechanisms for 3D Watermarking Systems

The requirements for a digital watermark and a digital watermarking system are different in nature. The basic requirements that watermarks need to fulfil are transparency, robustness to attacks, and no influence on, or interference with, the CAD model's functionality. The demands and requirements on a digital watermarking system however, are different:

- A Watermarking system should be able to encode and embed a set of information into the carrier model, and the detector should be able to retrieve and decode the embedded data (i.e. not just report whether or not a watermark is present).
- The encoding and embedding process must not change the shape (i.e. topology, or geometry) of the CAD model. Any changes to a model's shape would affect the results of any CAD data simulation and analysis (e.g. FEM or stress analysis).
- The carrier CAD data file should not be increased significantly in size when adding the watermark information because this could reveal the presence of a watermark.
- The watermark embedding/anchoring mechanism should create a robust connection between the watermark signal and the carrier object, so that the watermark does not get lost when the carrier data is modified, attacked or transformed.
- The watermarking system should be able to embed as much information as possible (companies may, for example, wish to add manufacturing instructions or archiving information instead of 'only' the company's name or logo for verification purposes).

- Last but not least, the watermark should be evenly distributed throughout the CAD model. This ensures that the watermark message will not be removed or destroyed by modifying a single or a specific part of the CAD object.

Considering the constraints and necessities for boundary representation data (presented in Section 3.3.2) and the requirements for watermarking systems (as listed above), which ‘watermarking’ method or technique might work for B-rep CAD data?

The following concepts were identified as possible mechanisms:

1. *Explicit attribute addition*: it would be an easy task to associate additional data (i.e. attributes) with the entities of a model’s data structure. With closed-proprietary formats this information would be hard to see or remove within a vendor’s system. However, these attributes would be lost during system or data translation (e.g. neutral formats), and consequently, such a system would be limited in its scope.
2. *Local operations to modify geometry*: there is a class of model editing operations known as ‘*local ops*’ that facilitate small, or local, changes in geometry (e.g. offsetting, surface substitution, etc.) that could be used to introduce a geometric watermark by modifying the object’s geometry. However, such an approach would not meet the ‘no change in shape’ requirement.
3. *Micro imprinting*: a Boolean operation could be used to ‘stamp’ a pattern onto the model’s faces (i.e. create inner edge loops). Such a pattern could be so small as to be invisible to the naked eye and would not cause any change to the overall shape of the model. However, it could be easily removed by repairing or healing

operations that detect the redundant nature of the faces and edges on the model's surface.

4. *Non-manifold⁷ entity addition (flap faces, internal shells, and wire edges)*: visible features of an object (e.g. an edge) could be shared and used to create another object inside the actual CAD model to carry the watermark information. Although this approach is potentially resistant to data healing functions (healing takes place on the outer surface of an object) and would not change the external shape, it could still create problems for computer numerical control (CNC) and rapid prototyping (RP) systems that derive their tool paths directly from the solid data structure. Furthermore, non-manifold geometry can be easily removed by standard modelling utilities (e.g. removing the concerned edge which both models are sharing).
5. *Vertex coordinate modifications*: using geometrical or topological modification tools to displace or modify vertex coordinates (analogous to many polygonal mesh watermarking schemes) would change a surface, because the modified points would be “outside” of the specified surface. Even if the face is recalculated to lie on the modified vertex, this alteration would breach the requirement not to change the object's shape.
6. *Adding new object branches/features*: additional ‘empty’ volume elements could be added as a watermark carrier (similar to CSG watermarking) and would potentially offer a large watermark storage space. However, they can be easily

⁷ Manifold is a geometric topology term that means: to allow disjoint lumps to exist in a single logical body, while non-manifold means: all disjoint lumps must be their own logical body. In other words, two blocks use one and the same edge or vertex where each block is a ‘disjoint lump’ yet there is one single body. The shared edge or vertex between the blocks is the actual non-manifold condition.

identified and erased, or copied onto other CAD data to obtain a ‘marked’ version since the watermark information is not directly ‘embedded’ into CAD model data structure. Also, they would increase a CAD model’s file size considerably, which again might hint that the carrier CAD data is watermarked.

7. *Watermark carrier objects*: similar to non-manifold entity addition approach, filling the 3D object with carrier objects (effectively creating topological separate “voids” within the body) has the advantage that a huge amount of information can be hidden while the overall shape of the CAD model does not change. Also, some modification functions like healing or translation would not have any effect since the watermark information is ‘inside’ the CAD model. However, editing the model via Boolean operations would erase the embedded watermark. Furthermore, the CAD file size would also increase considerably.
8. *Parametric orientation*: The characteristics of B-rep data (as described in Section 3.3.2) suggest that a simple but effective way of watermark embedding can be achieved by reorientation (or tweaking) of the parametric representation underlying a B-rep face. In other words, all faces of a B-rep model such as plane, cylinder, sphere and others have a parametric representation that allows them to be traversed in terms of (u, v) coordinates regardless of the face’s overall shape (e.g. plane, cylinder, cone, sphere, torus, etc.). The orientation of the u-v parameterisation is largely determined by the operations used to create the model (i.e. the orientation of bodies united and subtracted to define the shape).

When comparing the various listed possibilities for embedding text messages into CAD models, one can see that the *Parametric Orientation* (or PO) mechanism offers the best prospect of meeting the various requirements (see also Table 5.1). For

instance, the parametric orientation method would make only small invisible changes to the model's B-rep faces by re-orientating the underlying geometry, but without changing the overall shape of the CAD data. Also, since the created modifications are 'part' of the model, the watermark message is robustly anchored within the model so that it would withstand various transformation and geometrical modification attacks. Finally, because the PO-approach would only re-orientate existing faces' underlying geometry, no additional data is actually added, which again preserves the CAD object's file size.

Table 5.1 Comparison of potential B-rep watermark embedding mechanisms

requirements mechanisms	Invisible	Robust	Withstand data trans- formation	Withstand geometrical modification	No change in shape	Data healing	No change of file size	Size of the watermark
Explicit attribute addition	✓	✓	✗	?	?	?	✗	?
Local operation to modify geometry	?	?	?	?	✗	?	✓	✗
Micro imprinting	✗	?	?	?	✓	✗	✓	✗
Non-manifold entity addition	✗	?	?	✗	✓	✓	✗	✗
Vertex coordinate modifications	✓	?	?	?	✗	?	?	?
Adding new branches/features	✗	✗	?	?	?	?	✗	✓
Watermark carrier objects	✓	✓	✓	✗	✓	✓	✗	✓
Parametric reorientation	✓	✓	✓	✓	✓	✓	✓	✓

✓ = Yes; ✗ = No; ? = Uncertain

However, to verify and quantify this assessment the following questions need to be answered:

1. Which face types on a B-rep model and what parts of a face could be used to host the watermark information?
2. How does the size of the B-rep model affect the amount of information that can be embedded?
3. How can a watermark be inserted without breaking (or changing) the CAD model's shape?
4. How much can a parametric surface be rotated without affecting the model's size or functionality?
5. Should (or could) the watermark be inserted into only one part or certain face types, or distributed onto several parts of the object within different faces types?
6. What is the best strategy to embed text/watermark information: a short message (e.g. company name or logo) which is embedded multiple times to achieve robust characteristics, or one single but long message (e.g. authentication information) that is embedded only once to achieve fragile characteristics?
7. Which method might be the best to analyse and test the robustness of a CAD related B-rep watermark?
8. Can the embedded watermark survive common CAD related actions, modifications and attacks?
9. How can the proposed parametric reorientation method be compared to other relevant 3D-CAD watermarking approaches?

The following sections will describe the proposed PO-watermarking scheme for three dimensional B-rep CAD data and also the assessment methodologies used to answer the above questions.

5.4 Method Overview of the PO-watermarking Scheme

To find answers to the questions above a prototype Parametric Orientation watermarking (or ‘PO-watermarking’) system was designed and implemented. For the purposes of the investigation the ‘easily’ tweakable planar, closed conical, toroidal and spherical surfaces were modified and tested. Those with complex boundaries (i.e. geometry that could create chiralities in the solution) were excluded at this stage, but could be considered in future embedding and testing procedures (see Section 8.3).

The whole PO-watermarking scheme can be divided into two components: the embedding and retrieval. Each of these two components were designed and implemented to modify and interrogate the underlying parametric surface of B-rep faces on CAD models for both purposes: first, to enable information embedding into valuable CAD data without changing its shape; and second, to detect and retrieve the watermark message within the marked data with the input of the original unmarked model (i.e. non-blind detection). The applied functions and sub-programmes for the PO-watermarking scheme were designed and developed in Microsoft Visual C++ 6.0 using the 16th version of the ACIS 3DT kernel modeller (the basis for numerous mechanical CAD systems). The test parts were created and modified in Autodesk Inventor Professional Version 11.

The corresponding C++ codes for the embedding process are listed in Appendix A, the codes for the watermark retrieval process in Appendix B.

5.4.1 Watermark Embedding

The watermark embedding section of the PO-watermarking scheme consists of three sub-tasks. The first task deals with the preparation of the used/loaded carrier B-rep CAD model, while task two creates the information to be embedded. The third task combines the results of the first two sub-tasks and applies a tweaking function to embed the encoded information into the host CAD object. An overview of the PO-watermark embedding process is shown in the flowchart below (Figure 5.1), while each of the sub-tasks is described in detail in the following sections.

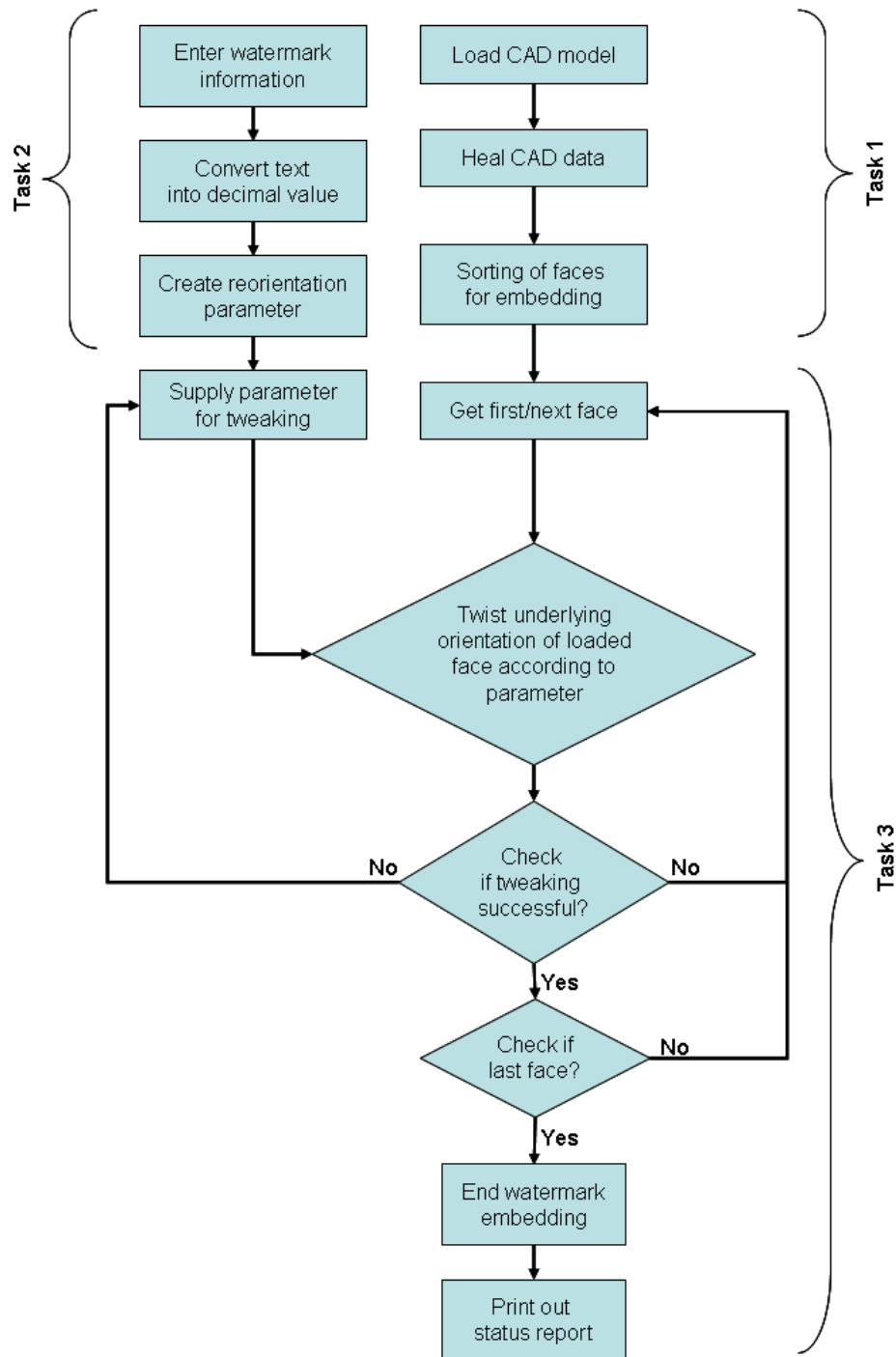


Figure 5.2 Flow chart of the watermark embedding process

5.4.1.1 Task One: Preparation of Carrier CAD Data

The aim of the first task is to prepare the carrier B-rep CAD model for watermark embedding. These preparations include the loading of the intended host data as *.sat* file format, elimination of any defects or faults inside the carrier CAD model,

creating a face list of all available faces of the model, and finally arranging all faces ready for the embedding process. The objectives of each sub-task are described as follows:

Load CAD model

There are many commercial B-rep formats available; however, they are all very similar (e.g. similar data structures). The choice to use ACIS *.sat* format was made because of significant previous experience of its use in the department, and secondly because it is used as the basis of many of today's commercial CAD systems. Also, comparisons with other kernel modeller suggests that ACIS is amongst the most precise⁸ for B-rep (Ranger, 2009). Since the *.sat* format is also a widely used CAD format, other mechanical design systems should be capable of reading and creating them. Furthermore, the system comes with numerous tools to support the development and debugging of programmes.

Heal CAD data

Before the text-watermark can be embedded into the B-rep host data, the CAD model needs to be 'healed' to ensure that any experiments performed are (as much as possible) unaffected by the 'noise' of errors in the CAD data.

Problems with CAD data, such as gaps between entities and the absence of connectivity information (topology), often appear after the data has been transformed or exchanged (Jackson and Mabbott, 1999). According to Gerbino (2003; p. 1) "The most critical problems in data exchange are the different internal mathematical representation schemes and the internal accuracy of the geometric definitions in the

⁸ Tolerance requirements of different CAD modellers: ACIS: 1.0 E-6 mm; Parasolid: 1.0 E-5 mm, CATIA: 0.1 mm.

modelling kernel of various CAD systems”. In particular, these problems arise from the accuracy and the convergence criteria used when performing intersection calculations with curves and surfaces (Corney and Lim, 2001). All this can occur either within the original design system or during the pre- and post-processing actions of the CAD data. According to Smith (2004) other inaccuracies might also appear due to several factors such as:

- Inadequate geometric modelling in the original CAD environment.
- Geometrical and topological model inconsistencies.
- When geometries are to be converted into other representation forms (e.g. into neutral file formats like IGES, STEP or STL).
- Different implementations of algorithms used during the translation.
- CAD systems using different standards of internal accuracy.

Mechanical CAD systems often use different tolerances on an entity-by-entity basis to provide robustness for model operations or data translation. For data transmission or translation, one or more of the CAD model’s components such as surface texture, colour, but mostly tolerances have been altered. These changed tolerances might produce gaps and/or overlaps in the geometry and topology of CAD models (as shown in Figure 5.2), which can cause representation problems when viewing the data in different CAD systems. Also, it can produce distorted results in CAD data simulation or analysis systems.

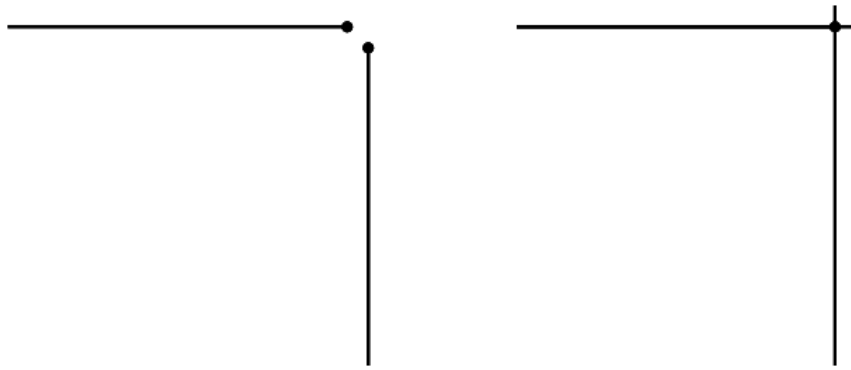


Figure 5.3 Problems within CAD data: gap (left) and overlap (right)

To overcome inaccuracy and tolerance issues, a ‘healing’ function attempts to recalculate and recover close faulty surfaces and surface-to-surface intersections. Geometric modelling kernels such as ACIS, Granite and Parasolid offer common methods for evaluating the validity of B-rep models, and can be used to recalculate and rebuild the boundaries of a CAD model (CADuser, 2000). This process of recalculation is also called *healing* or *CAD data repair*.

The applied ACIS healing function for the PO-watermarking scheme, called HEAL, contains API functions that can be used to ‘fix’ models imported from other modelling systems into ACIS in order to make them usable (i.e. accurate) and ‘error free’ in ACIS. However, HEAL is not a translator from another modelling system into ACIS. The model must already exist in ACIS format (for example, in a *.sat* file) before healing is carried out. The features of the HEAL function can be summarised as:

- Provide user-controllable tolerances for efficient healing (e.g. define the distance considered to be a gap).
- Simplify geometry by converting spline data into analytic data wherever possible (e.g. a flat B-spline to a planar face).

- Stitch faces together in the absence of connectivity (topology) information.
- Provide functionality to ‘tighten’ gaps (inaccuracies) in the model to the desired tolerance requirements.
- Support tightening of ‘smooth’ analytic or spline edges.
- Support edges that could not be tightened after being converted into tolerant edges.

The healing process in ACIS can be considered as a series of stages and tasks that progressively improves and adjusts the consistency of a model’s geometry and topology. The different stages/tasks can be described as follows:

1. Initialisation: The input model is analysed and attributes containing tolerance values of the various healing processes added to the model.
2. Pre-processing: Initial clean-up by removal of duplicate VERTICES, zero length EDGES and zero area FACES.
3. Geometry Simplification: Conversion of unnecessary spline surfaces into simple analytic ones (i.e. plane, cone, sphere, etc.) where possible.
4. Stitching: Pairing of adjacent EDGES and VERTICES to create topology (i.e. FACES which share bounding EDGES).
5. Geometry Repair: Adjusts ENTITIES so that geometry and topology (possibly also changed) are consistent. At the end of this phase every:
 - VERTEX lies on the underlying curve of the EDGE it bounds.
 - EDGE lies on two adjoining FACES.
 - Parametric curves lies on its corresponding FACE.
 - Pair of EDGES, if they meet at all, do so at VERTICES.
 - Pair of FACES, if they meet at all, do so in EDGES and/or VERTICES.

6. Postprocessing: Check geometry for consistent orientation and remove redundant FACES and VERTICES.
7. Termination: Removal of Healing component attributes.

The healed and repaired geometry (recalculated from existing intersections) can be used to update the actual B-rep data structure and produce an object that is topologically and geometrically identical to the original CAD model, but with a much higher accuracy.

The HEAL function in ACIS can handle most of the analytic cases of edge/vertex intersections, sharp spline intersections, tangent spline intersections (for instance when the edge curve is an isoparametric boundary line for both the spline surfaces), and cases in which the edge curve is along a non-isoparametric line for any of the adjoining spline surface. However, it is not able to cure perturbations between surfaces that would change the topology of the model, complex constraints involving noncylindrical cones, and tangent spline intersections that intersect along non-isoparametric lines (ACIS, 2009).

Sorting of faces for embedding

After loading and healing of the intended CAD model, a list of faces (i.e. entities) is created and a sorting function selects the nominated face types (i.e. planar, closed cylinder, sphere and torus) for PO-watermarking.

As the primary objective of this thesis is to investigate the capabilities and robustness of the PO-watermarking approach, an ACIS internal ordering function selects and orders the nominated faces being used for watermarking. The ordering process is carried out by an API function called *api_get_faces(block, faceList)* which is

implemented in the C++ programmes (see appendix A and appendix B). This function sorts faces according to a predefined order and repeats the same ordering process for watermark embedding and extraction. This ensures that the letters of the embedded watermark message are recovered in the same order as they were embedded to recreate and rebuild the full message.

This sorting function is based on the geometric modelling kernel of ACIS, therefore it only leads to accurate results within this modeller. However, to make it independent from the modeller type, a new sorting function could be introduced. To achieve this, the face list should be ordered against a number of different geometric and topological parameters. These include type, area, number of edges, orientation, and number of inner loops. Should two or more faces rank equally in the order list (in the case of a symmetrical object), the same watermark is applied to all of them.

The current prototype PO-watermarking scheme only supports the tweaking of the four face types mentioned above, but might be extended to consider other faces like spline or open conical. To enable this extension, further research is required to establish whether their underlying parametric surfaces can be modified to accommodate the watermark message (i.e. the tweaking parameter) without changing their shapes.

5.4.1.2 Task Two: Text Message Encoding

Having completed the loading, healing and ordering of faces to host the watermark information, the next step is to determine the datum (i.e. value) for surface modifications. In other words, the content of the watermark information will determine the degrees the underlying parametric representations of a B-rep faces are reoriented/modified. In the following section a process is described that transforms

the characters of a text message into decimal values, which are mapped to reorientation values.

There are several coding standards (e.g. ASCII, decimal, octal, hexadecimal, binary, etc.) that can be used to interchange text information into other character forms or formats for computing purposes. However, since the data must be represented as numbers and within the PO-watermarking scheme text information is going to be processed, an ASCII to decimal converter is a convenient way of creating the needed reorientation parameters. Using an ASCII to decimal converter also has the advantage that characters from both text and ASCII have a one to one correspondence (i.e. they do not need to be converted), and can be used directly for B-rep surfaces reorientation.

The decimal standard defines printable characters, numerical values, letters, punctuation marks, space between words and miscellaneous symbols using numbers between 0 and 255 (e.g. the number 101 is defined to be the lowercase letter 'e' in the English alphabet, and 035 equals #). In other words, a triple-digit is capable to encode a single letter or character of the English alphabet. Because the PO-watermarking approach operates/works with rotation values between 0 and 360 degree (or to be more precise 0 and 2π), only number values between 0 and 360 are appropriate. This works very well with ASCII because the text character values fall between 32 and 126 and are therefore well within the 0 to 360 degree range.

The text-rotation value transformation process for the PO-watermarking scheme can be achieved in the following way:

1. Take a given word and fragment it into letters/characters.

2. Convert the characters one by one into three digit decimal values (e.g. 089 for Y or 117 for u).
3. Finally, convert decimal values into radians with six decimal places accuracy for parametric surface reorientation. Investigations by the author have suggested that reorientation values with up to six decimal places can be robustly applied (no rounding influence). Radians are required, because ACIS, various *api* functions and some C++ related programmes operate only with radian values.

The principles of the text conversion process are also shown in Figure 5.3 which visualises the conversion process based on the word ‘of’. In this case the values of 111 or 1.937315° (rad) for the letter ‘o’ and 102 or 1.780235° (rad) for the letter ‘f’ are used to re-orientate two parametric surfaces on a B-rep model. In other words, two faces of a model hold the embedded message value of the word ‘of’.

<p>Example word: “ of ”</p> <p>Conversion from ASCII to decimal</p> <p style="margin-left: 40px;">o = 111</p> <p style="margin-left: 40px;">f = 102</p> <p>Conversion from decimal to radian</p> <p style="margin-left: 40px;">Degree: 111°</p> <p style="margin-left: 40px;">Radians: 1.937315°</p> <p style="margin-left: 40px;">Degree: 102°</p> <p style="margin-left: 40px;">Radians: 1.780235°</p>
--

Figure 5.4 Creation of reorientation/tweaking parameter for two faces

The described process is also executed within the PO-watermarking scheme, where the watermark message is converted into tweaking parameters. Figure 5.4 shows an example where the word “Strathclyde” is converted letter by letter first into ASCII code and then into radian to create the required rotation values.


```
"C:\Users\Csaba\Documents\C++ Codes\Embedding\MultiFaceEmbedding\Debug\MultiFaceEmbe...
Please enter file name <without .sat extension>: Cube
C:/Users/Csaba/Documents/C++ Codes/TestData/Cube.sat
Healing completed.
Please enter message <max. 11 characters, no free space>: Strathclyde
The ASCII decimal value is: 83
In radians: 1.44862
The ASCII decimal value is: 116
In radians: 2.02458
The ASCII decimal value is: 114
In radians: 1.98968
The ASCII decimal value is: 97
In radians: 1.69297
The ASCII decimal value is: 116
In radians: 2.02458
The ASCII decimal value is: 104
In radians: 1.81514
The ASCII decimal value is: 99
In radians: 1.72788
The ASCII decimal value is: 108
In radians: 1.88496
The ASCII decimal value is: 121
In radians: 2.11185
The ASCII decimal value is: 100
In radians: 1.74533
The ASCII decimal value is: 101
In radians: 1.76278
```

Figure 5.5 Tweaking parameter creation process within the PO-watermarking scheme based on the word “Strathclyde“

5.4.1.3 Task Three: Tweaking or Embedding Process

Despite the complexity of the B-rep data structure many operators like Booleans, blending, warping, tweaking or healing exist which can modify a model’s faces while maintaining the integrity of the data structure at the same time. The proposed PO-watermarking scheme is based on one of these operators termed *tweak* which modifies a face’s underlying parametric orientation (i.e. its geometry) and then automatically updates its topology (i.e. adjacency relationships) to accommodate the new surface. Figure 5.5 illustrates the basic principles of the tweaking process where an object’s face (a) has been modified (i.e. rotated) (b, c), and re-inserted (d).

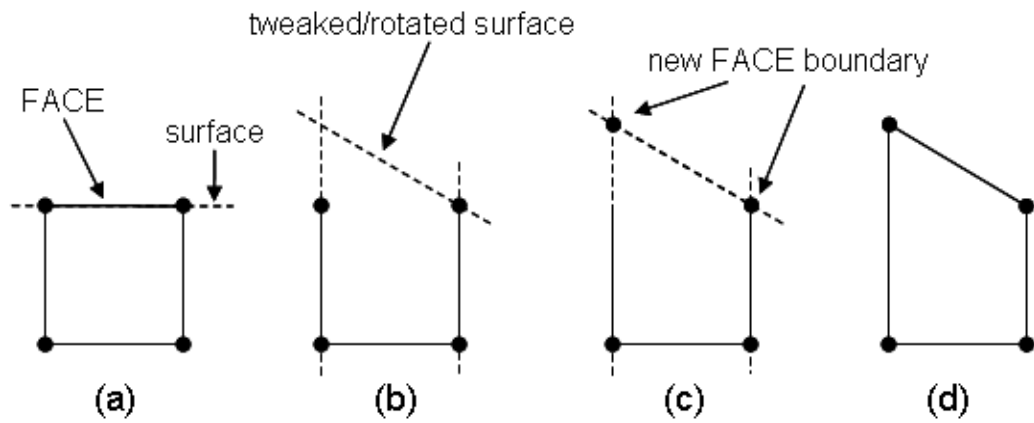


Figure 5.6 Example of surface tweaking
(source: Corney and Lim, 2001, p. 282)

The key observation underlying the tweaking process is that if a face (a) has been slightly changed (i.e. transformed) in some way (b), the edges which bound it can be recalculated by (Corney and Lim, 2001):

1. Intersecting the new (i.e. modified) surface with the surface of each adjacent face (c).
2. And each new curve next to its neighbours will determine the bounding vertices of the updated edges (d).

So it is essential when a new face is inserted, or an existing face is modified, that the tweaking process automatically intersects with all the adjacent faces and so recalculates the geometry of the bounding edge's curves to eliminate gaps or overlaps. This again will guarantee that, despite modifications on a face, the object's overall shape remains unchanged.

In this sense the tweaking operator not only offers the opportunity to modify the orientation of the underlying parametric surface of a face, but can also be used to hide a watermark message. To be more precise, faces with geometrical definitions such as plane, closed cylinder, sphere and torus can be modified, for instance, to

change the orientation of their defining surfaces but without changing their overall ‘shapes’.

Figure 5.6 illustrates the tweaking process of a solid object. It shows a rectangular component before (left) and after (right) its planar faces were tweaked (i.e. rotated anticlockwise) around their face normal vectors.

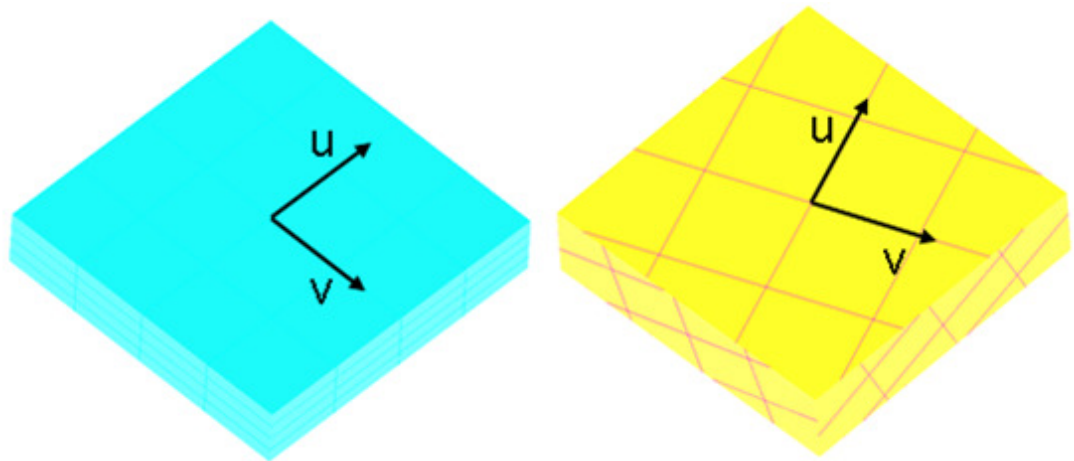
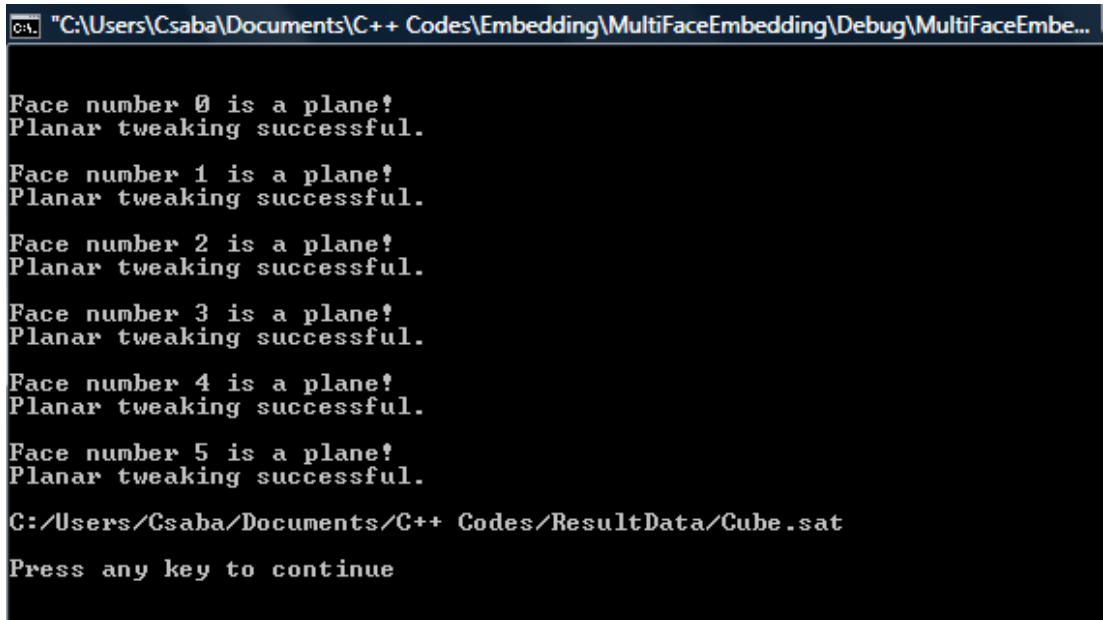


Figure 5.7 Iso-parameter lines showing surface orientations before (left) and after (right) tweaking

In this way all planar faces receive a new orientation (all together carrying the word “Strathclyde” as watermark message) for their parametric surfaces but without changing the face’s rectangular shapes. To make the orientation before and after tweaking visible the u and v axis on one planar face are highlighted with black arrows. The reorientation value of the first face equals 1.448623° rad (when converted back into decimal value equals 83° degree) and represents the letter ‘S’ of the embedded message ‘Strathclyde’ (see text message conversion into rotation values in section 5.2.1.2). However, it is important to note that the amount of each orientation is relative, since each tweaking parameter relates to each individual surface’s original orientation and coordinate system.

Figure 5.7 illustrates a screen-shot printout of the cube model (as shown in Figure 5.6) in which the first six characters of the watermark message “Strathclyde” were successfully embedded.



```
C:\Users\Csaba\Documents\C++ Codes\Embedding\MultiFaceEmbedding\Debug\MultiFaceEmbe...  
Face number 0 is a plane!  
Planar tweaking successful.  
Face number 1 is a plane!  
Planar tweaking successful.  
Face number 2 is a plane!  
Planar tweaking successful.  
Face number 3 is a plane!  
Planar tweaking successful.  
Face number 4 is a plane!  
Planar tweaking successful.  
Face number 5 is a plane!  
Planar tweaking successful.  
C:/Users/Csaba/Documents/C++ Codes/ResultData/Cube.sat  
Press any key to continue
```

Figure 5.8 Cube containing the message “Strath” embedded by the PO-watermarking scheme

Following the above tweaking process, any arbitrary text message can be embedded across ordered sequences of faces with the PO-watermarking scheme. Given a CAD model with N distinctly different (i.e. type, area, shape or relative location) faces, it can hold one character in each of its suitable faces by applying surface tweaking. This procedure is carried out until all the information is embedded into the model or the model is ‘full’. In other words, the number of available and suitable faces within a CAD model determines the length of a watermark message. For instance, a regular cube (as in Figure 5.6) has six faces that can accommodate a text message of a maximum length of six characters (one character in each face), while a more complex object with 150 faces can hide up to a 150 text characters. In other words,

the more faces a model has, the more information can be embedded with the PO-watermarking scheme.

The prototype PO-watermarking implementation has a variable in the code which determines the maximum string length that can be embedded (by default set to 20 characters). However, this default value can be adjusted to different text message length requirements. Also, to create robust characteristics for the embedded watermark the message is embedded multiple times (and is not restricted to a certain numbers of repetitions) until all available and suitable faces have been modified.

5.4.2 Watermark Retrieval Process

The PO-watermarking scheme described here is a ‘non-blind’ method, meaning that the original unmarked CAD model is required to retrieve the embedded watermark information. In other words, the unmodified parametric surfaces of the original CAD model are used to identify the watermarked faces and to calculate the relative orientation of the tweaked surfaces to regain the embedded text information. The PO-watermark retrieval process, as shown in Figure 5.8, can be described in five major steps:

Step 1: Load both original and watermarked CAD models (both as *.sat* file formats) and create a list of faces on both models. In contrast to the watermark embedding process, no healing function is executed.

Step 2: Identify and sequence the embeddable faces in both CAD models. Only the four face types planar, closed cylindrical, sphere and torus are currently implemented in the PO-watermarking scheme. The order of faces determines the sequence of the

letters within the watermark text. Within the retrieval process the same face sorting/ordering sequence is applied as for watermark embedding.

Step 3: Faces from both models with equal sequence are matched and aligned one by one and the orientation difference of their parametric surfaces are computed.

Step 4: The orientation differences of matching original and watermarked faces are measured in radians and converted back into decimal numbers. After that the decimal values are converted back into readable text characters with the ASCII – decimal converter. This is basically the watermark encoding and embedding process, but in reverse order.

Step 5: In the final step the text characters are assembled (i.e. the watermark message recreated) and the embedded information is displayed onto the computer screen and written into a *.wmk* archiving file. An example of a *.wmk* file, based on the cube in Figure 5.6 with the embedded message “Strath”, is shown in a later section in Figure 7.3 on page 184.

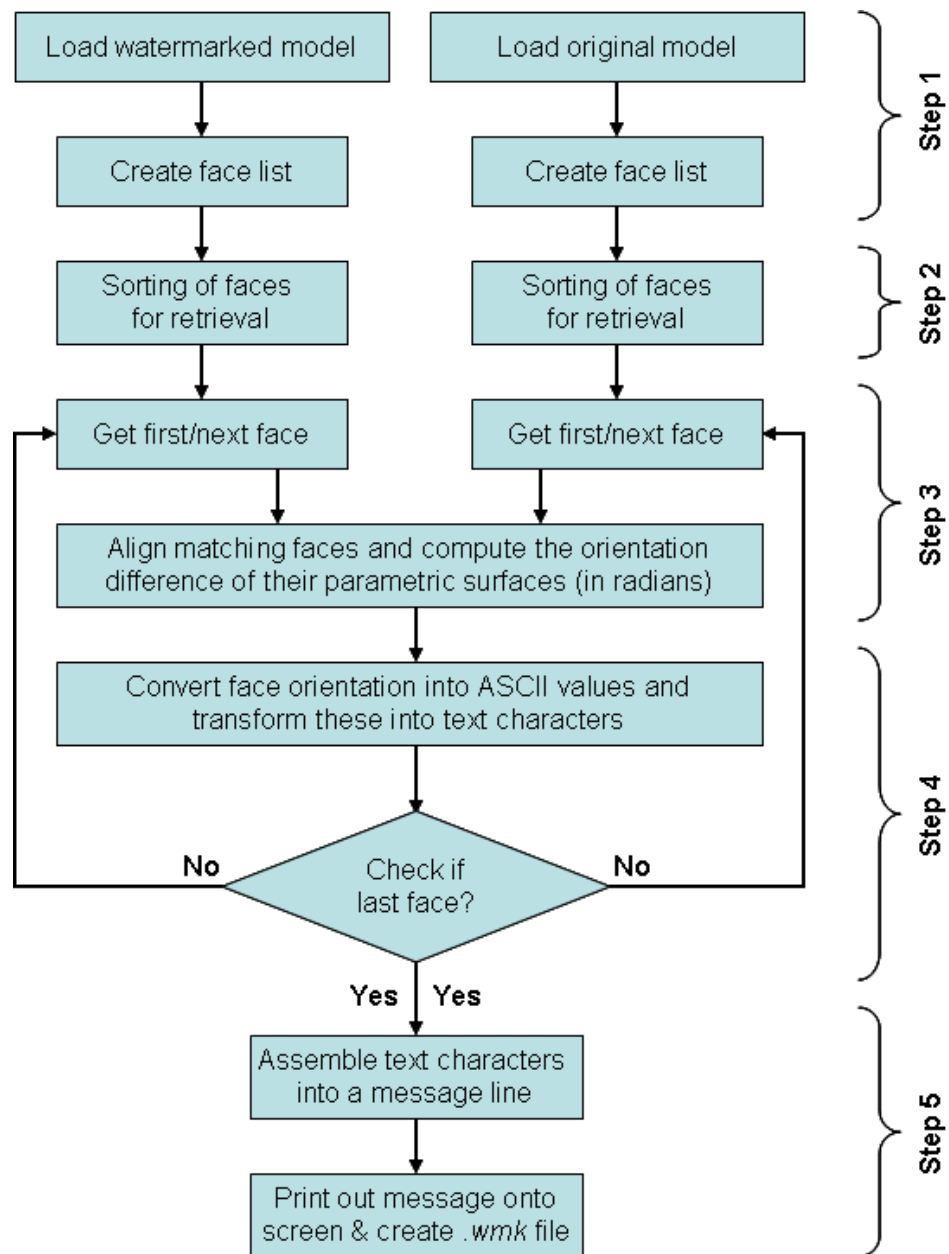


Figure 5.9 Flow chart of watermark retrieval process

Chapter 6

6 Testing Procedures for the PO-watermarking Scheme

This chapter discusses the design and implementation of a potential watermark assessment scheme. It also gives rise to a benchmarking scheme for evaluating watermarked mechanical 3D CAD data. As part of this contribution, a set of appropriate test objects and models are suggested that is used to test the PO-watermark's robustness, but could also be used by other researchers to assess competing schemes. Finally, the process is described for testing the data (created by the PO-watermarking scheme) using the proposed benchmarking method.

6.1 Lack of Common Standards for Testing Watermarked 3D Data

Besides designing and developing a new digital watermarking method, another important issue needs to be addressed: the appropriate evaluation of watermarked data. This not only requires testing of the embedded watermark's robustness, but also includes subjective and quantitative evaluation (e.g. change or alteration in shape or appearance) of the distortion introduced to the watermarked data. Currently, it is difficult to compare existing watermarking techniques and schemes for three-dimensional data for the following reasons:

- No common shape representation format (i.e. polygonal meshes, NURBS patches, Voxels, B-rep, etc.) is used for watermark evaluation. Having a watermark message embedded into a polygonal mesh model might not be present or detectable when the data is converted into say a STL or B-rep.
- No standard 3D models, or shapes, are defined for benchmarking. It seems that for each proposed 3D watermarking scheme a different set of 3D data is used for its assessment (e.g. a cat, Mozart sculpture, a horse, dragon, etc.). Consequently,

it is not easy to compare test results of the different schemes since they are achieved on different models.

- No standard evaluation and testing procedures are defined, i.e. there is no agreed procedure which defines what is going to be tested, how is it tested and in which order, and how the data is evaluated. For instance, test procedures might include a set of standard evaluation steps to define how the robustness or fragility of the embedded watermark will be evaluated. Furthermore, ideally the simplicity/complexity of the watermark embedding and retrieval process would also be benchmarked.
- No standard amount, or type, of information to be embedded exists. For instance, one approach uses white noise as the watermark signal in a 3D picture, while another hides text information into 3D objects. Because of this, the different approaches cannot be directly compared with each other.
- No standard set of attacks (both intentional and unintentional) on CAD data have been defined. Since attacks are potentially diverse (see section 2.2.6) they need to be specified and classified by type, e.g. unintentional CAD file handling and modification attacks or intentional, geometrical object modification attacks.
- Last but not least, no effective shape distortion measure exists, neither geometric based nor human-perception based. A slight geometrical distortion might be invisible for the naked eye but might generate fatal results or errors when analyzing the data in FEM or Ansys, for instance.

In conclusion, it can be said that for fair benchmarking and performance evaluation, both watermarking methods and CAD data need to be tested on a common basis.

Therefore, the following section proposes an appropriate evaluation scheme convenient for most mechanical 3D CAD data.

6.2 Possible Benchmarking Scheme for Watermarked 3D CAD Data

To be able to properly evaluate the robustness of the embedded watermark in 3D CAD models the different attack types need to be identified that are common to the CAD environment. This identification and classification of performance criteria might not only help the developer of a watermarking system to focus on the most important security and system development requirements, but also provides a metric for the potential user of a watermarking system for judging the operational area of the applied watermarking technology. Also, for proper testing appropriate test data need to be defined and categorised. The following sections will provide both the classification of attacks on CAD data and a set of appropriate 3D CAD models for testing.

6.2.1 Classification of Attacks on CAD Data

Because of the complex data structure of three dimensional objects and their range of applications there are two major fields of attacks for 3D CAD data. These threats can be categorised into unintentional and intentional attacks. The difference between the two groups of threats is that some practices, such as loading a model into a different CAD system, are not aimed to destroy the hidden mark, while others, such as editing the object's geometry, are. Table 6.1 lists some applications that are common in the CAD environment and groups them into intentional and unintentional attacks. The different threats can also be classified according four different attack levels:

- L1: Low level attacks that might be caused by simply handling the carrier data in its natural way but without the intention to destroy or alter the CAD file (and with it the embedded watermark).
- L2: Medium level attacks that are not intended to alter the shape of the 3D data but aim instead to change some of its visual features like its colour or shininess, its position in a coordinate system (e.g. after rotation), or the file size (e.g. for archiving purposes).
- L3: High level attacks that are aimed to slightly change the object's 'local' geometrical appearance or structure (e.g. when smoothing a surface) but not its overall form/shape.
- L4: Maximum level attacks that are specifically aimed to change the object's appearance, shape and even its function geometrically by adding or removing object features (e.g. through adding new faces, blending or chamfer edges) and with it the hidden watermark.

Table 6.1 Classification of threats and attacks in the CAD field

Level of attack	Intentional attacks	Unintentional attacks
L1 (Loading)		Loading of (watermarked) CAD data in a different system (e.g. in a CAD, simulation or data analysis system).
L1 (Export/Import)		Export/import and transfer of CAD models into different systems (e.g. MatLab, Ansys) for data representation or calculation.
L1 (Conversion)		Convert CAD data into other file formats for further processing, or for translation into neutral file formats (e.g. IGES, STEP) for data distribution.

Level of attack	Intentional attacks	Unintentional attacks
L2 (Compression/ Decompression)	Compression/decompression modifications (e.g. zip, PDF) to disturb or to destroy the hidden mark by changing the internal CAD file structure.	Compression/decompression modifications (e.g. zip, PDF) for changing the file size for distribution or storage purposes.
L2 (Rotation)	Rotation of entities by a specific parameter to change position of host vertices in the coordinate system to destroy the watermark signal.	Rotation of entities or the whole object to have a specific view for printing purposes, or positioning in an assembly.
L2 (Healing)	Object or surface healing (also called data repair) can remove anchored watermark data when recalculating or 'neutralising' an objects internal structure.	Object or surface healing or data repair are used on daily basis to repair gaps in the geometrical data structure to update CAD data before being loaded into data simulation or analysis systems.
L2 (Uniform Scaling)		Uniform scaling: the changing of the physical size of each entity or the whole object by a known factor relative to a known datum.
L3 (Mesh Simplification)	Mesh simplification, re-meshing, re-triangulation or mesh smoothing could be used to erase watermarked vertices.	Mesh simplification, re-meshing, re-triangulation or mesh smoothing are often applied to achieve a simplified object representation or to reduce CAD file size.
L3 (Polygon Simplification)		Polygonal simplification: this operation is often used to transmit a low-level version of the model or to optimise a model by eliminating most of the non-salient faces.

Level of attack	Intentional attacks	Unintentional attacks
L3 (Uniform Scaling)	Uniform scaling: by changing the physical size or shape of an entity or the whole object (called affine transformation) can result in breaking the connectivity between the mark and the host data.	
L4 (Boolean)	Boolean modification, object editing and modification, blending, chamfer, cropping, shear, removal or insertion of new faces will geometrically change the model's shape and appearance and with it the embedded watermark. It will also cause problems during the identification of marked faces and the recovery of the watermark message.	Geometric modifications like Boolean operations, object editing, blending, chamfer, cropping, shear, cropping, removal or insertions of new faces are day-to-day procedures of an engineer to edit or update existing CAD models.
L4 (Removal Attack)	Programmes specifically created to detect and destroy possible watermarks to create an 'unprotected' version (removal attack), or to copy the watermark information from an approved CAD model to another unauthenticated model (copy attack).	

The choice of these kinds of attacks is based on the KIM survey carried out by the author to identify modification types most common to the CAD environment. All of the listed attack types in Table 6.1 might have some effects on the robustness of embedded watermarks or on the CAD data structure. However, it should be noted that the transition between the two threat groups (unintentional and intentional) overlaps sometimes, which makes it difficult to differentiate between them. In other words, some attacks do not clearly belong to one group only. For instance, data

compression is often used to produce smaller data files for quick data transfer across the Internet or for archiving reasons. However, data compression also changes the internal data structure (especially polygonal mesh surfaces) which can either destroy the vertices carrying the watermark or the bonds between the CAD model and the watermark, or render the watermark information unreadable. Another example is when a design engineer edits a CAD model. On one hand, it can be considered as an attack to destroy the watermark or to remove a surface that contains the watermark information. On the other hand, geometrical modifications are part of his daily job to update or to modify CAD models to the meet new requirements or specifications.

6.2.2 Appropriate CAD Test Data

To assess the robustness of embedded watermarks in CAD data a series of different but representative test objects are necessary. Since this thesis focuses on watermarking mechanical 3D CAD data, appropriate test objects should obviously be a range of CAD objects varying from standard mechanical parts like nuts and bolts to highly complex engineering components. Testing this variety of different geometries might help to answers questions like the following:

- 1) Is the size or complexity of a CAD model crucial for the amount of information that can be embedded?
- 2) Is the watermark evenly distributed throughout the 3D model?
- 3) Can the embedded watermark survive common CAD related modifications?
- 4) What are the effects of data handling processes?

The suggested test objects are grouped into three different categories varying in their complexity from basic Level 1 to sophisticated Level 3 (Figure 6.1 to 6.3). Their file

sizes and number of faces also increase roughly proportionally to the grade of geometrical object complexity.



Figure 6.1 Level 1 test objects with low geometrical complexity



Figure 6.2 Level 2 test objects with advanced geometrical complexity

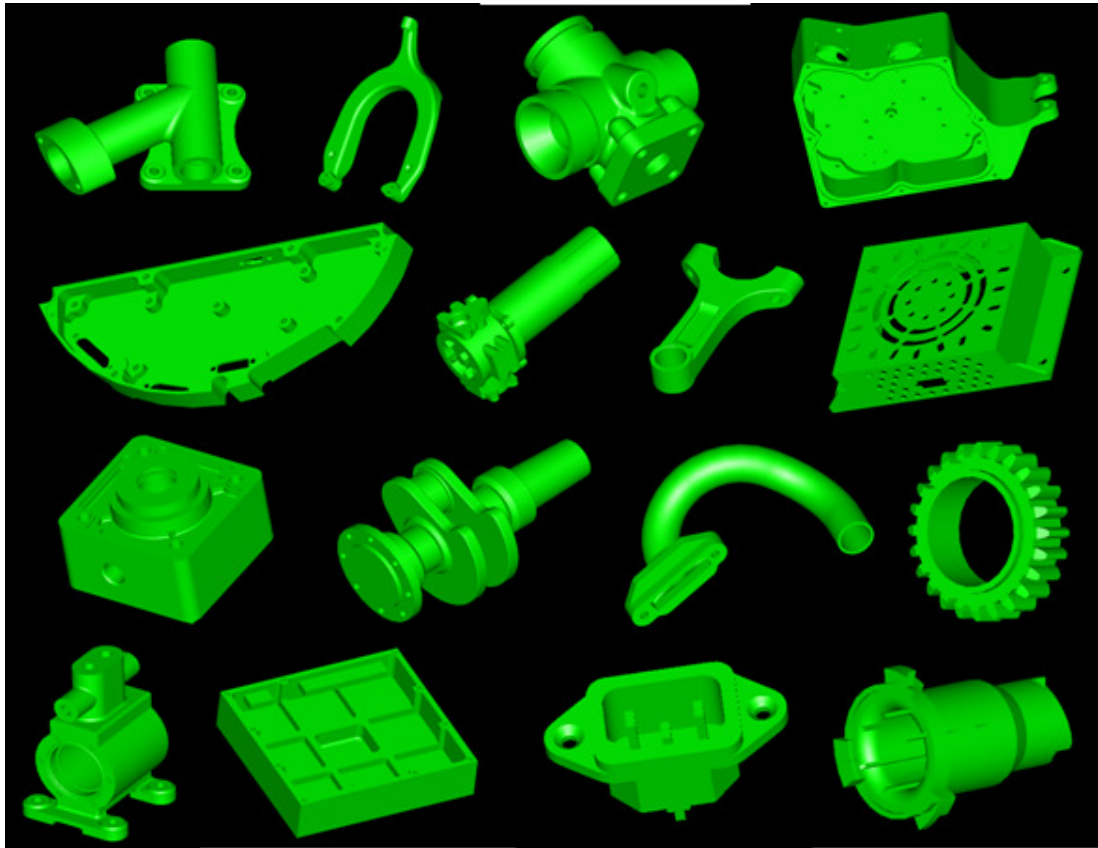


Figure 6.3 Level 3 test objects with sophisticated geometrical complexity

6.3 Modifications for Testing the PO-watermark's Robustness

After embedding the watermark information into the different test objects a series of data handling and geometrical transformations and modifications are carried out to assess the robustness of the embedded mark. The testing procedure includes the following actions:

- **1) [W faces] Watermarked faces:** Get reference readings for each watermarked model before any modifications are carried out to be able to verify any changes to the host model during the attack executions.
- **2) [Load] Loading:** Load the watermarked data into the CAD system (Autodesk Inventor Professional 11) and test whether it is fully functional.

- **3) [CAD] Conversion:** Convert watermarked CAD object into CAD related *.ipt* file format (Autodesk Inventor Part Document).
- **4) [IGES] Conversion:** Convert watermarked CAD model into IGES format and back into *.sat* format.
- **5) [STEP] Conversion:** Convert watermarked CAD model into STEP format and back into *.sat* format.
- **6) [Zip] Compression/Decompression:** Zip compression and decompression of watermarked CAD data.
- **7) [Rotation] Rotation:** Rotate and move a watermarked object inside the CAD system's 3D space.
- **8) [Healing] Healing:** Object and surface healing for CAD data repair.
- **9) [Blecha] Boolean:** Blend and chamfer edges with a constant radius blend and a 45 degree chamfer.
- **10) [Cutting] Boolean:** External Boolean modification: remove half of the model using a subtraction operation.
- **11) [Adding] Boolean:** Add new component parts/faces to the model.

The flowchart in Figure 6.4 illustrates the entire testing procedure starting with the embedding of the watermark message, followed by the execution of the various model transformations and watermark attacks, and finally the retrieval and decoding the hidden watermark. After each of the listed modifications the CAD models are converted back into ACIS *.sat* file format, and the orientation of the tweaked faces are recorded by the PO-watermark reader and the watermark message is transformed back into readable text.

The aim of this exercise is to find out how much of the embedded text message is still present after the execution of the different attacks. In other words, the more of the embedded watermark text message can be recovered the higher the success (survival or robustness) rate of the PO-watermark.

The listed modification procedures can roughly be classified into two groups, namely CAD file modification attacks (e.g. data export/import, zip compression) and CAD object geometrical modification attacks (e.g. cutting, blending and chamfer, adding or removing of faces). In the first case the CAD file is modified for different data handling and transfer purposes, while in the second case the CAD model itself is geometrically altered.

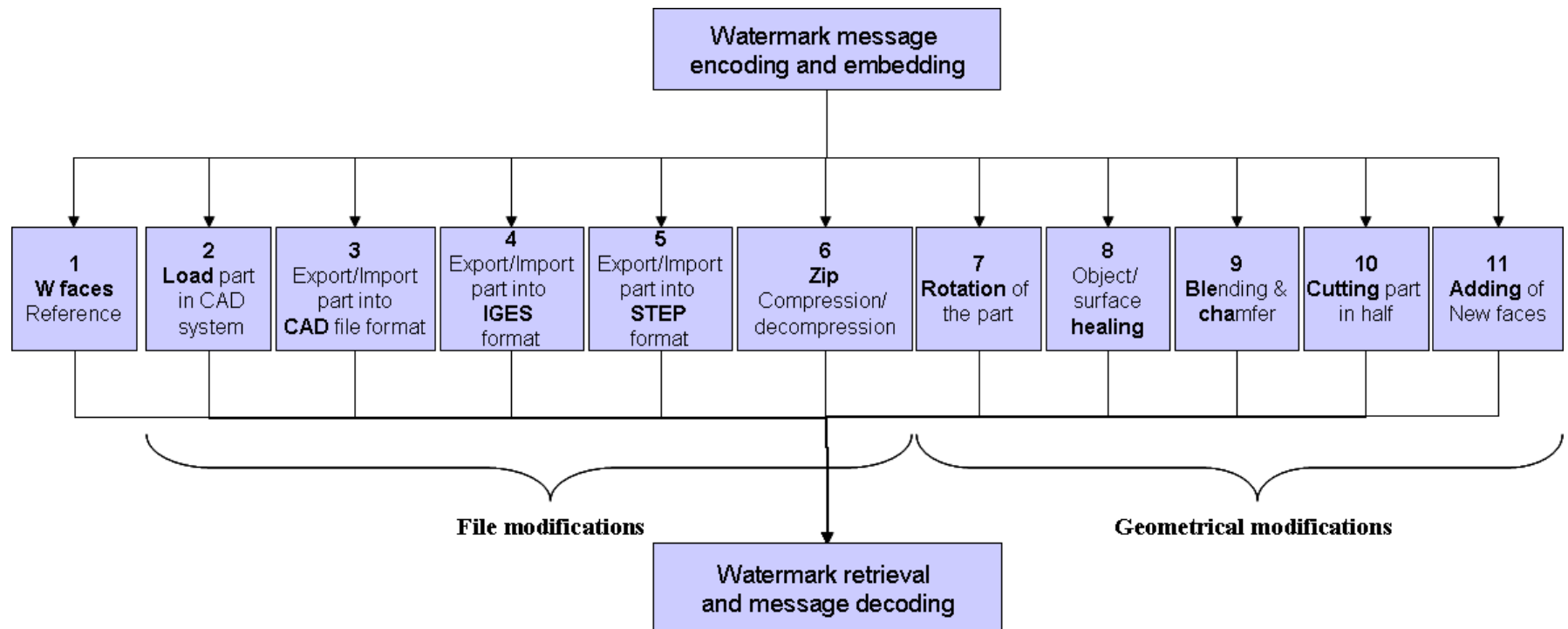


Figure 6.4 Testing procedure of the PO-watermarking scheme

Chapter 7

7 Evaluation and Discussion of Test Results

The first part of this chapter discusses the details of how the watermark message is hidden inside a B-rep model. In the following two sections the results of testing the experimental PO-watermarking scheme are presented. After that, the capabilities of the PO-watermarking scheme are discussed in the context of experimental implementation, and compared to 3D watermarking schemes reported in the literature. Finally possibilities of enhancing the PO-watermarking scheme are considered.

7.1 Mathematical Considerations of the Embedded Watermark Message

Working with native *.sat* files both watermark embedder and reader create an output summary after task completion that is displayed on the computer screen and saved to a *.wmk* text file. For further analysis the orientation data of the Cube object is summarised in Table 7.1.

Table 7.1 Output of the PO-watermark extractor based on Cube object

Original model				Watermarked model			
coordinate	X	Y	Z	coordinate	X	Y	Z
Face type:				Face type:			
Planar	0.000000	-1000.000000	0.000000	Planar	0.000000	-121.869343	992.546152
Planar	-1000.000000	0.000000	0.000000	Planar	438.371147	-0.000000	898.794046
Planar	1000.000000	0.000000	0.000000	Planar	-406.736643	913.545458	0.000000
Planar	0.000000	0.000000	-1000.000000	Planar	0.000000	992.546152	121.869344
Planar	1000.000000	0.000000	0.000000	Planar	-438.371147	-0.000000	898.794046
Planar	1000.000000	0.000000	0.000000	Planar	-241.921896	970.295726	0.000000

The data saved in the *.wmk* file shows the parametric orientation of the individual faces of a model before (left) and after (right) watermark embedding to six decimal places accuracy as shown in Table 7.1. Faces with zero values for x, y and z coordinates (as shown in the left column) carry no information. In other words, the

faces are still in their original, or initial positions (i.e. no tweaking or rotation has been conducted). Faces in the right column have a different orientation compared to the original model on the left, and therefore they might be carrying some text message. This is visible on the changed coordinate values for each face. The values for two out of three coordinates for each face have been altered. This shows that tweaking has been carried out. In some cases however, faces of watermarked models still show zero values for face coordinates after the watermarking process has been carried out. In this case, the embedding process was not successful (due to limitation of the ACIS tweaking API) or the individual faces were not considered for tweaking.

The following lines will explain how a watermark text character is incorporated into a model's face. This is illustrated by the planar face in Figure 7.1. On the left side of the figure a face's underlying parametric orientation is presented in its original position. Here, parameter lines (parallel to the face's outlines) show the parameterisation of the face before watermark embedding. In the right hand picture, however, the parameter lines have been rotated clockwise around the face's rotation axis x by angle θ .

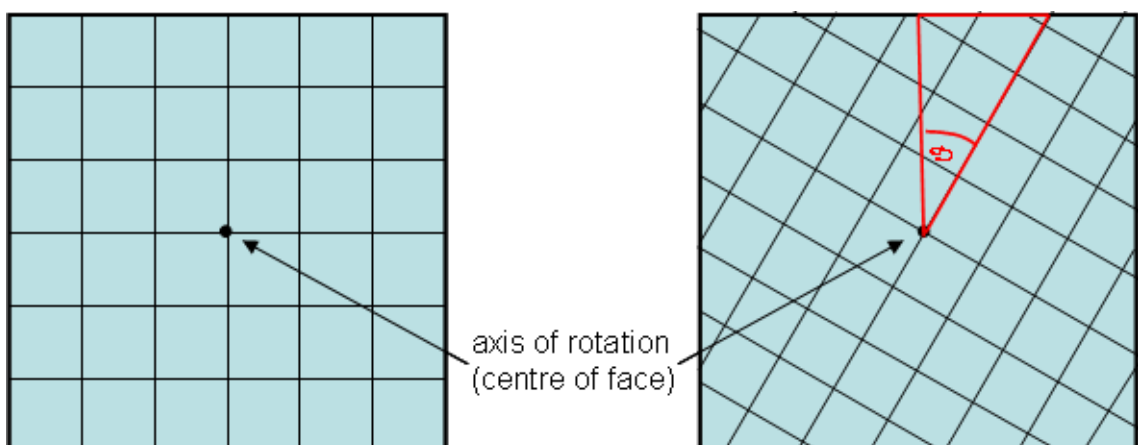


Figure 7.1 Parametric orientation of a planar face before (left) and after (after) tweaking

This angle of rotation contains the first letter of the watermark message that can be converted back into readable text. In Table 7.1 the X, Y and Z coordinates of the faces have been changed through tweaking. From these coordinates triangles can be created that represent the change in surface orientation (see example Figure 7.2).

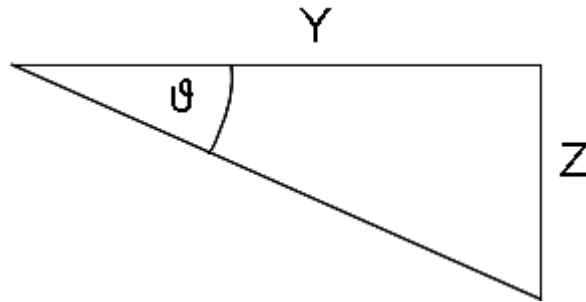


Figure 7.2 Schematic triangle representing parametric reorientation

By taking the Z and Y values of the first planar face from Figure 7.1 (circled in red) one can calculate/regenerate the embedded text character of the watermark in the following way:

$$\tan \vartheta = \frac{Z}{Y} \Rightarrow \vartheta = \tan^{-1}\left(\frac{Z}{Y}\right) = \tan^{-1}\left(\frac{992.546152}{121.869343}\right) = 83$$

The value 83 means that the first planar face has been rotated by 83 degrees. Now the decimal value 83 is converted back with the ASCII-decimal converter which is equivalent to the capital letter ‘S’ in the English alphabet, which again is the first character of the embedded watermark message “Strath”. By carrying on with the conversion of the triangle values into text characters of the remaining planar faces from the example object in Table 7.1 one can assemble the word ‘Strath’ that has been encoded with the PO-watermark embedder as the secret message.

The following Figure 7.3 shows how the reconversion procedure is executed within the PO-watermark recovery scheme. As described in chapter 5, the two matching faces (original face and watermarked face) are aligned and the angle difference computed (in radian). The gained value is then converted back first into decimal value and then into text characters. In the final stage the text characters are assembled to reproduce the watermark message (Output = embedded word). At the end of the recovery process a summary is printed to the computer screen and saved to a *.wmk* file.

```

Cuberes - Notepad
File Edit Format View Help
After watermark reading:
Plane:
Original face: 0.000000,-1000.000000, 0.000000
Watermarked face: 0.000000,-121.869343,992.546152
Angle difference: 1.448623
Decimal value is: 83
Text value is: s
Watermark = Embedded word = s

Plane:
Original face: -1000.000000, 0.000000, 0.000000
Watermarked face: 438.371147, -0.000000,898.794046
Angle difference: 2.024582
Decimal value is: 116
Text value is: t
Watermark = Embedded word = St

Plane:
Original face: 1000.000000, 0.000000, 0.000000
Watermarked face: -406.736643,913.545458, 0.000000
Angle difference: 1.989675
Decimal value is: 114
Text value is: r
Watermark = Embedded word = Str

Plane:
Original face: 0.000000, 0.000000,-1000.000000
Watermarked face: 0.000000,992.546152,121.869344
Angle difference: 1.692969
Decimal value is: 97
Text value is: a
Watermark = Embedded word = Stra

Plane:
Original face: 1000.000000, 0.000000, 0.000000
Watermarked face: -438.371147, -0.000000,898.794046
Angle difference: 2.024582
Decimal value is: 116
Text value is: t
Watermark = Embedded word = Strat

Plane:
Original face: 1000.000000, 0.000000, 0.000000
Watermarked face: -241.921896,970.295726, 0.000000
Angle difference: 1.815142
Decimal value is: 104
Text value is: h
Watermark = Embedded word = Strath

Output = Embedded word = Strath

```

Figure 7.3 Output of the PO-watermark reader

7.2 Produced Test Data

After demonstrating how the watermark message is incorporated into a CAD model, this section presents the results of the PO-watermark robustness testing. As mentioned in Section 6.2.2 the test CAD objects are grouped into three different classes:

- Level 1 CAD models with low geometrical complexity with up to 25 faces per object. Within this class, CAD objects have an average of 11 faces.
- Level 2 CAD models with medium geometrical complexity that have between 25 and 75 faces. The average number of faces in this class is 41.
- Level 3 CAD models with high geometrical complexity and more than 75 faces. The average number of faces in this class is 173.

In the first column called [Object] of Table 7.2 Level 1 test objects (as displayed in Figure 6.1 in Section 6.2.2) are listed. The message ‘Strathclyde’ was embedded in all CAD models using the PO-watermarking scheme. The CAD models were then subjected to the various types of intentional and unintentional attacks, as introduced in Section 6.3.

The second column [No all faces] shows the total number of all available faces in each model, while the third column [W faces] presents the number of faces that have been successfully watermarked. In the fourth column [Percentage] the PO-watermarking success rate for each individual CAD object is calculated. For example, looking at the objects Bolt1 and Clip: Bolt1 has a total number of 14 faces and all 14 faces were watermarked [W faces]. Clip, on the other hand, has 16 faces

and only 8 were watermarked [W faces]. This would give the Bolt1 a watermarking success rate of 100 percent and only 50 percent for Clip.

In the right bottom corner of the table the average watermarking success rate for Level 1 objects is shown.

Table 7.2 Watermarking success rate for Level 1 test objects

Watermarking success rate:			
Object	No all faces	W faces	Percentage
Ball1	1	1	100%
Ball2	4	3	75%
Bolt1	14	14	100%
Bolt2	20	14	70%
Clip	16	8	50%
Cone	2	1	50%
Cube	6	6	100%
Flange	15	11	73%
Link	8	6	75%
Linkrod	8	8	100%
Nozzle	20	18	90%
Nut	12	4	33%
Pipe1	4	4	100%
Pipe2	12	2	17%
Pumpshaft	21	18	86%
Retainer	12	8	67%
Shaft	12	12	100%
Average	11	8.0	75.00%

In Tables 7.3 and 7.4 the first three columns are organized in the same way as in Table 7.2 where the object names [Object], the total number of available faces [No all faces], and the number of watermarked faces [W faces] for each test object are presented. The columns four to eight show the number of faces that have ‘survived’ unintentional CAD data handling attacks (Table 7.3) and intentional geometrical modification attacks (Table 7.4). Unintentional attacks include loading of the marked CAD file into a CAD system [Load], conversion of the marked *.sat* file into CAD specific (i.e. *.ipt*) file format [CAD], neutral IGES [IGES] and STEP [STEP] formats, and compression/decompression [Zip] of the watermarked object.

Table 7.3 Level 1 test objects after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
Ball1	1	1	1	1	0	0	1
Ball2	4	3	3	3	0	0	3
Bolt1	14	14	14	14	4	4	14
Bolt2	20	14	14	14	0	0	14
Clip	16	8	8	8	0	0	8
Cone	2	1	1	1	1	1	1
Cube	6	6	6	6	0	0	6
Flange	15	11	11	11	1	2	11
Link	8	6	6	6	0	2	6
Linkrod	8	8	8	8	1	1	8
Nozzle	20	18	18	18	3	4	18
Nut	12	4	4	4	1	1	4
Pipe1	4	4	4	4	1	2	4
Pipe2	12	2	2	2	0	0	2
Pumpshaft	21	18	18	18	7	7	18
Retainer	12	8	8	8	0	2	8
Shaft	12	12	12	12	7	7	12

Similarly, Table 7.4 shows the number of faces that have ‘survived’ intentional geometrical modifications attacks including object and surface healing [Healing], rotation and movement of the watermarked model inside a CAD system’s 3D space [Rotation], blending and chamfer of edges [Blecha], external Boolean modification [Cutting], and adding new components and faces to the marked model [Adding].

Table 7.4 Level 1 test objects after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
Ball1	1	1	1	1	0	0	0
Ball2	4	3	3	3	0	0	0
Bolt1	14	14	14	14	4	11	10
Bolt2	20	14	14	14	7	12	10
Clip	16	8	8	8	2	3	6
Cone	2	1	1	1	0	0	1
Cube	6	6	6	6	0	1	4
Flange	15	11	11	11	4	2	8
Link	8	6	6	6	0	1	4
Linkrod	8	8	8	8	5	0	0
Nozzle	20	18	18	18	6	2	15
Nut	12	4	4	4	1	0	2
Pipe1	4	4	4	4	0	0	0
Pipe2	12	2	2	2	0	1	1
Pumpshaft	21	18	18	18	8	9	16
Retainer	12	8	8	8	3	3	6
Shaft	12	12	12	12	5	0	10

In Tables 7.5 and 7.6 the survival rate for each watermarked Level 1 object is presented (in %). Again, the first three columns are arranged in the same way as the previous tables above, where the object names [Object], the total number of available entities [No all faces], and the number of watermarked faces [W faces] for each test object are shown. The columns four to eight, however, show the survival rate of the PO-watermark after the different unintentional data handling attacks [Load], [CAD], [IGES], [STEP], and [Zip] (in Table 7.5), and intentional geometrical modification attacks [Healing], [Rotation], [Blecha], [Cutting], and [Adding] (in Table 7.6) for each Level 1 CAD object were executed. In the bottom row of each table the average survival rate after each modification type is displayed.

Table 7.5 Level 1 watermark survival rate after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
Ball1	1	1	100%	100%	0%	0%	100%
Ball2	4	3	100%	100%	0%	0%	100%
Bolt1	14	14	100%	100%	29%	29%	100%
Bolt2	20	14	100%	100%	0%	0%	100%
Clip	16	8	100%	100%	0%	0%	100%
Cone	2	1	100%	100%	100%	100%	100%
Cube	6	6	100%	100%	0%	0%	100%
Flange	15	11	100%	100%	9%	18%	100%
Link	8	6	100%	100%	0%	33%	100%
Linkrod	8	8	100%	100%	13%	13%	100%
Nozzle	20	18	100%	100%	17%	22%	100%
Nut	12	4	100%	100%	25%	25%	100%
Pipe1	4	4	100%	100%	25%	50%	100%
Pipe2	12	2	100%	100%	0%	0%	100%
Pumpshaft	21	18	100%	100%	39%	39%	100%
Retainer	12	8	100%	100%	0%	25%	100%
Shaft	12	12	100%	100%	58%	58%	100%
Average of each action:			100%	100%	18%	24%	100%

Table 7.6 Level 1 watermark survival rate after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
Ball1	1	1	100%	100%	0%	0%	0%
Ball2	4	3	100%	100%	0%	0%	0%
Bolt1	14	14	100%	100%	29%	79%	71%
Bolt2	20	14	100%	100%	50%	86%	71%
Clip	16	8	100%	100%	25%	38%	75%
Cone	2	1	100%	100%	0%	0%	100%
Cube	6	6	100%	100%	0%	17%	67%
Flange	15	11	100%	100%	36%	18%	73%
Link	8	6	100%	100%	0%	17%	67%
Linkrod	8	8	100%	100%	63%	0%	0%
Nozzle	20	18	100%	100%	33%	11%	83%
Nut	12	4	100%	100%	25%	0%	50%
Pipe1	4	4	100%	100%	0%	0%	0%
Pipe2	12	2	100%	100%	0%	50%	50%
Pumpshaft	21	18	100%	100%	44%	50%	89%
Retainer	12	8	100%	100%	38%	38%	75%
Shaft	12	12	100%	100%	42%	0%	83%
Average of each action:			100%	100%	23%	24%	56%

The results for Level 2 and Level 3 test objects are presented in the same way as for Level 1 test objects. Here, the number of faces vary between 25 and 75 for Level 2 models and more than 75 faces per model for Level 3 objects. The individual result tables can be found in Tables C0.1 to C0.10 in appendix C.

7.3 Discussion of Test-Result Data

When comparing the PO-watermarking success rate (i.e. the percentage of faces onto which the watermark had been successfully applied) between the three object level groups one can see that the rate is dropping as the CAD models are getting larger and more complex. For instance, the average watermarking success rate for Level 1 objects is 75 percent (Table 7.2). For Level 2 and Level 3 models the numbers are decreasing to around 65 percent (Table C.1) and 48 percent (Table C.6), respectively. One explanation for this is that when CAD models are getting larger and more complex some other face types like splines, open conical faces or curves are likely to be involved but are currently not considered for watermark embedding by the PO-

watermarking scheme. In other words, since the PO-watermarking method is based on marking simple geometrical faces (i.e. planar, closed conical, spherical, and torus) other more complex and differently shaped face types are not taken into account. As a result of this, they are not considered for watermarking and therefore they lower the overall watermarking success rate.

Focusing on simple geometrical faces does not severely restrict the practical use of the system since 97% of all mechanical CAD data contain at least one or more of the following five basic shapes: plane, sphere, cylinder, cone and torus (Mills et al., 2001). As the PO-watermarking scheme is based on four of the five listed face types this method is well suited for watermarking mechanical CAD data. However, since the PO-watermarking method is in the development stage further testing is required to establish its full potential and capabilities in terms of watermarking efficiency.

The test results in Tables 7.5 and 7.6 show that data handling (or unintentional) attacks like [Load] (loading data into a CAD system), [CAD] (conversion of *.sat* file format into CAD data format), [Zip] (Zip compression and decompression), [Healing] (repairing of gaps and overlaps in CAD models) and [Rotation] (moving and rotating a model inside the CAD system's working space) have no influence on the embedded watermark, i.e. despite these modifications the watermark reader was still able to fully (100 percent) read and recover the initially embedded watermark message. Seeing such high success rate is an indication that the PO-watermark is robust enough to withstand day-to-day data handling, or unintentional, attacks that are common in the CAD design world.

The tests have also shown some unexpected results regarding translation into neutral formats: for example, watermarks applied to planar faces survive the IGES

translation process, but not STEP, whereas cylinders survive STEP but not IGES translations. At first thought one might expect the behaviour of the IGES and STEP translators to be predictable. The IGES and STEP standards define the types of surface entities each can represent, and so it should be clear if parametric orientation will be preserved. However, the implementer has a choice in both cases as to which entities a cylinder is mapped to and how the geometry is represented internally within the translation systems. Some neutral systems translate all geometric surface entities to NURBS, while others hold analytical representations. In many cases the preservation of parametric orientation will be down to a programmer's whim rather than the specific requirements of Part AP204 of ISO 10303⁹. Also, CAD data translation procedures and processes into neutral file formats have always been considered unresolved and in need of further research (IronCAD, 2002). Consequently, the behaviour of translation PO-watermarks into STEP and IGES can only be determined experimentally at this stage and is an area of further study (see Section 8.3). However, the results also suggest that if a model has sufficient faces, a watermark that was duplicated (i.e. the same watermark is embedded multiple times to cover all relevant faces) between planar and cylindrical faces would survive at least one of the translation attacks which might be enough to re-assemble the embedded text message. Thus, the PO-watermarking scheme could be applied until a definite solution for CAD data translation into neutral file formats is found.

The results of [Cutting], blending and chamfer [Blecha] and [Adding] are self-explanatory: any efficient algorithm will leave geometry not involved in the

⁹ ISO 10303 is an ISO standard for the computer-interpretable representation and exchange of product manufacturing information. The International standard's objective is to provide a mechanism that is capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

operation untouched and so it is reasonable to conclude that unless the watermarked surface is removed from the model, its orientation will not be changed. Even adding new components and faces to the host object will have no influence on the watermark recognition and recovery. However, when reading out the watermark after blending and chamfer modifications [Blecha], the reader shows sometimes more and sometimes fewer tweaked faces than actually exist. This behaviour is also called false positive identification or detection. This performance reflects the process used to create chamfers where parametric orientation is inherited from the adjacent faces during the edge offset stage. In other words, these newly created faces have the same orientation as the faces they were created from, but have no matching faces to the original (i.e. unmarked) model for extraction/verification purposes. Here again the behaviour pattern of the Blecha modification on watermarked faces is an area of further study.

Another observation was made when comparing the average watermarking success rate of the three levels of objects after the different intentional and unintentional attacks had been carried out. As Figure 7.4 shows, the average watermark retrieval success rate for all three levels is the same for [Load], [CAD], [Zip], [Healing], and [Rotation] attacks types (the Healing rate for Level 3 objects is slightly lower because one object failed the read out phase; it is assumed that this is object-specific). For [IGES] and [STEP] modifications, it appears that the more complex a model gets the less likely it is that the watermark survives these modifications (the influence of IGES and STEP on CAD models has already been mentioned above).

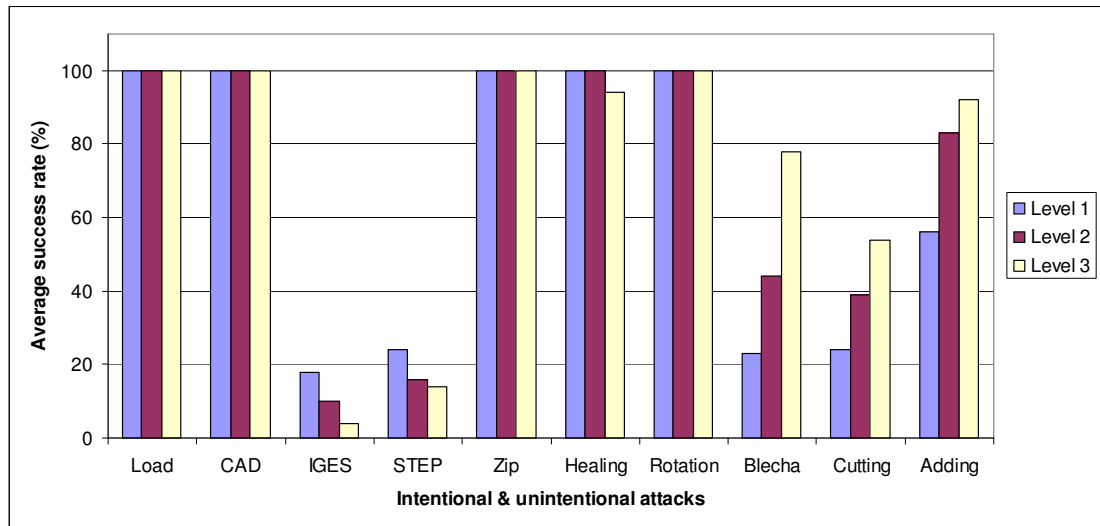


Figure 7.4 Watermark recovery success rate after various attacks

However, for [Blecha], [Cutting] and [Adding] modifications the opposite seems to be true: with increasing object size and complexity, success rates increase. This situation can be explained by the fact that a larger and complex model has more watermarked faces so that the same extent of modification affects fewer faces (e.g. blending of 5 faces on a 10-face model might result in a 50% successrate, while 5 faces of a 100-face model would result in a 95% success rate). Therefore the probability of intact watermarks is higher. Test results for geometric modifications indicate that as long as a CAD model has sufficient watermarked faces after a modification, the PO-watermark reader is able to recover, translate and assemble the hidden text message which makes this approach real-world applicable.

7.4 Discussion on the Requirements for a Watermarking Scheme

General requirements (as listed in section 5.1)

One of the requirements for a CAD data watermarking scheme is to embed a set of text information into a CAD model, and the detector should be able to retrieve and decode the embedded watermark message. As shown in Figure 5.2.1.2 the PO-

watermarking system is capable of converting text information into tweaking parameters and to hide the intended message into a B-rep CAD model by modifying the underlying parametric surface of the B-rep face. The watermark detector is also capable to identify and match modified faces, and to re-convert the embedded watermark message into readable text (as shown in Figure 7.3).

Another important requirement for a sophisticated CAD model watermarking system is to not to change the topology of an object's geometry to have no influence on the model's appearance. Since the PO-watermarking scheme is based on tweaking or rotating the underlying parametric orientation of a face, it will not change its topology (the two main topological items of a face are EDGES and VERTICES; see B-rep data structure in Section 3.3.2). Therefore, as long as these two components are not altered in any way the topology of the face (and so the CAD model as a whole) remains unchanged.

Also a major requirement is to not to increase the file size massively when embedding or hiding a watermark message into a CAD model. The tweaking Application Procedural Interface (API) inside the PO-watermarking scheme makes it possible to modify the geometry of a significant number of entities on moderately complex parts without significantly increasing the data size or changing the model's shape. As the example in Table 7.7 shows the PO-watermark mostly did not increase the size of the model data¹⁰ and still survived some of the file conversion and model modification processes.

Table 7.7 Data structure before and after surface tweaking

¹⁰ The difference in file size in Table 7.7 is due to a side effect of the tweaking api which created a bulletin board attribute (the "24 attribute records" in the right column) for each of the changed faces. These can be easily removed and are not an inherent part of the model.

Debug file of original model:		Debug file of watermarked model:	
1 body record,	40 bytes	1 body record,	40 bytes
1 lump record,	40 bytes	1 lump record,	40 bytes
1 shell record,	48 bytes	1 shell record,	48 bytes
6 face records,	336 bytes	6 face records,	336 bytes
6 loop records,	312 bytes	24 attrib records,	1104 bytes
6 surface records,	1056 bytes	6 loop records,	312 bytes
24 coedge records,	1248 bytes	6 surface records,	1056 bytes
12 edge records,	960 bytes	24 coedge records,	1248 bytes
8 vertex records,	256 bytes	12 edge records,	960 bytes
12 curve records,	1536 bytes	8 vertex records,	256 bytes
8 point records,	512 bytes	12 curve records,	1536 bytes
Total storage	6344 bytes	8 point records,	512 bytes
		Total storage	7448 bytes

Another crucial requirement for a watermarking system is to create a robust connection between the watermark message and the carrier object. As already discussed in sections 7.1 and 7.2 the PO-watermarking scheme fulfils this requirement because the watermark message itself is the rotation parameter, which is now part of the object, and was able to survive (i.e. stayed robustly attached to the model) most file conversion and geometrical modifications it was exposed to.

In the case of the PO-watermarking scheme the amount of information that can be robustly embedded into a CAD object depends on the actual carrier model's size (to be more precise on the number of faces inside the object). In other words, the larger a model, the more faces it has and so the more information can be embedded.

Last but not least, it is also an important requirement for a watermarking scheme to distribute the watermark message evenly throughout a CAD model to overcome directed attacks on specific parts or sections of the model (e.g. cutting parts of the model out to 'cut out' the watermark). The best example to show that the PO-watermarking scheme fulfils this requirement is when a watermarked B-rep model is cut in half. After this attack the watermark reader was still capable of recording enough bits and parts of the hidden message to recreate the embedded message

‘Strathclyde’. However, there is a limitation to this requirement specifically when very small objects with only a few faces are involved. In other words, the outcome of the Boolean operation (i.e. cutting the model in half) is determined by how the cut was made. Take a cylinder as an example which contains three faces: two ‘lids’ (top and bottom) and one cylindrical ‘jacket’. Now, there are two possibilities to cut the cylinder in half, namely horizontally and vertically, which are shown in Figure 7.5.

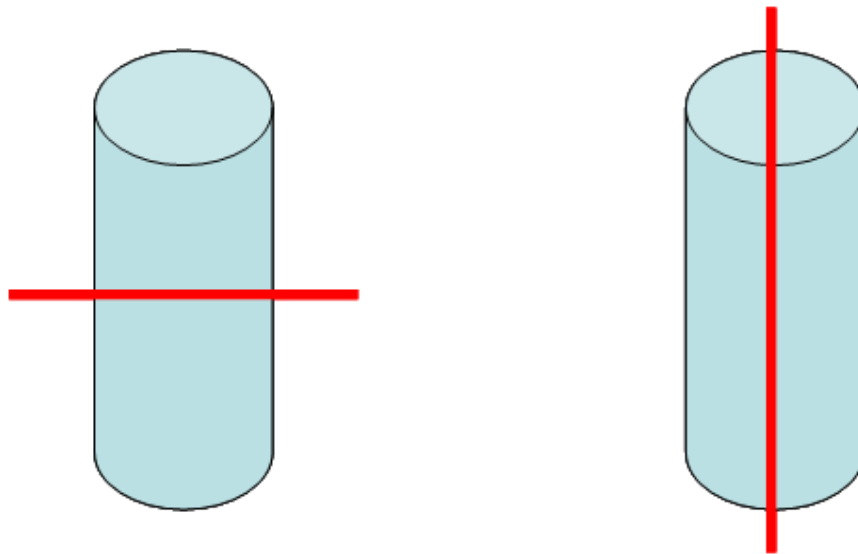


Figure 7.5 Ways of cutting objects

When the cut is executed horizontally and the bottom half is removed (like in the left picture) it is still possible to get the orientation of the top lid, since it can be matched against the top lid of the original model. However, if the cut is executed vertically (as shown in the right picture) all three faces are ‘destroyed’ and so matching it against the original model is no longer possible.

Industry’s requirements

Looking specifically at the requirements expressed by the surveyed companies, the PO-watermarking system fulfils most of them:

- Carry text information

Most surveyed industrial partners stated that they wanted to incorporate some sort of text information into the CAD data. This requirement is met fully since the PO-watermarking scheme was specifically designed to embed text messages into CAD models. An example of a text message being extracted from a CAD model is demonstrated in Figure 7.3.

- Watermark must survive modifications

Since modifications of any type are the highest possible ‘threats’ to the embedded watermark (see Table 6.1), it is very important that the watermark is anchored robustly within the CAD data. As demonstrated in sections 7.2 and 7.3 the PO-watermark withstands most modifications with very threatening its robustness. However, even after those modifications which had some impact on the watermark, the PO-watermark reader was able to recover enough information to reconstruct the embedded message.

- No interference with the CAD model’s capabilities

The PO-watermarking scheme embeds the watermark information into a CAD model by readjusting the faces’ underlying parametric orientations. As a result of this, the overall shape (i.e. geometry and topology) of the CAD model is unchanged and so are the model’s capabilities.

- Watermark message presented in front of CAD model (e.g. “confidential”)

Currently the PO-watermarking method is not designed to display the embedded watermark message when the CAD file is loaded. However, the scheme could easily be extended to offer a ‘display of the encoded message’ function, although further research is required.

When considering the test results in sections 7.2 and 7.3, one can see that the PO-watermarking method appears to offer a simple and effective way of encoding, embedding and retrieval of text messages that could be used to identify and track B-rep models within a homogenous industrial CAD environment (i.e. one supplier, no translation). Within an appropriate PDM or DRM system the embedded PO-watermark could be used to carry copyright information, transport product or project relevant information to restrict and regulate access, or to monitor CAD data distribution. However, this study's primary objective was to investigate the feasibility to encode and embed text information robustly within 3D mechanical B-rep CAD data and the design of a fully functional commercial watermarking system is beyond the scope of this thesis.

7.5 Comparison of the PO-watermarking Scheme against Existing Watermarking Methods and Techniques

As already mentioned in Section 4.2.5, most proposed watermarking methods and schemes focus on model representation types like polygonal mesh, Bézier, NURBS or CSG data rather than on boundary representation (B-rep). Also, since these methods are based on some form of transformation techniques (e.g. wavelet, frequency, Fourier or geometrical) which modify surface/vertex coordinate values to insert a watermark, it is difficult to compare the PO-watermarking scheme for the following reasons:

1. Most listed watermarking approaches embed random bit information into surface carrier models, while the PO-watermarking technique inserts text characters/information into solid models. This scheme also recovers and reassembles the watermark information, instead of only showing the presence of a watermark.

2. The PO-watermarking scheme does not change topology or geometry of a CAD model when inserting the watermark message, while most existing methods do typically by displacing vertices which results in a change of the model's overall shape. These geometrical and topological changes might not be visible to the naked eye, but are not tolerated in mechanical CAD systems, especially in finite element analysis where the preservation of connectivity of elements is essential. In this sense, the PO-watermarking scheme delivers a more appropriate and precise method for mechanical CAD data (i.e. less geometrically destructive).

3. As already shown in Section 4.2.5 most embedded watermarks of the existing methods have robust characteristics. However, those watermarks can withstand only certain types of attacks (e.g. affine transformations or mesh simplification which are merely of academic value) and are not tested against 'real-world' type attacks that are common to a CAD environment, e.g. file conversion, object healing, file zipping, etc. Therefore, it is difficult to compare the robustness of reported 3D watermarks and the PO-watermarking scheme because they were assessed by using different criteria (i.e. theoretical/mathematical modifications compared to the real-world modifications for the PO-watermarking method) based on a different set of test objects.

4. Similarly, it is also difficult to compare the test results from the PO-watermarking scheme with reported techniques, because they were created using a different set of test-data. Those approaches typically used 'random' polygonal mesh surface based models (e.g. rabbit, dragon, Beethoven statue, etc.) to test their capabilities, while for PO-watermarking a set of real-world solid mechanical CAD

objects were chosen and tested under real-world conditions. Therefore, the outcomes of the schemes are mostly very different and cannot be compared.

As a consequence, the academic watermarking and CAD community need to work together to create a system (i.e. testing procedures and test objects) that allows fair benchmarking and performance evaluation of the different watermarking approaches. However, if we consider the main focus of this thesis, it is clear that the PO-watermarking scheme is better suited for marking solid CAD data than any of the other previously presented approaches.

7.6 Possible Applications of the PO-watermarking Approach

The PO-watermarking approach described within this thesis could be used for both fragile and more robustness-enhanced watermarking purposes. The major difference between the two applications is the length and frequency with which the watermark message is embedded and retrieved. The next two sections will highlight the particular features of both fragile and robust watermarking.

7.6.1 Fragile Watermarking

The watermark embedding for a fragile system is accomplished as described in section 5.2.1, but with the difference that the watermark information can have an ‘unlimited’ length (as long as it does not exceed the maximum storage capacity of the model – i.e. one character per face multiplied by the number of available/suitable faces) and is embedded only once. The reason for using one long text message is to create enough reorientation parameters to include all, or as many faces of the model as possible in the embedding process. The reason for embedding the watermark message only once is to reduce the opportunity for the attacker to recreate the mark.

In other words, when someone modifies or changes the carrier CAD model (e.g. editing, erasing or adding new faces) the fragile watermark ‘brakes’ (in the sense that parts or characters of the message are lost and cannot be recovered) and consequently the decoding and reassembling of the embedded watermark message is not possible. In this case the carrier data loses its authenticity.

A fragile watermark message could be created as shown in Figures 7.6 and 7.7. First, the user enters the text information (e.g. product or security related information) into the system’s interface (Figure 7.6). Then the programme arranges the text information into one long single line (Text string) as shown in Figure 7.7 and converts the characters one by one into decimal values (Decimal string). Now the decimal string is divided up into three-digit numbers (Decimal parameters) which, in turn, are converted into radian values that represent the tweaking parameters. Finally, the message is embedded into all suitable faces of the CAD model with the help of the embedding algorithm (as described in Section 5.2.1.2).

Name:	Csaba_Salamon
Affiliation:	University_of_Strathclyde
E-mail address:	csaba.salamon@strath.ac.uk

Figure 7.6 Entry of information for fragile watermarking

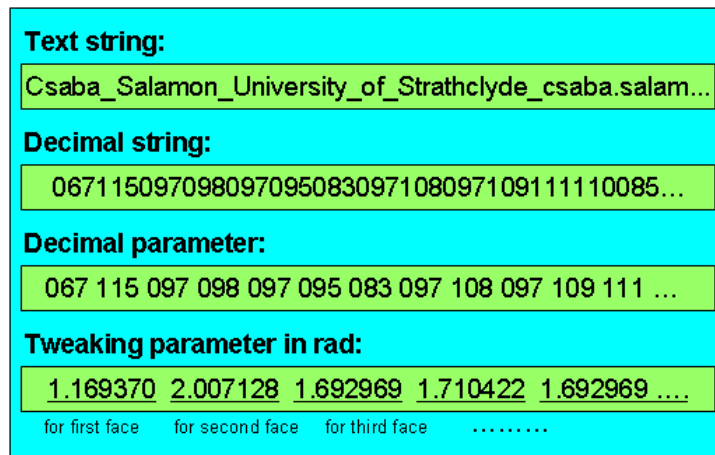


Figure 7.7 Information conversion and encoding for fragile watermarking

Because the embedding is carried out only once, the message retrieval for fragile watermarking is similarly carried out only once. The reason for that is if someone has attacked (i.e. modified) the CAD data, the internal structure and features of the model (such as face type, face area, position within the face list) might have changed. Therefore, the CAD model is no longer the same as the one before the attack and it is no longer possible to match the corresponding faces of the original and watermarked models to recreate the full hidden message. In this case, a CAD model would lose its authenticity and consequently the fragile watermarking delivers a 100% success rate.

7.6.2 Robustness-enhanced Watermarking

For enhanced robustness in watermarking schemes it is desirable that the embedded information is widely and evenly spread across the model. Although the exact nature of the embedded information is not crucial for the robust characteristic, it should be noted that even small amounts of data can verify the ‘correctness’ of large and complex files.

As shown in sections 5.2.1.2 and 5.2.1.3 the current PO-watermarking approach is able to embed one text character into one CAD model-face, and is able to withstand

various transformation and geometrical attacks (see Section 7.2). However, the kernel modeller within this scheme could be extended to have sufficient precision (i.e. by increasing decimal places for more accuracy) to embed two text characters into one single face to enhance robustness. To increase robustness and information-content per face the following steps are needed:

1. Take a given word and fragment it into characters.
2. Add a sequence or position number in front of each character (e.g. a number or letter).
3. Convert unified characters (position number and letter) one by one into three digit decimal values.
4. Join the two triple digit decimal numbers to create one six digit number with three decimal places (e.g. 049 and 077 to 49.077°).
5. Convert the newly created six digit degree value into radians for surface reorientation (e.g. 0.856555 rads)
6. Embed the watermark text message as often as possible into the carrier object (as long as suitable faces are available, or until model is “full”)

The following paragraphs provide a brief example of how the embedding procedure for enhanced robust watermarking could work. A short message code “Mouse” is typed into the system (Figure 7.8). Then, a sequence number is added to each character (e.g. position 1 for M, position 2 for o, position 3 for u, etc.) and the Text string is created (see Figure 7.9). In this way the characters of the watermark message on each face contain information analogous to a ‘packet’ number in digital communication protocols. Then the ‘packages’ are converted into decimal values, Decimal string, and assembled into a six digit number with three decimal places,

Decimal parameter. These decimal parameters are then converted one by one into Tweaking parameters (i.e. radiant values) and fed into the embedding algorithm.

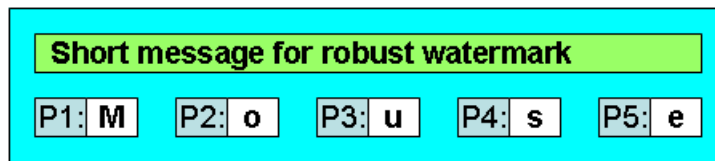


Figure 7.8 Entry of information for robust watermarking



Figure 7.9 Information conversion and encoding for robust watermarking

The message recovery for robust watermarking could be executed in the same way as described in section 5.2.2. Adequate faces of the original and the watermarked models are ordered and aligned and the orientation of their parametric geometry is computed. While the fragile watermark is read out only once, the robustness-enhanced watermark is retrieved several times. In this way several duplicates of the embedded message code can be recovered. Should there be any changes to the watermarked model through translation or any type of modification attack where parts of the object (and the embedded message) are removed or modified, the scheme would still be possible to recreate the whole watermark message (because each character of the message has been recorded several times) despite the risk that the

mark might have been damaged or destroyed. The recovery steps can be summarised as follows:

1. Read out orientation angles from 15 matching faces of the CAD model (Figure 7.9 illustrates only 15 readings but obviously this can be extended according to the user's specification).
2. Convert orientation/radian parameters back into decimal values (i.e. 0.856555 rads back to 049.077°).
3. Separate six digit numbers into two three digit numbers (i.e. 049.077° back to 049 and 077).
4. Reconvert three digit decimal values one by one into readable text by using ASCII-decimal converter (i.e. 049 = 1, and 077 = M).
5. Sort message characters according to their position/sequence number (the first character/number that will be recovered determines the following/text character's position within the watermark message) and assign them to rows in the message recovery matrix (see Figure 7.10).
6. By identifying common elements in each column (i.e. the character position) the message recovery matrix reconstructs the originally embedded watermark message.

Retrieved message codes					
	P1:	P2:	P3:	P4:	P5:
Reading 1	M				
Reading 2		o			
Reading 3	M				
Reading 4				s	
Reading 5		o			
Reading 6			u		
Reading 7					e
Reading 8					e
Reading 9				s	
Reading 10		o			
Reading 11			u		
Reading 12					e
Reading 13	M				
Reading 14				s	
Reading 15			u		
Result:	M	o	u	s	e

Figure 7.10 Message recovery matrix

The scheme described is theoretical and further work needs to be carried out to determine if the current six decimal places for radians are sufficient to recreate the exact same six digit number with three decimal places (i.e. degree value). In other words, if the watermark reader recovers, say, 1.939078 rads instead of 1.939095 rads the degree value would change from 111.102 to 111.101 deg and the embedded message would change it's meaning from 'of' to 'oe'.

Both of these PO-procedures, fragile and robustness-enhanced watermarking can be used to embed text information of any kind (e.g. archiving, copyright, product or

manufacturing, or even confidential information). The first procedure would be useful in establishing authenticity, while the second one is able to withstand CAD related attacks on the model. Especially the robustness-enhanced encoding method could be combined with a message digest algorithm (see section 1.2.2 for message digest) to embed larger amounts of text information. The advantage of using message digest codes is that, even if an attacker decodes the PO-watermarking algorithm, the hidden message is useless without the right message digest algorithm.

Chapter 8

8 Discussion and Conclusion

This chapter is organised in three parts: it starts with a summary of the research and the key findings; it continues with the study's contribution to knowledge, and ends with a discussion of its limitations and directions for future research.

8.1 Summary

This research presented a new digital watermarking method that allows the embedding of text information into Boundary representation surfaces of solid three dimensional CAD models. The study also developed a testing and evaluation process for assessing the effectiveness of the PO-watermarking scheme under real-world conditions found in industrial CAD environment.

The thesis started with an overview of various techniques for hiding information into various forms of digital data, and also presented some background information on industrial applications of digital watermarking. This was then followed by an overview of the historical development of digital watermarking on 3D data, a short general overview of the watermarking process, the various attack types on watermarked data, and the requirements for a watermarking approach for 3D CAD models.

Chapter 3 then presented various representation formats for three dimensional CAD data, while Chapter 4 listed watermarking techniques for one, two, and three dimensional digital data. The literature review showed that existing studies were largely focused on watermarking polygonal mesh, Bézier, and NURBS representation formats and a few methods developed for specific carrier data types. However, considering the need for CAD applications and comparing the

characteristics of the listed three-dimensional representation methods, only B-rep seemed to be able to deliver and support the required attributes and features for a complete, valid and unambiguous representation of 3D objects.

It was observed that most of the 3D watermarking approaches surveyed had robust characteristics, and about half of the schemes required access to the original data during the watermark detection and retrieval stage. However, current techniques and methods for 3D watermarking alter shape, geometry and/or topology for watermark embedding, which is rarely tolerated by most CAD design and analysis systems. This motivated the development of the PO-watermarking scheme, because it does not change or modify the overall shape of an object.

Chapter 4 also highlighted that no research has been conducted on watermarking 3D B-rep CAD data. Therefore, in Chapter 5, a new non-blind watermarking framework was proposed that is suitable to implement parametric changes to B-rep faces to encode text information as the watermark, but without geometrical or topological changes to the CAD model. Within this approach the underlying parametric orientation of planar, closed conical, spherical, and toroidal B-rep faces were modified to accommodate the information, while the text message itself determined the degree of modification.

No common testing standards or test objects for the evaluation of CAD data watermarks could be identified from the literature. Therefore, in Chapter 6, a new watermark testing procedure based on real-world modifications was developed and applied to assess the PO-watermarking approach. The new evaluation process could be used to evaluate and compare future CAD data watermarking schemes. Prior studies all used different evaluation methods and models which made comparisons

between different approaches very difficult (or even impossible). The new evaluation approach defines a consistent testing process using a set number of CAD models of different complexity, which could be used in further studies to enable comparisons between different approaches.

In Chapter 7, the newly created test data, watermarked by the PO-watermarking scheme, was presented and the test results of the various modification attacks discussed. Test results show that the introduced PO-watermark does not change the model's shape or file size, nor has any influence on the CAD model's capabilities. Also, the more faces a CAD model has the more text information can be embedded. However, with increasing complexity the overall watermark embedding/recovery rate declines. After the exposure to intentional and unintentional attacks the PO-watermark survived five (Load, CAD, Zip, Healing, and Rotation) out of ten modifications without any compromise. In terms of geometrical modifications (Blecha, Cutting, and Adding), it seems that the more complex a CAD model gets the less influence they have on the watermark's survival rate. However, certain file conversion modifications, like IGES and STEP, can erase the PO-watermark.

Considering the overall performance, PO-watermarking has shown to be a viable method for marking 3D B-rep CAD models. It works well within a single CAD environment (i.e. company network) where no data is being translated, and is robust for everyday use. However, it has a limited functionality when watermarked data is moved between different CAD systems using neutral CAD formats.

8.2 Contribution to Knowledge

This study contributes to an area of research that received little attention in the past.. Despite many attempts to hide watermark information into three dimensional shapes, no viable scheme has been specifically designed for valuable mechanical 3D CAD models. To fill this lacuna, this study has introduced a PO-watermarking scheme for mechanical 3D B-rep CAD data that enables the embedding and retrieval of text messages while preserving both topology and the exact geometrical shape of a CAD model.

In comparison to existing 3D watermarking schemes where the embedding processes are based on some sort of perturbation or alteration to surface mesh vertices and/or control points to embed a random bit of data, the proposed PO-watermarking approach enables the embedding and retrieval of meaningful text messages by modifying the underlying parametric surface of a CAD model's faces. The PO-watermarking scheme is based on four (i.e. planar, closed conical, sphere and torus) out of the five most frequently used basic shapes in mechanical CAD models.

Particular advancements on previous studies are that any desired text message can be embedded and extracted (as requested by the CAD industry in the survey); the watermark has no influence on the CAD model's capabilities (e.g. in a simulation or assembly);, and the marked CAD file is fully functional in any CAD system. Furthermore, the watermark is not visible to the naked eye, has a robust connectivity to the carrier CAD model, and is difficult to remove. In other words, the encoded and embedded watermark can survive most data handling (unintentional) and modification (intentional) attacks that are present within the CAD environment.

This study also introduced appropriate test procedures and practices, and test objects that are more adequate for commercial mechanical CAD applications. Test procedures were specifically selected because of their frequent occurrence in day-to-day CAD engineering applications, as specified in the KIM survey. Within this study the new evaluation approach was also used to investigate the PO-watermark's robustness and capabilities. These new test procedures and objects can serve as a benchmark to compare other CAD watermarking systems.

Since PO-watermarking does not change or transform the overall shape of the CAD model during watermark encoding and embedding and resists (most) CAD related real-world modifications, this approach is much more suitable for a homogenous industrial CAD environment (i.e. one supplier, no translation) than any approach presented so far. In this respect, the PO-watermarking approach fills a particular need of the CAD industry as demonstrated in section 7.4.

8.3 Limitations and Directions for Future Work

The strength of any study arguably lies in the recognition of its limitations. Specifically, the investigation has several limitations which are indicative of future research. The first three limitations are programme-related, while the remaining limitations are related to CAD data modifications after watermarking.

- *Ordering of faces for watermark embedding and retrieval*

The current face ordering process for both watermark embedding and retrieval is based on the geometric modelling kernel of ACIS. However, if the CAD data is loaded into a different CAD system on a different computer the watermark retrieval programme might struggle to create the same face order, specifically when the new

CAD system uses a different kernel or the model has been modified. To make the ordering of faces independent from the modeller type a new sorting function could be introduced that is based on face characteristics like face type, area, number of edges, orientation, and number of inner loops.

- *Including of new face types as data carrier*

The PO-watermarking scheme currently supports only four face types: planar, closed conical, spherical, and toroidal. However, to increase the PO-watermarking effectiveness/success rate other face types need to be included to the embedding and retrieval processes. New face types could include open conical, ellipse, helix, or spline. To enable this extension, further research is required to establish whether their underlying parametric surfaces can be modified sufficiently to accommodate the watermark message (i.e. the tweaking parameter) without changing their overall shapes.

- *Rounding errors*

For the PO-watermarking scheme it is important that the tweaking coordinates/values in the retrieval process are measured and reconverted as accurately as possible to avoid incorrect conversions of ASCII-decimal values during text message reconvery. Especially for the suggested robustness-enhanced approach (section 7.6.2) precision is crucial since it aims to embed two text characters into a single tweaking parameter and face. Therefore, further investigations and testing are required to determine (i) if the six decimal places currently used are sufficient to carry the data load for two characters; and (ii) how rounding errors/issues could be minimised to increase precision to avoid data loss?

- *Potentially easily recognised watermark*

Some viewing tools like ACIS viewer possess the functionality to render parameter lines of the objects faces visible. This also discloses the rotation of the watermarked faces which might indicate a hidden mark.

- *Translation into neutral IGES and STEP file formats*

Another issue that requires further research is the translation of watermarked B-rep data into neutral file formats like IGES and STEP. As often stated translation of CAD data into other formats is the major weakness of the CAD business (Gould, 2010; Goldsberry, 2010; Theorem, 2010; IronCAD, 2002; Heinzen, 2009). The test results showed that when the PO-watermarked data is transformed into neutral formats, the watermark in certain face types (i.e. closed conical and planar) are partially erased.

- *Blending and chamfer modifications on watermarked faces*

Blending and chamfer modifications to watermarked faces seem to confuse the PO-watermark reader. Occasionally, it recorded more watermarked faces than actually exist, which can lead to confusion when re-assembling the text message. Therefore the behaviour pattern of Blecha modification requires further research and testing. It needs to be established which face types are affected by this event and how it could be prevented.

- *Effects of PO-watermark on downstream use*

At this stage it is unproven which effects the PO-watermark might have on downstream uses like Finite Element Analysis, Rapid Prototyping, and CNC tool path creation. Theoretically, the watermarked data should behave exactly like unmarked data since only the underlying parametric surface structure has been

tweaked but the overall shape and geometry of the model remain unchanged. Therefore, the watermarked data should be directly useable in downstream applications without any further modifications. A reorientation back to the original position should not be required. However, this can ultimately only be proven through extended field experiment since the commercial systems' internal algorithms are not publicly available.

References

- 1888 (2008). "Digital Watermarking And Fingerprinting Could Surpass \$500 Million Worldwide By 2012", <http://www.1888pressrelease.com/digital-watermarking-and-fingerprinting-could-surpass-500-m-pr-ye4h7r614.html>, Accessed on: 5 April 2008.
- Achilles, A. & Ortyl, P. (2011). "The Collection of Computer Science Bibliographies", <http://iinwww.ira.uka.de/bibliography/>, Accessed on: 12. March 2011.
- Acis (2009). "Online Help for ACIS", C:\Users\Csaba\Documents\PhD\ACIS\help\ACIS.htm, Accessed on: 1 June 2009.
- Alface, P. & Macq, B. (2005). "Blind watermarking of 3D meshes using robust feature points detection". In Proc. of *IEEE International Conference on Image Processing (ICIP '05)*. 1, pp. 693-696. Genoa, Italy, 11-14 Sept. 2005.
- Arnold, M. (2000). "Audio watermarking: Features, applications and algorithms". In Proc. of *IEEE International Conference on Multimedia and Expo (ICME 2000)*. 2, pp. 1013-1016. New York, NY, USA, 30 July- 2 August 2000.
- Arnold, M. (2001). "Audio watermarking". *Dr. Dobb's Journal*, 26 (11), pp. 21-26.
- Arnold, M., Schmucker, M. & Wolthusen, S. (2003). *Techniques and applications of digital watermarking and content protection*, Boston, Artech House.
- Arnoud, E. (2005). "Message digest / cryptographic hash functions", <http://www.iusmentis.com/technology/hashfunctions/>, Accessed on: 21 October 2008.
- Aspert, N., Drelie, E., Maret, Y. & Ebrahimi, T. (2002). "Steganography for three-dimensional polygonal meshes". In Proc. of *The International Society for Optical Engineering*. pp. 705-708. Seattle, USA, 7 July 2002.
- Autobild (2009). "Autobild Motor Magazine", <http://www.autobild.de/>, Accessed on: 09. July 2009.
- Barni, M., Bartolini, F., Cappellini, V. & Piva, A. (1998a). "A DCT-domain system for robust image watermarking". *Signal Processing (Special Issue on Watermarking)*, 66 (3), pp. 357-372.
- Barni, M., Bartolini, F., Cappellini, V., Piva, A. & Rigacci, F. (1998b). "A M.A.P. identification criterion for DCT-based watermarking". In Proc. of *European Signal Processing Conference (EUSIPCO '98)*. Rhodes, Greece, 8-11 September 1998.
- Bassia, P. & Pitas, I. (1998). "Robust audio watermarking in the time domain". In Proc. of *European Signal Processing Conference (EUSIPCO '98)*. pp. 25-28. Rhodes, Greece, 8-11 September 1998.
- Bassia, P. & Pitas, I. (2001). "Robust audio watermarking in time domain". *IEEE Transactions on Multimedia*, 3 (2), pp. 232-241.
- Bauchle, R., Hazen, F., Lund, J., Oakley, G. & Rundatz, F. (2008). "Information security magazin", http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci212062,00.html, Accessed on: 17 March 2009.
- Bender, W., Gruhl, D., Morimoto, N. & Lu, A. (1996). "Techniques for data hiding". *IBM System Journal*, 35 (3), pp. 313-336.

- Benedens, O. (1999a). "Two high capacity methods for embedding public watermarks into 3D polygonal models". In Proc. of *International Conference on Multimedia and Security*. pp. 95 - 99. Orlando, Florida, U.S.A.
- Benedens, O. (1999b). "Watermarking of 3D-polygon-based models with robustness against mesh simplification". In Proc. of *Security and Watermarking of Multimedia Contents*. 3657, pp. 329-340. San Jose, CA, USA, 28 Jan. 1999.
- Benedens, O. (1999c). "Geometry-based watermarking of 3D models". *IEEE Computer Graphics and Applications*, (January/February 1999), pp. 46-55.
- Benedens, O. (2000a). "Affine invariant watermarks for 3D polygonal and NURBS based models". In Proc. of *International Workshop Information Security (ISW '2000)*. 1975, pp. 15-29. Wollongong, Australia, December 2000.
- Benedens, O. (2002). "3D Watermarking-Algorithms in Context of OpenGL Plus". *Fraunhofer-Institut für Graphische Datenverarbeitung, Darmstadt, Germany, 29 January 2002*,
- Benedens, O. & Bush, C. (2000b). "Towards blind detection of robust watermarks in polygonal models". *EUROGRAPHICS 2000*, 19 (3), pp. 199-208.
- Benham, D., Memon, N., Yeo, B. & Yeung, M. (1997). "Fast watermarking of DTC-based compressed images". In Proc. of *International Conference on Image Science, Systems and Technology (CISST '97)*. pp. 243-253. Las Vegas, NV, USA, June 1997.
- Bennour, J. & Dugelay, J. (2006). "Protection of 3D object through silhouette watermarking". In Proc. of *IEEE International Conference in Acoustics, Speech and Signal Processing (ICASSP '06)*. 2, pp. 1113-1116. Toulouse, France, 14-19 May 2006.
- Bentley, C. (2003). "Rendering cubic Bézier patches." http://www.cs.wpi.edu/~matt/courses/cs563/talks/surface/bez_surf.html, Accessed on: 12 January 2010.
- Berghel, H. & O'gorman, L. (1997). "Digital watermarking", http://berghel.net/publications/dig_wtr/dig_watr.php, Accessed on: 13 March 2009.
- Boney, L., Tewfik, A. & Hamdy, K. (1996). "Digital watermarks for audio signal". In Proc. of *IEEE International Conference on Multimedia Computing and Systems '96*. pp. 473-480. Hiroshima, Japan, 17-23 June 1996.
- Bors, A. (2004). "Watermarking 3D shapes using local moments". In Proc. of *International Conference on Image Processing. (ICIP '04)*. 2, pp. 729-732. Singapore, 24-27 Oct. 2004.
- Bors, A. (2006). "Watermarking mesh-based representations of 3-D objects using local moments". *IEEE Transactions on Image Processing*, 15 (3), pp. 687-701.
- Brassil, J., Low, S., Maxemchuk, N. & O'gorman, L. (1995a). "Electronic marking and identification techniques to discourage document copying". *IEEE Journal on Selected Areas in Communications*, 13, pp. 1495-1504.
- Brassil, J., Low, S., Maxemchuk, N. & O'gorman, L. (1995b). "Hiding information in document images". In Proc. of *Paper presented at: 29th Annual Conference on Information Science and Systems (CISS 95)*. pp. 482-489. Johns Hopkins University, Baltimore, MD, USA.

- Busch, C., Funk, W. & Wolthusen, S. (1999). "Digital watermarking: From concepts to real-time video applications". *IEEE Computer Graphics Applications*, 19 (1), pp. 25-35.
- Bush, C., Funk, W. & Wolthusen, S. (1999). "Digital watermarking: From concepts to real-time video applications". *IEEE Computer Graphics Applications*, pp. 25-35.
- Caduser (2000). "Healing the wounds of data conversion", http://www.caduser.com/reviews/reviews.asp?a_id=66, Accessed on: 12 August 2009.
- Ceusters, W. & Smith, B. (2006). "Referent tracking for digital rights management". *International Journal of Metadata, Semantics and Ontologies*, June 20.
- Chadwick, J., Bangay, S. & Wentworth, P. (2003). "A digital watermarking scheme for Bezier surfaces". In Proc. of *6th South African Telecommunications, Networks and Applications Conference (SATNAC '03)*. pp. 44-47. Southern Cape, South Africa, 7-10 September 2003.
- Chandru, V., Manivannan, M. & Manohar, S. (1999). "Minkowski operators and features of voxel models". In Proc. of *ASME Design Engineering Technical Conference*. Las Vegas, Nevada, USA, September 12-15.
- Chang, H., Chen, T. & Kan, K. (2003). "Watermarking 2D/3D graphics for copyright protection". In Proc. of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*. 4, pp. 720-723. Hong Kong, 6-10 April 2003.
- Chassery, J., Bas, P. & Davoine, F. (1998). "Self-similarity based image watermarking". In Proc. of *European Signal Processing Conference (EUSIPCO '98)*. pp. 99-102. Rhodes, Greece, 8-11 September 1998.
- Chen, B. & Wornell, G. (1998). "Digital watermarking and information embedding using dither modulation". In Proc. of *IEEE Workshop on Multimedia Signal Processing*. pp. 273-278. Los Angeles, California, USA, 7-9 December 1998.
- Chen, B. & Wornell, G. (1999). "Dither modulation: A new approach to digital watermarking and information embedding". In Proc. of *IS&T/SPIE's 11th Annual Symposium on Electronic Imaging '99: Security and Watermarking of Multimedia Contents*. 3657, pp. 342-353. San Jose, California, USA, 25-27 January 1999.
- Chou, C. & Tseng, D. (2006). "A public fragile watermarking scheme for 3D model authentication". *Computer -Aided Design*, 38, pp. 1154-1165.
- Collis, J. & Hussey, R. (2003). *Business Research - A practical guide for undergraduate and postgraduate students*, Houndmills, Palgrave Macmillan.
- Constructive Solid Geometry (2008). "Constructive Solid Geometry", <http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/csg.html>, Accessed on: 10 July 2008.
- Corney, J. (1997). *3D Modeling with the ACIS kernel and toolkit*, Chichester, John Wiley & Sons.
- Corney, J. & Lim, T. (2001). *3D Modeling with ACIS*, Stirling, Saxe-Coburg Publications.
- Corsini, M., Barni, M., Bartolini, F., Caldelli, R., Cappellini, V. & Piva, A. (2003a). "Towards 3D watermarking technology". In Proc. of *IEEE EUROCON 2003*. 2, pp. 393-396. Ljubljana, Slovenia, 22-24 September 2003.

- Corsini, M., Uccheddu, F., Bartolini, F., Barni, M., Caldelli, R. & Cappellini, V. (2003b). "3D watermarking technology: Visual quality aspects". In Proc. of *9th Conference on Virtual System and Multimedia (VSMM '03)*. Montreal, Quebec, Canada, 15-17 October 2003.
- Cotting, D., Weyrich, T., Pauly, M. & Gross, M. (2004). "Robust Watermarking of Point-Sampled Geometry". In Proc. of *International Conference on Shape Modeling and Applications (SMI '04)*. pp. 233-242. Palazzo Ducale, Genova, Italy, 7-9 June 2004.
- Cox, I., Kilian, J., Leighton, F. & Shamoon, T. (1997). "Secure spread spectrum watermarking for multimedia". *IEEE Transactions on Image Processing*, 6 (12), pp. 1673-1687.
- Cox, I., Kilian, J., Leighton, T. & Shamoon, T. (1995). "Secure spread spectrum watermarking for images, audio, and video". Report No. 95-10, *NEC Research Institute*,
- Cox, I., Kilian, J., Leighton, T. & Shamoon, T. (1996). "Secure spread spectrum watermarking for images, audio and video". In Proc. of *IEEE International Conference on Image Processing (ICIP '96)*. 3, pp. 243-246. Lausanne, Switzerland, 16-19 September 1996.
- Cox, I. & Miller, M. (2002a). "The first 50 years of electronic watermarking". *Applied Signal Processing*, 2, pp. 126-132.
- Cox, I., Miller, M. & Bloom, J. (2002b). *Digital watermarking*, San Francisco, Morgan Kaufmann Publishers.
- Coyle, K. (2003). "The technology of rights: Digital rights management", http://www.kcoyle.net/drm_basics.pdf, Accessed on: 27 October 2009.
- Daras, P., Zarpalas, D., Tzovaras, D. & Strintzis, M. (2004). "Watermarking of 3D models for data hiding". In Proc. of *International Conference on Image Processing ICIP '04*. 1, pp. 47-50. Singapore, 24-27 October 2004.
- Darmstaedter, V., Delaigle, J., Nicholson, D. & Macq, B. (1998). "A block based watermarking technique for MPEG-2 signals: Optimization and validation on real digital TV distribution links". In Proc. of *European Conference on Multimedia Applications, Services, and Techniques (ECMAST '98)*. pp. 190-206. Berlin, Germany, 26-28 May 1998.
- Davern, P. & Scott, M. (1996). "Fractal based image steganography". *Lecture Notes in Computer Science: Information Hiding*, Springer, 1174, pp. 279-294.
- Davies, B.L., Robotham, A.J. & Yarwood, A. (1991). *Computer-aided drawing and design*, London, Chapman & Hall.
- Deguillaume, F., Csurka, G., Ruanaidh, J. & Pun, T. (1999). "Robust 3D DFT video watermarking". In Proc. of *IS&T/SPIE's 11th Annual Symposium on Electronic Imaging '99: Security and Watermarking of Multimedia Contents*,. 3657, San Jose, California, USA, 25-27 January 1999.
- Denis, F., Lavoué, G., Dupont, F. & Baskurt, A. (2005). "Digital watermarking of compressed 3D meshes". In Proc. of *International Conference on Machine Intelligence*. pp. 648-654. Tozeur, Tunisia, 5-7 November 2005.
- Depovere, G., Kalker, T. & Linnartz, J. (1998). "Improved watermark detection reliability using filtering before correlation". In Proc. of *IEEE International Conference on Image Processing (ICIP '98)*. pp. 430-434. Chicago, IL, USA, 4-7 October 1998.

- Dieter, G. (1983). *Engineering design: A materials and processing approach*, New York, McGraw-Hill.
- Dittmann, J., Stabenau, M. & Steinmetz, R. (1998b). "Robust MPEG video watermarking technologies". In Proc. of *ACM Multimedia '98*. pp. 71-80. Bristol, UK, 12-16 September 1998.
- Djaghloul, H. (2007). "Wavelet-based medical augmented reality CSG object watermarking". *International Journal of Soft Computing*, 2 (1), pp. 212-215.
- Doërr, G. & Dugelay, J.-L. (2003). "A guide tour of video watermarking". *Signal Processing: Image Communication, Special Issue on Technologies for Image Security*, 18 (4), pp. 263-282.
- Drake, M.A. (2003). *Encyclopedia of library and information science*, Second Edition New York, Marcel Dekker Inc.
- Duffy, A.H.B. & O'donnell, F.J. (1998). "A Design Research Approach". In Proc. of *Workshop on Research Methods in Artificial Intelligence in Design (AID)*. Lisbon, Portugal.
- Dugelay, J., Garcia, E. & Mallauran, C. (2002). "Protection of 3-D Object Usage Through Texture Watermarking". In Proc. of *11th European Signal Processing Conference (EUSIPCO 2002)*. Toulouse, France, 3 - 6 September 2002.
- Easterby-Smith, M. & Thorpe, R. (1991). *Management Research: An Introduction*, London, Sage Publications.
- Felten, E. (2003). "A skeptical view of DRM and fair use". *Communications of the ACM*, 46 (4), pp. 57-59.
- Fornaro, C. & Sanna, A. (2000). "Public key watermarking for authentication of CSG models". *Computer-Aided Design*, 32 (12), pp. 727-735.
- Funk, W. (2004). "Impact of MPEG-4 3D mesh coding on watermarking algorithms for polygonal 3D meshes". *Security, Steganography, and Watermarking of Multimedia Contents*, 5306, pp. 336-344.
- Furon, T., Moreau, N. & Duhamel, P. (2000). "Audio public key watermarking technique". In Proc. of *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '00)*. 6, pp. 1959-1962. Istanbul, Turkey, 5-9 June 2000.
- Garcia, E. & Dugelay, J. (2003). "Texture-based watermarking of 3D video objects". *IEEE Transactions on Circuits and Systems for Video Technology*, 13 (8), pp. 853-866.
- Garcia, R. (1999). "Digital watermarking of audio signals using a psychoacoustic model and spread spectrum theory". In Proc. of *107th Audio Engineering Society Convention*. New York, NY, USA, September 1999.
- Gerbino, S. (2003). "Tools for the interoperability among CAD systems". In Proc. of *International Conference on Tools and Methods Evolution in Engineering Design*. Napoli and Salerno, Italy, 3-6 June 2003.
- Gill, J. & Johnson, P. (2002). *Research Methods for Managers*, London, SAGE Publications.
- Godwin, M. (2006). "Digital rights management: A guide for librarians", <http://nmlm.gov/scr/blog/?p=241>, Accessed on: 07 July 2008.
- Goldsberry, C. (2010). "Data file translation woes: A failure to communicate", <http://www.plasticstoday.com/imm/articles/moldmaking-data-file-transfer>, Accessed on: 20 September 2010.

- Gomes, L., Cano, P., Gómez, E., Bonnet, M. & Batlle, E. (2003). "Audio watermarking and fingerprinting: For which applications?" *Journal of New Music Research*, 32 (1), pp. 65-81.
- Gomes, L., Mboup, M., Bonnet, M. & Moreau, N. (2000). "Cyclostationarity-based audio watermarking with private and public hidden data". In Proc. of *109th AES Convention of Audio Engineering Society*. Los Angeles, California, USA, 22-25 September 2000.
- Gould, L. (2010). "Multi-CAD Environments", <http://www.autofieldguide.com/articles/090810.html>, Accessed on: 20 September 2010.
- Harte, T. & Bors, A. (2002a). "Watermarking 3D Models". In Proc. of *International Conference on Image Processing, 24-28 June 2002*. 3, pp. 661-664. Rochester, New York, USA.
- Harte, T. & Bors, A. (2002b). "Watermarking Graphical Objects". In Proc. of *14th International Conference on Digital Signal Processing*. 2, pp. 709-712. Aegean Island of Santorini (Thera) in Hellas, Greece, 1-3 July 2002.
- Hartung, F., Eisert, P. & Girod, B. (1998). "Digital watermarking of MPEG-4 facial animation parameters". *Computers & Graphics*, 22 (3), pp. 1-17.
- Hartung, F. & Girod, B. (1996). "Digital watermarking of raw and compressed video". *SPIE Digital Compression Technologies and Systems for Video Communication*, 2952, pp. 205-213.
- Hartung, F. & Girod, B. (1997). "Fast public-key watermarking of compressed video". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 1, pp. 528-531. Santa Barbara, California, USA, 26-29 October 1997.
- Hartung, F. & Kutter, M. (1999a). "Multimedia Watermarking Techniques". *Proceedings of the IEEE*, 87 (7), pp. 1079-1107.
- Hayden, D.S. (2004). *7 Easy Steps to CNC Programming Book II: Beyond the beginning*, Haydenpub Publishing.
- Heinzen, M. (2009). "3-D CAD Translation", <http://www.qcinspect.com/article/cadtranslation.htm>, Accessed on: 20 September 2010.
- Hsu, C. & Wu, J. (1997a). "Digital watermarking for images and videos". In Proc. of *Digital Signal Processing (DSP '97)*. Santorini, Greece, July 1997.
- Hu, Y., Lee, H.K. & Zeng, H. (2008). "Curve Watermarking Technique for Fingerprinting Digital Maps". In Proc. of *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP '08)*. pp. 223-226. Harbin, China, 15-17 August 2008.
- Hunter, D. (1978). *Papermaking, The History and Technique of an Ancient Craft*, Second, revised and enlarged edition, Dover Publications.
- Ironcad (2002). "IronCAD Data Translation Overview", <http://www.ironcad.com/download/ironcad/tutorials/IronCADDDataTranslation.pdf>, Accessed on: 20 September 2010.
- Jackson, D. & Mabbott, M. (1999). "Data exchange between incompatible modellers".
- Johnson, A. & Biggar, M. (1997). "Digital watermarking of video/image content for copyright protection and monitoring". ISO Doc. ISO/IEC JTC1/SC29/WG11 MPEG97/M2229,

- Jordan, F., Kutter, M. & Ebrahimi, T. (1997). "Proposal of a watermarking technique for hiding/retrieving data in compressed and decompressed video". ISO/IEC Doc. JTC1/SC29/WG11 MPEG97/M2281,
- Kahn, D. (1996). *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet, Revised and Updated Edition*, New York, Scribner.
- Kalay, Y.E. (1989). *Modelling Objects and Environments*, New York, John Wiley & Sons.
- Kalker, T., Depovere, G., Haitzma, J. & Maes, M. (1999). "A video watermarking system for broadcast monitoring". In Proc. of *SPIE IS&T/SPIE's 11th Annual Symposium on Electronic Imaging '99: Security and Watermarking of Multimedia Contents*. 3657, pp. 103-112. San Jose, California, USA, Jan. 1999.
- Kanai, S., Date, H. & Kishinami, T. (1998). "Digital Watermarking for 3D Polygons using Multiresolution Wavelet Decomposition". In Proc. of *International Workshop on Geometric Modeling: Fundamentals and Applications (GEO-6)*, 7-9 December. pp. 296 - 307. Tokyo, Japan.
- Kankanhalli, S., Rajmohan, M. & Ramakrishnan, K. (1998). "Content-based watermarking of images". In Proc. of *ACM Multimedia '98*. Bristol, UK, 12-16 September 1998.
- Katzenbeisser, S. & Petitcolas, F.A.P. (2000). *Information Hiding Techniques for Steganography and Digital Watermarking*, Boston, Artech House.
- Kessler, G.C. (1998). *Handbook on Local Area Networks*, Auerbach.
- Kim, H., Lee, B. & Lee, N. (2002). "Wavelet-based audio watermarking techniques: Robustness and fast synchronisation", <http://amath.kaist.ac.kr/research/paper/01-11.pdf>, Accessed on: 9 February 2009.
- Kim, S., Suthaharan, S., Lee, H. & Rao, K. (1999). "Perceptually tuned robust watermarking scheme for video using motion entropy masking". In Proc. of *International Conference on Consumer Electronics (ICCE '99)* pp. 104-105. Los Angeles, California, USA, 22-24 June 1999.
- Koch, E. & Zhao, J. (1995). "Towards robust and hidden image copyright labeling". In Proc. of *Workshop on Nonlinear Signal and Image Processing*. Marmaros, Greece, June 1995.
- Koller, D. & Levoy, M. (2005). "Protecting 3D Graphics Content". *Communications of the ACM*, 48 (6), pp. 74-80.
- Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P. & Scopigno, R. (2004). "Protected Interactive 3D Graphics Via Remote Rendering". In Proc. of *ACM Transactions on Graphics (TOG)*. 23, pp. 695-703.
- Kundur, D. & Hatzinakos, D. (1997). "A robust digital image watermarking method using wavelet-based fusion". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 1, pp. 544-547. Santa Barbara, California, USA, 26-29 October 1997.
- Kundur, D. & Hatzinakos, D. (1998). "Digital watermarking using multiresolution wavelet decomposition". In Proc. of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSAP '98)*. 5, pp. 2969-2972. Seattle, Washington, USA, May 1998.

- Kurowski, P.K. (2004). *Finite Element Analysis for Design Engineers*, Warrendale, PA, USA, SAE Permissions.
- Kutter, M. (1998a). "Watermarking resisting to translation, rotation, and scaling". *SPIE International Symposium on Voice, Video, and Data Communication*.
- Kutter, M., Jordan, F. & Bossen, F. (1997). "Digital signature of color images using amplitude modulation". In Proc. of *Electronic Imaging (EI '97)*. San Jose, California, USA, 9-14 February 1997.
- Kutter, M., Jordan, F. & Bossen, F. (1998b). "Digital signature of color images using amplitude modulation". *Electronic Imaging (EI '98)*, 7 (2), pp. 326-332.
- Lacy, J., Quackenbush, S., Reibman, A., Shur, D. & Snyder, J. (1998). "On combining watermarking with perceptual coding". In Proc. of *IEEE International Conference on Acoustic Speech and Signal Processing*. Seattle, Washington, USA, 12-15 May 1998.
- Langelaar, G. (1996). "Copy protection for multimedia data based on labeling techniques". In Proc. of *7th Symposium on Information Theory in the Benelux*. Enschede, The Netherlands, May 1996.
- Langelaar, G. & Lagendijk, R. (2001). "Optimal differential energy watermarking of DCT encoded images and video". *IEEE Transactions on Image Processing*, 10 (1), pp. 148-158.
- Langelaar, G., Lagendijk, R. & Biemond, J. (1997b). "Real-time labeling methods for MPEG compressed video". In Proc. of *18th Symposium on Information Theory in the Benelux*. Veldhoven, The Netherlands, May 1997.
- Langelaar, G., Lagendijk, R. & Biemond, J. (1998). "Real-time labelling of MPEG-2 compressed video". *Journal of Visual Communication and Image Representation*, 9 (4), pp. 256-270.
- Langelaar, G., Lubbe, J. & Lagendijk, R. (1997a). "Robust labeling methods for copy protection of images". In Proc. of *Electronic Imaging*. 3022, pp. 298-309. San Jose, CA, USA, Feb. 1997.
- Lee, J., Cho, N. & Lee, S. (2004). "Watermarking Algorithms for 3D NURBS Graphic Data". *EURASIP Journal on Applied Signal Processing* 14, pp. 2142-2152.
- Li, L., Pan, Z., Zhang, M. & Ye, K. (2004). "Watermarking Subdivision Surfaces Based on Addition Property of Fourier Transform". In Proc. of *2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* pp. 46-49. Singapore, 15-18 June 2004.
- Light, J., Lindsay, P., L., S. & Parnes, P. (1990). "The effects of message encoding techniques on recall by literate adults using AAC systems". *Augmentative and Alternative Communication*, 6 (3), pp. 184-201.
- Lin, H. & Liao, H. (2005). "Fragile watermarking for authenticating 3D polygonal meshes". *IEEE Transactions on Multimedia*, 7 (6), pp. 997-1006.
- Linnartz, J. (1998). "MPEG PTY marking". [Online] Available: <http://diva.eecs.berkeley.edu/linnartz/pty.html>.
- Low, S. & Maxemchuk, N. (1998). "Performance Comparison of Two Text Marking Methods". *IEEE Journal on Selected Areas in Communications*, 16 (4), pp. 561-572.
- Low, S., Maxemchuk, N., Brassil, J. & O'gorman, L. (1995). "Document Marking and Identification Using Both Line and Word Shifting". In Proc. of *Infocom '95*. pp. 853-860. Boston, MA.

- Maret, Y. & Ebrahimi, T. (2004). "Data Hiding on 3D Polygonal Meshes". In Proc. of *ACM Multimedia & Security Workshop*. pp. 68-74. Magdeburg, Germany, 20-21 September 2004.
- Matheson, L.R., Mitchell, S.G., Shamoan, T.G., Tarjan, R.E. & Zane, F. (1998). "Robustness and Security of Digital Watermarks". In Proc. of *Second International Conference on Financial Cryptography*. pp. 227 - 240. Berlin, Germany.
- Mcmahon, C. & Browne, J. (1998). *CADCAM Principles, Practice and Manufacturing Management, 2nd Edition*, Harlow, England, Addison-Wesley.
- Menezes, A., Van Oorschot, P. & Vanstone, S. (2001). *Handbook of Applied Cryptography*, CRC Press, Fifth Printing (August 2001), ISBN: 0-8493-8523-7.
- Merkle, R. & Brassard, I.G. (1990). "A certified digital signature". *Advances in Cryptology -- CRYPTO '89*, vol. 435, pp. 218-238.
- Mills, B., Langbein, F., Marshall, A. & Martin, R. (2001). "Estimate of frequencies of geometric regularities for use in reverse engineering of simple mechanical components". *Geometry and Vision Group, Cardiff University*,
- Mitchell, W.J. & Mccullough, M. (1995). *Digital design Media*, New York, Van Nostrand Reinhold.
- Mottram, J.T. & Shawn, C.T. (1996). *Using Finite Elements in Mechanical Design*, London, McGraw-Hill Book Company.
- Multimedia Intelligence (2008) MultiMedia Intelligence Identifies Digital Watermarking & Fingerprinting As Key New Opportunity.
- Murdoch, S.J. (2006). "Oracle attack on Wordpress", <http://www.lightbluetouchpaper.org/2006/06/22/oracle-attack-on-wordpress/>, Accessed on: 21 July 2009.
- Murotani, K. & Sugihara, K. (2005). "New spectral decomposition for 3D polygonal meshes and its application to watermarking". In Proc. of *Computer Graphics, Visualization and Computer Vision*. University of West Bohemia, Plzen - Bory, Czech Republic 31 January - 4 February 2005.
- Nagahashi, H., Mitsuhashi, R. & Morooka, K. (2004). "A Method for Watermarking to Bézier Polynomial Surface Models". *IEICE Trans. Inf. & Syst.*, E87-D (1), pp. 224-232.
- Neubauer, C. & Herre, J. (1998). "Digital watermarking and its influence on audio quality". In Proc. of *105th AES Convention, Audio Engineering Society*. Preprint 4823, San Francisco, CA, USA, Sept. 1998.
- Neubauer, C. & Herre, J. (2000a). "Audio watermarking of MPEG-2 AAC bit streams". In Proc. of *108th AES Convention, Audio Engineering Society*. Preprint 5101, Paris, France, 19-22 Feb. 2000.
- Neubauer, C. & Herre, J. (2000b). "Advanced audio watermarking and applications". In Proc. of *109th AES Convention, Audio Engineering Society*. Preprint 5176, Los Angeles, CA, USA, Sept. 2000.
- Nikolaidis, N. & Pitas, I. (1996). "Copyright protection of images using robust digital signatures". In Proc. of *ICASSP '96*. Atlanta, GA, USA, May 1996.
- Ohbuchi, R. & Masuda, H. (2000). "Managing CAD data as a Multimedia Data Type Using Digital Watermarking". In Proc. of *IFIP TC5 WG5.2 Workshop on*

- Knowledge Intensive CAD to Knowledge Intensive Engineering*. 207, pp. 103-116. Deventer, The Netherlands, 22 - 24 May 2000.
- Ohbuchi, R., Masuda, H. & Aono, M. (1997a). "Embedding Data in 3D Models". In Proc. of *4th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services '97*. 1309, pp. 1-11. Darmstadt, Germany, September 1997.
- Ohbuchi, R., Masuda, H. & Aono, M. (1997b). "Watermarking Three-Dimensional Polygonal Models". In Proc. of *ACM International Multimedia Conference '97*. pp. 261-272. Seattle, Washington, USA, 9-13 November 1997.
- Ohbuchi, R., Masuda, H. & Aono, M. (1998a). "Data Embedding Algorithms for Geometrical and Non-Geometrical Targets in Three-Dimensional Polygonal Models". *Computer Communications, Elsevier Science B. V.*
- Ohbuchi, R., Masuda, H. & Aono, M. (1998b). "Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications". *IEEE Journal on Selected Areas in Communications*, 16 (4), pp. 551-560.
- Ohbuchi, R., Masuda, H. & Aono, M. (1998c). "Watermarking Multiple Object Types in Three-Dimensional Models". In Proc. of *Multimedia and Security Workshop at ACM Multimedia '98*. Bristol, UK, 12-13 September 1998.
- Ohbuchi, R., Masuda, H. & Aono, M. (1999). "A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces". In Proc. of *Computer Graphics International 1999 (CGI '99)*. pp. 1 - 8. Canmore, Canada, 7-11 June 1999.
- Ohbuchi, R., Mukaiyama, A. & Takahashi, S. (2002). "A Frequency-Domain Approach to Watermarking 3D Shapes". *EUROGRAPHICS 2002*, 21 (3), pp. 373-382.
- Ohbuchi, R., Takahashi, S., Miyazawa, T. & Mukaiyama, A. (2001). "Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain". In Proc. of *Graphics Interface*. pp. 9-17. Ottawa, Ontario, Canada 7-9 June 2001.
- Petitcolas, F.A.P., Anderson, F.R. & Kuhn, M.G. (1999). "Information Hiding - A Survey". *IEEE*, 87 (7), pp. 1062-1078.
- Pfitzmann (1996). "Information Hiding Technology". in *Anderson R. (Ed) Lecturer Notes in Computer Science*, 1174, pp. 347-350.
- Piva, A., Barni, M., Bartoloni, E. & Cappellini, V. (1997). "DCT-based watermarking recovering without resorting to the uncorrupted original image". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 1, Santa Barbara, CA, USA.
- Podilchuk, C. & Delp, E. (2001). "Digital Watermarking: Algorithms and Applications". *Signal Processing Magazine*, 18 (4), pp. 33-46.
- Podilchuk, C. & Zeng, W. (1997). "Watermarking of the JPEG bitstream". In Proc. of *International Conference on Imaging Science, Systems, and Applications (CISST '97)*. pp. 253-260. Las Vegas, NV, USA, June 1997.
- Point Cloud (2009). http://en.wikipedia.org/wiki/Point_cloud, Accessed on:
- Polygonal Mesh (2008). http://en.wikipedia.org/wiki/Polygon_mesh, Accessed on: 10 July 2008.
- Praun, E., Hoppe, H. & Finkelstein, A. (1999). "Robust Mesh Watermarking". In Proc. of *International Conference on Computer Graphics and Interactive Techniques, August*. pp. 49-56.
- Ptc (2008). "MCAD Digital Rights Management, Protect Your Critical Design Intellectual Property (IP) Persistently and Dynamically",

- http://www.4point.com/pdf/3055_DRM_WP_FINAL.pdf, Accessed on: 10 February 2010.
- Puate, J. & Jordan, F. (1996). "Using fractal compression scheme to embed a digital signature into an image". In Proc. of *SPIE Photonics East '96 Symposium*. Boston, MA, USA, Nov. 1996.
- Qiao, L. & Nahrstedt, K. (1999). "Non-invertible watermarking methods for MPEG encoded audio". In Proc. of *Security and Watermarking of Multimedia Contents*. 3657, pp. 194-202. San Jose, CA, USA, 25-27 January 1999.
- Quan, L. & Xiaomei, Z. (2006). "SVD based digital watermarking algorithm for 3D models". In Proc. of *IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP '06)*. Toulouse, France, 15-19 May 2006.
- Ramkumar, M. & Akansu, A. (1999). "On the choice of transforms for data hiding in compressed video". In Proc. of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '99)*. Phoenix, AZ, USA, March 1999.
- Ranger, D.T. (2009) CAD/CAM Data Exchange: The transfer of data between dissimilar engineering computer systems. Theorem Solutions, Personal correspondence.
- Roche, S. & Dugelay, J.L. (1998). "Image Watermarking Based on the Fractal Transform". In Proc. of *IEEE Multimedia Signal Processing*. pp. 358-363. Los Angeles, California, USA, 7-9 December 1998.
- Rodriguez, W. (1992). *The Modelling of Design Ideas, Graphics and Visualization Techniques for Engineers*, New York, McGraw-Hill Book Company.
- Rooney, J. & Steadman, P. (1987). *Principles of Computer-aided Design*, London, Pitman Publishing.
- Rossignac, J.R. & Requicha, A.A.G. (1999) Solid Modeling. in J. G. Webster, Ed., *Encyclopedia of Electrical and Electronics Engineering*. New York, NY, Wiley.
- Ruanaidh, J., Dowling, W. & Boland, F. (1996). "Phase watermarking of digital images". In Proc. of *International Conference on Image Processing (ICIP '96)*. 3, pp. 239-242. Sept. 1996.
- Ruanaidh, J. & Pun, T. (1997). "Rotation, scale and translation invariant digital image watermarking". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 1, pp. 536-539. Santa Barbara, CA, USA, Oct. 1997.
- Ruanaidh, J. & Pun, T. (1998). "Rotation, scale, and translation invariant spread spectrum digital image watermarking". *Signal Processing (Special Issue on Watermarking)*, 66 (3), pp. 303-318.
- Rušinović, Z. & Mihajlović, Ž. (2005). "Robust Watermarking for 3D Objects". In Proc. of *MIPRO 2005: Hypermedia and GRID systems*. pp. 271-276. Opatija, Croatia, 30 May - 3 June 2005.
- Salzburg, U.O. (2009). "Still Image Watermarking", <http://www.cosy.sbg.ac.at/>, Accessed on: 27. September 2009.
- Schyndel, R.V., Tirkel, A. & Osborne, C. (1994). "A digital watermark". In Proc. of *International Conference on Image Processing (ICIP '94)*. 2, pp. 86-89.
- Simpson, J. & Weiner, E. (2000). *Oxford English Dictionary*, New York, Oxford University Press.
- Slowinski, F. (2003). "What Consumers Want in Digital Rights Management (DRM): Making Content as Widely Available as Possible In Ways that

- Satisfy Consumer Preferences". *AAP and ALA Release White Paper*, 21 March.
- Smith, S. (2004). "An evaluation of internet-based CAD collaboration tools", <http://scholar.lib.vt.edu/ejournals/JOTS/v30/v30n2/pdf/smith.pdf>, Accessed on: 2 August 2008.
- Sonnet, H. & Lange, S. (2005). "Data Storage: Carrier Objects as Illustration Watermarks for 3D Polygonal Models". In Proc. of *Simulation und Visualisierung*. pp. 305-316.
- Stroud, I. (1992). "Definition of solid modelling operations using uniform set of elementary procedures". *GML 1992/3 Computer and Automation Institute, Hungarian Academy of Sciences*, pp. 89.
- Swanson, M., Zhu, B. & Tewfik, A. (1996). "Transparent robust image watermarking". In Proc. of *International Conference on Image Processing (ICIP '96)*. Lausanne, Switzerland, Sept. 1996.
- Swanson, M., Zhu, B. & Tewfik, A. (1997). "Multiresolution video watermarking using perceptual models and scene segmentation". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 2, pp. 558-561. Santa Barbara, CA, USA, Oct. 1997.
- Swanson, M., Zhu, B. & Tewfik, A. (1998a). "Multiresolution scene-based video watermarking using perceptual models". *IEEE Journal on Selected Areas in Communications (Special issue on Copyright and Privacy Protection)*, 16 (4), pp. 540-550.
- Swanson, M., Zhu, B., Tewfik, A. & Boney, L. (1998b). "Robust audio watermarking using perceptual masking". *Elsevier Signal Processing*, 66 (3), pp. 337-355.
- Theorem (2010). "The Wide Angle on CAD Data Exchange", <http://www.theorem.co.uk/pr/PA4.pdf>, Accessed on: 20 September 2010.
- Thiemert, S., Steinebach, M. & Wolf, P. (2006). "A Digital Watermark for Vector-Based Fonts". In Proc. of *Multimedia and Security Workshop*. pp. 120-123. Geneva, Switzerland, 26-27 September 2006.
- Tilki, J. & Beex, A. (1996). "Encoding a hidden digital signature onto an audio signal using psychoacoustic masking". In Proc. of *7th International Conference on Digital Signal Processing Applications & Technology*. pp. 476-480. Boston, MA, USA, Oct. 1996.
- Tirkel, A., Rankin, G., Schyndel, R.V., Ho, W., Mee, N. & Osborne, C. (1993). "Electronic water mark". In Proc. of *Digital Image Computing, Techniques and Applications '93*. pp. 666-672. Macquarie University, Sydney, Australia, 8-10 December 1993.
- Tizzard, A. (1995). *An Introduction to Computer-Aided Engineering*, London, McGraw-Hill Book Company.
- Uccheddu, F., Corsini, M. & Barni, M. (2004). "Wavelet-Based Blind Watermarking of 3D Models". In Proc. of *International Multimedia Conference*. pp. 143-154. Magdeburg, Germany, 20-21 September 2004.
- Walker, K. (2004). "Audio watermarking". R&D White Paper, WHP 057, *Research & Development British Broadcasting Corporation*, <http://www.bbc.co.uk/rd/pubs/whp/whp057.shtml>.
- Wang, H. & Kuo, C. (1998). "An integrated progressive image coding and watermark system". In Proc. of *IEEE International Conference on Acoustics*,

- Speech, and Signal Processing (ICASSP '98)*. 6, pp. 3721-3723. Seattle, WA, USA, May 1998.
- Wang, X., Cui, Y., Yang, H. & Zhao, H. (2004). "A New Content-Based Digital Audio Watermarking Algorithm for Copyright Protection". In Proc. of *ACM International Conference Proceeding Series*. 85, pp. 62-68. Shanghai, China.
- Wikipedia (2008a). "Bézier curve", http://en.wikipedia.org/wiki/B%C3%A9zier_curve, Accessed on: 23. May 2008.
- Wikipedia (2008b). "Bézier Surface", <http://en.wikipedia.org/wiki/NURBS>, Accessed on: 24. November 2008.
- Wikipedia (2008c). "Egyptian War Symbols", http://www.google.co.uk/images?hl=en&biw=1362&bih=624&gbv=2&tbs=isch%3A1&sa=1&q=egyptian+war+symbol&aq=f&aqi=&aql=&oq=&gs_rfai, Accessed on: 18. February 2008.
- Wikipedia (2008d). "Polygonal Mesh", http://en.wikipedia.org/wiki/Polygon_mesh, Accessed on: 10 July 2008.
- Wikipedia (2008e). "Voxel", <http://en.wikipedia.org/wiki/Voxel>, Accessed on: 09. November 2008.
- Wikipedia (2009a). "Constructive Solid Geometry", http://en.wikipedia.org/wiki/Constructive_solid_geometry, Accessed on: 23. January 2009.
- Wikipedia (2009b). "Freeform surface modelling", http://en.wikipedia.org/wiki/Freeform_surface_modelling, Accessed on: April 2009.
- Wikipedia (2009c). "Polygon Mesh", http://en.wikipedia.org/wiki/Polygon_mesh, Accessed on: 17. January 2009.
- Wolfgang, R. & Delp, E. (1997). "A watermarking technique for digital imagery: Further studies". In Proc. of *Image Science, Systems, and Technology*. pp. 279-287. Las Vegas, NV, USA, Jun.-Jul. 1997.
- Wolfgang, R., Podilchuk, C. & Delp, E. (1999). "Perceptual watermarks for digital images and video". In Proc. of *IEEE*. 87, pp. 1108-1126.
- Wood, W.G.M. (1974). *Experimental method : a guide to the art of experiment for students of science and engineering* London, Athlone press
- Wu, H. & Cheung, Y. (2005a). "A fragile watermarking scheme for 3D meshes". In Proc. of *7th International Multimedia Conference on Multimedia and Security*. pp. 117-124. New York, NY, USA, 1-2 Aug. 2005.
- Wu, H. & Cheung, Y. (2005b). "A New Fragile Mesh Watermarking Algorithm for Authentication". In Proc. of *Information Security Conference*. pp. 509-523. Chiba, Japan, May 2005.
- Wu, M., Miller, M., Bloom, J. & Cox, I. (1999). "A rotation, scale, and translation resilient public watermark". In Proc. of *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '99)*. Phoenix, AZ, USA.
- Xia, X., Boncelet, C. & Arce, G. (1997). "A multiresolution watermark for digital images". In Proc. of *IEEE International Conference on Image Processing (ICIP '97)*. 1, pp. 548-551. Santa Barbara, CA, USA, Oct. 1997.
- Yeo, B. & Yeung, M. (1999). "Watermarking 3D Objects for Verification". *IEEE Computer Society Press Los Alamitos, CA, USA* 19 (1), pp. 36-45.

- Yin, K., Pan, Z., Shi, J. & Zhang, D. (2001). "Robust mesh watermarking based on multiresolution processing". *Computers and Graphics*, 25 (3), pp. 409-420.
- Youd, D. (1996). "What is a digital signature?" <http://www.youdzone.com/signature.html>, Accessed on:
- Zafeiriou, S., Tefas, A. & Pitas, I. (2005). "Blind robust watermarking schemes for copyright protection of 3D mesh objects". *IEEE Transactions on Visualization and Computer Graphics*, 11 (5), pp. 596-607.
- Zeijl J. Van (1994). "Constructive solid geometry", http://graphics.stanford.edu/~cek/rayshade/doc/guide/section2_7_4.html, Accessed on: 10 July 2008.

Appendices

Appendix A PO-watermark Encoding and Embedding Codes

Appendix B PO-watermark Recovery Codes

Appendix C Test-Result Data for L2 and L3 Test Objects

Appendix D Summary of Survey Results

Appendix A PO-watermark Encoding and Embedding Codes

```
/**
//
// PO-watermark Embedding.cpp
//
/**

#include <iostream>
#include "api.hxx" // Declares outcome class
#include <stdio.h> // Declares I/O functions, print functions
#include "cstrapi.hxx" // Declare construction APIs,
#include "kernapi.hxx" // Start modeller API,
#include "body.hxx" // Declares BODY class, Topological Class BODY,
#include "debug.hxx" // Debug routines
#include "lists.hxx" // Entity List Class, Declares ENTITY_LIST class
#include "curve.hxx"
#include "curdef.hxx"
#include "allsurf.hxx" // Declares SURFACE class
#include "bnd_crv.hxx"
#include "transf.hxx" // Declares TRANSFORM class
#include "alltop.hxx"
#include "tensor.hxx"
#include "boolapi.hxx" // Declares Boolean APIs
#include "stchapi.hxx"
#include "lop_api.hxx"
#include "fileinfo.hxx" // Declares FileInfo class
#include "errorsys.hxx" // Declare error functions
#include "getbox.hxx"
#include "rnd_api.hxx"
#include "ctapi.hxx"
#include "heal_api.hxx" // Declares healing
#include <math.h> // Declares math functions
#include <stdlib.h> // Includes standard library
#define pi 3.141592654 // Define PI

using namespace std;

/**

ENTITY* restore_ent(char*); // Restore CAD model attribute data
void save_ent(char*, ENTITY*); // Write and save header file Info
int face_loops(FACE*); // Number of EDGES and LOOPS
```

```

void check_result(outcome result, char *mess); // Check outcome of API functions
int main()
{
    char Filename[20]; // Filename restriction to 20 characters
    char FilenameCopy[20];

//***** Initialisation *****/

    api_start_modeller(0); // Initialisation of modeller
    api_initialize_local_ops(); // Initialisation of local operations
    api_initialize_booleans(); // Initialisation of Boolean library
    api_initialize_healing(); // Initialisation of healing
    BODY* block = NULL; // Declare a pointer to a BODY object

//***** Load file from *****/

    cout << "Please enter file name (without .sat extension): ";
    cin >> Filename;
    char DirName[] = "C:/Users/Csaba/Documents/C++ Codes/TestData/L3/";
        // defines location of file
    char *FilenameWhole = strcat(DirName, Filename); // combines
location and Filename
    cout << "\n";
    char *FilenameWholeIn = strcat(FilenameWhole, ".sat"); // combines
FilenameWhole with .sat ending
    cout << FilenameWholeIn << endl; // show combined file name on screen
    cout << "\n";
    block = (BODY*)restore_ent(FilenameWholeIn); // load file and declare it
as block

//***** Healing *****/

    api_hh_init_body_for_healing(block); // Initialise body for healing
    api_hh_simplify_auto(block); //
Autohealing/Simplification process
    api_hh_end_body_for_healing(block); // Stop body for healing
    cout << "Healing completed." << endl; // confirm healing
    cout << "\n";

//***** Message Converter *****/

    char message[12]; // defines message and restricts its length to 11
characters
    int i;
    cout << "Please enter message (max. 11 characters, no free space): ";
    cin >> message;
    char *t = message; // t equals message
    cout << "\n";

```

```

int sz = strlen(t);    // strlen = stringlength, captures automatically the
length of a string

int c[12]; //ASCII value of a letter
for (int ii=0; ii < 12; ii++)
{
    c[ii] = 0;    // initialize the array
}

for (i = 0; i < sz; i++) // while the string is not at the end do the following
{
    if (t[i] != '\0')    // fills the c array with values
    {
        c[i] = __toascii(t[i]); // converts message letter by letter into
ASCII decimal
        cout << "The ASCII decimal value is: " << c[i] << endl;
        cout << "In radians: " << (c[i]*pi/180) << endl;
        cout << "\n";
    }
}

//***** Face separation *****/

FACE *ff = block->lump()->shell()->face_list(); // go to face level
ENTITY_LIST faceList;    // place ENTITYs in an
ENTITY_LIST
api_get_faces(block, faceList);    // create a faceList from the block and
get first face

int index = 0;    // set start index to zero
int letter = 0;
int limit = faceList.count(); // set max limit = number of faces in faceList

if (limit > 1000)    // stipulation to 1000 faces
{
    printf("Too many faces!\n"); // print out message
    exit(1);    // stop processing
}

while(index < limit)    // while index value lover than limit
{
    ff = (FACE*)faceList[index];    // decleration of ff
    SURFACE *fs = ff->geometry();    // returns pointer to SURFACE
object, definition of face's shape

//***** Plane *****/

if (fs->identity() == PLANE_TYPE)// if the face's surface is a PLANE

```

```

    {
        printf("\nFace number %d is a plane!\n", index);

        PLANE* cs = (PLANE*)fs;          //cast pointer from
SURFACE to PLANE

        plane const planesurf = (plane&) (fs->equation());

        SPAunit_vector pdir = cs->normal();

        SPAvector u_dir = planesurf.u_axis();

        plane *copysurf = (plane*)planesurf.deep_copy(); // copy
original face

        if (c[letter] == 0)
        {
            letter = 0;    //reset letter counter to start of message
        }

        (*copysurf)*=rotate_transf((c[letter]*pi/180),pdir); //
rotate/tweak plane surface by 0.5 rads

        PLANE *rotplane = new PLANE(*copysurf);    // declare
rotated face = new plane face

        SURFACE* psurface_array[1];                //
put rotated/"new" face back

        psurface_array[0] = (SURFACE*)rotplane; // put
rotated/"new" face back

        FACE* pface_array[1];

        int prevflag[1];

        prevflag[0] = 0;

        pface_array[0] = ff;

        SPAbox pfb = get_face_box(ff);

        outcome result = api_tweak_faces(1,pface_array,
psurface_array, prevflag, pfb.low(), pfb.high());

        check_result(result, "Plane rotation operation");

        if(!result.ok()) // check outcome of tweak operation

```

```

        {
            printf("Planar tweaking failed!\n");
            letter--;          // put letter back for a new
tweaking
            api_rh_set_entity_rgb(ff,rgb_color(1));    // change
colour to red
        }
        else
        {
            printf("Planar tweaking successful.\n");
            api_rh_set_entity_rgb(ff,rgb_color(3));    // change
colour to blue
        }

        fs = pface_array[0]->geometry();

        plane const nplanesurf = (plane&) (fs->equation());

        SPAvector nu_dir = nplanesurf.u_axis();
    }

//***** Cone *****

        else if (fs->identity() == CONE_TYPE) // if the face's surface is a
cone
        {
            printf("\nFace number %d is a cone\n", index);

            CONE* cs = (CONE*)fs;          // cast
pointer from SURFACE to CONE

            cone const conesurf = (cone&) (fs->equation());

            if ((conesurf.closed_v() || conesurf.closed_u())
&& face_loops(ff) > 1) // if this face is a closed cylinder
            {
                // Get cone data to calculate transform to
origin

                SPAunit_vector dir = cs->direction();
                SPAposition pos = cs->root_point();

                SPAvector major = cs->major_axis();

```

```

major.orthogonal());
normalise(major);

origin to cone
coordinate_transf(pos, minor, majorv);

// Calculate transform to move from
SPAtansf move =
api_apply_transf(block, move.inverse());
// move hole axis to origin (0,0,0)

api_change_body_trans(block,NULL);
// force transform to be fully evaluated

// Define arguments for watermaking
"tweak" operation
FACE* face_array[1];
face_array[0] = ff;
SPAbox fb = get_face_box(ff);
SPAunit_vector tdir = cs->direction();

if (c[letter] == 0)
{
    letter = 0;    //reset letter
counter to start of message
}

// Carry out tweak operation, this
replaces the original face with the rotated one
outcome result =
api_move_faces(1,face_array, rotate_transf((c[letter]*pi/180),tdir), fb.low(),
fb.high());    // rotate CONE surface by 0.5 rads

check_result(result, "Tweak operation");

if(!result.ok()) // check outcome of
tweak operation
{
    printf("Cone tweaking
failed!\n");

    letter--;    // put letter back
for tweaking

```

```

        api_rh_set_entity_rgb(ff,rgb_color(1));    // change colour to red
    }
    else
    {
        printf("Cone tweaking
successful.\n");

        api_rh_set_entity_rgb(ff,rgb_color(3));    // change colour to blue
    }

    api_apply_transf(block, move);    //
move body back to origin

    api_change_body_trans(block,NULL);
// fully evaluate the transform

    SURFACE *fs = ff->geometry();

    CONE* cs = (CONE*)fs;    //cast
pointer from SURFACE to CONE

    SPAvector tmajor = cs->major_axis();
    }    // end if closed surface
    else
    {
        printf("Open conical face. Tweaking not
possible!\n");

        letter--;
        api_rh_set_entity_rgb(ff, rgb_color(4));
// change colour to lightblue
    }

}

//***** Sphere *****

    else if (fs->identity() == SPHERE_TYPE)    // if the face's
surface is a SPHERE
    {
        printf("\nFace number %d is a sphere!\n",
index);

```

```

                SPHERE* cs = (SPHERE*)fs;      //cast
pointer from SURFACE to SPHERE

                sphere const spheresurf = (sphere&) (fs-
>equation()); // get lower case surface object

                SPAposition pos = cs->centre();      //
get sphere centre point to calculate transform

                SPAunit_vector orn = spheresurf.uv_oridir; //
direction from the centre of the sphere to the origin of parameter space

                SPAunit_vector axis = spheresurf.pole_dir; //
direction normal to uv_oridir that points from the centre to the "north pole" of
the sphere

                // Calculate transform to move from origin to
sphere
                SPAtransf move = coordinate_transf(pos,
SPAunit_vector(1,0,0),SPAunit_vector(0,1,0));

                api_apply_transf(block, move.inverse()); //
move hole axis to origin (0,0,0)

                api_change_body_trans(block,NULL); //
force transform to be fully evaluated

                FACE* face_array[1];      // Define
arguments for watermaking "tweak" operation

                face_array[0] = ff;

                SPAbox fb = get_face_box(ff);

                if (c[letter] == 0)
                {
                    letter = 0;      //reset letter counter to
start of message
                }

                // rotate SPHERE surface by 0.5 rads
outcome result = api_move_faces(1,face_array,
rotate_transf((c[letter]*pi/180),axis), fb.low(), fb.high());

                // Carry out tweak operation, this replaces the
original face with the rotated one
                check_result(result, "Move operation");

```



```

operation
    if(!result.ok()) // check outcome of tweak
    {
        printf("Sphere tweaking failed!\n");
        letter--;    // put letter back for
tweaking
        api_rh_set_entity_rgb(ff,rgb_color(1));
        // change colour to red
    }
    else
    {
        printf("Sphere tweaking successful.\n");
        api_rh_set_entity_rgb(ff,rgb_color(3));
        // change colour to blue
    }
    api_apply_transf(block, move);    // move
body back to origin
    api_change_body_trans(block,NULL);    //
fully evaluate the transform
    SURFACE *ffs = ff->geometry();    // change
orientation of face
    SPHERE* ffcs = (SPHERE*)ffs;    //cast
pointer from SURFACE to SPHERE
    sphere const spheresurff = (sphere&) (ffs-
>equation());
    SPAunit_vector afterort = spheresurff.uv_oridir;
    SPAunit_vector tafterort =
spheresurff.uv_oridir;
    SPAunit_vector torn = spheresurf.uv_oridir;
    // direction from the centre of the sphere to the origin of parameter
space
    }

```

```

//***** Torus *****

```

```

else if (fs->identity() == TORUS_TYPE) // if the face's
surface is a torus
{
    printf("\nFace number %d is a torus\n", index);

    TORUS* cs = (TORUS*)fs; // Cast
pointer from SURFACE to TORUS

    torus const torussurf = (torus&) (fs-
>equation()); // needed to identify the closed toroidal faces

    // Get torus data to calculate transform to origin

    const SPAposition cen= torussurf.centre;
// get centre of torus face

    const SPAunit_vector norm = torussurf.normal;
// rotate about this vector

    const SPAunit_vector torplane =
torussurf.uv_oridir; // location of uv start point

    double bigr = torussurf.major_radius; //
get major radius of torus

    double smallr= torussurf.minor_radius; //
get minor radius of torus

    SPAunit_vector axis = normalise(norm *
torplane); // dot product to get vector for coordinate_transf

    SPAttransf move = coordinate_transf(cen,
torplane, axis); // Calculate transform to move from origin to torus

    api_apply_transf(block, move.inverse()); //
move hole axis to origin (0,0,0)

    api_change_body_trans(block,NULL);
// force transform to be fully evaluated

// Define arguments for watermaking "tweak"
operation
FACE* face_array[1];

face_array[0] = ff;

SPAbox fb = get_face_box(ff);

```

```

torus const torussurfM = (torus&) (fs-
>equation()); // needed to identify the closed toroidal faces

const SPAunit_vector normR =
torussurfM.normal; // rotate about this vector

if (c[letter] == 0)
{
    letter = 0; //reset letter counter to
start of message
}

// rotate torus surface around the vector
"normR" by 0.5 rads
outcome result = api_move_faces(1,face_array,
rotate_transf((c[letter]*pi/180),normR), fb.low(), fb.high());

// Carry out tweak operation, this replaces the
original face with the rotated one
check_result(result, "Tweak operation");

if(!result.ok()) // check outcome of tweak
operation
{
    printf("Torus tweaking failed!\n");
    letter--; // put letter back for
tweaking

    api_rh_set_entity_rgb(ff,rgb_color(1));
// ste colour to red
}
else
{
    printf("Torus tweaking successful.\n");
    api_rh_set_entity_rgb(ff,rgb_color(3));
// set colour to blue
}

api_apply_transf(block, move); // move
body block to origin

api_change_body_trans(block,NULL); //
fully evaluate the transform

SURFACE *ffs = ff->geometry();

```

```

        torus const mtorussurf = (torus&) (ffs-
>equation()); // needed to identify the closed toroidal faces

        const SPAunit_vector mtorplane =
mtorussurf.uv_oridir; //location of uv start point

        const SPAunit_vector ntorplane =
torussurf.uv_oridir; // location of uv start point

        const SPAunit_vector tmtorplane =
mtorussurf.uv_oridir; //location of uv start point
        } // end if torus surface
    else
    {
        printf("Face type not defined. Tweaking not possible!\n");
        letter--;
        api_rh_set_entity_rgb(ff, rgb_color(4)); // change colour to
lightblue
    }
    //letter--;

//*****
    letter++;
    index++;
}

//***** Save file to *****

api_change_body_trans(block,NULL);
char DirName2[] = "C:/Users/Csaba/Documents/C++
Codes/ResultData/L3/Reference/"; // defines location for saving
char *FilenameWhole2 = strcat(DirName2, Filename); // combines
location and Filename
char *FilenameWholeOut = strcat(FilenameWhole2, ".sat"); // combines
FilenameWhole2 and .sat ending, removed w before .sat to eliminate
problems at later stages
cout << "\n";
cout << FilenameWholeOut << endl; // print out combined file name
cout << "\n";
save_ent(FilenameWholeOut, block); // function to save watermarked
part

//***** End Initialisation *****

api_terminate_healing(); // Termination of healing
api_terminate_booleans(); // Termination of Boolean library
api_terminate_local_ops(); // Termination of local operations

```

```

        api_stop_modeller();          // Termination of modeller

//*****

    return 0;
}

//***** Restoring *****

ENTITY *restore_ent(char *filename)
{
    file
    ENTITY *ent = NULL;
    FILE *fp = fopen(filename, "r");
    if (fp != NULL)
    {
        ENTITY_LIST restorelist;
        outcome check = api_restore_entity_list(fp,TRUE,restorelist);
        fclose(fp);
        if (!check.ok() || restorelist.count() < 1 )
        {
            printf("Error reading file\n");
            exit(0);
        } // Assuming only one entity is restored
        else
            ent = restorelist[0];
    }
    else
    {
        printf("Unable to open file!\n");
        exit(0);
    }
    return ent;
}

//***** FileInfo Saving *****

void save_ent(char *filename, ENTITY *ent)
{
    FileInfo info;          // Create FileInfo Object
    info.set_product_id("Strathclyde"); // set info's data
    info.set_units(1.0);    // Millimeters
    api_set_file_info(FileId | FileUnits, info); // Sets header info to be written to
    sat file

    FILE *fp = fopen(filename, "w");
    if (fp != NULL)
    {

```

```

        ENTITY_LIST *savelist = new ENTITY_LIST;
        savelist->add(ent);
        api_save_entity_list(fp,TRUE,*savelist);
        delete savelist;
    }
    else
        printf("Unable to open file!\n");
    fclose(fp);
}

//***** EDGES and LOOPS *****

int face_loops(FACE* f)
{
    ENTITY_LIST ll;
    api_get_loops(f,ll);
    return ll.count();
}

//***** Check API functions *****

void check_result(outcome result, char *mess)
{
    if(!result.ok())
    {
        err_mess_type err_number=result.error_number();
        printf("Error in %s : %s\n",mess,find_err_mess(err_number));
    }
}

//*****

// colour codes are: 0=black, 1=red, 2=green, 3=blue, 4=lightblue, 5=yellow, 6=pink,
// 7=white, 8=grey, 9=red

//***** Bug fix for torus *****
// After the whole body has been moved (transformed) to the origin
// the original value of the "norm" vector (ie the plane of the torus
// has changed. So rotation about it only works if its alignment has not
// altered (as in the case of the x-axis, all other axes fail).
// So SOLUTION is to assign new variables to hold the new value of norm.
// the next two line simply repeat ealier operations to get an updated value of
// norm, called normR, after the transform.

```

Appendix B PO-watermark Recovery Codes

```
/**
//
// PO-watermark Recovery.cpp
//
/**

#include <iostream>
#include "api.hxx" // Declares outcome class
#include <stdio.h> // Declares I/O functions, print functions
#include "cstrapi.hxx" // Declare construction APIs,
#include "kernapi.hxx" // Declares kernel APIs, start modeller API,
#include "body.hxx" // Declares topological class (BODY)
#include "debug.hxx" // Debug routines
#include "lists.hxx" // Declares ENTITI_LIST class,
#include "curve.hxx"
#include "curdef.hxx"
#include "allsurf.hxx" // Declares SURFACE classes
#include "bnd_crv.hxx"
#include "transf.hxx" // Declares the TRANSFORM class
#include "alltop.hxx"
#include "boolapi.hxx" // Declares Boolean APIs
#include "stchapi.hxx"
#include "lop_api.hxx"
#include "fileinfo.hxx" // Declares fileinfo class
#include "errorsys.hxx" // Declare error functions
#include "getbox.hxx"
#include "sphere.hxx"
#include "mprop.hxx"
#include "intrapi.hxx"
#include "geom_utl.hxx"
#include "acistol.hxx" // is_equal
#include <math.h>
#include <stdlib.h>
#include <string>

#define pi 3.1415926535897932384626433832795
using namespace std;

/**

ENTITY* restore_ent(char*); // Gets test BODY from file
void save_ent(char*, ENTITY*); // Function for writing BODY to file
int face_loops(FACE*); // Prints number of LOOPS
int face_edges(FACE*); // Prints number of EDGES
void check_result(outcome result, char *mess); // Check API functions
```

```

FACE *get_matching_face(FACE*f, ENTITY_LIST& listm, int& matches);    //
    Get the matching face
int align_matching_faces(FACE*, FACE*);// align matching faces

//***** Definition for rounding *****

double round0(double value)
{
    if ((value - abs(value)) >= 0.5)
        return ceil(value);
    else
        return floor(value);
}

//*****
//int main();
int main(int argc, char* argv[])
{
    char Filename[20];
    char FilenameCopy[20];
    string ostr("Embedded word = ");

//***** Initialisation *****
    api_start_modeller(0);           // Initializes the modeller
    api_initialize_local_ops();      // Initializes local operations
    api_initialize_booleans();       // Initializes the boolean library

    BODY* blockO = NULL; // Declare a pointer to a BODY object
    BODY* blockW = NULL; // Declare a pointer to a BODY object

//***** Load file *****

    // enter name
    cout << "Please enter file name (without .sat extension): ";
    cin >> Filename;

    // load original part
    char DirNameO[] = "C:/Users/Csaba/Documents/C++ Codes/TestData/L1/";
    char *FilenameWhole1 = strcat(DirNameO, Filename);
    cout << "\n";
    char *FilenameWholeO = strcat(FilenameWhole1, ".sat");
    cout << FilenameWholeO << endl;
    cout << "\n";
    blockO = (BODY*)restore_ent(FilenameWholeO);

    // load watermarked part
    char DirNameW[] = "C:/Users/Csaba/Documents/C++
Codes/ResultData/L1/Adding/";

```



```

char *FilenameWhole2 = strcat(DirNameW, Filename);
cout << "\n";
char *FilenameWholeW = strcat(FilenameWhole2, ".sat");           // removed
w before .sat to eliminate problems a later stages
cout << FilenameWholeW << endl;
cout << "\n";
blockW = (BODY*)restore_ent(FilenameWholeW);

// save check report
char DirNameR[] = "C:/Users/Csaba/Documents/C++
Codes/ResultData/L1/Adding/";
char *FilenameWhole3 = strcat(DirNameR, Filename);
cout << "\n";
char *FilenameWholeR = strcat(FilenameWhole3, "res.wmk");
cout << FilenameWholeR << endl;
cout << "\n";
FILE* output = fopen(FilenameWholeR, "w");

//***** Face alignment *****/
// HAD TO CHANGE THIS BECAUSE EVEN IDENTICAL OBJECT HAD
// SLIGHTLY DIFFERENT VOLUMES AFTER
// PARAMETERS HAVE BEEN REORIENTATED (EG 14.567345 !=
// 14.567999)!
mass_props_options ops;
ops.set_req_rel_accy(SPAresabs); //max accuracy
mass_props massprops1;    // mprop.hxx
mass_props massprops2;

api_body_mass_props(blockO, massprops1, &ops);
api_body_mass_props(blockW, massprops2, &ops);

double vol1int;
double vol2int;
double vol1 = modf(massprops1.get_volume(), &vol1int);
double vol2 = modf(massprops2.get_volume(), &vol2int);

double angp = (0);    // extra
const char* c_str(); // extra

// should really round to 3 or 4 significant decimal places
// but for now just compare the integer part of the volume values

// if (!is_equal(vol1int, vol2int))
// {
//     printf("Original and watermarked bodies have different volumes, so
// exiting \n");
//     printf("Original vol = %10.6f, water vol = %10.6f \n",
// massprops1.get_volume(), massprops2.get_volume());

```

```

//          exit(0);
//      }
//  else // ok these could be the same model because they have roughly the
//  same volume
//  {
//      printf("Volumes the same\n");

//      fprintf(output, "\nAfter watermark reading: \n\n");

//***** Face separation *****

// place ENTITYs in an entity list
ENTITY_LIST faceListO; // original
ENTITY_LIST faceListW; // watermarked M

// create a faceList from the block and get first face
api_get_faces(blockO, faceListO); // original
api_get_faces(blockW, faceListW); // watermarked M

int index = 0; // set start index to zero
int limit = faceListO.count(); // set max limit = number of faces in
faceList
while (index <limit)
{
    FACE *ff = (FACE*)faceListO[index];
    int matchresult = 0;
    FACE *fm = get_matching_face(ff, faceListW, matchresult);
// get the matching faces M
    printf("Number of matches = %d\n",matchresult);
// matchresult = 1 if only one face matches, 0 if no faces
match, 2 if two faces match

    if (matchresult > 0 && fm != NULL)
    { // then at least one face could match,

        // so check alignment
        int alignresult = align_matching_faces(ff, fm);
        if (alignresult == 0)
        {
            index++; continue;
        } //not aligned, jump to end of loop and next face
        if (alignresult == 1)
        {
            printf("matching and aligned face found, now to
check para alignment\n");
        }
        SURFACE *fs = ff->geometry();
        SURFACE *fsm = fm->geometry();
    }
}

```

```

//***** PLANE *****
if (fs->identity() == PLANE_TYPE && fsm-
>identity() == PLANE_TYPE)
{
    printf("\nFace number %d is a plane.\n", index);
    //***** Get same vector for matching
    face *****
    PLANE* cs = (PLANE*)fs; //cast pointer from
    SURFACE to PLANE
    plane const planesurf = (plane&) (fs-
>equation());
    SPAvector norm = planesurf.normal;
    //new
    SPAvector u_dir = planesurf.u_deriv;
    //new
    PLANE* csm = (PLANE*)fsm; //cast pointer
    from SURFACE to PLANE
    plane const planesurfm = (plane&) (fsm-
>equation());
    SPAvector normm = planesurfm.normal;
    //new
    SPAvector u_dirm = planesurfm.u_deriv;
    //new

    //***** Output of coordinates
    *****
    fprintf(output, "Plane: \n");
    fprintf(output, "Original face:
%10.6f,%10.6f,%10.6f \n", u_dir.x(),u_dir.y(),u_dir.z());
    fprintf(output, "Watermarked face:
%10.6f,%10.6f,%10.6f \n", u_dirm.x(),u_dirm.y(),u_dirm.z());

    //***** Calculate angle between u_dir
    and u_dirm *****
    double angp = angle_between(u_dir, u_dirm);
    // in radians

    //***** Output of angle
    *****
    fprintf(output, "Angle difference: %10.6f \n",
angp);

    //***** Conversion to decimal/text
    *****
    //int x = 0;

```

```

round0(angp/pi*180));
(char)round0(angp/pi*180));

fprintf(output, "Decimal value is: %10.0f\n",
fprintf(output, "Text value is:   %c\n",
if (round0(angp/pi*180) != 0)
{
    char l=(char)round0(angp/pi*180);
    outstr = outstr + l;
}
cout<< "wOutput" << outstr << endl;

fprintf(output, "Watermark = %s\n\n",
outstr.c_str());
} // end if planar

//***** CONE *****

else if (fs->identity() == CONE_TYPE && fsm-
>identity() == CONE_TYPE)
{
    printf("\nFace number %d is a cone.\n", index);
    //***** Get same vector for matching
faces *****
    CONE* cs = (CONE*)fs; //cast pointer from
SURFACE to CONE
    SPAvector major = cs->major_axis();
    SPAunit_vector minor = major.orthogonal();
    SPAunit_vector majorv = normalise(major); //
for angle calculation
    cone const conesurf = (cone&)(fs->equation());

    CONE* csm = (CONE*)fsm; // cast
pointer from SURFACE to CONE
    SPAvector majorm = csm->major_axis();
    SPAunit_vector minorm =
majorm.orthogonal();
    SPAunit_vector majorvm = normalise(majorm);
// for angle calculation
    cone const conesurfm = (cone&)(fsm-
>equation());

    //***** Output of coordinates
*****
    fprintf(output, "Cone: \n");
    // need to identify the closed conical faces
    if ((conesurf.closed_v() || conesurf.closed_u())
&& face_loops(ff) >1) // if face is a closed cylinder

```

```

        {
            fprintf(output, "Original face:
%10.6f,%10.6f,%10.6f \n", major.x(),major.y(),major.z());
        } // end if closed

        if ((conesurfm.closed_v() ||
conesurfm.closed_u()) && face_loops(fm) >1)
        {
            fprintf(output, "Watermarked face:
%10.6f,%10.6f,%10.6f \n", majorm.x(),majorm.y(),majorm.z());
        } // end if closed
        else
            fprintf(output, "Face not closed conical!
\n");

        //***** Calculate angle between majorv
and majorvm ***
        double angc = angle_between(majorv,
majorvm); // in radians

        //***** Output of angle
        *****
        fprintf(output, "Angle difference: %10.6f \n",
angc); // fprintf is output in wmk file

        //***** Conversion to text
        *****
        fprintf(output, "Decimal value is: %10.0f \n",
round0(angc/pi*180));
        fprintf(output, "Text value is: %c \n",
(char)round0(angc/pi*180));

        if (round0(angc/pi*180) != 0)
        {
            char l =(char)round0(angc/pi*180);
            outstr = outstr + l;
        }
        cout<< "wOutput" << outstr << endl;

        fprintf(output, "Watermark = %s \n\n",
outstr.c_str());

    } // end if closed cone

//***** SPHERE *****
        else if (fs->identity() == SPHERE_TYPE && fsm-
>identity() == SPHERE_TYPE)

```

```

        {
            printf("\nFace number %d is a sphere.\n",
index);
            //***** Get same vector for matching
faces *****
            SPHERE* cs = (SPHERE*)fs;          //
casts a pointer from SURFACE to SPHERE
            sphere const spheresurf = (sphere&) (fs-
>equation()); // get the lower case surface object
            SPAunit_vector tafterort = spheresurf.uv_oridir;
            SPAunit_vector torn = spheresurf.uv_oridir;
            // direction from the centre of the sphere to the origin of parameter
space

            SPHERE* csm = (SPHERE*)fsm;          //
casts a pointer from SURFACE to SPHERE
            sphere const spheresurfm = (sphere&) (fsm-
>equation()); // get the lower case surface object
            SPAunit_vector tafterortm =
spheresurfm.uv_oridir;
            SPAunit_vector tornm = spheresurfm.uv_oridir;
            // direction from the centre of the sphere to the origin of parameter
space

            //***** Output of coordinates
            *****
            fprintf(output, "Sphere: \n");
            fprintf(output, "Original face:
%10.6f,%10.6f,%10.6f \n", tafterort.x(),tafterort.y(),tafterort.z());
            fprintf(output, "Watermarked face:
%10.6f,%10.6f,%10.6f \n", tafterortm.x(),tafterortm.y(),tafterortm.z());

            //***** Calculate angle between tafterort
and tafterortm
            double angs = angle_between(tafterort,
tafterortm); // in radians

            //***** Output of angle
            *****
            fprintf(output, "Angle difference: %10.6f \n",
angs);

            //***** Conversion to text
            *****
            fprintf(output, "Decimal value is: %10.0f \n",
round0(angs/pi*180));
            fprintf(output, "Text value is:   %c \n",
(char)round0(angs/pi*180));

```

```

        if (round0(angs/pi*180) != 0)
        {
            char l =(char)round0(angs/pi*180);
            outstr = outstr + l;
        }
        cout<< "wOutput = " << outstr << endl;

        fprintf(output, "Watermark = %s \n\n",
outstr.c_str());
    } // end if sphere

//***** TORUS *****/

        else if (fs->identity() == TORUS_TYPE && fsm-
>identity() == TORUS_TYPE)
        {
            printf("\nFace number %d is a torus.\n", index);
            //***** Get same vector for matching
faces *****
            TORUS* cs = (TORUS*)fs; // Cast pointer
from SURFACE to TORUS
            torus const torussurf = (torus&) (fs-
>equation()); // needed to identify the closed toroidal faces
            const SPAunit_vector ntorplane =
torussurf.uv_oridir; // location of uv start point

            TORUS* csm = (TORUS*)fsm; // Cast
pointer from SURFACE to TORUS
            torus const torussurfm = (torus&) (fsm-
>equation()); // needed to identify the closed toroidal faces
            const SPAunit_vector ntorplanem =
torussurfm.uv_oridir; // location of uv start point

            //***** Output of coordinates
*****
            fprintf(output, "Torus: \n");
            fprintf(output, "Original face:
%10.6f,%10.6f,%10.6f \n", ntorplane.x(),ntorplane.y(),ntorplane.z());
            fprintf(output, "Watermarked face:
%10.6f,%10.6f,%10.6f \n", ntorplanem.x(),ntorplanem.y(),ntorplanem.z());

            //***** Calculate angle between
ntorplane and ntorplanem
            double angt = angle_between(ntorplane,
ntorplanem); // in radians

```

```

//***** Output of angle
*****
fprintf(output, "Angle difference: %10.6f \n",
angt);

//***** Conversion to text
*****
fprintf(output, "Decimal value is: %10.0f \n",
round0(angt/pi*180));
fprintf(output, "Text value is:          %c \n",
(char)round0(angt/pi*180));

if (round0(angt/pi*180) != 0)
{
    char l =(char)round0(angt/pi*180);
    outstr = outstr + l;
}
cout<< "wOutput = " << outstr << endl;

fprintf(output, "Watermark = %s \n\n",
outstr.c_str());

} // end if torus

//*****
}
else
{
    fprintf(output, "No matching faces. Reading not
possible!\n\n");
}
index++;
}

if(round0(angp/pi*180) >= 0)
{
    char l =(char)round0(angp/pi*180);
    outstr = outstr + l;
}
cout<< "wOutput = " << outstr << endl;
fprintf(output, "Output = %s \n", outstr.c_str());

}

//***** Debug file *****

fprintf(output, "\n");
fprintf(output, "Debug file of original model: \n");

```



```

debug_size(blockO, output); // original
fprintf(output, "\n");
fprintf(output, "Debug file of watermarked model: \n");
debug_size(blockW, output); // watermarked
fclose(output);

//***** End Initialisation *****/

api_terminate_booleans();
api_terminate_local_ops();
api_stop_modeller();

//*****

return 0;
}

//***** Restoring *****/

ENTITY *restore_ent(char *filename) // Load the file from the sat file
{ // Loads in a sat file
    ENTITY *ent = NULL;
    FILE *fp = fopen(filename, "r");

    if (fp != NULL)
    {
        ENTITY_LIST restorelist;
        outcome check = api_restore_entity_list(fp, TRUE, restorelist);
        fclose(fp);

        if (!check.ok() || restorelist.count() < 1 )
        {
            printf("Error reading file\n");
            exit(0);
        } // Assuming only one entity is restored

        else
            ent = restorelist[0];
    }

    else
    {
        printf("Unable to open file!\n");
        exit(0);
    }

    return ent;
}

```

```
//***** FileInfo Saving *****
```

```
void save_ent(char *filename, ENTITY *ent)
{
    FileInfo info;          // Create FileInfo Object
    info.set_product_id("Salamon"); // set info's data

    info.set_units(1.0);           // Millimeters

    // Sets header info to be written to sat file
    api_set_file_info(FileId | FileUnits, info);

    FILE *fp = fopen(filename, "w");

    if (fp != NULL)
    {
        ENTITY_LIST *savelist = new ENTITY_LIST;
        savelist->add(ent);
        api_save_entity_list(fp,TRUE,*savelist);
        delete savelist;
    }
    else
        printf("Unable to open file!\n");
    fclose(fp);
}
```

```
//***** Counts number of LOOPS *****
```

```
int face_loops(FACE* f)
{
    ENTITY_LIST ll;
    api_get_loops(f,ll);
    return ll.count();
}
```

```
//***** Check API functions *****
```

```
void check_result(outcome result, char *mess)
{
    if(!result.ok())
    {
        err_mess_type err_number=result.error_number();
        printf("Error in %s : %s\n",mess,find_err_mess(err_number));
    }
}
```

```
//***** Counts number of EDGES *****
```

```

int face_edges(FACE* f)
{
    ENTITY_LIST ll;
    api_get_edges(f,ll);
    return ll.count();
}

//*****

FACE *get_matching_face(FACE *ff, ENTITY_LIST &faceListW, int& matches)
    //M
{
    // THIS FIRST VERSION ONLY RETURNS ONE MATCHING FACE,
    // LATER VERSIONS NEED TO RETURN A LIST
    // OF MATCHING FACES AND THEN USE ALIGNMENT FUNCTION
    // TO IDENTIFY THE CORRECT ONE!
    // BUT FOR NOW ALTHOUGH ONLY ONE FACE IS IDENTIFIED IT IS
    // A START !

    int length = faceListW.count();    // number of matches M

    // declare face parameter variable of target face
    double area = 0;
    int edgeCount = 0;
    int loopCount = 0;
    FACE *fm = NULL;

    // get values for face parameter of target face
    double acacho = 0;
    api_ent_area ((ENTITY*)ff, 0.01, area, acacho);    // cstrapi.hxx
    edgeCount = face_edges(ff);
    loopCount = face_loops(ff);

    for (int c = 0; c < length; c++)
    {
        FACE *ft = (FACE*)faceListW[c];    // get face out of list M

        // declare face parameter variable of candidate face
        double areaT = 0;
        int edgeCountT = 0;
        int loopCountT = 0;

        // get values for face parameter of candidate face
        double acach = 0;    // api_ent_area accuracy

        api_ent_area((ENTITY*)ft, 0.01, areaT, acach);    //cstrapi.hxx
    }
}

```

```

        edgeCountT = face_edges(ft);
        loopCountT = face_loops(ft);

        // test for match

        if (is_equal(area, areaT) && ff->identity() == ft->identity() &&
            edgeCount == edgeCountT && loopCount == loopCountT)
        {

            int distflag=0;
            distflag = align_matching_faces(ft,ff);
            printf("Distance matching flag is %d\n", distflag);
            if (distflag == 1)
            {
                fm = ft;
                matches++;
            };
        }
    } // end for

    return fm;
}

/*****

int align_matching_faces(FACE *ff, FACE *fm)
{
    SPAposition pff;
    SPAposition pfm;
    double dist = 100000;// distance between faces

    SPAbox b1 = get_face_box(ff);
    SPAbox b2 = get_face_box(fm);

    SPAposition b1c = b1.mid();
    SPAposition b2c = b2.mid();

    if (is_equal(b1c,b2c))
        return 1; //it is a match
    else
        return 0; //it is too far away to be a match without re-orientation
}

```

Appendix C Test-Result Data for L2 and L3 Test Objects

Level 2 Test Data

Table C0.1 Watermarking success rate for Level 2 test objects

Watermarking success rate:			
Object	No all faces	W faces	Percentage
4pin	71	7	10%
6Hole	25	13	52%
Bar	42	34	81%
Basesource	47	35	74%
Bracket	62	30	48%
Carriage	54	50	93%
Casting	67	30	45%
Chamfillet	28	28	100%
Clamp	25	18	72%
Claw	28	25	89%
Conrod	28	22	79%
Cover	46	37	80%
Crankshaft	38	24	63%
Cubehole	28	8	29%
Female	26	15	58%
Joint	25	14	56%
Link	33	22	67%
Pan1	42	20	48%
Pan2	71	49	69%
Screw	48	11	23%
Shaft	26	23	88%
Spaceship	51	29	57%
Average	41	25	62.74%

Table C0.2 Level 2 test objects after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
4pin	71	7	7	7	0	0	7
6Hole	25	13	13	13	1	9	13
Bar	42	34	34	34	0	0	34
Basesource	47	35	35	35	1	1	35
Bracket	62	30	30	30	0	2	30
Carriage	54	50	50	50	0	0	50
Casting	67	30	30	30	1	7	30
Chamfillet	28	28	28	28	1	1	28
Clamp	25	18	18	18	1	1	18
Claw	28	25	25	25	0	0	25
Conrod	28	22	22	22	2	4	22
Cover	46	37	37	37	4	4	37
Crankshaft	38	24	24	24	5	6	24
Cubehole	28	8	8	8	0	0	8
Female	26	15	15	15	2	2	15
Joint	25	14	14	14	5	8	14
Link	33	22	22	22	0	0	22
Pan1	42	20	20	20	0	0	20
Pan2	71	49	49	49	4	11	49
Screw	48	11	11	11	6	6	11
Shaft	26	23	23	23	9	9	23
Spaceship	51	29	29	29	0	3	29

Table C0.3 Level 2 test objects after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
4pin	71	7	7	7	2	1	4
6Hole	25	13	13	13	2	3	10
Bar	42	34	34	34	15	15	32
Basesource	47	35	35	35	18	16	33
Bracket	62	30	30	30	3	11	26
Carriage	54	50	50	50	21	15	48
Casting	67	30	30	30	24	15	20
Chamfillet	28	28	28	28	13	9	26
Clamp	25	18	18	18	7	9	16
Claw	28	25	25	25	12	9	20
Conrod	28	22	22	22	14	4	18
Cover	46	37	37	37	21	9	28
Crankshaft	38	24	24	24	11	13	18
Cubehole	28	8	8	8	0	1	4
Female	26	15	15	15	11	5	10
Joint	25	14	14	14	6	5	14
Link	33	22	22	22	2	13	20
Pan1	42	20	20	20	2	7	18
Pan2	71	49	47	49	45	29	46
Screw	48	11	11	11	5	6	9
Shaft	26	23	23	23	11	13	21
Spaceship	51	29	29	29	22	14	27

Table C0.4 Level 2 watermark survival rate after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
4pin	71	7	100%	100%	0%	0%	100%
6Hole	25	13	100%	100%	8%	69%	100%
Bar	42	34	100%	100%	0%	0%	100%
Basesource	47	35	100%	100%	3%	3%	100%
Bracket	62	30	100%	100%	0%	7%	100%
Carriage	54	50	100%	100%	0%	0%	100%
Casting	67	30	100%	100%	3%	23%	100%
Chamfillet	28	28	100%	100%	4%	4%	100%
Clamp	25	18	100%	100%	6%	6%	100%
Claw	28	25	100%	100%	0%	0%	100%
Conrod	28	22	100%	100%	9%	18%	100%
Cover	46	37	100%	100%	11%	11%	100%
Crankshaft	38	24	100%	100%	21%	25%	100%
Cubehole	28	8	100%	100%	0%	0%	100%
Female	26	15	100%	100%	13%	13%	100%
Joint	25	14	100%	100%	36%	57%	100%
Link	33	22	100%	100%	0%	0%	100%
Pan1	42	20	100%	100%	0%	0%	100%
Pan2	71	49	100%	100%	8%	22%	100%
Screw	48	11	100%	100%	55%	55%	100%
Shaft	26	23	100%	100%	39%	39%	100%
Spaceship	51	29	100%	100%	0%	10%	100%
Average percentage of each action:			100%	100%	10%	16%	100%

Table C0.5 Level 2 watermark survival rate after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
4pin	71	7	100%	100%	29%	14%	57%
6Hole	25	13	100%	100%	15%	23%	77%
Bar	42	34	100%	100%	44%	44%	94%
Basesource	47	35	100%	100%	51%	46%	94%
Bracket	62	30	100%	100%	10%	37%	87%
Carriage	54	50	100%	100%	42%	30%	96%
Casting	67	30	100%	100%	80%	50%	67%
Chamfillet	28	28	100%	100%	46%	32%	93%
Clamp	25	18	100%	100%	39%	50%	89%
Claw	28	25	100%	100%	48%	36%	80%
Conrod	28	22	100%	100%	64%	18%	82%
Cover	46	37	100%	100%	57%	24%	76%
Crankshaft	38	24	100%	100%	46%	54%	75%
Cubehole	28	8	100%	100%	0%	13%	50%
Female	26	15	100%	100%	73%	33%	67%
Joint	25	14	100%	100%	43%	36%	100%
Link	33	22	100%	100%	9%	59%	91%
Pan1	42	20	100%	100%	10%	35%	90%
Pan2	71	49	96%	100%	92%	59%	94%
Screw	48	11	100%	100%	45%	55%	82%
Shaft	26	23	100%	100%	48%	57%	91%
Spaceship	51	29	100%	100%	76%	48%	93%
Average percentage of each action:			100%	100%	44%	39%	83%

Level 3 Test Data

Table C0.6 Watermarking success rate for Level 3 test objects

Watermarking success rate:			
Object	No all faces	W faces	Percentage
Body	92	54	59%
Brace	137	10	7%
Carburator	241	33	14%
Case1	396	197	50%
Case2	273	157	58%
Connector	151	12	8%
Conrod	81	15	19%
Cover1	240	122	51%
Cover2	212	81	38%
Crank	112	60	54%
Exmanifold	101	12	12%
Gear	197	9	5%
Housing	190	87	46%
Mould	132	88	67%
Power	123	80	65%
Shaver	92	64	70%
Average	173	68	48%

Table C0.7 Results for Level 3 test objects after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
Body	92	54	54	54	8	9	54
Brace	137	10	10	10	0	3	10
Carburator	241	33	33	33	8	10	33
Case1	396	197	197	197	1	47	197
Case2	273	157	157	157	2	29	157
Connector	151	12	12	12	0	1	12
Conrod	81	15	15	15	0	2	15
Cover1	240	122	122	122	0	45	122
Cover2	212	81	81	81	0	11	81
Crank	112	60	60	60	10	11	60
Exmanifold	101	12	12	12	0	0	12
Gear	197	9	9	9	0	0	9
Housing	190	87	87	87	0	11	87
Mould	132	88	88	88	0	0	88
Power	123	80	80	80	0	0	80
Shaver	92	64	64	64	3	4	64

Table C0.8 Results of Level 3 test objects after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
Body	92	54	54	54	40	30	47
Brace	137	10	0	10	10	9	10
Carburator	241	33	33	33	24	16	31
Case1	396	197	197	197	173	100	191
Case2	273	157	157	157	134	105	154
Connector	151	12	12	12	12	12	12
Conrod	81	15	15	15	7	4	14
Cover1	240	122	122	122	105	50	117
Cover2	212	81	81	81	67	27	73
Crank	112	60	60	60	43	36	55
Exmanifold	101	12	13	12	7	3	9
Gear	197	9	9	9	7	6	7
Housing	190	87	87	87	66	45	81
Mould	132	88	88	88	52	41	86
Power	123	80	80	80	62	42	70
Shaver	92	64	64	64	60	29	59

Table C0.9 Level 3 watermark survival rate after data handling (or unintentional) attacks

Object	No all faces	W faces	Load	CAD	IGES	STEP	Zip
Body	92	54	100%	100%	15%	17%	100%
Brace	137	10	100%	100%	0%	30%	100%
Carburator	241	33	100%	100%	24%	30%	100%
Case1	396	197	100%	100%	1%	24%	100%
Case2	273	157	100%	100%	1%	18%	100%
Connector	151	12	100%	100%	0%	8%	100%
Conrod	81	15	100%	100%	0%	13%	100%
Cover1	240	122	100%	100%	0%	37%	100%
Cover2	212	81	100%	100%	0%	14%	100%
Crank	112	60	100%	100%	17%	18%	100%
Exmanifold	101	12	100%	100%	0%	0%	100%
Gear	197	9	100%	100%	0%	0%	100%
Housing	190	87	100%	100%	0%	13%	100%
Mould	132	88	100%	100%	0%	0%	100%
Power	123	80	100%	100%	0%	0%	100%
Shaver	92	64	100%	100%	5%	6%	100%
Average percentage of each action:			100%	100%	4%	14%	100%

Table C0.10 Level 3 watermark survival rate after geometrical modification (or intentional) attacks

Object	No all faces	W faces	Healing	Rotation	Blecha	Cutting	Adding
Body	92	54	100%	100%	74%	56%	87%
Brace	137	10	0%	100%	100%	90%	100%
Carburator	241	33	100%	100%	73%	48%	94%
Case1	396	197	100%	100%	88%	51%	97%
Case2	273	157	100%	100%	85%	67%	98%
Connector	151	12	100%	100%	100%	100%	100%
Conrod	81	15	100%	100%	47%	27%	93%
Cover1	240	122	100%	100%	86%	41%	96%
Cover2	212	81	100%	100%	83%	33%	90%
Crank	112	60	100%	100%	72%	60%	92%
Exmanifold	101	12	108%	100%	58%	25%	75%
Gear	197	9	100%	100%	78%	67%	78%
Housing	190	87	100%	100%	76%	52%	93%
Mould	132	88	100%	100%	59%	47%	98%
Power	123	80	100%	100%	78%	53%	88%
Shaver	92	64	100%	100%	94%	45%	92%
Average percentage of each action:			94%	100%	78%	54%	92%

The KIM survey investigated how companies use CAD systems and what their requirements are regarding watermarking systems.

Use of CAD systems

Companies tended to use several CAD systems simultaneously. The most common systems were AutoCAD, followed by Catia and ProEngineer. CAD models and design data were most frequently stored as .dwg/.dxf and .asm/.prt files. Other file extensions like .model/.dlv/.exp, .iges/.igs or .step were also used. One company stored data centrally, one in distributed systems and the third used both methods.

Two companies possessed more than 250,000 3D-CAD models, and the third had more than 100,000 but less than 250,000 of these models. The average size of a 3D-CAD model was between 1-50 MB in two cases and between 100-250 MB for the third.

CAD data is sent frequently (daily or at least every 2-3 days) to subcontractors or customers. To facilitate the sending of data, one company uses ZIP compression and one PDF conversions.

In all surveyed companies, engineers are allowed to copy and rename existing CAD models, and make changes to the existing model. Common modifications include revision of fixings, design modifications, and feature changes.

Watermarking requirements

None of the surveyed companies had any prior experience with watermarking systems. The companies saw the main use of watermarking for identifying

ownership, copyright and the original data source but also for monitoring access. Tracking lost data and storing product information were considered the least important applications of watermarking. Companies were particularly interested in storing text and numbers with the help of watermarking. All companies wanted to see the watermarked information displayed in front of the model. Respondents perceived watermarking to be most useful when sending models to subcontractors or storing models in a database. Other applications like selling or data reuse were considered less important.

The single most important requirement for a watermarking scheme was the ability to survive modifications of the original model. Other requirements like robustness, non-interference with the model's capabilities and detectability by authorised sources were also considered important.

Summary of Questionnaire Results

Number of completed questionnaires: 3
Response rate: 33%

1. What kind of CAD system/s are you using at your company? Please tick all that apply.

AutoCAD	3 ¹¹	IronCAD	•	Solid Edge	•
Catia V4/V5	2	Mechanical Desktop	1	Solid Works	•
CADKEY	•	Microstation	•	Think Design	•
I-Deas	1	Pro/ENGINEER	2	Unigraphics	1
Rhino-3D	•	Autodesk Inventor	•		
Other: CADD5, CADCAM, STHEN02D					

2. In what form are the data/models stored at you company?

CAD Format	File Extensions	
ACIS SAT	.sat	•
Autodesk Inventor	.ipt, .iam	•
CATIA	.model, .dlv, .exp	1
DWG/DXF	.dwg, .dxf	2
IGES	.iges, .igs	1
Parasolid	.x_t	•
Pro/Engineer	.asm, .prt	2
PRO/DESKTOP	.des	•
Rhino-3D/Open NURBS	.3dm	•
SolidEdge	.asm, .prt	•
SolidWorks	.sldprt, .sldasm	•
STEP	.step	1
VDA-FS	.vda	•
Alias	.wire	•
IronCAD	.ics	•
Raindrop Geomagic	.wrp	•
Other: CADD5 (no extension – UNIX-based), UG .prt, CATIA 5 .catpart etc, IDEAS, CADAM native format ‘nurestor’, .tif		

3. What kind of data needs to be watermarked? Please tick all that apply.

Word documents	•
Text any kind	•
Documentations	•
2D CAD drawings	1
3D CAD models	2

¹¹ Figure indicates how often item was ticked.

Other: _____

4. What kind of information is meant to be hidden with the help of watermarking?

Text, Numbers 2

Image, Logo •

Other: _____

5. How and where should the watermark information be presented?

In front of the model (see image) 2

In a separate dialog window •

On call/separate function •

CAD system Plug-in •

Other: _____

6. Please rank the importance of watermarking for your purpose.

(1 = very important, 5 = not important)

Identify the source, owner, creator and distributor 1__

Identify the original data 3__2

Copyright notification 1__

Prevent or avoid theft or illegal use 3__

Restrict or regulate access to the (original) data 4__1

Monitor the distribution of data 4__2

Track lost data 5__

Use it as evidence in the court 3__

Add product/project/component information 4__

Other: _____

7. Which characteristic should the watermark have?

General 2 (the same identifier pattern for each data be watermarked)

Fingerprint • (unique identifier of each individual item, but a database is needed for storage of these fingerprints)

8. Please rank the requirements which need to be met by the watermark.

(1 = very important, 5 = not important)

Difficult or impossible to remove 2

Watermark must survive modifications 1

Should not interfere or influence the models capabilities 2

Readily detectable by the proper authorities 2

Other: _____

9. Please rank the application of the watermarked data.

(1 = very important, 5 = not important)

Stored in a database 2____
Used by subcontractors 1____
For sale 4____
Reuse data 5____
Other: To mark the approval status of drawings

10. Is the CAD data at you company stored in a central database or distributed to several databases in different departments?

Central database 2
Distributed systems 2
Other: _____

11. How many 3D-CAD models do you have at your company?

less than 10,000 10,00 to 50,000 50,000 to 100,000 100,000 to 250,000 more than 250,000
• • • 1 2

12. What is the average size of a 3D-CAD model at your company?

up to 500 KB 500-1000 KB 1-50 MB 50-100 MB 100-250 MB 250-500 MB more than 500 MB
• • 2 • 1 • •

13. What is the percentage of planar and cylindrical surfaces at your CAD models?

planar: up to 10% 10-20% 20-30% 30-40% 40-50% 50-60% 60-70% 70-80% more than 80%
cylindrical: up to 10% 10-20% 20-30% 30-40% 40-50% 50-60% 60-70% 70-80% more than 80%

14. How often do you send out CAD data to subcontractors or for sale, and to what percentage?

up to 10% 10-20% 20-30% 30-40% 40-50% more than 50%
every day 2 • • • • •
every 2-3 days • 1 • • • •
every 4-6 days • 1 • • • •
once a week • 1 • • • •
less frequent 1 • • • • •

15. For data and information exchange do you use data compression to limit file sizes?

No use of data/information compression 1
Use of ZIP compression/decompression 1
Use of PDF file format 1
Other: _____

16. For data and information exchange do you use any kind of coding?

No 1

Yes 2

If yes, please state the kind of coding you use: IGES, STEP, STL

17. Are engineers allowed to copy and rename existing CAD models and components?

Yes 3

No •

18. If your engineers are allowed to reuse existing CAD models, what are the common modifications?

Blending of holes and edges •

Remove parts of the model •

Adding new holes and parts 1

Other: Changed aerofoils, revised fixings, coatings, manufacturing alleviations etc., to create new parts or designs or modify existing parts or design, change of several features

19. Do you have any experience with one or more of the following Digital Rights management (DRM) technologies? Please tick all that apply.

- | | |
|---------------------|-----------------------------------|
| Encrypting • | Marking • |
| Watermarking • | Fingerprinting • |
| Steganography • | Labelling • |
| Digital Signature • | Digital Object Identifier (DOI) • |
| Message Digest • | Illustration Watermarking • |
| Other: _____ | |

No experience at all 2

20. What do you expect from digital watermarking of 3D CAD data?

Visible mark of data to identify release status or lack of: e.g. "NOT FOR MANUFACTURE" on a model that has not been fully approved

21. Could you provide us with real world CAD data for watermark testing?

Yes •

No 2

If yes, please insert here your or the relevant person's e-mail address: