Towards Blind and Robust
Perceptual Watermarking
of 3D Objects

Thesis of
Massimiliano Corsini

Tutors:
Prof. Vito Cappellini
Prof. Alberto Del Bimbo
Prof. Mauro Barni

PhD Coordinator:
Prof. Giacomo Bucci
Acknowledgements

First of all, I would like to thank very much Prof. Mauro Barni of the Department of Information Engineering of the University of Siena for his continuous support during my research and for reading and reviewing the thesis.

I am grateful to Prof. Touradj Ebrahimi from the Signal Processing Institute (STI) of the Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, for his suggestions and interest in the development of new techniques for perceptual quality evaluation of watermarked 3D objects and for giving me the opportunity of a seven-month research period at the EPFL. I would like to thank very much Elisa Drelie Gelasca for her great help in the development of such research.

I would also like to thank Edoarda for her patient and love, especially during the final drafting of this dissertation.

Above all, I want to express my deepest love and gratitude to my parents and my grandmother Maria for supporting me during all the years of my studies.
A Franco Bartolini
# Contents

1 Introduction

1.1 Summary of Contributions ...................................... 3
1.2 Organization of the Dissertation ................................. 3

I Digital Watermarking of 3D Objects ................................. 5

2 Digital Watermarking ................................................. 7

2.1 Introduction .......................................................... 7
2.1.1 History ............................................................... 8
2.1.2 Terminology .......................................................... 9
2.1.3 Requirements ......................................................... 9
2.1.4 Digital Watermarking Systems ................................. 12

2.2 Watermarking System Properties .................................... 15
2.2.1 Applications .......................................................... 20

3 Watermarking of 3D Objects ........................................... 25

3.1 3D Objects Representation .......................................... 26
3.1.1 Polygonal Meshes .................................................... 26
3.1.2 Implicit Surfaces ..................................................... 29
3.1.3 Parametric Surfaces ............................................... 30
3.1.4 Voxels ................................................................. 36
3.1.5 CSG ................................................................. 38

3.2 3D Watermarking ....................................................... 38
3.2.1 Peculiarities of Geometric Data ............................... 38
3.2.2 Watermark payload ............................................... 39
3.2.3 Imperceptibility ..................................................... 41
3.2.4 Robustness .......................................................... 42
3.3 State of the Art ............................. 47
  3.3.1 Asset Domain ...................... 49
  3.3.2 Transformed Domain ............. 59
  3.3.3 Hybrid Domain .................. 61
  3.3.4 Compressed Domain ............. 62
  3.3.5 Conclusions ..................... 63

II Wavelet-based Blind and Robust Watermarking
of 3D Objects ............................. 65

4 DGP Fundamentals .......................... 67
  4.1 Digital Geometry Processing ......... 67
  4.2 Subdivision Methods ................. 68
     4.2.1 Subdivision Surfaces .......... 70
     4.2.2 Subdivision schemes .......... 71
     4.2.3 Subdivision Analysis .......... 76
  4.3 Multiresolution Techniques .......... 78
  4.4 Mesh Parameterization ............... 83
     4.4.1 Differential Geometry Background 83
     4.4.2 Parameterization methods ... 85
     4.4.3 Remeshing ................... 88

5 Blind and robust watermarking of semi-regular meshes 95
  5.1 Introduction .......................... 95
  5.2 Wavelet Analysis/Synthesis ......... 97
  5.3 Synchronization ..................... 99
  5.4 Embedding ................................ 100
     5.4.1 Watermarking Maps .......... 102
     5.4.2 Watermark casting .......... 103
  5.5 Detection ................................ 105
     5.5.1 Statistical Decision Theory 106
     5.5.2 Threshold Computation ...... 109
  5.6 Experimental Results .................. 113
     5.6.1 Watermark Perceptibility 113
     5.6.2 False alarm rate .......... 115
     5.6.3 Robustness .................. 119
## Contents

5.7 Conclusions and Further Improvements .................................. 124

III Visual Quality Assessment of Watermarked 3D Objects 127

6 Visual Appearance of 3D Models 129

6.1 Rendering Techniques .................................................. 129
6.1.1 The rendering pipeline ............................................. 130
6.1.2 Ray Tracing .......................................................... 132
6.2 Illumination Models ...................................................... 133
6.2.1 Light Sources ......................................................... 134
6.2.2 Phong Model ......................................................... 136
6.2.3 Cook and Torrance .................................................... 136
6.2.4 BRDF ................................................................. 138
6.2.5 Basic Shading Techniques ......................................... 140
6.3 Texturing ................................................................. 141
6.3.1 Texture Mapping ..................................................... 142
6.3.2 Image texturing ...................................................... 143
6.3.3 Bump Mapping ....................................................... 143
6.3.4 Gloss Mapping ....................................................... 144
6.4 Global Illumination ........................................................ 144
6.5 Non-Photorealistic Rendering ............................................. 148
6.6 Final Considerations ...................................................... 149

7 3D Watermarking Quality Assessment 153

7.1 Introduction ............................................................. 153
7.2 Related Work ............................................................ 154
7.2.1 Perceptual Image Watermarking .................................. 154
7.2.2 Mesh simplification .................................................. 155
7.2.3 Perceptually-Guided Rendering .................................... 158
7.2.4 Our Approach ......................................................... 159
7.3 3D watermarking algorithms and artifacts ............................. 159
7.4 Experimental Method ..................................................... 161
7.4.1 Rendering Conditions .............................................. 161
7.4.2 Experimental Procedure .......................................... 166
7.4.3 Experiment I .......................................................... 169
7.4.4 Generation of Stimuli .............................................. 170
Chapter 1

Introduction

This thesis treats a new challenge in Digital Watermarking of multimedia data: *watermarking of 3D objects* or, more simply, *3D watermarking*. In the last two decades multimedia data such as audio, images and video have reached an enormous diffusion in several applications. Such enormous diffusion and the explosion of Internet peer-to-peer programs have increased the importance of Digital Watermarking technology, in particular for Intellectual Property Rights (IPR) purposes. Many researchers have contribute to develop reliable technologies to protect and authenticate multimedia data by indissolubly associate information within the data itself. Such information may regard the author(s) of the digital good, the legitimate owner(s), or the media itself. In particular, certain watermarking technologies, for example for copyright protection of still images or in the field of video broadcasting are now mature for the real world application as demonstrated by the recent “Hollywood case” [1] where a content protection system based on watermarking used by the Academy Awards has made possible to track back the Internet diffusion of illegal copies of nominated film before the Oscar Night, permitting to identify the involved members of the Academy.

Nowadays, a new kind of multimedia data has reached the analogous diffusion of audio, still images and video: *geometric data*. Architecture, Design, Mechanical Engineering, Entertainment and Cultural Heritage are some of the main important areas in which 3D models are widely applied. Often, the creations of a 3D model, in particular in certain fields, like in Cultural Heritage, require a lot of resources in terms of time and costs. The Digital Michelangelo Project [2] by the University of Stanford, for example, has required 30 nights of scanning and 22 people only to acquire an high-quality 3D model of the David of Michelangelo (Figure 1.1) and about 1 year to create a complete archive of the principal statues and
architecture of the great artist. These two factors strongly motivate the demand of reliable watermarking methods for 3D objects. Relatively recent, researchers in Digital Watermarking have moved their attention to this problem, and some algorithms to embed information within geometric data have been developed. In this field we have concentrated our effort in the development of blind and robust algorithms for watermarking of 3D objects. Such decision relies on the fact that the “absence” of such kind of techniques is one of the main weak point in the current panorama of 3D watermarking. In fact, with minor exceptions, most of the existing 3D watermarking algorithms require both the original 3D model and the watermarked one in order to recover the embedded information, i.e. most of the existing techniques are not blind. Moreover, most 3D watermarking algorithms are not able to extract the embedded information in presence of model alterations, i.e. they are not robust. Nevertheless, blindness and robustness are two desirable properties for any watermarking system. In fact, this kind of techniques are particularly suitable for important real world application, like copyright protection. The results of our research activities in this direction are reported in the second part of this dissertation.

In general, watermarking introduces distortions into the watermarked media. In order to preserve the value of the digital good it is important that such distortions would be imperceptible. For example, in image watermarking, the knowledge of the Human Visual Systems (HVS) are exploited in order to guarantee that the image modifications made by the watermarking algorithm result imperceptible to
the human eye. *Imperceptibility* is a crucial aspect in 3D watermarking. In fact, typically, the model is used to be visualized. Furthermore, it is very important that the watermarked model looks like the original one. Despite this importance, from our knowledge, no specific studies about the perceptibility of geometric defects caused by 3D watermarking over the model’s surface have been conducted since now. The goal of this research work is also to begin such kind of studies. In particular, the third part of this thesis is dedicated to 3D watermarking quality assessment. Such investigations have been conducted at the Signal Processing Institute (ITS) of the Ecole Politechnique Fédérale de Lausanne (EPFL), under the supervision of Prof. Touradj Ebrahimi, in collaboration with the PhD Student Elisa Drelie Gelasca.

1.1 Summary of Contributions

The main contributions of this work can be summarized as the following:

- A deeply analysis of the peculiarities of watermarking technology for 3D objects
- A detailed review of the current State-of-the-Art in 3D watermarking
- A new blind and robust multi-resolution algorithm for watermarking of 3D objects represented as semi-regular meshes
- A new experimental methodology for collecting subjective data for 3D watermarking quality assessment
- Two new perceptual metrics for impairment evaluation of 3D watermarking

1.2 Organization of the Dissertation

This thesis is organized in three parts. In the first one the general notions about Digital Watermarking are presented, analyzing in depth the peculiarities of watermarking technology for 3D objects. In the second part our new 3D watermarking algorithm is described. Finally, the third part is dedicated to the quality assessment of watermarked 3D objects. In particular, a novel experimental methodology for collecting subjective data for quality assessment of 3D objects is proposed and two new perceptual metrics to predict distortions introduced by 3D watermarking algorithms over the models’ surface are presented.
Part I

Digital Watermarking of 3D Objects
Chapter 2

Digital Watermarking

2.1 Introduction

In the last 50 years data hiding technology has received great interest from the information and telecommunication research communities. The effort spent in these years has been motivated by the great number of applications that data hiding technology encompasses and by the increasing diffusion of digital data, mainly thanks to Internet. One of the main application of data hiding is copyright protection. Data hiding used in this way is usually termed as digital watermarking. More precisely digital watermarking regards the embedding of a piece of digital information, called watermark in a host digital asset, such as audio, image or video. Digital watermarking is a multi-disciplinary field, in fact it is based on concepts belonging to cryptography, steganography, information theory, communication theory, and signal processing. The basic idea of watermarking is to associate a piece of information to the digital asset in a persistent way, i.e. not using a header but by modifying some properties, some features of the digital media. In this way the associated data are indissolubly tied to the digital host asset. This association can be used for several different applications. For example, in the case of copyright protection, the watermark conveys data about the owner/creator of the digital media, and it can be used to proof ownership in legal dispute. Depending on the data embedded other kinds of applications are possible, such as authentication, tracking content distribution, annotations, linking content, and so on.

In the following we provide a detailed description of digital watermarking and its applications. We start with some history about data hiding, continuing with some terminology issues, and then presenting the basic requirements of any wa-
termarking system. After this, the fundamental components of a generic water-
marking system will be described, along with the properties that characterize the 
watermarking systems. At the end, an overview of the applications of digital 
watermarking will be presented. The most recent challenge of watermarking tech-
nology, i.e. digital watermarking of 3D objects or 3D watermarking, will be deeply 
analyzed in the next Chapter.

2.1.1 History

The concept of data hiding has been known from the beginning of human his-
tories. One example can be found from the histories of Herodotus, about 2500 
years ago; the noble Histiaeus, regent of the city of Aristagoras, shaved the head 
of his slaves and tattooed it with a message which disappeared after the hair has 
regrown. These slaves were used to communicate with the city of Miletus: upon 
their arriving, after shaved, their head revealed the message of Histiaeus. Another 
example comes from Demaratus, a Greek at the Persian court, that warned Sparta 
about the intention of invasion by the king Xerxes by hiding the messages in a 
writing tablet. The method of Demaratus consisted in removing the wax from 
the writing tablet, wrote the messages on the wood, and then cover the message 
with the wax again. The tablet looked exactly like a blank one and could ar-
rive in Greece. In 1954, for the first time the term “electronic watermarking” 
was used, Emil Hembrooke of the Muzac Corporation filed a patent [3] in which 
he described a method for imperceptibly embedding an identification code into 
music for the purpose of proving ownership. This patent states: “The present 
invention makes possible the positive identification of the origin of a musical pre-
sentation and thereby constitutes an effective means of preventing such piracy, 
i.e. it can be likened to a watermark in paper”. The analogy between paper 
watermarks, steganography, and digital watermarking, in particular paper water-
marks in money bills or stamps, inspired the use of the term watermarking in 
the context of digital data. Tirkel et al. [4] coined the word “water mark” which 
become “watermark” later. From 1993 to 1996 digital watermarking began to re-
cieve remarkable attention from the research community. Since then, such interest 
has increasing continuously: after 1998, more than 250 research papers in digital 
watermarking are produced every year.
2.1.2 Terminology

Since digital watermarking technology shares concepts and is based on similar principles of other data hiding techniques, it is important to underline the distinction of watermarking with respect to other disciplines such as *steganography*. Steganography is a term derived from the Greek words *steganos*, which means “covered”, and *graphia*, which means “writing”. In other words *steganography* is the art to communicate a message secretly by hiding the communication in another message. The examples of Histiaeus and Demeratus are examples of steganography. The main difference between steganography and watermarking is that the watermark data has to be robust against *attacks*, i.e. manipulations of the watermarked asset. There are two kinds of attacks possible on a watermarking asset, the malicious ones and the non-malicious ones. All the manipulations of the watermarking asset are not explicitly aimed at removing the watermark are termed as *non-malicious attacks*. An example of a non-malicious attack for still images is JPEG compression. In fact, an user have to be allowed to compress his images without taking in account if such images are watermarked or not. On the contrary *malicious* attacks are performed by hackers with the purpose to destroy or reproduce the watermark. The term *data hiding* is used in various context and can be valid to indicate both steganography and watermarking applications even if it is most used in the context of steganography.

2.1.3 Requirements

All watermarking system must cope with three basic requirements: *capacity*, *robustness*, and *imperceptibility*.

**Capacity**

The capacity, or to-better-say the payload, of a watermarking system is the amount of information, measured in bits, that the watermark carries on. The amount of information indicates the number of different watermarks that the system is able to insert in the host asset. In particular, if the watermark carries $N$ bits, $2^N$ different watermarks are possible. It is important to remark that the capacity of the watermark does not depend only on the watermarking algorithm but even on the characteristics of the host asset. The requested capacity can vary a lot by changing the applications. For example, applications like captioning or labelling, require several thousands of bits of capacity, while security-oriented applications
such as copyright protection and fingerprinting, require that only few tens of bits are embedded in the host data. In general, high capacity is obtained at the expense of robustness and/or imperceptibility. This trade-off between capacity, robustness and imperceptibility is a constant of all watermarking algorithms and it is mandatory to find the best balance between these conflicting requirements after the application context of a certain techniques is specified.

Robustness

Robustness is the ability of the watermarking system to recover the watermark in the presence of modifications of the watermarked asset. As stated previously, these manipulations can have a malicious intent, if the purpose of the people who alter the watermarked media was to remove the watermark, or could be non-intentional. The robustness constraints depend by the specific application. According to [5], it is possible to consider 4 levels of robustness:

Secure watermarking. Applications that require secure watermarking mainly deal with copyright protections, ownership verification and so on. In secure watermarking the loss of watermark should be obtainable only at the expense of a significant degradation of the watermarked media. In this case the watermark must survive both malicious and non-malicious manipulations. When considering malicious manipulations it has to be assumed that the attackers know the watermarking algorithm and so they can conceive ad-hoc removal strategies. Non-malicious manipulations include a wide variety of digital and analog processing tools, such as lossy compression, cropping, editing, scaling, linear and non-linear filtering, D/A and A/D conversions, noise addition and many others. Usually in image watermarking the following attacks are considered: zoom, shrink, rotations, contrast enhancement, histogram manipulations, row/column removal and/or exchange. In the case of video watermarking instead: frame removal, frame exchange, temporal fil-
tering and temporal resampling. It is important to underline that the most secure system does not need to be perfect, but it is only needed that a high enough degree of security is reached. In other words, watermark breaking probably will be always possible, the aim of secure watermarking is to make this task very difficult.

Robust watermarking. In this case it is required that the watermark be resistant only against non-malicious manipulations. Robust watermarking is less demanding than secure watermarking. The typical application scenario of robust watermarking is in all cases in which normal use of data requires several manipulations but at the same time it is unlikely that someone purposely manipulates the host data with the intention to remove the watermark. Even in copyright protection applications, the adoption of robust watermarking instead than secure watermarking may be considered due to the use of a copyright protection protocol in which all the involved actors are not interested in removing the watermark.

Semi-fragile watermarking. In general, the watermark is said to be semi-fragile if it has to survives only a limited, well-specified, set of manipulations. This is the case, for example, of data labelling for improved archival retrieval. In this application the watermark is only needed to retrieve the host data from an archive, and thereby, it can be discarded once the data has been correctly accessed. It is likely though, that the data are archived in a compressed format, and that the watermark is embedded prior to compression. In this case, the watermark needs to be robust against lossy-coding.

Fragile watermarking. In this case the watermark is lost as soon as any modification is applied to the host asset. The watermark loss can be global, if all the embedded data are compromised, or local, if a part of the watermark can be still recovered. Usually, fragile watermarking is used for authentication: the loss or alteration of the watermark is taken as evidence that the watermarked data has been modified, and so, the asset is not authentic.

Imperceptibility

This basic requirement concern the quality of the watermarked asset. The data embedding process should not introduce any perceptible artifacts. Usually this is accomplished by taking into account the characteristics of the Human Visual System (HVS), in the case of image and video watermarking, or the characteristics
of the Human Auditory System (HAS), in the case of audio watermarking. For example, in image watermarking, the watermark is usually embedded according to a visual mask in order to reduce the perception of the watermark. This visual mask is computed according to a mathematical model of human visual perception. This point is crucial in 3D watermarking, since until now no studies about the perception of distortions on geometric surfaces have been carried out. The third part of this thesis is dedicated to this topic. In this part the perceptual issues of 3D watermarking are deeply analyzed and a novel perceptual metric to predict the visual distortions introduced by 3D watermarking algorithms is proposed. Usually imperceptibility involves a trade-off with robustness (see Figure 2.1). The imperceptibility of the embedded data is another difference between watermarking and steganography, in fact while in watermarking the imperceptibility is required also to preserve the quality of the watermarked asset, otherwise the value of the digital media are not increased by the watermarking process, but compromised; in steganography the imperceptibility is required to make the stego-asset unsuspicous, otherwise, people who have interest in detecting the stego-messages can try to analyze whose stego-assets that seems more probably containing hide information.

2.1.4 Digital Watermarking Systems

A digital watermarking system is typically formed by three distinct components, an information coding component in which the information to be embedded ($I$) is coded in a more suitable way forming the watermark $w$, a second component that modifies the host digital asset ($A$) in order to embed the watermark, and a third one, the detector, which extracts the embedded information from the watermarked asset ($A_w$). These three components can be described by three corresponding functions.

The information coding component can be modelled by the following function:

$$w = f(I, K)$$

where $I = \{I_1, I_2, \ldots, I_n\}$ is the information to embedded in form of binary string, $K$ is a key and $w$ is the returned watermark. The key $K$ is used as an optional password for user-restriction purposes; in fact, if the watermark depends both on the information and the key $K$, only the owner of the key is able to recover the embedded data. This is accomplished during the recovery phase, where, if another key is used, the detector attempts to recover a different watermark producing
incorrect results. Usually \( w \) is a sequence of values \( w = \{w_1, w_2, \ldots, w_n\} \) more suitable for embedding than the sequence \( I \). To be more clear considering a digital communication system, the information \( I \) may be used to modulate a much longer spread-spectrum sequence; in this case \( I \) may be transformed into a bipolar signal composed by +1 and -1 values; i.e. \( w = \{+1, -1, -1, +1, \ldots, +1\} \). In some special cases \( f(.) \) could be the identity function \( (w = I) \), in other words the information bits string is directly embedded in the asset.

The second component of a generic watermarking system, i.e. the embedding component, can be written as:

\[
A_w = \mathcal{E}(A, w)
\]

(2.2)

Often the embedding function \( \mathcal{E}(.) \) does not work directly on the asset data but, instead, on a set of features extracted from the original asset. In this case, denoting with \( f_A \) the host features of the asset \( A \), the watermark embedding can be expressed as:

\[
f_{A_w} = \mathcal{F}(A_w) = \mathcal{E}(f_A, w) = \mathcal{E}(\mathcal{F}(A), w)
\]

(2.3)

where \( \mathcal{F}(.) \) is a feature extraction function and \( f_{A_w} \) are the watermarked host features. In other words that are two ways to achieve the embedding process. According to (2.2) embedding is obtained by modifying the host asset \( A \) so that when the feature extraction function \( \mathcal{F}(.) \) is applied to \( A_w \), the desired set of features \( f_{A_w} = \{f_{1,w}, f_{2,w}, \ldots, f_{n,w}\} \) is obtained. In the second case, described by the equation (2.3), first the host feature of \( A \) are extracted, then the watermark is embedded by modifying \( f_A \), and, finally, the watermarked asset is constructed by inverting the feature extraction process \( \mathcal{F}(.) \):

\[
A_w = \mathcal{F}^{-1}(f_{A_w})
\]

(2.4)

This watermarking scheme is said \emph{watermark embedding via invertible feature extraction}. It is important to underline that the invertibility of \( \mathcal{F}(.) \) may be relaxed by assuring that \( A_w \) can be constructed from \( f_{A_w} \) and \( A \):

\[
A_w = \mathcal{F}^{-1}(A, f_{A_w})
\]

(2.5)

In this case the feature extraction process is said \emph{weak invertible}. As a simple example of a feature-based, weak invertible watermarking system, consider an image watermarking algorithm based on the Discrete Fourier Transform (Figure 2.2). In this simple system the image is watermarked by modifying the magnitude of DFT coefficients. So, the Discrete Fourier Transform works as the feature extraction
function $F(\cdot)$. After the DFT coefficients are watermarked the watermarked image is obtained by the Inverse Discrete Fourier Transform (IDFT). The IDFT, in this specific case, represents a weak invertible feature extraction function, since it requires not only the watermarked host features, i.e. the magnitude of the DFT coefficients, but even the phase of the DFT coefficients, that can be computed from the original asset $A$.

The third component of any watermarking system is the detector. The detector can be characterized by several properties depending on the way the watermark is recovered. For example the detector can be able to read the embedded information or only to verify the presence of a given watermark. In the former case we are in the presence of readable watermarking, while the last case is referred as detectable watermarking. Another important property that characterizes the watermark recovery is blindness. A blind detector does not need the original asset to recover the watermark. Said in another manner, a non-blind detector requires the original asset $A$ to extract the watermark. Figure 2.3 summarizes all of these possibilities. The characterization of the watermarking recovery phase can be highlighted by...
considering the different form assumed by the detection function $D(.)$. In the case of readable watermarking the detection function can be written as:

$$D(A_w, K) = I$$  \hspace{1cm} (2.6)

for the blind case and

$$D(A, A_w, K) = I$$  \hspace{1cm} (2.7)

for the non-blind case. Note that in readable watermarking, the decoding process always results in a bitstream of bits even if the input asset $A_w$ is not marked. Obviously, in such case the result of the decoding has no meaning. In the case of detectable watermarking the detection function becomes:

$$D(A, K, I) = \{\text{yes, no}\}$$  \hspace{1cm} (2.8)

for the blind detector and

$$D(A, A_w, K, I) = \{\text{yes, no}\}$$  \hspace{1cm} (2.9)

for the non-blind case. As in equation (2.1) $K$ is the optional security key.

### 2.2 Watermarking System Properties

Any digital watermarking system is characterized by several properties. The overall properties of a specific watermarking system determine the context for which a certain algorithm is or is not suitable. Some of such properties regard the way the watermark is recovered (readability, detectability, blindness), the information carried by the watermark (public and private watermark), if the watermark can be suitable for IPR protection (invertibility, quasi-invertibility), if the system is capable to manage multiple watermarks in the same asset and many others. In the next the most important properties of a watermarking system will be described in detail.

**Readable vs Detectable** A watermarking system can be able to read the information inserted in the host media, or verifies if a certain information is present or not. In the former case we have a readable watermarking system, and the recovery of the watermark is stated as watermark decoding. In the case of detection, instead, the recovery process is indicated with watermark detection. The detectable watermarking is also known as 1-bit watermarking, since the detector provides only a yes-no answer. More precisely the amount
of information carried by the watermark in the detectable case depending by the number of different watermark the system is able to distinguish; said $N$ the number of different watermark the capacity of the information is $\log_2(N)$. It is not surprising that detectable watermarking is less useful than readable watermarking for practical applications.

**Blind vs Non-blind** The watermark recovery can be accomplished in two distinct way; by comparing the watermarked asset data with the original asset, or by processing the watermarked asset alone. This latter case is called blind recovery, while the former case non-blind recovery. In other words a blind watermarking algorithms is able to extract the watermark information without the original data. Blindness is an important properties of a watermarking algorithm, in fact a non-blind watermarking system requires to store, for each watermarked asset, the original asset before the watermarking process, in order to make the recover of the watermark possible. Blind techniques do not have this necessity making them more useful for the real applications.

**Public vs Private** A watermark is said to be private if only authorized users can recover it. Privateness can be achieved by assigning to each user a different secret key, whose knowledge is necessary to extract the watermark from the watermarked asset. As previously stated this can be accomplished by an appropriate definition of the embedding and detection function. As the opposite, a public watermark carries on public information, for example; information about the asset like information usually contained in the headers, the copyright holder, the creator of the asset, and so on, in other words information that anyone can extract to know something about the watermarked digital content.

Private watermarking requires to be more robust than public watermarking for two reasons. The first one relies on the Kerkhoff’s principle that security must not be based on algorithm ignorance. So, if the watermark information can be easily extracted, assuming the attacker has a good knowledge of the watermarking algorithm, it is easier for him to remove the watermark or to make it unreadable. Another reason is that there are less motivations to eliminate public information from a digital content. Typically the capacity requirement is relaxed for private watermarking. On the contrary, for its intrinsic nature, the basic requirement of public watermarking is high
2.2. Watermarking System Properties

Multiple Embedding In some cases, the possibility to insert multiple watermarks inside the host asset is required. One example could be a copyright protection scheme in which two watermarks co-exist: one identifying the author of the work and another one indicating the name of the customer. Obviously, the detector must be able to read all the watermarks contained in the watermarked asset. In many cases, it is necessary to allow multiple insertions, in fact, if the insertion of a second watermark destroy the first watermark, any attempt to make such watermark robust fails since the pirates can use the watermarking embedding algorithm itself as an attack.

Invertibility and Quasi-Invertibility The concept of invertibility and quasi-invertibility arises analyzing the so-called SWICO [6] (Single-Watermarked-Image-Counterfeit-Original) attack, a simple attack that make possible to a pirate to exhibit a fake watermarked asset. Suppose that Alice creates an asset $A$ and protects it with her identification code $w_A$, producing the watermarked asset $A_{w_A} = A + w_A$. Then she makes $A_{w_A}$ publicly available. At this point, Bob, the pirate, watermarks with the same watermarking algorithm Alice’s asset inserting his own watermark, $w_B$, producing the asset $A_{w_Aw_B} = A + w_A + w_B$. Now this asset contains both the watermarks; to prove her ownership Alice can exhibit the asset containing her watermark but that does not contain the watermark of the other contender. But Bob can use the SWICO attack to provide the same demonstration of Alice. In fact, let’s assume, for instance, that the watermarking technique is not blind and that the watermark is revealed by subtracting the original asset. Alice can show that her asset contains her identification code and that even Bob’s asset contains her watermark:

$$A_{w_Aw_B} = A + w_A + w_B - A = w_A + w_B \quad (2.10)$$

$$A_{w_A} = A + w_A - A = w_A \quad (2.11)$$

The problem is that Bob can do the same by building a fake original asset $A_f$: it is sufficient that Bob subtracts his watermark from $A_{w_A}$, and consider the results as the original asset $A_f = A_{w_A} - w_B = A + w_A - w_B$. Thanks to this procedure Bob can prove that his asset contains $w_B$ but does not contain $w_A$:

$$A_{w_A} - A_f = A + w_A - (A + w_A - w_B) = w_B \quad (2.12)$$
So, the SWICO attack demonstrates that it is possible in some cases, to reverse engineering the watermarking process, i.e. there is the possibility to building a fake original asset and a fake watermark such that the insertion of the fake watermark in the fake original asset produces a watermarked asset equal to the initial one. These considerations are at the base of the definition of invertibility: a watermarking scheme is said to be invertible if for any asset $A$ exists a computationally feasible inverse mapping $E^{-1}(.)$ such that $E^{-1}(A) = \{A_f, w_f\}$, $E(A_f, w_f) = A$ and the asset $A$ and $A_f$ are perceptually similar. Otherwise the watermarking scheme is said to be non-invertible. In the simplified version of the SWICO attack presented above the inverse mapping is accomplished by:

$$E^{-1}(A) = \{A - w_f, w_f\}$$  \hspace{1cm} (2.13)

A more sophisticated version of the SWICO attack, the TWICO (Twin-Watermarked-Images-Counterfeit-Original) attack leads to the concept of quasi-invertibility. The extension of the SWICO attack to the TWICO one is obtained by noting that it is not needed that the fake asset and the fake watermark produce an asset that is identical to the original one but it is sufficient that the fake asset confuse the detector, i.e. $D(A, A_f, w_f) = \text{yes}$. In this context a watermarking scheme is said to be quasi-invertible if for any asset $A$ a computationally feasible inverse mapping $E^{-1}(.)$ exists such that $E^{-1} = \{A_f, w_f\}$, $D(A, A_f, w_f) = \text{yes}$ and the asset $A$ and $A_f$ are perceptually similar. Otherwise the watermarking scheme is said non-quasi-invertible. In the case of blind schemes the inversion problem reduces to finding a fake watermark $w_f$ such that its presence is revealed by the detector, i.e. $D(A, w_f) = \text{yes}$. So, blind watermarking has only one degree of freedom with respect to the two degrees of freedom of the non-blind techniques, thus making easier to prevent it by acting at a protocol level, i.e. by requiring that the watermarks are assigned by a Trusted Third Party (TTP) avoiding the use of ad-hoc watermarks. This observation underlines, that to be effective, watermarking technology by itself is not sufficient to prevent misuse unless a proper protection protocol is established.

**Reversibility** A watermark is said to be strict-sense reversible (SSR) if once it has been decoded/detected it is possible to completely remove it from the watermarked asset restoring exactly the original asset. This concept can be extended by saying that a watermark is wide-sense reversible (WSR) if
once it has been decoded/detected it can be made uncodable/undetectable without producing any perceptible distortion of the host asset. From the definitions of SSR and WSR it is clear that the first implies the second. Watermark reversibility complicates considerably the application protocols in the case that robustness/security of the embedded information is a major concern, for this reason it is suitable to carefully considered if our watermarking system has to have this properties.

Symmetric vs Asymmetric The decoding/detection process of a symmetric watermarking system uses the same set of parameters used in the embedding phase. This parameters depend on the specific watermarking algorithms, such as the number of the host features to watermark, the amount of modifications suffered by the host features, the optional secret key $K$, and so on. Such parameters can be considered all part of the secret key $K$. The knowledge of this set of parameters, in fact, can give to pirates enough information to remove the watermark from the host asset. So, symmetric watermarking can present security problems, especially in the case of consumer applications. Asymmetric watermarking systems try to overcome these problems by using two different keys, a secure, private key, $K_s$, used to embed the information, and another public key $K_p$, used to detect/decode the watermark. The final goal of asymmetric watermarking is to build these two keys to make impossible to deduce $K_s$ from $K_p$ and to remove the watermark.

Computational Cost It is important to consider the computational cost of the embedding and detection phase, in particular when the watermarking algorithm is intended to be used for commercial applications. Speed requirements are highly application dependent and often very different from the embedding and the extraction phase. For example, in fingerprinting video application, the watermark must be inserted in real-time in the video, but the real-time constrain is not necessary in the extraction phase. As the opposite, in the case of copy control systems (e.g. DVD copy control), the detection must be achieved in real-time on cheap hardware while the insertion can be done on high-cost professional equipments.

Another important issue related to computational costs is the scalability of the watermarking algorithm. Since the computational power of standard PC increases every year it can be very useful to design embedders/detectors with the capability to exploit all the power of a specific machine. In this
way, for example, the detector can take into account simple attacks when it works on a machine with low computational power and afford more complex attacks when it runs on a powerful machine.

### 2.2.1 Applications

In this section we give a panoramic of applications of digital watermarking. In fact, mainly used for IPR protection and data authentication, watermarking can be also used in many other practical situations such as transmission error recovery, annotation, covert communications, and so on.

**IPR protection**

The protection of Intellectual Property Right, or IPR protection is the very first targeted application of digital watermarking. This term includes the protection of the rights of the creator, the rights of the legitimate owner, copyright protection, moral rights protection (e.g. the integrity of the work in the respect of the moral beliefs of the creator), and so on. Three of the major tasks in IPR protection area are: demonstration of the ownership in legal disputes, fingerprinting, and copy control.

**Demonstration of rightful ownership** In this context the author of a work (e.g. a song, a picture, a movie) wishes to prove that he/she is the only legitimate owner of the work. To do so, a watermark identifying him/her unambiguously is embedded in the work. As previously stated, for this kind of application, it is necessary to use a watermarking algorithm that assures invertibility or non-quasi invertibility of the watermark. A common way to confer a legal value to the verification procedure through watermark detection is to introduce the presence of a Trusted Third Party (TTP) that assigns a unique registration code to the owner of the work in order to proof the ownership of the registered asset without ambiguity. In this case the main requirement of a watermarking system is secure robustness, in fact pirates are obviously interested in removing the watermark.

**Fingerprinting** In this case the watermark identifies the buyer of a digital content. This mechanism represents a deterrent against illegal copy of digital contents by discouraging people to make illegal copy of the watermarked content. In fact, potentially, it is possible to trace back the illegal copies to identify the hackers of the content. From a commercial point of view,
with the explosion of Internet and peer-to-peer downloading programs, this application is becoming one of the most attractive applications of digital watermarking. Even in this case the watermarking system must exhibit secure robustness. Watermark readability is also an useful property in this context.

**Copy control** When copy deterrence systems, such as fingerprinting and ownership demonstration, are not sufficient to protect legitimate right-holders an effective copy protection mechanism must be used. Watermarking technologies, in this context, is only an important part of the whole copy protection system. In fact, to make a copy protection system effective, a lot of different aspects and problems must be taken in account. For example, recording and playback devices should be designed to give them the capability to recover watermark information inside an asset, and, to decide whether to allow or not, the copy of such asset. A good example of this approach is the copy protection system being developed for DVD technology [7].

**Authentication**

In this application the watermark encodes information required to determine if a digital content is authentic. Imagine a journalist that takes a photo of a certain event; it is important to guarantee that such photo is authentic and that it has not been manipulated or altered in any way. For this purpose a (semi-)fragile watermark containing, for example, the information about when the photo was taken and by which camera, can be used. This information can be watermarked in the image by the digital camera itself. After this, if the watermark is correctly recovered, we have the guarantee that the asset is authentic, i.e. it has no suffered any kind of manipulations or alterations. Authentication can be achieved even by robust watermarking. In this case, the watermark carries a summary of the original work. After extraction, to prove data integrity, the summary is compared with the expected one, at this point any mismatch indicates that data tampering has occurred.

**Error recovery in multimedia transmission**

The transmission of data in compressed form, such as JPEG for still images, or MPEG-2 or H.263 for video, is very vulnerable to transmission errors. In particular, for video streaming, a single bit error can cause a loss of synchronization that will be visible over an entire group of frames (GOP). To circumvent this
problem a controlled amount of redundancy can be introduced at the transmission level in order to recover from errors. For example, one solution could be the use of error correcting codes. Watermarking could be a valid alternative to solve this problem: the redundant information is embedded within the compressed bitstream and used by the decoder to recover from transmission errors.

In this kind of application robustness requirement is relaxed since the watermark must survive no attacks except transmission errors. On the other hand, capacity assumes greater importance, since high capacity allows the transmission of a large amount of redundancy, thus resulting in excellent robustness against errors. The readability of the watermark is mandatory since, otherwise, it cannot be possible to correct transmission errors, but only to detect them. Blindness is also mandatory for obvious reasons.

Annotation

In this context the watermark conveys simple annotation or labelling data. Watermarking presents a lot of advantages with respect to conventional techniques to associate such data, for example by using an external database or a header. One of this advantages is the capability of the watermark to survive the digital to analog and analog to digital conversion. Other advantages will be clear in the next, when we describe a couple of examples just to give an idea of the potential of watermarking technology in this field. The main requirement for such applications is high capacity, while the robustness requirement is usually relaxed.

Labelling for data retrieval Content-based access to digital archives is receiving more and more attention since nowadays the efficient management of database of multimedia objects, such as images and video, has assumed great importance. Since it is very difficult to automatically analyze the semantic content of the digital objects and retrieve them in the database, a semantic description is associated to each object, typically through a header. The usefulness of watermarking with respect to conventional data labelling, in this context, can be caught by considering an archival of video sequences in MPEG-4 format. Imagine that each video object of the MPEG-4 stream is watermarked with its associated information. At this point, if a watermarked video object is edited to create a different video sequence its associated information are automatically copied with it avoiding the necessity of labelling it again. Similarly, if the object is pasted to a new video after going in analog
and back to digital domain, the annotation watermark is not lost, making the semantic description of the new video easier.

**Linking real objects to the digital world**  Another remarkable example concerns the association between a real object and the digital world. The Mediabridge™ [8] system developed by Digimarc Corporation is an example of such vision of watermarking. In this case the value of an image is augmented by embedding within it a piece of information that can be used to link the image to additional information stored on the Internet. For example, such an information can be used to link a picture on a newspaper to a web page further exploring the subject of the article. The embedded url is activated by showing the printed picture to a video camera connected to a PC.

**Covert communications**

One of the earliest application of watermarking is sending secret messages. This application has been modelled by Simmons [9] as the “prisoner’s problem” in which two prisoners want to communicate each other in order to run away from the prison. The problem is that they cannot send directly messages but the prisoner warden act as a messenger. The warden is willing to carry innocuous message but will punish them if he finds that, such messages contain information about escape-plan. The solution is to disguise the escape-plane messages as innocuous messages.

As previously stated this application is more correlated with steganography than watermarking. In fact, in steganography, imperceptibility assumes a wider sense, i.e. the presence of the hidden message cannot be revealed by any means, such as visual inspection, statistical analysis, etc. So, in this kind of application, the undetectability of watermark presence becomes a new requirement for the digital watermarking system.
Chapter 3

Watermarking of 3D Objects

The watermarking of 3D objects, usually referred to as 3D watermarking, is a new frontier of digital watermarking concerning the embedding of information into 3D objects in an imperceptible way. The interest on 3D watermarking technology is constantly growing in these last years due to the large diffusion reached by 3D models. In fact, geometric data are widely used in a large number of contexts, for example in mechanical engineering for virtual prototyping and simulation, in architecture to preview the final impact of a project, in Cultural Heritage to realize virtual museums and archeological sites reproduction, in scientific visualization, in entertainment industries for movies and video-games, in Internet to build virtual-world and for e-commerce applications, and so on. The explosion of this new multimedia data, has been supported both by the tremendous graphics power reached by the modern consumer graphics accelerator board and by the effort spent by the Computer Graphics Research community to develop efficient algorithms to build, edit, and compress 3D objects. Now, this trend has reached the digital watermarking community and the number of works related to 3D watermarking is increasing every year. In this Chapter we provide a deep analysis of 3D watermarking technology. First of all, we give a panoramic of the different kinds of representations of geometric data. In the second section we describe the peculiarity of 3D watermarking; in particular we analyze the intrinsic nature of geometric data, specifying why this kind of data is more difficult to manage with respect to other multimedia data such as audio, image and video. Then, we analyze the particular characterization of each watermarking system requirements, i.e. capacity, robustness and imperceptibility, in the case of 3D watermarking. After this, in the last section, we present a critical review of 3D watermarking algorithms.
3.1 3D Objects Representation

Basically, a 3D model is composed by a collection of geometric data, representing the shape of the model, plus other data defining the visual appearance of the model, such as surface properties, textures and colors. As other kind of digital media such as audio, image and video, 3D objects data can be represented in a lot of different ways. In this section we focus on geometric data representation, providing a panoramic of the most used representations and presenting the peculiarities of each. The visual appearance of the model will be discussed in the third part, where the perceptual issues related to 3D watermarking will be debated.

One first categorization of 3D objects representation can be done by considering whether the surface or the volume of the object is represented:

- **Boundary-based:** the surface of the 3D object is represented. This representation is also called *b-rep*. Polygonal meshes, implicit and parametric surfaces are common *b-rep* representations.

- **Volume-based:** the volume of the 3D object is represented. Voxels and Constructive Solid Geometry (CSG) are commonly used to represent volumetric data.

Usually the representation of a 3D model depend on the way the model has been created and by its application context. The typical applications of each representation will be provided in the following. Concerning 3D model generation a lot of different ways are possible; to give some examples, a 3D model can be generated by scanning a real 3D object, can by designed by hand using geometric modelling software, can be automatically obtained from the analysis of medical images, can be reconstructed from a set of images through the use of Computer Vision techniques, etc.

3.1.1 Polygonal Meshes

Intuitively, a 3D *polygonal mesh* is a collection of vertices, edges and faces in 3D space joined together to form the shape of the 3D object.

More precisely, a mesh $M$ can be seen as a tuple $(K, V)$ where $V = \{v_i \in \mathbb{R}^3 | i = 1 \ldots N_v\}$ is the set of the vertices of the model (points in $\mathbb{R}^3$) and $K$ is a set encoding adjacency information for vertices, edges and faces of the mesh. In particular $K$ is formed by subsets of $I = \{1, \ldots, N_v\}$ called *simplices*. The set $I$ contains the indices of the mesh vertices, i.e. at each index $i$ is associated
its corresponding vertex \( v_i \in V \). There are three types of simplices: vertices \( V = \{\{i\}|i \in I\} \), edges \( E = \{\{i,j\}|i,j \in I, \{i,j\} \text{ is an edge}\} \) and faces \( F = \{\{i,j,k\}|i,j,k \in I, \{i,j,k\} \text{ is a face}\} \). The set \( K \) is called simplicial complex and is defined as the union of the simplices, i.e. \( K = \cup E \cup F \). The geometric realization of a simplex \( s \in K \), denoted with \( \varphi(s) \), is the strictly convex hull of the vertices \( v_i \) with \( i \in s \). For example the geometric realization of an edge \( \{i,j\} \in E \) is the segment connecting the vertex \( v_i \) with the vertex \( v_j \), the realization of a face \( \{i,j,k\} \in F \) is the triangle defined by the vertices \( v_i, v_j \) and \( v_k \), and so on. The 3D model is the geometric realization of the mesh \( \varphi(K) \) defined as \( \bigcup_{s \in K} \varphi(s) \). A vertex \( v_i \) is said to be a neighbor of another vertex \( v_j \) if an edge exists that connects \( v_i \) and \( v_j \). The set of all the neighbors of a vertex \( v_i \) is called 1-ring of the vertex and is defined as \( v_1(i) = \{j|\{i,j\} \in E\} \). The cardinality of \( v_1(i) \) is called degree or valence of the vertex \( v_i \).

Generic meshes are rarely used in computer graphics, the most used meshes are triangular and quadrilateral ones. A triangular mesh is a mesh composed by triangles only, in the same manner a quadrilateral mesh is composed only by quadrilaterals. In the following, when we refer to a mesh, we always intend a triangular mesh. This assumption implies no loss of generality since every polygon of a non-triangular mesh can be triangulated to obtain a triangular mesh.

A sequence of adjacent triangles sharing the same vertex is called a fan of triangles. A strip is a sequence of triangles that can be specified by listing its vertices without ambiguity. To be more specific given an ordered vertex list \( \{v_0, v_1, \ldots, v_n\} \) the triangle \( i \) is represented by the vertices \( \{v_i, v_{i+1}, v_{i+2}\} \). Strips and fans are used to compact the mesh representations. In fact, considering a strip of triangles, given \( n \) vertices \( n - 2 \) triangles can be represented. So, a strip of 100 triangle requires 102 vertices to be stored instead of 300 as in the normal case. The amount of vertices saved increases with the number of triangles; the average number of vertices \( \overline{\tau} \) to represent a triangle in a strip with \( m \) triangles is \( \overline{\tau} = 1 + 2/m \). Fans of triangles have the same performances, i.e. the same average number of vertices per triangles. The difference between these two entities is mainly topological.

A mesh is said to be a 2-manifold, abbreviations of bi-dimensional manifold, if for every vertex \( v \) the faces incident on \( v \) are homeomorphic to a disk, or a semi-disk if \( v \) is on the boundary of the mesh. Considering the edges instead of the vertices it is possible to say that a mesh is a 2-manifold if each edge has two faces incident on it (or one if the edge is a boundary edge). Figure 3.2 shows an example of 2-manifold (right) and an example of a non-manifold (left).
A mesh is *orientable* if each of its faces is oriented in the same way. The *orientation* of a face is given by the order of the vertices that form it; in particular two faces $f_1$ and $f_2$ have consistent orientation if for each shared edge its vertices appears in opposite order in the description of $f_1$ and $f_2$ (see Figure 3.3). Conventionally, the mesh orientation is called clockwise or anti-clockwise. After assuming that a face has a clockwise (anti-clockwise) orientation, the face oriented in the opposite way has an anti-clockwise (clockwise) orientation.

Another important characterization of meshes is their *regularity*. A mesh is called *irregular* if its vertices can have any valence, *completely-regular* if all vertices have the same valence and *semi-regular* if most of its vertices have the same valence except a small number that can have any valence. This last definition arises because a semi-regular mesh is obtained by repeatedly subdividing an irregular mesh. During the subdivision process the irregular vertices of the initial mesh remain irregular while most of the newly inserted vertices converge to valence six (for a triangular semi-regular mesh). This classification is very important since regular meshes are at the basis of *digital geometry processing*. For this reason in the following we will give a detailed description of semi-regular meshes and other strictly related topics such as subdivision surface, mesh parameterization, and
remeshing. In particular, we will see how the properties of semi-regular meshes can be used to build a multi-resolution framework based on wavelet analysis; our 3D watermarking method is based on such framework.

Polygonal meshes present several limitations. First of all, since they are a discrete representation, curved surfaces can only be approximated; in fact each vertex can be seen as a sample of the curved surface the mesh represents. So, this representation is not compact, i.e. a high-detailed model requires a huge amount of data to be represented. Another problem is that direct editing is not easy, in fact, for this purpose, other representation are typically used, in particular parametric surface such as NURBS. Finally, there is no natural parameterization. Despite all of this problems the Computer Graphics Research community has put a lot of effort in mesh processing and a huge number of applications use this kind of representation. One of the main reasons of this is that meshes are the common denominator of the other representations, i.e. it is easy to convert other representation to this one. Another motivation is that modern graphics hardware are able to render thousand millions of triangles every seconds, in other words graphics accelerated board are designed to work with triangular meshes.

3.1.2 Implicit Surfaces

An implicit surface is defined as the set of three-dimensional points $S$ such that a given trivariate function $f(.)$ is equal to zero:

$$S = \{(x, y, z) \in \mathbb{R}^3 | f(x, y, z) = 0\}$$  \hspace{1cm} (3.1)

where $(x, y, z)$ are cartesian coordinates. The set $S$ is also known as the zero set of $f(.)$. According to the implicit surface theorem, if $f(.)$ in non-singular in zero, its zero set is a two-manifold. For example a sphere of radius $r$ can be represented by the equation $x^2 + y^2 + z^2 = r^2$, that becomes $x^2 + y^2 + z^2 - r^2 = 0$ in the canonical
form. A plane in the space can be represented by the function $ax + by + cz - d = 0$, and so on. So, a 3D object is represented in this way by a set of implicit surface functions each describing a part of its shape.

Algebraic surfaces are a particular kind of implicit surfaces for which $f(.)$ is a polynomial in three variables. The degree of an algebraic surface is given by the sum of the maximum powers of all terms $a_{m}x^{i_m}y^{j_m}z^{k_m}$ of the polynomial. An algebraic surface of degree two describes quadratic surfaces, a polynomial of degree three cubic surfaces, of degree four quartic surfaces and so on. Quadratic surfaces, also called quadrics, are very important in geometric modelling. This kind of surfaces intersect every plane in proper or degenerate way forming 17 standard-form types of surfaces [10]. To mention someone: parallel planes, ellipsoid, elliptic cone, elliptic cylinder, parabolic cylinder, hyperboloid of one sheet, hyperboloid of two sheets.

One of the main peculiarities of implicit representation is its compactness. Such representation is rarely used, since free-form surfaces are difficult to describe in this way. Moreover, parametric surfaces are traditionally preferred over implicit ones because they are easier to render and it is simpler to perform geometric operations such as curvature computation, tangent computation, and so on.

### 3.1.3 Parametric Surfaces

A three-dimensional parametric surface is defined by a mapping from the parameter domain (usually $\mathbb{R}^2$) to the Euclidean space $\mathbb{E}^3$. Such parameterization can be accomplished by three bivariate functions in the following way:

$$S(u, v) = (X(u, v), Y(u, v), Z(u, v))$$ (3.2)

where $u$ and $v$ are the surface parameters. Usually $u$ and $v$ range from 0 to 1.

Parametric curves or surfaces are characterized by two ingredients: a set of control points, and some blending functions used to join together the control points in a smoothing way. The blending functions define the properties of the final curve/surface such as continuity, differentiability, if the curve/surface is an approximation or an interpolation of the control points and so on. The typical expression of a parametric curve is:

$$C(u) = \sum_{i=0}^{n} P_i B_i(u)$$ (3.3)

where $P_i$ are the control points and $\{B_i(.)\}$ are the blending functions. Usually the parameter $u$ ranges from 0 to 1. The set of control points is usually called
control polygon. The equation (3.3) can be extended to the case of surfaces in several ways. The most used one is the tensor product surfaces defined as:

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_i(u) B_j(v)$$

(3.4)

where $P_{ij}$ are the initial control points and $\{B_i(.)\}$ and $\{B_j(.)\}$ are the blending functions. In this case the control points $P_{ij}$ are referred as the control net of the surface $S$. Tensor product surfaces are also called rectangular patches, since the domain of the parameter $(u, v)$ is a rectangle (in $\mathbb{R}^2$). Typically, such parameters range from 0 to 1.

Bernstein polynomials

A Bernstein polynomial of degree $n$ is defined as:

$$B_{i,n}(t) = \binom{n}{i} t^i (1 - t)^{n-i} \quad i = 0 \ldots n$$

(3.5)

where $\binom{n}{i}$ is the binomial coefficient, i.e. $\binom{n}{i} = \frac{n!}{i!(n-i)!}$.
Bernstein polynomials are widely used as blending functions for parametric curves and surfaces since polynomials are “easy” to treat. Furthermore the properties of Bernstein polynomials make them suitable for efficient implementation. The set of Bernstein polynomials of degree \( n \), i.e. \( \{ B_{0,n}(.), B_{1,n}(.), \ldots, B_{n,n}(.) \} \) forms a basis of the vector space of polynomials, called Bernstein basis. Figure 3.4 shows the Bernstein basis of degree 1 (top-left), 2 (top-right) and 3 (bottom).

### Bézier Curves and Surfaces

This kind of parametric curves is one of the most frequently used in computer graphics and was independently developed by two engineers both working for France automobile companies: Pierre Bézier, who was an engineer for Renault and Paul de Casteljau, who was an engineer for Citroën. The mathematical definition of a Bézier curve is:

\[
P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t) \quad 0 \leq t \leq 1
\]

where \( P_i \) are the control points and \( \{ B_{i,n}(.) \} \) are Bernstein polynomials of degree \( n \). According to the tensor product surface the definition of Bézier curves can be extended to surfaces in the following way:

\[
S(u, v) = \sum_{j=0}^{m} \sum_{i=0}^{n} P_{ij} B_{i,n}(u) B_{j,m}(v)
\]

where \( P_{ij} \) are the points of the control net, \( \{ B_{i,n}(.) \} \) are Bernstein polynomials of degree \( n \) and \( \{ B_{j,m}(.) \} \) are Bernstein polynomials of degree \( m \). Figure 3.5 shows an example of a bi-cubic Bézier patch. In this case the control net of the patch is formed by \( 4 \times 4 \) control points.

The Bézier patches can be assembled together to represent the shape of complex 3D objects. In Figure 3.6 an example is showed. The model represented in this example is the Utah teapot, a model of a teapot realized in 1975 by Martin Newell, a member of the pioneering graphics programme at University of Utah.
This simple, round, solid (i.e. not hollow), partially concave mathematical model has become a standard reference object (and something of an in-joke) in the computer graphics research community.

**B-Spline Curves**

The definition of a B-Spline curve of order \( k \) is:

\[
P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t)
\]

(3.8)

where \( n \) is the number of control points, \( N_{i,k}(t) \) are blending functions defined recursively by:

\[
N_{i,1}(t) = \begin{cases} 
1 & t \in [t_i, t_{i+1}) \\
0 & \text{otherwise}
\end{cases} 
\]

(3.9)

and, for \( k > 1 \):

\[
N_{i,k}(t) = \left( \frac{t - t_i}{t_{i+k-1} - t_i} \right) N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)
\]

(3.10)

where the set \( \{t_0, t_1, \ldots, t_{n+k}\} \) is a sequence of values referred as knots sequence. Such sequence influences the shape of the B-Spline. In particular, if the knots sequence is uniform, i.e. the knots values are equidistant, the B-Spline definition becomes \( N_{i+1,k}(t) = N_{i,k}(t - t_i) \), i.e. the blending functions moves along the sequence (see Figure 3.7). Note that the number of knots \( k \) determines the degree of the curve\(^1\) and not the number of control points as for Bézier curves. In other

\(^1\)An uniform B-Spline blending function \( N_{i,k}(.) \) is a piecewise \( k - 1 \) degree functions having support in the interval \([t_i, t_{i+k})\).
words B-Splines are local, to be more precise $N_{i,p}(t) \geq 0$, $t \in [t_i, t_{i+p+1})$, while the degree of a Bézier curve is constrained by the number of its relative control points. Additionally, Bézier curves must pass through their initial and final control points making the continuity between curves joined together more difficult to achieve than in the case of B-Splines. For these and other reasons B-Spline are more flexible than Bézier curves.

**NURBS surface**

The Non-Uniform Rational B-Splines, or NURBS, are the generalization of the non-rational B-Splines (3.8) just seen. Such generalization consists in the use of ratios of blending functions since polynomials are not sufficient to represent conic curves but the ratio of polynomials can parameterize them. The term non-uniform refers to the fact that the knots sequence is not uniform. So, a NURBS curve of order $k$ is defined as:

$$P(t) = \frac{\sum_{i=0}^{n} w_i P_i N_{i,k}(t)}{\sum_{i=0}^{n} w_i N_{i,k}(t)} \quad (3.11)$$
where \( n \) is the number of control points, \( P_i \) are the control points, \( \{N_{i,k}(t)\} \) are the same blending function of the B-Splines curves and \( w_i \) are weights used to alter the shape of the curve. In the same manner as Bézier patches we can extend the definition of NURBS curves to NURBS surfaces:

\[
S(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} P_{ij} N_{i,k}(u) N_{j,m}(v)}{\sum_{i=0}^{n} \sum_{j=0}^{m} w_{ij} N_{i,k}(u) N_{j,m}(v)} \quad (3.12)
\]

Thanks to the properties of B-Splines NURBS surfaces do not interpolate their control points, except at places where the knot multiplicities lead to \( C^0 \) continuity. Also the local control property still remains valid, i.e. the modification of a control point only affects the surface shape in its neighborhood. So, it is easy to control the shape of a large surface. For this reason NURBS surfaces are the base modelling tool of the most powerful and famous geometric modelling software such as Maya® and Rhino®. Figure 3.8 depicts an example of NURBS modelling.

Parametric surfaces representation is a very flexible representation of 3D objects with a lot of interesting properties; for example, parameterization is trivial, geometry processing is analytical, i.e. a tangent of the surface is computed by derivation of the parametric equation, they can be easily converted to other representations, and so on. The main limitation of this representation regards the difficulties to automatically generate a model composed by a set of parametric surfaces. In fact, as just mentioned, the most typical usage of parametric surfaces is in geometric modelling software, i.e. in the manual generation of 3D objects.
3.1.4 Voxels

This representation can be seen as the natural extension of two-dimensional images to the third dimension. In fact, a digital image is represented as a matrix of values where each element corresponds to a picture element, called pixel. A voxels is represented by a set of values disposed on a regular 3D grid where each element of this 3D matrix provides information about a volume element, called voxel (see Figure 3.9). A voxels can carry different type of information depending on the specific application the voxels is used in. Such information could be the density of the volume element, the temperature, the color, etc.

One of the main domains of application of the voxels is in medical imaging. A voxels can easily represent such kind of data by assigning to each organ a different value in the 3D grid to identify it. In Figure 3.10 an example of such application is showed.

Some of the advantages of voxels representation are: it is easy to represent volumetric data, inside/outside test is trivial, the extension of image processing algorithms to voxels is natural. Nevertheless voxels present even a lot of limitations. One of the main limitations is the huge amount of memory necessary to obtain a high-resolution representation. In fact, a grid of \( n \times n \times n \) elements requires \( n^3 \) bytes to be stored, if each volume element is represented by 1 byte. The problem is that the resolution of each cartesian coordinates is only \( 1/n \), so, for example, a voxels with a resolution of \( 1/128 \), that it is not very high, requires about 128 Megabytes to be stored. For analogous reasons, voxels processing is usually computationally expensive. Finally, it is difficult to visualize a voxels. One of the common way is to extract an iso-surface from the voxels, approximate it with a polygonal mesh and display the polygonal mesh. An iso-surface represents a boundary of the volume, such boundary could be extracted considering the values of interest of the volume elements. In particular, the function \( f(i, j, k) = c \), where \( i,j \) and \( k \) are the indices of the volume elements, represents an iso-surface of iso-value \( c \). After the extraction, the iso-surface can be converted in a mesh using an old algorithm being the standard for this kind of conversion called Marching Cubes [12]. Other more recent approaches to extract a mesh from volume data are [13, 14]. On the contrary the mesh-to-voxels conversion is difficult and computationally expensive. One of the last presented solutions about this topic is [15].
3.1. 3D Objects Representation

Figure 3.9: From pixels to voxels.

Figure 3.10: An example of voxels in medical imaging. The image has been produced by the VOXEL-MAN® system of the University of Hamburg, Department of Informatics in Medicine.

Figure 3.11: Constructive Solid Geometry. (Left) A CSG model (from [11], plate III.2). (Right) Example of a CSG tree.
3.1.5 CSG

The constructive solid geometry (or CSG) represents the volume of a 3D object by combining simple solid shapes called primitives with boolean operations such as union, difference, and intersection. Usually the model is stored as a binary tree where, the leaf nodes are the primitives, correctly sized and positioned, and each branch node is a Boolean operator (see Figure 3.11 for an example).

The primitives can be defined, similarly as the implicit surfaces, as the set of points that satisfy the equation \( f(.) < 0 \), where \( f(.) \) is an implicit surface function. From this definition the points such that \( f(.) > 0 \) are outside the volume bounded by the surface defined through \( f(.) \). This last fact is assumed conventionally.

The constructive solid geometry finds its main application field in CAD/CAM design where the exact geometric modelling of the parts of the modelled object (e.g. a mechanical piece) is required. Useful for such applications, talking in general CSG is not an efficient representation; the rendering must pass through the conversion in triangular meshes, edit complex shape it is difficult, and the compactness of the representation depends heavily on the detail of the represented model.

3.2 3D Watermarking

In the following, except otherwise specified, when we refer to a 3D object we assume a triangular mesh representation because this is the most used representation for 3D models. Such assumption will be more carefully motivated in the next Chapter, where we analyze our 3D watermarking algorithm and propose a complete blind and robust watermarking system for 3D objects.

3.2.1 Peculiarities of Geometric Data

Geometric data present a lot of particular aspects that make them “difficult” to handle with respect to other kinds of multimedia data. One of the first problems is that no implicit ordering exists for this kind of data. For example, changing the order of the samples of an audio song, contemporarily changes the music you listen. The same happens to both still images and video that have an implicit order of their samples given by their corresponding domains of definition. On the contrary vertices and faces of a polygonal mesh can be ordered only in an explicit way, i.e. by arbitrarily defining some kind of ordering, for example vertices can be ranked accordingly to a given direction in the 3D space. Another important
peculiarity of geometric data regards the properties of its samples. Audio, still images and video data have uniform, equi-spaced sampling on Euclidean domains; audio is a function of time over a 1D line, still images can be defined as a function over a bi-dimensional Euclidean domain and video over a section of $\mathbb{E}^3$, with two dimensions in space and one in time. The problem is that if the domain of definition is not Euclidean it is not possible to achieve uniform sampling. Additionally, geometric data are characterized by their topology. For these reasons the extension of the digital signal processing tools widely used, such as FFT and DCT, cannot be achieved in a natural way but presents several complications. Computer Graphics research community has put a lot of effort to overcome these difficulties and build new tools for geometric processing, which mimic as well as possible what has already been developed for signal processing. Peter Schröder and Wim Swelden were the first to use the term Digital Geometry Processing [16] to indicate this modern research area of Computer Graphics. Some fundamental notions about Digital Geometry Processing will be presented in the next Chapter since our approach to 3D watermarking follows the basic ideas proposed by this promising research field. The last peculiarity of 3D object data is that, given a particular shape, its mesh representation is not unique\(^2\); meshes with different number of vertices and faces, and completely different topology can represent the same shape with the same degree of approximation.

### 3.2.2 Watermark payload

The *payload* of a watermarking system is the amount of information bits that the watermark is able to convey. In general, watermark payload depends on the particular watermarking algorithm and it is related to the characteristic of the host data. This is also true for 3D watermarking but in this case a lot of important considerations arise.

First of all the amount of information available to host the watermark heavily depends on the representation of the 3D object. A simple example is given by the representation of a sphere. If a sphere is represented by its implicit function the information is given only by the coefficients of the polynomial implicit function that determine the position and the radius of the sphere. On the contrary, if the sphere is represented by a polygonal mesh, a huge amount of information is necessary to obtain a good degree of approximation. Even thousands of vertices

\(^2\)A given shape does not admit unique representation even if other kind of 3D data representations, such as parametric surfaces and CSG, are used.
could be necessary if we want a real good approximation of a sphere (see Figure 3.12). Hence, the amount of information conveyable by a polygonal mesh depends on the density of the samples of the surface. Furthermore, as previously noticed, the same number of vertices can represent the same shape and a less or high number of vertices, re-positioned, could represent the same shape with the same geometric precision. Also, the sampling of the surface could be non-uniform. For all of these reasons it is very difficult to understand, from an information theory viewpoint, the capacity of the “3D object” channel. A work related to this issue is the one of Page et al. [17], where the information intrinsically conveyed by a shape is computed as the entropy of mesh curvature. The measure proposed in Page’s work seems to catch well shape redundancy, i.e. shape symmetry and repeating features. Another interesting study about this topic is that by King and Rossignac [18] where the optimal trade-off between number of vertices and the number of allocated bits for vertex is established. In particular a shape complexity measure $K$ is evaluated by considering the number of vertices of the sphere that best approximate locally the mesh. The theoretical relationships between the shape complexity $K$, the number of vertices of the surface, the number of bits per vertices and the total amount of bytes necessary to represent the overall shape are established and then used to find a mesh that represents the shape in an optimal way from the storage viewpoint.

Another important consideration about the payload achievable on meshes is that this kind of representation is *highly redundant*, i.e. in fact it is possible to reach high compression ratios with both lossless and near-lossless schemes. Meshes are usually stored using floats for vertex coordinates and integers for connectivity. Vertex coordinates require 3 floats, i.e. $32 \times 3 = 96$ bits to be stored. A triangle of the mesh is defined by 3 integers representing the indices of the three vertices that
3.2. 3D Watermarking

form it. So, considering 32-bits integers other 96 bits are necessary to represent the connectivity of each triangle of the mesh. Obviously, this way to encode the connectivity is inefficient but it is also the most immediate and intuitive way to do. The compression of meshes can be separated in two parts, the compression of the geometry, i.e. the compression of the vertices data, and the compression of the topology, i.e. the compression of the connectivity of the mesh. The geometry is compressed usually by combining three steps: vertex quantization, prediction and statistical coding of the residuals. Typically the vertex coordinates are quantized using 12 bits per coordinate instead of 32 bits\(^3\). One of the most popular prediction rules is the parallelogram rule [19]. Sometimes some residuals may be large using the parallelogram prediction. However, the distribution of the residuals makes them suitable for statistical compression [20]. The combination of these steps results typically in near-lossless compression of the geometry to about 7 bits per vertex. Concerning connectivity compression, today, a lot of mesh topology compression schemes are near to the theoretical lower bound for encoding planar triangular graphs established by Tutte [21], that is 1.62 bits/triangle. Most of these techniques are variant of the Edgebreaker [22] scheme that guarantees an upper bound of 1.78 bits/triangle for any mesh homeomorphic to a sphere. These high performances, in term of compression ratio, demonstrate the high redundancy of the mesh representation.

3.2.3 Imperceptibility

Watermark imperceptibility is a crucial requirement for 3D watermarking. In fact, it is very important that a watermarking system for 3D objects guarantees that the watermarked model has the same visual appearance of the original one since the typical intended use of a 3D model is viewing. The problem is that any watermarking algorithm introduces distortions in the watermarked media. Such distortions are caused by the alterations affecting the host features during the embedding process. In the case of image and video watermarking the knowledge of the Human Visual System (HVS) has been widely applied to develop methods to make such distortions imperceptible [23, 24, 25]. The imperceptibility is achieved by processing the image or video content and evaluating, for each region of the image, how much the modifications will result perceptible. Then, the watermark

\(^3\)It is well-known that the quantization of the vertex coordinates with 12 bits ensures a sufficient geometric fidelity for most applications and most models. For this reason we consider this kind of compression near-lossless.
is embedded only in those parts of the image, or of the video, such that theirs changes will result less perceptible. This process is called visual masking. From our knowledge, no analogous studies exists for 3D objects. In fact, to mask the visual distortions in a 3D model is very difficult for two main reasons. The first one is that the visibility of the watermark depends on the particular rendering conditions used to visualize the mesh, i.e. by the shading algorithm used, by the surface properties, by the textures, and so on. The second reason is that, typically, the user interacts with the 3D model and hence the model can be observed from several viewpoints, thus making the perceptual analysis of the introduced visual artifacts more difficult than in the case of image and video. So, watermark imperceptibility for 3D watermarking requires new studies and approaches. The third part of this thesis is dedicated to our investigations about this complex topic. In particular, we will propose a new subjective test methodology for the evaluation of the visual artifacts introduced in a 3D model, and, two new novel perceptual metrics for 3D watermarking developed on the basis of the experimental data obtained with this experimental methodology.

3.2.4 Robustness

The most important requirement for watermarking algorithms used in certain application contexts such as for Intellectual Property Right (IPR) protection, is robustness against malicious and non-malicious manipulations. For this reason, one of the main issue in Digital Watermarking of the last years, it has been the evaluation of the robustness of watermarking algorithms. In particular, a series of benchmarks has been studied and developed. One of the first benchmark for image watermarking has been the StirMark [26] package, developed by Fabien A. F. Petitcolas [27, 28] during its PhD at Cambridge University in 1997. StirMark benchmark is one of the most used benchmark for still image watermarking. Other contributors currently included researchers at INRIA, Eurécom, University of Magdeburg and USTL-LIFL. Some of the image alterations implemented in StirMark are: cropping, flip, rotation, rotation+scaling, Gaussian filtering, sharpening filtering, linear transformations, random bending, aspect ratio changes, line removal and color reduction. Jane Dittmann, Andreas Lang and others are also working on an audio version of StirMark called AudioStirMark [29]. Other benchmarks for still image watermarking are the Optimark [30] software, developed by Artificial Intelligence and Information Analysis Laboratory of the Aristotle University of Thessaloniki, Greece, and the Checkmark [31] developed at the Com-
3.2. 3D Watermarking

<table>
<thead>
<tr>
<th>Image Attacks</th>
<th>Mesh Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation/Rotation/Scaling</td>
<td>Translation/Rotation/Scaling</td>
</tr>
<tr>
<td>Additive Noise</td>
<td>Additive Noise</td>
</tr>
<tr>
<td>Down-sampling</td>
<td>Simplification</td>
</tr>
<tr>
<td>Up-sampling</td>
<td>Refinement</td>
</tr>
<tr>
<td>Compression (e.g. JPEG)</td>
<td>Mesh Compression</td>
</tr>
<tr>
<td>Low-pass Filtering</td>
<td>Smoothing</td>
</tr>
<tr>
<td>Cropping</td>
<td>Cropping</td>
</tr>
<tr>
<td>Bi-dimensional Free-Form Deformation</td>
<td>Three-dimensional Free-Form Deformation</td>
</tr>
<tr>
<td>Second watermark insertion</td>
<td>Second watermark insertion</td>
</tr>
<tr>
<td>★ ★ ★</td>
<td>Vertices and/or faces re-ordering</td>
</tr>
<tr>
<td>★ ★ ★</td>
<td>Re-triangulation</td>
</tr>
<tr>
<td>★ ★ ★</td>
<td>Remeshing</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison between image and mesh attacks.

Computer Vision and Multimedia Laboratory of the University of Geneve. Optimark benchmark includes several image attacks plus some watermarking algorithms performances evaluation methods such as statistic to evaluate detector performances (e.g. Bit Error Rate (BER), probability of false detection, probability of missing detection), and the estimation of mean embedding and detection time. The CheckMark software is one of the most recent benchmarks for still images and includes some new classes of attacks such as non-linear line removal, collage attack, denoising, wavelet compression (JPEG2000), projective transformations, copy attack, etc. The CheckMark package is primarily developed by Shelby Pereira [32].

3D watermarking demonstrates to be an young technology even under this aspect, in fact no benchmark tools for 3D watermarking algorithms has been developed since now. So, there are no “standards” to evaluate the robustness of 3D watermarking algorithm. Typically the attacks used to test the robustness performances of 3D watermarking are extrapolated from the image attacks. Obviously, such attacks require to be adapted to the meshes. In addition to these adapted image attacks a lot of other sophisticated and difficult-to-prevent attacks such as mesh optimization and remeshing are possible on a polygonal mesh. Table 3.1 shows the battery of mesh attacks we propose and the corresponding image attacks. As it is possible to notice some mesh attacks have not a corresponding image attack. In the following we describe in detail such attacks.
Chapter 3. Watermarking of 3D Objects

Translation/Rotation/Uniform Scaling

These simple geometric transformations are commonly used in Computer Graphics to position a 3D model inside a scene. In fact, usually a virtual scene is composed by hundred of objects that must be rotated, scaled and translated to obtain the final results. So, this is a basic non-malicious attack to consider. Other more complex geometric transformations such as affine and projective transformations may be used to attack the model, even if they are less common than plain translations, rotations and isotropic scaling.

Noise

Noise attack consists in the addition of uniform or gaussian noise to the digital media in order to compromise the watermark. Several ways to add noise to a geometric surface are possible. One way is to perturb mesh vertices position by adding them random displacement vectors. If the maximum modulus of such vectors is small with respect to the mesh dimensions the overall shape of the model is preserved. Figure 3.14 shows an example of this process.

Re-triangulation

This attack regards the topology of the mesh. It is performed by changing the connections between mesh vertices leaving their position unaltered (Figure 3.13). One important characteristic of this attack is that it is very easy to implement and very fast to apply, so a lot of pirates may use it to attack the watermarked mesh.
3.2. 3D Watermarking

Vertices and/or faces re-ordering

This attack consists in re-ordering the vertices, or the faces, of the mesh preserving the aspect of the mesh exactly. For example the vertex $v_i$ can be swap with the vertex $v_j$ by taking care to modify opportunely the indices of those triangles $v_i$ and $v_j$ belong to. Further, the list of the triangles of the mesh can be order in any way. This attack is specific of 3D objects since, as previously noticed, the samples of other media such as audio, image and video, have an implicit order that preserve them to the intensive use of this attack. The conclusion is that it is not possible to assign to a vertex a particular meaning during the embedding or the decoding/detection phase unless such vertex is identified without the use of its indices.

Mesh smoothing

Mesh smoothing is a geometric operation that is usually performed on the meshes obtained through three-dimensional scanning. In fact, usually 3D scanners produce “noisy” surfaces due to the approximation errors introduced during the surface reconstruction process. Such errors could be compensated by fairing mesh techniques [33, 34, 35, 36, 37, 38, 39]. In Figure 3.14 an example of the Bunny model after the application of Taubin filter is showed. This filter acts on a mesh as a low-pass filter attenuating the roughness of the surface.

Cropping

Cropping concerns the disjunction of a part of the model. Pirates can discard the pieces of the model that they do not need (e.g. the hand of a statue). This simple attack presents a lot of complications since often it can cause synchronization problems during the extraction phase. This important aspect will be debated in Chapter 5. Figure 3.14 shows the Bunny model cropped with a plane.

Mesh Compression

Despite many years of compression research a widely supported compression standard for 3D models, such as JPEG for images and MPEG for videos, does not exist in practice. The attempts to establish similar compression standard (e.g. MPEG-4, compressed-binary VRML) did not succeed. The main reason of this is that a single compressed format suitable for all cases is difficult to achieve. In
fact, 3D models can either have or not have texture information, there can be pre-
computed colors (rarely), material attributes present high variability in their data
form, and so on. Furthermore, it is difficult to define a widely accepted standard
and, consequently, the robustness performance against compression is difficult to
evaluate. As just pointed out the compression schemes for polygonal meshes are
typically lossless for the connectivity and near-lossless for the geometry. Hence,
in general, to take into account the effects of the compression of the mesh on the
watermark it should be sufficient to consider the effect of the quantization of the
vertex coordinates. We remember that typically each vertex coordinate is quan-
tized with 12 bits instead of the 32 bits of the standard float representation of the
uncompressed case.

Simplification

Mesh simplification, also referred to as decimation, regards the reduction of the
number of vertices and triangles of a polygonal mesh model while preserving its
shape. Typically, an atomic simplification operation is defined and applied several
times to obtain the simplified mesh. An example of atomic simplification is the
removing of a vertex with a successive retriangulation of the produced hole. The
sequence of atomic simplification steps to apply to the model is builded using a
similarity metric. The similarity metrics measure the impact of each atomic step
and determine the quality of the simplified output mesh. Such metrics can be
of two types: geometric and image-based. Some examples of algorithms that use
geometric metrics are [40, 41], and image-based metrics are [42, 43, 44]. Since
we are interested in measuring the impact of watermarking process, in Section
7.2.2 we give more detail about such metrics, as a good starting point to get
ideas for developing metrics for watermarking impairments evaluation. This kind
of processing can be considered as a non-malicious attack when it is applied to
optimize the mesh, i.e. to eliminate over-sampling of the model’s surface.

Refinement

The refinement operation regards the insertion of new vertices, edges and faces
into a triangular mesh. This operation could be performed for different reasons.
For example, a refinement could be done to improve the quality of the mesh, i.e.
the uniformity of the area and angles of its triangles. Another application context
is finite element analysis of a certain model and/or surface, e.g. the evaluation of
the stress suffered by a mechanical piece during its work condition. Even in this
3.3. State of the Art

case the mesh may be refined to build the basic elements where to perform the finite element analysis. So, this attack belongs to the category of non-malicious attacks. Obviously, the insertion of new vertices, edges and faces with malicious intentions is always possible.

Remeshing

Remeshing is used to regularize a mesh converting an irregular mesh into a semi-regular [45, 46, 47] or a completely-regular [48, 49, 50] one. In few words this operation can be described as a geometric re-sampling of the shape of the model followed by a re-definition of the vertices connections in order to give the mesh vertices the desired valence. From our knowledge there are no experimental results about the resistance of 3D watermarking algorithms against this attack. Our feeling is that to make a watermarking system robust against remeshing is very difficult. Fortunately, the use of this attack is limited by its characteristics since it is very complex to implement and high computationally expensive.

3.3 State of the Art

Digital watermarking of 3D objects is far from the level of maturity of other watermarking technologies such as the ones for audio, images and video.

One of the first works about 3D watermarking was done in 1997 by Ryutarou Ohbuchi and other researchers of the IBM Tokyo Research Laboratories [51]. In 1998 the first multiresolution approach was proposed by Kanai et al. [52] and after one year a non-blind technique with very good robustness properties was developed by Praun et al. [53]. At the middle of 2000 Oliver Benedens from the Fraunhofer Institute of Technology, Germany, in his work “Towards Blind Detection of Robust Watermarks in Polygonal Models”, presented the first paper about blind and robust watermarking for 3D objects [54]. Until now, many other techniques have been proposed by many other researchers and the effort to develop 3D watermarking techniques is constantly increasing. The development of a blind and robust 3D watermarking algorithm yet is still a difficult task. Our work tries to move other steps in this direction.

In the following, a detailed review of the most important 3D watermarking algorithms for polygonal meshes will be presented. The algorithms have been categorized according to the host features used for the embedding. Usually watermarking techniques are divided into four main categories: those operating in the
Figure 3.14: Mesh Attacks. (a) Original Model. (b) Additive Noise attack. (c) Smoothing. (d) Cropping.
asset domain, be it the spatial domain or the time domain depending by the asset, those operating in the transformed domain, for example in the DCT or DFT domain, those operating in the hybrid domain, retaining both spatial/temporal and frequency characterization of the host asset, and, finally, those operating in the compressed domain, i.e. working directly on the compressed version of the host asset.

3.3.1 Asset Domain

The most straightforward way to embed information within a digital host asset is to modify the asset in its original signal space, in other words the host features correspond to the signal samples. For audio signal this means that the watermark is embedded in the time domain, for still images this means to work in the spatial domain, and so on. The main peculiarity of embedding in the asset domain is that in this way the temporal/spatial localization of the watermark is trivial, thus permitting a good control of the local distortions introduced by the watermark. For 3D watermarking in the asset domain several options are available; the watermark can be embedded in the topology, in the geometry, or in the appearance attributes of the 3D model.

Topological Features

For topological embedding, or embedding using the topology of the model, we intend both those algorithms that encode the information in the connectivity of the model and those algorithms that use the topology of the model to drive the alterations of the geometry of the mesh. In other words all the algorithms for which the topology plays a fundamental rule in watermark casting.

Some remarkable examples of this kind of algorithms can be found in the first work by Ohbuchi et al. [51], where four watermarking algorithms for polygonal meshes based on topological features were presented. These algorithms were named Triangle Similarity Quadruple (TSQ), Tetrahedral Volume Ratio (TVR), Triangle Strip Peeling Sequence (TSPS) and Macro Density Pattern (MDP).

The Triangle Similarity Quadruple (TSQ) algorithm is an excellent example of embedding by topology-driven mesh alterations. In fact, a combination of topological and geometric features is used to embed the watermark. The base host feature of the TSQ is the Macro-Embedding-Primitive, or MEP (Figure 3.15 (right)). Each MEP stores a marker $M$, a subscript $S$, and two data values $D_1$ and $D_2$. The marker triangle is used to identify the MEP, the subscript triangle
maps the set of MEPs into a sequence and the triangles \( D_1 \) and \( D_2 \) stores the data. Embedding is achieved by modifying the dimensionless quantities that identifies *similar triangles*; referring to Figure 3.15 (left) such quantities are the pair of values \{b/a, h/c\} and the angles \{\theta_1, \theta_2\}. To be more specific, the TSQ embedding phase can be summarized by the following steps:

1. Given a pair of suitable values the triangles of the mesh are traversed and the set of MEPs is computed. Triangles whose dimensionless quantities are too small are skipped.

2. For each MEP the marker \( M \) is created by changing the pair of values \{e_{14}/e_{24}, h_0/e_{12}\} by modifying the vertices \( v_1, v_2 \) and \( v_4 \). The symbols \( e_{ij} \) indicate the length of the edge between the vertex \( v_i \) and \( v_j \).

3. The subscript is embedded in the triangle \( S \) by changing the pair of values \{e_{02}/e_{01}, h_0/e_{12}\}. Then, the two data symbols are embedded in the pairs of values \{e_{13}/e_{34}, h_3/e_{14}\} and \{e_{45}/e_{25}, h_5/e_{24}\} by modifying the position of the vertices \( v_0, v_3 \) and \( v_5 \).

The extraction is achieved by traversing the triangles of the watermarked mesh, identifying the MEPs thanks to the marker triangle, and re-arrange the data symbol according to theirs subscript.

Another example of topology-driven mesh alterations embedding is the Tetrahedral Volume Ratio (TVR) algorithm. This algorithm has been designed to be robust against affine transformation. In fact, its host feature, i.e. the ratio of two tetrahedral volume, it is invariant against affine transformation. In few words, the embedding consists in the identification of a sequence of tetrahedron on the input
3.3. State of the Art

Figure 3.16: Example of information encoding in a triangle strip (TSPS).

Figure 3.17: Example of Macro Density Pattern (MDP) embedding (from [51]).

mesh and then to embed the information by modifying the ratio of the volume of these tetrahedrons. Like TSQ algorithm, TVR is a blind, readable watermarking scheme.

The connectivity of a strip of triangles encodes the information in the Triangle Strip Peeling Sequences (TSPS) algorithm. The base idea is to peel-off a sequence of triangles, to be more specific a strip of triangles, and then encodes the information in the strip. By assuming that the information to embed has the form of a string of bits, the strip is created in the following way: starting with an edge of a triangle, the two opposites edge identify two corresponding triangles. If the mesh to watermark is orientable these two triangles can be ordered in a clockwise or counter-clockwise way. According to the selected orientation, a ‘0’ or ‘1’ is assigned to the first and the second triangle. Obviously, some strings cannot be fitted by the topology of the input mesh, for example the strip can hit a boundary of the mesh or circle-back to itself. To avoid such problems the shape of the triangle strip is manipulated by alternating data symbols with steering symbols, i.e. symbols that steer the direction of growth of the triangle strip. An example of such kind of connectivity-based information encoding is shows in Figure 3.16 (left). The decoding of the watermark is achieved by identifying the peeled-off strip into the watermarked mesh and then directly reading the bits by analyzing the strip’s connections.

The Macro Density Pattern (MDP) is a visual watermarking method, in other words the watermark is retrieved by visual inspection and not by the computer, likely as a logo. The idea is to modulate the density of the triangles by a proper

---

4 After the “peeling” operation the strip is connected to the rest of the mesh only by its first edge.
tessellation of the input models. If this tessellation respects the local curvature of the mesh the density changes result invisible when the model is rendered with common shading algorithms, but result visible when the edges of the mesh are visualized. Figure 3.17 (right) shows an example of mesh watermarked with this technique.

Another topology-driven watermarking scheme is the Triangle Flood Algorithm (TFA) [55] developed by Oliver Benedens. This algorithm uses both topological and geometric information to generate a unique path of triangles on the mesh. Vertices of triangles along this path are modified both to embed bits (by modifying triangles height) and to order the triangles yielding a unique traversal path.

Another remarkable example of 3D watermarking algorithm that operates in the topological domain is the recent blind method for readable watermarking proposed by François Cayre and Benoit Macq [56]. This algorithm takes the basic ideas of TSPS and improves it by adding security properties. In fact the major drawback of the original TSPS implementation is that it is easy for an opponent to locate the payload and hence to change or discard it. In this implementation the triangle strip does not encode the watermark but it is builded according to a sequence of bits given by a secret key $K$ (private watermarking). After build, each triangle of the strip is used to embed 1 bit of the watermark by modifying its geometric properties. In particular the state of the triangle (“0” or “1”) depends on the position of the orthogonal projection of the triangle summit $C$ on the “entry edge” $AB$ (see Figure 3.18). This way to proceed can be seen as a quantization index modulation (QIM) scheme extended on a discrete partition of a physical measurement. The starting triangle of the strip is determined on the basis of specific geometric characteristics. In this way, the peel-off operation is avoided and it is impossible to locate the embedded information if $K$ is unknown, thus resulting in a secure topological embedding scheme.

The main problem of watermarking algorithms based on topological embedding is their robustness. In fact, a simple re-triangulation attack destroys the watermark. An exception to this is the MDP that does not suffer re-triangulations attacks; nevertheless, the watermark is known and it is very simple to modify or remove it by using a simple geometric editing software. Concluding, this kind of algorithms are suitable for annotation or similar applications, where the robustness requirements are relaxed.
3.3. State of the Art

Figure 3.18: Watermark embedding in modified TPS.

Geometric Features

The most intuitive way to embed information into a 3D model is through modifications of its geometry, e.g. vertex coordinates. Several host geometric features are possible but, at the end, the most used features for watermark embedding are basically two: the position of the mesh vertices and the vertex (or face) normals. Concerning vertex and face normals some clarifications are opportune. The direction of the face normal must be consistent with the orientation of the face. Typically, once computed, the direction of each normal is adjusted such that all the face normals point outward the 3D object. This adjustment is trivial if all the faces of the mesh have the same orientation. The vertex normal is an approximation of the surface normal in the continuous case and it is usually defined as:

\[
\vec{n}_v = \frac{1}{|F(v)|} \sum_{f \in F(v)} \vec{n}_f
\]  

(3.13)

where \(F(v)\) is the set of the faces adjacent to the vertex \(v\), \(|F(v)|\) its cardinality and \(\vec{n}_f\) is the normal of the face \(f\). Another possible definition of the vertex normal is:

\[
\vec{n}_v = \frac{1}{A} \sum_{f \in F(v)} A_f \vec{n}_f
\]  

(3.14)

where \(A_f\) is the area of the face \(f\) and \(A\) is the total area of the faces adjacent to \(v\). Face or vertex normals are intrinsically tied to the shape of the model, in fact, while vertices position is determined by how the surface is sampled, the influence of surface sampling on this kind of feature is negligible if the density of the samples is sufficiently high to accurately approximate the surface.

In the following we give a brief description of three algorithms based on vertices position \([55, 57, 58]\) and other two algorithms based on shape-related geometric features, i.e. vertex and face normals \([59, 60]\).
The first algorithm we describe that uses vertices position to embed the water-
mark is the Vertex Flood Algorithm (VFA) [55] developed by Olivier Benedens. 
This algorithm was designed for public watermarking, hence the high capacity is 
one of its main characteristic. The vertices of the mesh are clustered in subsets 
\(M_k\) according to their distance between a pre-defined point on the mesh \(c\) that is 
the center of a reference triangle \(R\) chosen on the basis of its geometric properties:

\[
M_k = \left\{ v \in V \setminus R \mid k \leq \frac{\|v - c\|}{W} < k + 1 \right\}, \quad 0 \leq k \leq \left\lfloor \frac{d_{MAX}}{W} \right\rfloor 
\]  

(3.15)

where \(R = \{v_1^R, v_2^R, v_3^R\}\) is the set of vertices of the reference triangle, \(c\) is the 
centroid of such triangle, i.e. \(c = (v_1^R + v_2^R + v_3^R)/3\), \(d_{MAX}\) is the maximum 
allowed distance from \(c\) and \(W\) is the width of each set. The reference triangle is 
selected as the one whose edges are closer to a predefined edge length ratio. After 
populating such sets the bits of the watermarked are encoding loop from \(M_0\) to \(M_k\) 
and skipping the empty sets. Each set is sub-divided in \(m\) intervals in order to 
encode \(m\) bits:

\[
\text{buf} I_0 \ I_1 \ \ldots \ \ I_{m-2} \ I_{m-1} \ \text{buf} \\
m + 2 \ \text{sub-intervals}
\]

(3.16)

The distance of each vertex in the set (from \(c\)) is modified so that it comes to 
lie in the middle of the sub-intervals representing the desired bits. For example, 
if \(m = 3\), the bits 101 are encoded moving all vertices in the sub-interval \(I_5\). 
The purpose of the two intervals named “buf” is to prevent that modifications 
of vertices in one interval cause effects on the vertices on another interval. This 
algorithm is robust against geometric transformations such as translation, rotation 
and uniform scaling. The embedded bits are recovered by building the sets \(M_k\) 
of the watermarked mesh and then directly read the watermark bits by analyzing 
the sub-intervals of these sets.

T. Harte and G. Bors [57] developed another blind and readable watermarking 
scheme to embed the bits of the watermark in the position of the vertices. In 
this scheme each vertex encodes 1 bit. A vertex encodes 1 or 0 if its position 
is outside or inside a bounding volume defined by its neighborhoods vertices. 
Two kind of bounding volumes have been proposed; a “bounding planes” and a 
“bounding ellipsoid”. Here, we describe the last one. The vertices considered for 
the embedding are those which have the distance to its neighborhoods smaller 
than a certain threshold \(T\):

\[
\sum_{j \in V_i(i)} \|v_i - v_j\| < T 
\]

(3.17)
A ellipsoidal bounding volume can be associated to the vertex \( v_i \) by considering the centroid \( \mu_i \) and the second order moment \( S_i \) of its neighborhood. Such ellipsoid is defined as:

\[
(x - \mu_i)^T S_i^{-1} (x - \mu_i) = K
\]  

(3.18)

where \( K \) is a normalization factor. The embedding approach based on the bounding ellipsoid considers that the vertex \( v_i \) encodes 1 if it is inside the ellipsoid, i.e. if \((x - \mu_i)^T S_i^{-1} (x - \mu_i) < K\). Otherwise the vertex encodes 0. It is important to underline that the position of \( v_i \) does not affect the shape of the bounding ellipsoid. During the embedding and the decoding phase the vertices are ranked with respect to their distance between their relative neighborhood ellipsoid center. This algorithm is robust against geometrical transformation such as translation, rotation and scaling. No evaluation tests about the robustness of the technique against other attacks such as noise addition, simplification, and so on, has been led.

Yeo and Yeung [58] developed an authentication algorithm for 3D objects based on small perturbations of mesh vertices. Embedding is achieved by adjusting vertex coordinates to make the following expression valid for each vertex:

\[
K (p(v)) = W (L(v))
\]  

(3.19)

In the above expression \( K(.) \) is the verification key, i.e. a set of lookup tables with binary entries, \( p(v) \) is an index value depending on vertex coordinates, \( W(.) \) is a bi-dimensional matrix of binary values representing the watermark and \( L(v) \) is a location of such matrix. This technique allows quick localization and visualization of slight modifications of the watermarked model. The proposed method avoids problems related to cropping attack by making the function \( p(v) \) dependent even on the neighborhood of \( v \). In this way the system is able to recognize if cropping has occurred.

One of the most interesting algorithm that exploits face normal features is the Normal Bin Encoding (NBE) [59] of Oliver Benedens. As previously stated face normals are directly related to the shape of the models, in fact this non-blind algorithm for readable watermarking is robust against attack like mesh simplification, i.e. all those attacks that alter the mesh considerably but that preserve the overall shape of the model. First of all, the face normals are subdivided in different sets called bins. Each bin is defined by a normal \( \vec{n}_B \) named the center normal of the bin and an angle \( \phi_R \) called bin radius. If the angle between the normal bin center and the face normal is less then \( \phi_R \) this face normal is assigned to such bin.
Each bin encodes 1 bit of information by using different features obtained from the analysis of the bin. Such bin features are defined as:

\begin{align*}
\text{Bin Feature I} & : \text{com}_i = \frac{1}{|B_i|} \sum_{j=1}^{|B_i|} \tilde{n}_{ij} \\
\text{Bin Feature II} & : \text{ma}_i = \frac{1}{|B|} \sum_{j=1}^{|B|} |B_i| \cos^{-1}(\tilde{n}_{ij} \cdot \tilde{n}_{B_i}) \\
\text{Bin Feature III} & : \text{nk}_i = \frac{|BK_i|}{|B_i|}
\end{align*}

where \( \tilde{n}_{ij} \) indicates the \( j^{th} \) normal belonging to the bin \( B_i \) and \(|B_i|\) is the number of the normals of that bin. The first feature is the mean normal. The second bin feature is the mean angle difference between the bin normals and the bin center normal. The third bin feature is the ratio between the number of the normals that are inside a kernel area defined by another angle \( \phi_K \) such that \( \phi_K < \phi_R \) and the total number of bin normals. The first feature is used to embed bits by considering the projection of the mean normal on a local planar frame tangent to the bin normal center. It is trivial to notice that the bin feature II and III are related. These two last features are used to embed bits considering how many normals of the bin are inside/outside the kernel area. The embedding is achieved by choosing one of the proposed bin features, the type I or the type II-III, and perturbing the position of the vertices in order to assign to the selected bin feature the target value. To do this the vertices are re-positioned by solving an optimization problem. This optimization problem concerns the minimization of a cost function that measures, for each bin, how far the value of the selected bin feature is from its target value. The decoding is achieved by building the bins from the watermarked object and then computing the bin features. Before the watermark extraction phase, in order to correctly build the bins, the watermarked model is re-oriented by a pre-processing stage using the original model.

Another interesting algorithm based on shape-related geometric features is the one by Wagner [60]. Such algorithm uses as host features an approximation of the surface normals different from the definitions (3.13) and (3.14):

\[ \tilde{n}_i = \frac{1}{|v_1(i)|} \sum_{j \in v_1(i)} (v_i - v_j) \]

where \( v_1(i) \) is the 1-ring of the vertex \( v_i \) and \(|\cdot|\) is the usual cardinality operator. This definition of vertex normal relies on the discrete approximation of Laplacian
operator and it is usually referred to as the *umbrella-operator*. The Euclidean norm of \( \vec{n}_i \) is altered according to a function \( f(.) \) that encodes the watermark. Such function could describe pixel intensity value of a company logo projected on the unit sphere or could be a pseudo-random noise function. The desired value of \( ||n_i^*|| = f(\vec{n}_i) \) is obtained by perturbing the position of the vertices according to:

\[
n_i^* = \frac{1}{|v_1(i)|} \sum_{j \in v_1(i)} (v_i^* - v_j^*) ,
\]

that yields a linear system of \( n \) equations in \( n \) unknowns, where \( n \) is the number of neighborhood vertex of \( v_i \), i.e. \( n = |v_1(i)| \). As demonstrate by Wagner, this algorithm can be made robust against affine transformation, by replacing the Euclidean norm with an affine-invariant vector norm.

From what exposed so far we can conclude that techniques based on geometric features can be used to develop 3D watermarking algorithms with different characteristics in terms of capacity and robustness. Hence, it is possible to conclude that geometric-based techniques are more flexible than those based on topological embedding. Robustness is achieved at the expense of imperceptibility, i.e. more high are the robust requirements more high is the visual impact of the distortions introduced by the watermark. The techniques based on shape-related geometric features, such as face normal, have good robustness against complex attack such as mesh simplification. On the contrary, they could suffer attacks like noise addition. Following these considerations further studies about the embedding by the combination of different geometric features would be desirable. One last consideration about robustness of geometry-based techniques is that, often, the impact of re-triangulation attack is negligible.

### Surface Attributes

The last set of host features belonging to the asset domain that can be used to embed information are the surfaces attributes. Typical surface attributes are colors, textures, and other additional data that can be used to model the material of the surface such as density, transparency, and so on. Since now, this kind of features has been rarely used to embed watermarks in 3D objects. In the next we illustrate the basic principles of texture-based 3D watermarking following what stated in a recent work of Garcia and Dugelay [61].

The basic idea of texture-based watermarking of 3D objects is to embed the watermark into the texture of the 3D model. A texture is an image associated
Chapter 3. Watermarking of 3D Objects

Figure 3.19: Texture-based 3D watermarking.

to the geometry of the model to enhance the visual content of the model itself. *Texture mapping* is the name of the process that maps the vertices to the texture. Texture mapping will be detail more in Chapter 6. Since the texture is an image it is possible to watermark it by any image watermarking algorithm. So, the embedding/detection phase consists of two steps: in the first the texture is reconstructed from a visual representation of the model, i.e. a single or a set of images depicting the model; in the second step the watermark is extracted from the reconstructed texture. The first step of the extraction phase concerns the reconstruction of the texture from images of the model since the main goal of texture-based 3D watermarking is the protection of the visual representation of the object and not the protection of the 3D object itself. In fact, the crucial point of this approach, is the reconstruction of the watermarked texture. The overall process is schematized in Figure 3.19. Garcia and Dugelay [61] propose a 2D/3D registration scheme to estimate the projection matrix and then invert the texture map process. To do this the geometry of the original model is required, hence this kind of technique is not blind. Sometimes the projection parameters are not sufficient to reconstruct the texture but even other parameters that influence the visualization of the model, for example the position of the light, have to be estimated to correctly reconstruct the original texture. The robustness performances tested by Garcia and Dugelay demonstrate that small errors in projection parameters estimation have the same
effect of severe image attacks, such as JPEG compression at 75% level of quality, confirming the importance of the reconstruction phase.

3.3.2 Transformed Domain

These kind of techniques perform the insertion of the watermark into the coefficients of a mathematical transformation of the host asset. The most used transformation are the ones related to the frequency domain, typically the Discrete Fourier Transform (DFT) or the Discrete Cosine Transform (DCT). Usually, these techniques exhibit good robustness against attacks. For example, by spreading the watermark over the whole asset, they are intrinsically more resistant to cropping than the techniques operating in the asset domain. Another example is robustness against shifting and scaling, that it is easily achieved in transformed domain. For instance, in the DFT domain, the magnitude of the DFT coefficients is invariant to shifting. One first disadvantages of these methods is that the localization of the distortions introduced by the watermarking process is more difficult with respect to the techniques that operate in the asset domain. Another drawback of transformed domain is the high computational cost of such techniques. In fact, it is necessary to pass from the asset domain to the transformed domain and, after embedding, to apply the inverse transformation to obtain the watermarked asset. As previously mentioned, mathematical tools such as DFT or DCT are not available for geometric data, so today few techniques for mesh watermarking work in the transformed domain. One recent extension of the Fourier analysis to mesh, called mesh spectral analysis has been employed for the first time by Karni and Gotsman [62] for lossy compression of polygonal meshes. Mesh spectral analysis is based on the extension of the classical spectral graph theory [63, 64]. In particular, by indicating with $A$ the adjacency matrix, i.e. a matrix encoding the connectivity of the mesh in the following way:

$$A_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (3.25)$$

and with $D$ a diagonal matrix whose diagonal element are defined as $D_{ij} = 1/d_i$, i.e. as the reciprocal of the vertex valence ($d_i$); the Laplacian matrix can be defined as $L = I - DA$:

$$L_{ij} = \begin{cases} 1 & i = j \\ -1/d_{ij} & \text{if } v_i \text{ and } v_j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (3.26)$$
The eigenvectors of $L$ form an orthogonal basis of $\mathbb{R}^n$. The associated eigenvalues may be considered frequencies, and the projections of each of the coordinate vectors\(^5\) on this basis functions is the spectrum of the geometry.

Ohbuchi et al. have been developed a 3D watermarking algorithm based on mesh spectral analysis [65]. The watermark is inserted in mesh spectral domain by modifying the mesh spectral coefficients and then, the inverse transformation is used to obtain the watermarked mesh. Ohbuchi et al. use a different definition of the Laplacian matrix $L$ for the mesh spectral analysis, called combinatorial laplacian matrix or Kirchoff matrix. The Kirchoff matrix is defined as:

$$K = D - A$$  \hspace{1cm} (3.27)

where $D$ is a diagonal matrix whose diagonal element are the valence of the mesh vertices, i.e. $D_{ij} = d_i$ and $A$ is the same adjacency matrix defined in (3.25). If the mesh has $n$ vertices $K$ has dimensions $n \times n$. The computational complexity to calculate the eigenvectors is $O(n^3)$, that is prohibitive for mesh with thousands of vertices. In order to reduce the complexity, the input mesh is partitioned in sub-meshes [62], then each sub-mesh is watermarked separately. The watermark extraction requires the original mesh spectral coefficients to be achieved. This non-blind algorithm is robust against additive vertex noise, partial resection (cropping) of the mesh, similarity transformation (translation, rotation and uniform scaling) and mesh smoothing. An improved version of this algorithm [66] uses a resampling phase to obtain robustness against remeshing and mesh simplification.

Another 3D watermarking algorithms based on mesh spectral decomposition has been proposed by Cayre et al. [67]. In this work they propose some improvements to reduce the computational burden of mesh spectral decomposition, like the use of a fixed spectral basis, and a new partitioning scheme of the input mesh characterized by overlapping between the sub-meshes to improve the quality of the reconstructed mesh and the performance of the watermarking algorithm. The robustness performance against several geometric attacks such as noise addition, mesh smoothing, geometric transformation, and so on, are very good. The proposed method is also able to resist to compression of the spectral mesh coefficients. No connectivity-based attacks have been tested.

As previously stated, typically, this kind of techniques are more robust than the ones operating in the asset domain. From this point of view 3D watermarking

\(^5\)For coordinate vectors, here, we intend that all the coordinates of mesh vertices are arranged into three distinct vectors.
does not present exceptions. All the techniques based on mesh spectral domain, from our knowledge the only transformed domain for polygonal meshes used for 3D watermarking, present very good robustness against several attacks.

### 3.3.3 Hybrid Domain

The development of hybrid techniques has been inspired by the attempt to obtain both the main advantages of the techniques operating in the asset domain, i.e. the localization of the watermarking disturb, and the good robustness properties typical of the transformed domain techniques. Hybrid techniques keep trace of the temporal/spatial characterization of the host signal and, at the same time, using frequencies interpretation to achieve better information analysis and processing. The hybrid techniques that have received more attention since now are the ones based on block DCT/DFT and those relying on wavelet decomposition.

Wavelet decomposition has been the first multiresolution analysis tool used in 3D watermarking. In 1998, Kanai et al. [52] proposed one of the first multiresolution approach to 3D watermarking based on wavelet decomposition of meshes developed some years before, in 1995, by Lounsbery et al. [68]. A detailed description of wavelet decomposition algorithm is given in Chapter 4, where the theoretical background of our approach, based on the same wavelet analysis, will be presented. In few words, the wavelet coefficients\(^6\) computed by this analysis are vectors. Embedding is performed by changing the modulus of such vectors. In particular, the least significant bits of the modulus of each wavelet are used to insert the watermark. Not all the wavelet coefficients are watermarked in this way but a selection policy based on geometric thresholds is used to minimize the visual impact of the wavelets modifications. The watermark is extracted in a non-blind way by applying the wavelet decomposition and comparing the original and the watermarked wavelet coefficients. This techniques is robust against affine transformation, additive noise and partial resection of the mesh.

Praun et al. [53] developed, in 1999, one of the most robust algorithm for mesh watermarking. In fact, this techniques is robust against several attacks such as translation/rotation/uniform-scaling, additive noise, simplifications, mesh smoothing, re-triangulation and re-meshing. The main idea of this algorithm is to displace the vertices of the mesh according to a set of radial basis functions defined over the surface of the model. These basis functions are builded on salient

\(^6\)To be more specific, these wavelets are *lazy wavelets* since the mathematical framework is not whole applied.
Chapter 3. Watermarking of 3D Objects

features of the mesh by a multiresolution approach performed with a progressive mesh [69] encoding of the input model. The watermark is extracted by considering the linear correlation between the original and the watermarked vertices position. The watermark recovery phase is performed on a registered and re-sampled version of the watermarked mesh. For re-sampling, here, we intend that both the vertices’ position and the connectivity of the original mesh is reconstructed. Such registration/resampling phase is the kernel of the excellent robustness properties of this algorithm. Nevertheless, it needs the original model to be performed, thus resulting in a non-blind method with more limitations in its practical applicability.

A completely different approach is used by Song et al. [70]. In this work a virtual 3D scanner is simulated to create several range images of the model. A range image is an image whose pixels value corresponds to depth distances. So, a range image contains the information about the shape of the model. The obtained range image is watermarked used a standard image watermarking method based on DCT [71]. Then, the vertices of the model are moved accordingly to watermarked range image. This technique is not hybrid in the strict sense of the term but it uses the spatial localization given by the range image and the frequency analysis for the embedding. For this reason we decided to classify it as a hybrid technique.

3.3.4 Compressed Domain

The watermarking techniques that embed the information directly in the compressed domain present several advantages, in particular if a compression standard is chosen for the embedding (e.g. MPEG2). For example it is possible to avoid all the problems relative to watermark coefficients that are typically discarded or coarsely quantized after the compression. Additionally, to work directly in the compressed domain makes the computational burden low. This is especially true for video applications, where the videos are typically compressed using some video compression standard such as MPEG2, MPEG4, H.263 or H.264. In fact, in this way, the watermarking can be achieved without the necessity of decompress the video, saving a considerable amount of time. A disadvantage of such techniques is that often they are sensitive to transcoding, i.e. to the change of compressed domain (e.g. convert an MPEG2 bitstream to an MPEG4 bitstream).

From our knowledge no 3D watermarking algorithms work in compressed domain. Most probably one of the main cause is the fact that no recognized standard for geometry and/or topology compression exists.
3.3.5 Conclusions

This review about the current State-of-the-Art in 3D watermarking poses in evidence one of the main drawbacks of the current watermarking technology for 3D objects, i.e. the existing techniques with good robustness properties for 3D watermarking are not blind. Hence, most of the presented algorithms are not suitable for important practical applications, for example in the field of IPR protection.

To be more specific, most of the algorithms that work in the asset domain, in particular those ones based on the topology of the model, are blind, readable and capable to achieve high payload, but not robust, making them suitable only for all those applications where the robustness requirements are relaxed. Geometric-based methods are the more flexible, in term of characteristics, in the asset domain. All the blind techniques of this type are not robust against both malicious and non-malicious manipulations. In general, such algorithms achieve robustness at the expense of imperceptibility. The methods based on shape-related features seem to be intrinsically robust to certain complex attacks, such as mesh simplification. Watermarking through surface’s attributes is still an unexplored topic except for the texture-based 3D watermarking method. It is important to remark that this kind of watermarking algorithms appear to be more suitable to embed data into the visual representation of a 3D object instead of into the 3D object itself. The algorithms that work in the transformed domain exhibit good robust properties against several and complex attacks. The main drawback of this kind of techniques is the difficulty to control the distortions caused by the watermarking process and hence the difficult to obtain watermarked model with high visual quality. Those algorithms that work in hybrid domain have good robustness properties. From our knowledge no blind algorithms of this kind have been developed so far.

Concluding, we can state that those algorithms that work in the hybrid domain seems to be the most promising to develop blind and robust algorithm to produce high-quality watermarked 3D models. For these motivations we decide to develop our technique following a multiresolution approach. In the next part of the thesis, in particular in Chapter 5, our technique will be described in detail.
Part II

Wavelet-based Blind and Robust Watermarking of 3D Objects
Chapter 4

DGP Fundamentals

4.1 Digital Geometry Processing

Multimedia data types such as digital audio, images and video are widely used in many applications that we experience in our daily-life; from mobile phone to DVD players, from MP3 portable devices to digital cameras, and so on. A key factor of this enormous diffusion is the easy and economy of acquiring multimedia data. Another key factor is the existence of efficient techniques for the creation, editing, transmission and compression of such data. In other words, in the progress of audio, image and video processing, Digital Geometry Processing (DGP) [16] concerns the extension of Digital Signal Processing (DSP) techniques to geometric data. Such extension requires the development of new theoretical and algorithmic tools and it cannot be a simple adaptation of the existing signal processing techniques (e.g. Fourier analysis). In fact geometric data are characterized by non-uniform sampling, topology and intrinsic curvature while audio, images and video are represented by uniform sampling functions defined on Euclidean domains. These peculiarities cause a lot of difficulties in the design of efficient algorithms for manipulating 3D objects. Often these difficulties cannot be overcome. For example, the uniform sampling permits the definition and efficient application (FFT) of Fourier analysis; the same cannot be achieved for geometric data since it is not possible to define an uniform, i.e. equi-spaced, sampling pattern on a 2-manifold. A possible solution to this problem is the use of local parameterization of the 2-manifold in order to achieve local regular sampling and apply (windowed) standard signal processing tools, like the FFT. It is important to remark that many parameterizations are possible and hence that the final result of the processing depends on the particular parameterization chosen.
Recently, researchers with background in geometric modelling, signal processing, approximation theory and differential geometry have put a lot of effort in the advancement of Digital Geometry Processing providing the basis of a new mathematical framework for the construction of efficient DGP algorithms for polygonal geometry for smoothing, enhancement, editing, compression, transmission and so on. In particular, Schröder and Swelden [16, 72], indicate the semi-regular meshes as the fundamental representation on which new DGP algorithms can be developed. First, semi-regular meshes provide semi-regular sampling of the surface, increasing the simplicity of the design of analysis and processing tools. Second, semi-regular meshes are the basis of several multiresolution techniques for polygonal geometry. In fact, this kind of mesh is obtained through a process of subdivision, that, in general, can be seen as an up-sampling of the original surface plus a displacement of the added samples in order to obtain a surface with more details. In other words, subdivision naturally provides a hierarchy of levels of resolution of the models (from the coarse to the fine one) that can be used to build efficient scalable algorithms for geometry processing.

In the following we give a detailed description of subdivision methods for surfaces generation and their relationships with semi-regular meshes. Then, we describe a multiresolution technique based on subdivision that it is the base of our watermarking algorithms: wavelet decomposition of meshes. At the end, some basic notions about mesh parameterization and remeshing, two important elements of DGP algorithms, are presented. Such concepts will be useful in the next Chapter to describe our approach to mesh watermarking.

4.2 Subdivision Methods

The techniques to create curves and surfaces through a process of subdivision are relatively recent. The growing interest about subdivision methods in geometric modelling is motivated by the fact that subdivision bridges the gap between discrete and continuous representations of curves and surfaces.

Before describing subdivision methods for surfaces we present an example of generation of curves through subdivision by describing Chaikin’s algorithm [73]. Let assume that the initial curve $P^0$ is represented by the sequence of vertices $\{p_0^0, p_1^0, \ldots, p_n^0\}$. The superscript of the points indicates the level of subdivision of the curve. The zero level corresponds to the original control polygon. At each subdivision step Chaikin’s scheme creates two new vertices between each
4.2. Subdivision Methods

consecutive ones using the following subdivision rules:

\[
q_{2i}^{k+1} = \frac{3}{4}p_i^k + \frac{1}{4}p_{i+1}^k
\]

\[
q_{2i+1}^{k+1} = \frac{1}{4}p_i^k + \frac{3}{4}p_{i+1}^k
\]

where \(q_{i}^{k+1}\) are the new generated points at level of subdivision \(k + 1\). After the generation of the new points the old vertices are discarded and only the new points \(q_{i}^{k+1}\) form the curve at level \(k + 1\). Figure 4.1 shows the result of the first step of subdivision on a curve formed by 4 points. By iteratively applying (4.1) we obtain the curves \(P^1, P^2\) and so on. When \(k\) approaches infinity a continuous curve is generated. This limit curve is indicated with \(P^\infty\). The geometric properties of the limit curve depend on the specific subdivision rules used to generate it. In particular, Chaikin’s scheme generates a quadratic B-Spline. A subdivision scheme is called interpolating, if the limit curve interpolates the points of the initial control polygon, i.e. if after each subdivision step both the old and the new generated vertices belong to the curve, and the old vertices remain in the same position. If after each subdivision step the old vertices are discarded or moved according to some rules the limit curve does not interpolate the initial control points thus resulting in an approximating subdivision scheme. The Chaikin’s scheme belongs to this last category since, after each subdivision step, the old vertices are discarded. Hence, the initial vertices cannot be part of the limit curve.

To conclude, we give an example of interpolating scheme for curve generation: the 4-point algorithm [74]. This scheme uses the four nearest points to create a new point. The subdivision rules are:

\[
q_{2i}^{k+1} = p_i^k
\]

\[
q_{2i+1}^{k+1} = \left( \frac{1}{2} + w \right) \left( p_i^k + p_{i+1}^k \right) - w \left( p_{i-1}^k + p_{i+2}^k \right)
\]
The first equation in (4.2) means that the original points do not change (in fact, this scheme is interpolating). The second equation is for creating the new points between $p_i^k$ and $p_{i+1}^k$. The weight $w$ is called tension parameter and permits to control the behavior of the interpolation. When $w = 0$ the resulting curve is a linear interpolation of the starting points. For $w = 1/16$ a cubic interpolation is achieved. For $0 < w < 1/8$ the resulting curve is always $C^1$.

4.2.1 Subdivision Surfaces

Subdivision methods for surfaces generation work in the same way of those for curves: starting from an initial control mesh ($M^0$) the subdivision scheme is iteratively applied to obtain a finer mesh ($M^k$) at level of subdivision $k$. Different subdivision schemes generate surfaces with different geometric properties. Here, we present some of the major stationary schemes. A subdivision scheme is said to be stationary if the subdivision rules do not depend on the level of subdivision. Other classes of subdivision schemes, such as variational subdivision, are not discussed here.

In these last years several stationary schemes have been developed. A classification of them can be achieved by considering the basic properties of each method:

Type of mesh. Subdivision schemes work on polygonal meshes. The initial control mesh can be of two types: triangular or quadrilateral. Hence, two kinds of schemes are possible: those specific for triangular meshes and those specific for quadrilateral meshes.

Type of refinement rule. There are two main approaches to refine the polygonal net. The first one is based on face splitting and the second one is based on vertex splitting. The first schemes are referred to as primal and the second ones are referred to as dual. In the primal schemes each face is subdivided into four new faces (1-to-4 split) by inserting a new vertex for each edge of the coarse mesh, retaining old vertices, and then connecting the new inserted vertices together. In the dual case, for each old vertex, several new vertices are created, to be more specific, one for each face adjacent to the vertex. Then, for each old face, a new face is created inside the new one. For quadrilateral can be done in such a way that the refined mesh has only quadrilateral faces. In the case of triangles, vertex split (dual) schemes result in non-nesting hexagonal tilings. In this sense quadrilateral tilings are
special: they support both primal and dual subdivision schemes naturally. In Figure 4.2 the primal and dual schemes for triangular and quadrilateral mesh are shown. For clarification, the dual mesh of a given one, it is defined as those mesh obtained by connecting the centers of the adjacent faces with an edge.

**Approximation vs Interpolation.** As previously stated, a subdivision method can produce a curve, or a surface, that is an interpolation of the initial curve/surface, or that is an approximation of the initial control net. By considering this property we can classify the subdivision schemes in interpolating and approximating. Primal schemes can be approximating or interpolating. Dual schemes are always approximating for their definition. Interpolation is an attractive property for several reasons. First, the control points are also points of the limit surface. Second, many algorithms can be considerably simplified improving computational efficiency. Additionally, if needed, the refinement of the mesh can be tuned locally by perform a certain number of subdivision steps in some parts of the mesh and other numbers of subdivision in other parts. Nevertheless, the quality of the surfaces generated by interpolating schemes is lower than the one of approximating schemes. Also, the convergence to the limit surface is typically slower with respect to approximating schemes.

**Smoothness.** The smoothness of the limit surface, i.e. of the surface obtained applying the subdivision scheme an infinite number of times, is measured by its continuity properties ($C^1$, $C^2$, etc.). The continuity properties can be used by further classify the subdivision scheme.

Table 4.1 summarizes the basic properties of several subdivision schemes. The $\sqrt{3}$ scheme of Kobbelt et al. [75] differs from the schemes depicted in Figure 4.2 since it produces a 1-to-9 split every two subdivision steps instead of a 1-to-4 split every subdivision step.

### 4.2.2 Subdivision schemes

In this section we present some classical subdivision schemes: the approximating Loop [81] scheme for triangular meshes, the interpolating Butterfly [76] scheme for triangular meshes and the approximating Catmull-Clark [78] scheme for quadrilateral meshes. All of these schemes are primal. The subdivision rules of such
schemes are visualized as masks of weights used to compute the new vertices. The notation used for the description of the schemes is:

**Regular and Extraordinary vertices.** As it is possible to notice in Figure 4.2 subdivision schemes defined on triangular meshes create new vertices of valence 6 in the interior of the net, while subdivision schemes for quadrilateral meshes result in vertices with valence 4. This fact relies on the definition of *semi-regular meshes*, where all the regular vertices have valence 6 (for a triangular meshes) except the vertices of the initial control mesh, that can have any valence. For a quadrilateral mesh the regular vertices have valence 4. The non-regular vertices are called *extraordinary vertices*. 

---

### Table 4.1: Classification of major subdivision schemes.

<table>
<thead>
<tr>
<th>Subdivision Scheme</th>
<th>Basic Properties</th>
<th>Continuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Butterfly [76, 77]</td>
<td>interpol. triangle primal</td>
<td>$C^1$</td>
</tr>
<tr>
<td>Catmull-Clark [78]</td>
<td>approx. quadrangle primal</td>
<td>$C^2$</td>
</tr>
<tr>
<td>Doo-Sabin [79]</td>
<td>approx. quadrangle dual</td>
<td>$C^1$</td>
</tr>
<tr>
<td>Kobbelt (quad.) [80]</td>
<td>interpol. quadrangle primal</td>
<td>$C^1$</td>
</tr>
<tr>
<td>Kobbelt ($\sqrt{3}$) [75]</td>
<td>approx. triangle dual</td>
<td>$C^2$</td>
</tr>
<tr>
<td>Loop [81]</td>
<td>approx. triangle primal</td>
<td>$C^2$</td>
</tr>
</tbody>
</table>
Odd and Even vertices. For primal schemes the vertices of the coarser mesh are also vertices of the refined mesh. For any subdivision level, we referred to the vertices created at that level as odd vertices. The vertices inherited from the previous level of subdivision are called even. This notation comes analyzing the one-dimensional case, when vertices of the control polygon can be enumerated sequentially, thus resulting in odd numbers for the new inserted vertices [82].

Boundaries and Creases. The subdivision rules have to be adapted for the boundary of the mesh. Typically, these rules are defined in such a way that the boundary curve of the limit surface does not depend on any interior control vertices, and is smooth or piecewise smooth ($C^1$ or $C^2$ continuous). The same rules can be used to preserve sharp features of the initial control mesh by tagging the interior edges of interest as boundary edges. Such tagged edges are called creases.

Masks. Often, the subdivision rules of a specific scheme are represented using masks. The mask is a picture that shows which control points are used to compute the new ones and the relative weights. The new vertices is showed in the mask by a black dot. For example, referring to the masks of the Loop scheme of Figure (4.3), the interior odd vertex $v$, i.e. the new inserted vertex, is computed by centering the relative mask over it obtaining:

$$q^{j+1}(v) = \frac{3}{8}p(v_1^j) + \frac{3}{8}p(v_2^j) + \frac{1}{8}p(v_3^j) + \frac{1}{8}p(v_4^j)$$

(4.3)

where $v_1$ and $v_2$ are the immediate neighbors of the new vertex $v$, and $v_3$ and $v_4$ are the other two vertices of the triangles that share the considered edge. The mask for the even vertices, that are present only in the approximating schemes, are used to change the position of the existing vertices.

Loop scheme

The Loop scheme is an approximating scheme for triangular meshes proposed by Charles Loop [81]. This scheme is based on the three-directional box spline, which produces $C^2$-continuous surface over regular meshes. The Loop scheme produces surfaces that are $C^2$-continuous everywhere except at extraordinary vertices where they are $C^1$-continuous. The masks for the Loop scheme are shown in Figure 4.3. As proposed by Loop the parameter $\beta$ can be computed as $\beta = \frac{1}{n} \left( \frac{5}{8} - \left( \frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{n} \right)^2 \right)$. 

4.2. Subdivision Methods
Calculating the surface tangent vectors for the Loop scheme is particularly simple. The formula for an interior vertex is:

\[
\hat{\mathbf{t}}_1 = \sum_{i=0}^{k-1} \cos \left( \frac{2\pi}{k} \right) p_{i,1} \\
\hat{\mathbf{t}}_2 = \sum_{i=0}^{k-1} \sin \left( \frac{2\pi}{k} \right) p_{i,1}
\]

(4.4)

where \(p_{i,1}\) are 1-ring neighborhood of the considered vertex. This formula can be applied to the control points at any subdivision level. When the tangents of the surface are calculated determining the surface normals at the considered point is trivial (\(\hat{n} = \hat{t}_1 \times \hat{t}_2\)). For the computation of the tangents at a boundary vertex and across the boundary/creases we refer to [82]. Also the limit position (\(\lim_{j \to \infty} p^j\)) of a control point can be computed in a easy way. For interior vertices, the mask for computing the limit value is the same as the mask for computing the value on the next level with \(\beta\) replaced by \(\chi = \frac{1}{(3/8)^{\beta+n}}\). For boundary and crease vertices the formula is the same for even boundary vertices, but with different weights. In particular:

\[
d_0^\infty = \frac{1}{5} p_{0,1} - \frac{3}{5} p_0 + \frac{1}{5} p_{1,1}
\]

(4.5)

where \(p_{0,1}\) and \(p_{1,1}\) are the immediate neighbors of the vertex \(p_0\).
4.2. Subdivision Methods

The Butterfly scheme was first proposed by Dyn et al. [76]. The limit surface is not $C^1$-continuous at extraordinary points of valence $k = 3$ and $k > 7$ [83], while it is $C^1$ on regular meshes. A modification of the original Butterfly scheme was proposed by Zorin et al. [77]. This variant guarantees $C^1$-continuous surface for arbitrary meshes (see [83] for a proof). The masks are shown in Figure 4.4. The coefficients $s_i$ for the extraordinary vertices are $\{s_0 = \frac{5}{12}, s_1 = -\frac{1}{12}, s_2 = -\frac{1}{12}\}$ for $k = 3$, $\{s_0 = \frac{3}{8}, s_1 = 0, s_2 = -\frac{1}{8}, s_3 = 0\}$ for $k = 4$, and, the general formula for $k > 5$ is:

$$s_i = \frac{1}{k} \left( \frac{1}{4} + \cos \left( \frac{2i\pi}{k} \right) + \frac{1}{2} \cos \left( \frac{4i\pi}{k} \right) \right)$$  \hspace{1cm} (4.6)

The tangent vectors at extraordinary interior vertices can be computed with the same formula (4.4) for the Loop scheme. For regular vertices the formula is more complex and involves all the control points in a 2-ring neighborhood of the considered vertex [82]. The limit position of the control points does not need to be determined since the scheme is interpolating.

Catmull-Clark scheme

The Catmull-Clark scheme [78] is based on the tensor product bicubic spline. This approximating scheme for quadrilateral meshes is $C^2$-continuous everywhere ex-
cept at extraordinary vertices, where they are $C^1$, like as the Loop scheme. The tangent plane continuity of the scheme was analyzed by Ball and Storry [84], and the $C^1$ continuity by Peters and Reif [85]. The standard masks of the Catmull-Clark scheme are shown in Figure 4.6. The value of $\alpha$ and $\beta$ can be chosen from a wide range of values [82]; $\alpha$ is the coefficient of the central vertex, i.e. $\alpha = 1 - \gamma - \beta$. In the case of extraordinary vertices Catmull and Clark suggest $\beta = \frac{3}{2k}$ and $\gamma = \frac{1}{4k}$.

### 4.2.3 Subdivision Analysis

The analysis of subdivision schemes concerns the convergence and the continuity of the limit surfaces. A subdivision scheme can be expressed as a matrix of weights that encodes the subdivision rules. Representing subdivision with matrices is very useful to analyze the properties of the generated surface. In general we can write:

$$
\begin{bmatrix}
q_0 \\
q_1 \\
\vdots \\
q_m
\end{bmatrix} =
\begin{bmatrix}
w_{00} & w_{01} & w_{02} & \cdots & w_{0m} \\
w_{10} & w_{11} & w_{12} & \cdots & w_{1m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
w_{n0} & w_{n1} & w_{n2} & \cdots & w_{nm}
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_1 \\
\vdots \\
p_m
\end{bmatrix}
$$

(4.7)

where $p_i$ are the vertices before the subdivision (properly arranged in the vector) and $q_i$ are the new points. To analyze the particular scheme is not necessary to consider the entire set of vertices but it is sufficient to restrict the study to a fixed neighbours of a generic vertex:

$$
q^{k+1} = Sp^k
$$

(4.8)

where $p^k$ is the vector of considered vertices at level $k$ and $q^{k+1}$ are the new vertices at level $k+1$. The matrix $S$, called subdivision matrix, is a square sub-matrix of the
4.2. Subdivision Methods

![Subdivision Methods Diagram](image)

Figure 4.6: Catmull-Clark subdivision scheme.

The matrix of weights in (4.7) which completely characterizes the subdivision scheme. In particular, by taking into account the eigenvectors \( (x_i) \) and the eigenvalues \( (\lambda_i) \) of \( S \) it is possible to write:

\[
q^j = S^j p^0 = \sum_i (\lambda_i)^j a_i x_i
\]

(4.9)

where the coefficients \( a_i \) are used to express \( p^0 \) in the eigenvector’s basis, i.e. \( p^0 = \sum_i a_i x_i \). Hence, in order to guarantee convergence all the \( \lambda_i \) have to be less than 1. In particular \( \lambda_0 \) has to be 1 to guarantee invariance with respect to translation and rotation [82]. Further analysis of the eigenvalues and eigenvectors of \( S \) allow to compute the limit position reached by the considered point when the level of subdivision approaches infinity, the tangents of the surface at the considered point and continuity properties of the surfaces away the extraordinary point. For further details about subdivision analysis we refer to the excellent course on subdivision by Zorin and Schröder [82]. For a more rigorous treatment of the general theory of \( C^k \) continuity of subdivision schemes we refer to [86, 87, 83, 88].
4.3 Multiresolution Techniques

Multiresolution is the basic element to build scalable algorithms, i.e. algorithms that are able to analyze and process signals at different spatial/frequency scales. In the cases of geometry processing, multiresolution techniques naturally separate the model into different level of details, from coarse to fine. Also, typically, multiresolution improves the speed of the algorithms. Multiresolution analysis and synthesis for polygonal meshes can be achieved using two mathematical frameworks: wavelets and lifting scheme. Here, we are interested in wavelet decomposition of semi-regular meshes since our watermarking algorithm relies on this multiresolution analysis/synthesis tool. For this reason in the following we give a detailed description of wavelet analysis of meshes referring to the literature for the lifting scheme [89, 90, 91].

Basics of Wavelets Theory

Wavelets are a mathematical tool for hierarchically decomposing functions. In particular, any kind of function can be decomposed in a base function plus details that range from broad to narrow. Hence, wavelets allow the analysis and processing of signals, image and surfaces at different spatial/frequency scales. Wavelet theory had been primarily developed in signal processing [92, 93] and approximation theory [94] and then it has been rapidly applied to many problems in Computer Graphics. Wavelets are used in Computer Graphics for fast global illumination computation [95, 96], for hierarchical geometric modelling by adding flexibility and simplicity to the editing process [97, 98], to solve efficiently physical simulations in computer animation [99], and so on.

The basic theory of multiresolution analysis with wavelets, as formulated by Mallat [92], states that the fundamental ingredients for multiresolution analysis are an infinite chain of nested linear function spaces $V^0 \subset V^1 \subset V^2 \subset \ldots$ and an inner product $\langle f, g \rangle$ defined on any pair of functions $f,g \in V^j$. Intuitively, $V^j$ contains functions at level of details $j$, for $j < \infty$. The inner product is used to define the orthogonal complement spaces $W^j$:

$$W^j := \{ f \in V^{j+1} | \langle f, g \rangle = 0, \forall g \in V^j \}.$$  \hspace{1cm} (4.10)

Typically, the inner product used on the real line is:

$$\langle f, g \rangle := \int_{-\infty}^{+\infty} f(x)g(x)dx$$  \hspace{1cm} (4.11)
4.3. Multiresolution Techniques

with \( f, g \in V^j \). This definition requires that both \( f \) and \( g \) are integrable over their respective domain.

The standard terminology refers to the bases of the spaces \( V^j \) as scaling functions \( (\phi(x)) \) and to the bases of spaces \( W^j \) \( (\psi(x)) \) as wavelets. By considering the real line, the sequence of nested linear spaces can be obtained if the scaling functions satisfy the, so-called, two-scale relation:

\[
\phi(x) = \sum_i c_i \phi(2x - i) \quad (4.12)
\]

for some fixed constants \( c_i \). The (4.12) guarantees that the linear spaces defined as:

\[
V^j := \text{span} \{ \phi(2^j x - i) | i \in \mathbb{Z} \} \quad (4.13)
\]

are nested. In other words, the nested spaces are generated by translations and dilations of a single refinable function \( \phi(x) \). A scaling function is said to be refinable if each scaling function of the level \( j - 1 \) can be expressed as a linear combination of (finer) scaling functions of the level \( j \). From the definition (4.13) the scaling functions \( \{ \phi_i^j(x) \} \) span the space \( V^j \). In a similar fashion, the orthogonal complement spaces of \( V^j \), i.e. \( W^j \), are generated by wavelet functions \( \psi_i^j(x) \) that satisfy the two-scale relation.

From the previous definitions, a generic function of the level \( j + 1 \) can be expressed as:

\[
f^{j+1}(x) = \sum_{i=0}^{n_j} c_i^j \phi_i^j(x) + \sum_{i=0}^{m_j} d_i^j \psi_i^j(x) \quad (4.14)
\]

for some coefficients \( c_i^j \) and \( d_i^j \). In other words, the function \( f^{j+1} \in V^{j+1} \) can be decompose in a low resolution part, i.e. \( f^j(x) = \sum_{i=0}^{n_j} c_i^j \phi_i^j(x) \), plus some details encoded by the wavelet coefficients \( d_i^j \). The multiresolution analysis/synthesis can be easily achieved using the matrix notation. Let \( \Phi^j(x) = \left[ \phi_0^j(x), \phi_1^j(x), \ldots, \phi_{n_j}^j(x) \right] \) and \( \Psi^j(x) = \left[ \psi_0^j(x), \psi_1^j(x), \ldots, \psi_{m_j}^j(x) \right] \) the refinability properties tell us that two matrices \( \mathbf{P}^j \) and \( \mathbf{Q}^j \) exist such that:

\[
\Phi^{j-1}(x) = \Phi^j(x) \mathbf{P}^j
\]

\[
\Psi^{j-1}(x) = \Phi^j(x) \mathbf{Q}^j \quad (4.15)
\]

By using \( \mathbf{P}^j \) and \( \mathbf{Q}^j \) equation (4.14) can be rewritten as:

\[
\mathbf{C}^{j+1} = \mathbf{P}^j \mathbf{C}^j + \mathbf{Q}^j \mathbf{D}^j \quad (4.16)
\]
where $C^{j+1}$ is a vector of coefficients such that $f^{j+1}(x) = \sum_i \phi_i^{j+1}(x) C_i$, $C_j$ is the vector of coefficients of the low resolution part, and $D_j$ are the details coefficients. $P_j$ and $Q_j$ are called synthesis filters. The analysis filters, also called decomposition filters, can be obtained by inverting the synthesis filters:

$$
\begin{bmatrix}
A^j \\
B^j
\end{bmatrix} = \begin{bmatrix}
P_j & Q_j
\end{bmatrix}^{-1}
$$

The procedure to split $C^j$ into a low-res part $C^{j-1}$ plus a detail part $D^{j-1}$ can be applied recursively building a multiresolution hierarchy $C^0, C^1, \ldots, C^{j-1}$ and details $D^0, D^1, \ldots, D^{j-1}$. This recursion is known as filter banks. The transformation of the original coefficients $C_j$, representing the functions at the full level of resolution, to the sequence \{C^0, D^0, D^1, \ldots, D^{j-1}\} is called wavelet transform.

### Wavelet Analysis and Synthesis of Meshes

Starting from the general theoretical framework presented above, Lounsbery [100] has developed a multiresolution wavelet analysis for polygonal meshes based on subdivision surfaces. The basic idea of Lounsbery was to start with defining refinable scaling functions over a two-manifold polygonal domain of arbitrary topological type using subdivision. Then, such functions can be used to compute wavelets for semi-regular meshes after the definition of a suitable inner product. Lounsbery has proof that for any continuous and uniformly convergent subdivision procedure there exist continuous scalar-valued scaling functions $\phi_i^j(x)$, $x \in M^0$ such that:

$$S(x) = \sum_i v_i^j \phi_i^j(x)$$

where $S(x)$ is the limit surface, $v_i^j$ are the vertices of the mesh at level of subdivision $j$ and $M^0$ is the initial control mesh. In this case the level of details corresponds to the level of subdivision. Notice that $x$ is defined over the initial control mesh, furthermore a proper parameterization that associates points over $M^0$ to points on the limit surface $S(x)$ must be envisaged. Lounsbery proposed to track the points through the subdivision process using barycentric coordinates. The scaling functions defined through subdivision are refinable since it is possible to write:

$$\Phi^{j-1}(x) = \Phi^j(x) P_j = \begin{bmatrix}
\mathcal{O}^j(x) & \mathcal{N}^j(x)
\end{bmatrix} \begin{bmatrix}
O^j \\
N^j
\end{bmatrix}$$

indicating with $\mathcal{O}^j(x)$ the scaling functions associated with the “old” vertices of the mesh $M^j$ and with $\mathcal{N}^j(x)$ the scaling functions associated with the “new”
vertices of the mesh $M^{j+1}$. $O^j$ and $N^j$ represent the portions of the subdivision matrix which weight the “old” and the “new” vertices, respectively. In the case of piecewise linear subdivision, leading to polyhedral surfaces, the scaling functions $\phi_i^j(x)$ are hat functions, i.e. $\phi_i^j(x)$ associates the value 1 at vertex $i$ of the level $j$ and zero at all the other vertices (see Figure 4.7). The support of the hat function $\phi_i^j(x)$ is determined by the level of subdivisions. Hence, such scaling functions have local, compact support. Since the scaling functions so defined are refinable, they can be used to build a sequence of nested linear spaces that depend on the mesh $M^0$:

$$V^j(M^0) := \text{Span}(\Phi^j(x)) \quad (4.20)$$

In order to construct wavelets an inner product must be defined. A variant of the standard inner product (4.11) is used:

$$\langle f, g \rangle := \sum_{\tau \in \mathcal{T}(M^0)} \frac{1}{\mathcal{A}(\tau)} \int_{x \in \tau} f(x)g(x)dx \quad (4.21)$$

where $\mathcal{A}(\tau)$ indicates the area of the triangle $\tau$ and $\mathcal{T}(M^0)$ denotes the set of triangular faces of $M^0$. In this way this inner product is independent of the geometric positions of the vertices $M^0$ since triangles of different area are weighted equally. By letting $f(x) = \sum_i f_i^j \phi_i^j(x)$ and $g(x) = \sum_i g_i^j \phi_i^j(x)$, from the bilinearity of the inner product the (4.21) can be express in the following matrix form:

$$\langle f, g \rangle = \mathbf{g}^T \mathbf{I}^j \mathbf{f} \quad (4.22)$$

where $\mathbf{f}$ and $\mathbf{g}$ are the vectors of the coefficients $f_i^j$ and $g_i^j$ respectively, and $\mathbf{I}^j$ is the matrix of elements $(\mathbf{I}^j)_{i,i'} = \langle \phi_i^j, \phi_{i'}^j \rangle$, i.e:

$$\mathbf{I}^j = \int_{x \in M^0} (\Phi^j(x))^T \Phi^j(x)dx \quad (4.23)$$
It is possible to demonstrate that a recursive relation between $I^j$ and $I^{j+1}$ exists. Such relationship depends on the subdivision matrix. Hence, the inner product (4.21) can be computed by solving a linear system without resorting to numerical integration.

At this point it is possible to construct the subdivision wavelets, i.e. a set of functions that span the orthogonal complement space of $V^j(M^0)$, that is $W^j(M^0)$. A basis for $W^j(M^0)$ can be build by considering the projection of $N^{j+1}(x)$ over it. In fact, $\Phi^j(x)$ and $N^{j+1}(x)$ together span $V^{j+1}(M^0)$ if, and only if, the matrix $O^j$ is invertible. This is true for most primal subdivision methods. Expressed in matrix form, this means to solve:

$$N^{j+1}(x) = \Psi^j(x) + \Phi^j(x)\alpha^j$$

for some coefficients $\alpha^j$. Such coefficients can be calculated as the solution of the linear system obtained taking the inner product of both sides of the equation (4.24):

$$\langle \Phi^j(x), \Phi^j(x) \rangle \alpha^j = \langle \Phi^j(x), N^{j+1}(x) \rangle = (P^j)^T(\Phi^{j+1}(x), N^{j+1}(x))$$

where $\langle \Psi^j(x), \Psi^j(x) \rangle$ is simply the matrix $I^j$ and the matrix $\langle \Psi^{j+1}(x), N^{j+1}(x) \rangle$ is the sub-matrix of $I^{j+1}$ formed by the columns that correspond to the elements of $N^{j+1}(x)$.

The above results can be used to build a filters bank for wavelet analysis/synthesis of semi-regular meshes. The synthesis filters are:

$$[P^j \quad Q^j] = \begin{bmatrix} O^j & -O^j\alpha^j \\ N^j & I - N^j\alpha^j \end{bmatrix}$$

The analysis filters are obtained by inverting the (4.26), in the same way of (4.17). For interpolating schemes the matrix $O^j$ coincide with the identity matrix; in this case the analysis and synthesis filters are greatly simplified:

$$[P^j \quad Q^j] = \begin{bmatrix} I & -\alpha^j \\ N^j & I - \alpha^j N^j \end{bmatrix} [A^j \quad B^j] = \begin{bmatrix} I - \alpha^j N^j & \alpha^j \\ -N^j & I \end{bmatrix}$$

For further details about scaling functions built through subdivision, the computation of $I^j$, the construction of subdivision wavelets and the filters banks for analysis/synthesis of semi-regular meshes we refer to the Lounsbery’s PhD thesis [100].
Our watermarking algorithm is based on the wavelet analysis/synthesis here described. As suggested by Lounsbery the filter banks can be considerably simplified to obtain “lazy” wavelets that can be computed in linear time for any subdivision schemes. In Section 5.2 we discuss how to efficiently implement such multiresolution techniques.

4.4 Mesh Parameterization

Mesh parameterization is the process of mapping the vertices of a triangular mesh over a planar domain. In the general case, a parameterization \( \phi \) of a triangulated two-manifold \( T \) over a parameter domain \( \Omega \subset \mathbb{R}^k \) is a homeomorphism between this domain and the surface of \( T \) (\( \Omega_T \)):

\[
\phi : \Omega \rightarrow \Omega_T
\]  (4.28)

Differential geometry tell us that such a homeomorphism and the inverse parameterization \( \psi = \phi^{-1} \) exist if and only if \( \Omega \) and \( \Omega_T \) are topologically equivalent. Hence, a triangle mesh can only be parameterized over a planar domain when it is topologically equivalent to a disk, i.e. when it has a boundary and is of genus\(^1\) 0. The partition of the mesh to be parameterized into patches that are homeomorphic to a disk can be achieved in several ways [48, 101, 102].

It is important to underline that only developable surfaces can be parameterized without distortions. Therefore, the proposed parameterization techniques try to preserve as good as possible one or more properties of the surfaces, such as angles, area or to minimize geometric stretch.

Mesh parameterization has many applications in various fields of science and engineering, including scattered data fitting, repair of CAD models, texture mapping and remeshing. In this section we describe some basic concepts of surface parameterization. For further details we refer to the relative literature.

4.4.1 Differential Geometry Background

Let us assuming that a surface \( S \) has the parametric form \( \mathbf{x} \), defined as in (3.2):

\[
\mathbf{x}(u, v) = (X(u, v), Y(u, v), Z(u, v))
\]  (4.29)

with \( u \) and \( v \) in some domain in \( \mathbb{R}^2 \).

\(^1\)The genus is the number of handles of a manifold. A sphere has genus 0, a torus genus 1, and so on.
Such representation is called *regular* if the functions \(X(u,v), Y(u,v)\) and \(Z(u,v)\) are differentiable and the vectors:

\[
\begin{align*}
\mathbf{x}_1 &= \frac{\partial \mathbf{x}}{\partial u}, \\
\mathbf{x}_2 &= \frac{\partial \mathbf{x}}{\partial v}
\end{align*}
\] (4.30)

are linearly independent at every point. Many properties of the surface \(S\) are given by its *first fundamental form*, a quadratic form which is the square of the element of arc of a curve in \(S\):

\[
ds^2 = \begin{pmatrix} du & dv \end{pmatrix} I \begin{pmatrix} du \\ dv \end{pmatrix}
\] (4.31)

where \(I\) is a symmetric matrix defined as:

\[
I = \begin{pmatrix} g_{11} & g_{12} \\
g_{21} & g_{22} \end{pmatrix}
\] (4.32)

with \(g_{\alpha\beta} = \mathbf{x}_\alpha \cdot \mathbf{x}_\beta\). Often the matrix \(I\) is itself referred to as the first fundamental form. If \(I\) has a strictly positive determinant the surface is regular. Consider now a mapping \(f\) from \(S\) to a second surface \(S^*\). We can define the parameterization \(\mathbf{x}^* = f \circ \mathbf{x}\) of \(S^*\) so that the coordinates of any image point \(f(p) \in S\) are the same of the corresponding pre-image point \(p \in S\). We say that \(f\) is *allowable* if the parameterization \(\mathbf{x}^*\) is regular. Using this notions is possible to characterize different kinds of mappings.

**Isometric Mappings** An allowable mapping from \(S\) to \(S^*\) is *isometric* or *length-preserving* if the length of any arc on \(S^*\) is the same as the pre-image on \(S\). For an isometric mapping the coefficients of the first fundamental forms are the same, i.e. \(I = I^*\).

**Conformal Mappings** An allowable mapping from \(S\) to \(S^*\) is *conformal* or *angle-preserving* if the angle of intersection of every pair of intersecting area on \(S^*\) is the same as that of the corresponding pre-images on \(S\) at the corresponding point. In this case the coefficient of the respective first fundamental forms are proportional, i.e. \(I = \eta(u,v)I^*\) for some scalar function \(\eta(.)\).

**Equiareal Mappings** An allowable mapping from \(S\) to \(S^*\) is *equiareal* if every part of \(S\) is mapped onto a part of \(S^*\) with the same area. For an equiareal mapping the determinants of the respective first fundamental forms are equal, i.e. \(\det I = \det I^*\).
It is important to notice that every isometric mapping is also conformal and equiareal. The contrary is also valid, a conformal and equiareal mapping is also isometric.

**Conformal and Harmonic Mapping**

Harmonic mapping is another kind of mapping related to conformal mapping. Consider the case of mapping an arbitrary planar region $S$ to a plane. Such mapping can be seen as a function of a complex variable $z = u + iv$, i.e. $\omega = f(z)$, with $\omega = x + iy$. If $f$ is conformal it satisfies the Cauchy-Riemann equation:

\[
\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}
\]  

(4.33)

From this equation is possible to derive the two Laplacian equations:

\[
\Delta u = 0, \quad \Delta v = 0
\]

(4.34)

where $\Delta$ is the Laplacian operator, i.e. $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. Any mapping which satisfies the two Laplacian equations (4.34) is an harmonic mapping. Since conformal mapping satisfies the (4.33) it is also harmonic. To summarize an isometric mapping is also conformal and a conformal mapping is also harmonic.

Harmonic mappings are widely used in mesh parameterization since they are easy to compute and guarantee to be one-to-one for convex regions. Typically, a harmonic mapping is computed by choosing a suitable boundary map and solving a linear elliptic partial differential equation (PDE) which can be approximated by various method (e.g. finite differences) thus resulting in a linear system of equations. A disadvantage of the harmonic mappings is that they are not conformal, and so they do not preserve angles. Another disadvantage is that, in general, the inverse map of an harmonic mapping is not harmonic.

### 4.4.2 Parameterization methods

The task of mesh parameterization is to map a given disc-like surface $S \subset \mathbb{R}^3$, in particular a triangular mesh $S_T$, into the plane. To be more specific the goal is to find a piecewise linear mapping $f : S_T \rightarrow S^*$ where $S^* \subset \mathbb{R}^2$ is a suitable polygonal domain. Such mapping is uniquely determined by the images $f(v) \in \mathbb{R}^2$ of the mesh vertices. The ideal parameterization is an isometric mapping since it preserves angles, areas and lengths of the given mesh. But, as previously stated, isometric mappings exist only in some very special cases (e.g. developable surface mapped on a plane). In the general case of a triangular mesh the existing
approaches attempt to find a mapping without angles distortions (conformal), without area distortions (equiareal) or minimizing a combination of angle distortion and area distortion. In the following we give a panoramic of harmonic, conformal and equiareal parameterization methods.

**Discrete Harmonic Mappings**

One of the earliest method for mapping a disk-like polygonal surface was to approximate the harmonic map using the finite elements theory. This method was first introduced by Eck et al. [45] and consists of two steps. First, the boundary mapping is fixed, i.e. $f_{\partial S_T} = f_0$, by mapping the boundary of $S_T$ homeomorphically to some polygon in the plane. Second, the piecewise linear mapping $f : S_T \rightarrow S^*$ which minimizes the Dirichlet energy:

$$E_D = \frac{1}{2} \int_{S_T} \|\nabla f\|^2$$

subject to the condition $f_{\partial S_T} = f_0$ is calculated. Solving the minimization problem (4.35) reduces to solve a linear system of equations [103].

Another linear method to build discrete harmonic maps is based on the solution of the linear system:

$$f(v_i) = \sum_{j \in v_1(i)} \lambda_{ij} f(v_j)$$

with

$$\lambda_{ij} = \frac{w_{ij}}{\sum_{j \in v_1(i)} w_{ij}}$$

(4.37)

The weights $w_{ij}$ are associated to each edge of the mesh connecting the vertex $v_i$ with the vertex $v_j$; $v_1(i)$ is the set of indices of the 1-ring of $v_i$. The weights $\lambda_{ij}$ are the corresponding normalized weights (see 4.37). If $\lambda_{ij}$ are positive each interior mapped vertex $f(v_i)$ is a convex combination of its neighbors, and hence $f(v_i)$ lies in the convex hull formed by its neighbors. Any piecewise linear mapping of this kind is called *convex combination map*. It is possible to demonstrate that, under certain conditions, such parameterization is one-to-one and harmonic [104, 105].

Several choices for the weights $w_{ij}$ are possible. The special cases in which all the weights are uniform ($w_{ij} = 1$) is called *barycentric mapping* (Tutte [104]). The *shape-preserving* method of Floater [106] is another example of convex combination mapping. More recently, Floater [107] proposes an alternative to its shape-preserving method by constructing a convex combination mapping based on the concept of *mean value coordinates*, that is a generalization of the barycentric coordinates.
Discrete Conformal Mappings

The main drawback of methods based on harmonic mapping is that the boundary mapping must be specified in advanced, and preferably it has to be a convex polygon. This constraint may generate high distortions near the boundary, in particular if the boundary of the surface is not convex.

Hormann and Greiner [108] propose a method based on measuring the conformality of a (non-degenerate) bivariate linear function \( g : \mathbb{R}^2 \to \mathbb{R}^2 \) by the condition number of its Jacobian \( J \) with respect to the Frobenius norm:\footnote{The Frobenius norm of a matrix \( A \) is defined as the square root of the sum of the absolute squares of its elements.}

\[
E_M(g) = \| J \|_F \| J^{-1} \|_F
\]

(4.38)

The conformality of the piecewise linear mapping \( f \) is then defined as:

\[
E_M(f) = \sum_{T \in S} E_M(f|_T)
\]

(4.39)

where \( f|_T \) is the atomic map of the piecewise linear map \( f \). In other words, \( f|_T \) is the map of a single triangle over the parameter plane. The energy \( E_M(f) \) is called MIPS energy, since the minimum of (4.39) can be obtained if and only if \( f \) is conformal and hence isometric. In fact, a piecewise linear mapping is conformal only for developable surface, and in this case conformality implies isometry. The term MIPS stands for Most Isometric Parameterizations.

Sheffer and de Sturler [109] propose a completely different method based on minimizing an energy functional that depend on the angles of the meshes and on the corresponding planar angles mapped on the parameter domain. Even in this case the conformality of the mapping results in a minimum value of the energy functional used.

Levy et al. [110] propose measuring the violation of the Cauchy-Riemann equations (4.33) in a least square sense:

\[
E_C(g) = \frac{1}{2} \left( (u_x - v_y)^2 + (u_y + v_x)^2 \right)
\]

(4.40)

and finding the mapping \( f \) that minimizes the energy:

\[
E_C(f) = \sum_{T \in S} E_C(f|_T) A(T)
\]

(4.41)

Desbrun et al. [111] independently propose another method, based on finite element theory, that results in the same linear system used to solve (4.41).
Discrete Equiareal Mappings

Typically, the base idea of these methods is to fix the polygonal region $S^*$ to have the same area of $S_T$ and then to find $f$ which minimize functionals in the following form:

$$E(f) = \sum_{T \in S_T} (A(f(T)) - A(T))^2 \quad (4.42)$$

Not surprisingly, such methods are not stable, in the sense that the final result may be unpredictable. In fact, multiple solutions for which $E(f) = 0$ could be possible. Sometimes, long and thin triangles appear after the parameterization. In some cases triangles flip over. Maillot et al. [112] try to improve numerical stability of the (4.42) by dividing each term of the sum by $A(T)$.

Sander et al. [113] explore methods based on minimizing functionals that measure the "stretch" of a mapping. Numerical examples showing comparison with discrete harmonic maps can be found in [113].

4.4.3 Remeshing

Another important element in geometry processing is remeshing. Remeshing concerns the re-arrangement of the sampling and the topology of the mesh in order to obtain polygonal meshes with specific properties in terms of surface sampling, regularity and shape of elements (triangles or quadrilaterals). For example, remeshing can be used to build high-quality meshes for numerical simulation (e.g. to calculate mechanical stress, to solve heat differential equations). In this case, for high-quality, we intend a mesh composed by triangles all with similar area and angles (pseudo-equilateral triangles). This kind of remeshing is usually called uniform or isotropic remeshing. Instead, anisotropic remeshing can be used to make an efficient sampling of the surface reducing the number of vertices and faces necessary to represent the same shape. In this case the shape of the triangles (or quadrilaterals) that compose the mesh is not uniform and, usually, the elements are elongated along the curvature of the mesh. Another remeshing technique is semi-regular remeshing that is based on subdivision and it is used to transform irregular meshes in semi-regular ones. In the following we give an overview of these remeshing techniques.

Isotropic Remeshing

Highly regular meshes, both in terms of geometry and connectivity, are necessary to perform numerical computations such as finite element analysis, for example
4.4. Mesh Parameterization

Darmstadt model (4100 vertices, 7138 triangles)

Uniform convex combination mapping
(Tutte [104])

Shape-preserving parameterization
(Floater [106])

Most Isometric Parameterizations (MIPS)
(Hormann et al. [108])

Figure 4.8: Examples of different mesh parameterization methods. (from [103])
for calculating mechanical stress. Uniform meshes provide good conditioning of the involved systems and minimizes the numerical errors and singularities that might otherwise arise.

Typically, such methods are based on mesh parameterization: the mesh is parameterized over a planar domain, the planar domain is uniformly sampled, and then the inverse mapping is applied to obtain the uniform mesh (see Figure 4.9). The parameterization and the sampling method used result in different remeshing algorithms. Some recent remeshing techniques based on parameterization are [114, 115, 116, 48]. The main problem of these methods is the distortions that might be arise due to the parameterization process. Moreover, such methods are computational expensive since mesh parameterization is a complex process that typically requires the solution of large linear (or non-linear) system of equations.

An alternative to parameterization is to work directly on the given mesh. Remeshing algorithms based on this approach usually involves optimizations problems defined in the 3D space. One of the most recent techniques of this type is the “explicit” remeshing of Surazhsky et al. [117] where the input mesh is locally modified according to an area-based smoothing technique and the regularity of
the connectivity is treated as a post-processing problem. Other remarkable works in this area are [118, 119, 120, 121].

**Anisotropic Remeshing**

Any remeshing algorithm that is not constrained to produce uniform meshes can be categorized as *anisotropic remeshing*. Nevertheless, recently, anisotropic remeshing has put the emphasis on align the shape and orientation of the mesh elements along the local curvature directions. In fact, most of the modern anisotropic remeshing techniques are based on the analysis of the principal directions of mesh curvature. The sampling is adapted to the intrinsic isotropy (high curvatures) or anisotropy (flat regions) of the different surfaces’ parts. Canonical shapes such as cylinders are useful to illustrate the power of the anisotropic alignment. In fact, increasing uniformly the number of elements along the circumference results in better approximation, while do the same along the zero curvature direction, i.e. along the axis of the cylinder, does not improve the quality of the polygonal approximation. Anisotropic remeshing is a fundamental tool for those applications that require intensive computations over the mesh elements. Moreover, in interactive visualization, large meshes can be re-sampled with a considerably reduced number of polygons avoiding bottleneck in the transmission of data to the graphics hardware.

Bossen and Heckbert [123] proposed an anisotropic triangle meshing technique based on a metric tensor defined over flat regions. They proceed through successive vertex insertion, vertex removal and iterative relaxations. Shimada et al. [124] developed a quad-dominant anisotropic remeshing algorithm by stretching rectangles according to a specified vector field using a computational intensive rectangle packing method. Heckbert and Garland [125] investigated the theoretical link between the quadric error metric used by Garland in his simplification algorithm [40] and anisotropic remeshing. In particular they demonstrated that the elongation of the triangles of the simplified mesh follows the principal curvature directions. Motivated by these and other works Alliez et al. [122] develop a method that consists in estimate the principal curvature directions, integrate it to obtain lines of curvatures, finding the intersections of such lines, and then sampling it properly to obtain an anisotropic mesh (see Figure 4.10). The obtained mesh is also smoothed to obtain a well-shaped one. Recently, this technique has been extended by Marinov and Kobbelt [126] by improving the scalability and the performances of the original algorithm.
Semi-regular Remeshing

Semi-regular remeshing permits to build a semi-regular mesh by starting from an irregular one. This type of remeshing is typically based on the following two main steps: the construction of a base mesh from the given irregular one, and then the subdivision of this base mesh in order to build the semi-regular mesh. The base mesh is a mesh composed by few triangles such that each base triangle is associated to a disk-like patch of the input irregular mesh by a proper parameterization. By resampling the base mesh iteratively applying a subdivision scheme and then using the inverse parameterization of each base triangle is possible to build a semi-regular mesh. The only extraordinary vertices of the semi-regular meshes so obtained are...
4.4. Mesh Parameterization

![Figure 4.11: Semi-regular remeshing. (from Eck et al. [45])](image)

Eck et al. [45] propose one of the first semi-regular remeshing algorithm (see Figure 4.11). In this technique, first the input mesh is subdivided in patches by a Voronoi tessellation. Then, the base mesh is constructed by parameterize each patch with a harmonic mapping. Khodakovsky et al. [46] build the base mesh through a simplification process. During the simplification each vertex is tracked over the simplified mesh in order to build a proper parameterization over the final base domain. Guskov et al. [47] developed an algorithm to approximate with a desired degree of accuracy any irregular mesh with a normal mesh, that is a particular kind of semi-regular meshes where the fine level of resolution can be obtained from the coarser one by using a normal offset defined for each vertex.

Recently, the concept of semi-regular remeshing has been extended to obtain completely regular mesh. In particular Gu et al. [48] open this new frontier by proposing to remesh an arbitrary polygonal surface onto a completely regular structure called geometry image. A geometry image is a bi-dimensional matrix of values capable to encode the geometry of a 3D mesh. Geometry images are
created by a global parameterization approach consisting in cut the input mesh along a network of edge paths, and then parameterize the resulting chart over a square. Praun et al. [127] use spherical parameterization to build geometry images. Sander et al. [49] improve the flexibility of geometry images by piecewise parameterize the surface obtaining a multi-chart geometry images. This approach permits to reduce the distortions caused by parameterization.
Chapter 5

Blind and robust watermarking of semi-regular meshes

5.1 Introduction

From the beginning of our investigations about 3D watermarking the goal has been the development of a blind watermarking system for 3D objects with strong robustness properties. In Chapter 3, after the review of several 3D watermarking algorithms, we have established that this is an ambitious goal. Here, we propose a system for watermarking of 3D objects represented as semi-regular meshes. This system is the result of many considerations about most of the concepts and problems exposed since now. A first schematization of the proposed system is shown in Figure 5.1.

![Figure 5.1: A sketch of the proposed 3D watermarking system.](image)

The system takes as input a polygonal mesh, in particular a semi-regular mesh and returns a watermarked semi-regular mesh. We assume that the 3D objects to watermark are represented by semi-regular meshes for several reasons. Most of these reasons have been explained in the previous chapters; here we report a summary of such motivations:

- Graphics hardware is designed to work with triangular meshes.
It is easy to convert other representations to polygonal meshes.

Most of the existing tools for geometry processing work with polygonal meshes, and in particular with semi-regular meshes.

It is always possible to transform any irregular mesh to a semi-regular one with an operation of remeshing.

Semi-regular meshes allow wavelets analysis/synthesis in a natural way.

Before the embedding/decoding phase the input mesh is pre-synchronized by a proper pre-processing phase. In fact, the problem of several watermarking algorithms is that certain attacks could de-synchronize the watermarked data hence making impossible watermark decoding/detection. In many cases the embedding techniques are designed to make the system resistant against several attacks even without a pre-synchronization phase. For example, many of the algorithms reviewed in Section 3.3 are invariant with respect to translation, scaling and rotation. Nevertheless, since a lot of different and difficult-to-prevent attacks are possible on a polygonal mesh, we think that is better to improve the reliability of watermark extraction by inserting a proper synchronization phase in the watermarking system instead of looking for geometric features intrinsically robust to several attacks. Stated in another way, the robustness of the overall system is achieved not only by the characteristics of the embedding/decoding algorithm but both by the synchronization phase and by the embedding/decoding algorithm. An emblematic example of this approach is given by the algorithm by Praun et al. [53], where the excellent robustness properties of the algorithm is given by the registration/resampling phase before the watermark extraction. We follow the same strategy. Specifically, our synchronization is based on Principal Components Analysis (PCA) and it accounts for simple geometric attacks such as translation, rotation and uniform scaling, since our embedding features are not able to deal with those attacks.

After synchronization the model can be watermarked, or analyzed in order to recover the embedded information. Our watermarking algorithm works is based on the wavelet decomposition of semi-regular meshes described in Section 4.3. We opted for a multiresolution approach in order to obtain both good spatial localization, and hence control, of the watermark distortions, and good robustness properties typical of the transformed domain. These two benefits can be achieved by inserting the watermark into a low level of resolution of the mesh. In particular, embedding is achieved by encoding the watermark into the modulus of the wavelet
coefficients of a given level of resolution by an additive watermarking scheme. For additive watermark we intend that the watermark signal is added to the host signal. In this case the host signal is represented by the modulus of the wavelet coefficients. The watermark signal is generated by sampling a watermarking map which depends on the information to embed. The detection of the watermark is accomplished by statistical analysis of the wavelet coefficients of the mesh. The wavelet coefficients are obtained with the same wavelet decomposition used for the embedding. The statistical analysis follows the Neyman-Pearson criterion applied to a proper statistical hypothesis test. The detector is only able to detect the presence of the watermark, so the watermark is of the kind of detectable watermark.

In the following, we first give the details about the practical implementation of the wavelet decomposition of the model. Then, we describe in depth the synchronization, the embedding, and the detection phases, and we present the experimental results about the robustness properties of the method. At the end of the Chapter we draw the conclusions about the developed technique and we propose some variants to further improve its robustness performances.

5.2 Wavelet Analysis/Synthesis

The wavelet analysis, and the subsequent reconstruction, is obtained by applying the Lounsbery’s framework for wavelet decomposition of semi-regular meshes previously seen. To describe the implementation of such framework we use the following terminology: we indicate the semi-regular meshes at full resolution with \( M^0 \), with \( M^0 \) the base mesh, with \( M^l \) the mesh at level of resolution \( l \), and with \( V^l \) and \( W^l \) the vertices and the wavelet coefficients of the mesh \( M^l \), respectively. Such terminology is adopted also in the next sections.

Any primal subdivision scheme can be split into two distinct steps: a polyhedral subdivision plus a displacement of the subdivided mesh vertices. Polyhedral subdivision simply concerns the split of each face of the meshes into other 4 faces. In the case of interpolating schemes, after the splitting, only the new vertices are displaced, while in the case of approximating scheme all the vertices are moved in a new position. The idea of Lounsbery to greatly simplify the implementation of the filter banks for polyhedral surfaces is to encode the vertices displacement as vectors that, in this way, represents the detail coefficients to pass from the coarse mesh \( M^j \) to the fine one \( M^{j+1} \). Figure 5.2 illustrates the simplified wavelet
Chapter 5. Blind and robust watermarking of semi-regular meshes

Hence, imagine to have just computed the wavelet coefficients, the synthesis is obtained in two steps. First, the 1-to-4 split is applied to the mesh $M^j$ producing a mesh $\hat{M}^{j+1}$ with new vertices inserted in the middle of each edge. The mesh $\hat{M}^{j+1}$ has the same connectivity of $M^{j+1}$. Then, the wavelet coefficient vectors are used to move the new vertices of $\hat{M}^{j+1}$ to obtain the mesh at the higher level of resolution $M^{j+1}$.

The analysis is achieved by inverting the synthesis process. To do so we need to know the information about the connectivity of the semi-regular meshes. In particular, the information needed are the number of levels of resolution and the level of resolution each vertex belongs. If these information are unknown it is possible to retrieve it by analyzing the connectivity of the mesh [128]. Thanks to these information the wavelet coefficients $\omega^l_i$ can be computed as:

$$\omega^l_i = v^{l+1}_i - \hat{v}^l_i = v^{l+1}_i - \frac{v^{l}_{i,1} + v^{l}_{i,2}}{2} \quad (5.1)$$

where $\hat{v}^{l+1}_i$ is the middle point of the edge connecting the vertex $v^{l}_{i,1}$ with the vertex $v^{l}_{i,2}$. As it is possible to notice from the (5.1), the calculation of the wavelet coefficients depends by the levels of resolution of the involved vertices.
5.3 Synchronization

The synchronization stage has been designed to normalize the position, orientation and dimension of the input mesh in order to avoid de-synchronization problems caused by translation, rotation and uniform scaling attacks. Synchronization is achieved in two steps. First, the model orientation is normalized by means of Principal Component Analysis (PCA). Then the model is fitted into a bounding box consisting of a cube of dimensions $2.0 \times 2.0 \times 2.0$ centered at the origin of the Cartesian axis. So, the corner of the cube in the positive quadrant has coordinates $(1.0, 1.0, 1.0)$ and the corner in the negative quadrant has coordinates $(-1.0, -1.0, -1.0)$. This fitting is obtained by applying a proper translation and scaling operation to the model. In the next we detail these two sub-phases.

Model re-orientation

The base idea of model re-orientation is to orient the model along the principal components of its associated inertia tensor. Since the inertia tensor depends on the overall shape of the model we expect that, even in presence of attacks, the final orientation is always the same with good approximation. To do this, we have to compute the center of mass and the inertia tensor of the model. The ideal approach is to compute such entities using a volumetric approach, but, in order to make such computation simpler we use a surface approach. In particular, we consider the centers of each face of the mesh, instead of the vertices of the mesh, and we associate to each face center a mass equals to the area of the face $p_i$ belongs to. After the center of mass is calculated, it is possible to compute the inertia tensor $I_M$ as:

$$O = \frac{1}{n_f} \sum_{i=1}^{n_f} p_i m_i \quad (5.2)$$

where $n_f$ is the number of faces of the input mesh, $p_i$ is the center of the $i$-th face of the model and the mass $m_i$, associated to $p_i$, is the area of the face $p_i$ belongs to. After the center of mass is calculated, it is possible to compute the inertia tensor $I_M$ as:

$$I_M = \begin{bmatrix} \sum_{i=1}^{n_f} (x'_i)^2 m_i & \sum_{i=1}^{n_f} y'_i x'_i m_i & \sum_{i=1}^{n_f} z'_i x'_i m_i \\ \sum_{i=1}^{n_f} x'_i y'_i m_i & \sum_{i=1}^{n_f} (y'_i)^2 m_i & \sum_{i=1}^{n_f} z'_i y'_i m_i \\ \sum_{i=1}^{n_f} x'_i z'_i m_i & \sum_{i=1}^{n_f} y'_i z'_i m_i & \sum_{i=1}^{n_f} (z'_i)^2 m_i \end{bmatrix} \quad (5.3)$$

where $(x'_i, y'_i, z'_i)$ are the coordinates of the points $p_i$ relative to the center of mass $O$, i.e. $(x'_i, y'_i, z'_i) = (p_{x,i} - O_x, p_{y,i} - O_y, p_{z,i} - O_z)$. The eigenvectors of $I_M$ are
used to align the principal axis of $I_M$ to the coordinate axis. This is done by simply multiplying the position of each vertex by the PCA matrix, that is the matrix composed by the eigenvectors of $I_M$.

**Model Normalization**

The procedure to fit the re-oriented model into a cube of a given edge length, centered in the origin, is quite obvious but for completeness we provide the details of the process. The translation is trivial; the absolute coordinates of the vertices of the re-oriented mesh are translated according to the center of mass $O$. In this way the center of mass of the normalized model coincides with the origin of the Cartesian axis. Then, the model has to be scaled in order to fit the cube of dimension $2.0 \times 2.0 \times 2.0$. To do this, the maximum absolute value of the new coordinates ($l_{MAX}$) is taken and the model is scaled uniformly by dividing each coordinates for such value:

$$
l_{MAX} = \max \left\{ \max_{i \in \mathcal{V}} |x_i'|, \max_{i \in \mathcal{V}} |y_i'|, \max_{i \in \mathcal{V}} |z_i'| \right\}
$$

(5.4)

where $\mathcal{V}$ contains the indices of the mesh vertices and $(x_i', y_i', z_i')$ are the coordinates translated second $O$.

Figure 5.3 shown the Venus model before, and after the synchronization stage.

**5.4 Embedding**

The embedding algorithm inserts the watermark by modifying the modulus of the wavelet coefficients at a pre-specified level of resolution. As just stated in the previous Section, the mesh $M^{l+1}$ can be obtained from the mesh $M^l$ by subdividing it with a polyhedral subdivision rules and then displacing the new vertices with
5.4. Embedding

the wavelet coefficients $W^l$. Hence, a mesh with $n$ level of resolutions can be decomposed into a coarse base mesh $M^0$ with vertices $V^0$ plus $n$ sets of wavelet coefficients $\{W^0, W^1, \ldots, W^{n-1}\}$. The topology of the different level of resolution is determined by the subdivision process starting from $M^0$. The embedding algorithm requires three parameters to be specified:

1. a numeric key $K$

2. the resolution level $l$ that will host the watermark

3. a coefficient $\gamma$ determining the strength of the watermark

The numeric key $K$ represents the piece of information embedded within the 3D object. This numeric code can be used to proof ownership if assigned by a third part authority, or to index a database with licence plate or other kind of information about the model. The resolution level $l$, together with the parameter $\gamma$ permit to control the trade-off between impact of the watermark distortions and robustness.

An overall picture of the watermark insertion process is depicted in Figure 5.4. In particular the following steps are performed:

1. The input model $M^n$ is decomposed into $n$ sets of wavelet coefficients $W^0$, $W^1, \ldots, W^{n-1}$ and a base domain $M^0$.

2. A watermark-dependent structure called *watermarking map* is generated according to $K$. 

![Figure 5.4: Scheme of the watermark embedding algorithm.](image)
3. The vertices of the model at level \( l \) (\( V^l \)) and the wavelet coefficients at the same level (\( W^l \)) are used to encode the watermark within the model by altering the modulus of a subset \( W^l \) of the wavelets coefficients of \( W^l \).

4. The watermarked wavelet coefficients \( W^l \) together with the other sets of wavelets \( W^{l+1}, \ldots, W^{n-1} \) are used to reconstruct the model, obtaining the watermarked model at full resolution.

In the next sections we give a detailed description of the above steps, in particular of the watermarking map generation and of the encoding step (the step 3). This phase of the embedding is usually referred to as \textit{watermark casting}.

### 5.4.1 Watermarking Maps

The numeric key \( K \) is used to generate a pseudo-random matrix of values called \textit{watermarking map} (\( W_{MAP} \)). The building of \( W_{MAP} \) constitutes the information coding (2.1) of our watermarking system. This map is used during watermark casting to find the values whereby alter the modulus of the host wavelet coefficients. The map \( W_{MAP} \) is build as following: the key \( K \) is used as a seed to generate a pseudo-random sequence of displacement values \( D = \{d_1, d_2, \ldots, d_N\} \) uniformly distributed in the \([-\Delta/2, \Delta/2]\) interval. Then, the sequence \( D \) is arranged into a matrix with \( n \) rows and \( m \) columns (\( N = n \times m \)) in the following way:

\[
W_{MAP} = \begin{bmatrix}
  d_1 & d_2 & \ldots & d_m \\
  d_{m+1} & d_{m+2} & \ldots & d_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  d_{(n-1)m+1} & d_{(n-1)m+2} & \ldots & d_{nm}
\end{bmatrix}
\]  

Our aim is to associate the values of the \( W_{MAP} \) to directions in the space represented with spherical coordinates, i.e. with the pair \((\theta, \phi)\), where \( \phi \in [0, \pi) \) is the elevation angle and \( \theta \in [0, 2\pi) \) is the azimuth angle. To do this, we associate the rows of the map to the elevation angle, and the columns of the map to the azimuth angle. In other words, the row index of \( W_{MAP} \) samples the azimuth angle and the column index samples the elevation angle. The number of rows and columns of \( W_{MAP} \) determines the accuracy with which the polar representation of the unitary sphere is sampled. For example, a \( W_{MAP} \) of \( 180 \times 360 \) elements corresponds to 1 degree of resolution; a map of \( 180 \times 90 \) to 2 degree of resolution, and so on. Figure 5.5 shows a \( W_{MAP} \) with 4 rows and 6 columns thus result in a map of 60 degree of resolution.
5.4.2 Watermark casting

The set of wavelet coefficients $W^l$ and the vertices $V^l$ are used in conjunction to obtain the embedding of the watermark. More specifically, the polar coordinates of the vertices $v^l_i \in V^l$ are used to sample the generated watermarking map. Then, such values, are used to modify the modulus of the wavelet vectors $\omega^l_i \in W^l$. To be more precise, only a part of the wavelet coefficients of $W^l$ are modified. In formula:

$$|\omega^l_{i,s}| = |\omega^l_i| + \gamma \Delta_D(\hat{v}(\omega^l_i), W_{MAP}), \ \forall \omega^l_i \in W^l, \tag{5.6}$$

where $\overline{W}^l$ is a subset of $W^l$ whose exact meaning will be explained below. The exact amount of perturbation the modulus of the wavelet coefficient undergoes, i.e. $\gamma \Delta_D(\hat{v}(\omega^l_i), W_{MAP})$, is determined by the parameter $\gamma$, and by the mapping function $\Delta_D(.,)\,$, which depends on the application point $\hat{v}(\omega^l_i)$ of the wavelet coefficient $\omega^l_i$ and on the watermarking map $W_{MAP}$.

The function $\Delta_D(\hat{v}(\omega^l_i), W_{MAP})$ maps the polar coordinates $(\theta_\theta, \phi_\phi)$ of $\hat{v}(\omega^l_i)$ to a pair of indices of the watermarking map permitting to obtain values of perturbation specific for a certain position in 3D space and for a certain watermark. Of course, the polar coordinates of $\hat{v}(\omega^i)$ usually do not correspond to a location expressed by integer indices over the $W_{MAP}$, hence a simple interpolation rule is adopted to calculate the exact value of $\Delta_D(\hat{v}(\omega^l_i), W_{MAP})$:

$$\Delta_D(\hat{v}(\omega^l_i), W_{MAP}) = W_{MAP}(\theta_i, \phi_i) \tag{5.7}$$

$$\theta_i = \text{round}\left(\frac{m}{360} \theta_\theta\right)$$

$$\phi_i = \text{round}\left(\frac{(n - 1)}{180} \phi_\phi\right) \tag{5.8}$$
where \( \text{round}(x) \) is the nearest integer to the value of \( x \). Figure 5.6 shows graphically how \( \Delta_D(\hat{\omega}_l, W_{MAP}) \) is calculated. Note that it is possible that two or more points \( \hat{\omega}_l \)'s are mapped into the same location of \( W_{MAP} \). When this happens some correlation within the watermarking sequence \( \Delta_D(\hat{\omega}_l, W_{MAP}) \) is introduced; hence, it is desirable that this event is as rare as possible. In our system the resolution of 1 degree has been found to be a good trade-off between independence of perturbation values and robustness. Robustness and the resolution of the watermarking map are tied by an inverse proportional relationship. In fact, a \( W_{MAP} \) with high resolution makes the mapping more sensitive to the position of the vertices. On the contrary the sampling of a map with very low resolution, for example 20-30 degrees, produces the same perturbation values even if the vertices positions have been considerably altered by attacks.

The use of spherical coordinates to sample \( W_{MAP} \) produces non-uniform sampling over it. In particular, the mapping distortions increasing near the poles. Such distortions cause problems similar to the one above described. For this reason we chose to mark only those wavelets for which the elevation angle \( \phi_\theta \) is far from the poles. Thus, only wavelets with an application point that satisfies the constrain \( \Delta \phi \leq \phi_\theta \leq 180.0 - \Delta \phi \) are marked. In our implementation \( \Delta \phi \) is set to twenty degrees assuring a good uniformity of the watermark sampling. We indicate the set of wavelet coefficients far from the poles with \( \bar{W}_l \subset W_l \).

At the end, We have to pay attention to avoid that the watermarking process produces negative modules. To do so we replace negative modules with zero, that
5.5. Detection

Figure 5.7: Watermark embedding. The modulus of the wavelet coefficient $|\omega^j_i|$ is perturbed according to the value of the watermarking map indexed by the polar coordinates $(\theta_v, \phi_v)$ of wavelet’s application point $\hat{v}(\omega^j_i)$.

is:

$$|\omega^{l,i}| = \min(0, |\omega^j_i| + \gamma \Delta_D(\hat{v}(\omega^j_i), W_{MAP})) , \ \forall \omega^j_i \in \mathbb{W}^l$$  \hspace{1cm} (5.9)

5.5 Detection

The detection procedure requires that the user specifies two input parameters: the numeric key $K$ and the level of resolution $l$ where he/she wants to verify the presence of $K$. Then, the detector analyzes the model in order to establish if the given $K$ has been embedded in the input model.

The first step of watermark detection is to decompose the synchronized mesh $M^n$ in the same way of the embedding obtaining the two sets $V^l$ and $W^l$. Then, the watermarking map $W_{MAP}$ is generated by starting from $K$. At this point the wavelet coefficients $W^l$ are analyzed in order to verify the presence of the watermark $K$. To be more specific, the detector verifies the presence of the watermark by computing the correlation between the watermarking signal, that is the sequence of values given by the mapping function $\Delta_D(.,.)$, and the host wavelet coefficients ($\omega^j_i \in \mathbb{W}^l$):

$$\rho = \frac{1}{|\mathbb{W}^l|} \sum_{\omega^j_i \in \mathbb{W}^l} |\omega^j_i| \Delta_D(\hat{v}(\omega^j_i), W_{MAP})$$  \hspace{1cm} (5.10)

where $|\mathbb{W}^l|$ is the cardinality of $\mathbb{W}^l$. After calculating $\rho$, if it results greater than a certain threshold $T_\rho$ the watermark is assumed to be present, otherwise the model
is assumed to be non-marked. This way to proceed relies on statistical considerations applied to a specific hypothesis test used to model the detection problem. In the next section we give a brief description of such statistical analysis and we demonstrate that the correlation is a sufficient statistics to identify the watermark. Additionally, we explain how to compute the threshold $T_\rho$ in order to guarantee certain theoretical performances of the detector (Neyman-Pearson criterion). In the experimental results section we validate the theoretical performances of the detector.

5.5.1 Statistical Decision Theory

In the general case, the detection problem can be summarized as to decide if the asset $A'$, i.e. the asset under analysis, coincides with the original asset $A$ or is marked with the given watermark. In the framework of statistical decision theory, the above problem corresponds to deciding in which states the asset $A'$ is, choosing between a certain number of states. In our specific case the problem can be modelled by the following hypothesis test:

$$H_0 : A' \text{ does not contain } w$$
$$H_1 : A' \text{ contains } w$$

The hypothesis $H_0$ takes into account both the case in which the model is not watermarked and the case in which another watermark is present.

In Bayes theory of hypothesis testing, the usual criterion to decide between the hypothesis is the minimization of the risk to make a bad choice. The probability to make an error in the selection of the valid hypothesis can be computed by considering two kinds of errors. The first one occurs when the hypothesis $H_1$ is accepted but the watermark is not present. This error is usually referred to as error of false alarm. The second kind of error occurs when the model contains the given watermark but the hypothesis $H_0$ is accepted. This error is usually referred to as error of missing detection. Indicating with $P_f$ the probability of false alarm and with $P_m$ the probability of missing detection we can express the probability of error as:

$$P_e = p_0 P_f + p_1 P_m$$

where $p_0$ and $p_1$ are the a priori probabilities of the hypothesis test $H_0$ and $H_1$, respectively.
The Bayes criterion relies on the following decision rule:

\[
\Phi(f') = \begin{cases} 
1 & l(f') > \lambda \\
0 & \text{otherwise}
\end{cases} 
\]  
(5.13)

As it is possible to notice the decision rules \(\Phi(.)\), function of the host features \(f'\) of the asset \(A'\) under inspection, is based on the comparison of the likelihood ratio \(l(f')\) to a certain threshold \(\lambda\). The likelihood ratio \(l(.)\) is defined as:

\[
l(f') = \frac{p(f'|H_1)}{p(f'|H_0)}
\]  
(5.14)

where \(p(f'|H_i)\) is the probability of \(f'\) conditioned to the hypothesis \(H_i\).

Let \(C_{01}\) be the “cost” to opt for the hypothesis \(H_0\) when \(H_1\) is the right one and \(C_{10}\) the “cost” to opt for the hypothesis \(H_1\) when the other one is in force, the threshold \(\lambda\) can be stated as:

\[
\lambda = \frac{p_0 C_{01}}{p_1 C_{10}}
\]  
(5.15)

With this definition of \(\lambda\), from decision theory is well known that the minimum value of \(P_e\) is obtained by assuming that \(p_0 = p_1\) and \(C_{01} = C_{10}\). Hence, the minimum Bayes risk is obtained for \(\lambda = 1\).

For watermarking applications, in most cases, the criterion above described is not suitable. In fact, in many cases, it is preferable to minimize the probability of missing the watermark subject to a constrained false alarm probability. This is the main idea of the Neyman-Pearson criterion: finding the threshold \(\lambda\) in order to maximizing the probability of correctly detecting the watermark \((1 - P_m)\) subject to a prefixed value of \(P_f\).

For an Additive White Gaussian Noise (AWGN) channel both the host features \((\{f_i\})\) and the potential attacks are modelled as uncorrelated, Gaussian noise added to the watermark signal, i.e. :

\[
\hat{f}_{i,*} = f_i + \gamma w_i + \eta_i
\]  
(5.16)

where \(w_i\) is the watermark signal, \(\hat{f}_{i,*}\) are the watermarked host features and \(\eta_i\) is a white gaussian noise that accounts for attacks. By comparing equation (5.16) with (5.6) it is easy to see that our embedding rule agree with this model. In such case the hypothesis test (5.11) can be rewritten as:

\[
\begin{align*}
H_0 : & \begin{cases}
\text{case 1} : & f_i' = f_i \\
\text{case 2} : & f_i' = f_i + \gamma \tilde{w}_i
\end{cases} \\
\text{case 2} : & f_i' = f_i + \gamma w_i
\end{align*}
\]  
(5.17)
where the second case of the composite hypothesis $H_0$ accounts for the presence of another watermark $\tilde{w} \neq w$. This complicates a lot the calculation of the likelihood ratio due to the fact that the presence of any other watermark must be taken into account. In formula:

$$l(f') = \frac{p(f'|w)}{\int_{\mathbb{R}^n} p(f'|\tilde{w})p(\tilde{w})d\tilde{w}}$$  \hspace{1cm} (5.18)$$

The numerator of $l(f')$ can be computed by remembering that in the case of AWGN $f$ is assumed a gaussian, white, normally distributed sequence of values. We have:

$$p(f'|w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi \sigma_f^2}} \exp \left( -\frac{(f'_i - \mu_f - \gamma w_i)^2}{2\sigma_f^2} \right)$$  \hspace{1cm} (5.19)$$

where $\mu_f$ and $\sigma_f^2$ indicates the mean and variance of the sequence $f$, respectively.

For the same assumptions the denominator of (5.18) can be calculated as:

$$p(f'|H_0) = \prod_{i=1}^{n} \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi \sigma_f^2}} \exp \left( -\frac{(f'_i - \mu_f - \gamma \tilde{w}_i)^2}{2\sigma_f^2} \right) p(\tilde{w}_i)d\tilde{w}_i$$  \hspace{1cm} (5.20)$$

The evaluation of the (5.20) is too cumbersome, hence it is necessary to introduce some assumption in order to simplify it. A simplification commonly used in watermarking literature is to neglect the presence of other watermarks:

$$p(f'|H_0) = p(f'|0)$$  \hspace{1cm} (5.21)$$

where $0$ indicates the null watermark. In other words, we assume that $H_0$ only consists of the case 1. Neglecting the presence of other watermarks the asset is not always possible, for a rigorous justification of such assumption we refer to [5]. Combining the (5.19) with the (5.20) and taking into account the simplification assumption (5.21) we obtain the likelihood for the AWGN case:

$$\frac{p(f'|w)}{p(f'|0)} = \prod_{i=1}^{n} \exp \left( -\frac{(f'_i - \mu_f - \gamma w_i)^2}{2\sigma_f^2} \right)$$  \hspace{1cm} (5.22)$$

It is convenient to replace $l(f')$ with a log-likelihood ratio, i.e. with $\mathcal{L}(f') = \ln l(f')$. Considering the log-likelihood ratio instead of the likelihood ratio requires to modify the decision rule in the following way:

$$\mathcal{L}(f') > \ln \lambda = \Lambda$$  \hspace{1cm} (5.23)$$
Rewriting the (5.22) using a logarithmic formulation, yields:

\[
L(f') = \sum_{i=1}^{n} \frac{1}{2\sigma^2_f} [(f'_i - \mu_f)^2 - (f'_i - \mu_f - \gamma w_i)^2]
\]

\[
= \frac{1}{2\sigma^2_f} \left[ \sum_{i=1}^{n} 2\gamma f'_i w_i - \sum_{i=1}^{n} 2\gamma \mu_f w_i - \sum_{i=1}^{n} \gamma^2 w_i^2 \right]
\]

By noting that the last two terms in square brackets do not depend on \( f' \) it is possible to conclude that the linear correlation between \( f' \) and the watermark, that is:

\[
\rho = \sum_{i=1}^{n} f'_i w_i
\]

is a sufficient statistics for watermark detection. In other words such correlation completely characterizes the performance of the detector. In the next Section we compute the threshold \( \Lambda \), that we will indicate with \( T_\rho \) for our specific cases.

### 5.5.2 Threshold Computation

In order to compute the threshold of the decision rule we adopt the Neyman-Pearson criterion, which consists, as previously stated, in minimizing the missed detection probability \( (P_m) \) for a given probability of falsely revealing the watermark \( (P_f) \) in a non-marked host. Before proceeding with the evaluation of \( P_f \) and \( P_m \) we precise that in this Section we avoid to explicitly indicate the dependence of the wavelet coefficients upon the level of resolution, in other words the wavelet coefficient under analysis \( \omega'_i \) are indicated with \( \omega'_i \).

According to the hypothesis test (5.17), under the simplification assumptions (5.21) the probability of false alarm is:

\[
P_f = P\{\rho > T_\rho | H_0\}
\]

where \( H_0 \) indicates the hypothesis that the wavelet coefficients are not marked, i.e. that \( |\omega'_i| = |\omega_i| \). Hence, \( P_f \) can be calculated as:

\[
P_f = \int_{T_\rho}^{\infty} p(\rho | H_0) d\rho
\]

Similarly, the probability of missing the watermark results:

\[
P_m = P\{\rho < T_\rho | H_1\}
\]
that can be evaluated using the expression:

\[ P_m = \int_{-\infty}^{T_p} p(\rho|H_1) d\rho \]  

(5.29)

where \( H_1 \) indicates the hypothesis that \( \omega'_i \) are marked with the watermark whose presence is under verification, i.e. that \( \omega'_i = |\omega_i| + \gamma \Delta_D(\hat{v}(\omega_i), W_{MAP}) \).

In order to calculate the relations (5.27) and (5.29) we fix the host model and averaging over different watermarks \( K \). In the sequel we will assume that watermark samples, that are the sequence of values \( \Delta_D(\hat{v}(\omega_i), W_{MAP}) \), are zero mean i.i.d. random variables. As we anticipated when describing the watermark casting, this assumptions about watermark samples is true only if the application points of wavelet coefficients \( \hat{v}(\omega'_i) \) are mapped into different elements of \( W_{MAP} \).

As just stated, to achieve this we carefully select the watermarking map resolution and we avoid to sample the watermarking map near the poles (see Section 5.4.2).

Furthermore \( \Delta_D(\hat{v}(\omega_i), W_{MAP}) \) are independent random variables and \( \omega'_i \) are fixed parameters which are known to the detector, hence, by invoking the central limit theorem, we can conclude that \( \rho \) follows a normal distribution. Furthermore, to completely characterize the probability distribution of \( \rho \), it is sufficient to estimate its mean and variance. By estimating the mean and variance of \( \rho \) in the case \( H_0 \) and \( H_1 \) it is possible to solve the integral expressions (5.27) and (5.29), respectively, thus obtaining the values of \( P_f \) and \( P_m \) in function of the statistics of \( \rho \). In particular the expressions (5.27) and (5.29) become:

\[ P_f = \int_{T_p}^{\infty} p(\rho|H_0) d\rho = \frac{1}{2} \text{erfc} \left( \sqrt{\frac{(T_p - \mu_{\rho|H_0})^2}{2\sigma_{\rho|H_0}^2}} \right). \]  

(5.30)

\[ P_m = \int_{-\infty}^{T_p} p(\rho|H_1) d\rho = \frac{1}{2} \text{erfc} \left( \sqrt{\frac{(\mu_{\rho|H_1} - T_p)^2}{2\sigma_{\rho|H_1}^2}} \right). \]  

(5.31)

Let us assume that \( H_0 \) holds. In this case we have \( \omega'_i = |\omega_i| \) and hence the mean of \( \rho \) results:

\[ \mu_{\rho|H_0} = E[\rho|H_0] = \frac{1}{n} E \left[ \sum_{\omega_i \in W} |\omega_i| \Delta_D(\hat{v}(\omega_i)) \right] = \mu_{\Delta_D} \bar{\omega} \]  

(5.32)

where \( \bar{\omega} = \sum |\omega_i|/n \) is the average of the elements in \( W \) and \( \mu_{\Delta_D} \) is the expected value of the sequence \( \Delta_D(\hat{v}(\omega_i), W_{MAP}) \). Under the same hypothesis the variance
of $\rho$ is:

$$
\sigma^2_{\rho|H_0} = \text{var}\left( \frac{1}{n} \sum_{\omega_i \in \overline{W}} |\omega_i| \Delta_D(\hat{v}(\omega_i)) \right) = \frac{1}{n} \sigma^2_{\Delta_D} \overline{\omega}^2
$$

(5.33)

with

$$
\overline{\omega}^2 = \frac{1}{n} \sum_{\omega_i \in \overline{W}} |\omega_i|^2
$$

(5.34)

denoting the mean square value of $|\omega_i| \in \overline{W}$. The symbol $\sigma^2_{\Delta_D}$ denotes the variance of the sequence $\Delta_D(\hat{v}(\omega_i), W_{MAP})$. The quantities $\mu_{\Delta_D}$ and $\sigma^2_{\Delta_D}$ can be easily computed since the values generated to build the watermarking map follows an uniform distribution in the interval $[-\Delta/2, \Delta/2]$. Furthermore, we have:

$$
\mu_{\Delta_D} = E[\Delta_D(.)] = 0
$$

(5.35)

$$
\sigma^2_{\Delta_D} = \text{Var}[\Delta_D(.)] = E[(\Delta_D(.) - \mu_{\Delta_D})^2] = E[\Delta_D(.)^2] = \frac{\Delta^2}{12}
$$

(5.36)

Now, we are able to evaluate the (5.30). In other words for a pre-fixed model, and a fixed threshold, we know in advance the probability that the detector gives a false alarm error.

Following the Neyman-Pearson criterion, we want to set a value of $T_{\rho}$ that guarantees a pre-fixed value of $P_f$. To achieve this we invert the expression (5.30). In this way we can calculate the detection threshold $T_{\rho}$ in function of the pre-fixed value of $P_f$, yielding:

$$
T_{\rho} = \sqrt{2} \sigma_{\rho|H_0} \text{erfc}^{-1}(2P_f) + \mu_{\rho|H_0}
$$

(5.37)

By inserting equations (5.32) and (5.33) in the above expression, we finally obtain:

$$
T_{\rho} = \sqrt{\frac{2\sigma^2_{\Delta_D} \overline{\omega}^2}{n}} \text{erfc}^{-1}(2P_f).
$$

(5.38)

To completely characterize the detector performances we have to evaluate $P_m$. Such performance are usually summarized through Receiving Operator Characteristics (ROC) curves where the missed detection probability is plotted against $P_f$. The evaluation of $P_m$ is achieved analogously to $P_f$; by estimating the mean and variance of $\rho$ under the hypothesis $H_1$ and then substituting them in the (5.31). In the $H_1$ hypothesis the correlation becomes:

$$
\rho = \frac{1}{|W|} \sum_{\omega_i \in \overline{W}} |\omega_i| \Delta_D(\hat{v}(\omega_i), W_{MAP}) + \gamma \Delta_D(\hat{v}(\omega_i), W_{MAP})^2
$$

(5.39)
and hence:
\[ \mu_{p|H_1} = \mathbb{E}[\Delta_D(.)] \bar{\omega} + \gamma \mathbb{E}[\Delta_D(.)^2] \]  
(5.40)
that, by taking into account the statistics properties of the watermark perturbations, in particular the equation (5.35), it assumes the simpler form:
\[ \mu_{p|H_1} = \gamma \sigma^2_{\Delta_D} \]  
(5.41)
The estimation of \( \sigma^2_{p|H_1} \) is a bit more complicated due to the fact that the watermark samples \( \Delta_D(.) \) depend on the host signal samples through \( \hat{\nu}(\omega_i) \). For this reason, the variance estimation involves the co-variance between the square of the watermark sample and the product \( \omega_i \Delta_D(.) \). In particular:
\[
\sigma^2_{p|H_1} = \text{Var} \left( \frac{1}{n} \sum \omega_i \Delta_D(.) + \gamma \Delta_D(.)^2 \right) \\
= \frac{1}{n^2} \left[ \text{Var} \left( \sum \omega_i \Delta_D(.) \right) + \gamma^2 \text{Var} \left( \sum \Delta_D(.)^2 \right) \right. \\
+ 2 \text{cov} \left( \sum \omega_i \Delta_D(.) , \gamma \Delta_D(.)^2 \right) \right] 
(5.42)
that, after several passages can be reduced to:
\[
\sigma^2_{p|H_1} = \frac{1}{n} \left[ \sigma^2_{\Delta_D} \bar{\omega}^2 + \gamma^2 \left[ \mathbb{E}[\Delta_D(.)^4] - \mathbb{E}[\Delta_D(.)^2]^2 \right] \right. \\
\left. + 2 \gamma \mathbb{E}[\Delta_D(.)^3] \mathbb{E}[\Delta_D(.)^2] \right] 
(5.43)
that can be further simplified by using the (5.35) and (5.36) into the final form:
\[
\sigma^2_{p|H_1} = \frac{1}{n} \left[ \sigma^2_{\Delta_D} \bar{\omega}^2 + \gamma^2 \left[ \mathbb{E}[\Delta_D(.)^4] - (\sigma^2_{\Delta_D})^2 \right] \right] 
(5.44)
The quantity \( \mathbb{E}[\Delta_D(.)^4] \) can be explicitly evaluated by taking into account the distribution of \( \Delta_D(.) \), i.e:
\[
\mathbb{E}[\Delta_D(.)^4] = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} x^4 dx = \frac{1}{5} \left( \frac{\Delta}{2} \right)^4 
(5.45)
By substituting the equations (5.41) and (5.44), and the expression for \( T_p \) (5.38) into (5.31) we can express the missed detection probability as a function of \( P_f \), in particular:
\[
P_m = \frac{1}{2} \text{erfc} \left( \sqrt{\frac{\left( \frac{\gamma \sigma^2_{\Delta_D} - \sqrt{\frac{2}{n} \sigma^2_{\Delta_D} \bar{\omega}^2 \text{erfc}^{-1}(2P_f)}}{\frac{2}{n} \sigma^2_{\Delta_D} \bar{\omega}^2 + \gamma^2 \left[ \mathbb{E}[\Delta_D(.)^4] - (\sigma^2_{\Delta_D})^2 \right]}}}{\frac{2}{n} \sigma^2_{\Delta_D} \bar{\omega}^2 + \gamma^2 \left[ \mathbb{E}[\Delta_D(.)^4] - (\sigma^2_{\Delta_D})^2 \right]} \right) 
(5.46)
The above equation (5.46) completely characterizes the detector performance.
In the next Section we validate the theoretical performance here obtained, in particular we show the validity of the false alarm probability.
5.6 Experimental Results

In this Section we report a selection of the results that we obtained while testing the performance of the proposed watermarking algorithm. In particular, in the next, we give only a brief excursus about the watermark perceptibility since the third part of the thesis is dedicated to this topic. Then, we estimate the actual false detection probability of the algorithm, in order to validate the (5.38). After this we, evaluate the robustness of the system against several attacks.

The test models used are the “Bunny”, the “Horse”, the “Feline” and the “Venus” (see Figure 5.8 and Figure 5.9) models. These models are commonly used for algorithms comparison by the Computer Graphics research community and can be found in several versions. In the experiments we used the semi-regular models made public available by the Caltech Multi-resolution Modeling Group [130]. Those semi-regular models have been generated by remeshing the corresponding irregular version with the MAPS [46] algorithm. The following tables (5.1), (5.2), (5.3) and (5.4) summarize the characteristics of these models in terms of number of faces, vertices and wavelet coefficients for each level of resolution.

5.6.1 Watermark Perceptibility

In order to achieve high visual quality of the watermarked model we have to carefully consider the problem of watermark perceptibility after its insertion in the model. Nevertheless, the objective evaluation of the perceptibility of the geometric distortion introduced by the watermarking process is a very difficult task to achieve that requires many considerations to be carried out. For this reason we dedicate the next part part of this dissertation to 3D watermarking quality assessment.

The watermarking parameters that influence perception of the watermark are the watermark strength ($\gamma$) and the level of resolution ($l$) used for the embedding. More specifically, more high is the value of $\gamma$, more high is the amount of distortions introduced into the model, and hence the perceptibility of the watermark. The same is for the level of resolution. In fact, low levels of resolution reduce the visual impact of the watermarking process by spread off the distortions over the model’s surface. For all the experiments here reported, we evaluated the watermark distortions by visual inspection, and we used values of $\gamma$ and $l$ that ensure invisibility for all the watermarked models. Stated in another way, all the watermarked models relative to the experiments discussed in this Section are characterized by an high visual quality. An example of the influence of $\gamma$ on the
### Table 5.1: Bunny model information.

<table>
<thead>
<tr>
<th>Level of Resolution</th>
<th>Number of Faces</th>
<th>Number of Vertices</th>
<th>Number of Wav. Coeff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^5$</td>
<td>235,520</td>
<td>118,206</td>
<td>—</td>
</tr>
<tr>
<td>$M^4$</td>
<td>58,880</td>
<td>29,662</td>
<td>88,544</td>
</tr>
<tr>
<td>$M^3$</td>
<td>14,720</td>
<td>7,470</td>
<td>22,192</td>
</tr>
<tr>
<td>$M^2$</td>
<td>3,680</td>
<td>1,894</td>
<td>5,576</td>
</tr>
<tr>
<td>$M^1$</td>
<td>920</td>
<td>486</td>
<td>1,408</td>
</tr>
<tr>
<td>$M^0$</td>
<td>230</td>
<td>127</td>
<td>359</td>
</tr>
</tbody>
</table>

### Table 5.2: Feline model information.

<table>
<thead>
<tr>
<th>Level of Resolution</th>
<th>Number of Faces</th>
<th>Number of Vertices</th>
<th>Number of Wav. Coeff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^5$</td>
<td>516,096</td>
<td>258,046</td>
<td>—</td>
</tr>
<tr>
<td>$M^4$</td>
<td>129,024</td>
<td>64,510</td>
<td>193,536</td>
</tr>
<tr>
<td>$M^3$</td>
<td>32,256</td>
<td>16,126</td>
<td>48,384</td>
</tr>
<tr>
<td>$M^2$</td>
<td>8,064</td>
<td>4,030</td>
<td>12,096</td>
</tr>
<tr>
<td>$M^1$</td>
<td>2,016</td>
<td>1,006</td>
<td>3,024</td>
</tr>
<tr>
<td>$M^0$</td>
<td>504</td>
<td>250</td>
<td>756</td>
</tr>
</tbody>
</table>

### Table 5.3: Horse model information.

<table>
<thead>
<tr>
<th>Level of Resolution</th>
<th>Number of Faces</th>
<th>Number of Vertices</th>
<th>Number of Wav. Coeff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^5$</td>
<td>225,280</td>
<td>112,642</td>
<td>—</td>
</tr>
<tr>
<td>$M^4$</td>
<td>56,320</td>
<td>28,162</td>
<td>84,480</td>
</tr>
<tr>
<td>$M^3$</td>
<td>14,080</td>
<td>7,042</td>
<td>21,120</td>
</tr>
<tr>
<td>$M^2$</td>
<td>3,520</td>
<td>1,762</td>
<td>5,280</td>
</tr>
<tr>
<td>$M^1$</td>
<td>880</td>
<td>442</td>
<td>1,320</td>
</tr>
<tr>
<td>$M^0$</td>
<td>220</td>
<td>112</td>
<td>330</td>
</tr>
</tbody>
</table>
### 5.6. Experimental Results

#### Table 5.4: Venus model information.

<table>
<thead>
<tr>
<th>Level of Resolution</th>
<th>Number of Faces</th>
<th>Number of Vertices</th>
<th>Number of Wav. Coeff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^5$</td>
<td>397,312</td>
<td>198,658</td>
<td>—</td>
</tr>
<tr>
<td>$M^4$</td>
<td>99,328</td>
<td>49,666</td>
<td>148,992</td>
</tr>
<tr>
<td>$M^3$</td>
<td>24,832</td>
<td>12,418</td>
<td>37,248</td>
</tr>
<tr>
<td>$M^2$</td>
<td>6,208</td>
<td>3,106</td>
<td>9,312</td>
</tr>
<tr>
<td>$M^1$</td>
<td>1,552</td>
<td>778</td>
<td>2,328</td>
</tr>
<tr>
<td>$M^0$</td>
<td>388</td>
<td>196</td>
<td>582</td>
</tr>
</tbody>
</table>

visibility of the watermark is shown in Figure 5.10, where the Venus model has been watermarked with different values of $\gamma$.

#### 5.6.2 False alarm rate

The equation (5.38) permits to determine a threshold $T_\rho$ that guarantees the performance of the detector in terms of false detection probability. To validate this theoretical result, we tried to detect 10,000 different watermarks on the same non-marked model for a fixed value of $P_f = P_f^*$; we expect that the actual false alarm rate equals $P_f^*$. The results we obtained are summarized in Table 5.5.

As it is readily seen, the agreement between theory and practice is rather good, with some exceptions for the Feline and the Horse models. The slight differences that can be observed can be explained by considering how watermark coefficients are mapped onto the watermarking map $W_{MAP}$. In fact, as previously stated, if the model has a great density of vertices it may happen that a certain number of different points $\hat{v}(\omega_i)$ are mapped on the same location on $W_{MAP}$. The effect is that $\Delta(\hat{v}(\omega_i), W_{MAP})$ values are no more independent, thus decreasing the actual variance of $\rho$ and causing an actual value of $P_f$ greater than expected. This problem can be alleviated either by embedding the watermark at a lower resolution level or by augmenting the resolution of $W_{MAP}$. Moreover, the high differences for the medium level of resolution ($l = 3$) of the Feline and the Horse model can be explained by considering the vertices distribution of such models. In fact, the particular shape of these models, i.e. the legs and the protrusion of the neck, make the sampling of the $W_{MAP}$ non-uniform. Hence, the statistics properties of the watermark samples, $\mu_{\Delta_D}$ and $\sigma_{\Delta_D}^2$, results different from what
Chapter 5. Blind and robust watermarking of semi-regular meshes

Figure 5.8: Semi-regular models. (Top) Bunny. (Bottom) Horse.
5.6. Experimental Results

Figure 5.9: Semi-regular models. (Top) Feline. (Bottom) Venus.
Figure 5.10: Watermark Perceptibility. (Left) Original Venus model. (Center) Venus model watermarked with $l = 3$, $\gamma = 0.0004$, the watermark is imperceptible. (Right) Venus model watermark with $l = 3$, $\gamma = 0.0020$, the geometric manipulations introduced by the watermarking process become visible.

<table>
<thead>
<tr>
<th>Model</th>
<th>$l$</th>
<th>$P_f^* = 10^{-2}$</th>
<th>$P_f^* = 10^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>1</td>
<td>$1.11 \times 10^{-2}$</td>
<td>$1.1 \times 10^{-3}$</td>
</tr>
<tr>
<td>Bunny</td>
<td>2</td>
<td>$1.35 \times 10^{-2}$</td>
<td>$1.8 \times 10^{-3}$</td>
</tr>
<tr>
<td>Bunny</td>
<td>3</td>
<td>$3.43 \times 10^{-2}$</td>
<td>$8.4 \times 10^{-3}$</td>
</tr>
<tr>
<td>Feline</td>
<td>1</td>
<td>$2.07 \times 10^{-2}$</td>
<td>$2.5 \times 10^{-3}$</td>
</tr>
<tr>
<td>Feline</td>
<td>2</td>
<td>$5.68 \times 10^{-2}$</td>
<td>$17.4 \times 10^{-3}$</td>
</tr>
<tr>
<td>Feline</td>
<td>3</td>
<td>$16.96 \times 10^{-2}$</td>
<td>$95.6 \times 10^{-3}$</td>
</tr>
<tr>
<td>Horse</td>
<td>1</td>
<td>$1.44 \times 10^{-2}$</td>
<td>$2.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>Horse</td>
<td>2</td>
<td>$4.22 \times 10^{-2}$</td>
<td>$10.5 \times 10^{-3}$</td>
</tr>
<tr>
<td>Horse</td>
<td>3</td>
<td>$13.78 \times 10^{-2}$</td>
<td>$69.3 \times 10^{-3}$</td>
</tr>
<tr>
<td>Venus</td>
<td>1</td>
<td>$0.97 \times 10^{-2}$</td>
<td>$1.0 \times 10^{-3}$</td>
</tr>
<tr>
<td>Venus</td>
<td>2</td>
<td>$1.07 \times 10^{-2}$</td>
<td>$1.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>Venus</td>
<td>3</td>
<td>$3.14 \times 10^{-2}$</td>
<td>$6.4 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Table 5.5: Actual false alarm rate for different levels of resolution ($l$). $P_f^*$ is the target value of $P_f$. 
5.6. Experimental Results

theoretically expected. A practical solution to these problem is to estimate the real statistics of the watermark samples by computing the sample average and variance and use such values to compute the threshold $T_\rho$.

5.6.3 Robustness

One of the main problem of 3D watermarking is the wide variety of different attacks possible on a polygonal mesh. This is one of the main reason why many 3D watermarking algorithms use the original model to achieve good robustness properties.

Since our technique is specific for semi-regular meshes, we do not take into account those attacks that alter this property of the mesh such as re-triangulation, simplification or re-meshing. Instead we tested the robustness of the algorithm against additive noise, low-pass filtering, geometric manipulations, cropping and a combination of the above.

Additive noise

Additive noise is a standard attack to evaluate the performance of a watermarking system. In our case the noise is added to the watermarked model by perturbing all the vertices of the watermarked mesh in a random way. More specifically, for each vertex a different displacement vector $\Delta_{\text{noise}} = (\Delta_x, \Delta_y, \Delta_z)$ is applied. The vector components $\Delta_x, \Delta_y$ and $\Delta_z$ are random variables with uniform distribution in the interval $[-\Delta, \Delta]$. In Figure 5.11 the value of $\rho$ and $T_\rho$ for increasing values of $\Delta$ is given. In particular the plot is given as a function of the quantity $\Delta/l_{\text{med}}$, where $l_{\text{med}}$ is the average length of the edges of the model at full resolution. All the models used in this test have been watermarked at a medium level of resolution ($l = 3$) with $\gamma = 0.0004$ and $P_f = 10^{-8}$. The level of resolution chosen guarantees a good trade-off between robustness performance and imperceptibility. In fact, the robustness of the algorithm increases with $\gamma$, but also with the number of watermarked wavelet coefficients. Hence, low levels of resolution provide minor robustness with respect to the high ones. To give a visual idea of the maximum amount of noise the watermark can survive, in Figure 5.12, a noisy version of the Bunny and of the Horse model are shown. The noise strength is set to the maximum level for which the watermark can be recovered. As it can be seen, though the mesh is significantly deteriorated, the watermark is still detectable.
Figure 5.11: Robustness against Additive Noise attack. All the models have been watermarked with the same watermarking parameters ($l = 3$, $\gamma = 0.0004$, $P_f = 10^{-8}$).

Figure 5.12: Robustness against Additive Noise attack. Two examples of models attacked with the maximum amount of noise for which the watermark is detected.
5.6. Experimental Results

Low-pass filtering

Mesh filtering is used by several applications, e.g. to eliminate the surface imperfections of a 3D object acquired by a laser scanner, for editing purposes and so on. Usually mesh filtering is a local surface operations, in other words it affects the details of the high-resolution part of the model without degrading the overall shape of the object. Hence, we expect that by embedding the watermark at a low-medium resolution level a good resistance against this kind of attacks is obtained. This is indeed the case. In particular we evaluated the performance of the watermarking system against the Taubin filter [33]. The coefficients of the filter used in the experiments here reported are $\lambda = 0.6307$ and $\mu = -0.6352$, producing a strong smoothing effect. This filter has been applied to each watermarked model an increasing number of times to evaluate the robustness against smoothing. The watermarking parameters used are the same for the additive noise performance test. The results obtained are summarized in Figure 5.13. The visual impact of the filtering is shown in Figure 5.14.

Geometric transformations

In the absence of further attacks, robustness against geometric manipulations such as translation, rotation and uniform scaling is guaranteed by the normalization phase preceding both watermark insertion and detection. Some problems may arise when a geometric manipulation is accompanied by other attacks that may cause an error in the synchronization stage. Hence, we measured the robustness of the watermark when the marked mesh is filtered, degraded by noise addition and then rotated. In Figure (5.15), the watermarked Bunny model attacked with a rotation of 22° degree around the $x$ axis, 11° degree around $y$ axis, 5 applications of Taubin filter with $\lambda = 0.6307$ and $\mu = -0.6352$ coefficients and noise addition ($\Delta/l_{med} = 0.06$) is shown. The other example we report is the Feline model attacked with 4 applications of the same Taubin filter, noise addition ($\Delta/l_{med} = 0.05$) and rotated by about 25° degree around the $x$ and $y$ axis. In both these cases the watermark was successfully recovered ($\rho = 4.78 \times 10^{-5}$, $T_p = 4.61 \times 10^{-5}$ for the Bunny and $\rho = 3.92 \times 10^{-5}$, $T_p = 2.16 \times 10^{-5}$ for the Feline).

Cropping

Finally, we verified whether the watermarked can be recovered even when a part of the mesh is removed. This may cause synchronization problems. In fact, if a part
Figure 5.13: Robustness against Smoothing Attack. The smoothing is achieved by iterating the Taubin filter. The coefficients of the filter are $\lambda = 0.6307$, $\mu = -0.6352$. For all the models watermarking parameters are the same used for the Additive Noise Attack ($l = 3$, $\gamma = 0.0004$, $P_f = 10^{-8}$).

Figure 5.14: Visual impact of Taubin filter. (Left) Bunny (15 iterations). (Right) Feline (20 iterations). The watermark can be recovered from the smoothed mesh.
5.6. Experimental Results

Figure 5.15: Robustness against combined attacks. (Left) The Bunny model has been watermarked with $l = 3$, $\gamma = 0.0004$ and $P_f = 10^{-8}$, then it has been smoothed by Taubin filter ($\lambda = 0.6307$, $\mu = -0.6352$, 5 iterations), attacked with noise $(\Delta/l_{med} = 0.1)$ and rotated by 22° around x axis and 11° around y axis. (Right) The Feline model has been watermarked with the same watermarking parameters, then smoothed by Taubin filter $\lambda = 0.6307$, $\mu = -0.63502$, 4 iterations), attacked with noise $(\Delta/l_{med} = 0.05)$ and rotated by about 25° around x and y axis. In both cases the detector is still able to recover the watermark.

Figure 5.16: Robustness against Cropping. (Left) Bunny model watermarked with $l = 3$, $\gamma = 0.0004$ and $P_f = 10^{-8}$ and then cut by a plane. (Right) Venus model watermarked with $l = 3$, $\gamma = 0.0004$ and $P_f = 10^{-8}$ and then cut by a plane. In both cases the system is able to recover the watermark.
of the mesh is cut, the position of the center of mass and the orientation of the model change desynchronizing the watermarking map used by the embedder and that available at the detector. Hence in order to verify whether the watermark could be recovered on a subpart of the mesh, we skipped the synchronization step. It goes without saying that the synchronization algorithm needs further improvements in order to ensure full robustness against cropping. For a possible way to achieve this goal see the next Section. Apart from the above considerations, the watermark exhibited an excellent robustness against cropping. Two examples are shown in Figure 5.16.

5.7 Conclusions and Further Improvements

Here, we have presented a new blind and robust watermarking algorithm for 3D objects represented as semi-regular meshes. The proposed technique employs a multiresolution approach in order to achieve both good robustness performances and high control of the watermark insertion. The experimental results demonstrate the robustness of the proposed algorithm against several attacks such as geometric transformations, noise addition, smoothing, cropping and combined attacks. The current implementation of this watermarking system presents two major drawbacks. First, the technique is specific for semi-regular meshes; this limits the applicability of the technique. Second, synchronization stage is not able to deal with a particular kind of attack, i.e. cropping combined with other attacks.

Concerning the limitation to work on semi-regular meshes it is important to remark that any 3D objects can be converted into a semi-regular models. The real problem of applicability is that the robustness of the technique is compromised by those attacks that destroy the semi-regularity of the mesh. Furthermore, an appropriate pre-processing to reconstruct the semi-regularity of the model starting from its shape it is required. This can be achieved by an ad-hoc remeshing operation applied in the first part of the synchronization stage to re-synchronize the sampling and the connectivity of the input mesh without resorting to the original model.

The de-synchronization problems due to cropping can be easily overcome by subdividing the mesh to watermark in sub-parts and then apply the watermarking procedure separately to each sub-mesh. Several algorithms to achieve this goal are available [131, 132, 133].

Concluding, we can state that even if the overall performances of the devel-
opend watermarking algorithm are good, especially with respect to the algorithms
developed so far, further improvements are opportune. In particular, the role
of remeshing to reconstruct semi-regularity on the attacked watermarked models
require further investigations.
Chapter 5. Blind and robust watermarking of semi-regular meshes
Part III

Visual Quality Assessment of Watermarked 3D Objects
Chapter 6

Visual Appearance of 3D Models

Three-dimensional objects find uses in several applications, from virtual design to videogames and movies. One of the most typical use of a 3D model is viewing. In other words, often, the most important characteristic of a 3D model, is its visual appearance. The way in which a 3D object is visualized depends on the application context, on the rendering technique used to display it, on the richness of the visual information of the model, on the graphics capability of the graphics hardware of a specific machine, and so on. Hundreds of possibility exist; from methods that display a simple graphical idea of the object to complex rendering algorithms that create images with an incredible level of photo-realism. So far, we have considered mainly the geometry of the 3D models and how to watermark it. The purpose of this Chapter is to provide a panoramic of the techniques used to render a 3D model. These notions are important to understand the enormous number of factors that influence the visual aspect of a 3D object, and useful for the evaluation of the visual distortion introduced by the 3D watermarking process, that will be the topic of the next Chapter.

6.1 Rendering Techniques

The term rendering indicates the set of methods and algorithms used to generate a two-dimensional image starting from a scene described by geometric primitives. The results of the rendering process, i.e. the final rendered image, depends on the rendering techniques used and on the visual properties of the rendered scene.
Nowadays, several rendering techniques exist. One first important distinction subdivide these techniques in two categories: the real-time rendering (RTR) methods and the off-line rendering ones. For real-time rendering we intend all of those techniques capable to generate the images so quickly to allow interaction. The sense of interactivity with the 3D scene is constrained to the rate at which the images are displayed, measured in frames per seconds (fps). Low frame rate, from 4 to 10 fps, gives a first sense of interaction. A frame rate of 15-30 fps provides good interaction, the system is able to react quickly to the action of the user. The typical frame rate of videogames is around 50; the interaction is fluid and very quick in order to respond to the frenetic actions of the gamer. Over 70 fps no noticeable differences exist. In contrast to real-time rendering, off-line rendering techniques process the 3D scene without the aim of interaction. In this case, the accent is pose on the creation of realistic images. Some popular off-line rendering methods are Ray Tracing, Radiosity and Photon Mapping. In the next we give an introduction of the core of any real-time rendering system, that is the rendering pipeline, and a description of the basic idea of Ray Tracing by presenting a simple Ray Tracing algorithm. Such algorithm will be extended in Section 6.4, where an overview of other photo-realistic rendering techniques, such as Monte Carlo Ray Tracing and Photon Mapping will be presented.

6.1.1 The rendering pipeline

The graphics rendering pipeline is the core of any real-time rendering system. The function of the rendering pipeline is to generate, or render, a bi-dimensional image given a virtual camera, a description of the object to render, in terms of geometric data and appearance data, and a lighting model. Obviously several 3D objects can be joined together to form a complex three-dimensional scene. The term “pipeline” is used to underline that this process is subdivided in stages that act on the 3D data as other common pipeline processes. In an oil pipeline, for example, the oil cannot move from the first stage to the second one unless the oil in the second stage moves in the third one and so on. Hence, the slowest stage determines the speed of the overall pipeline. In other words, a pipeline process can be subject to bottleneck; if one stage is particularly slow with respect to the other stages the pipeline becomes inefficient. The graphics rendering pipeline can be composed, from a conceptual point of view, essentially by three different stages: the application stage, the geometry stage and the rasterizer. Each of these stages can be further subdivided in sub-stages which depend on the architecture of the
pipeline.

The application stage, as the name implies, depends on the particular application of the graphics system. The user interacts with the application and his/her actions determine what have to be visualized. The data to be rendered are sent to the next stage, i.e. the geometry stage.

The geometry stage determines what is to be drawn, how objects are to be drawn and where objects are to be drawn. Generally, the modern graphics pipeline assume meshes representation for the 3D models. Objects can be also modelled using NURBS or other representations but in those cases they must be converted to polygons at render time. The majority of the per-polygon and per-vertex operations are performed in this stage. A typical geometry stage can be subdivided in sub-stages in the following way:

![Rendering Pipeline Diagram](image)

Figure 6.1: The rendering pipeline: geometry stage.

Depending on the graphics hardware parts or the entire geometry stage can be implemented in hardware. The input of the first stage, i.e. “Model and View Transformations” are the coordinates of each model, in object space. In fact, each 3D object is modelled individually in its own coordinate system, hence it is necessary to transform all objects in a common coordinate system called 3D world coordinate system. The objects are translated, rotated, and scaled in order to compose correctly the final 3D scene in the 3D world coordinates system. This is what concern the Model transformation. The View transformation regards the conversion of the 3D world coordinates to the eye coordinates system, determined by the position and the direction of viewing of the virtual camera. The scene is modified to place the camera at the origin. At this point vertex normals are computed for each vertex and a lighting model (see section 6.3) is applied to illuminate the scene. After lighting, the view volume of the scene is computed on the basis of the projection used, that can be orthographic or perspective. The view volume of a rectangular projection is a rectangular box, while the view volume of a perspective projection is a frustum, i.e. a truncated pyramid with rectangular base. The parts of the scene outside the view volume are clipped before the mapping on the screen. Each vertex in screen coordinates is characterized by two coordinates relative to its position on the screen plus a value identifying its depth.
Chapter 6. Visual Appearance of 3D Models

The goal of the rasterizer is to determine, on the basis of the output of the geometry stage, that is the projected vertices, colors, and texture coordinates, the color of each pixel of the final image. This process is called rasterization or scan conversion. This stage performs per-pixel operations. The rasterizer is responsible also to resolve the visibility of each polygon using the z-value information (the depth information) of each vertex. For further details about the rendering pipeline we refer to the excellent book on real-time rendering of Thomas Akenine Möller and Eric Haines [134].

6.1.2 Ray Tracing

Ray Tracing is one of the most used rendering methods to produce photo-realistic images. In ray tracing, a ray of light is traced in a backward direction, i.e. the ray starts from the eye of the observer and it is traced through a pixel in the image plane into the scene determining what it hits. The pixel is then set to the color values returned by the ray. This basic idea is repeated for thousands of rays opportunely in order to sample the entire image plain and so produce the final image (see Figure 6.2). Ray tracing differs from the rendering pipeline described above since the act of tracing rays through space implicitly accomplishes the perspective transformation, the clipping phase, and the hidden surfaces removal.

In its basic form ray tracing can be described by the following algorithm:

- For each pixel of the images:
  1. Construct a ray from the viewpoint
  2. For each objects in the scene
     2.1. Find intersection with the ray
     2.2. Keep the closest intersection point
  3. Shade the point the ray hits

This simple algorithm could be highly computationally expensive due to the huge number of intersection tests needed to compute the location of the scene the ray hits. Several acceleration techniques to reduce the computational burden of the calculus of each ray-scene intersection have been developed. Some of them are based on caching [135, 136, 136]; such techniques exploit the fact that near rays with high probability will hit the scene around the same location. Other techniques to speed-up ray tracing are based on spatial partitioning of the scene [137, 138, 139] in order to efficiently compute rays/objects intersections. Also, this basic ray
tracing scheme could be modified to take into account sophisticated photorealistic visual effects, such as soft shadows or caustics. Such extensions will be discussed in Section 6.4.

6.2 Illumination Models

A lighting model is a mathematical description of the interaction between the light incident on a 3D object and its surface. Such models range from simple to very complex ones. More high is the complexity of the lighting model more high is the level of simulation of the physical interaction between the incident light and the material of the 3D objects, in other words more high is the degree of photo-realism. Typically real-time rendering systems require to find the right trade-off between level of realism, that requires a high computational effort, and an acceptable frame rate. On the contrary, off-line rendering systems may use sophisticated lighting models.

Before describing some classical lighting models we introduce some basic notions about light-material interaction. Figure 6.3 summarizes what happens when the photons hit a generic surface. Part of the incident light is reflected, part is absorbed, part is transmitted and so on. In formula:

$$I_i = I_r + I_t + I_s + I_a$$

where $I_i$ is the incident light, $I_r$ is the light reflected by the material, $I_t$ is the light transmitted, $I_s$ is the light scattered and $I_a$ is the light absorbed by the material. The reflected light can be described by considering two different effects, the diffusion reflection and the specular reflection. The diffusion reflection is responsible of the color of the objects. A yellow object, for example, when it is illuminated with a white light, reflects the yellow component of the light. The
colored reflected light is due to the diffuse reflection. A perfect diffusive surface scatters light uniformly in all directions, hence the diffuse light does not depend on the observer’s position. The intensity of the diffuse light can be computed by Lambert’s Law \cite{140}:

\[ I_d = I_i K_d \cos(\theta) \]  

(6.2)

where \( \theta \) is the angle between the surface normals at the considered point and a line connecting such point with the light sources. In the following we indicate the direction of such line with the versor \( \vec{L} \). The maximum light received by the surface is when the surface normal is parallel to the direction of the incident light, i.e. the surface is perpendicular to the incident light. The specular reflection depends on the degree of glossiness of a surface. A matt surface has no specular effect but high diffusive behavior, a perfect glossy surface is a mirror. The light reflected from a mirror leaves the surface with the same angle of incidence, computed with respect to the surface normals. Hence, the amount of specular light seen by the viewer depends on the viewer’s position. Simplifying the real phenomenon, we can say with a good degree of approximation that the color of the specular light is the same of the incident light, i.e. the highlight\(^1\) color of an object illuminated with a white light is white.

### 6.2.1 Light Sources

The first step to simulate lighting is to model the light sources. Typically simple lighting models approximate the light sources as a point sources since volumetric lights are complex to simulate. The three kinds of light sources commonly used

\(^1\)The highlight is the name of the area over the specular reflection is seen.
are the directional light, the point light and the spot light. Figure 6.4 summarizes these three kinds of light sources.

The point light models a light source as a point that emanates photons uniformly in all direction. This kind of light source is characterized completely by its position and by the color of the emitted light.

The directional light can be seen as a point light sources moved at infinity. This light is completely defined by the light direction. Hence, the direction of the light received by the model is constant over all its surface. This kind of light can be used, for example, to model the light emitted from the sun in outdoor scenes.

The spot light simulates a cone of light. This kind of light requires several parameters to be defined: the position, a vector indicating the direction of the light and a cut-off angle, which is the half of the angle of the spotlight cone. Finally, to control the attenuation of the light within the cone, another parameter called spot exponent might be used. The spot exponent modulates the concentration of the light distribution in the central part of the cone.

Usually the color of the light emitted by the light source is defined by a set of three values representing the RGB components of the color. Another important property of the light emitted by a light source is its intensity. The intensity of the light decreases with the square of the distance from the light source. Often, in real-time rendering systems this physical property is not taken into account, and the intensity of the light received by the objects is assumed independent from the distance to the source.
Chapter 6. Visual Appearance of 3D Models

6.2.2 Phong Model

The Phong lighting model [141] is the “standard” model used in Computer Graphics. This simple model has been developed by Bui-Tuong Phong in 1975 on the basis of empirical observations. This model provides a good trade-off between accuracy and complexity. Phong described the reflected light using three terms:

\[ I_{\text{Phong}} = I_{\text{amb}}K_a + I_iK_d(\vec{N} \cdot \vec{L}) + I_iK_s(\vec{R} \cdot \vec{V})^n \]  

(6.3)

where \( \vec{N} \) is the surface normal, \( \vec{L} \) is the vector that represents the direction of the incident light, \( \vec{V} \) is the vector that describes the direction of the line connecting the viewer with the considered point (see Figure 6.5). The first term of the equation (6.3) is the ambient term; this term models the light that the object receives from the surrounding environment. The second term models the diffusion component of the reflected light. This term follows the Lambertian law (6.2), in fact it is proportional to the dot product between the (normalized) vectors \( \vec{N} \) and \( \vec{L} \). If \( \vec{N} \cdot \vec{L} < 0 \) the point receives no light. The third term models the specular light, in fact is proportional to the light reflected in the direction of the viewer. The coefficient \( n \) depends on the surface’s material.

6.2.3 Cook and Torrance

The first reflection model based on physical considerations to model the specular light was proposed by James Blinn [142]. Instead to provide an empirical formulation as Phong’s model, Blinn’s model was based on a surface model introduced by Torrance and Sparrow (1967) [143]. Such surface model assumed that a surface is composed of a collection of microfacets, each of them behaving like a mirror. The distribution of the directions of the microfacets determines the specular component of the light. Cook and Torrance [144] enhanced Blinn’s model by introducing two new physical aspects in it: energy conservation between the incident and the
6.2. Illumination Models

Microfacets model.

Masking effect of the microfacets.

Shadowing effects of the microfacets.

Figure 6.6: Microfacets model of a reflecting surface.

reflected light and the change of the color within the specular highlight. According to the Torrance and Sparrow model the specular component of the light can be modeled as:

$$R_s = \frac{DG F_\lambda(\theta_i)}{\pi(N \cdot L)(N \cdot V)}$$

(6.4)

where the term $D$ depends on the distribution of the microfacets’ directions, $G$ is a geometric attenuation factor and $F_\lambda(\theta_i)$ is the Fresnel term.

To model the microfacets’ directions Cook and Torrance used a distribution proposed by Beckmann and Spizzichino [145]:

$$D = \frac{1}{m^2 \cos^4(\alpha)} \exp\left(\frac{\tan^2(\alpha)}{m^2}\right)$$

(6.5)

where $m$ is the root-mean-square (RMS) of the slope of the microfacets and $\alpha$ is the angle of the microfacets with respect to the surface normals at the considered point.

The geometric attenuation factor $G$ takes into account the masking and shadowing effect of the microfacets (Figure 6.6). The masking effect can be mathematically expressed in the following way:

$$G_m = \frac{2(N \cdot \vec{H})(N \cdot \vec{V})}{(\vec{V} \cdot \vec{H})}$$

(6.6)

where the $\vec{H}$ is a (normalized) vector pointing halfway between $\vec{L}$ and $\vec{V}$, i.e. $\vec{H} = (\vec{L} + \vec{V})/2$. The shadowing effect presents the identical geometric aspects with the roles of the vectors $\vec{L}$ and $\vec{H}$ interchanged:

$$G_s = \frac{2(N \cdot \vec{H})(N \cdot \vec{L})}{(\vec{V} \cdot \vec{H})}$$

(6.7)

Thanks to the previews definition the factor $G$ could be defined as:

$$G = \min(1, G_m, G_s)$$

(6.8)
Chapter 6. Visual Appearance of 3D Models

Lambertian model  Phong model  Cook and Torrance model

Figure 6.7: Lighting models. (from [146])

The Fresnel term accounts for the color change of the specular highlight as a function of the angle of incidence and of the wavelength, i.e. the color, of the light. In general the color of the specular highlight depends on the physical characteristics of the material except when the value of the angle of incidence of the light is low.

To account for energy conservation Cook and Torrance consider energy instead of intensity of the light. The total reflectance of the light results a linear combination of the diffuse and specular components:

\[ R_{bd} = K_d R_d + K_s R_s \] (6.9)

where \( R_{bd} \) is the reflected intensity for one particular direction \( R_d \) is the radiance of the diffuse component and \( R_s \) is the radiance of the specular term. Energy conservation is achieved by adding the constraint \( K_s + K_d = 1 \). The equation (6.9) will become more clear in the next section where a brief explanation of the concepts about BRDF lighting are provided.

Figure 6.7 shows an example of application of Lambertian, Phong and Cook and Torrance model. The Lambertian model is based on the Lambert’s Law (6.2) and take into account only the diffuse component of the reflected light.

6.2.4 BRDF

The Bi-directional Reflectance Distribution Function (BRDF) is a mathematical description of the manner in which the incident light is scattered by a particular surface. The BRDF was introduced by Nicodemus [147] as a simplification of the more general Bi-directional Surface Scattering Reflectance Distribution Function (BSSRDF). The BRDF is function of the incoming (light) direction, of the outgoing (view) direction, of the position on the surface, and of the wavelength (the
6.2. Illumination Models

Figure 6.8: BRDF angles.

color) of the light. Typically the incoming and outgoing directions are represented in spherical coordinates. In the next we indicate with \((\theta_i, \phi_i)\) the incident direction and with \((\theta_o, \phi_o)\) the outgoing direction assuming that \(\theta\) is the elevation angle and \(\phi\) is the azimuth angle (see figure 6.8). Since the surface’s properties are often considered homogeneous, the dependency on the position is not considered. More specifically, BRDF is the ratio between the radiance in the outgoing direction and the irradiance incident at a surface. The irradiance measures the amount of energy per time (the power) of the incident light in terms of flow through an area. So, the unit of measure of the irradiance is \(\text{Watt/m}^2\). To determine the amount of energy received by the surface element the irradiance contribution must be spread out onto the surface element:

\[
L(\theta_i, \phi_i) = \cos(\theta_i) d\omega_i
\]  

(6.10)

where \(d\omega_i\) is the differential solid angle around the direction \((\theta_i, \phi_i)\). The radiance measures the power of the outgoing light traveling in a specific direction, per area unit, perpendicular to the direction of travel, per solid unit angle (\(\text{Watt/m}^2/\text{sr}\)). Hence, the BRDF can be expressed as:

\[
\rho_{o,d}(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{L_o(\theta_i, \phi_i, \theta_o, \phi_o)}{L_i(\theta_i, \phi_i) \cos(\theta_i) d\omega_i}
\]  

(6.11)

where \(L_o(.)\) is the radiance in the outgoing direction and \(L_i(.)\) is the incident irradiance. Note that \(L_o(.)\) is function of the both incoming and outgoing directions. Since the solid angle is measured in steradians (sr), i.e. square radians, the BRDF is measured in inverse steradians (sr\(^{-1}\)).
Chapter 6. Visual Appearance of 3D Models

A generic BRDF, to be physically plausible, must respect two important properties: reciprocity and energy conservation. The property of reciprocity states that the values of the BRDF must be the same even if the incident and the outgoing directions are swapped, i.e. $\rho_{bd}(\theta_i, \phi_i, \theta_o, \phi_o) = \rho_{bd}(\theta_o, \phi_o, \theta_i, \phi_i)$. Reciprocity underlies the theoretical validity of the backward rendering used by the Ray Tracing [148]. The property of energy conservation states that the total amount of light reflected must be less than or equal to the amount of incident light. This constraint can be formulated mathematically considering the sum of the radiance reflected in all directions:

$$\int_{\Omega} \rho_{bd}(\theta_i, \phi_i, \theta_o, \phi_o) \cos(\theta_o) \, d\omega_o \leq 1 \quad (6.12)$$

where $\Omega$ indicates the entire hemisphere.

To use a BRDF for lighting we have to determine the values of the BRDF for each incoming and outgoing direction. This can be achieved in two different ways. In the first way the values are derived from an analytical model, like the Phong and the Cook and Torrance ones. For example, the Cook and Torrance model can be rewritten as [149]:

$$\rho_{bd} = \frac{D \rho_{\lambda}(\lambda, \theta_i) \cos(\theta_o)}{\cos(\theta_o)} + \frac{\rho_{d}}{\pi} \quad (6.13)$$

where the first term is the specular component and the second term the diffuse component. The factor $\pi$ normalizes the reflectance distribution over the whole hemisphere, since the solid angle of a hemisphere is $2\pi$.

The second way consists of acquiring BRDF data directly from the real world. Device like the gonioreflectometer are able to measure the BRDF of a material. Hence, in this case, the BRDF is represented by a lookup table with 4 entries.

6.2.5 Basic Shading Techniques

The term shading is the process of performing lighting computations, on the basis of the chosen lighting models, and determining the colors of the pixels. The basic shading techniques are flat, Gouraud and Phong shading [140]. Referring to polygonal meshes, these techniques correspond to compute the light per-face, per-vertex and per-pixel, respectively.

In flat shading, the light is computed for each triangle using the face normal. Hence, the effect of shading is highly dependent on the level of detail, i.e. the

---

2The Phong shading must not be confused with the Phong lighting model. The last one is a reflection model while the former one is a way to interpolate the values of light obtained from a generic lighting model.
number of faces, of the objects that are rendered. It could be useful when the visualization purpose is to well-distinguish the faces that compose the model.

In *Gouraud shading* the light is computed for each vertex using the vertex normals. The values of light over each face are interpolated. This method provides better visualization of the curved surface than the flat shading, i.e. the curve looks more smooth and realistic. Some problems of the Gouraud shading include missing highlights, failure to capture spot light effects and Mach banding [150, 151].

In *Phong shading* per-pixel lighting is computed by interpolating the vertex normals instead the color of the vertices as in Gouraud shading. Phong shading is rarely used since it is computationally expensive and Gouraud shading can provide the same visual results if the triangles of the model are smaller than a pixel.

### 6.3 Texturing

*Texturing* is a process to modify the appearance of a surface locally by using images or repeating motifs. In other words the surface properties are modulated by particular images called *textures*. For example, if we want to render a brick wall the image of a brick wall can be spread over planar polygons giving the impression of the geometric details of the wall, even if they are not present in the geometric data of the wall. This kind of texturing is called *image texture mapping*. Another example could be the modulation of the surface transparency to simulate particular objects, such as clouds. Alternatively, the modulation of the surface’s properties can be achieved by using bi- or three-dimensional functions instead of use textures. In the following we give a brief description of some texturing methods such as *image texture mapping*, *bump mapping* and *gloss mapping*. Other kind of mapping, such as alpha mapping and environment mapping are not discusses here. Several kinds of texturing can be combined in a multi-texture framework to augment considerably the visual richness of the rendered 3D object.
6.3.1 Texture Mapping

Before describing some texturing methods we explain how to associate the points over the polygonal surface to the texture. In general, each vertex of the 3D object, expressed in the object space \((x, y, z)\), is associated to a suitable parameter space \((u, v)\), typically through projector functions. Then, a corresponder function is used to extract the texel\(^3\) from the texture. The value of the texel is then used to modify the color, computed by the lighting and the material properties, of the considered point.

Projector functions work by converting a three-dimensional point in space into texture coordinates. The most commonly used projector functions include spherical, cylindrical and planar projections [152, 153]. The spherical projection casts points onto an imaginary sphere centered around some location. Cylindrical projection compute the \(u\) texture coordinate in the same way of spherical projection, the \(v\) texture coordinate is associated to the cylinder’s axis. This projection is useful for objects that have a natural axis. The planar projection act as an slide projector, projecting along a direction and applying the texture to the whole surface. It uses orthographic projection. In Figure 6.10 several projection functions are shown. For complex multi-textured model the artist has to manually decompose the mesh in subparts and associate to each of these parts the proper textures.

An alternative to use projections, it is to map directly parts of the mesh onto the texture by using the mesh parameterization techniques seen in Section 4.4. Such way of proceed is more computationally expensive and complex to achieve but make the work of the artists easier and allow to obtain more accurate results [110].

The corresponder functions convert parameter-space values, i.e. the texture coordinates, in a texture-space location. Typically, texture coordinates range from

\(^3\)A pixel of a texture is called texel.
0 to 1. If the resolution of our texture is, for example, 256 × 256 the \((u, v)\) coordinates must be multiplied by a scale factor of 256 to obtain the location of the texel. The corresponder function provide flexibility in applying the texture, for example the texture coordinates can be applied to a rotate and scaled version of the same texture. Another example is the control of repeating motifs; a corresponder function can map the \((u, v)\) coordinates such to obtain a repeated tiling of the same texture.

### 6.3.2 Image texturing

In image texturing, one or more images are applied on the model’s surface. As previously stated, the main problem is the association between the texture coordinates and the surface, in other words to parameterize the surface. The texture image used may be magnified or made smaller depending on the mapping. Since the texture image is a discrete collection of values, in case of magnification interpolation is necessary. The most common interpolation used for texture magnification is the nearest neighbor method or the bilinear interpolation. On the contrary, in the case of texture minification, when more texels cover the same pixel, the influence of each texel should be considered. One possible solution is to use the nearest neighbor method as for texture magnification, i.e. taking the texel closer to the center of the pixel's cell, but this filter may cause severe aliasing problems. An advanced solution that avoids aliasing is mipmaping [154]. For further detail about texture antialiasing we refer to [155].

### 6.3.3 Bump Mapping

Bump mapping techniques use the texture information to modulate the surface normals. The actual shape of the surface remains the same, but thanks to bump mapping the surface is rendered as if it were a different shape with more details. Bump maps is used to simulate complex features of the surface that, otherwise, would require a lot of geometric details to be represented, such as musculature in animals, folds in clothes, and so on. The concept of bump mapping was first introduced by Blinn [156]. Nowadays several ways to achieved bump mapping have been developed.

One way to represent bumps is to use a heightfield to modify the surface normal’s direction (see Figure 6.11). Each monochrome texture value represents a height. The slope of the normals in the \(u\) and \(v\) directions is obtained by derivation of heightfield’s values along the \(u\) and \(v\) axis. Another bump mapping
Chapter 6. Visual Appearance of 3D Models

Figure 6.11: An example of Bump Mapping.

The method is the *Emboss Bump Mapping* [157], that is based on the *embossing* [158], an image processing technique to give a chiselled look to a given image. The *dot product bump mapping* [159], also called *dot3 bump mapping* is the primary method of performing bump mapping on graphics hardware. In this method the actual normals of the surface are stored as \((x, y, z)\) vectors in a special texture called *normal map* [160, 161, 159]. The normal map is an RGB image such that each pixel represents a vector. In particular, each vector is represented by normalizing its coordinates in the \([-1, 1]\) interval and then by encoding each coordinate using 8 bits, i.e. within 0 and 255. For further details about emboss, dot3, and others bump mapping techniques we refer to [134].

### 6.3.4 Gloss Mapping

This kind of texturing is used to simulate non-uniformity in shiny surfaces. More specifically, a *gloss map* is a texture that modulates the contribution of the specular component over the surface. The idea at the base of gloss mapping is that the material properties can be encoded with textures, instead of use per-vertex values. Figure 6.12 shows an example of gloss mapping.

### 6.4 Global Illumination

The concept of *global illumination* is at the base of the generation of synthetic images with an outstanding level of realism. The illumination models presented so far are not able to capture a lot of sophisticated visual effects that are present in
6.4. Global Illumination

Figure 6.12: Gloss mapping. The shininess of the shield on the right is modulated in order to appear more realistic.

real scenes. In fact, those models are local, i.e. they do not take into account the contributions of the indirect light, i.e. the light coming from the others objects in the scene. In this sense, Ray Tracing is a first step in the direction of global evaluation of the illumination. The basic Ray Tracing algorithm proposed above can be easily modified to properly render shadows and reflections. More complex algorithms such as Path Tracing or Photon Mapping are necessary to taking into account many of the global illumination visual effects present in the real scene. These effects (see Figure 6.13) are:

- **Indirect light.** The light received by an object is not only the light emitted from the light sources but also the light reflected (or diffuse) by the other objects. In Figure 6.13 we can notice that the light reaches the roof of the box thanks to the contributions of the indirect light.

- **Soft shadows.** The contour of the shadows in real images is smooth due to the fact that real light sources are volumetric.

- **Color bleeding.** This particular effect of the indirect light regards the fact that the color of an object is influenced by the color of the near objects. Such visual effect depends on the light diffuse by the near objects. In Figure 6.13 it is possible to see that the the roof is red near the red wall.

- **Caustics.** The caustics are region of the scene where the reflected light is concentrated. An example is the light concentrated around the base of a glass. Such effect require global illumination techniques to be properly simulate.
In simple words, the goal of global illumination rendering techniques is to trace all photons through a scene in order to simulate all the global illumination visual effects that characterize a real scene. States in a mathematical way, global illumination techniques attempt to solve the, so-called, rendering equation (Kajiya, 1986 [162]):

\[
L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} \rho_{bd}(x, \omega_i, \omega_o)L_i(x, \omega_i)(\omega_i \cdot \vec{n})d\omega_i
\]

where, in this notation, the vector \( \omega_i \) represents the incoming light, the vector \( \omega_o \) the outgoing direction, \( x \) is the position over the model’s surface, \( \vec{n} \) is the surface normal and \( L_e(.) \) is the light emitted by the material.

The idea at the base of many algorithms for global illumination is to solve the rendering equation (6.14) by using Monte Carlo Integration. In a nutshell, Monte Carlo Integration theory states that, the integral of a function \( f(.) \) defined over the interval \([a, b]\):

\[
I = \int_{a}^{b} f(x)dx
\]

(6.15)
can be estimated by sampling \( f(x) \) a huge number of times, in particular:

\[
I_{MC} = (b - a) \frac{1}{n} \sum_{i=1}^{n} f(X_i)dx
\]

(6.16)

where \( X_i \) are samples uniformly distributed over the interval \([a, b]\). For \( n \to \infty \) the value of \( I_{MC} \) converges to the exact value of the integral \( I \).
Kajiya [162] first propose to use Monte Carlo sampling to solve the (6.14) with a rendering technique called path tracing. Path Tracing is an extension of Ray Tracing that consists to shoot many rays for each pixel and randomly shooting new rays at each object/ray interaction. More specifically, the standard Ray Tracing algorithm, with these modifications, becomes:

- For each pixel of the images:
  1. Construct a ray from the viewpoint
  2. For each objects in the scene
     2.1. Find intersection with the ray
     2.2. Keep the closest intersection point
  3. From the point the ray hits:
     3.1. Pick a random number \( r \in [0, 1] \)
     3.2. if \( r < K_d \) send a “diffuse ray” (shoot a new ray in a random direction)
     3.3. if \( r < K_d + K_s \) send a “reflected ray” (shoot a new ray along the reflection direction)
     3.4. else absorb the ray
  4. Sum up the contribute of all rays and determine the pixel’s color

All the rays shoot contribute to determine the color of each point of the scene. The mechanism to shoot a new ray (step 3) are usually called Russian roulette. Path Tracing works well when the effects of the indirect illumination vary slowly. Hence, this techniques is not capable to produce realistic caustics. The main drawback of this method is that an huge number of rays are necessary to obtain high-quality images. Some extensions to path tracing have been proposed in those last years, for example the bi-directional path tracing by LaFortune and Willems [163] and the Metropolis light transfer by Veach and Guibas [164].

The state of the art algorithm in global illumination is the Photon Mapping technique developed by Henrik Wann Jensen [165]. Photon mapping is a two-pass algorithm capable to solve the rendering equation. In the first pass, called photon tracing photons are emitted from the light sources and traced through the three-dimensional scene. When a photon hits a surface, a new photon is shoot with the Russian roulette mechanism. A special data structure called photon map store the data of the photons. In particular, two distinct photon maps are built during photon tracing, one for the global illumination and another one specific for rendering caustics. In the second pass the data of these two photon maps are used by a modified Ray Tracing algorithm to calculate direct illumination, the contributions of indirect illumination, specular reflections and caustics.
For a detailed description of Photon Mapping, and further details about global illumination, we refer to the Jensen’s book [166].

6.5 Non-Photorealistic Rendering

Non-Photorealistic Rendering (NPR) concerns the creation of images that do not inspire to realism. Since now we have exposed several ways to obtain images that look like natural ones. But photorealism is not appropriate for certain applications; for example in scientific visualization for communicating information about physical structures or phenomena, in technical illustrations where the designer poses the accent on the communication of ideas, shapes and actions through images instead of mimic the reality, and, more generally, in artistic productions. The term Non-Photorealistic Rendering arises since this branch of Computer Graphics is relatively recent and difficult to summarize, so it is more convenient to specify what it is not than what exactly NPR concerns. In this Section we give a short overview of some visual effects that can be obtained with different NPR techniques.

Historically, NPR research has its origin in 2D interactive paint systems. These systems provide new forms of expression to the digital artist making available special brushes and pencils to create several pixel-based effects. Nowadays the research on NPR is focused on 3D techniques, and in particular on real-time NPR techniques since the rendering pipeline exposes a lot of opportunities to achieve an enormous number of different effects. Teece [167] provides a first classification of NPR systems subdividing them in 2D with user intervention, 2D without user intervention, 3D with user intervention and 3D without user intervention. Here we are interested in the lasts two kinds of systems.

Often NPR techniques attempt to reproduce styles typical of paintings or other styles like the ones used in artistic illustration, such as sketch, pen and ink, hatching, etc. Two nice examples of painterly rendering are the work of Barbara Meier [168], that proposes a method to extend painterly rendering of still images [169] to animations, and the work of Cassidy Curtis [170] for simulating watercolor images. To be more precise Cassidy’s technique is classified as a 2½D method by Teece, but, with minor modifications this method can be used to make non-photorealistic rendering animations [171].

With regard to techniques related to artistic illustration, Winkenbach and Salesin [172] developed one of the first automatic 3D rendering systems that pro-
vides emulation of pen-and-ink illustration by rendering the geometry of the model in conjunction with stroke textures. Other interesting recent examples about this style are the works on hatching of Praun et al. [173] and Webb et al. [174] (Figure 6.18). A lighting model that tries to mimic the expressive richness of the hand-drawn technical illustration was developed by Gooch et al. [175]. Gooch’s model uses luminance and changes in hue to indicate surface orientation, and gives a clear picture of the shape of the model (Figure 6.19).

Of particular commercial interest are techniques that render 3D scenes in styles which match the “look” of traditionally animated films. Usually referred to as toon shading, those techniques allow for seamless combination of 3D elements with traditional cel animation. Recently, toon shading has been used in the productions of several movies, like “Monster and Co.,” “Finding Nemo” and “Shrek 2”, and videogames, such as “Cel damage” and “The Legend of Zelda” (Figure 6.15).

Today, most of the previously cited NPR techniques can be found in several commercial rendering tools that put emphasis on non-photorealistic rendering such as Kazoo [176] and Piranesi [177] systems. Piranesi is a rendering system with “3D painting” functionality useful to enhance the communication effects, in particular in architectural projects presentations (Figure 6.16).

6.6 Final Considerations

The visual appearance of 3D objects is determined by an enormous number of factors: the lighting model used, the shading techniques employed, if textures are applied to the model, the properties of the surface’s material, the level of photorealism required by the application, the particular visual effects used for specific purposes, etc. This wide range of variability in the final result of model appearance makes the evaluations of visual artifacts introduced by 3D watermarking algorithms a difficult and complex task. But such aspect is crucial to obtain high-quality watermarked models. Hence, investigations in this direction are very suitable. In the next Chapter we present our studies in the perceptual evaluation of 3D watermarking impairments.
Figure 6.14: Painterly rendering for NPR animations.

Figure 6.15: Toon shading. (from *The Legend of Zelda* ©)

Figure 6.16: Example of rendering obtained with *Piranesi* rendering system.
6.6. Final Considerations

Figure 6.17: Pen-and-ink rendering. (from [172])

Praun et al. (from [173])

Webb et al. (from [174])

Figure 6.18: Hatching rendering.

Figure 6.19: Non-photorealistic lighting models for technical illustration. (from [175])
Chapter 7

3D Watermarking Quality Assessment

7.1 Introduction

In this Chapter, we propose two perceptual metrics for the quality assessment of watermarked 3D objects. The reasons for proposing perceptual metrics are the evaluation and comparison of perceptual artifacts introduced by 3D watermarking algorithms. The final aim of evaluation is to minimize extraneous details introduced by watermarking by modulating the watermark insertion in order to obtain little or no perceptual artifacts. The second is to use such metrics for comparing the performance of different 3D watermarking algorithms on the basis of the artifacts perceived on the 3D model.

Effectively, many perceptual metrics have been proposed in the image domain. A possible approach could be to simply apply such image-based perceptual metrics to the final rendered images of the 3D model. The main problem of this approach is that the perceived degradation of still images may not be adequate to evaluate the perceived degradation of the equivalent 3D model. The approach we chose is to evaluate the human perception of geometric defects of watermarked models and then to build an ad-hoc perceptual metric that works directly on the model’s surface. In such a case, subjective experiments dealing directly with the 3D models are needed. In particular, we propose two subjective experiments with different purposes. The first experiment (Experiment I), is carried out to investigate the perception of artifacts caused by our watermarking algorithm on 3D models and to find suitable metrics to measure artifacts’ perceptual severity. On the basis of
the subjective data collected with this experiment two metrics based on roughness estimation of the model’s surface have been devised to perceptually measure the amount of visual distortions introduced by the watermarking algorithm over the surface of the model. Then, a second experiment (Experiment II) has been conducted in order to validate the proposed metrics with other watermarking algorithms.

This Chapter is organized as follows. Previous works related to perceptual image watermark insertion, to mesh simplification and perceptually-guided rendering are reviewed in Section 7.2. In Section 7.3 we describe the artifacts introduced by common 3D watermarking algorithms. Our experimental methodology to carry out subjective experiments on 3D model quality evaluation is described in Section 7.4. Subjective data analysis is performed in Section 7.5. Section 7.6 describes the proposed metric. Finally, results are presented and discussed in Section 7.7.

7.2 Related Work

The knowledge of the human visual system has been widely applied by perceptual image watermarking to obtain high quality watermarked images, i.e. watermarked images undistinguishable from the original ones. Our investigation concerns the extension of this idea to 3D watermarking. The goal is to develop a perceptual metric to estimate the perception of visual artifacts introduced by watermarking. The evaluation of the visual impairment introduced by a watermarking algorithm can be used to adjust the watermarking parameters in order to obtain a watermarked model that looks like the original one. Perceptual metrics are not limited to perceptual watermarking, but they have also been used in two other fields of Computer Graphics: mesh simplification and perceptually-guided rendering. The three issues related to our investigations, concerning perceptual image watermarking, mesh simplification and perceptually-guided rendering, will be discussed in the following.

7.2.1 Perceptual Image Watermarking

It is widely known among researchers working in Digital Watermarking that HVS characteristics have to be carefully considered in order to minimize the visual degradation introduced by the watermarking process while maximizing robustness [178, 179, 180]. Considering a noisy image, some aspects of human visual perception that anyone can easily experience are that i) disturbs in the uniform
regions of the image are more visible than those in textured regions, ii) noise is more easily perceived around edges and iii) the human eye is less sensitive to disturb in very dark and very bright regions. These basic mechanisms of the human visual perception can be mathematically modeled considering two main concepts: the contrast sensitivity function (CSF) and the contrast masking model. CSF is a measure of the responsiveness to contrast for different spatial frequencies. Typically, CSF models the capability of the human eye to perceive sinusoidal patterns on a uniform background. The contrast perception varies with the frequency of the sinusoidal pattern, with the orientation of the pattern, with the observer’s viewing angle and with the luminance of the background where the stimulus is presented. Many analytical expressions of CFS can be found in literature, one of the most used is the Barten’s model [181].

The masking effect concerns the visibility reduction of one image component due to the presence of other components. In other words, while CSF considers the visual perception of a sinusoidal pattern on an uniform background the visual masking model considers the perception of a sinusoidal pattern over spatially changing background. The non-uniform background may be modelled with another sinusoidal pattern with different properties. Some models of visual masking have been developed by Watson [182, 183] and by Legge and Foley [184].

Many methods have been proposed so far to exploit the models of the HVS to improve the effectiveness of existing watermarking systems [180, 24]. We can divide the approaches proposed so far into theoretical [24, 185, 25] and heuristic [186, 23]. Even if theoretically grounded approach to the problem would be clearly preferable, heuristic algorithms sometimes provide better results due to some problems with the HVS models currently in use [23, 187].

### 7.2.2 Mesh simplification

Mesh simplification regards the reduction of the number of vertices and triangles of a polygonal mesh while preserving its visual appearance. In general, the simplification process is driven by a similarity metric that measures the impact of the changes of the model after each simplification step. So, one of the most important characteristics of a simplification method is the error metric it uses. Two kinds of metrics are considered for simplification: geometric metrics and (perceptual) image-based metrics.
Geometry-based metrics

Metrics for simplification are commonly used for two distinct purposes: evaluating the quality of the final model and determining where and how to simplify the model. The most used global geometry-based metrics for off-line quality evaluation of 3D models are based on the Hausdorff distance.

The Hausdorff distance is one of the most well-known metrics for making geometric comparisons between two point sets. Assuming that the shortest distance between a point \( x \) and a set of points \( Y \) (e.g. the vertices of the 3D model) is the minimum Euclidean distance, i.e:

\[
d(x, Y) = \min_{y \in Y} d(x, y)
\]  
(7.1)

the asymmetric Hausdorff distance between two point sets can be defined as:

\[
\tilde{d}_\infty(X,Y) = \max_{x \in X} \min_{y \in Y} d(x, y)
\]  
(7.2)

Since \( \tilde{d}_\infty(.) \) is not symmetric, i.e. \( \tilde{d}_\infty(X,Y) \neq \tilde{d}_\infty(Y,X) \), this distance is not a metric in mathematical sense. To obtain symmetry it can be redefined as:

\[
d_\infty(X,Y) = \max \left( \tilde{d}_\infty(X,Y), \tilde{d}_\infty(Y,X) \right)
\]  
(7.3)

The (7.3) is usually referred as the maximum geometric error. This metric is not able to catch well geometric similarity since a single point of the set \( X \), or \( Y \), can determine the Hausdorff error. One possible alternative based on the average deviation that best measures geometric similarity is:

\[
\bar{d}_1(X,Y) = \frac{1}{A_X} \int_{x \in X} d(x, Y) dX
\]  
(7.4)

where \( A_X \) is the area of the surface \( X \). Even this metric is asymmetric. The symmetric version of this metric assumes the following form:

\[
d_1(X,Y) = \frac{A_X}{A_X + A_Y} \bar{d}_1(X,Y) + \frac{A_Y}{A_X + A_Y} \bar{d}_1(Y,X)
\]  
(7.5)

and it is usually referred to as mean geometric error. Two tools for geometric meshes comparison based on the maximum (7.3) and on the mean geometric error (7.5) are the Metro [188] and the Mesh [189] tool. Several researchers have proposed other geometry-based metrics to evaluate 3D model quality. Most of them are variations of the \( d_\infty(.) \) and \( d_1(.) \) metrics. In Section 7.7, we will analyze the performance of these two geometric metrics in the case of 3D watermarking quality evaluation.
7.2. Related Work

Image-based metrics

Image metrics are adopted in several graphics applications. In fact, since most computer graphics algorithms produce images, it makes sense to evaluate their results using image differences instead of metrics based on geometry. A lot of simple image metrics such as the Root Mean Square (RMS) and the Peak Signal Noise Ratio (PSNR) have been widely used in the past, but, such metrics are not able to measure the differences between two images as perceived by a human observer [190]. For example, Figure 7.1 shows that the values or RMS do not correlate with the perception of image distortions. For this reason, nowadays, most applications move to perceptual-based image metrics. Two of the most perceptually accurate metrics for comparing images are the Visual Difference Predictor by Daly [191] and the Sarnoff model developed by Lubin [192]. Both of these metrics include models of different stages of the human visual system, such as, opponent colors, orientation decomposition, contrast sensitivity and visual masking.

Concerning perceptually-based mesh simplification, Lindstrom and Turk [42] propose an image-driven approach for guiding the simplification process: the model to be simplified is rendered by considering several viewpoints and an image quality metric, based on a simplified version of the Sarnoff Model [192], is used to evaluate the perceptual impact of the simplification operation. More recently, Luebke et al. [44] developed a view-dependent simplification algorithm based on a simple model of CSF that takes into account texture and lighting effects. This method provides also an accurate modelling of the scale of visual changes by using parametric texture deviation to bound the size (represented as spatial frequencies) of features altered by the simplification. Other studies related to perceptual issues...
in mesh simplification have been conducted by Rogowitz and Rushmeier [193] and by Yixin Pan et al. [194]. In particular, Rogowitz and Rushmeier analyze the quality of simplified models perceived by human observers in different lighting conditions by showing to the observers still images and animations of the simplified objects. From the experiments they draw a lot of interesting conclusions. The most important one is that the perceived degradation of the still images cannot be adequate to evaluate the perceived degradation of the equivalent animated objects. This result suggests that an experimental methodology to evaluate the perceived alterations of 3D objects should rely on the interaction with the model.

### 7.2.3 Perceptually-Guided Rendering

The aim of perceptually-guided rendering is to accelerate photo-realistic rendering algorithms to avoiding computations for which the final result will be imperceptible.

One of the first works of this type has been done by Reddy [195]. Reddy analyzed the frequency content of 3D objects in several pre-rendered images and used these results to select the "best" version of the objects from a pool of models representing the same shape with different level of details in order to speed-up the visualization of a virtual environment. If the high-resolution version of the model differs only at frequencies beyond the modeled visual acuity or greatest perceptible spatial frequency, the system selects a low-resolution version of the model.

Other remarkable works in this field include Bolin and Meyer [196] that use a simplified Sarnoff Visual Discrimination Model [192] to speed-up rendering techniques based on sampling (e.g. Monte Carlo Ray Tracing), Myszkowski et al. [197] that incorporate spatio-temporal sensitivity in a variant of Daly Visual Difference Predictor [191] to create a perceptually based animation quality metric (AQM) to accelerate the generation of animation sequences and Ramasubramanian et al. [198] that use perceptual models to improve global illumination techniques used for realistic image synthesis.

Another excellent work related to study of human visual perception in rendering is the one by Ferwerda and Pattanaik [199]. In this work a sophisticated perceptual metric for the evaluation of how much a visual pattern, i.e. a texture, hides geometry artifacts is proposed. The visual masking effect caused by texturing is taken into account by analyzing the final rendered images.
7.2.4 Our Approach

Our goal is to develop a perceptual metric that measures the human perception of geometric artifacts introduced over a 3D surface by watermarking. Two approaches to develop a perceptual metric for 3D watermarking are possible. The first one follows the (perceptual) image-based approach for simplification seen before [42, 44]. Instead of driving the simplification process, the perceptual image metric can be used to evaluate the visual effects of the watermark insertion by computing the perceptual differences between several images rendered from the original and the watermarked model. This way of proceed presents two advantages. First, since it is rendering-dependent, complex lighting and texturing effects can be taken into account in a natural way. The second advantage is that all possible kinds of visual artifacts can be evaluated with the same approach. The main disadvantage is that the rendering conditions must be known in advance. The other possible approach is to evaluate in which way the human visual system perceives geometric distortions on the model surface and build an ad-hoc perceptual metric for geometric artifacts. Moreover, this approach is more interesting from a research viewpoint, since no similar studies have been conducted at now. The potential field of application of this kind of studies is not limited to improve imperceptibility of 3D watermarking algorithms, but, other Computer Graphics applications can benefit from them. For these reasons we decided to follow the second approach, i.e. to work directly on the geometry of the 3D model.

7.3 3D watermarking algorithms and artifacts

As previously stated in Chapter 3, digital watermarking algorithms can be classified in algorithms that work in the asset domain, in the hybrid domain and in the transformed domain. Here, for each class of algorithms, we describe the geometric artifacts that they introduce in the watermarked model.

First of all, we consider the algorithms working in the asset domain. The algorithms based on topological embedding [51, 56] produce small geometric distortions that can be described by the addition of a small amount of noise to the position of the mesh vertices. When only the connectivity of the mesh is used to embed the watermark, such as in the TSPS and in the MDP algorithms [51], the amount of introduced distortions is imperceptible, since topology changes usually do not produce noticeable visual effects. Concerning geometric features embedding we have to distinguish between those algorithms that embed the watermark
by using vertices position and those algorithms that are based on shape-related features, such as vertex normals. Changes in the vertices position produce the same effect of topology-driven embedding, i.e. a “noisy” watermarked surface, but, in this case the amount of distortion may be considerably high, due to the vertices displacements needed to embed the watermark. For example the Vertex Flood Algorithm [55], may introduce moderate-to-strong distortions depending on the embedding parameters. In the same manner, the method of Harte and Bors [57] may produce perceptible distortions depending on how many vertices are watermarked and on the dimension of the bounding volume used. Shape-related algorithms, like the Normal Bin Encoding (NBE) [59] and the method by Wagner [60], instead, introduce artifacts that look very different from the noisy effect of the other techniques. This kind of surface alterations produce soft changes in the shape of the model thus resulting in artifacts difficult to perceive.

The algorithms that work in the hybrid domain are able to spread the distortions smoothly over the whole surface of the model by introducing the watermark in the low resolution of the model. Typically, this permits to reduce the previously described ”noise” effect. In fact, the amount of distortion produced by our algorithm, that works in the hybrid domain, heavily depends on the level of resolution used to embed the watermark. In particular, for a fixed watermark strength, the higher is the level of resolution used and the higher is the amount of visual impairment of the watermark model. In the same way the algorithm by Kanai et al. [52], that it is also based on wavelet decomposition, may introduce geometric artifacts since all the levels of resolution are used to embed the watermark. Kanai proposes a geometric tolerance threshold to limit the introduction of these visual artifacts.

Concerning the transformed domain, mesh spectral methods [65, 66, 67] cause a vertices perturbation due to the modifications of the mesh spectral coefficients, thus resulting in a moderate “noisy” watermarked surface. Ohbuchi [65] suggests to reduced this effect by watermarking those mesh spectral coefficients that are related to low frequencies content of the model.

Concluding, we can say that, in general, 3D watermarking algorithms produce “noisy” surfaces. The characteristics of the noise depend on the specific algorithms; noise can have different granularity and size, and may be uniform or not over the model surface. The watermarking techniques that do not introduce perceptible artifacts are typically those techniques that have relaxed robustness requirements. In our subjective experiments, that will be described in the next
Section, we have considered the following four different watermarking algorithms: the Vertex Flood Algorithm (VFA) [55], the Normal Bin Encoding (NBE) [59], the method by Kanai et al. [52], and our algorithm. The algorithm by Kanai et al. and the our one will be indicated in the following using the acronyms of the authors, i.e. KDK and UCB respectively. Figure 7.2 shows the artifacts introduced by these watermarking algorithms.

7.4 Experimental Method

A set of standards and grading techniques to evaluate quality of video and multimedia content have been defined by ITU-R [200] and ITU-T [201]. However, there are no prescribed standards for the evaluation of 3D objects with impairments. In this Chapter, we propose a method for subjective evaluation of 3D watermarked objects. This experimental methodology attempts to make subjective evaluations in this field more reliable, comparable and standardized.

The starting point for the design of a subjective experiment for the evaluation of the quality of 3D objects is to define how to render the object under examination. By specifying appropriate rendering conditions, we aim at putting the human observer in favorable conditions to make a fair judgment on the three-dimensional object. The rendering conditions should not bias the human perception of the 3D model by privileging, for example, one view of the 3D object rather than another one.

7.4.1 Rendering Conditions

The rendering of a three-dimensional model is accomplished via a combination of techniques such as associating a material to each surface of the model, applying various kinds of light sources, choosing a lighting model, adding textures and so on. In our investigations we assumed that the rendering conditions have to be as simple as possible, because very few works have dealt with psychophysical tests of perceived 3D object quality as reported in Section 7.2 and no experimental data are available. Moreover, too many or complicated rendering effects would involve many and mutually linked aspects of spatial vision that have to be avoided to obtain more reliable results. Such results can be further extended by taking into account more aspects of visualization techniques, such as the role of photorealism in the perception of impairments. In fact, by keeping plain but effective rendering conditions, we do not influence or bias the human perception and, then, the
Chapter 7. 3D Watermarking Quality Assessment

Figure 7.2: Geometric defects introduced by 3D watermarking algorithms.
subjects’ evaluation. The rendering conditions that we have chosen are described below.

• **Light sources.** Humans can see an object because photons are emitted from the surface of the object and reach the eyes of the viewer. These photons may come from light sources or from other objects. As previously stated, the three common types of light sources are the directional, the point, and the spotlight. Point and spot light sources are also called *positional lights* because they are characterized by a location in space. Spotlights are not suitable for our purposes since this kind of light source could privilege some parts of the model better illuminated with respect to others. Multiple lights can cause effects that may confuse the human observer and provide contradictory results more complex to evaluate [202]. Additionally, the HVS tends to assume that the scene is illuminated by a single light source and that light illuminating the scene is coming from above. For all of these reasons, in our experiments, each model is illuminated with one white point light source located in a top corner of the Object Bounding Box (OBB) of the 3D object. Achromatic light is used in order to preserve the colors of the material.

• **Lighting and shading.** A good choice of the lighting model and of the shading method succeeds in effectively communicating to a human observer the 3D shape and the fine geometric details of a 3D object. The influence of the lighting model is very important since it may affect the perceived quality of the 3D model considerably. Ideal lighting and shading conditions are very hard to find and some methods have been proposed to optimize rendering conditions in order to improve the perceptual quality of the rendered model [202]. To narrow the scope, we use a simple local illumination lighting model where only the diffusive component of the reflected light is considered. In fact, the diffusive component is well-connected to physical reality since it is based on Lambert’s Law (6.2) which states that for surfaces that are ideal diffusive (totally matte, without shininess) the reflected light is determined by the cosine between the surface normal and the light vector. For this reason the diffuse component does not depend on viewer’s position making this model suitable to unbias the human perception of the 3D object under examination. The specular component of the reflected light is not considered even if it would improve the photorealism of the objects. In fact, while the diffuse component catches the behavior of matte surfaces, the specular component models the shininess of the surfaces. The highlights created by the spec-
ular component help the viewer to better perceive the surface’s curvature. Moreover, other more complex photorealistic issues such as self-shadowing are not introduced in our model, as they would unnecessarily complicate the experimental method and introduce too many variables to evaluate during results analysis. So, the implemented lighting model is:

\[ I_r = I_{\text{amb}}K_a + I_iK_d \min(0, \ N \cdot L) \] (7.6)

where \( \vec{N} \) is the surface normal at the considered point, \( \vec{L} \) is the incident light direction vector and the constant \( K_a \) and \( K_d \) depend on the material properties. About shading methods that deal with triangular meshes, we can choose among flat, Gouraud and Phong shading (see Section 6.2.5). Flat shading is not suitable for our purposes since it produces the well-known unnatural faceting effect. Since Gouraud and Phong shading produce almost the same visual effects if the models resolution, i.e. the number of triangles of the model, is high, we decide to use the Gouraud shading that it is more common and less computationally expensive than the Phong method. Finally, we have decided to show the model on a non-uniform background since an uniform background highlights too much the countour edges of 3D objects.

- **Texturing.** We want to evaluate the perception of artifacts on the surface of the 3D objects, hence textures or other effects are avoided as they usually produce a masking effect on the perceived geometry [199]. In fact, image texture mapping, bump mapping, and other kind of texturing may hide the watermark artifacts. This is partially due to the visual masking perceptual effect, in which frequency content in certain channels suppresses the perceptibly of other frequencies in that channel [199]. We do not account for visual masking, leaving that as an important and interesting area for future researches.

- **Material properties.** The color of a surface is determined by the parameters of the light source that illuminate the surface, by the lighting model used and by the properties of the surface’s material. We consider only gray, stone-like objects. This choice is made for different reasons: first, if all models are seen as “statues” the subjects perceive all the models in the same manner and enough naturally; second, in this way we avoid the memory color phenomenon experimented in psychology studies [203]. This phenomenon
regards the fact that an object’s characteristic color influences the human perception of that object’s color, e.g. shape such as heart and apple are characteristically red. A particular choice of a specific color for all the models could have mislead the perceived quality of the object and introducing too many colors for different objects would have made the experimental method less general by introducing too many degrees of freedom.

- **Screen and Models Resolution.** The monitor resolution used in the experiments is $1280 \times 600$ and each model is displayed in a window of $600 \times 600$ pixels. The model occupies around 80% of the window and the resolution of the models used in the experiments ranges between 50,000 and 100,000 triangles. This screen resolution and the level of details of the models allow a good visualization of the model details, and hence of the model distortions. In particular the blurring effect of the Gouraud shading interpolation is negligible. Such blurring effect increases when the subject observes the model closely. Moreover, the complexity of the models used allows us to render them quickly (using a powerful graphics accelerator board) making fluid user-interaction possible. A minimum frame rate of 50 fps for each of the visualized models is guaranteed.

- **Interaction.** One essential feature of an interactive application is that objects are observed in motion. In our experimental method, we decide to allow the subject to interact with the model by rotation and zoom operations. The user interacts with the model by using a mouse. Three-dimensional interaction is achieved by *ARCBALL rotation control* [204]. The motion of the 3D object is then interactively driven by the subject and not pre-registered like in other subjective experiments in literature [193, 194]. This avoids to detect less details in frames that pass quickly. It has to be mentioned that in previous work [193, 205], 2D images of 3D objects have been used for subjective experiments. The problem is that different static views of 3D objects can have significantly different perceived quality depending on the direction of illumination. Subjective experiments that were conducted to address this question suggest that judgments on still images do not provide a good predictor of 3D model quality [193]. Rogowitz’s work confirmed that the use of still images produce results that are view-dependent and not correlate well with the real perception of the 3D object.
7.4.2 Experimental Procedure

Our test subjects were drawn from a pool of students from the École Polytechnique Fédérale de Lausanne (EPFL). They were tested one at a time and not specifically tested for visual acuity or color blindness. The 3D models were displayed on a 17-inch LCD monitor, with participants sitting approximately 0.4 meters from the display. The experiment followed a five-stage procedure [206]. The stages were: (1) oral instructions, (2) training, (3) practice trials, (4) experimental trials, (5) interview.

**Oral Instructions**

Test subjects have to be told how to perform the experiment. Prior to each experiment, the instructions, also known as *experiment scripts*, are elaborated to help the experimenter to define the task. The script contains details of what the experimenter should do at each step of the experiment. More importantly, it contains oral instructions that should be given to the subject to make sure he/she understands the task to be performed. An introductory explanation about what 3D models are and what watermarking is is given. Different sections of the instructions actually apply to all the various stages of the experiment. However, the most important part of the instructions comes before the training stage. After the subject is properly seated and made comfortable, the main task is explained. The instructions for both experiments can be found in Appendix A.

**Training**

In each experiment the subject is asked to perform a task which consists of entering a judgment about an impairment detected in the 3D object. In order to complete this task subjects need to have an idea of how the original 3D objects with no impairments look. Therefore, a training session is included in the procedure which consists of displaying the four original models used to embed the watermark. In this phase, only the experimenter interacts with the graphical interface, the subject is asked to look carefully to the models displayed on the screen. In the next phase, a set of 3D models with the typical distortions introduced by watermarking is shown.

Another set of 3D objects is required to set a point on the scale of judgements. The end of the scale is set by 3D objects with the strongest defect (in this case the perceptually strongest watermarking). A total of 12 and 16 models were shown.
as worst examples respectively for the Experiment I and II. Therefore, the test subjects are instructed to pick the strongest stimulus in the training set and assign that stimulus a number from the upper end of the scale. In our experiments, the subject is asked to assign 10 to the worst example (on a discrete scale ranging from 0, which means that no distortions are perceived, to 10). However, due to visual masking and to the variety of the originals, it is not possible to anticipate which defects the subjects will consider worst or strongest. Therefore, the subjects are asked to record values greater than the nominal upper value if 3D objects are observed to exceed the expectations established by the training phase. For example, if a test subject perceives a defect twice as bad as the worst in the training set, he/she is asked to give it a value of 20. Finally, in the last phase of the training stage, subjects are told how to use the graphical interface to interact with the 3D models.

Practice Trials

In the practice trial stage, subjects are asked to make judgments for the first time. Because of the initial erratic responses of the subjects, ITU Recommendation [201] suggests to throw away the first five to ten trials of an experiment. In our case, instead of discarding the first trials, we included practice trials. This also gives other benefits. It exposes the test subject to 3D models throughout the impairment stage. It gives the test subject a chance to try the data entry and above all the chance to get familiar with the graphical interface for the virtual interaction with the 3D object (rotation and zooming). The number of practice trials is 6. The
subject has to perform 3 tasks at most. The first one is to detect the distortion
and he/she has to answer to the question *did you notice any distortion?*. In case
of positive answer, the subject has to give a score to indicate *how much the distortions
are evident*. The subjects have 30 seconds at their disposal to interact with the
model and to make their judgement. Then, they have to input the score in a
dialog box. Figure 7.3 shows the interface for the subjective tests. On the left a
time bar advices the user of the remaining interaction time. The box indicates the
progression of the test by showing the number of the model under examination.
The model is displayed in the center of the screen. Finally, the third question is
*where* he/she noticed the distortion on the 3D model. To answer to this question
he/she has to indicate, by selection, the part of the model where the distortions
are the most evident (figure 7.4).

![Figure 7.4: The subject during the selection of the part where the distortions are more
evident.](image)

**Experimental Trials**

The subjective data is gathered during the experimental trials. In this stage, a
complete set of 3D objects is presented in random order. To be more specific, for
each experiment, several random-ordered lists of test watermarked 3D objects are
generated. In this way the results of the test are made independent of the order
in which the models are presented. All the test subjects see all the 3D objects.
The number of test 3D objects is limited so that the whole experiment lasts no
more than one hour. This limit translates to 40 models in Experiment I and 48
in Experiment II.
7.4. Experimental Method

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak luminance</td>
<td>( \leq 0.04 )</td>
</tr>
<tr>
<td>Maximum observation angle</td>
<td>10 degrees</td>
</tr>
<tr>
<td>Monitor resolution</td>
<td>1280 \times 600</td>
</tr>
<tr>
<td>Interaction window resolution</td>
<td>600 \times 600</td>
</tr>
<tr>
<td>Viewing Distance</td>
<td>35 – 45 cm</td>
</tr>
<tr>
<td>Monitor Size</td>
<td>17”</td>
</tr>
</tbody>
</table>

Table 7.1: Viewing conditions during subjective test.

**Interview**

After the trials are complete, the test subjects are asked a few questions before they leave. The kind of questions depends on the experiment, mainly test subjects are asked for qualitative descriptions of the impairment. The questions asked in our case are:

1. Did you experience any problem correlated to a specific model in identifying the distortion?
2. How would you describe the distortions that you saw?
3. Have you general comments or remarks about the test?

These questions gather interesting information, for example, they are useful for categorizing the distortion features seen in each experiment and to help in the design of next experiments.

**7.4.3 Experiment I**

The goal of the first experiment was to make an initial study about the perception of artifacts caused by watermarking on 3D models and to find suitable metrics to measure the perceptual severity of such artifacts. The output of the experiment was a collection of subjective evaluations of a set of watermarked 3D objects. The watermarking artifacts, varying in strength and resolution, were generated using our watermarking algorithm. This subjective data set allowed us to confirm some basic findings such as commonly used geometric-based metric (already mentioned in Section 7.2) are not a good measure for subjective quality evaluation of watermarked 3D objects. Additionally, we were interested in how the test subjects...
would describe the watermarking defects that were produced for this experiment. In particular, the appearance related questions included in the interview provided us some directions to design a perceptually driven objective metric. The experiment have been performed by 11 subjects. The methodology for the experiment has been just described in Section 7.4.2.

### 7.4.4 Generation of Stimuli

The test models for this experiment were generated by applying our algorithm to the previously described “Bunny”, “Feline”, “Horse” and “Venus” models. Figure 7.5 shows these models rendered with the rendering conditions used in the experiments. These models are suitable for perceptual studies thanks to the wide range of characteristics presented by their surface. For example, the Bunny model surface is full of bumps, all parts of the Horse model are smooth, Feline model presents a wide range of characteristics such as parts with high curvature, parts with low curvature, moderate bumps, smoothed parts and several protrusions, and the Venus model has the same range of characteristics of the Feline but without consistent protrusions.

The watermark was uniformly distributed over all the surface of the 3D objects. As previously stated, the amount of distortions introduced by the watermarking varied according to two parameters: i) the resolution level $l$ that hosts the watermark and ii) the coefficient $\gamma$ determining the strength of the watermark. Table 7.2 shows the values of the watermarking parameters used for the experiment. Three amount of watermarking strength (low, medium and high) and three levels of resolution (low, medium and high) were applied to each model. In addition to the watermarked models, the 4 original models were included in the complete model set. In fact, the original may present impairments unrelated to the watermarking ones deliberately inserted into the test models. To separate the effects of the deliberate and the pre-existing defects, the originals had to be inserted. A total of 40 (4 originals $\times$ 3 watermarking strength $\times$ 3 resolution level + 4 originals) test models were used in the experiment.

### 7.4.5 Experiment II

In Experiment I, test subjects evaluated differently watermarked models ranging from severe down to weak visual impairments. Those different distortions’ strengths were generated using a specific watermarking algorithm, i.e. the UCB algorithm. With this experiment we wanted to test by means of subjective vali-
Figure 7.5: Rendering Conditions.
Table 7.2: Experiment I: Watermarking parameters values used for each model.

<table>
<thead>
<tr>
<th>Level of Resolution ((l))</th>
<th>Corresponding Value</th>
<th>Watermarking Power ((\gamma))</th>
<th>Corresponding Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>4</td>
<td>Low</td>
<td>0.0003</td>
</tr>
<tr>
<td>Medium</td>
<td>3</td>
<td>Medium</td>
<td>0.0015</td>
</tr>
<tr>
<td>High</td>
<td>2</td>
<td>High</td>
<td>0.003</td>
</tr>
</tbody>
</table>

dation the perceptually-based objective metrics for the quality assessment of 3D watermarking we obtained from the subjective data of the Experiment I. Therefore, we chose three different watermarking algorithms: NBE, VFA and KDK. Technically, the defects inserted are slightly different from the ones studied in the Experiment I (see Figure 7.2). In fact, while the UCB algorithm produces an uniform kind of noise that can be described as an increase of the roughness of the watermarked surface, VFA produces a kind of noise that looks like marble streak, depending on the viewpoint. The artifacts of the KDK algorithm are the same of the UCB algorithm but due to the geometric tolerance introduced by Kanai to limit the visual impact of the watermark the final visual effects of such distortions is not uniformly distributed over the model's surface. Concerning NBE, the visual aspect of its artifacts is very different from those UCB, VFA and KDK and more difficult to perceive. The methodology for this experiment is practically the same of the Experiment I. The only difference is that no location information was gathered since the metric developed on the basis of the data collected in the Experiment I does not take into account the location information.

7.4.6 Generation of Stimuli

The test models for this experiment were generated using the same four original models of the Experiment I. As just stated, the three watermarking algorithms used to generate the watermarked models are the VFA, the NBE and the KDK algorithms. Each watermarking algorithm is characterized by its own embedding parameters that are qualitatively and quantitatively different. For an exact description of each parameters we refer to the literature. The watermarking parameters of the VFA are the number of clusters used to embed the bits (one bit for each cluster) and the maximum allowable distance (\(D_{\text{MAX}}\)) from the starting application points. NBE is characterized by the feature type used, by the number.
of bins ($N_B$), and by the search range ($\Delta R$) and the number of iterations ($n_I$) of the optimization process. KDK parameters are the selection threshold ($\delta_1$), the geometric tolerance threshold ($\delta_2$) and the least significant decimal digits used to embed the watermark ($d_w$); if $d_w = 2$ this means that the second least significant decimal digits of the wavelet coefficients is modified to embed the watermark, $d_w = 3$ indicates the third least significant decimal digits, and so on. The watermarking parameters for the three algorithms are reported in Table 7.3. In our test set, we have tried to range from severely to weakly watermarked model as in Experiment I. The 11 level of impairment are also reported in Table 7.3. A total of 48 (4 models $\times$ 11 watermarking settings + 4 originals) test models were used in this experiment.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Watermarking Parameters</th>
<th>Impairment</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDK1</td>
<td>$\delta_1 = 0.001$, $\delta_2 = 0.005$, $d_w = 2$</td>
<td>medium-strong</td>
</tr>
<tr>
<td>KDK2</td>
<td>$\delta_1 = 0.001$, $\delta_2 = 0.008$, $d_w = 3$</td>
<td>medium</td>
</tr>
<tr>
<td>KDK3</td>
<td>$\delta_1 = 0.001$, $\delta_2 = 0.02$, $d_w = 3$</td>
<td>medium-strong</td>
</tr>
<tr>
<td>NBE1a</td>
<td>Feature Type I, $N_B = 80$, $\Delta R = 0.0015$, $n_I = 3$</td>
<td>medium</td>
</tr>
<tr>
<td>NBE1b</td>
<td>Feature Type I, $N_B = 80$, $\Delta R = 0.0008$, $n_I = 3$</td>
<td>weak-medium</td>
</tr>
<tr>
<td>NBE2a</td>
<td>Feature Type II, $N_B = 80$, $\Delta R = 0.0001$, $n_I = 1$</td>
<td>weak</td>
</tr>
<tr>
<td>NBE2b</td>
<td>Feature Type II, $N_B = 20$, $\Delta R = 0.0004$, $n_I = 1$</td>
<td>medium</td>
</tr>
<tr>
<td>VFA1</td>
<td>600 clusters, $D_{MAX} = 1.8$</td>
<td>strong</td>
</tr>
<tr>
<td>VFA2</td>
<td>960 clusters, $D_{MAX} = 1.8$</td>
<td>medium</td>
</tr>
<tr>
<td>VFA3</td>
<td>1320 clusters, $D_{MAX} = 1.8$</td>
<td>weak</td>
</tr>
<tr>
<td>VFA4</td>
<td>200 clusters, $D_{MAX} = 0.6$</td>
<td>medium</td>
</tr>
</tbody>
</table>

Table 7.3: Experiment II: Watermarking parameters values.

### 7.5 Data Analysis

During the experiments, if a subject notices surface defects, he/she is supposed to enter a value proportional to the amount of distortions perceived on the model surface. In the following we refer to these values as **subjective scores**. The subjective scores have to be condensed by statistical techniques used in standard methods [200, 206] to yield results which summarize the performance of the system under test. The averaged score values, called **Mean Opinion Score** or **MOS**, are considered as the amount of distortions that anyone can perceive on a particu-
lar watermarked 3D object. However, impairment is measured according to some scale, and such scale may vary from person to person. In this section, we report the methods used for matching the scales of our test subjects. Then, we describe the methods used to combine the subjective data and evaluate the precision of the estimates. The subjects are screened and outliers are discarded. Finally, the results are checked for error due to the methodology of the experiment.

7.5.1 Normalization

As a measurement device, a test subject may be susceptible to both systematic and random errors. The purpose of the normalizing procedure is to compensate for any systematic error. The procedure must be applied to the measurement gathered from each test subject prior to the combination of measurements across subjects. The unscaled annoyance value, \( m_{ij} \) obtained from subject \( i \) after viewing the test object \( j \) can be represented by the following model:

\[
m_{ij} = g_ia_j + b_i + n_{ij}
\]

(7.7)

where let \( a_j \) be the true annoyance value for test object \( j \) in the absence of any error, \( g_j \) is a gain factor, \( b_i \) is an offset, and \( n_{ij} \) is generally assumed to be a sample from a zero-mean, white Gaussian noise.

In this model, the gain and offset could vary from subject to subject. If the variations are large across the subjects or the number of subjects is small, a normalizing procedure can be used to reduce the gain and the offset variations among test subjects. In order to check if the offset and gain factors vary significantly from subject to subject, a two-way analysis of variance (ANOVA) approach was used [207]. A two-way ANOVA divides the total variations of a table of data into two parts: the variation that is attributed to the columns of the data table, and the variation that is attributed to the rows of the data table. Specifically, the F-test can be used to determine the likelihood that the means of the columns, or the means of the rows, are different. The experimental data, \( m_{ij} \) is arranged so that each row represents data for one test object and each column represents all the data for one test subject. The analysis assumes that the data can be modeled as [207]:

\[
m_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij}
\]

(7.8)

where \( \mu \) is the overall mean, \( \alpha_i \) stands for the subject effects, \( \beta_j \) stands for the model effect, and \( \epsilon_{ij} \) are the experiment errors. In Table 7.4 and Table 7.5 the ANOVA results are indicated for objects and subjects from Experiment I and
Experiment II respectively. The *F*-values for both subjects (*F* = 12.22 and *F* = 18.3 for Experiment I and Experiment II respectively) and objects (*F* = 23.01 and *F* = 21.16) were large. The right-most column of the table contains the probabilities that the subject effect and the object effect are constant, i.e. that there are no differences among the subjects or among the test objects. It is important to underline that the object effect is not expected to be constant since it depends on the particular 3D object and the 3D objects were deliberately varied.

<table>
<thead>
<tr>
<th>Source</th>
<th>SoS</th>
<th>df</th>
<th>MS</th>
<th>F</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
<td>372.22</td>
<td>10</td>
<td>37.24</td>
<td>12.22</td>
<td>0</td>
</tr>
<tr>
<td>Models</td>
<td>2735.54</td>
<td>39</td>
<td>70.14</td>
<td>23.01</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>1188.83</td>
<td>390</td>
<td>3.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>4296.82</td>
<td>439</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.4: Experiment I: ANOVA analysis results.

<table>
<thead>
<tr>
<th>Source</th>
<th>SoS</th>
<th>df</th>
<th>MS</th>
<th>F</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
<td>382.84</td>
<td>10</td>
<td>38.28</td>
<td>18.3</td>
<td>0</td>
</tr>
<tr>
<td>Models</td>
<td>2080.63</td>
<td>47</td>
<td>44.26</td>
<td>21.16</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>983.34</td>
<td>470</td>
<td>2.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3446.82</td>
<td>527</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: Experiment II: ANOVA analysis results.

<table>
<thead>
<tr>
<th>Source</th>
<th>SoS</th>
<th>df</th>
<th>MS</th>
<th>F</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subjects</td>
<td>21.79</td>
<td>10</td>
<td>2.18</td>
<td>8.92</td>
<td>0</td>
</tr>
<tr>
<td>Sequences</td>
<td>151.454</td>
<td>39</td>
<td>3.88</td>
<td>15.89</td>
<td>0</td>
</tr>
<tr>
<td>Error</td>
<td>95.33</td>
<td>390</td>
<td>0.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>268.58</td>
<td>439</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.6: Experiment I: ANOVA analysis results for ln(*m_{ij} + 1*).

For the F-values in Tables 7.4 and 7.5, the probabilities were zero. This means that there is a significant variation in the subjective value means from subject to subject. To check if the variation is caused by variations in the gain factor *g_i*, an
ANOVA is also informed for the natural logarithm of \( m_{ij} \). In fact, by taking the logarithm of the Equation 7.7 we obtain:

\[
\ln(m_{ij}) = \ln(g_i a_j + b_i + n_{ij}) \approx \ln(g_i) + \ln(a_j) + b_i/g_i a_j + n_{ij}/g_i a_j
\]

In this equation \( \alpha_i = \ln(g_i) \), \( \mu + \beta_j = \ln(a_j) \) and \( \epsilon_{ij} \approx b_i/g_i a_j + n_{ij}/g_i a_j \) (see equation 7.8). \( \epsilon_{ij} \) is no longer independent of the other factors, however if \( b_i \) and \( n_{ij} \) are small, this will not matter much in the analysis. Table 7.6 contains the ANOVA results for \( \ln(m_{ij}+1) \). Here, we decide to use the \( \ln(m_{ij}+1) \) instead of the \( \ln(m_{ij}) \) to avoid numerical problems to the presence of zero scores. The F-values are large and the probabilities near zero. This means that there were significant subject-to-subject variations in the gain factors and then some form of subject-to-subject correction was required. Several methods for estimating the offsets and the gain are possible. The probability of the null hypothesis (no variation in the subject means) for each correction method and the ANOVA results are summarized in Table 7.8. The measurements are adjusted prior to combination in the following way:

\[
\hat{m}_{ij} = \frac{1}{\hat{g}_i}(m_{ij} - \hat{b}_i)
\]

where \( \hat{g}_i \) is the corrected gain, \( \hat{b}_i \) is the corrected offset and \( \hat{m}_{ij} \) is the normalized score.

In the first correction method the offsets are estimated using the mean of all measurements made by each subject:

\[
\hat{b}_i = \frac{1}{J} \sum_{j=1}^J m_{ij} - \mu
\]

Since the mean is not a robust estimator of the center of a distribution the median is also tried as an estimate of the offset.

\[
\hat{b}_i = \text{median} \{m_{ij}, \forall j \in J\} - \mu
\]
where $J$ is the set of test models. The results are shown in Table 7.8. The mean estimate removes the subject to subject variations for both Experiments.

To adjust the gain as well, two gain estimation methods have been considered. The first gain estimation evaluates the measurements in terms of the experiment instructions and corrects the gain if the instructions were not followed exactly. In fact, the test subjects are told to assign a value of 10 to the worst of the test models seen during the training session. The corrected gain is set to make this true:

$$\hat{g}_i = \frac{1}{K} \max_{j \in J}(m_{ij})$$

(7.13)

where $K$ is equal to the upper end of the scale (10) in our testing procedure. The second method for correcting the gain variation relies on a statistical estimate of each test subject’s range. The standard deviation of the values is used to estimate the range. In this case, the gain factor becomes:

$$\hat{g}_i = \frac{4\delta_i}{N}$$

(7.14)

where $\delta_i$ is the standard deviation of all the values recorded by the $i$-th subject. The results are summarized in Table 7.8. The gain correction (7.14) combined to the mean offset correction provides the best results. Hence, after the normalization the collected data depend on the model but do not depend on the subject. This fact indicates that the experiment is well-designed, i.e. the experimental methodology is not affected by any systematic error.

<table>
<thead>
<tr>
<th></th>
<th>Experiment I</th>
<th></th>
<th>Experiment II</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Correction</strong></td>
<td><strong>Offset</strong></td>
<td><strong>Gain</strong></td>
<td><strong>Offset</strong></td>
<td><strong>Gain</strong></td>
</tr>
<tr>
<td>None</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>1</td>
<td>0.5557</td>
<td>1</td>
<td>0.6903</td>
</tr>
<tr>
<td>Median</td>
<td>0.0006</td>
<td>0.0001</td>
<td>0.0006</td>
<td>0</td>
</tr>
<tr>
<td>Mean + max</td>
<td>0.998</td>
<td>0.7962</td>
<td>1</td>
<td>0.8286</td>
</tr>
<tr>
<td>Mean + std</td>
<td>1</td>
<td>0.9996</td>
<td>1</td>
<td>0.999</td>
</tr>
</tbody>
</table>

Table 7.8: ANOVA analysis results after normalization (F-Test values for subject Offset and Gain dependency). Probabilities near one means that there is no difference across subjects.
Figure 7.6: Subject data correction results for Experiment I: (a) the mean scores values and (b) the confidence intervals.

Figure 7.7: Subject data correction results for Experiment II: (a) the mean scores values and (b) the confidence intervals.
7.5.2 Overall Scores and Screening

After the normalization process, the subjective scores values are combined into a single overall score for each test model using the sample mean. The subjective Mean Opinion Score (MOS) is given by:

\[ \mu_j = \frac{1}{N_j} \sum_{i=1}^{N} m_{ij} \] (7.15)

where \( N_j \) is the number of test subjects in the experiment. This measure represents the subjective scores of the non-detected and detected defects. A score of zero is assigned to the non-detected defects. Confidence intervals for the mean subjective scores were calculating using Student’s \( T \)-distribution. The \( T \)-distribution is appropriate when only a small number of sample is available [207]. The sample standard deviation \( s_j \) was calculated for each sequence \( j \) using \( \mu_j \) and the limits were calculated by:

\[ l_j = t_{(0.05,N_j)} s_j \] (7.16)

where \( t_{(0.05,N_j)} \) is the \( t \)-value associated with a probability of 0.95 and \( N_j \) degrees of freedom. As the number of observation \( N_j \) increases, the confidence interval decreases. The final result is \( j \) Mean Opinion Score values with associated 95% confidence interval, \( \mu_j \pm l_j \). The MOS values and sample standard deviations were first used in the subject screening procedure. In the screening procedure used (Annex 2 of ITU BT.500 Recommendation [200]) an expected range of values is calculated for each model. A subject is not rejected for always being above the expected range or always being below the expected range but for being erratic on both sides of the expected range. This procedure is appropriate to reduce the variability of the data in small sample sets. If necessary, after the screening procedures, the MOS and their relative confidence intervals were recalculated without the data from rejected test subject. After the normalization and screening, for the Experiment I the data of one subject were discarded, and for the Experiment II the data of three subjects were rejected.

7.5.3 Data Evaluation

The subjective data after normalization, averaging and screening procedures were used to test the proposed objective metric of the watermarked 3D object defects. In this section we check the validity of the obtained data and evaluate whether the methodology can be improved. Figures 7.6 and 7.7 show the overall data
spread for Experiment I and II. Note that the data spread was good for all these experiments. For Experiment I the MOS values ranged from 0.36 to 10.0 before the normalization and screening and ranged from 0.45 to 8.76 after (Figure 7.6 (a)). For Experiment II the ranges before and after are (0.45, 9.17) and (0.64, 9.44) respectively (Figure 7.7 (a)). The large number of data points (test 3D models) compensated for the lack of precision of individual points.

In summary, the experiments provided good data for most of the test models. The confidence intervals were reduced after the normalization and screening procedure (figures 7.6 (b) and 7.7 (b)). The confidence intervals were large for the test models that were hard to examine. This situation could be improved in future experiments by ensuring that the weakly watermarked models are closer to perceptual threshold or using many more test subjects. In conclusion, the data were good for the overall fits.

7.6 Proposed Perceptual Metric

Perceptual metrics that compute predictions of human visual task performance from input images, are usually based on a vision model. For any application in which a vision model produces reliable performance predictions, its use is almost always preferable to psychophysical data collection. One reason for this preference is that running a model generally costs much less than running a psychophysical experiment to generate the same system performance information, especially when the system evaluation in question is still in the design phase as in our case. There are two approaches [208] to model psychophysical quantities: performance modeling and mechanistic modeling. Although the distinction is more a continuum than a strict dichotomy, the performance models tend to treat the entire visual system as a "black box" for which input/output functions need to be specified. For the mechanistic model, physiological and psychophysical data are used to open the black box. As a result, input/output functions are needed not only for the system as a whole but for a number of component mechanisms within. These components of the model have the same functional response as physiological mechanisms of different stages of the HVS. It is important to underline that, at this time, it does not appear feasible to build a sophisticated model of visual perception of geometric artifacts since such models could become too complex to be handled in practice. In fact, one essential feature of any interactive application is that 3D objects are observed in motion, so the classical visual models used for still im-
ages should be integrated with other perceptual models that take into account the behavior of the human perceptions in a dynamic scene. Additionally, this model should take into account a lot of parameters that depend on the rendering, such as the lighting model, the texturing, and so on. For all of these reasons we opt for the “black box” approach. In particular, we use an objective metric based on surface roughness estimation combined with a standard psychometric function to model the black box. The purpose of a psychometric curve is to associate the values given by the objective metric to the subjective score values provided by the subjects. In this way a match between the human perception of geometric defects and the values provided by the objective metric is established obtaining a perceptual metric. Three kinds of psychometric functions are in commonly used [209]: the cumulative Gaussian distribution, the logistic psychometric function, and the Weibull psychometric function. In particular, we use a Gaussian psychometric function:

\[
g(a, b, x) = \frac{1}{2\pi} \int_{a+bx}^{\infty} e^{-t^2/2} dt
\]

where \(a\) and \(b\) are the parameters to be estimate by fitting the objective metrics values versus the subjective data. To estimate such parameters we use a nonlinear least-squares data fitting by the Gauss-Newton method. We chose this psychometric function since it provided the best fit between our objective metrics and subjective data.

The intuition and the interviews in Experiment I and II confirm us that the watermarking artifacts that produce different kind of noise on the surfaces can be described essentially with roughness. Hence, the objective metric which we choose to measure the strength of the defects is based on an estimation of the surface roughness.

\subsection{Roughness Estimation}

During our studies about 3D watermarking, we have realized that a good measure of the visual artifacts produced by watermarking should be based on the amount of roughness introduced on the surface [210]. Moreover, as just said, the interview phase of the two experiments confirmed us that “roughness” is a good term to describe, in a general way, the defects introduced over the surface of the model. Hence, two objective metrics based on roughness estimation of the surface have been developed. In the following we give a detailed description of these metrics.
The first roughness measure we propose is a variant of the method of Wu et al. [211]. This metric measures the per-face roughness by making statistical considerations about the dihedral angles associated to each face. Wu et al. developed this measure in order to preserve significative shape features in mesh simplification algorithm.

The dihedral angle is the angle between two planes. For a polygonal mesh, the dihedral angle is the angle between the normals of two adjacent faces (Figure 7.8). The basic idea of this method is that the dihedral angle is related to the surface roughness. In fact, the faces normals of a smoothed surface vary slowly over the surface, consequently the dihedral angles between adjacent faces are close to zero. To be more specific, Wu et al. associate to each dihedral angle an amount of roughness given by the quantity $1 - (\vec{N}_1 \cdot \vec{N}_2)$. Given a triangle $T$ with vertices $v_1$, $v_2$ and $v_3$, its roughness is computed as:

$$ R_1(T) = \frac{G(v_1)V(v_1) + G(v_2)V(v_2) + G(v_3)V(v_3)}{V(v_1) + V(v_2) + V(v_3)} \quad (7.18) $$

Referring to Figure 7.9, $G(v_1)$ is the average of the roughness associated the dihedral angles $T - T_1$, $T_1 - T_2$, $T_2 - T_3$, $T_3 - T_4$, $T_4 - T_5$ and $T_5 - T$. In the same way $G(v_2)$ and $G(v_3)$ are the mean roughness associated to the dihedral angles of the faces adjacent to the vertices $v_2$ and $v_3$. Instead, $V(v_1)$, $V(v_2)$ and $V(v_3)$ are the variance of the roughness associated to the dihedral angles of the faces adjacent to the vertex $v_1$, $v_2$ and $v_3$.

A roughly surface can be considered as a surface with a high concentrations of bumps of different size over it. This metric is able to measure 'bumpiness' of the surfaces at face level, but, if the granularity of the surface roughness, i.e. the size of the bumps, is higher than the medium dimension of one face this metric fails to
measure them correctly. In other words this measure does not take into account the scale of the roughness. Our idea is to modify the (7.18) in order to account for different bumps scale. The first step to achieve this goal is to transform this surface roughness estimation in a per-vertex roughness estimation in the following way:

\[
R_1^N(v) = \frac{1}{|S^N_T|} \sum_{i \in S^N_T} R_1(T_i) A_{T_i} \tag{7.19}
\]

where \(S^N_T\) is the set of the faces of the \(N\)-ring\(^1\) of the vertex \(v\), \(|.|\) is the usual cardinality operator and \(A_{T_i}\) is the area of the face \(T_i\). The reason to consider the \(N\)-ring in the roughness evaluation accounts for different scale of bumpiness. Referring to Figure 7.10; the bump of size equivalent to the 1-ring (A) is well measured by \(R_1^1(v)\), a correct value of roughness for the vertex \(v\) in the case (B) is provided by \(R_1^2(v)\). Approximatively, we can state that the roughness of a vertex \(v\) centered on a bump of area close to the area of the faces that form the \(N\)-ring is well measured by \(R_1^N(v)\). This approximation could not be valid in certain cases, for example for high values of \(N\), or when a surface presents high curvature. Hence, a real multi-scale measure of bumpiness would require further developments but we assume that this approximation is sufficient. In order to obtain a single value of roughness for each vertex that accounts for the roughness evaluated at several scales we take the maximum value produced by \(N\)-ring of different size. In particular, in our objective metric we have chosen 3 scales of

\(^1\)The \(N\)-ring neighborhood vertices of a vertex \(v\) is an extension of the 1-ring neighborhood. A 2-ring neighborhood is created from the 1-ring by adding all of the vertices of any face containing at least one vertex of the 1-ring. Additional rings can be added in the same way to form the 3-ring, the 4-ring and so on.
Chapter 7. 3D Watermarking Quality Assessment

Input model

Local variance

Smoothing

Computing differences $d_v(.)$

R$(M)$

Figure 7.11: Smoothing-based Roughness Estimation.

roughness:

$$\mathcal{R}_1(v) = \max\{\mathcal{R}_1^1(v), \mathcal{R}_1^2(v), \mathcal{R}_1^4(v)\}$$  \hspace{1cm} (7.20)

The total roughness of the 3D object is the sum of the roughness of all vertices:

$$\mathcal{R}_1(M) = \sum_{i=1}^{N_v} \mathcal{R}_1(v_i)$$  \hspace{1cm} (7.21)

where $N_v$ is the total number of mesh vertices. In the following we will see how to transform this multi-scale roughness estimation in an objective metric that correlates well to the human perception of geometric defects.

Smoothing-based Roughness Estimation

The second method we developed to measure surface roughness is based on considerations arising during the subjective tests. Since most of the subjects have said, during the interview, that the defects are perceived better on smooth surfaces we decided to develop a smoothing-based roughness estimation. The basic idea of this approach is to apply to the model a smoothing algorithm and then to measure the roughness of the surface as the variance of the differences between the smoothed version of the model and the original one. A sketch of the smoothing-based roughness is depicted in Figure 7.11.

The first step of this approach is to build a smoothed version of the model ($M^S$) by applying a smoothing algorithm to the input model ($M$). Several possibilities for smoothing exist [33, 36, 35, 34]. Here, we decide to use the Taubin filter [33] for its simplicity of implementation. The parameters of the Taubin filter used are the usual $\lambda = 0.6307$, $\mu = -0.6352$. This filter is iterated 5 times. When the
smoothed model is obtained, the distance between each vertex of \( M \) and \( M^S \) is computed in the following way:

\[
d_{OS}(v, v^S) = \text{proj}_{\hat{n}_v}(v - v^S)
\]

(7.22)

where \( \text{proj}(.) \) indicates the projection of the vector \((v - v^S)\) on the vertex normals of the smoothed surface \((\hat{n}_v^S)\). At this point the per-vertex roughness is computed by evaluating the local variance of the distances \( d_{OS}(\cdot) \) around each vertex. To be more specific, for each vertex \( v \), the set of distances associated to its 2-ring \((S^2_\partial(v))\) is built and the variance of this set evaluated. Then, the per-vertex smoothing-based roughness is computed by:

\[
R_2(v) = \frac{V(S^2_\partial(v))}{A_{S^2}}
\]

(7.23)

where \( A_{S^2} \) is the area of the faces that form the 2-ring of \( v \). This area is used at the denominator since surfaces with the same local variance of the distances but smaller area are assumed to be more rough. The roughness of the input model is the sum of the roughness of all vertices of the model:

\[
R_2(M) = \sum_{i=1}^{N_v} R_2(v_i)
\]

(7.24)

where \( N_v \) is the number of vertices of the model.

**Objective Metrics**

Now, we describe how to use the roughness estimation to predict the visual distortions produced by a certain 3D watermarking algorithm. On the basis of several evaluations we decided to define our objective metric as the increment of roughness between the original and the watermarked model. This increment is normalized with respect to the roughness of the original model. In formula:

\[
\mathcal{R}(M, M^w) = \log \left( \frac{\mathcal{R}(M) - \mathcal{R}(M^w)}{\mathcal{R}(M)} + k \right) - \log (k)
\]

(7.25)

where \( \mathcal{R}(M) \) is the total roughness of the original model and \( \mathcal{R}(M^w) \) is the total roughness of the watermarked model. Both \( \mathcal{R}_1(.) \) and \( \mathcal{R}_2(.) \) can be used to obtain two different objective metrics for 3D watermarking quality evaluation. The logarithm is employed to better discriminate low values of relative roughness increments. The constant \( k \) is used to avoid numerical instability of the (7.25) since the logarithm tends to \(-\infty\) for \( M^w \) very close to \( M \). In particular the value
of $k$ has been set to normalize the values provided by the metric between 0 and 10, that is the same range of values used by the subjects during the experiments. In the following, we indicate with $R_1(M, M^w)$ the objective metric based on the multi-scale roughness and with $R_2(M, M^w)$ the objective metric based on the smoothing-based roughness estimation.

7.7 Experimental Results

In this section we analyze the performances of the two proposed objective metrics and we compared them with geometric metrics usually adopted in literature for model quality evaluation. First, the correlation between the subjective Mean Opinion Score (MOS) collected in Experiment I and the distances given by two geometry metrics based on Hausdorff distance for model similarity is evaluated. In this way we obtain a term of comparison for the evaluation of our metrics. Then, the objective metrics are fitted with a gaussian psychometric curve to match the subjective data collected in the first experiment. The performances of the perceptual metrics so obtained, are evaluated using the subjective MOS provided by the Experiment II. In other words the subjective data of the Experiment II are used to validate the developed perceptual metrics. The results obtained will be discussed at the end of the Section.

7.7.1 Hausdorff distances

As previously stated (Section 7.2) two of the most common geometric metrics used to measure the similarity between two 3D objects are the Maximum (7.3) and the Mean geometric error (7.5). These two metrics are based on the Hausdorff distance between models’ surface. Here, we want to evaluate if the distance between the original and the watermarked model could be a reliable metric for perceptual watermarking impairments prediction. To do this, the Hausdorff distances of each watermarked models from the original are plotted versus the MOS provided by the Experiment I. At this point, the linear correlation coefficient of Pearson ($r_P$) [212] and the non-linear (rank) correlation coefficient of Spearman ($r_S$) [213] are calculated in order to evaluate the global performances of the metric obtained by fitting these geometric data with a cumulative gaussian (7.17). The Spearman rank correlation coefficient is a measure of the strength of monotone association between two variables. Even if a psychometric curve is used to fit the geometric measures the results do not correlate well with subjective MOS. This underlines
the fact that $d_\infty(.)$ and $d_1(.)$ are not designed on the basis of how humans perceive geometric defects. The results are summarized in Figure (7.12). Such results will be used as a reference to compare the performances of the perceptual metrics based on roughness estimation.

7.7.2 Roughness-based metrics results for Experiment I

As we stated in Section 7.5 the goal of the first experiment was to make an initial study on the perception of the geometric defects caused by watermarking algorithms. The experimental data confirm that the subjective perception of the impairments is well-described by a measure of roughness. The subjective data of this experiment are used to obtain two perceptual metrics, named $\mathcal{R}_1^*(M, M^w)$ and $\mathcal{R}_2^*(M, M^w)$, from the corresponding two proposed objective metrics $\mathcal{R}_1(M, M^w)$ and $\mathcal{R}_2(M, M^w)$. Those perceptual metrics are obtained by fitting these subjective data with a gaussian psychometric curve (7.17). In this way two kind of perceptual metrics are obtained, one based on multi-scale roughness measure and another one based on smoothing-based roughness estimation. The parameters of the gaussian psychometric curve after the fitting are $(a = 1.9428, b = -0.2571)$ for $\mathcal{R}_1(M, M^w)$ and $(a = 2.0636, b = -0.2981)$ for $\mathcal{R}_2(M, M^w)$. The smoothing-based one provides a better fit ($r_P = 0.6730$, $r_S = 0.8680$) than the multi-scale one ($r_P = 0.8286$, $r_S = 0.8919$) as depicted in Figure 7.13. The 95% confidence intervals of the subjective MOS versus the roughness metric are depicted in Figure 7.13 (Top). Few confidence intervals are large, approximatively the 20% of the maximum range scale. Note that the width of the intervals would have been reduced if the experiment had been carried out with more subjects. On the right top
of the graphs it is possible to notice some points outside ones of the fitting curve. Most of these outliers correspond to the Venus model. This is due to the fact that the Venus model represents a human face. Human face images are well-known in subjective experiments as a high level factor attracting human attention, i.e. people are more able to deal with human faces, so the distortions on the Venus head results more visible and annoying with respect to the other models.

7.7.3 Objective Metrics Performances

As discussed in previous section, the two proposed objective metrics have been transformed into two corresponding perceptual metrics using the data from the Experiment I. In order to evaluate such metrics Experiment II was carried out with three other watermarking algorithms: KDK, NBE and VFA. The validation is very simple: the perceptual metrics obtained in Experiment I are used to predict the MOS obtained in the second experiment and their correlation coefficients are computed. The correlation coefficients \( r_P \) and \( r_S \) are reported in Table 7.9.
The rows indicate the watermarking algorithm groups. The first two columns of this table report the Spearman correlation coefficient of the Maximum and Mean geometric error for comparison. The third and the fourth column shown the values of $r_P$ and $r_S$ for $\mathcal{R}_1^*(M, M^w)$, while the last two columns are the $r_P$ and $r_S$ values for $\mathcal{R}_2^*(M, M^w)$. Referring to this table we can make the following important considerations:

- Overall, both geometric metrics based on the Hausdorff distance do not correlate well with the subjective data. On the other hand the developed metrics exhibit strong correlation with subjective data, in particular concerning the Spearman’s coefficient.

- The Spearman’s coefficients for the NBE and VFA algorithms (second and third rows respectively) demonstrate that both metrics are able to predict impairments introduced by these two algorithms.

- The worst performances of the proposed metrics are obtained for the KDK algorithm. This can be explained by considering that the distortion produced by the KDK algorithm on the surface are non-uniform.

- The overall performances of the perceptual metrics for the watermarking algorithms that introduce uniform distortions on the surface shown are reported in the 6th row of the table. The values of the correlation coefficients ($r_P = 0.6455$ and $r_S = 0.8416$ for the first metric, $r_P = 0.7383$ and $r_S = 0.8954$ for the second metric) are very high. Hence, the developed metrics provide a very good prediction of the impairment caused by 3D watermarking.

- The overall performances of the perceptual metrics considering all the uniform and non-uniform watermarking algorithms tested are reported in the last row of the table. Despite the presence of the KDK algorithm, for which the performance are not high, the global prediction of the metrics still remains good. In particular, such performances are excellent comparing with the ones of the two geometry-based metrics.

In order to visualize the results of the Table 7.9 the graphs of Figure 7.14 show the values of the objective metrics plotted versus the subjective MOS for several watermarking algorithm groups. The curve drawn on this figure does not represent the result of a fit; the same gaussian curve obtained with the data of the
Table 7.9: Perceptual metrics performances.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Hausdorff Distance</th>
<th>( R_1(M, M^w) )</th>
<th>( R_2(M, M^w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCB</td>
<td>Max ( r_S )</td>
<td>Mean ( r_S )</td>
<td>( r_P )</td>
</tr>
<tr>
<td></td>
<td>0.6672</td>
<td>0.6595</td>
<td>0.6730</td>
</tr>
<tr>
<td>KDK</td>
<td>0.6904</td>
<td>0.3230</td>
<td>0.6154</td>
</tr>
<tr>
<td>NBE</td>
<td>0.7087</td>
<td>0.7026</td>
<td>0.5597</td>
</tr>
<tr>
<td>VFA</td>
<td>0.4951</td>
<td>0.8815</td>
<td>0.7472</td>
</tr>
<tr>
<td>KDK, NBE, VFA</td>
<td>0.3759</td>
<td>0.4853</td>
<td>0.4877</td>
</tr>
<tr>
<td>UCB, NBE, VFA</td>
<td>0.5219</td>
<td>0.6183</td>
<td>0.6455</td>
</tr>
<tr>
<td>ALL</td>
<td>0.4993</td>
<td>0.5352</td>
<td>0.6098</td>
</tr>
</tbody>
</table>

Experiment I are drawn for all the picture. In other words these graphs visualize the behavior of the KDK, the NBE and the VFA algorithm with respect to the perceptual metrics developed after Experiment I, that is represented by the red curve (the dashed line is the confidence interval for that curve).

Since the non-linear correlation coefficient of Spearman is based on the rank of the data instead on the data values itself like the Pearson’s coefficients, it is interesting to compare the watermarked models ranking by the impairments perceived by the subjects and by the impairments predicted by the metrics. An example of this comparison is reported in Figure 7.15 where the Bunny and the Feline models are considered. It is possible to see that the smoothing-based perceptual metric, that has values of \( r_S \) slightly higher with respect to the multi-scale metric, is able to rank the watermarked models in a way very close to the subjective rank.
Figure 7.14: Experiment II: Subjective MOS vs objective metric curves. The parameters of the fitting curve are the same of the Experiment I.
Figure 7.15: Experiment II: Comparison between models’ impairment ranking. On the left the models are ranking by subjective MOS. On the right the models are ranking by smoothing-based roughness.
7.8 Conclusions

In this work, our investigations about the extension of the ideas of perceptual image watermarking to 3D watermarking have been presented. In particular, a new experimental methodology for subjective quality assessment of watermarked 3D objects has been proposed. The analysis of the data collected by two subjective experiments that use this methodology demonstrates that such methodology is well-designed and provides reliable subjective data about quality evaluation of watermarked 3D objects. Moreover, two perceptual metrics for 3D watermarking impairment prediction have been developed by combining roughness estimation with subjective data. The performances of these metrics have been deeply analyzed. The results of this analysis demonstrate the effectiveness of the proposed perceptual metrics with respect to the state-of-the-art geometric metrics commonly used for models comparison. More important, the experimental results show that the proposed metrics provide a good prediction of the human perception of the distortions introduced by 3D watermarking over the model’s surface. Hence, these metrics could be used in a feedback mechanism to tune the watermarking parameters of 3D watermarking algorithms optimizing the watermark insertion. For example, referring to our watermarking algorithm, for each level of resolution the maximum amount of watermark strength before to reach watermark perceptibility can be easily computed using these metrics, thus improving the robustness of the algorithm while ensuring imperceptibility.

Concluding we can state that, despite the perceptual evaluations of geometric defects is a very difficult task due to the enormous number of factors that influence it, these first results are very encouraging. Further researches can be regard the evaluation of the performances of the proposed metrics under different rendering conditions and the extension of the proposed metrics to taking into account the influence of the local properties of the surface (e.g. curvature, protrusions) on the perception of the geometric artifacts.
Appendix A

Experiment Scripts

After getting the subject into position, centered in front of the screen and at the correct distance (about 0.4 cm), the following instructions are read:

• “This test concerns the evaluation of the distortions or impairments introduced by watermarking algorithms on the surfaces of 3D models.

What is a 3D model? A 3D model is a collection of data that represent a 3D shape. 3D models are used in particular in entertainment industries, for examples movies and video games use a lot of 3D models.

What is watermarking? Digital watermarking is a technology used to embed information inside a digital media. Imagine you want to associate some information with a digital media, i.e. the name of the owner of an image to the image itself. A watermarking algorithm, specific for images, can process the image and embed this information inside the image data itself. So, this information can be eliminated only modifying the watermarked image. In order to embed the data watermarking algorithms modify some properties of the digital media producing always some distortions in the watermarked media. The purpose of the test is the evaluation of the distortions introduced by watermarking algorithms on 3D models. Usually these distortions are visible on the model surfaces. So, the test is very simple: you interact with some models and you have to indicate if you see or not a certain kind of distortions. Obviously, I will show to you these distortions so you can understand what you have to evaluate.

• During the test you interact with the models and you have to evaluate these impairments. In particular you will indicate whether you detect any distortions.
tion or impairment.

- For those 3D models you detect a distortion you will indicate *how much* you perceive such distortion by entering a number that is proportional to its distortion value.

- Additionally you will have to indicate the part of the models *where* the distortions are more evident (this task is present only for Experiment I).

- Here I will show you the models without any distortion. The test includes four models. You have to imagine these models like *statues*. In particular I will show to you a model called "Venus" that represent the head of a statue of Venus, a mythological feline with wings called "Feline", a "Horse" and, finally a "Bunny".

- [Show originals]

- Are you able to remember these models? Do you want to see it again?

- Before we start the experiment, you will see how the typical distortions introduced by watermarking process look like. In few words the roughness of the surface of the model is increased in some way. We have to recognize this roughness, so it is important that you remember, for each model, the roughness of its parts. Another important aspect that you have to consider during your evaluation is that the distortions introduced by the algorithm is uniform on the surface.

- [Show the watermarked models]

- Have you understood how these distortions look like? Do you have any questions?

- In this phase, you will learn how to interact with the 3D model. You interact with the models by using the mouse. To rotate the model push the left button and moves the mouse. When you want to stop to rotate the model release the left button. To zoom the model push the right button and move the mouse ahead or back to zoom in or to zoom out. When you want to stop to zoom the model release the mouse button.

- [Interaction trial - Try to rotate the model. Try to zoom it. You can move the model left/right/up and down with the arrow keys. Try.]
• (Only for Experiment I) Remember that you have to decide even \textit{where} the distortions are more evident. This is very important, so take in account, while you are interacting with the model, that you have to decide how much you perceive the distortion and where these are more evident. To indicate the part of the model that presents the most evident distortions we have to use this viewfinder, this sight [activate selection mode]. You can move the models as usual. The rectangle can be resized by using the keys 'A', 'D', 'X' and 'W'. Pay attention. You have to move the model to select the part, not the selection rectangle. Press <ENTER>, on the keyboard, to confirm your selection. Now, try to indicate some parts on this model. [Interaction trial - selection]

• Have you understood? Do you have any questions?

• Now I will show to you some cases that help you to evaluate numerically \textit{how much} you perceive the distortions. You have to choose the worst case and mentally assign a value of 10 to it. This will give you an idea of the distortions that you will be seeing. The distortions could be present or not on the model. So, you are assigning a score of 10 to the most evident distortions. If the perception of the distortions during the test is half of the worst examples you chose, give it 5; if it is 1/10th as bad give it 1, if it is 1.5 times as bad, give it 15. This is important, you can give score higher than 10. Remember that the question is \textit{how much you perceive, how much you notice such kind of distortions}. I will show these examples now.

• [Show worst cases]

• Before we start the experiment you will have six practice trials to be sure that you understand the task. You will respond in these trials just like you will in the main experiment. The questions appear on the screen. So, you have to provide three answers (two for the Experiment II) after the interaction with the model. The first question is: did you notice any distortion? You can answer <YES> or <NO> at this question. Then, in case of positive answer you have to give a score to indicate \textit{how much} the distortion is evident. You use the numeric keypad to enter the perception value. And finally, the third question, (only for Experiment I) \textit{where} you noticed the distortions. To answer to this question you have to select the part of the model with the most evident distortions.
Appendix A. Experiment Scripts

- Do you have any question? Do you want to repeat it?

- [Practice Trials]

- 40 models will be shown to you during the test (48 in Experiment II). This takes about 20 minutes (24 minutes for Experiment II) plus the time you need to indicate where the distortions are more evident (only for Experiment I). So, the test will takes about 30 minutes.

- Before to start the test I would like to give you the following practical recommendations:

  1. In case of input error, please tell me what you want to do and I correct your answer at the end of the test.
  2. You can take a break at any time by entering your answers (score and part) for the most recent models, but waiting to hit <ENTER> until you are ready to go on.
  3. Finally, at the end of the test I will ask to you few questions.

- Do you have any question before to start?

- [Start the experiment]

- [Interview]

  1. What is your feeling with the models? I mean, have you experienced any problem to identify the distortions on a specific model and why?
  2. How would you describe the distortions that you saw?
  3. Have you comments or remarks about the tests?
Appendix B

Publications related to the PhD research

International Conferences Papers


National Conferences Papers

Bibliography


[130] [Online]. Available: http://www.multires.caltech.edu/software/pgc


