Part II

Shape Perception

Chapter 3

Perception of a 3D Shape

3.1 Introduction

This Chapter presents the issues related to the perception of 3D shapes rendered on 2D screens. We focus on the way a shape is visually perceived in function of its sampling and of the chosen rendering conditions as well as in function of the type of distortions this shape has undergone. The aim of the overview presented here is twofold: determining factors which influence visual perception of a shape as well as exploring qualitatively how people are more sensitive to local or global distortions. These considerations are important for many Computer Graphics applications and are crucial to improve the design of watermarking schemes which attempt to hide data, generally by introducing small unnoticeable distortions.

3D shape visualization is usually performed on 2D screens. There exist other ways to perceive 3D objects such as stereovision techniques (Crystal Eyes, interlaced, Dresden, red-blue, ... [51]) and haptic devices which enable the user to feel the shape with a specific interface (e.g. a glove, a stylus... [110]). In this thesis, we restrict our study to rendering on 2D screens. This is the most popular way to use 3D shapes [87]. Of course, it could be interesting to analyze how shape distortions perception is different under other modalities such as explored in [110].

The organization of this Chapter is the following. First, Section 3.2 presents and illustrates the most common rendering techniques. Emphasis is put on the consequences of the selected rendering conditions on the shape perception (i.e. from real-time rendering to off-line photorealistic

rendering). Subsampling is also considered for Level-Of-Detail dependent perception. Section 3.3 then presents state of the art 3D objective metrics for 3D surfaces which have been almost all developed for simplification purposes. Perceptive metrics are also introduced in order to give a large overview of this field which will be useful to introduce the next Chapter.

3.2 Visualization Techniques

Applications making use of 3D shapes for visualization range from Computer Aided Design (CAD), simulation and medical imaging to movies and video games. The way 3D objects are displayed on a screen is generally referred to as *rendering*. This section introduces basic elements and ideas of most usual rendering techniques from real-time interactive displays to photo-realistic images or movies. The reader may find more detailed information in [51].

3.2.1 Rendering

Rendering refers to the algorithms and techniques involved in the process of generating a 2D image from a scene composed of geometrical primitives. A wide set of techniques have been developed in two major directions : *real-time rendering* and *off-line rendering*. Real-time rendering enables *interactions* and its speed is often measured in frames per second (fps). Scientific applications may already provide real-time interactions between 15 to 30 fps, while video games are typically rendered at 50 fps for more fluidity. Off-line rendering, used for example for producing movies, targets photo-realistic or aesthetic images without any interaction. Most popular off-line rendering techniques are *Ray Tracing*, *Photon Mapping* and *Radiosity*.

Real-time rendering pipeline

The graphics rendering pipeline is the core of real-time rendering systems. The rendering pipeline processes geometrical and appearance data, lighting conditions, camera position etc. The pipeline is composed of three main stages : the application stage, the geometry stage and the rasterizer.

The application stage depends on the application interactions and determines what has to be visualized.

The geometry stage usually processes 3D triangular meshes (other representations such as NURBS must be converted before rendering). The first operation is the conversion of the coordinates of each 3D mesh included in the scene into a common coordinate system referred to as *3D world* coordinate system. The next step consists in computing *eye coordinate system* based on the virtual camera position and viewing direction. Vertex normal directions are computed and a lighting model is applied to illuminate the scene. Then the view volume of the scene is calculated on the basis of the projection used (e.g. *orthographic* or *perspective*). The view volume is usually called *frustum* and is a rectangular box for orthographic projection and a truncated pyramid for perspective projection. The parts outside this volume are *clipped* before the mapping on the screen. Then each vertex receives two *screen coordinates* as well as an elevation coordinate (a.k.a. *z-value*).

The rasterizer stage determines the color of each screen pixel through the analysis of the output of the geometrical stage, i.e. projected vertices, colors and textures. The rasterizer also resolves the visibility of each polygon by using the z-value of each vertex.

Ray Tracing

Ray Tracing is the most well-known technique to achieve photo-realistic images (see Fig.3.1). The basic idea is to cast a ray of light from the eye of the observer to the (screen) image plane and then to the scene to determine what it intersects. Each pixel of the screen is then colored accordingly to the information returned by the ray. Thousands of such tracing rays are casted to produce the rendering of the scene. In opposition to the rendering pipeline, this technique implicitely solves the perspective transformation, the clipping phase and the hidden surfaces removal. Knowing the intersections of each ray with the scene, the closest intersection is shaded. This technique also enables transparency but becomes computationally expensive. More details can be found in [51].

3.2.2 Lighting Models

The mathematical description of the interactions between the lights of the scene and the surface of a given object of this scene is called *lighting model*. The quality of photo-realistic images strongly depends on the complexity and accuracy of the lighting model and the material information

associated to the 3D objects. On the contrary, real-time rendering needs a trade-off between the quality of the light reflectance perception and a good frame rate.

The interactions of light with a surface are usually described by the following classical model:

$$I_i = I_r + I_t + I_s + I_a \tag{3.1}$$

where I_i is the incident light, I_r is the light reflected by the surface material, I_t is the light scattered and I_a the light absorbed by the material. The reflected light can also be decomposed in two different effects : the *diffusion reflection* and the *specular reflection*. Diffusion reflection is responsible of the color of the objects. A perfect diffusive surface scatters light uniformly in all directions and consecutively, does not depend on the observer position. The intensity of the diffuse light is usually computed by the Lambert's Law [51]:

$$I_d = I_i K_d cos(\theta) \tag{3.2}$$

where K_d is the diffusion reflection coefficient, I_i is the intensity of the incident light and θ is the angle between the surface normal at the considered point and a line connecting this point with the light source. This direction is generally denoted by $\vec{L}$. The maximum received light is obtained when the normal of the object is aligned with $\vec{L}$. The specular reflection depends on the glossiness of the illuminated surface. A matt surface has no specular effect and a high diffusive behavior while a glossy surface is a mirror. Specular reflection is known to reflect the incident light in a symmetric way: the angle between the incident ray and the normal direction is equal to the angle between the normal direction and the reflection direction. So, the perceived specular reflection depends on the position of the observer.

Light Sources

Lighting simulations are often based on several light sources. A light source is often approximated by a point. Volumetric light sources also exist but require much more complex computations. Light sources can be classified in three different categories:

- point light models: photons are emitted uniformly in all directions. This light source is defined by its position and by the color of the light.

- directional light models: can be viewed as a point light positioned at infinity. This light source is completely defined by the direction of the emitted light. It is often used to model the light of the Sun.
- spot light models: emit light in a cone. Defining this source of light requires its position, the directional vector and the cut-off angle. Light is generally not uniform in the cone, a light intensity modulation is controlled by a parameter called *spot exponent*.

The color of the light is usually defined by its RGB components. Its intensity should physically decrease proportionally to the square of the distance to the light source but is assumed to be constant for real-time rendering.

Phong Model

The Phong lighting model is the standard model used in Computer Graphics [51]. The light is described by three components:

$$I_{Phong} = I_{amb}K_a + I_iK_d(\vec{N} \cdot \vec{L}) + I_iK_s(\vec{R} \cdot \vec{V})^n \tag{3.3}$$

where $\vec{N}$ is the surface normal, $\vec{L}$ is the direction of the incident light and $\vec{V}$ is the vector supporting the line between the observer and the considered point of the object. The first term of this equation is the *ambient term* which 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 3.2. The third term models the specula light and is parameterized by n which depends on the object material. K_a, K_d and K_s are coefficients determined by empirical results.

Cook and Torrance

Cook and Torrance [51] have proposed a photo-realistic model based on the idea that a surface is composed of a collection of microfacets, each behaving like a mirror. This model also considers the energy conservation between the incident and the reflected light as well as the change of the color within the specular highlight. The specular component of the light is modeled as:

$$R_s = \frac{DGF_\lambda(\theta_i)}{\pi(\vec{N} \cdot \vec{L})(\vec{N} \cdot \vec{V})} \tag{3.4}$$

where the term D depends on the distribution of the microfacets orientations, G is the geometric attenuation factor and $F_\lambda(\theta_i)$ is the Fresnel term. The geometric attenuation models the masking and shadowing effects of the microfacets with respect to the normal directions. The Fresnel term takes into account the color change of the specular highlight as a function of the angle of incidence and of the wavelength.

Energy conservation is also modeled to compute light intensities. The total reflectance of the light results in a linear combination of the diffuse and specular components:

$$R_{bd} = K_d R_d + K_s R_s \tag{3.5}$$

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 $K_s + K_d = 1$.

BRDF

The Bi-directional Reflectance Distribution Function (BRDF) [51] is a mathematical description of the scattering of incident light by a particular surface. The BRDF is function of the incoming light direction, of the viewer direction, of the position on the surface and of the wavelength. BRDF is the ratio between the radiance in the viewer direction and the irradiance incident at a surface. The irradiance measures the amount of incident light power in terms of flow through an area and is expressed in $Watt/m^2$. The radiance measures the power of the outgoing light per area and solid unit angle. To use a BRDF for lighting, its values must be determined in each incoming and outgoing directions. This can be achieved either by an analytical model, either by acquiring data from the real world through gonioreflectometers.

Basic Shading Techniques

Shading refers to the process of performing lighting computations and determining the colors of the pixels. The most used shading techniques are the *flat*, *Gouraud* and *Phong* shading [51]. These respectively correspond to per-face, per-vertex and per-pixel light. Flat shading computes light reflection by using the face normal direction. The quality of the shading

strongly depends on the level of detail (LOD). Gouraud shading computes light reflection by using vertex normals. Light and colors are interpolated on the adjacent faces. This shading technique does not behave well with highlights and spot lights. Phong shading interpolates vertex normals instead of the color of the vertices. This model is much more computationally expensive but provides high quality images (see Fig.3.1).

3.2.3 Textures

3D objects can also be visualized with textures (see Fig.3.2). Textures are images which are mapped on the triangles of the object to display. These images can give a better perception of the object shape with much less LOD than only rendering the 3D object itself. Another kind of textures consists in modulating the surface opacity to simulate transparent or particular objects such as clouds or water. There exist a wide variety of texturing techniques: *image texture mapping*, *bump mapping*, *gloss mapping*, *alpha mapping*, *environment mapping*...

Texture mapping may be viewed as the process of associating image pixels (a.k.a. *texels*) to the vertices or faces of a 3D object. A parameterization (i.e. (u, v) coordinates) of the object must be found for each vertex ((x, y, z) coordinates). The function providing such a mapping between the 3D coordinates to the 2D parameterization is referred to as *projector function*. A *corresponder* function maps parameter coordinates to the texels. The information of the corresponding texel is then used to modify the color of the resulting pixel, accordingly to the selected shading and lighting models.

Simple projector functions include spherical, cylindrical and planar projections. More complex and accurate projectors are based on digital parameterization algorithms. These algorithms presented in Chapter 2 usually minimize angle, edge length and/or triangle area distortions. The corresponder function maps parameter-space values to the texels allowing periodical repetitions of a same image or the mapping of several textures on a single object (*chart mapping*).

Image texture mapping also relies on *interpolating functions*. This process is often called *texture magnification* and is based on nearest-neighbor,

bilinear or trilinear interpolation. Aliasing can appear if the texture resolution is higher than the object resolution. In this case, several texels are mapped on the same vertex. Several techniques enable to avoid such visual distortions and are referred to as *mipmapping* [51].

Bump mapping techniques modulate the *surface normals* on the basis of the texture information (see Fig.3.3). The shape is then represented with more detailed curvature information which follows the texture variations. This kind of mapping allows to render objects with a low mesh LOD while representing a very detailed geometry such as musculature, folds in clothes, hair and so on. The texture is mapped to a heightfield which is generally used for modifying the surface normals. The slope of the normals in u and v parameter directions is obtained by derivation. Another method stores normals in a texture called *normal map*. This normal map is a RGB image where pixels represent normal vectors. Image processing is then applied on this texture in a process called *embossing*.

Gloss mapping simulates non-uniformity in shiny surfaces. It modulates the contribution of the specular component over the surface. The material properties of the surface are stored in textures instead of per-vertex attributes.

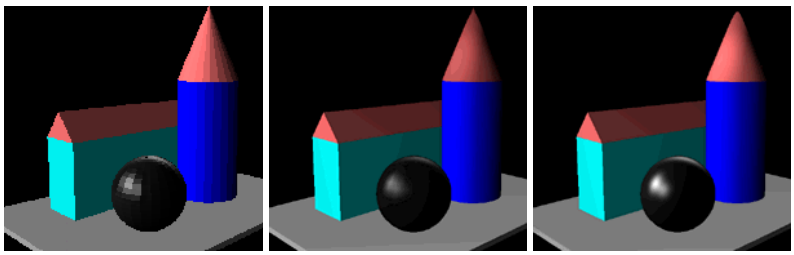


Figure 3.1: *From left to right: the same scene rendered with flat, Gouraud and Phong shadings respectively (image courtesy of [98]).*

3.2.4 Global Illumination

The generation of synthetic images with outstanding levels of realism is often based on *global illumination*. Compared to the already presented lighting models, global illumination enables to capture a lot of complex

Figure 3.2: *Image texture mapping on the Stanford bunny model (image courtesy of [88]).*

Figure 3.3: *From left to right: the same textured model with bump mapping and environment mapping, respectively (image courtesy of [98]).*

visual effects. Lighting models are *local* and do no take into account the effects of indirect light i.e. the interactions of the light reflected by other objects of the scene. Ray Tracing is a possible algorithm to compute global illumination effects such as indirect light, soft shadows (contours of shadows are smoothed thanks to volumetric light sources), color bleeding (the color of an object influences the color of other objects) and caustics (concentrations of reflected light e.g. light concentrated by a glass). Path Tracing and Photon Mapping are other techniques which are more accurate than Ray Tracing. Monte Carlo sampling and integration are often necessary to solve the equations generated by the system of rendering conditions imposed by light sources, objects and their properties, etc.

3.2.5 Non-Photorealistic Rendering (NPR)

Some applications do not require photorealistic rendering. This is the case of scientific visualization tools dedicated to the analysis or understanding of physical phenomena or complex biological structures [67]. There exists a wide variety of techniques to produce such renderings. NPR

systems often require real-time processing. They are classified in four categories: 2D with user intervention, 2D without user interaction, 3D with user interaction and 3D without user interaction. We focus in this section on 3D techniques.

Some NPR techniques reproduce painting techniques such as sketch, pen and ink, hatching etc. Emulation of pen and ink illustrations by rendering (usually in conjunction with stroke textures) follow principal curvatures directions, feature lines and contours in order to mimic the painting techniques of an artist. Hatching which mimics hand-drawn illustrations, also follows curvature information but must modulate the density of hatches to render light and shadow effects.

3.2.6 Level of Detail

The number of factors determining the visual appearance of 3D shapes is considerable. Lighting models, shading techniques, the presence of textures, the properties of the 3D shape material and particular effects provide very different rendering qualities. Other important factors influencing visual quality of a 3D model are its scale and its Level of Detail (LOD). Indeed, a recurring theme in Computer Graphics is to trade fidelity for performance as illustrated on Fig.3.4 and Fig.3.5. Two different LODs are difficult to distinguish if after rendering each pixel represents several triangles. It is indeed possible to link scale, screen resolution and LOD [79].

Modifying the LOD of a polygonal mesh is performed by simplification algorithms [54, 61]. These algorithms iteratively subsample the original model by removing vertices (*vertex removal*), edges (*edge or halfedge collapse*) or faces (*triangle removal*). At each iteration, the element to remove is selected so that its removal minimizes a given distortion metric or energy. These distortion metrics or energies basically use some measures of 3D distance. This is the subject of the next section.

Notice the perception of different LODs strongly depends on the selected rendering technique. At a given scale, the simplification of 30% of a 3D model vertices can be unnoticeable using Ray Tracing while distortions are clearly visible when using flat or Gouraud shading. Textures also hide the effect of reducing the LOD or of the presence of noise on vertex

positions. For example, normal directions are resampled and smoothed by bump mapping. Attribute data such as light reflectance properties and color also modify the perception of the 3D model fidelity. This partly explains why 3D model quality assessment is complex.

Figure 3.4: *The same model with different LODs. From left to right:* $250,000$*;* $62,000$*;* $6,800$ *and* 975 *polygons (image courtesy of [132]).*

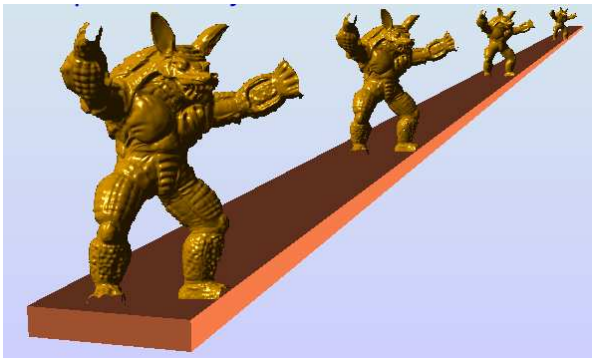


Figure 3.5: *Scale and LODs (corresponding to Fig.3.4). Lower LODs can produce the same perceived quality at lower scales (image courtesy of [132]).*

3.3 Three-Dimensional Distance Metrics

As introduced by the precedent section, simplification algorithms have been the first application for which 3D metrics have been developed. These metrics usually measure a function of 3D Euclidian distances between points. Since real-time processing is the most desirable quality of a 3D distortion measure in the case of a simplification algorithm, these metrics are known to give only rough approximations of the shape quality

perception. Other metrics have been designed for 3D compression algorithms, 3D model retrieval and 3D watermarking. These metrics can be classified in the following way:

- Hausdorff distance
- Volume based measure
- Quadric Error Measure
- Curvature based distance

3.3.1 Hausdorff distance

The *symmetric Hausdorff distance* $H(M_1, M_2)$ between two meshes denoted M_1 and M_2 is defined as :

$$H_{max}(M_1, M_2) = \max\{\max_{a \in M_1} \min_{b \in M_2} d(a, b), \max_{a \in M_2} \min_{b \in M_1} d(a, b)\}, \tag{3.6}$$

where $d(a, b)$ stands for the Euclidian distance between points a and b in the 3D space. This metric is usually referred to as *maximum geometric error*. Another definition of the Hausdorff distance between 3D meshes is the *mean geometric error*:

$$H_{mean}(M_1, M_2) = \frac{1}{A_{M_1} + A_{M_2}} \{\int_{a \in M_1} \min_{b \in M_2} d(a, b) + \int_{b \in M_2} \min_{a \in M_1} d(a, b)\} \tag{3.7}$$

where A_{M_1} and A_{M_2} are the area of meshes M_1 and M_2, respectively. This metric is better correlated to perception for quite similar meshes, guaranteeing a specific error bound between two meshes. However, it tends to be less meaningful as the measure grows larger. It also does not take into account linear transformations of the meshes to be compared. Furthermore, computing this distance is computationally expensive. Several optimization heuristics have been developed to provide bounds of the Hausdorff distance in order to apply it to the simplification application [14, 72, 82]. The most popular implementation that has been widely used in the literature to compare results is the Metro tool [35].

3.3.2 Volume based metric

The first *volume based metric* has been developed by Alliez et al. [10]. The error between two meshes M_1 and M_2 is defined as $V(M_1, M_2)$, where V

is a Lebesque formula. The main argument is that if M_2 is a simplified version of M_1, then M_2 is the best approximation of M_1 if the volume between both meshes is minimized. It is also shown that this metric enables to restore original sharp edges when the triangle count is sufficient.

3.3.3 Quadric Error Measure

The *Quadric Error Measure* (QEM) is a very common local metric to perform edge collapse operations. Usually, the error is locally estimated on the neighborhood of a point of mesh M_1 and its corresponding neighborhood on M_2. A quadric approximation of these neighborhoods enables to find corresponding pairs of points [54].

3.3.4 Curvature Based Distance

The *curvature based distance* has been proposed by Kim et al. [79]. Their approach is motivated by the fact that human vision is sensitive to curvature direction changes. They use differential geometry and decompose the *local* error in three distinct components: distance, tangential and discrete curvature. This error computed for edge e of M_1 is expressed as:

$$E(e) = f(e) + f_1(e) + f_2(e), \tag{3.8}$$

where $f(e)$ is a QEM, $f_1(e)$ is a tangential error function based on the magnitude of the difference between normal vectors of the facets incident to e and $f_2(v)$ is a discrete curvature error function, for example Gaussian, mean or principal curvatures.

3.3.5 RMSE, VSNR and Geometric Laplacian

Other 3D distance metrics are also used for comparing compressed meshes sharing the same connectivity: the *VSNR* (Vertex Signal to Noise Ratio a.k.a. RMSE Root Mean Square Error) and the *Geometric Laplacian* [74]. The first metric consists in adding euclidian distances between corresponding vertices of M_1 and M_2:

$$RMSE = VSNR = \frac{1}{N} \sum_{i=0}^{N-1} d(a_i, b_i), \tag{3.9}$$

where N is the number of vertices in M_1 and M_2, $d()$ is the euclidian distance in 3D space, a_i and b_i are respectively the i^{st} points in M_1 and M_2.

The second one (denoted by GLD) has been designed to capture the local smoothness of the mesh using a Geometric Laplacian:

$$GLD(M_1, M_2) = \frac{1}{2N}\left(\sum_{i=0}^{N-1} \|a_i - b_i\| + \sum_{i=0}^{N-1} \|GL(a_i) - GL(b_i)\|\right), \quad (3.10)$$

where notations are similar to Equation 3.9 and GL is given by:

$$GL(a_i) = a_i - \frac{\sum_{j\in N(a_i)} l_{ij}^{-1} a_j}{\sum_{j\in N(a_i)} l_{ij}^{-1}}, \quad (3.11)$$

where $N(a_i)$ is the neighborhood of a_i in the mesh and l_{ij} is the euclidian distance between a_i and a_j. The advantage of this metric is to differentiate random noise addition on the vertices and poor reconstruction quality. As human eye is very sensitive to local surface smoothness, the metric increases in case of local disturbances detected by the geometric laplacian. However, as illustrated on figure Fig.3.6, these metrics based on 3D point to point distances do not capture at all the perception of distortions on a given shape.

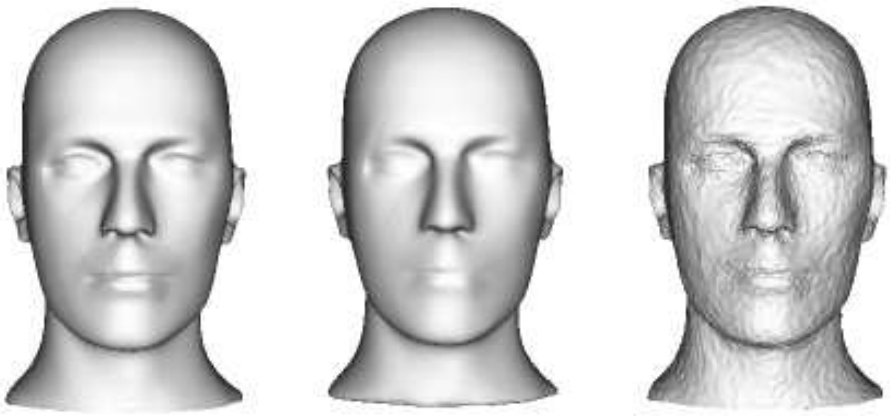
In conclusion, while being widely used to compare results, three-dimensional distance metrics usually do not capture visual similarity. A few papers proposed to tackle this issue with essentially two distinct strategies: image-driven distance measure and perceptual models.

3.3.6 Image-based metrics

Assessing the visual similarity of two meshes can be done by comparing their 2D projections. Lindstrom and Turk [89] propose to compute root mean square differences of pairs of corresponding 2D views. They only use one single luminance channel and generate the different views by moving a camera on the 24 vertices of a bounding *rhombicuboctahedron*. This metric has been used to assign edge collapse costs in a simplification scheme. Furthermore they show this type of metric provides better results than some 3D distance based metrics.

3.3.7 Perceptive models

Surprisingly, the modelization of visual perception of 3D meshed surfaces has not been deeply investigated so far. However, Reddy [112] assesses perceptual issues relating to the visualization of 3D meshes and



(a) **(b)** **(c)**

Figure 3.6: *Perception and usual 3D metrics. (a) the Stanford* Head model*, (b) the same model after* 430 *iterations of Laplacian smoothing, (c) the same model after a noise addition of* .17 *percent. Comparing (a) with (b) and (a) with (c) with the RMSE metric leads to the same score of approximately* .0001 *while the perceived distortion is much more important for model (c).*

proposes a mathematical model of human visual acuity based on contrast and spatial sensitivity measures. *Perceptually-guided rendering* is another research domain that addresses the perceptive visual quality of 3D objects. The motivation is the acceleration of photorealistic rendering algorithms by avoiding computations that do not bring any perceptually significant improvement on the final result. Bolin and Meyer [25] use a simplified Sarnoff Visual Discrimination Model to speed-up rendering techniques based on sampling. The visual and spatio-temporal sensitivities have been analyzed by Myszkowski et al. [94] to create a perceptually driven animation quality metric. Ferwerda et al. [49] propose a sophisticated perceptual metric based on the masking effect of a visual pattern, (e.g. a texture) on geometry distortions. More recently, Williams et al. [132] have developed a view-dependent simplification algorithm based on a simple model of Contrast Sensitivity Function (CSF) that takes into account texture and lighting effects.

Comparing image-based metrics and 3D metrics, Rogowitz et al. [113] as well as Cleju et al. [36] conclude that the Bolin-Meyer metric [25], the mean geometric error and the image mean-squared distances correlate well to human rankings and preferences when applied to simplification schemes. The image-based distances have proved to be better than geometric distances to predict human perception. Rogowitz et al. also discuss the impact of rendering conditions as well as the illumination model on human perception. Their conclusion is that people perceive still images differently than animations of the same models.

Finally two other works tackle the perceived visual distortion for 3D objects. Pan et al. [105] address texture and wireframe resolution perceptions for simplification purposes. Their analysis has led to the development of a perceptual metric assessed by statistical data collected from a 3D quality evaluation experiment. Corsini et al. [38] have studied the perception of artifacts caused by several 3D watermarking schemes. They also propose a new metric based on surface roughness estimation and validate it by psychovisual experiments.

3.4 Conclusion

This Chapter has presented the most common techniques used to render 3D objects on screens. We have highlighted the many factors that influence the perception of a rendered shape. The most used 3D metrics have also been presented as well as their limitations. Finally, we have also introduced the growing research related to perceptual metrics which tend to quantify how much the human eye is sensible to 3D shape modifications.

The next Chapter presents our contribution to this particular field. We study the perception of watermarking attacks on rendered 3D models by using an image-based metric relying on a Mutual Information criterion.