Chapter 9

3D Blind Local Watermarking and Robustness to Resampling and Cropping Attacks

9.1 Introduction

The two precedent contributions of this thesis to 3D blind watermarking cannot deal with the cropping attack. Indeed, Chapter 7 has presented a blind watermarking technique based on mesh spectral decomposition. This scheme is able to resist against RST transforms, noise addition and smoothing. Chapter 8 has presented the detection of robust feature points (the so called *umbilical points*) for the re-synchronization of the spectral watermarking scheme with respect to the resampling attacks. However, since these feature points depend on the scale, the re-synchronization cannot survive to cropping. In this Chapter, we propose the last contribution of this thesis which is a blind scheme which has been designed to be robust against cropping as well as resampling attacks.

As presented in the survey of existing 3D watermarking algorithms (see Chapter 6), in the case of blind watermarking, robustness against cropping can be achieved by indexed watermarking (i.e. embedding the watermark bit as well as its index) and robustness against resampling can be achieved by statistical methods (shape radius histogram embedding) or by using a remeshing based on robust feature points. The issue is that resampling erases indexed embedding and that cropping modifies the center of gravity (leading to a different shape radius histogram) as well as the boun-

ding box size, changing the proposed multiscale analysis for the detection of feature points. These attacks are however supposed to be very common and a blind copyright protection watermark should be robust against them and their combinations.

As illustrated previously, an attack has to be considered only if the distortion it causes is imperceptible. In the case of the cropping this assertion deserves more comments. A cropping can be imperceptible (it can be seen as the creation of one or several small holes in the shape) or perceptible (a significant part of the shape has been deleted). In the case of perceptible croppings some are relevant because the parts left after the cropping still carry (part of) the semantic wealth of the 3D shape (e.g. the head of the Michelangelo's David) while other have no semantical meaning (see Fig. 9.1). However, in practice, robustness against cropping is tested by using a cutting plane normal to and displaced along the x-axis. One possibility to estimate which croppings should be considered is to follow how people segment a shape into different meaningful parts. If the result of a cropping is a part (or several parts) resulting from a perceptual segmentation, we may think this cropping is meaningful.

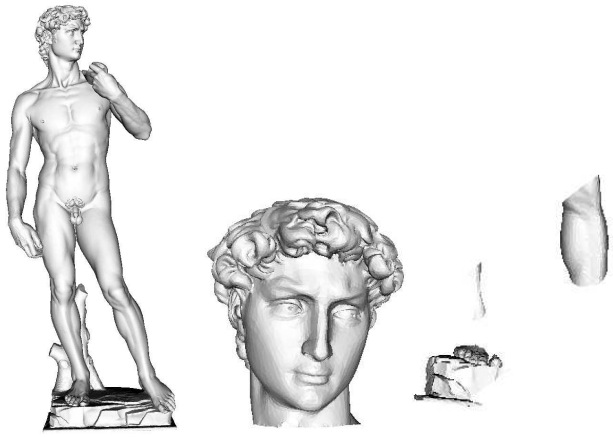


Figure 9.1: *On the left, the entire Michelangelo's David. On the middle, a meaningful cropping attack, the head of the model carries information and should be protected by the watermark embedded in the whole model. On the right, a cropping attack results in a mesh with poor semantic information that should not be protected.*

Exploring the complex way human decompose objects into parts, Hoffman and Singh [60] have proposed the *minimal rule theory*. This theory states that three factors rule this decomposition: the relative sizes of the parts, their *protrusion* and the strength of their boundaries. Several papers [77, 103, 127] have explored this theory to develop perceptual segmentation algorithms. The protrusion of a point is an estimation of how much it is «geodesically» distant to the set of all other points of the shape. Large protrusion values indicate the point is an extremity of the shape (the so called *prongs*), low values indicate the point belongs to the core of the shape (see Fig. 9.2).

Figure 9.2: *On the left, the* Feline *model segmented with core extraction and prong detection (image courtesy of [77]). On the right, the* Dinosaur *model segmented by protrusion conquest (image courtesy of [127]).*

The next Section presents our contribution which is a blind watermarking scheme based on these considerations and able to resist (at some extent) against combined resampling and cropping.

9.2 Overview of the proposed scheme

The proposed scheme can be decomposed in the following steps:

1. Robust prongs automatic detection
2. Detection of a robust neighborhood for each prong
3. Spatial radial watermark embedding (or retrieval) on each neighborhood

The first step is the automatic detection of robust *prong* feature points. These points, unlike the umbilical points presented in the precedent Chapter, do not depend neither on curvature nor on scale. They are defined as local maxima of the *protrusion* function. We show these points are particularly robust against noise addition, resampling as well as cropping (if the cutting plane is not in the neighborhood of the considered prong). We also restrict these points to be placed on the convex hull of the model, this allows to filter out local maxima near a peak of protrusion. These points however, cannot be used for surface partioning since they do not cover the core of the surface. It could be interesting to combine them with umbilical points for improving the scheme proposed in Chapter 8.

The second step relates to the estimation of a robust neighborhood of each prong. These neighborhoods are the charts (a.k.a. patches) that will be watermarked in the third step. These neighborhoods generally do not cover the entire shape. As illustrated on Fig. 9.2, the neighborhoods of prongs do not cover the core of the shape. In the sequel we do not define the neighborhoods following this segmentation technique but we propagate a geodesic wavefront which stops when the resulting neighborhood satisfies some necessary conditions such as a maximal allowed distance as well as a minimal distance determined by a semi-star shaped behavior which will be explained later. These neighborhoods (their prong as well as their geodesic radius) resist against resampling and cropping provided that the cutting plane (or more generally the cutting function) does not intersect the neighborhood. In order to prevent the detection of all prongs and/or corresponding neighborhoods, one has to perform as many croppings as the number of neighborhoods. The resulting shape then loses its semantic value.

Finally, the third step is dedicated to the watermark embedding or retrieval in each prong neighborhood. In order to build a watermarking scheme able to survive to cropping, a local approach is proposed. The surface patches defined by the neighborhoods of each prong are watermarked in the spatial domain. The watermarking technique is inspired by the radius histogram embedding techniques already proposed for star shapes (i.e. the center of gravity of such a shape can be linked to any point of the shape by a straight line without intersecting another point of this shape) in [34, 141]. We complement their approaches to provide a better invariance of the host shape center of gravity after resampling. We show the full

scheme is robust against combined resampling and cropping attacks.

9.3 Robust Prongs Detection

In this Section, we present how to detect robust prong features of a shape. As introduced before, these points are the maxima of the local shape protrusion. This function is an estimation of how much a point on a shape is perceived as a protuberant extremity of the shape or if it belongs to the *core* of the shape. The core of a shape can be seen as the set of points that are the less distant to all other points of the shape.

9.3.1 Protrusion Function

The *protrusion* μ of a point v on a surface S can be estimated by [77]:

$$\mu(v) = \int_{p \in S} g(v,p)^2 dS \tag{9.1}$$

where $g(v,p)$ is the *geodesic distance* from v to p on S. Since we work with a 3D mesh representation M of the surface S, this function is discretized as follows:

$$\mu(v) = \frac{1}{area(M)} \sum_{p_i \in M} g(v,p_i)^2 area(p_i), \tag{9.2}$$

with

$$area(M) = \sum_{p_i \in M} area(p_i), \tag{9.3}$$

where $area(p_i)$ is the area of the neighborhood of point p_i and $area(M)$ is the sum of the area contributions of each point of the mesh M. The weighting areas can be chosen in many different ways with similar results. Most approaches are based on the 1-ring neighborhood (i.e. the set of points connected to the considered point by one and only one edge) [127]. Three possible weighting areas are the total area of the 1-ring A_{1-ring}, the barycentric area $A_{barycentric}$ and the *Voronoi cell area* $A_{Voronoi}$ (see Fig. 9.3). The sum of the barycentric areas is equal to the total area of the shape, while the 1-ring area must be divided by three (since each triangle contributes to three points). Cumulative Voronoi cell areas do not add up to the total area, that is why we define $area(M)$ as the sum of all local areas. For the case of the protrusion function, the choice of point neighborhood does not significantly influence the accuracy of the results but in our tests,

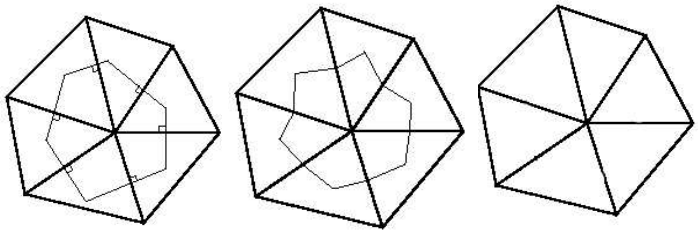


Figure 9.3: *Three different point neighborhoods. From left to right, the Voronoi neighborhood (a.k.a. Voronoi cell), the barycentric neighborhood and the 1-ring neighborhood.*

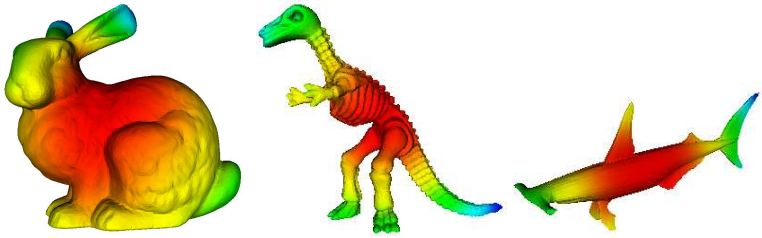


Figure 9.4: *Illustration of the protrusion function on several models. On the left, the* Bunny model*, on the middle, the* Dinosaur *model and on the right, the* Hammerhead *model.*

the barycentric areas have produced the best results. Fig. 9.4 illustrates the protrusion function estimated with 1-ring areas and the fast marching algorithm of Peyre et al [107] for estimating geodesics on several models.

9.3.2 Geodesics

The *geodesic distance* (see Chapter 2) between points p_i and p_j on a surface mesh M is given by the length of the *shortest path* on M linking these points. The Dijkstra algorithm is a very well-known upper-bound estimation of this distance since the shortest path has to follow the edges of the mesh. Other algorithms allow to interpolate on the faces of the mesh: *fast marching algorithms* [80, 107, 108, 119] and *propagating windows algorithms* [81]. If the propagating windows algorithm is the only one providing exact results, its time complexity $O(n^2 log(n))$ (and a memory complexity

of about $O(n^3 log(n))$ is more expensive than the fast marching algorithms which behave like the Dijkstra algorithm in $O(nlog(n))$. Since the computation of the protrusion of all points of the mesh M requires $n\frac{n}{2}$ geodesic distances, the propagating windows are not the best choice. Since the protrusion function can be seen as the mean of the squared geodesic distances to a given point, the already very good approximations provided by fast marching algorithms are sufficient (see [81] for a detailed comparison of the existing geodesics estimation algorithms). The protrusion estimation complexity is then of $O(n^3 log(n))$.

9.3.3 Detecting Local Maxima

Detecting the maxima of the protrusion function has already been proposed for segmentation purposes in [77, 103]. Two major strategies have been investigated. A prong is then defined as a point such as its protrusion is higher than the protrusion of its neighborhood (usually the 1-ring). However, this definition leads to the detection of too many local maxima even in the core of the shape and is very sensitive to remeshing. Katz et al. have proposed to use a variation of *Multi-Dimensional Scaling* (MDS) such that the positions of the points depend on their protrusion value as well as on their connectivity. This strategy allows to unfold the shape in a pose-less representation. It also avoids prong detections in the core of the shape and is particularly interesting for 3D animated shapes. Nevertheless, this MDS transform is very sensitive to resampling as well as to cropping. Prongs and MDS are illustrated on Fig. 9.5.

Here, we propose to define a prong as a point $p \in M$ satisfying the following conditions:

1. p belongs to the convex hull of M
2. $\forall q \in Nk(p), \mu(p) > \mu(q)$
3. p is not on the border of M

where $Nk(p)$ is the set of k geodesically nearest neighbors of p. We thus select geodesic extremities of the shape which are also euclidian extremities of the shape. Moreover, we reject shape boundaries. Indeed, all the points of a border are geodesically far from the other points of the shape and should then be considered as prongs. As we want to resist against cropping attacks, it is important to avoid the borders these attacks create.

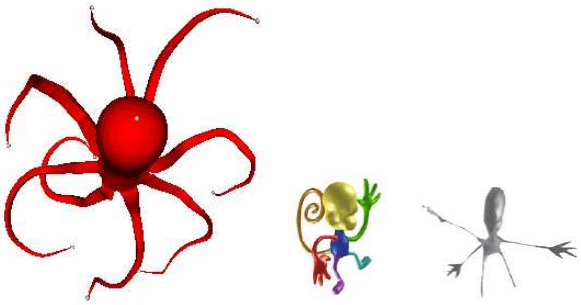


Figure 9.5: *Prongs of the* Octopus *model are represented on the left. Each extremity has been detected by the protrusion maxima. On the right, illustration of the MDS algorithm used by Katz et al. (image courtesy of Katz et al. [77]).*

This definition loses some intuitive prongs that do not belong to the convex hull or the borders but enables to filter undesirable local maxima.

Convex Hull

The *convex hull* (a.k.a *convex envelope*) H of a shape M is defined by the minimal volume convex shape containing M. The bounding box of M is the most well-known example of convex shape containing M but its volume is usually not minimal. Several techniques exist to estimate approximations of the convex hull of a 3D polygonal mesh. We use a method based on sweeping planes. The hull is generated by squeezing the planes towards the input mesh, until the distance between the planes and the mesh is zero. We use an efficient way to compute these distances by relying on a multiresolution octree representation of the mesh. Then, the resulting planes are used to generate a polyhedron (i.e., hull) that is represented by triangles. We define the initial planes as the cell centers of a recursively subdivided bounding octahedron. The resulting hulls are illustrated on Fig. 9.6. In our tests based on thirty models, five subdivisions of the initial octahedron provide a good trade-off between speed and quality of the approximation.

Local maxima

Once we have computed the protrusion of each point as well as a good approximation of the convex hull, we have to state whether the intersec-

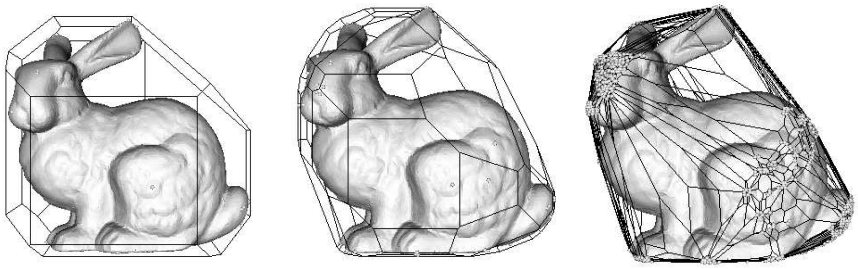


Figure 9.6: *From the left to the right, hulls obtained by squeezing planes to the* Bunny *model with a respectively* 1, 2 *and* 5 *recursively subdivided bounding octahedron. The intersections between the mesh and the hull are represented with small spherical glyphs.*

tions of the convex hull with the shape are prongs or not. According to the definition of prongs, we have to select the k geodesically nearest neighbors of the considered point and to compare their protrusion values. It is clear that this strategy does not enable to find the same neighbors after resampling and cropping attacks. However, this choice gives better results than the 1-ring neighborhood as well as a geodesic circle neighborhood. Indeed, the 1-ring is not representative of the local behavior of the shape and a geodesic circle has to be defined by a radius. This radius should be selected in reference to a given quantity of the shape in order to be insensitive to uniform scaling. Examples of such quantities are the bounding box diagonal or the shape total area which both are sensitive to cropping attacks. The heuristic we propose with 10 geodesically nearest neighbors has survived our benchmarking experiments for remeshing operations which produce a visually equivalent shape and cropping attacks. These robustness results are shown in Section 9.6.

9.3.4 Computational Cost Optimization

The method presented to detect the prongs is still computationally expensive and ruled by the estimation of the protrusion of each point of the shape. The first observation that can be made is that the integral on the shape is very robust against resampling. This means that it is possible to estimate the protrusion by computing the geodesic distances to a given set of well distributed and placed seeds than to the entire set of points of the

shape. The approximated protrusion is then given by:

$$\mu(v) = \sum_{b_i \in B} g(v, b_i) area(b_i) \tag{9.4}$$

where B is the set of seeds b_i and $area(b_i)$ the area these seeds cover on the shape. The set B can be chosen in different ways such as distributing m seeds on the shape by iteratively inserting the geodesically farthest point in the shape or, following Valette et al. [127], by using a simplification algorithm (see Fig. 9.7). The advantage of the simplification algorithm is that points are decimated if their collapse causes a minimal distortion. It follows that the remaining points are not regularly sampled on the shape but their density is higher near important features. The results of the protrusion approximation are however very similar for both strategies. If m seeds are selected, the computational cost is then of $O(mn^2 logn)$. Regarding the area contribution of each seed $area(b_i)$, we select the area of their geodesic Voronoi region. The Voronoi regions V_i of a set of seeds s_i in a shape are the sets of points of the shape that are geodesically closer to their corresponding seed than to the other seeds. To compute the areas of such regions, we cumulate the barycentric areas of all the points belonging to those regions.

Another way to speed up computations is to take advantage of our definition of prongs. Since these points must belong to the convex hull and not to the borders, it is not necessary to compute the protrusion of all the points. Only the intersections of the shape with the convex hull must be estimated as well as their k geodesic nearest neighbors. If r intersections are detected, then the computational cost is reduced to $O((kr)mnlogn)$ in the worst case. In our experiments, the maximum number of points in need of a protrusion value has always been inferior to 5 pc. However, notice when dealing with a sphere, all points intersect the convex hull and the computational cost is not reduced. Indeed all points of a sphere or a plane are prongs. Fortunately, there is no evident need to protect such shapes.

9.4 Robust Local Neighborhoods

Now that the prongs have been detected, we must define their neighborhood. These neighborhoods are patches that do not have to entirely cover the shape. We show later that embedding a watermark in each of

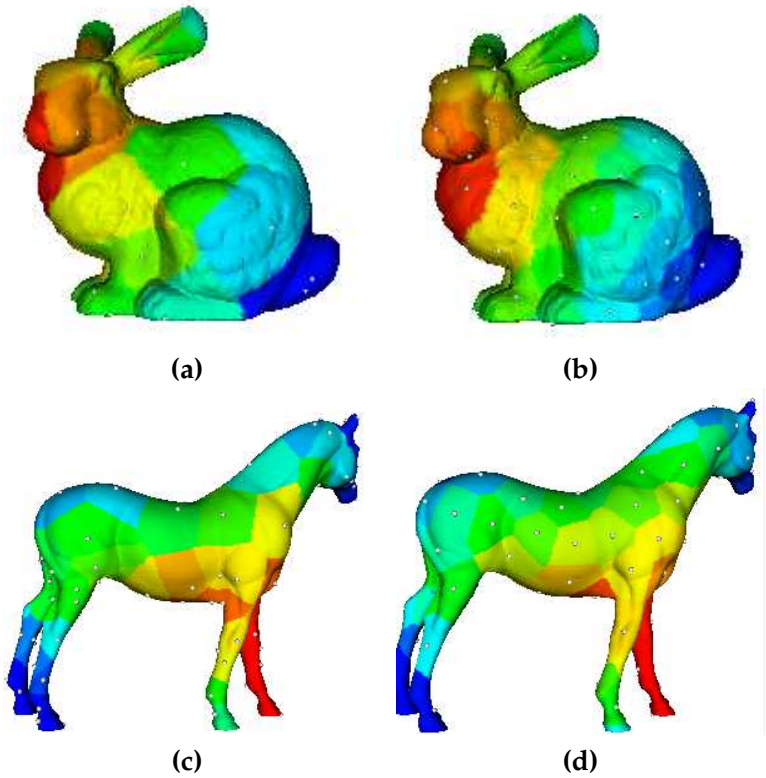


Figure 9.7: *Shape subsampling examples for the optimization of the computational cost related to the estimation of the protrusion function. The idea is to estimate the protrusion of a point by computing its geodesic distance to each of this seeds instead of to all points of the shape. (a)* Bunny *model subsampled through a simplification algorithm, (b)* Bunny *model subsampled through geodesically uniform distribution of* 100 *seeds, (c)* Horse *model subsampled through a simplification algorithm, (d)* Horse *model subsampled through geodesically uniform distribution of* 100 *seeds. The* Voronoi region *of each seed is represented with different colors.*

these patches is sufficient to protect the shape. Since these patches are hosts of the watermarking technique we have selected, their shape must satisfy some conditions:

1. the patch is a geodesic circle centered on the prong,
2. the geodesic radius of the patch cannot be superior to the half of the geodesic distance value R_g to the nearest prong,
3. the line intersecting each point of the patch to its center of gravity cannot intersect with other points of the patch (a.k.a. star-shape condition). The minimal geodesic distance value at which this problem occurs is denoted R_s.

Considering a prong v, let us denote R_{max} the geodesic distance to the most distant point from v. We thus compute for a given prong v, the values R_g and R_s which are set to R_{max} if they do not exist. This occurs when only one prong is detected or when the shape has always a star shape behavior (i.e. the shape is convex). The neighborhood is therefore defined by the surface region covered by the wavefront of radius R which is given by:

$$R = min(R_g, R_s) \leq R_{max}. \tag{9.5}$$

The neighborhood covers the whole shape if only one prong is connected and if the shape is convex. This obviously only happens if the prong detection is incorrect.

In case the nearest prong has been lost after a (cropping) attack, R_{max}, R_s and R_g can be modified and the robustness of the patch is not ensured. Therefore, a limitation of our approach is that we must recover a pair of prongs that are nearest neighbors. However in practice, if the geometry implies $R = R_s$ because of condition 3, the presence of the nearest prong would not be necessary to recover the correct patch. An example of neighborhoods is given in Fig. 9.8.

9.5 Patch Radial Watermarking

This Section is dedicated to the third step of our algorithm. Our watermarking technique is inspired by works of Zafeiriou et al. [141] and Cho et al. [34] for 3D blind spatial robust watermarking. These similar schemes

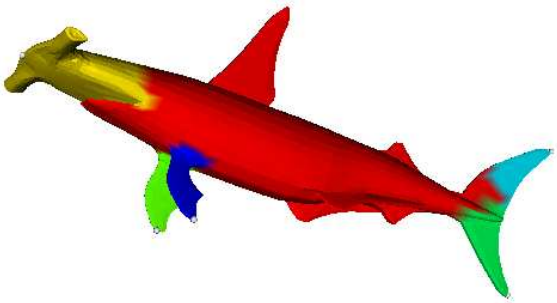


Figure 9.8: *Example of prong neighborhoods for the* Hammerhead *model. Patch neighborhoods which are selected for watermarking have a different color corresponding to their prong. The red region is not watermarked.*

are both based on a representation of the geometry in spherical coordinates with respect to the shape center of gravity. Cho et al. only use the radial component of the geometry to embed a watermark, while Zafeiriou et al. also use the angles θ and ϕ. Their robustness results are similar and very good against most watermarking attacks. The disadvantage of their techniques is that the center of gravity is not correctly estimated after cropping and some resampling attacks and the watermark detection fails. We use the prongs and their corresponding patches to retrieve at least a prong and its local shape. Then the center of gravity of this patch is correctly retrieved as well as the embedded watermark. Moreover, we propose some modifications of the scheme of Cho et al. to better resist against resampling.

9.5.1 Robust Center of Gravity Estimation

First of all, we must define the center of gravity C_g of the shape M. Following [34, 141], this point is simply estimated by the mean of the geo-

metry:

$$C_g(x) = \frac{1}{n} \sum_{p_i \in M} p_i(x), \tag{9.6}$$

$$C_g(y) = \frac{1}{n} \sum_{p_i \in M} p_i(y), \tag{9.7}$$

$$C_g(z) = \frac{1}{n} \sum_{p_i \in M} p_i(z), \tag{9.8}$$

where n is the number of points of M. Since our observations are the same for the coordinates x, y and z, we can restrict the equations to the x coordinate without loss of generality. For a regularly sampled mesh M (i.e. points are uniformly distributed on the shape), this estimation of C_g is correct. However, if the shape is irregularly sampled, the estimation of C_g is biased, and its position is shifted towards the part of the shape which has the highest density of points. We propose to use different weights to estimate the position of C_g:

$$C_g(x) = \frac{1}{area(M)} \sum_{p_i \in M} area(p_i) p_i(x), \tag{9.9}$$

where $area(p_i)$ is the third of the area of the 1-ring neighborhood of p_i and $area(M)$ is given by the sum of the $area(p_i)$. The estimation of C_g is now robust against resampling attacks. In the case of very noisy meshes, local areas as well as the total area of the shape can be significantly increased. Since only imperceptible attacks should be considered, a small number of smoothing iterations enable to get rid of the noise addition influence.

9.5.2 Shape Histogram

The *radius shape histogram* (a.k.a. shape histogram) of a star shape is the histogram of the distance between each point of the shape and its C_g. The distance between a point p and C_g is noted $\rho(p)$ and is given by:

$$\rho(p) = \sqrt{(p(x) - C_g(x))^2 + (p(y) - C_g(y))^2 + (p(z) - C_g(z))^2}. \tag{9.10}$$

The histogram can be represented by regrouping ρ values in bins and is characterized by a minimal radius ρ_{min} and a maximal radius ρ_{max} as illustrated on Fig. 9.9.

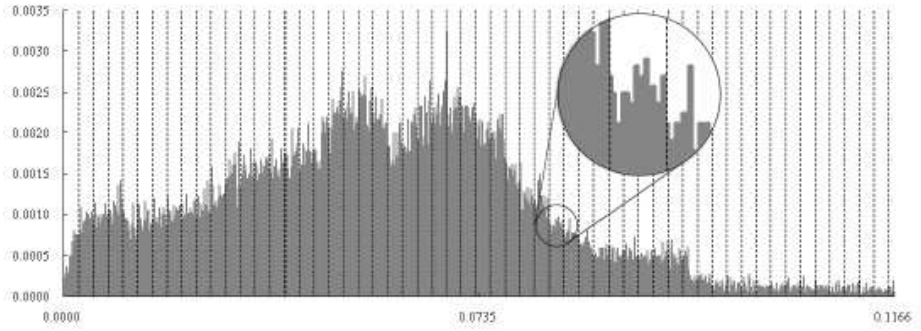


Figure 9.9: *The radius shape histogram of the* Bunny *model. Dashed vertical lines represent the borders of the histogram bins.*

To build the histogram, several choices are possible. Cho et al. propose to simply count the number of points in a given bin. However, we observe the same sensitivity to resampling. Indeed, the sampling of the shape radius distribution is not uniform. Here, we propose to add for each point the ratio between its Voronoi area and the total area of the shape. This approach is much more robust against resampling. However, this strategy can lead to errors if the sampling rate is low (i.e. there are too many large triangles). Osada et al. have proposed to generate a random distribution of points on the mesh [102]. Using a triangle traversal of the mesh, the area of each triangle is computed and stored in an array along with the cumulative area of triangles visited so far. Next, a triangle is selected with probability proportional to its area by generating a random number between 0 and the total cumulative area and performing a binary search on the array of cumulative areas. For each selected triangle with vertices A, B, C, a point is generated on its surface by producing two random numbers, r_1 and r_2, between 0 and 1, and evaluating the following equation:

$$P = (1 - \sqrt{r_1})A + \sqrt{r_1}(1 - r_2)B + \sqrt{r_1}r_2C. \tag{9.11}$$

This random point generation on a triangle is illustrated on Fig. 9.10. In our case, this pseudo-random generation does not enable to directly modify the histogram by changing points of the mesh. In order to build a valid histogram we complement our simple strategy by adding a random point in the triangles covering several bins of the histogram. We compute the Voronoi area of the pseudo-randomly generated point and add its con-

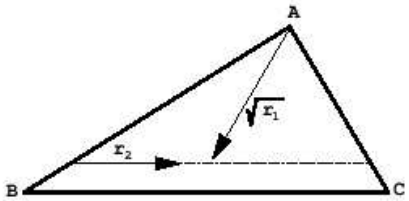


Figure 9.10: *Generating a random point on a triangle surface (image courtesy of Osada et al. [102]).*

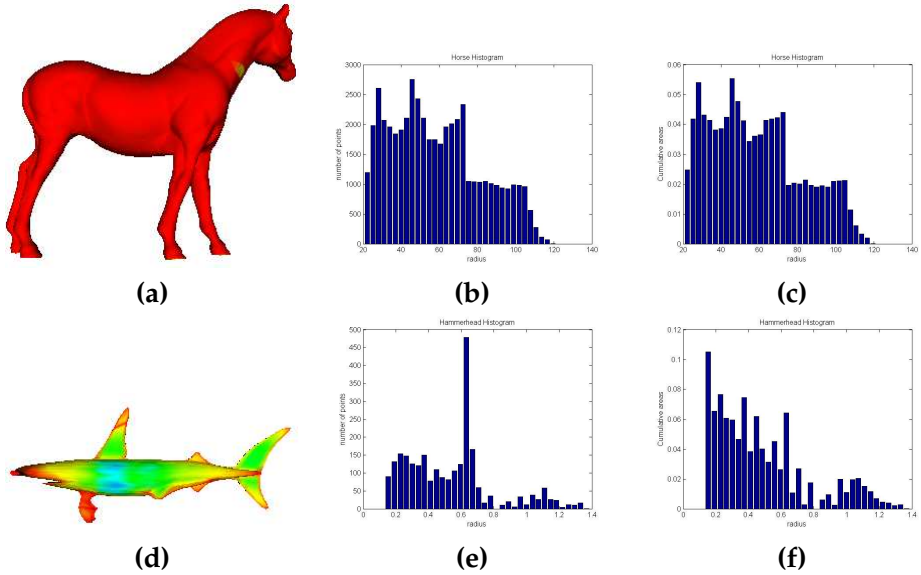
tribution to the histogram in its corresponding bin. In our tests, such case has only appeared for consequent simplification attacks. The differences between shape histograms computed by the method of Cho et al. [34] and our proposed strategy are illustrated on Fig. 9.11. Taking the areas into account allows invariance of the histogram with respect to local point sampling density variations and therefore to resampling attacks.

Some papers propose different shape histograms such as gaussian curvature or mean curvature shape histograms [104]. There is however no robust watermarking technique that enables to modify all points of a shape while directly controlling local curvatures (and therefore the shape histogram). This kind of histograms could be however interesting to investigate in a future work.

9.5.3 Watermark Embedding

The watermarking embedding consists in modifying the distribution of points in each of the $N + 2$ bins of the shape histogram. For that purpose, each histogram is subdivided in two sub-intervals that carry a 0 or a 1 information bit. The mean of the distribution of points in this interval is displaced or not towards the other sub-interval accordingly to the watermark bit to embed. The extremity bins (i.e. those containing ρ_{min} or ρ_{max}) are not modified for imperceptibility reasons as well as for preventing a modification of the bins construction. Likewise Cho et al. [34] we embed $N = 64$ bits.

The displacement of the mean of the distribution of the points in the histogram bin i is performed by first projecting the bin values $\rho(i)$ from the



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

(d) **(e)** **(f)**

Figure 9.11: *The shape histogram construction proposed by Cho et al. [34] is very sensitive to local point sampling density. Our method based on cumulative areas as well as point random generation for large triangles crossing several bins allows to capture the histogram of the shape with a good robustness against resampling. We represent here two models with different point sampling densities with colors from red for high density to blue for low density. (a) The* Horse *model point sampling density shows this model is globally regularly sampled, (b) The* Horse *model histogram based on [34], (c) The* Horse *model histogram computed by our method, (d) The* Hammerhead *model density is not uniform, (e) The* Hammerhead *model histogram computed by [34] shows undesirable peaks near radius bins of high density sampling, (f) The* Hammerhead *model histogram computed by our method is not influenced by the point density.*

interval $[\rho_i, \rho_{i+1}]$ to $[0, 1]$, producing values $\hat{\rho}(i)$. Then, we simply compute a new distribution of values $\rho^k(i)$ with $k > 1$ if the mean of the distribution must be changed towards the upper bound of the bin, and $0 < k < 1$ otherwise. Finally, the new values are re-projected towards the original bin in the interval $[\rho_i, \rho_{i+1}]$. This is illustrated on Fig. 9.12. Since the origi-

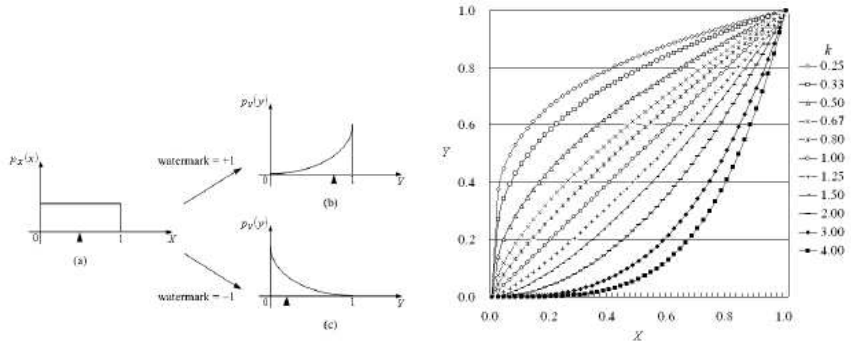


Figure 9.12: *Mean distribution shift. On the left, assuming the bin distribution is continuous and uniform in each bin, the distribution is modified such as its mean is placed in the correct subinterval accordingly to the watermark value to embed. On the right, the distribution mean shift for several values of k.*

nal distribution of the histogram bins are neither continuous nor uniform, the exact power k cannot be analytically computed. An iterative strategy is used to augment or diminish k until the distribution mean is correctly placed.

9.5.4 Watermark Decoding

The watermark is simply read in the histogram bins by determining whether the bin distribution mean is in the O or 1 subinterval. Since the watermark is repeated in each prong neighborhood, we compute the final retrieved watermark by a simple majority rule over the watermark bits retrieved in each neighborhood. A Bit Error Rate (BER) is computed with the embedded watermark to evaluate the correctness of the retrieved watermark.

9.6 Robustness Experimental Results

We have tested the robustness of the prong detection under a wide set of attacks. The robustness of the neighborhoods under the cropping attacks has not been deeply tested. A visual confirmation is provided by Fig. 9.13 This choice is motivated by the fact that it is more interesting to directly test the robustness of the watermarking scheme itself.

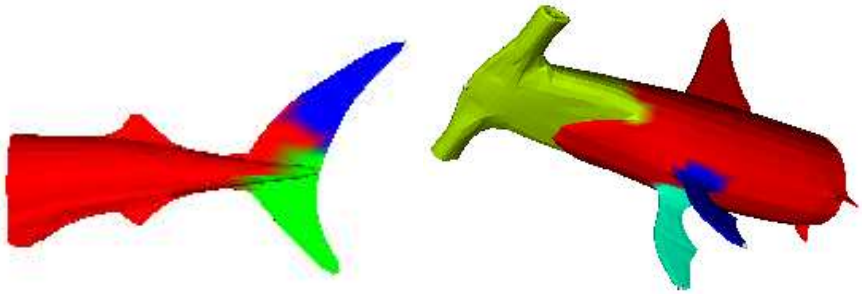


Figure 9.13: *Cropping attack by a vertical plane. The prongs and their neighborhoods (see Fig. 9.8) are well detected on both sides of the cropping.*

9.6.1 Robustness of Prong Detection

The results of the prong robustness benchmark is given in Fig. 9.14. We can see they are more robust than the umbilical points studied in Chapter 8. This can be explained by the fact that these points are detected by the maxima of an integral function while umbilical points are the singularities of the topology of a differential function.

9.6.2 Robustness of the Watermarking Scheme

These preliminary results show the robustness of the whole scheme. The indicated values are those from which higher intensities of the attack lead to a non-null BER. At this stage, the watermark robustness is far from beeing as robust as the prong detection. However, the watermarking scheme can afford some center of gravity displacements which are due to small prong displacements if the sampling density is sufficient. This point needs further research. Some optimizations should be developed to improve the

Models vs. Attacks	noise	smooth.	decim.	RST	subdiv.	cropping
Hammerhead	0.3%	200	25%	OK	1	OK
Bunny	0.4%	100	20%	OK	1	OK
Dinosaur	0.35%	120	30%	OK	1	OK

Figure 9.14: *Robustness of prong detection against usual watermarking attacks: noise, smoothing, decimation, RST, subdivision and cropping. The cropping attack has is made with a vertical plane intersecting the center of gravity of the models. The prongs considered are those on the remaining part of the model. For other attacks, a prong is considered as correctly detected if the projection of this point along its normal towards the original surface is at a distance from the original prong inferior to* 10^{-8} *pc. of the bounding box diagonal, even though, empirically, such precision is not required for the whole scheme.*

Models vs. Attacks	noise	smooth.	decim.	RST	subdiv.	cropping
Hammerhead	0.14%	15	10%	OK	1	OK
Bunny	0.09%	10	15%	OK	1	OK
Dinosaur	0.1%	15	10%	OK	1	OK

Figure 9.15: *Watermark robustness. The cropping attack is the same as for benchmarking prong detection.*

watermarking scheme. The strength of the watermark should be measured and tuned in terms of the distance to the sub-interval limit. We have only modified the distribution mean until it is located in the good sub-interval. This may lead to a gain in robustness at the expense of the very good imperceptibility properties of the watermark embedding. The variations of the maximum and minimum rays of each prong neighborhood histogram should also be investigated to better understand the loss of robustness. However, this blind scheme is very promising as it is the first to resist against cropping and resampling attacks (of small amplitude).

9.7 Conclusion

We have presented the final contribution of this thesis. The results are still preliminary and the whole scheme needs some improvements and optimizations to improve its robustness. Some other tests should be per-

formed to assess how the watermark strength variations will influence imperceptibility. The cropping attack should also be considered with user interaction, defining the most interesting croppings for each models.