Robust Mesh Watermarking

Emil Praun* Hugues Hoppe† Adam Finkelstein*

*Princeton University †Microsoft Research

Abstract

We describe a robust method for watermarking triangle meshes. Watermarking provides a mechanism for copyright protection of digital media by embedding information identifying the owner in the data. The bulk of the research on digital watermarks has focused on media such as images, video, audio, and text. Robust watermarks must be able to survive a variety of "attacks", including resizing, cropping, and filtering. For resilience to such attacks, recent watermarking schemes employ a "spread-spectrum" approach – they transform the document to the frequency domain and perturb the coefficients of the perceptually most significant basis functions. We extend this spread-spectrum approach to work for the robust watermarking of arbitrary triangle meshes.

Generalizing spread spectrum techniques to surfaces presents two major challenges. First, arbitrary surfaces lack a natural parametrization for frequency-based decomposition. Our solution is to construct a set of scalar basis function over the mesh vertices using multiresolution analysis. The watermark perturbs vertices along the direction of the surface normal, weighted by the basis functions. The second challenge is that simplification and other attacks may modify the connectivity of the mesh. We use an optimization technique to resample an attacked mesh using the original mesh connectivity. Results show that our watermarks are resistant to common mesh operations such as translation, rotation, scaling, cropping, smoothing, simplification, and resampling, as well as malicious attacks such as the insertion of noise, modification of low-order bits, or even insertion of other watermarks.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Surface Representations.

Keywords: copyright protection, steganography.

1 Introduction

We describe a robust watermarking scheme suitable for proving ownership claims on triangle meshes representing surfaces in 3D. The explosive growth of the Web has led to a blossoming interest in the electronic publication of various media. Unfortunately, many owners of digital materials, such as images, video, audio, text, and 3D models, are reluctant to distribute their documents on the Web or other networked environment, because the ease of duplicating digital material facilitates copyright violation. Digital watermarks provide a mechanism for copyright protection of digital media by allowing people to permanently mark their documents and thereby prove claims of authenticity or ownership.

The bulk of the research into digital watermarks has focused on media such as images, video, audio, and text, because these media dominate the proprietary material distributed on the Web. As of the writing of this paper, there are eight commercial Web sites offering watermarking technologies or services for images, video, or audio. In contrast, the problem of watermarking 3D models has received less attention from researchers, in part because the technology that has emerged for watermarking images, video, and audio cannot be easily adapted to work for arbitrary surfaces.

1.1 Background

The field of *steganography* addresses the problem of hiding information within digital documents. The information, called the embedded object, is inserted into the original document, called the cover object, to produce a stego object. The term *watermarking* loosely refers to the use of steganography in the application areas of ownership assertion, authentication, content labeling, content protection, and distribution channel tracing [1, 4, 13, 15, 21]. Note that no single watermarking scheme is suitable for all of these applications, as some of their goals are incompatible. We limit the scope of this paper to the use of watermarking triangle meshes for ownership assertion.

To be effective for ownership claims, the watermarking scheme must minimize the probability of *false-positive* results – asserting incorrectly that a document is watermarked when it is not. To decrease the probability of false-positive claims, the watermark is usually encoded in the document using a vector of coefficients. This vector is compared to that observed in the suspect document, and the ownership claim is based on their statistical correlation.

The probability of *false-negative* results, that of failing to detect a watermarked document, is of lesser importance, and is more difficult to analyze since it depends on the type and magnitude of attacks. Such attacks may include inadvertent alterations like compression, blurring, sharpening, cropping, scaling, darkening, and format conversion, but may also include malicious operations like intentional addition of noise, modification of low-order bits, or even insertion of other watermarks.

According to their resilience, watermarks can be *fragile* or *robust*. Fragile watermarks are used for authentication and for localization of modifications. Like digital signatures, their goal is to detect the slightest change to the document. In contrast, robust watermarks are designed to survive (remain detectable) through most attacks. While an attack of sufficient magnitude could erase the watermark, the hope is that an attacker would have to significantly degrade the document in order to destroy the watermark (i.e. achieve false-negative results).

Watermarks can be *perceptible* or *imperceptible*, depending on whether they are directly detectable by the human senses. Perceptible watermarks are often used to display copyright notices or to lower the commercial value of public or preview documents. In contrast, imperceptible watermarks can only be detected using a computational algorithm, which may or may not require the original unwatermarked document. Typically, imperceptible watermarks are advantageous because they are more robust to malicious attacks.

http://www.cs.princeton.edu/gfx/proj/meshwm

A particularly mischievous approach to defeating ownership claims is for an attacker to manufacture a counterfeit original and to allege that the true original document contains the *attacker's watermark*. This scenario can be prevented by requiring the watermarking scheme to be *non-invertible*: forcing the watermark to be a non-invertible function of the original document [5].

Another application of watermarking is tracing distribution channels. In this case, the document is provided to each recipient with a distinct watermark, referred to as a *fingerprint*, which encodes enough data bits to uniquely identify the recipient. Unfortunately, such an approach is susceptible to collusion attacks, in which several recipients compare their fingerprinted documents to produce a new, seemingly unwatermarked document [22].

The method we present is a robust, imperceptible, non-invertible watermarking scheme designed to serve ownership claims.

1.2 Modus Operandi

Here is a typical scenario for creation and detection of a watermark:

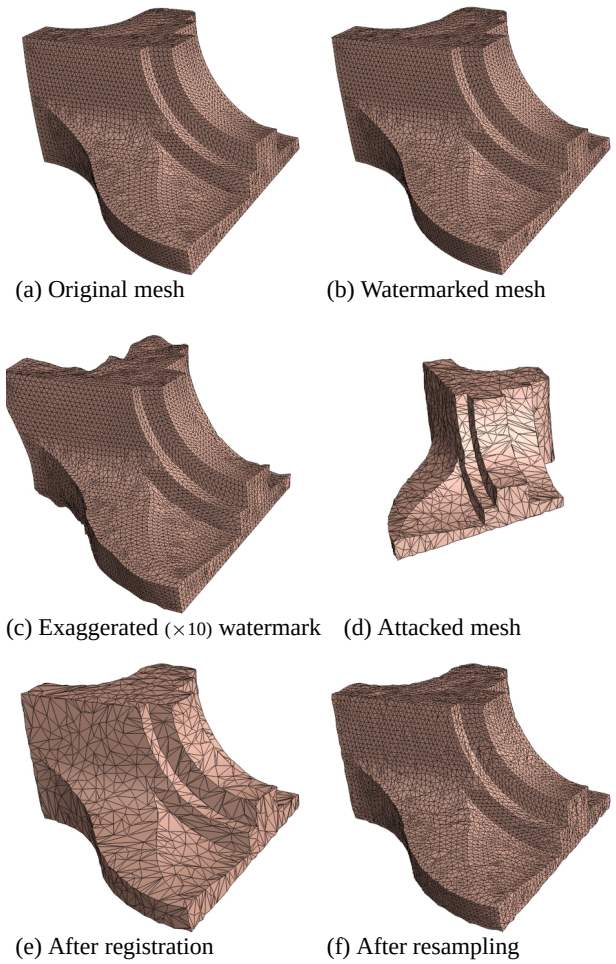
1. The owner, Alice, starts with the original mesh. (Figure 1a)
2. Alice embeds a watermark into the original mesh, creating the watermarked mesh. (Figure 1b)
3. Alice hides the original mesh and the watermark information in a safe place, and publishes the watermarked mesh.
4. An attacker, Bob, takes a copy of the watermarked mesh and alters it – either inadvertently or deliberately attacking the watermark – thereby creating the suspect mesh. (Figure 1d)
5. Bob publishes the suspect mesh, claiming it to be his own. Alice obtains a copy of the suspect mesh, believing it is hers.
6. Alice compares (Figure 1f) the suspect mesh to the original mesh in order to extract a suspect watermark. Alice demonstrates that the suspect watermark and the original (secret) watermark are essentially identical, proving that Bob's model is derived from hers.
7. Bob pays Alice a lot of money.

1.3 Previous watermarking methods

In this section we describe previous watermarking efforts for other media, followed by related work specifically in the area of meshes.

Image, audio and video watermarking. Early watermarking techniques for images and sound used the least significant bits of the document to encode the watermark [21]. Somewhat more robust techniques encode the watermark in the differences of numerous pairs of pixel values [14].

To date, the most robust watermarking schemes for images, video, and sound are based on the *spread-spectrum* method of Cox *et al.* [4], which embeds the watermark in the most perceptually salient features of the data. More precisely, their scheme transforms the data to the frequency domain using a discrete cosine transform (DCT) or wavelet transform, and identifies the largest coefficients, which correspond to the basis functions with the most "energy" in the data. A randomly chosen watermark $w = \{w_1 \dots w_m\}$ is inserted by scaling the m largest coefficients by small perturbations $(1 + \alpha w_i)$. Given a suspect document, an extracted watermark w^* is computed as the difference on the same set of frequency coefficients between the suspect data and the original unwatermarked data. The watermark is declared to be present based on the statistical correlation of w^* and w. The robustness of this scheme derives from hiding the watemark in many different frequencies, and from targeting those frequencies with the most energy. Podilchuk and Zeng [17] achieve greater robustness by employing visual models based on JPEG and wavelet image compression to focus the watermark in regions of high perceptual impact. Our method also relies on a compression scheme to identify significant features in the model.



(a) Original mesh (b) Watermarked mesh

(c) Exaggerated $(\times 10)$ watermark (d) Attacked mesh

(e) After registration (f) After resampling

Figure 1: Given an arbitrary triangle mesh, our robust digital watermarking scheme applies small perturbations to the mesh geometry along the surface normal direction. To detect the presence of the watermark, we register the attacked mesh and project it onto the original mesh using a resampling process.

Mesh watermarking. While watermarking image, video, and audio has been the subject of much research (and a number of commercial systems), only recently have a few papers addressed watermarking 3D models.

Yeung and Yeo [25] present a scheme for fragile watermarking. They slightly perturb the vertices such that a certain hash function of each vertex's coordinates matches another hash function applied to the centroid of its neighboring vertices. Note that the goals of fragile watermarking are different from those of robust watermarking, and accordingly, successful approaches to these two problems have little in common.

Ohbuchi *et al.* [15] introduce several schemes for watermarking polygonal models. One scheme embeds information using groups of four adjacent triangles: they perturb the vertex coordinates to obtain certain desired values for ratios of edge lengths in the group or for ratios of triangle height over triangle base. Another scheme they propose uses ratios of tetrahedra volumes. The tetrahedra are formed by the three vertices of each face and a common apex vertex that is computed by averaging a few fixed mesh vertices. Finally they propose a way of visually embedding information by subdividing some triangles of the mesh so as to produce recognizable patterns in the wireframe rendering of the model.

These schemes provide high steganographic bandwidth and are therefore useful for model annotation, and for carrying ownership information when faced with benevolent users. However, they are not robust against many of the attacks addressed here, particularly simplification, remeshing, and the addition of noise.

Our strategy is perhaps most similar to that of Benedens [1], who also embeds watermark information in surface geometry. His discussion addresses many of the challenges and properties of *any* system using surface geometry to embed information in a 3D model. Benedens maps surface normals onto the unit sphere, and then subtly alters groups of similar normals in order to embed the individual bits of the watermark sequence. However, he demonstrates robustness only with simplification attacks. His results seem roughly comparable to some of the tests we describe in Section 6 (although it is very difficult to compare quantitatively since even if we used the same models there is a subjective aspect to choosing the strength of an "invisible" watermark.)

Like Ohbuchi *et al.* and Benedens, we address the problem of robust watermarking of polygonal meshes. Our approach differs in that it is based on the principles of spread-spectrum watermarking as previously developed for images, sound, and video. The spread-spectrum method, which embeds the watermark at multiple scales into the perceptually salient features of the model, is robust against a much broader range of attacks (see Section 6) than are reported by previous efforts.

2 Our approach

Surface models come in a variety of representations, such as meshes, B-spline patches, subdivision surfaces, implicit surfaces. We opt to perform the watermarking process on meshes, because they are considered the "lowest common denominator" of surface representations – it is easy to convert other representations to meshes.

There are a number of obvious techniques for embedding information within meshes. For example, one can use comments inside the file containing the mesh, or permute the order of vertices, the order of faces, and even the order of vertices within faces. However, such information is easily lost, even during innocent mesh processing operations.

As in the previous work on images, robust watermarking requires that the watermark be embedded deep within the content data, in this case the mesh geometry. Given a mesh with vertex positions $\mathbf{v} = (\mathbf{v}_1 \ldots \mathbf{v}_n)^{\mathrm{T}}$ and arbitrary connectivity (Figure 1a), we propose to embed a random watermark $w = (w_1 \ldots w_m)^{\mathrm{T}}$ by letting each coefficient w_i induce small displacements on a subset of the vertices (Figure 1b-c). These displacements are achieved by generalizing the spread-spectrum approach of Cox *et al.* [4] described briefly in Section 1.3.

Generalizing the spread-spectrum approach to the case of arbitrary triangle meshes presents two major challenges. The first is that arbitrary meshes lack a natural parametrization for frequency-based decomposition. Fortunately, recent advances in analysis of meshes have led to multiresolution surface representations that share similar properties to traditional wavelet transforms [6, 8, 10, 11]. Moreover, many of these constructions (particularly those that focus on approximation or compression) automatically identify significant features in the surface. We develop a technique derived from *progressive meshes* [8] to construct a multiresolution set of scalar basis function $\Phi = (\phi^1 \ldots \phi^m)$ over the mesh (Section 3). The watermarking scheme should perturb vertices without changing the mesh connectivity. Therefore we define the basis functions on the original set of vertices in the mesh, instead of on a resampled set as in other multiresolution schemes [6, 11]. Section 4 describes how the basis functions Φ are used to insert and extract the watermark w in a given mesh.

The other challenge in watermarking arbitrary meshes is that the attacker may modify not only the geometry coefficients (the vertex positions), but also the structure of the vertex sampling itself. By comparison, in image watermarking, the image may be misregistered through a *similarity transform* (translation, rotation, and uniform scaling), but the image format always consists of a Cartesian grid sampling. With meshes, besides misregistration, the mesh connectivity may have been modified by the attacker. To address this challenge, we develop an optimization technique to resample the attacked mesh using the original mesh connectivity (Figure 1d-f), as described in Section 5.

Finally, Section 6 shows the resilience of the watermark under a variety of attacks.

Thus, the contributions of this paper can be summarized as follows. First, we provide a scheme for constructing a set of scalar basis functions over the mesh vertices. We then adapt the spread-spectrum principles used in image watermarking to embed information into the basis functions corresponding to perceptually significant features of the model. Finally we provide a method for resampling a suspect mesh in order to obtain a mesh with the same geometry but with a given connectivity. These contributions combine to form a robust scheme suitable for watermarking 3D models.

3 Surface basis functions

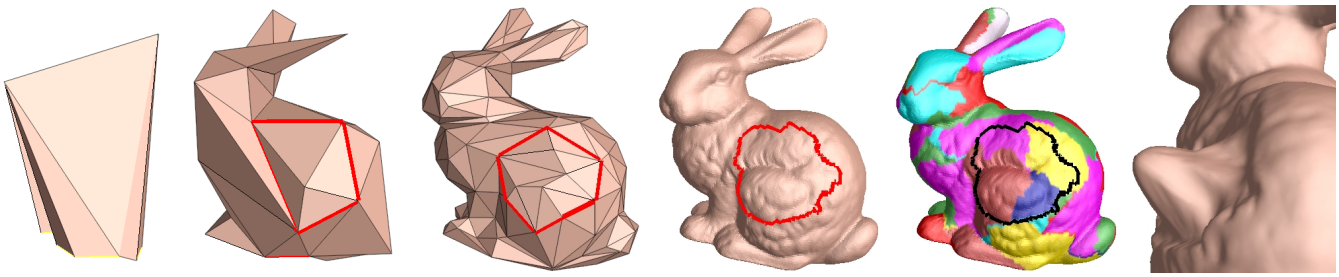
For each coefficient w_i of the watermark, we construct a scalar basis function ϕ_j^i over the mesh vertices v_j, and associate with it a global displacement direction $\mathbf{d}_i$. During the watermarking process (Section 4), the effect of the watermark coefficient w_i is to perturb each vertex v_j by a vector proportional to $w_i \phi_j^i \mathbf{d}_i$. In this section, we describe the construction of these basis functions $\Phi = (\phi^1 \ldots \phi^m)$.

For the watermarking scheme to be robust, the basis functions Φ should correspond to large, perceptually significant features of the model. Recall that in earlier image watermarking work, these were obtained as the DCT basis functions with the largest amplitude. We also need information about how much change can be inserted using a given basis function without perceptibly degrading the model. This information was provided by the magnitude of the selected DCT coefficients.

Our approach is to convert the original triangle mesh into a multiresolution format, consisting of a coarse base mesh and a sequence of refinement operations. We identify the m refinement operations that cause the greatest geometric change to the model. For each of these m refinements, we define a scalar basis function over its corresponding neighborhood in the original mesh. These m basis functions form Φ, and are used to insert the watermark. The same basis functions Φ, computed on the original mesh, are later used for the extraction of the watermark (see Section 4). We now describe in more detail the steps for creating ϕ^j.

Our first step is to simplify the given mesh through a sequence of *restricted edge collapse* operations. Each operation collapses an edge to one of its endpoints [9, 18]. The edge collapses are chosen deterministically with the goal of preserving the geometric shape of the original mesh [8]. Recording this simplification sequence gives rise to a *progressive mesh* (PM) representation [8], which encodes the mesh as a coarse base mesh together with a sequence of *vertex split* operations (Figure 2a-d).

We measure the geometric "magnitude" h of a vertex split operation as follows. First, we predict the position of the newly introduced vertex using the centroid of its immediate neighbors. Next, we compute a surface normal based on these neighbors. Finally, h is computed as the dot product between the surface normal and the difference between the actual and predicted positions. Intuitively, h measures the distance between the new vertex and the coarser mesh.



(a) Base mesh (b) 1-ring area (c) Some refinement (d) Finest mesh (e) 50 basis functions (f) Sombrero $\times 200$

Figure 2: Progressive mesh representation, tracking of a basis function boundary through successive vertex splits, 50 overlapping basis functions, and the sombrero basis function (exaggerated by a factor of 200) applied to the original mesh.

We select the m vertex splits with the largest geometric magnitude h, and proceed to construct their associated basis functions ϕ^i. In addition, the magnitude h_i is used later to scale the contribution of the watermark coefficient w_i. Because we use restricted edge collapses, each vertex split i of a vertex c_i is naturally associated with a neighborhood in the original mesh. The neighborhood is taken to be the ring of edges about the collapsed vertex (Figure 2b), and these edges are tracked through the subsequent vertex split refinements. In particular, each edge of a mesh is naturally mapped onto one or two edges by a vertex split operation. After all refinements are applied, the edges define the boundary B_i of the neighborhood in the original mesh (Figure 2d). Figure 2e shows the support neighborhoods of 50 overlapping basis functions, drawn from coarse to fine.

In each neighborhood we construct a basis function by mapping a radially symmetric function to this region. To begin, we define a "radius" function r_j^i on the vertices v_j such that it is 0 at the center vertex c_i, is 1 on and outside the boundary B_i, and varies linearly in between. More precisely, for vertices v_j in the neighborhood:

$$r_j^i = \frac{d(v_j, \{c_i\})}{d(v_j, \{c_i\}) + d(v_j, B_i)}$$

where $d(v, S)$ is the length of the shortest path between v and any vertex in the set S.

The distances $d(v_j, \{c_i\})$ and $d(v_j, B_i)$ are computed using two instances of Dijkstra's algorithm on the edges of the mesh graph. The cost of each edge equals its length, and the search is constrained within the interior of the boundary B_i. For notational simplicity we shall henceforth refer to r_j^i as r.

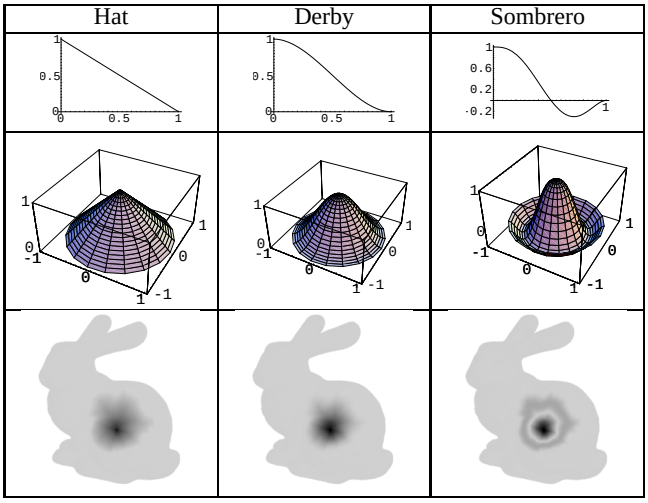


Figure 3: Basis functions

There are two main considerations to take into account when designing basis functions: the changes induced on the mesh should be unnoticeable to a human observer, and the functions should not result in a translation or scale bias when we register the model.

We explore the use of three types of basis functions. Section 6 compares their effectiveness.

Hat. The linear mapping $\phi_j^i = 1 - r$ creates the usual *hat function*.

Derby. We obtain smoother-looking functions using higher degree polynomials. In considering the goal of imperceptibility, we note that since the polygonal mesh itself has only C^0 continuity, we do not really need C^∞ basis functions. However, the human eye is very good at picking up derivative discontinuities, so basis functions that do not have C^1 continuity are easily noticed when rendering the model flat-shaded. Derivative discontinuities can appear at the apex and at the boundary of the hat function. We eliminate these discontinuities using a third degree polynomial: $\phi_j^i = 2r^3 - 3r^2 + 1$. This basis function, applied on a disc, looks like a smooth bump, or colonial ("derby") hat.

Sombrero. We would like a function that does not induce a translation bias during mesh registration, so we now add the constraint that the area integral over the unit disc $r \leq 1$ be zero. This results in a sombrero-like function with a somewhat narrow middle. When this function is discretized on a mesh with few faces, the center is visibly pointy. We smooth the function with the additional constraint of zero second derivative at the apex. The resulting polynomial is $\phi_j^i = -21r^5 + 45r^4 - 25r^3 + 1$.

In our watermarking application, we only construct basis functions for vertex splits with the largest magnitudes h. However, if we were to construct basis functions associated with all vertex splits, these, together with the vertices of the base mesh, would form a basis for scalar functions defined on all vertices. A sketch of the proof goes as follows. If one considers the basis functions in reverse order of the vertex splits, each basis function includes a vertex not referenced by any of the previous ones. Therefore, they are linearly independent. Also, the number of vertex splits added to the number of vertices of the base mesh equals the number of original vertices, thus the basis functions also form a complete set.

4 Watermarking process

We now use the basis functions to insert the watermark vector into the mesh and to later extract it.

First we generate the watermark vector $w = (w_1 \ldots w_m)^T$. Its coefficients w_i are real numbers sampled from a Gaussian distribution with zero mean and variance 1. To make the watermark a non-invertible function of the original document (in order to prevent false ownership claims [5]), the original document (possibly concatenated with some extra information like serial number, date,

owner's name, etc) is passed through a cryptographic hash function (like MD5, or SHA-1 [20]). The result is used to seed a cryptographic random number generator, that produces the watermark with the required length.

Insertion. The watermark is inserted as follows. Each basis function is multiplied by a coefficient, and added to the 3D coordinates of the mesh vertices. Each basis function i has a scalar effect ϕ_j^i at each vertex j and a global displacement direction $\mathbf{d}_i$. We express this process as a matrix multiplication:

For each of the three spatial coordinates X, Y, and Z:

$$\begin{bmatrix} \\ v'_{\mathrm{x}} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ v_{\mathrm{x}} \\ \\ \end{bmatrix} + \epsilon * \begin{bmatrix} \\ \Phi \\ \\ \end{bmatrix} * \begin{bmatrix} h_1 d_{1\mathrm{x}} & & 0 \\ & \ddots & \\ 0 & & h_m d_{m\mathrm{x}} \end{bmatrix} * \begin{bmatrix} \\ w \\ \\ \end{bmatrix}$$

where v'_{x} are the X coordinates of watermarked mesh vertices,
v_{x} are the X coordinates of original vertices $\mathbf{v}$,
ϵ is a user-provided global parameter that scales the energy of the watermark,
Φ is an $n \times m$ matrix with the scalar functions ϕ^i as columns,
hd_{x} is an $m \times m$ diagonal matrix whose entries are the X components of the displacement directions $\mathbf{d}_i$ scaled by the basis function heights h_i, and
w is the watermark.

By concatenating the rows of the matrices and vectors corresponding to the three coordinate components (X,Y,Z) we can express the insertion process as a single equation

$$\mathbf{v}' = \mathbf{v} + \mathbf{B} * w \, .$$

The original document $\mathbf{v}$, along with the watermark w are stored and kept secret, and the watermarked document $\mathbf{v}'$ is published.

Extraction. Before we can check for watermark presence in a suspect mesh, we have to both bring it into the same coordinate frame as the original, and resample it in order to produce a mesh with the same connectivity as the original (see Section 5). By taking the difference between the 3D coordinates of the vertices of this resampled mesh and those of the original, we accumulate a vector of 3D residuals. Next, we extract a watermark from these residuals by solving the sparse linear least squares system

$$\mathbf{B}\, w^* = (\mathbf{v}^* - \mathbf{v})$$

where w^* is the extracted watermark,
$\mathbf{v}^*$ are the vertex coordinates of the resampled attacked mesh
$\mathbf{v}$ are the vertex coordinates of the original mesh.

Analysis. We compare the inserted and extracted watermarks using a statistical analysis. First, like Cox *et al.*[4] we filter the extracted coefficients. Since they are expected to have a normal distribution with mean 0 and deviation 1, we discard coefficients whose absolute value exceeds a given threshold.[1] We then compute the linear correlation [19] between the remaining coefficients and their corresponding values in the inserted watermark:

$$\rho = \frac{\sum_i (w_i^* - \overline{w^*})(w_i - \overline{w})}{\sqrt{\sum_i (w_i^* - \overline{w^*})^2 \times \sum_i (w_i - \overline{w})^2}}$$

[1] We use a threshold of 2.5 for rejecting outliers. Given the normal distribution, we expect to reject about 1.24% of the elements of our watermark vector, which has minimal impact on the analysis.

where $\overline{w}$ denotes the average of the vector elements. The result ρ is a number between -1 and 1.

Finally, we turn the correlation ρ into a probabilistic answer using a standard statistical analysis. We compute the probability P_{fp} that the correlation of w^* with a randomly generated watermark would be as high as the observed ρ, using Student's t-test [19].

If a yes/no answer is required, we compare P_{fp} with a given threshold P_{thresh}: we answer *yes* iff $P_{fp} < P_{\mathrm{thresh}}$. Since the argument is probabilistic, we may sometimes be wrong. The risk of declaring a false positive (saying that the watermark is present when in fact it is not) can be computed analytically, and is in fact, P_{fp}. The risk of declaring a false negative (the watermark is there but we fail to identify it) cannot be computed analytically in general, since it depends on the specific attack that was applied. By changing the decision threshold P_{thresh} we can trade off false positives against false negatives. An analysis of this trade-off using real data is provided in the appendix in the electronic version of the paper.

5 Registration and Resampling

Registration. When including a mesh within a graphical scene, it is common to apply a similarity transform: the object is translated, rotated, and scaled uniformly (Figure 1d). While such a transformation is often kept separate from the object, an attacker might fold the transformation directly into the vertex coordinates. In order to extract the watermark, we need to bring the object back to its original location and scale (Figure 1e).

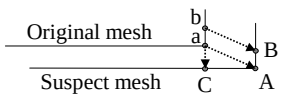
Several methods have been developed for the rigid registration of meshes (e.g. [2, 3, 7]). We use the algorithm by Chen and Medioni [3], but we allow one additional degree of freedom: uniform scaling of the mesh. The algorithm can deal with meshes that represent only parts of the object, which is useful when our watermarked mesh has been cropped during an attack.

The registration algorithm is an iterative process and requires a reasonable initial condition. Sometimes user intervention is required to provide the initial alignment, especially for cropped objects or for objects with strong symmetries.

Note that the alignment process must be performed between the attacked mesh and the original, unwatermarked mesh. The reason is that if we align against the watermarked model, we might falsely increase the apparent presence of the watermark in an unwatermarked model.

Resampling. As part of the attack, the topology of the mesh (number and order of vertices, and the number, order and connectivity of the faces) may have changed (Figure 1d-e). In this case, for the watermark extraction process, we need to obtain a mesh with the topology of the original and the geometry of the attacked object (after alignment). This gives rise to a resampling problem. The basic question we need to answer is: for a given initial mesh vertex $\mathbf{v}_j$, what is the corresponding point $\mathbf{v}_j^*$ on the surface of the attacked mesh? The residuals vector $(\mathbf{v}^* - \mathbf{v})$ provides the input for the extraction process (see Section 4). A very simple answer to the question can be obtained by pairing each original mesh vertex with the closest point on the attacked surface. If the residual is larger than a given threshold, we assume that the object has been cropped, and the vertex has no corresponding point.

Closest point projection may give rise to problems, as shown on the right. In the picture, the watermark followed by the attack moved the surface right and down, but the corner *a* projects to point *C* instead of the corner *A*. Furthermore, point *b* also projects to *C* instead of *B*, causing a degeneracy.



In general, any local technique that only considers one vertex at a time may encounter problems. We cast the resampling problem as the fitting of the original mesh to the suspect mesh while minimally

deforming it. We implement this as an energy-minimization, which we solve using a *conjugate gradient* method [19]. The energy functional has three terms:

$$E(\mathbf{v}^*) = E_{\text{dist}}(\mathbf{v}^*) + c_d * E_{\text{deform}}(\mathbf{v}^*) + c_f * E_{\text{flip}}(\mathbf{v}^*)$$

where

E_{dist} measures the distance between the meshes. Specifically, it is the sum of squared distances between points randomly sampled on the attacked mesh and their projections onto the original mesh.

E_{deform} measures the deformation of the original mesh. A spring is placed on each mesh edge, with rest length equal to the original edge length, as done in [12].

E_{flip} is a term that penalizes surface flipping. The term E_{deform} penalizes surface stretching and contraction, but fails to prevent the surface from "buckling" in areas of high contraction [12]. We let E_{flip} be a sum of penalties over each face. On each face, we compute the dot product of its normal with the the corresponding face normal of the original mesh. If the result is positive, the penalty is zero. Otherwise it is the square of this dot product.

c_d and c_f give relative weights to the terms. Empirically we have found that $c_d = 10^{-3}$ and $c_f = 10^7$ give a reasonable tradeoff.

As a final note, in the extraction process, one can try to skip the registration and/or the resampling steps if the object appears to be in the same location and/or to have the same topology. Multiple attempts at extracting the watermark can be made with or without registration and/or resampling, and the lowest P_{fp} value is kept. As noted by Cox *et al.* [4], since the probability of declaring a false positive when we can choose from several extraction variants is lower than the sum of the individual probabilities, a conservative approach is to divide our proposed false-positive decision threshold by the number of extraction schemes.

6 Results

In this section, we demonstrate that our watermarking scheme is effective in the presence of various real-world attacks. We first present quantitative results for a variety of attacks and then we motivate the choice of parameters used in the tests and describe some typical running times.

Since the watermarking process and the attacks depend on random numbers, there is variability in the results. We therefore ran each test 5 times, using different random seeds, and report the median value.

Battery of attacks. Table 1 shows the detection results for a host of attacks. The second and third columns specify if the registration and/or resampling steps were used as part of watermark detection. Entries with asterisks correspond to the attacked meshes pictured in Figure 4.

Row B (the reorder attack) shows the impact of the resampling process on the detection scheme. Even though the geometry of the watermarked model is not changed by the attack, the recovered false-positive probabilities are nonzero. The reason is that the resampling process is not given any knowledge of the watermarked mesh, in order to prevent any information from the watermark to "leak" into the suspect mesh. For the same reason, the probabilities in row G are also nonzero.

Rows C-E demonstrate the resilience of the watermark under the addition of white noise. The percentage represents the noise amplitude as a fraction of the largest dimension of the object.

Attack	Reg.	Res.	Fandisk	Head	Dragon	Bunny
A. No attack			0	0	0	0
B. Vertex reorder		√	10^{-14}	10^{-15}	10^{-16}	10^{-21}
C. Noise 0.2%			10^{-14}	10^{-12}	10^{-17}	10^{-35}
D. Noise 0.45%			10^{-5}	$\mathbf{10^{-6}}$ *	10^{-9}	10^{-21}
E. Noise 0.7%			10^{-3}	10^{-5}	10^{-5}	10^{-15}
F. Smoothing			$\mathbf{10^{-12}}$ *	10^{-38}	10^{-32}	0
G. Transform	√		10^{-33}	10^{-24}	$\mathbf{10^{-29}}$ *	10^{-29}
H. Simplify 1/2		√	10^{-12}	10^{-9}	$\mathbf{10^{-7}}$ *	10^{-17}
I. Simplify 1/8		√	10^{-11}	10^{-3}	10^{-2}	$\mathbf{10^{-13}}$ *
J. 2nd watermark			10^{-8}	$\mathbf{10^{-7}}$ *	10^{-17}	10^{-6}
K. Crop			0	0	0	**0** *
L. B + C		√	10^{-12}	10^{-6}	10^{-11}	10^{-28}
M. B + G	√	√	10^{-12}	10^{-12}	10^{-15}	10^{-20}
N. C + G	√		10^{-11}	10^{-12}	10^{-15}	10^{-24}
O. G + L	√	√	10^{-22}	10^{-9}	10^{-20}	10^{-18}
P. B + C + G	√	√	10^{-12}	10^{-4}	10^{-14}	10^{-22}
Q. B + G + H	√	√	10^{-13}	10^{-9}	10^{-6}	10^{-17}
R. All (B,C,F,G,H,J,K)	√	√	$\mathbf{10^{-2}}$ *	10^{-2}	10^{-2}	10^{-5}

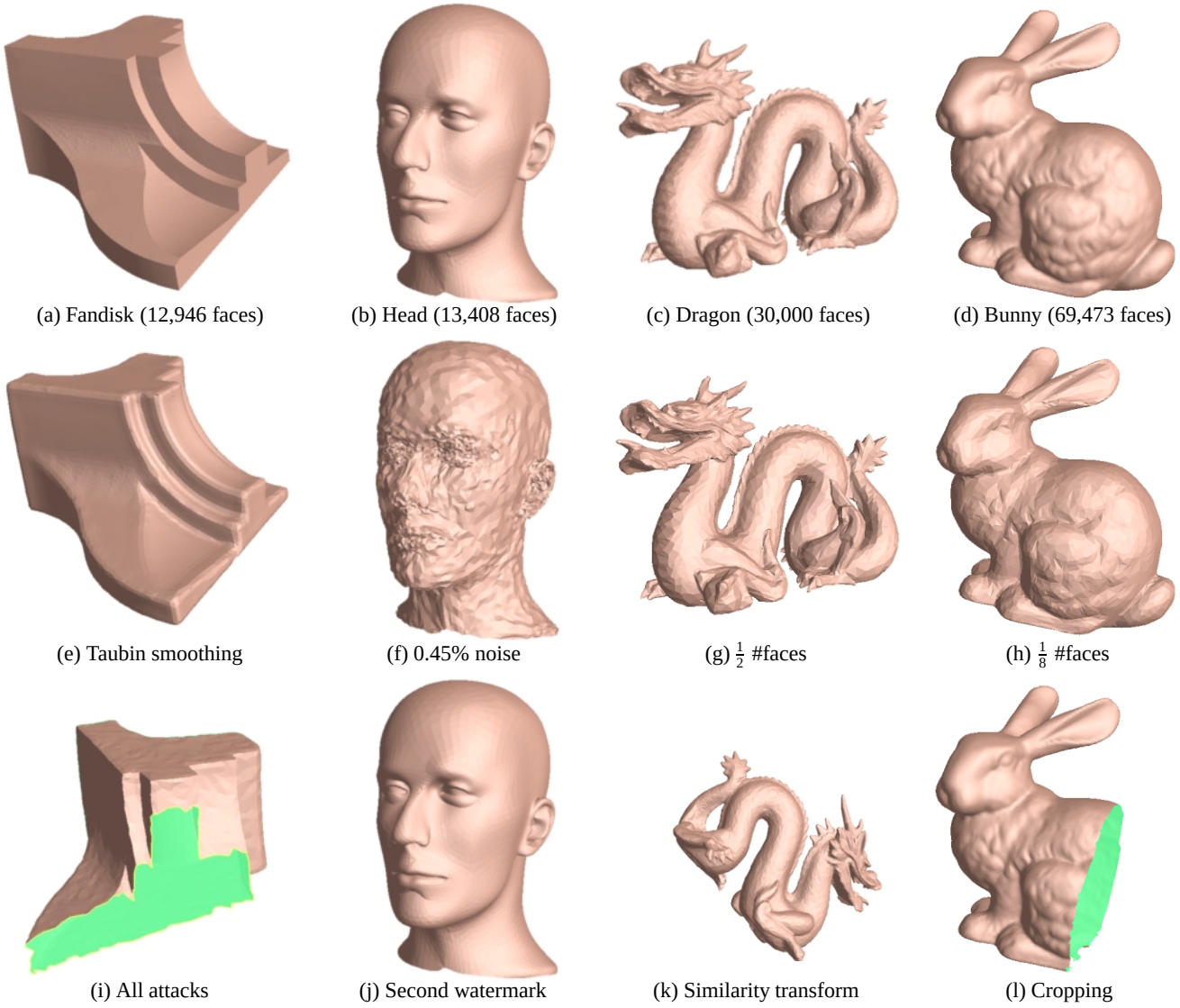
Table 1: Median of 5 tests for various attacks. Entries show the false-positive probability P_{fp}. Entries with asterisks correspond to pictures in Figure 4.

Row F shows the results of applying 10 iterations of the Taubin smoothing filter [23] to the vertex coordinates. The effect can be seen in the rounded edges of Figures 4e and 4i. Since the head and bunny models are already relatively smooth, this attack has little impact on them.

The simplification scheme used for rows H and I is based on full edge collapse operations, which replace a mesh edge with a vertex at an optimized location. For simplification factors less than 1/2, it is likely that few vertices keep their original positions, so this can attack can be considered an instance of "remeshing". We note that the head and the dragon suffer most from this kind of attack, which can be explained by the fact that these models have a higher ratio of detail to number of faces. The watermark coefficients hidden in the fine details of the ears, eyes and lips of the mannequin head and in the head, legs and tail of the dragon are lost during aggressive simplification. Of course, the visual appearance of the models is also degraded by such severe attacks.

Row J addresses to the addition of a second watermark. Because the watermarked geometry is different from the original, the sequence of edge collapses used for inserting the second watermark is different from that used in the initial watermark. By inserting a third and fourth watermark the attacker may further degrade the original watermark, so we believe that a barrage of many watermarks would form an effective attack. However, such an attack would impose changes on the mesh that, if still imperceptible, might have been used to strengthen the original watermark.

The crop attack presented in row K consists of discarding all vertices in the right third of the object's bounding box. As mentioned in Section 5, the registration algorithm can handle incomplete meshes. The resampling step declares any vertex in the optimized mesh to be missing if it lacks a corresponding point on the attacked mesh. During watermark extraction, equations corresponding to missing vertices are deleted from the system. Watermark coefficients that only affect removed vertices (zero columns in the **B** matrix) are also removed. Since missing vertices may cause some columns to become linearly dependent, the remaining system is solved using SVD. We observe that the surviving watermark coefficients are perfectly recovered, so the false-positive probability for this attack is 0.



(a) Fandisk (12,946 faces) (b) Head (13,408 faces) (c) Dragon (30,000 faces) (d) Bunny (69,473 faces)

(e) Taubin smoothing (f) 0.45% noise (g) $\frac{1}{2}$ #faces (h) $\frac{1}{8}$ #faces

(i) All attacks (j) Second watermark (k) Similarity transform (l) Cropping

Figure 4: Watermarked models (top row) and various attacks.

Parameter settings. The test results in this section were obtained using a watermark length of m=50 coefficients, an energy scale factor ϵ=0.01, and the sombrero basis function.

We experimented with different watermark lengths. Ideally, the length of the watermark should be maximized so that an observed correlation value corresponds to a low false-positive probability P_{fp}. However, making the watermark long requires that we include basis functions that correspond to features of low significance, and such features have high frequency components which are most affected by many types of attacks. We find that using more than 100 coefficients yields results that are significantly worse than the ones reported here. A watermark length of 50 coefficients is comparable to the lengths (16 and 50) used by Benedens [1] but much smaller than the length (1000) used by Cox *et al.* [4] when watermarking images. One explanation is that photographs contain much more detail than the relatively smooth surfaces present in our test models. Ultimately, the number of coefficients should be model-specific, based on some "information complexity" of the model, and should be carefully selected to maximize robustness.

We determined the value ϵ=0.01 experimentally, as providing reasonable robustness while still leaving the watermark imperceptible, as demonstrated in Figures 1b and 4. At levels above ϵ=0.03 the watermark became noticeable.

Table 2 compares the three choices of basis function. The letters preceding the attacks are the same as in Table 1. For a noise attack, the three basis functions give similar results. The sombrero outperforms its counterparts under the similarity transform attack, as well as under simplification, which requires a resampling step. We conjecture that accurate resampling expects that the mesh be least deformed from its original shape, and that, of the three basis functions, the sombrero induces the least overall distortion.

Model	Attack	Hat	Derby	Sombrero
Fandisk	C. Noise 0.2%	10^{-13}	**10^{-15}** †	10^{-14}
	E. Noise 0.7%	10^{-2}	10^{-2}	**10^{-3}** †
	G. Similarity transform	10^{-17}	10^{-6}	**10^{-33}** †
	H. Simplify 1/2 # faces	10^{-7}	10^{-5}	**10^{-12}** †
Head	C. Noise 0.2%	**10^{-18}** †	10^{-12}	10^{-12}
	E. Noise 0.7%	**10^{-5}** †	10^{-3}	10^{-5}
	G. Similarity transform	10^{-1}	10^{-3}	**10^{-24}** †
	H. Simplify 1/2 # faces	10^{-2}	10^{-1}	**10^{-9}** †

Table 2: Resilience of the three basis function types. Each entry shows the false-positive probability P_{fp} for the median of 5 tests. Entries with daggers highlight the best result for each attack.

Stage	Fandisk	Head	Dragon	Bunny
Conversion to PM	3	3	9	24
Watermark insertion	0.1	0.1	0.1	0.5
Registration	2	3	6	13
Resampling	15	15	45	120
Watermark extraction	0.2	0.2	0.8	2

Table 3: Typical running times in minutes (MIPS R10000 195MHz) for various stages of the pipeline.

Table 3 shows some typical running times for various steps in the watermark insertion and extraction processes. We have not tried to optimize for execution time in this work.

7 Summary and future work

In this paper, we address the problem of robustly watermarking 3D models. We present a practical method for encoding information in the model geometry by imperceptibly displacing the vertices. To maximize robustness, the watermark is hidden in the perceptually significant features of the model, which are identified using a multiresolution approach.

To detect the watermark in a suspect mesh, we first optionally register this mesh with the original and/or resample it. The watermark is then extracted using a sparse linear least-squares solution. Finally, we compare the inserted and extracted watermarks in order to determine the probability that the suspect model was created independently of the watermarked model.

Our scheme has proven to be robust against a wide variety of attacks, including vertex reordering, addition of noise, similarity transforms, cropping, smoothing, simplification, and insertion of a second watermark.

This project suggests a number of areas for future work:

Fast rejection. Given a suspect model, it would be desirable to more quickly determine that it does not match an owner's set of watermarked models. This would enable an automated agent (such as a web crawler) to search for possible stolen watermarked documents, without having to run the complete, high-accuracy detection scheme.

Other surface representations. We would like to investigate the direct watermarking of other surface representations such as subdivision surfaces, CSG, and Bezier/B-spline patches. The challenge is that such representations often contain fewer coefficients.

Other attacks. It is impossible to anticipate all possible attacks to a 3D model. It is also difficult to assess the degree of damage that a certain attack inflicts upon a model. Some attacks we have not considered thus far include general affine transforms, projective transforms, and free-form deformations. It would be easy to extend the registration algorithm to handle arbitrary affine transforms or even projective transforms. Handling free-form deformations (FFD) is more problematic, however, since these can have an arbitrary number of degrees of freedom. A watermark detection algorithm would have the task of separating vertex motion due to the watermark from vertex motion due to the FFD, without knowledge of either the watermark sequence or the FFD parameters. An alternative would be to create a watermarking scheme that is orthogonal to FFD. We note that to date FFD seems to be the most successful attack against image watermarks [16].

Acknowledgements

This work was supported in part by Microsoft Research and the National Science Foundation. For the use of the 3D models we would like to thank Stanford University, Pratt & Whitney, and Technical Arts Co. Special thanks go to Talal Shamoon and Harold Stone for many helpful discussions.

References

[1] BENEDENS, O. Geometry-based watermarking of 3d models. *IEEE Computer Graphics and Applications* (Jan. 1999), 46–55.

[2] BESL, P. J., AND MCKAY, N. D. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (Feb. 1992), 239–256.

[3] CHEN, Y., AND MEDIONI, G. Object modelling by registration of multiple range images. *Image and Vision Computing 10*, 3 (Apr. 1992), 145–155.

[4] COX, I. J., KILLIAN, J., LEIGHTON, T., AND SHAMOON, T. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing 12*, 6 (1997), 1673–1687.

[5] CRAVER, S., MEMON, N., YEO, B.-L., AND YEUNG, M. M. Resolving rightful ownership with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal on Selected Areas in Communications 16*, 4 (May 1998), 573–586.

[6] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH 95 Conference Proceedings* (Aug. 1995), pp. 173–182.

[7] FAUGERAS, O. D. The representation, recognition, and locating of 3-d objects. *International Journal on Robotic Research 5*, 3 (1986), 27–52.

[8] HOPPE, H. Progressive meshes. In *ACM SIGGRAPH 96 Conference Proceedings* (Aug. 1996), pp. 99–108.

[9] KOBBELT, L., CAMPAGNA, S., AND SEIDEL, H.-P. A general framework for mesh decimation. In *Proceedings of Graphics Interface* (1998).

[10] KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. Interactive multi-resolution modeling on arbitrary meshes. In *ACM SIGGRAPH 98 Conference Proceedings* (July 1998), pp. 105–114.

[11] LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution adaptive parameterization of surfaces. In *ACM SIGGRAPH 98 Conference Proceedings* (July 1998), pp. 95–104.

[12] MAILLOT, J., YAHIA, H., AND VERROUST, A. Interactive texture mapping. In *Computer Graphics (SIGGRAPH 93 Proceedings)*, vol. 27, pp. 27–34.

[13] MEMON, N., AND WONG, P. W. Protecting digital media content. *Communications of the ACM 41*, 7 (July 1998), 35–43.

[14] NIKOLAIDIS, N., AND PITAS, I. Copyright protection of images using robust digital signatures. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (May 1996), pp. 2168–2171.

[15] OHBUCHI, R., MASUDA, H., AND AONO, M. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE Journal on Selected Areas in Communications 16*, 4 (May 1998), 551–559.

[16] PETITCOLAS, F. A., ANDERSON, R. J., AND KUHN, M. G. Attacks on copyright marking systems. In *Second Workshop on Information Hiding* (Apr. 1998).

[17] PODILCHUK, C. I., AND ZENG, W. Image-adaptive watermarking using visual models. *IEEE Journal on Selected Areas in Communications 16*, 4 (May 1998), 525–539.

[18] POPOVIĆ, J., AND HOPPE, H. Progressive simplicial complexes. In *ACM SIGGRAPH 97 Conference Proceedings* (Aug. 1997), pp. 217–224.

[19] PRESS, W. H., TEULKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C*, 2nd ed. Cambridge University Press, 1996.

[20] SCHNEIER, B. *Applied Cryptography: protocols, algorithms, and source code in C*, 2nd ed. John Wiley & Sons, Inc., 1996.

[21] SCHYNDEL, R. V., TIRKEL, A., AND OSBORNE, C. A digital watermark. In *Proceedings of ICIP* (Nov. 1994), IEEE Press, pp. 86–90.

[22] STONE, H. S. Analysis of attacks on image watermarks with randomized coefficients. Tech. rep., NEC Research Institute, May 1996.

[23] TAUBIN, G. A signal processing approach to fair surface design. In *ACM SIGGRAPH 95 Conference Proceedings* (Aug. 1995), pp. 351–358.

[24] WOODS, K. Generating ROC curves for artificial neural networks. *IEEE Transactions on medical imaging 16*, 3 (June 1997), 329–337.

[25] YEUNG, M. M., AND YEO, B.-L. Fragile watermarking of 3d objects. In *International Conference on Image Processing* (1998).

Note: The electronic version of the paper contains an appendix.

Appendix A. Receiver operating characteristic

A standard technique for visualizing the effectiveness of a probabilistic detection mechanism is to plot a *receiver operating characteristic* (ROC) curve (Figure 6). The ROC curve has found a variety of applications ranging from analysis of radar during World War II to medical imaging and evaluation of neural networks [24].

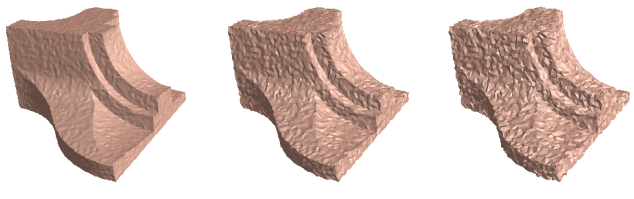
In the case of watermarking, each ROC curve shows the resilience of a watermarked model to a particular attack. The probability of false positives is plotted against the probability of false negatives by varying the decision threshold for declaring the watermark present. The ideal response curve is one that passes as close as possible to the origin on the lower left, where false-positive and false-negative probabilities are simultaneously low.

In practice, we are most concerned about the level of false-positive probability, which is the likelihood of falsely accusing an innocent party of having a watermarked model. For a given level of false-positive probability on the horizontal axis, we can then look up the corresponding false-negative probability value on the vertical axis, which is the likelihood that we will fail to catch a guilty party.

The three response curves of Figure 6 correspond to noise attacks of different amplitudes, equivalent to the C, D and E attacks in Table 1, and shown in Figure 5. In a noise attack, each vertex is translated by a random vector. This attack is representative of routine processing operations such as requantization, digital-to-analog-to-digital attacks, certain format conversions, etc. For the worst of the 3 noise attacks (amplitude 0.7% of the object diameter), the ROC curve shows that if we want to be 99.9% sure not to accuse innocent parties (i.e. false-positive threshold = 0.001), we will fail to identify the watermark 37% of the time. For less severe attacks, we can afford to have more confidence in the accusation for the same level of false-negative results.

For the watermark to be robust, it must survive attacks that do not perceptually degrade the model. As Figure 5 shows, the robustness results are quite good even when the model is clearly damaged.

Each ROC curve is obtained by running 200 tests, in which the mesh is watermarked and then attacked with random noise of fixed amplitude. Each test k uses a different watermark and different noise, and reports a false-positive probability P^k_{fp} (as in Table 1). If the decision threshold for declaring the watermark present were to be exactly P^k_{fp}, then all the tests that returned probability larger than this value would yield false negatives. Thus, we calculate the false-negative probability P^k_{fn} corresponding to P^k_{fp} as the fraction of the 200 tests that have false positive probabilities larger than P^k_{fp}. The ROC curve then simply consists of the 200 points with coordinates (P^k_{fp}, P^k_{fn}).



(a) 0.2% noise (b) 0.45% noise (c) 0.7% noise

Figure 5: Noise attacks on the fandisk model. The percentage represents the ratio between the largest displacement and the largest side of the object's bounding box.

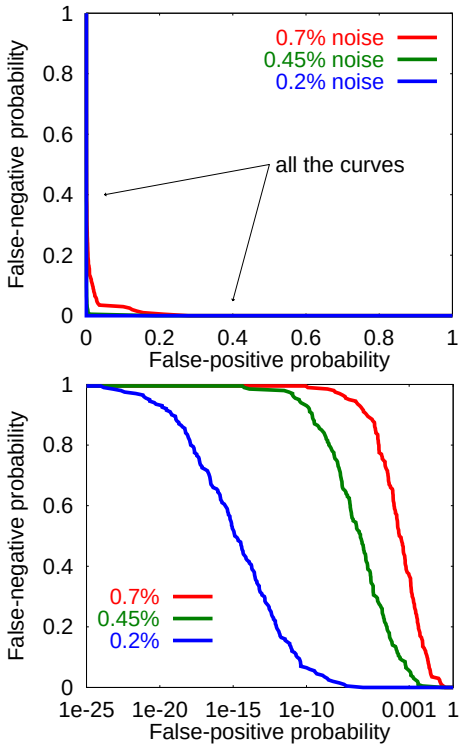


Figure 6: ROC curves for noise attack on the fandisk model. The same data appears in both plots; the bottom one uses a log scale on the X axis. For the worst noise, if we choose a decision threshold of 0.1% false positives, we "lose" the watermark about one third of the time.