EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (*2000*), *Number 3*

Towards Blind Detection of Robust Watermarks in Polygonal Models

Oliver Benedens and Christoph Busch

Department of Security Technology for Graphic and Communication Systems,
Fraunhofer Institute for Computer Graphics, Darmstadt, Germany
email: {benedens,busch}@igd.fhg.de

Abstract

We describe a Digital Watermarking system dedicated for embedding watermarks into 3D polygonal models. The system consists of three watermarking algorithms, one named Vertex Flood Algorithm (VFA) suitable for embedding fragile public readable watermarks with high capacity and offering a way of model authentication, one realizing affine invariant watermarks, named Affine Invariant Embedding (AIE) and a third one, named Normal Bin Encoding (NBE) algorithm, realizing watermarks with robustness against more complex operations, most noticeably polygon reduction. The watermarks generated by these algorithms are stackable. We shortly discuss the implementation of the system, which is realized as a 3D Studio MAX plugin.

1. Introduction

Research on digital watermarking techniques has for a long time focused on the media types images (bitmaps), audio- and video data. In contrast to bitmap-based data, the data types in a variety of fields are vector based, including 3D virtual environments represented by VRML scenes and 2D vector graphics which are parts of PostScript and PDF documents.

The recent publication[1] was the first publication in the field of watermarking 3D polygonal models concerned with the problem of realizing watermarks that withstand remeshing operations in general, polygon reduction in particular. For VRML-scene designers, polygon reduction is an operation frequently performed to achieve pleasant rendering speeds. The central idea to achieve independence of a particular mesh representation was to partition a models face normals into distinct bundles and then to modify the normals by vertex movements, so each bundle (or bin) encodes a value with respect to a particular feature. In [1] the feature used was the mean angular deviation of bin contents to the bin center normal, in [2] we added as a feature the ratio of the number normals within a specific angle to the bin center, this region was called *kernel*, to the number of normals of the bin not within this specific angle.
The watermarking system outlined was of the *forensic* type since it required user assistance in watermark retrieval:

- In retrieval of the watermark, the feature values of the watermarked copy and the original are compared. The original (or a feature vector) has to be accessible at retrieval time.
- The user has to choose the corresponding original model from a database since no automated data recognition and retrieval system may assumed to be in place.
- Since the features are orientation dependent, the watermarked copy has to be reoriented with respect to the original prior to the retrieval process.

In context of secret watermarks, one may accept the burdens associated with a forensic technique, computational costs and required user interaction, if the algorithms offers sufficient robustness for embedding information related to customer tracing or proof of ownership.

1.1. Related Work

Praun et al.[10] presented a forensic technique suitable for embedding secret watermarks. It is driven by multi resolution theory, detail preserving and generally generates smooth meshes. Due to construction of filters applied to the mesh, the changes affect the overall sweep of surfaces. The algorithm requires user assistance in retrieval process, at least for recognizing the original model. Position and orientation of original and watermarked copy must be the same in or-

der to compare feature values. The matching process, which involves removal of affine transformations is currently non-automated.

Most practical applications however will demand a more convenient way, to extract watermarks. Therefore we propose with our approach the realization of a blind detection concept: We extended the NBE algorithm published in [1, 2] for embedding of a *One Bit watermark* which requires no knowledge of a priori data at retrieval time except a secret key. As before, for retrieval, the watermarked copy has to be reoriented with respect to the original but there are possibilities for performing this step in an automated way.

Ohbuchi et al.[8, 9] presented the first algorithm for embedding affine invariant watermarks, named *Tetrahedral Volume Ratio Embedding* (TVR). The affine invariant features used were ratios of volumes of tetrahedrons generated of three "neighbored" triangles along a sequence of triangles. It is restricted to the case of orientable 2-manifolds. Since geometric measures cannot be supposed to be stable if the model undergoes scaling with largely deviating factors or shearing, the algorithm consequently relies on topology for generating/finding the mentioned sequence, for which there are ambiguities in case of non-manifolds. It is furthermore restricted to meshes consisting entirely of triangles, no planar polygons with more than three vertices can be processed. The TVR algorithm offers high capacity. In this paper we propose a method named Affine Invariant Embedding (AIE) which is applicable to the general case of non-manifolds.

The *Triangle Similarity Quadruple Embedding* (TSQ) algorithm as presented by Ohbuchi et al.[8, 9] can be applied for embedding public watermarks (or labels). It can be easily extended to the non-manifold case but is restricted to watermarking of meshes consisting entirely of triangles. For the purpose of authentication of models, it lacks the important property that several watermarks of the TSQ-algorithm cannot "cover" all vertices as we will explain later.
We developed an algorithm for the purpose of embedding public watermarks with optional model authentication called *Vertex Flood Algorithm* (VFA). Our algorithm has the following benefits: If required, e.g. in general case of watermarking point clouds, it operates by taking only vector coordinates into account. Other properties play a role for the purpose of model authentication: Restrictions regarding the maximum allowed vertex displacements can be assured, the algorithm is fragile and can take all vertices of mesh into account.
Boon-Lock and Minerva[3] published the first work concerned with the problem of authenticity and integrity protection for 3D polygonal models. Using their scheme one can verify, in knowledge of a *secret* verification key, if the model has been modified after embedding of a fragile watermark (geometric or topological modifications) if a constant reference watermark is used. The algorithm is even able to narrow the vertices/regions that have been tampered with. In contrast to their work, we target for a different application: Assume a user downloaded a model from the internet and is curious about license conditions: He wants to extract the following information from the model with zero a priori knowledge: Proof of the identity of the creator/license holder of the object, proof that the object has not been tampered with respect to certain tolerances, e.g. truncation of mantissas of vertex coordinates in effect of format conversions should be tolerated up to a certain amount. So in contrast to their work, in our solution everyone is able to retrieve the watermark, get knowledge of the content and even positions of the watermark but is not able to alter the model, exceeding specified tolerances, without removing proof of authenticity/ownership and integrity.

2. The Watermarking Algorithms

2.1. General Philosophy

Watermarking takes place as the last step in the refining process of a model or virtual scene. The topology and connectivity of the object to be watermarked may be the result of the creators intentions and efforts or (partially) result of an automated process (e.g. the outcome of a photogrammatic process or free-form-based surfaces converted to triangular meshes). Depending on the application the mesh to be watermarked may be subject to certain tolerances and restrictions for example restrictions regarding topology changes and vertex displacements. Certain attributes may be associated with the vertices (colors), which may be difficult to maintain or reproduce if general re-meshing is applied in the watermarking process.

Therefore we restrict our algorithms to alter vertex coordinates only. Connectivity and topology remains unchanged. The algorithms allow fine control over vertex movements. Furthermore we preserve all scalar attributes associated with objects (colors, textures,visibility properties etc.) and group-structure of single objects and the overall scene.

The hybrid approach, presented in this paper, is based on three watermarking algorithms namely a Affine Invariant Embedding (AIE), Normal Bin Encoding (NBE) and Vertex Flood Algorithm (VFA) that are under certain restrictions non-interfering and thus stackable. All of them are able to process objects consisting of:

- several (small) unconnected meshes
- meshes that are not manifold
- meshes including doubled faces
- meshes including degeneracies (small faces and angles)

All three algorithms will be sketched in the next Sections.

2.2. Vertex Flood Algorithm for Model Authentication With Embedded Public Watermarks

In order to satisfy the demand for model authentication we developed the so called Vertex Flood Algorithm. Basically

the algorithm modifies vertices, so their distances to the center of mass of a designated start triangle encode the watermark bits. Except for the start face, the algorithm operates solely on vertices and does not take further topological relationships into account. It does not require connectivity of the faces in the input mesh.
Let *com* be the center of mass of the start triangle with edgepoints $S = \{s_1, s_2, s_3\}$. Next we populate the sets

$$M_k = \{v \in V \setminus S \mid k \leq \left\lfloor \frac{|v-com|}{W} \right\rfloor < k+1\}$$

for $0 \leq k \leq N = \lfloor \frac{r}{W} \rfloor$. r is the maximum allowed distance of a vertex v from *com* to be considered for embedding watermark bits. The width of each interval associated with a set is W. Next we loop through the sets M_0 to M_N, skipping empty sets. The vertices of each set are modified in order to embed $m = 2^n$ bits. Each set covers an interval of length W. This interval is subdivided as follows:

buf	I_0	$\cdots$	I_{m-1}	buf

For embedding the value val $(0 \leq val \leq m-1)$, the distance of each vertex in the set to *com* is modified so it comes to lie in the middle of sub-interval I_{val}. The purpose of the two intervals named with *buf* is to prevent modifications of vertices in one interval from causing effects on the embedded values in other intervals.
In the retrieval process, the *mean* distance of all vertices contained in an interval can be used for decoding the embedded bits or the sub-interval containing most of the vertices could be chosen (*majority voting*). This way the robustness with respect to randomization of single vertices can be increased. The values r and W are derived from the start triangle. For fast lookup of the start triangle we demand a certain triangle edge length ratio $1 \leq R_1 \leq R_2$ for it. We choose the triangle closest to this ratio as start triangle and make it the perfect match. The watermark embedded is *fragile*, since it is very sensitive to displacements of vertices, in particular of the start triangle vertices. The interval width W is calculated by $1/3(e_1+e_2+e_3)$, with e_i $(1 \leq i \leq 3)$ being the edge length values of the start triangle.

Comparing our scheme to the perhaps most simple approach in which vertices of a model are ordered by value and watermark bits are embedded in the least significant bits of vertex coordinate mantissas, we find following advantages of our scheme:

- In the simple scheme we need to reverse applying orthogonal transformations, because translation, rotation and uniform scaling affect mantissa bits of coordinates. Our scheme is not affected by orthogonal transformations, neither the ordering of vertices, nor the embedded values.
- Our scheme allows embedding of watermarks locally. Of course, for our method, closeness depends entirely on euclidian distances which, if no further precautions are taken, causes spreading of watermarks over unconnected parts of a model.

2.2.1. Realizing Model Authentication

We provide optional model authentication by applying the following steps:

1. First we choose the maximum distance, vertices are allowed to move without disturbing the fragile watermark. From this distance, we derive according edge lengths and make the triangle closest to these edge lengths the perfect match (ratio & edge length).
2. Next we build up a string to be hashed from the usual labeling and certificate information. The latter one may consist of the object creators certificate (we apply the RSA scheme, certificates are in X509v3 format), or in cases where only small capacity is provided, issuer and subject distinguish name of the certificate or just a weblink for downloading/lookup of the certificate. Next we iterate through the intervals (it should be noted, that there are no buf named subintervals used for model authentication and the last interval encodes bits). For a sequence of empty intervals, we add their number to the string. For each nonempty interval, we append the number of vertices in the interval, the sum of the number of faces and edges adjacent to each vertex in the interval to the string. The final string is hashed with a cryptographic secure hash function (Secure Hash Algorithm, SHA) and signed with the private key of the creator.
3. Finally we embed the labeling info, creator certificate info (issuer and subject distinguish name or URL pointing to certificate) and the signed hash value in the usual way, encoding one bit per interval if possible (depends on capacity requirements). The whole embedding string is appended to itself until it's length matches or exceeds the number of nonempty intervals.

The recipient of the model retrieves the labeling info and certificate information, calculates the hash value and verifies the signature. This scheme detects removal of a single vertex, displacement of the vertex exceeding the specified limit and changes in connectivity of the mesh. Generating a different (*faked*) mesh that can be authenticated with the signed hash value of another mesh, should be a hard task because of the sharp restrictions on point-density and connectivity. There are possibilities for increasing security in this respect:

- If we demand a fixed ordering of vertices and faces, we can simply make the list of faces together with vertex indices part of the hashed string and thus fix topology. For each interval, we add the vertex indices to the hashed string.
- We could adjust vertices, so there is no more than one in an interval. This gives us a fixed numbering of vertices. Using this numbering we add the list of faces and vertex indices to the hashed string.

The algorithm can be modified to rely on vertex coordinates only, which makes it applicable to the general case of watermarking n-dimensional point clouds: From the two

points with maximum distance D to each other we select one as start point from which all point distances are measured. The interval width W is derived from D. However, in contrast to the start triangle case, W cannot be altered to a specific value, so it is difficult to realize *small* tolerances regarding maximum tolerated vertex movement for model authentication.

2.3. The Affine Invariant Embedding Algorithm (AIE)

3D models, which are composed for virtual world scenes will face a rather unlimited number of possible operations. One of the most obvious operations, when placing an object in a virtual scene, is the (non-uniform) scaling to its desired size and shape. Therefore robustness with respect to affine transformations has been the design principle of the AIE algorithm, which is outlined in the following. The embedding algorithm is based on local application of the Nielson Foley norm $\|\cdot\|_V$ as introduced in [6]. Lets repeat its definition for $\mathbf{E}^3$. The norm depends on data, namely the vertices $V_i = (x_i, y_i, z_i)$:

$$\|P\|_V^2 = P \cdot n \cdot [\bar{V} * \bar{V}]^{-1} \cdot P^T$$

$$\bar{V} = \begin{pmatrix} x_1 - \bar{x}, & y_1 - \bar{y} \\ x_2 - \bar{x}, & y_2 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x}, & y_n - \bar{y} \end{pmatrix}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

(with $P \in \mathbf{E}^3$).
We apply it on a set of $n = 4$ neighbored vertices $V = \{V_1, .., V_4\}, V_i \in \mathbb{R}^3$. Next we define

$$M_V = \frac{1}{4}\sum_{i=1}^{4} V_i$$

$$Norm(P)^2 = \|P - M_V\|_V^2 \quad (P \in \mathbb{R}^3)$$

We use the following relationship in order to change a norm value for the purpose of embedding information, given a vertex $P \in \mathbb{R}^3$ not lying in a plane with vertices of set V:

$$P' = M_V + (1+k)(P - M_V) \Rightarrow \frac{Norm(P')^2}{Norm(P)^2} = (1+k)^2$$

If the distance of point P to center of mass M is changed by factor $1+k$, the norm value changes by factor $(1+k)^2$.
We subdivide the continuous one end open interval of possible norm values into intervals of length w where each interval has an associated value: Let n be the number of bits we want to encode into the norm value. We number the i-th interval with i mod 2^n. If the value $val \in \{0, .., m-1\}$ $(m = 2^n)$ is to be embedded, we change the norm value, such that it is moved in the middle of the nearest interval numbered with val which is illustrated in Figure 1.

When choosing the interval width w certain constrains apply:

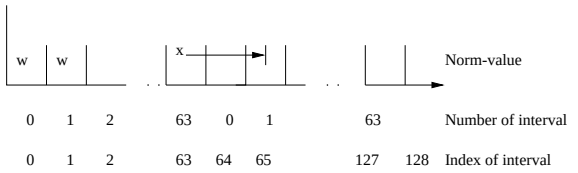

Figure 1: *Adjustment of a norm value: Current norm value falls in interval with index 63 ($m = 2^6 = 64$) and value 1 (6 bits) is going to be encoded. The norm-value is moved to the middle of the nearest interval numbered with 1, in this case interval with index 65.*

Because of limited (fixed) precision of vertex coordinates a certain robustness level dictates a minimum scale factor $(1 + k_{MIN})$.
To guarantee $|k| \geq k_{MIN}$ (movement by at least one interval assumed) for scale factor k we choose

$$w = N((1 + k_{MIN})^2 - 1)$$

Let $N' = N + \Delta N$ denote the new changed norm value, ΔN the applied change. With w chosen as above the following holds:

$$|\Delta N| \leq \Delta N_{MAX} = w(m/2) \tag{1}$$

and

$$|k| \leq \sqrt{1 + (\Delta N_{MAX}/N)} - 1$$

(1) holds since we can move from an interval labeled with val to an interval labeled with val' by adding an increment d with $|d| \leq m/2$ (we exclude the specific case of the first $m/2$ intervals).

We require to use the same interval width in retrieving and embedding process, so w must be derived from a value not or only slightly affected by the embedding process itself. Furthermore w must be constant after subsequent affine transformations of the model. We simply derive the interval width w from the norm value itself. First we normalize the norm value N:

$$N = a \cdot 10^{-i},\ 1 \leq a < 10, i \in \mathbb{Z}$$

with 10^i being the normalization factor. Using this factor We choose a new w as

$$w = b \cdot 10^{-i}((1 + k_{MIN})^2 - 1) \tag{2}$$

with $1 \leq b < 10$. b is called a *secret scale factor*. Given an $\Delta N = \alpha \cdot w$, e.g. $\alpha \in [-128..128]$ for moving the norm value into the middle of the desired interval, the factor for scaling the distance from point P to M is calculated by

$$(1+k) = \sqrt{1 + \frac{\Delta N}{N}}$$

If we want to use the AIE algorithm for secret watermarks,

e.g. for embedding customer specific serial numbers, several parameter should be derived from a secret key. For this reason we added the so called *secret scale factor*. It may be necessary to derive further parameters, e.g. an offset to k_{MIN}. Varying these parameters through the embedding process is not possible, since the *embedding primitives* must be assumed to be accessed in random order. These primitives are explained in the next section.

2.3.1. The Embedding Process

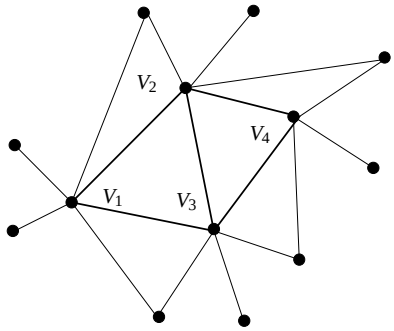

Figure 2: *Embedding primitive is constituted by two start faces with vertices* $V = \{V_1, .., V_4\}$.

In the embedding process, an *embedding primitive* is selected as follows: first a single face with edge vertices $V_1, .., V_3$ is selected. Next an edge adjacent face is selected. Denote the unshared vertex with V_4. These four vertices now constitute the set of vertices V for the local calculation of the norm value $Norm(P)$. They are not allowed to lie in or close to a plane. We ensure this by requiring a minimum distance of the vertex V_4 to the plane constituted by $V = \{V_1, .., V_3\}$. Next we build up two sets G_1 and G_2: G_1 consists of all vertices which are adjacent to *exactly one* vertex of V. G_2 consists of all vertices, which are element of a face that is edge adjacent to one of the start faces. If $|G_1| < 4$ and $|G_2| < 4$ we set $G := G_1 \cup G_2$ else $G := min_{1 \le i \le 2}\{G_i \mid |G_i| \ge 4\}$. If $|G| < 4$ we reject the start faces combination. Figure 2 shows a embedding primitive where G equals G_2. Next we partition G in in four sub-sets $g_1, .., g_4$ with equal number of elements (if possible). The information is embedded into the vertices of $g_1, .., g_4$ as follows:

group	embedded bits				
g_1	00	I_5	I_4	I_3	I_2
g_2	01	I_1	I_0	D_9	D_8
g_3	10	D_7	D_6	D_5	D_4
g_4	11	D_3	D_2	D_1	D_0

In this example the global bit string to be embedded has been split into fragments of length 10 bits each. An index value of length 6 bit is used for addressing these fragments, so the length of the global bit string is limited to 640 bits. The first two embedded bits are used for ordering the parts of data bits and index values. $I_5, .., I_0$ is the 6 bit index value, $D_9, .., D_0$ are the 10 data bits.

Each fragment is embedded multiple times (in current implementation we embed three times by default) in the way described. Adjusted vertices and vertices of start faces are marked as "visited" and used only once.

2.3.2. The Retrieval Process

In the retrieval process, we iterate through the set of faces. For each face we test edge adjacent faces for being the second start face of an embedding primitive: We again calculate vertex sets $g_1, .., g_4$ as described in the embedding process and retrieve the embedded bits for each vertex. If each of the order bits (00,01,10,11) occurs at least once and the same order bits are followed by the same bit-pattern, we store the retrieved pattern (index and data bits) in a global hash table. After testing all *relevant* two face combinations, the patterns with most occurrences for an index constitute the retrieved bit string. The term *relevant* indicates that only a subset of the faces with certain characteristics may be used in the embedding process and therefore be considered as candidates in retrieval process. However these characteristics have to be affine invariant ones, e.g. based on connectivity. In the current implementation we reject faces whose number of edge adjacent vertices exceeds a certain limit. Please note that after we tested the start face combination f_1, f_2 we don't need to test f_2, f_1.

If meshes consist of several planar faces of more than three vertices (e.g. quadrilaterals) our scheme could either ignore embedding primitives for which at least one vertex is part of non-triangular face or we could apply following technique:

- Two edge adjacent planar polygons can be converted to triangular start faces by connecting the shared edge with their center of mass.
- Assume the faces of set $PS = \{P_1, .., P_n\}$ are adjacent to the start polygons. Next we remove faces that are adjacent to a polygon P which is not a triangular face and not element of set PS. For the remaining faces we build vertex sets $VS = \{S_1, .., S_m\}$ by applying following rules: The vertices of adjacent co-planar non-triangular faces are grouped together, while the vertices of adjacent non-triangular and not co-planar faces are removed. A vertex shared between a triangular and non-triangular face is assigned to the vertex set of the latter one. Please note that sets do not contain vertices of the start polygons. We embed a value using vertex set S_i by moving the center of mass calculated over all vertices of S_i (moving the center of mass by vector $\vec{d}$ means moving all vertices contributing to it by $\vec{d}$).

This scheme copes only with "infrequently" appearing non-triangular faces.

An interesting point stems from the fact that the affine

invariant norm is applicable to vector spaces of arbitrary dimensions. As in the VFA case, the AIE algorithm is in principle applicable to watermarking in $\mathbb{R}^2$, e.g. bitmap images converted to vector representation.

2.4. The Normal Bin Encoding Algorithm (NBE)

The algorithm as published in [1, 2] allowed the embedding of a secret watermark with robustness against complex geometry and topology changing operations, most noticeably polygon reduction. The algorithm was of the *forensic type* since it required user interaction at retrieval time. The user had to recognize the model and to select a feature vector of the original, then retrieve the watermark by comparing features of original and watermarked copy. In this paper we describe a variant of the Normal Bin Encoding (NBE) algorithm which embeds a 1-bit watermark and does not require knowledge of a priori data except the secret key at retrieval time. Next we describe how the watermark detector is realized. Finally we give a step by step description of the optimized embedding algorithm, for which computational costs were significantly reduced.

2.4.1. The One-Bit Watermarking Scheme

For the proof of ownership on Mulitmedia data the exploitation of statistical approaches for the embedding of a so called One-Bit watermark is widespread. Despite the first glance drawback of limited capacity, the method shows various advantages and has successfully been used for proof of ownership purposes in a number of applications[7, 10]. The watermarking technique presented in this paper has been inspired from earlier work [1, 2] and is based on distinct bundles of surface normals (bins) and is therefore called Normal Bin Encoding (NBE). Even though our approach with bins as embedding primitives provides due to the limited resolution of the subdivided unit sphere only a restricted number of information carrier units, we can transfer the statistical approach to our geometric problem.

We subdivided the gaussian sphere into N_B non-overlapping bins. Each bin is uniquely defined by a bin center normal and a boundary angle ϕ. A bin is assigned all model face normals whose angle to the bin center normal is less or equal ϕ. For the tests performed, we used 100 equally distributed bins, non-overlapping with radius 12 degree each. Denote the number of normals sampled to bin with index i with n_i $(1 \leq i \leq N_B)$, the sampled normals with N_{ij} $(1 \leq j \leq n_i)$.

Based on a secret key, we generate two disjunct sets of bin indices, $M_1 = \{m_{11}, .., m_{1n}\}$ and $M_2 = \{m_{21}, .., m_{2n}\}$, each consisting of $n = N_B/2$ elements. In the embedding process we try to move normals contained in bins of set M_1 towards the according center. Normals in bins of set M_2 are not changed by intention, although they may change in order to optimize normals of the other set.

In the retrieval process we apply a threshold detector, which, given a secret key K_s, outputs a value estimating the probability of the case the model *does not* contain the suspected watermark. Given N keys $K_{s_1}, .., K_{s_N}$ of suspected watermarks, the detector tells us which one of the watermarks is contained in the model with highest probability.

Denote the mean angular difference of normals in bin i to their center normal with $\bar{Y_i}$, the differences with X_{ij} $(1 \leq j \leq n_i)$.

$$\bar{Y_i} = \frac{X_{i1} + \ldots + X_{in_i}}{n_i}$$

In order to apply a statistically based model we allow the *approximating assumption*, that faces are equally distributed in their representation with their normals on the unit sphere. Under this assumption we can consider the angle difference X_{ij} of a normal to the according center normal as a statistical sample. More precisely: Let us assume the X_{ij} values for all bins are identical distributed with unknown mean value and variance and unrelated. Next assume we have observed mean value m and variance s^2 from $\bar{Y_i}$ values belonging to M_2. Applying the central limit theorem, the variable $\bar{Z}$ with

$$\bar{Z} = \frac{\bar{Y_1} + \ldots + \bar{Y_n}}{n}$$

is $N(m, \frac{s^2}{n})$ distributed (for $m >> 30$).
Next we calculate the value m_2 and s^2 from the $\bar{Y_i}$ of set M_2 and m_1 from the $\bar{Y_i}$ of set M_1. The probability

$$P = \Phi\left(\frac{m_1 - m_2}{\sqrt{s^2 \cdot n_2/n_1}}\right)$$

can be interpreted as follows: It tells how likely it is, to observe the current mean value deviation for sets M_1 and M_2 even if no modifications were applied to set M_1 (false positive rejection). The lower the probability the more likely intentional modifications were applied to normals of bins of set M_1.

There are reasons why the detector is not mathematically sound:

- The X_{ij} are not identically distributed. The directions of face normals of a model are generated in a human driven (non random) construction process.
- The items (normals) of a bin are related. The faces belonging to normals of the same bin, are located next to each other (or near) with high probability.
- both sets M_1, M_2 may contain not enough elements ($>>$ 30 demanded for applying central limit theorem, we used 50 in tests).

So we still have to define the term *well conditioned* models with respect to this detector and to prove that these models meet the assumptions regarding identical distribution and independence of face normals.

In tests the generation of sets M_1 and M_2 was secret key

depended. Of course we could derive additional parameters from that key, e.g. bin center normal (position) and boundary angle of bins to increase mixing of altered an unaltered normals in bins generated for different keys.

2.4.2. The Optimized Embedding Algorithm

Denote the set of model vertices with V. The general structure of the embedding process in pseudocode is:

```
for (i=0;i<numIterations;i++) {
  for each v ∈ V do
    optimizeVertex(v);
  lower tolerances;
}
```

In *optimizeVertex*(), the optimal normal values for adjacent faces of v which are contained in bins of set M_1, achievable by displacements of v, are calculated. We define a function to be minimized which gives the squared distance to these optimal values for a given displacement of v. We calculate a gradient from this function. Since the negative gradient direction may not a valid direction with respect to restrictions regarding tolerated normal changes, we apply a linear simplex for calculating a valid search direction close to it. Next we calculate the optimal vertex displacement in this direction fulfilling the restrictions. We will now present the necessary steps for optimizing a vertex v in detail:

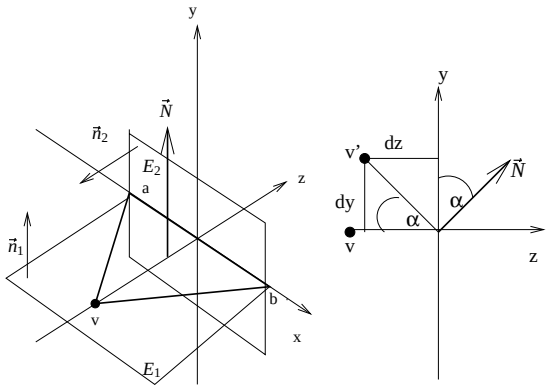

Figure 3: *When vertex v is moved, the angle between plane $z = 0$ and the face normal of triangle consisting of vertices a,b and c changes and can be expressed in terms of distances of v to planes E_1 and E_2.*

Vertex v is going to be optimized. Let f be a face adjacent to v and consisting of vertices v, a, b. Further assume face f is assigned to a bin contained in set M_1 with bin center normal $\vec{N}_B$. Now assume face f has been transformed as shown in Figure 3. If v is moved, the normal of the face $\vec{N}$ rotates around the x-axis. The angular difference α of f normal before and after the movement can be calculated as follows:

$$E_1 : \vec{n}_1 * \vec{x} - d_1 = 0$$
$$E_2 : \vec{n}_2 * \vec{x} - d_2 = 0$$
$$m = \frac{\vec{n}_1 * \vec{x} - d_1}{\vec{n}_2 * \vec{x} - d_2}$$
$$\alpha = atan(m)$$

d_1 and d_2 are 0 in the actual case, but we will transform the whole problem later, so just keep these variables. The denominator is always ≥ 0 because our restrictions assure the normal change of face f to be less than 90 degrees. Next we calculate the Taylor approximation $T(x,y,z)$ of first degree for m at position $v = (v_x, v_y, v_z)$. Now assume there are n face adjacent to vertex v which are contained in bins of set M_1. We get according Taylor approximations $T_1, .., T_n$. Next assume that the face normals should be pushed towards their according bin center (faces in bins of set M_2 don't contribute to global function to minimize). We calculate the optimal value α^* (m^*) for each case, that means the angle, which moves the face normal $\vec{N}$ (which is (0 1 0)) closest to the according bin normal $\vec{N}_B$.

$$Rot_x(\alpha) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{vmatrix}$$

$$\alpha^* = \alpha \; that \; minimizes \left| \vec{N}_B - Rot_x(\alpha) * \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right|$$

$$m^* = tan(\alpha^*)$$

Now we assemble the global function to minimize:

$$F_{MIN}(\vec{x}) = \sum_{i=0}^{n} (T_i(\vec{x}) - m_i^*)^2$$
$$= \vec{x}^T C \vec{x} + \vec{p}^T \vec{x} + c$$

Next we calculate the gradient of F_{MIN}:

$$\nabla F_{MIN}(\vec{x}) = \left(\frac{\partial F_{MIN}(\vec{x})}{\partial x^{(0)}}, \frac{\partial F_{MIN}(\vec{x})}{\partial x^{(1)}}, \frac{\partial F_{MIN}(\vec{x})}{\partial x^{(2)}} \right)$$

The gradient points into the direction of the largest ascend (possibly not leading to a global maximum) from point v so we have to continue in direction $-\nabla F_{MIN}$ in order to minimize the the function. Since this direction may not be valid search direction with respect to certain (to be defined) restrictions, we want to apply a gradient method in which a valid search direction is calculated using linear programming (simplex). We now face the problem that the simplex implementation does only work with non-negative variables. We solve this by transforming our problem, so $-\nabla F_{MIN}(\vec{v})$ (normalized) points in direction $(\frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}})$ (vertices, planes, T_i's and F_{MIN} are re-calculated accordingly). Additionally we assure that v is placed in the origin.

Next we calculate the restrictions used in optimization process. There are four different types of restrictions:

1. new face normal is not allowed to deviate by more than α_1 from original face normal
2. new face normal is not allowed to deviate by more than α_2 from bin center normal (to stop normal from leaving bin)
3. new face normal is not allowed to deviate by more than α_3 from current face normal ("cool down")
4. the vertex v is not allowed to move by more than d_{max} in any direction

For each face two angles, α_{min} and α_{max} ($\alpha_{min} \leq \alpha_{max}$) are calculated for which the face normal $\vec{N}$ (see figure 3) satisfies the restriction with equality. Next, the maximum α_{min} and minimum α_{max} are calculated. Denote these with α^*_{min} and minimum α^*_{max}, the according m-values with m^*_{min} and m^*_{max}. This gives two restrictions:

$$(ax+by+cz+d) - m^*_{max}(ex+fy+gz+h) \leq 0$$
$$-(ax+by+cz+d) + m^*_{min}(ex+fy+gz+h) \leq 0$$

The last type of restriction is added as (since we have assured $v = (0,0,0)$, and all variables are ≥ 0:

$$x \leq d_{max}$$
$$y \leq d_{max}$$
$$z \leq d_{max}$$

No we optimize the system. First we search for a valid search direction. The value $-\nabla F_{MIN}(\vec{v})$ may not yield a valid direction. In this case we apply the simplex method to find a valid one close to it.

$$\begin{aligned} maximize \quad & -\nabla F_{MIN}(\vec{v})^T \vec{d} \\ & a_i^T \vec{d} + c_i \leq 0 \\ & |\vec{d}| \leq 1 \end{aligned}$$

with m restrictions explained before as a_i's. We use the $\|\cdot\|_\infty$ instead of $\|\cdot\|_2$ to get linear restrictions.

Next we have to find out how far to go in the valid direction to get the minimum of F_{MIN} (without taking restrictions into consideration):

$$\frac{\partial F_{MIN}(\alpha\vec{d})}{\partial\alpha} = 2\alpha\vec{d}^T C\vec{d} + \vec{d}^T = 0$$
$$\Leftrightarrow \alpha = \frac{-\vec{d}^T\alpha}{2\vec{d}^T C\vec{d}}$$

$$\frac{\partial F_{MIN}(\alpha\vec{d})}{\partial\alpha} = 2\alpha\vec{d}^T C\vec{d} + \vec{d}^T = 0$$
$$\Leftrightarrow \alpha = \frac{-\vec{d}^T\alpha}{2\vec{d}^T C\vec{d}}$$

In the last step, we search for the largest $\alpha'' \leq \alpha$ which satisfies the constrains:

$$a_i^T \alpha' \vec{d} + c_i \leq 0$$
$$\Leftrightarrow \alpha' \leq \frac{c_i}{a_i^T \vec{d}}$$

$$\alpha'' = max\{min\{\alpha, \frac{c_i}{a_i^T \vec{d}}\} \mid 1 \leq i \leq m\}$$

Next v is replaced by (re-transformed) $\alpha'' \cdot \vec{d}$ and the current normals of faces adjacent to v are updated.

In tests performed we only applied one iteration of the algorithm in combination with restrictions of the first, second and fourth type.

3. Implementation and Results

All three algorithms presented in this paper have been implemented as the Geomark watermarking system as a utility plugin for 3D Studio MAX R2.5. The watermarking plugin accesses objects after they flowed down the pipeline associated with their scene node and converts them to triangle representation using 3D Studio in built functions. A Nurbs-based object for example is replaced with a triangular mesh representation after watermark embedding. Figure 4 shows a snapshot of the plugin when retrieving an AIE watermark (bitmap).

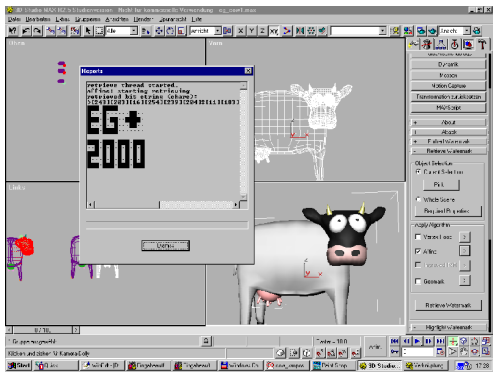

Figure 4: *Screenshot of the Geomark plugin.*

The three watermarking algorithms presented alter vertex coordinates only, and thus only affect the geometry channel of the rendering pipeline. They don't change connectivity or topology, with one exception: No tests for self intersection of faces due movement of vertices are performed for all three algorithms. This of course had no impact for the tests performed. These characteristics allow us to preserve scalar attributes as colors texturization just by copying them from the original to the watermarked version of a model.

The plugin utilizes its own mesh, matrix and vector classes. It operates with double floating point precision, while 3D Studio MAX operates only with single precision (24 bit mantissa, 6.92 decimal digits). To clarify: The watermarks generated are stable for single point precision or less (customizable). For example, the AIE watermarks embedded in experiments were embedded for a precision of 6 decimal digits.

3.1. Practical Experiments

First we embedded an NBE watermark into each model. We set restrictions regarding tolerated vertex displacements and normal deviations by hand to assure that there was no visual impact. Then we added an AIE watermark consisting of a bitmap of 15x11 bits size. We encoded 24 Bits in each AIE embedding primitive, giving 16 Bit capacity per primitive from which we used 6 Bits as index. Each fragment of the global bitstring was embedded three times. Finally we embedded a label with VBA. It consisted of the text "This is a demonstration<lf>of stacked watermarks for the<lf>eurographics 2000 conference<lf>http://www.eg.org/EG2000/<lf>", which is 109 bytes. We embedded 1 bit per interval. The total number of embedded bits was 888 (including 8 bit leadin- and leadout-values preceding and following the data bits). The left image in first row of Figure 6 visualizes the vertices encoding watermark bits (please note that vertices of AIE and VBA do overlap). The second and third rows in Figure 6 show the other models used in tests after embedding all three stacked watermarks. It should be stressed that we achieve "non-interference" of watermarks simply by applying different embedding-strengths (tolerances regarding vertex movements) for the algorithms.

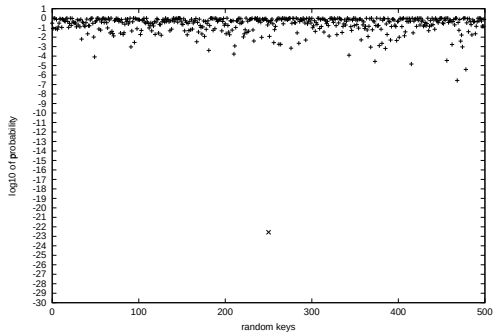

Figure 5: *Image shows detector probabilities with respect to outputs for 500 random generated keys for watermarked copy of the viper model.*

We found AIE and VBE watermarks are in no way perceptible and AIE withstands affine operations exemplified in right image in first row of Figure 6, so we concentrate further tests on evaluation of the NBE algorithms detector. We perfomed tests using two decimation implementations, *qslim*[5] and *plycrunch* which is part of the Simplification Envelopes distribution[4], capable of handling non-2-manifold models. We applied the first one, for getting a measure of robustness with respect to sophisticated simplification while the latter one gives a measure for robustness with respect to most simple schemes, in this case nearby vertex collapsing. The original probabilities yielded by the detector are listed in Table 1. Table 2 lists the probabilities after applying *qslim*, Table 3 for the *plycrunch* case.
The detector gives a measure of deviation of the two distributions associated with the two sets M_1 and M_2. An important issue is how the detector reacts to random key patterns. In Figure 5 we plotted the detectors output for 500 randomly generated keys for the original watermarked copy.

Summarizing the results, AIE and VBA proved to be applicable, while the NBE detector requires more refinement. There are several possibilities for enhancements: First the mixture between sets M_1 and M_2 can be increased by generating bin positions and sizes from the secret keys (we used fix values in tests and just partitioned the bins to sets M_1 and M_2 based on the private key). Second, we might get more robust normals if the whole algorithm operates on a coarser level of approximation of a mesh (see Section 5).

No	Model	vertices	faces	orig. probability
1	cow	6083	12075	3.241080E-45
2	skateboard	23584	46785	1.486888E-10
3	bunny	22610	44622	1.709180E-22
4	viper	20759	38732	9.982807E-23
5	bridge	16177	24714	3.536744E-22

Table 1: *Model size and original detector probability.*

No	reduction	vertices	faces	probability
1	25%	1550	3000	8.778222E-09
1	8%	545	1000	2.775433E-05
2	11%	2694	5000	4.116683E-13
2	4%	1183	2000	1.427411E-06
3	45%	10235	20000	1.750314E-10
3	34%	7720	15000	8.759356E-07
4	26%	6222	10001	4.230604E-22
4	15%	4084	6001	5.942947E-05
5	61%	10162	15000	4.529004E-06
5	40%	7042	10000	7.861884E-01

Table 2: *Detector probability after applying qslim.*

No	reduction	vertices	faces	probability
1	29%	1540	3504	2.292422E-11
1	23%	1257	2819	1.599228E-07
2	60%	13522	27995	7.196427E-11
2	44%	9683	20599	2.588654E-07
3	77%	17237	34311	2.608362E-10
3	58%	12944	25816	1.937350E-05
4	37%	7486	14435	2.379935E-12
4	31%	5895	11822	1.139079E-05
5	78%	10837	19299	1.057597E-08
5	67%	9164	16471	5.908176E-04

Table 3: *Detector probability after applying plycrunch.*

4. Blind Detection of Watermarks

The AIE and VFA algorithms allow for completely blind detection. For VFA, the specifications of the start triangle must be known to the detector, e.g. hardcoded. For AIE parameters are hardcoded (e.g. number of index and data bits in embedding-primitive) or are derived from a secret key (e.g. secret scale factor). The NBE One-Bit variant requires preprocessing prior to watermark retrieval:

- orientation with respect to original
- consistent normals
- matching original with respect to affine transformations

Our current work covers an approach to orient the model with respect to the central moments. If the reorientation is not precise enough, more orientations can be tested brute force (detector value decreases near correct orientation). The normals of faces we rely on do not need to be correct in terms of in/outwards but we require them to have same *consistent* direction (in/outwards) in retrieving process and embedding process. We orient the normals simply by casting a ray from face midpoint to the center of mass of the model and choosing the normal with minimum angle to this ray. If the watermarked copy undergoes affine transformations other than uniform scaling or non-uniform scaling with small deviation in scaling factors, we need to remove the applying transformation prior to watermark retrieval. The AIE watermark can carry information for supporting this process, but it is lost in remeshing-operations. So for a model to which polygon reduction *and* affine transformations have been applied, we fall back to a forensic detection since we require the original model for reversal.

5. Conclusion and Future Work

We presented a public watermarking scheme, VFA, suitable for authenticating models to a user with zero a priori knowledge. As outlined, the VFA algorithm is applicable to non-manifolds and even to the general case of n-dimensional point clouds. The AIE scheme extends the class of meshes that can be embedded with affine invariant watermarks to non-manifolds with the possible extension to process meshes containing non-triangular planar faces. The scheme is by nature applicable to vector spaces of arbitrary dimensions, most interestingly of dimension 2 (e.g. bitmap images converted to vector representation). Finally the described NBE algorithm takes a step towards the realization of blind detection of robust watermarks since it only requires a secret key for testing the watermarks presence. Through stacking of all three algorithms (first applying NBE, then AIE then VFA), we achieve labeling in combination with model authentication and watermarks resistant *either* to affine transformations or polygon reduction. As explained in Section 2.4.2, in NBE vertices are optimized taking only local features into account. Vertex movement is restricted by general tolerances and tolerated normal differences of vertex adjacent faces. For now, the NBE algorithm works best with already polygon reduced models, which possess larger angles between adjacent faces. To cope with highly tesselated smooth models, our current work follows the multiresolution approach of Praun et al. [10] in the sense, that we want to apply NBE to a coarser representation of a mesh which is converted back to full resolution after embedding.

Origins of models used in experiments

The skateboard (creator Philip van Hoof) model was downloaded from avalon1.viewepoint.com, viper and bridge models from www.3dcafe.com, the bunny model from www-graphics.stanford.edu/data/3Dscanrep/ (The Stanford 3D Scanning repository) and the cow model from sweeet.com.

References

1. O. Benedens. Geometry-Based Watermarking of 3D Models. *IEEE Computer Graphics, Special Issue on Image Security*, pp. 46–55, January/February, 1999. 1, 2, 6
2. O. Benedens. Watermarking of 3D polygon based models with robustness against mesh simplification. *Proceedings of SPIE: Security and Watermarking of Multimedia Contents*, pp. 329–340, 1999. 1, 2, 6
3. Y. Boon-Lock and M. Minerva. Watermarking 3D Objects for Verification. *IEEE Computer Graphics, Special Issue on Image Security*, pp. 36–45, January/February 1999. 2
4. J. Cohen et al. Simplification Envelopes. *SIGGRAPH 96 Proceedings*, pp. 119–128, 1996. 9
5. M. Garland and P. Heckbert. Surface Simplification Using Quadric Error Metrics. *SIGGRAPH 97 Proceedings*, pp. 202–216, 1997. 9
6. G. Nielson and T.Foley. A Survey of Applications of an Affine Invariant Norm. *Mathematical Methods in Computer Aided Design*, Academic Press, pp. 445–467, 1989. 4
7. N. Nikolaidis and I. Pitas. Robust image watermarking in the spatial domain. *Signal Processing*, **66**(3), pp. 385–403, May 1998. 6
8. R. Ohbuchi, H. Masuda, and M. Aono. Watermarking Three-Dimensional Polygonal Models. *ACM Multimedia 97*, pp. 261–272, 1997. 2
9. R. Ohbuchi, H. Masuda, and M.Aono. Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications. *IEEE Journal on selected areas in communications*, **16**(4), pp. 551–559, May 1998. 2
10. E. Praun, H. Hoppe, and A. Finkelstein. Robust Mesh Watermarking. *SIGGRAPH 99 Proceedings*, pp. 69–76, 1999. 1, 6, 10

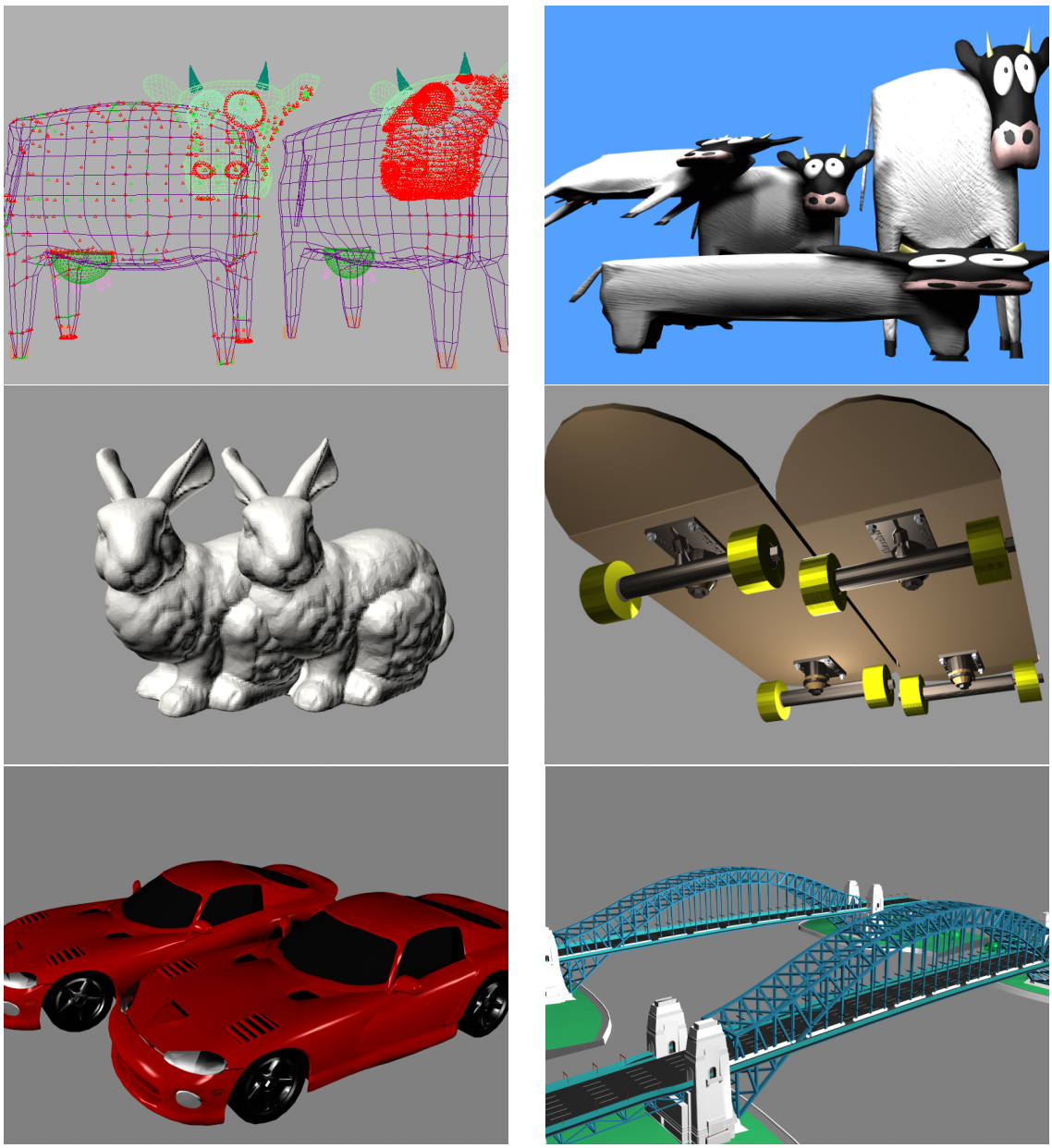

Figure 6: *First row: In the left image vertices encoding AIE (left cow) and VFA (right cow) watermarks are highlighted. For AIE, start face vertices are highlighted with green color. Please note that the watermarks do overlap. The right image shows effects of various affine transformations, for example the leftmost cow is result of non-uniform scaling followed by shearing. In all cases the AIE watermark (15x11 bitmap) was still present. Second and third row: Models of test cases. Left model is original, right (front) model is copy watermarked with all three algorithms.*