

Robust Blind Watermarking Mechanism For Point Sampled Geometry

Parag Agarwal

Balakrishnan Prabhakaran

Department of Computer Science, University of Texas at Dallas
MS EC 31, PO Box 830688, Richardson, TX 75083

Email {parag.agarwal}@student.utdallas.edu, praba@utdallas.edu

ABSTRACT

Watermarking schemes for copyright protection of point cloud representation of 3D models operate only on the geometric data, and are also applicable to mesh based representations of 3D models, defined using geometry and topological information. For building such generic copyright schemes for 3D models, this paper presents a robust spatial blind watermarking mechanism for 3D point sampled geometry. To find the order in which points are to be encoded/decoded, a clustering approach is proposed. The points are divided into clusters, and ordering is achieved using inter-cluster and intra-cluster ordering. Inter-cluster ordering achieves local ordering of points, whereas intra-cluster ordering does it globally. Once ordered, a sequence of clusters is chosen based on nearest neighbor heuristic. An extension of quantization index of bit encoding scheme is proposed, and used to encode and decode inside the clusters. The encoding mechanism makes the technique robust against uniform affine transformations (rotation, scaling, and transformation), reordering attack and topology altering (e.g. retriangulation) attack when applied to 3D meshes as well. Replication of watermark provides robustness against localized noise addition, cropping, simplification and global noise addition attacks. Security of the scheme is analyzed, and the time complexity is estimated as $O(n \log n)$, where n is the number of 3D points. Theoretical bounds on hiding capacity are estimated, and experiments show that a high hiding capacity is high, with embedding rate greater than 3 bits/point. The bit encoding method reduces the distortions and makes the watermark imperceptible, indicated by a signal to noise ratio greater than 100 dB.

Categories and Subject Descriptors

H.4.m [Information Systems Applications] Miscellaneous; K.6.5 [Management of Computing and Information Systems] Security and Protection

General Terms

Algorithms, Security

Keywords

Watermarking, blind, spatial, encoding, decoding

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM&Sec'07, September 20–21, 2007, Dallas, Texas, USA.
Copyright 2007 ACM 978-1-59593-857-2/07/0009...\$5.00.

1. INTRODUCTION

3D objects have found applications in CAD/CAM, animations, movies, video games, and medical informatics (3D Images). 3D objects can be created from software such as Maya [2], or derived using 3D data acquisition devices [14, 17]. These models have commercial value and the process of generating them is a time consuming effort. Theft can lead to illegal usage of the dataset, and this would result in loss of time, money, and effort. In order to avoid such thefts, we can resort to watermarking [13, 19]. Watermarking achieves protection by hiding information (watermark) inside digital medium (3D object in our case). The presence of the watermark verifies the copyright.

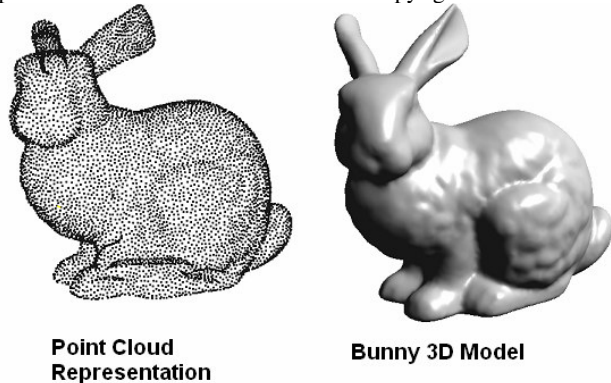


Figure 1. Point Representation of Stanford Bunny Model

Point sampled geometry consists of 3D geometry (see Figure 1 with point cloud representation of Stanford bunny) i.e. point information only. Schemes related to watermarking points (geometry) give us a generic solution to different representations of 3D models, such as 3D mesh and 3D point sampled geometry. This comes from the fact that generic schemes operate only on geometric information of 3D meshes and 3D point sampled geometry. We also observe that in case of 3D meshes, geometry based encoding is robust against connectivity altering attacks, since the encoding is independent of this information. Therefore, this paper focuses on generic schemes of watermarking of point sample geometry representations of 3D objects. In doing so, we have to make sure that we satisfy the properties such as imperceptibility, and robustness.

Imperceptibility: The addition of watermark can lead to distortions, which can produce visual artifacts, resulting in watermark being perceptible. The watermarking technique should try to minimize the distortions produced, in order to satisfy the watermark imperceptibility criteria.

Robustness: An adversary can try to process the data set in order to remove the watermark. These modifications can be as follows:

- *Re-ordering:* 3D models can be represented in different file formats (e.g. polygon format - .ply, virtual reality format - .VRML) and the 3D points are given identified as a (order) sequence. The sequence numbering (ordering) can be changed, but it will not alter the 3D shape. So, once the ordering is changed, any attempt to watermark using such an ordering would result in loss of watermark.

- *Value Change:* During value change attacks the geometric information related values change. These can occur as uniform affine transformations (scaling, rotation and translation occurring without change in 3D shape). Another form of value change occurs due to noise addition during lossy compression and decompression mechanisms. Also, noise can be defined as deformations in the shape of the 3D model. In all such cases, watermark information related to the value of the geometric information is lost.

- *Sample Loss:* In such cases, either sub-parts of the watermark are cropped e.g. head of the Stanford bunny, or points are removed when we simplify the 3D model e.g. represent it in a lower resolution. In both cases, points are removed and watermarking dependent on these points will be lost.

Imperceptibility and robustness are in inverse relation to each other. Robustness can be achieved by replicating the watermark in the 3D data set, by devising a technique with *high hiding capacity*. However, the addition of watermark perturbs the original data set; it adds distortions, which can lead to visual artifacts, and the eventual loss of imperceptibility condition. Therefore, due to distortions induced, devising a technique that can achieve high hiding capacity and improve robustness is difficult. Robustness can also be enhanced if an attacker did not have the knowledge of watermarked locations in the 3D model. So, we need to secure these locations by increasing the search space for the adversary i.e. enhances the *security* of the watermarking scheme.

The amount of information required to verify the presence of watermark should not depend on the size of the data set. This would help the verification information stay minimal in cases where the motion capture data set increases by large amounts. Such storage efficiency can be achieved by making the scheme blind, which requires only a key and a watermark for verification purpose. Therefore, it is necessary that the design of a scheme must be storage efficient i.e. blind.

To solve the above problem, we need to embed and extract watermark related bit information in the 3D points. As mentioned above, we have to make the scheme independent of the order of 3D points given in the file describing the 3D models. For n given 3D points, we can have $n!$ number of orders. Since 3D models can comprise of a large number of points, designing a scheme that finds the order from this search space for watermark embedding and extraction, and satisfying properties such as storage efficiency, high hiding capacity, security, robustness, and watermark imperceptibility is a challenge.

1.1 Related Work and Issues with State of Art

Watermarking techniques [13, 19] developed for other kinds of multimedia such as audio, video, and images data are not generalized enough to be applicable to 3D point clouds. The state of art in watermarking point clouds [11, 21] is private, which makes it storage inefficient. Blind watermarking achieves storage efficiency as it requires only a stego-key and watermark to verify

the copyright. Therefore, the methodology should be blind in nature. As a possible solution, we can reuse techniques such as [22] for 3D meshes that rely only on geometry information for watermarking 3D meshes. However, such techniques use principal component analysis (PCA), which is vulnerable to cropping attacks. In such a case, these techniques need to be treated as private in nature, which is not a suitable criterion for our scheme design.

The problem of watermarking 3D point clouds might be solved by using solutions related to other representations of 3D objects. 3D meshes are one such form, where data set is represented as a graph by geometry (vertices or points) and topological (connectivity) information. A lot of watermarking techniques have been proposed for 3D meshes. These techniques are based on spatial information or some kind of transform. Spatial techniques [3, 4–9, 15, 20, 23] operate on the data, whereas transform techniques [16, 18, 19, 21] apply mathematical techniques to derive information from the original data. Informed or private watermarking scheme for 3D meshes [4 - 6, 16, 18, 20, 21] are not suitable for our scheme. Some of the blind techniques for 3D meshes operate on the geometry (vertex or point) and topological (edges) information of the 3D mesh model. These techniques cannot be applied directly to point clouds since in this case the topological information is absent. We can apply the blind 3D mesh schemes to 3D point sampled geometry, by representing the 3D point sampled geometry as a 3D mesh. In order to achieve the same, we first need to find the connectivity information that is consistent even after the encoding process. This would make the decoding process reversible of encoding process. One of the methods to derive connectivity information is triangulation, which can result in large amount of connectivity information. Choosing points for triangulation can be subject to change due to change in sample points, which can result in inconsistent triangulation and connectivity information i.e. failure to extract the watermark.

In order to overcome the above-mentioned problems, this paper does not rely on existing mesh-based blind watermarking mechanisms and proposes a spatial robust blind watermarking mechanism using geometric information only. The following sub-section lists down the proposed technique and related contributions.

1.2 Proposed Approach and Contributions

The paper contributes by suggesting a clustering approach that helps in finding an order among points to embed and extract the watermark related bit information. The scheme first divides the 3D points (or vertices) into clusters using nearest neighbor heuristic. To trace the order of points inside the cluster that need to be encoded, intra-cluster (local) ordering scheme is used. Once local ordering is achieved, the clusters are ordered globally using inter-cluster ordering, which achieves total ordering of the points. The global ordered clusters are used to arrange clusters in a sequence that are use to embed or extract the watermark. The bit-encoding and decoding scheme used is a proposed extension of quantization index modulation (QIM) [10]. The time complexity of the scheme is estimated as $O(n \log n)$, where n is the number of points. The scheme is spatial, blind (i.e. storage efficient), and also satisfies properties as follows:

High hiding capacity with reduced distortions: The encoding scheme achieves high hiding capacity with embedding rate greater than or equal to 3 bits/point or vertex. Meanwhile, the watermarks are still imperceptible with reduced distortions, indicated by high signal to noise ratio (> 100 dB).

Robustness against Attacks: The mechanism is shown to be robust against attacks such as global noise addition, localized cropping and noise addition, reordering and uniform affine transformations. When applied to 3D meshes, it is robust against all the above mentioned attacks and also, topology change (e.g. retriangulation) attack.

Security: Randomness in vertex selection, cluster formation, and sequence tracing during the encoding process enhances the security of the scheme.

Ideas related to clustering based encoding and ordering to achieve localized watermarking were first suggested by Ohbuchi [20], and have readily been used for 3D meshes in [3, 7, 8, 9, 15]. However, as mentioned above, ideas related to 3D meshes are not directly applicable to 3D point sampled geometry, since these methods require connectivity information, which is not given in case for point sampled geometry. Therefore, we need a novel scheme, in case of point-sampled geometry.

2. SCHEME DESIGN

Point clouds can be visualized as set of spatial information represented as geometrical information in the form 3D Points or vertices with positional information (x, y, z). Unlike 3D meshes, the connectivity information (topological) between vertices is not given. Watermarks represent the copyright information, and are represented as a sequence of bits. In order to embed and extract the bit sequence in these vertices, we need to determine the order in which the vertices are to be chosen. However, since there is no order defined for vertices, we need to find an ordering technique.

The set of ordered vertices can be achieved by using a divide and conquer methodology. The vertices are divided into non-intersecting clusters using a nearest neighbor heuristic. Once the clusters are chosen, we need to find an order for storing the watermark related bits. Ordering is achieved by intra-cluster and inter-cluster ordering. Intra-cluster ordering orders set of vertices per cluster (locally), and by ordering the clusters (inter-cluster) we achieve the global ordering of vertices. The clusters are then grouped by finding sequence of clusters that are nearest to each other. The sequencing and ordering is the conquer phase in our approach. Using the total ordering for each group, we can embed or extract the watermarks. Inside each cluster, we encode the watermark related bit information in the vertices, by using our extension of quantization index modulation [10]. Watermark extraction involves decoding of bit information from the vertices.

The idea to find only the connectivity by identifying the nearest neighbor does not require complete triangulation, and proximity is a criterion based on Euclidian distance that does not change due to uniform affine transformations (scaling, rotation and translation). In case of geometry change attack (e.g. vertex removal), only the vertices nearest to other vertices change and the damage is localized. This is more robust against triangulation based connectivity information which changes globally due to loss of vertices used to triangulate the mesh before watermarking embedding and extraction. Therefore, it is a better decision to use clustering against complete triangulation of the mesh.

The following Sub-sections describe the technique in detail.

2.1 Clustering

The point cloud can be visualized as a graph, where each point is a vertex and the edges are determined by placing an edge between a vertex and its nearest neighbor. It should be observed that more

than one vertex can be near to a given vertex. From this graph, we find the connected sub-graphs that are equivalent to clusters. Since every vertex is connected to at least one vertex, the minimum size of a cluster is '2'. This can be visualized in Figure 2 (a & b), where set of vertices that are not ordered in 3D space are broken down into clusters. The set of clusters is defined by the set $C = \{C_i, 1 \leq i \leq \text{Number of Clusters}\}$.

For each cluster C_i , we identify a graph. The graph has a set of vertices, categorized as cluster heads and encoding vertices. A cluster head is a vertex whose degree is greater than or equal to (\geq) 2, whereas an encoding vertex has degree equal to 1. The graph has two levels: first level defined by the connectivity between two heads, and the other defined by the (encoding) vertices connected to the cluster heads. There can be cases where cluster heads are only connected to other cluster heads. This can be visualized in Figure 2(b), where $\{v_3\}$ is a cluster head, and $\{v_1\}$ or $\{v_5\}$ can be encoding vertices. Also $\{v_0, v_6\}$ are cluster heads, and $\{v_2, v_7, \text{ and } v_8\}$ are encoding vertices.

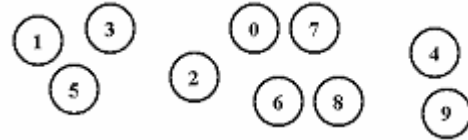


Figure 2 (a). 3D points (vertices)

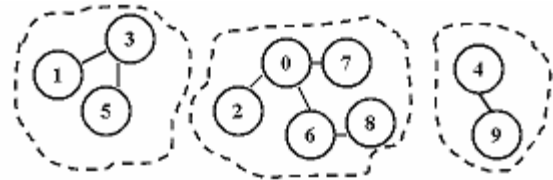


Figure 2 (b). Cluster formations from 3D points (vertices)

For each cluster, we find a reference vertex. In cases, where the number of cluster heads is greater than or equal to (\geq) 2, this reference is the centroid of the cluster heads, and is referred as r (see equation 1).

$$x_r = \frac{1}{nh} \left(\sum_{i=1}^{nh} x_i \right), y_r = \frac{1}{nh} \left(\sum_{i=1}^{nh} y_i \right), z_r = \frac{1}{nh} \left(\sum_{i=1}^{nh} z_i \right), \text{ where } nh \sim \text{number of cluster heads} \quad (1)$$

In case, there is only one cluster head, the reference vertex r is the encoding vertex whose Euclidian distance from the cluster head is maximum or minimum, depending on encoding (see Sub-section 2.4.1). This reference vertex does not act as the encoding vertex and is only used for defining the encoding and decoding scale. This can be visualized in Figure 3, where reference is vertex v_i , and cluster head is vertex v_0 .

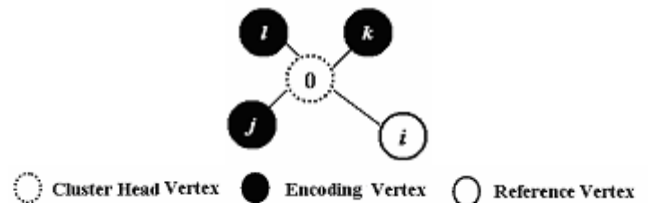


Figure 3. Best Case example for encoding, bits encoded ~ 3

The number of bits that can be encoded in a cluster is determined by the total number of encoding vertices for all cluster heads. The

best case for a cluster is when the number of cluster heads is equal to '1' (see Figure 3), where for '5' vertices, encoding vertices equal to '3'. The scheme uses only the encoding vertices to hide information, since they are connected to one vertex (cluster head) and change in their geometric information impacts their connectivity with the cluster head only. In case, we had chosen the cluster heads to hide information it would have impacted the connectivity as they are connected to more than one vertex, and this would have resulted in clusters being split or merged into new clusters, resulting in loss of hidden information.

The bit encoding scheme (see Sub-section 2.4) suggests that we need at least '3' vertices to encode. This implies that cluster of size '2' cannot be used to encode data as an individual cluster. In order to utilize this cluster of size '2' for encoding purpose, we merge this cluster to another nearest cluster of size greater than (>) '2'. In order to find the nearest cluster, we find the Euclidian distance of the vertices of clusters with size '2' to the reference r of a given cluster, and take the minimum of the two. Once the nearest cluster to is found, we make it a merge both the clusters. We assign the vertex whose distance to the reference is minimum, as the cluster head, and the other vertex as the encoding vertex. This can be visualized in Figure 4, where the example given in Figure 2(b) shows cluster of size '2', being merged to the cluster next to it, using its reference r . The net output of the process is observed in Figure 5, where the size '2' cluster $\{v_4, v_9\}$ is merged into cluster $\{v_0, v_2, v_6, v_7, v_8\}$ to form the clusters C_2 given by $\{v_0, v_2, v_4, v_6, v_7, v_8, \text{ and } v_9\}$. Since vertex v_4 is nearer than vertex v_9 , we assign it as the cluster head and vertex v_9 as the encoding vertex, as seen in Figure 4.

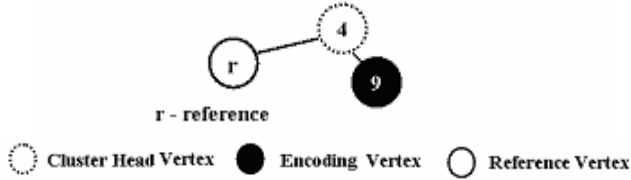


Figure 4. Merging Process for cluster size = 2

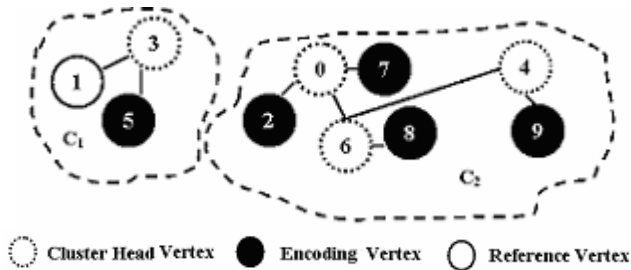


Figure 5. Clusters (C_1 and C_2) for Figure 2 with encoding vertex, cluster head vertex and reference vertex

The net output of the clustering process is a set of clusters with size greater than (>) '2' that are defined by cluster heads, reference, encoding vertices per cluster head, and possible cluster size '2' type cluster members. The number of bits that can be encoded in a cluster is defined by the number of encoding vertices. In our example for Figure 2(b), Figure 5 shows that we have two clusters C_1 and C_2 , with bits to be encoded in $\{v_1\}$ for C_1 , and $\{v_2, v_7, v_8, \text{ and } v_9\}$ for C_2 . Once clustering is achieved, we need to find the order in which bits will be encoded. This can be achieved using inter-cluster and intra-cluster encoding, as explained in the next Sub-section.

2.2 Ordering

Ordering can be defined between clusters and inside the clusters. It should be observed that the order should be based on scalar quantities that do not change as a result of encoding scheme and stay invariant during uniform affine transforms (scaling, rotation and translation). This is a necessary condition since decoding should result in accurate extraction of the encoded bits.

- Intra-Cluster Ordering: To achieve our goal of intra-cluster ordering, we first do ordering at first level i.e. between cluster heads. Cluster heads that are not connected to any encoding vertex are not considered for ordering. Once this ordering is achieved, we do a second level of ordering, and order the encoding vertices per cluster head. The process of inter-cluster ordering is as follows:

a) *First Level:* This case is valid if the number of cluster heads is greater than (>) 1. For each cluster, we determine the tuple $\langle \text{degree, bits, greatest angle} \rangle$. Degree is defined by the number of vertices connected to the cluster head. The number of bits is determined from the number of encoding vertices connected to the cluster head. The greatest angle is determined by finding the angle subtended between encoding vertices, cluster head, and the reference. The order between clusters is found from left to right occurrence of the factors mentioned in the tuple.

b) *Second Level:* The order between encoding vertices is defined by the angle subtended between encoding vertex, cluster head and the reference. This level basically covers only the encoding vertices attached to the cluster heads, and not the member clusters of size '2'.

The order for the encoding vertices for member cluster of size '2' is determined on the basis of $\langle \text{Distance, Angle} \rangle$. The *Distance* is defined by the Euclidian distance between cluster head and reference, and *Angle* is defined by the angle subtended between reference, cluster head, and encoding cluster. The output of the operation will be ordered cluster heads that belong to clusters of size '2'. This order follows the first level order as explained above.

The output of this process is a tuple of ordered cluster heads defined by a cluster head tuple (HT) $\langle H_i, H_j \dots H_k \rangle$ (such that $H_i \prec H_j \dots \prec H_k$). For every cluster head H_i , we have ordered set of encoding vertices defined by encoding vertex set tuple (EVST $_i$) = $\langle v_m, v_n \dots v_t \rangle$, such that $v_m \prec v_n \dots \prec v_t$.

- Inter-Cluster Ordering: The clusters are ordered by the tuple $\langle \text{number of vertices, distance} \rangle$. The distance is defined by the maximum Euclidian distance between cluster heads. For cases where number of cluster heads is '1' (as in Figure 3), this distance is given by distance between the reference vertex and the cluster head. The output of this process is a tuple of clusters (T) = $\langle C_i, C_j \dots C_l \rangle$ such that $C_i \prec C_j \dots \prec C_l$.

Figure 6 shows the example (presented in Figure 2) being ordered by using inter-cluster and intra-cluster ordering. It should be noted that intra-cluster and inter-cluster ordering are independent of each other, and can be implemented to be executed in parallel. Intra-cluster ordering orders the vertices locally, and by ordering the clusters, we order the vertices globally, thereby achieving total ordering of the vertex set. If the clusters are watermarked in the order as shown above, the technique would be vulnerable to cropping attack. This can be visualized in Figure 7, which shows that two clusters can be ordered next to each other but separated due to their locations, for example, the pairs (C_1, C_2), (C_3, C_4) and (C_5, C_6). An attempt to crop the 3D data set can result in loss of continuity of the watermark, and eventual loss of copyright information. Therefore, it is necessary to choose

clusters from the ordered set, such that they are nearest to each other. This would preserve the watermark, since cropping will not result in loss of the continuity of the clusters. The following Sub-section explains the derivation of continuous sequence of clusters.

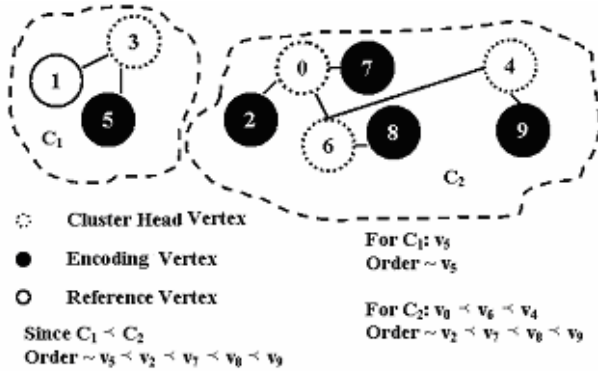


Figure 6. Ordering of Points (vertices) for Figure 2

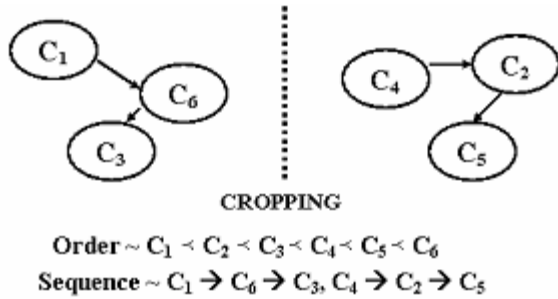


Figure 7. Cropping of 3D Model separates clusters

2.3 Tracing Sequences

Once inter-cluster and intra-cluster ordering has been achieved, we find the sequence of clusters for watermarking. The process finds a set of clusters that can encode watermark of size ($WSize$). The input to this process is a cluster tuple T , which is derived using inter-cluster ordering. As shown in Table 1, we first select a cluster C_i , which is the first cluster in the tuple T . We remove the cluster from the tuple T , and add it to the watermark tuple (WT) that defines an ordered set of clusters used for watermarking. Next, we update the $Bits$ remaining to be encoded by reducing the bits that can be encoded in C_i . After this initialization step, we trace the clusters until the tuple T has no clusters or the number of bits need to be encoded. During this process, we find the next cluster in T such that the Euclidian distances between their reference vertices is minimal. We remove this cluster from T , and reduce the $Bits$ by the number of bits that can be encoded in this cluster. The process is repeated in case T is not empty and $Bits$ greater than ($>$) 0, by considering repeating the above process for this cluster. The process of finding sequences results in watermark tuples (WT) of clusters where each tuple represents clusters ordered from left to right. It is in this order that we can encode the watermark related bit information. For example, in Figure 7, the sequence tracing steps results in a tuple $\langle C_1, C_6, C_3 \rangle$, and it can encode the watermark in each cluster, such that the watermark related bits are stored in the sequence (order) of occurrence of clusters in the tuple (C_1 first, C_6 second, and C_3 last). The bit encoding is done inside each cluster, and is explained in the next Sub-section.

We observe that we start finding the sequence from the first cluster in the tuple T . This helps us chose a cluster in an order in which clusters are arranged globally. Meanwhile, when finding sequence, we resort to proximity that helps choose clusters arranged locally. In case of local attack such as cropping, the clusters are still arranged in the same order and local arrangements are not lost. For example, in Figure 7, when cropping occurs, the watermarks are conserved in the cluster sequence $\langle C_1, C_6, C_3 \rangle$ and $\langle C_4, C_2, C_5 \rangle$. So, our choice of the sequence tracing steps help build groups of clusters that are robust to attacks done on sub-parts of the data set i.e. for attacks such as cropping or local noise addition.

Table 1. Sequence Tracing Steps

Table 1. Sequence Tracing Steps	
Step 1 :	1: Bits = Watermark size ($WSize$)
	2: $C =$ first member in the tuple T , where $T = \langle C_i, C_j, \dots C_l \rangle$
	3: New Tuple (T) = $\langle C_j, \dots C_l \rangle$, Derived by removing C_i from the tuple.
	4: Watermark Tuple (WT) = $\langle C_i \rangle$
Step 2 :	<i>While</i> (Bits > 0 AND Not Empty (T)) <i>Begin</i>
	1: Find the cluster C_n in T , such that the reference vertex of C_i has least Euclidian distance to its reference vertex.
	2: New Tuple (T) = $\langle C_j, \dots C_l \rangle$, Derived by removing C_n from the tuple.
	3: New Watermark Tuple $\langle WT \rangle = \langle C_i, \dots C_n \rangle$ Derived by adding C_n
	4: Bits = Bits – Number of bits in C_n
	5: $C_j = C_n$
	<i>End</i>

2.4 Bit Encoding in Clusters

Bits are encoded or decoded per cluster using the ordered cluster head tuple HT . For each cluster head chosen in this tuple, we have an ordered set of vertices defined by the encoding vertex set tuple ($EVST$). Using the encoding vertices, the cluster heads and a scale for encoding, we can define the bit encoding method. In order to encode a bit inside each encoding vertex v belonging to $EVST$, we consider the $scale$ S defined as the Euclidian distance between the reference vertex and a cluster head in the given cluster. Sub-section 2.4.1 gives detailed analysis for the choice of $scale$. The bit encoding scheme explained below is an extension of quantization index modulation [10].

Encoding is explained in Figure 8, where the scale S stays invariant. However, the Euclidian distance a between H and v is variant, and the changes due to encoding as explained below. The scalar S can be divided into p number of equal intervals, where $p_{min} = 2$. The interval set of 0(s) is identified by Set_0 and 1(s) by Set_1 , ($Set_i; i \in \{0, 1\}$). Let $Pos(v)$: position of v on scale S . The

bit i can be determined based on the position of v on scale S using the following rules:

- $Pos(v) \in Set_i$: No modifications required
- $Pos(v) \notin Set_i$: $Pos(v)$ has to be shifted to a $Pos(v')$ so that $Pos(v') \in Set_i$

As observed in Figure 8, which is a case, where the bit to encode is '1', but v lies on an interval where the corresponding bit is '0'. As a consequence, we need to shift the position of v , which results in change of the scalar from a to a' . This change is implemented by decreasing the Euclidian distance between vertex v and cluster head H , along the straight line joining H and v .

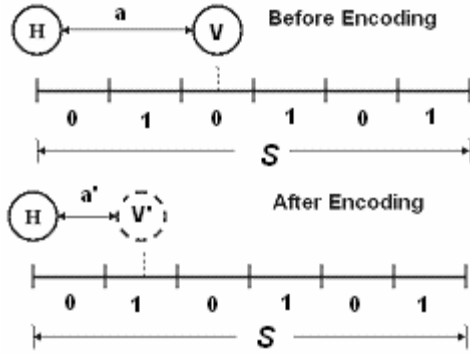


Figure 8. Logical representation of bit encoding using scale S , encoding vertex v and cluster head H . Encoding bit 1 changes vertex v to v' , and Euclidian distance from a to a'

Minimal Vertices for Encoding: The scale S is derived using two invariant vertices in the cluster, and the variant distance a is dependent on two vertices. We note that the cluster heads used to find the scale S and variant a can be different. However, in case we use the same cluster head for *scale* S and *variant* a , the number of vertices used for encoding is '3'. Therefore, we require minimum '3' vertices for encoding.

The advantages of the technique can be listed as follows:

- 1) *Cluster Invariance:* During encoding the variant a decreases (see Figure 8). This is done to increase the chances for maintaining the minimum distance criteria used during cluster formation i.e. vertex v will remain closest to vertex H , even after encoding. As a result, decoding will be possible; since clustering would result in same clusters i.e. clusters are invariant.
- 2) *Uniform Affine Transformation Invariance:* The encoding is dependent on scalars a and S , and they do not change in proportions even due to uniform affine transformations. As a result, the encoding is robust against uniform affine transformation.
- 3) *Controlling Distortions:* The size of the interval determines the displacement of v on S . Smaller the size of the interval less is the distortion produced. Therefore, we can reduce the distortions.

2.4.1 Choosing Bit Encoding Scheme Parameters

The bit encoding scheme requires the use of three vertices to encode a bit; the encoding vertex v , cluster head H and the reference r . These vertices are used to define two scalars S and a , as shown in Figure 8. During the encoding process, S is divided into intervals (see Figure 8). The intervals size depends on the number of intervals. Since different models have different vertices with different scale S , each may need different number of intervals to minimize distortion. It can be observed from Figure 8

that the scale S should be greater than a , as we have to measure the variant a against the scale. In order to make sure that such a condition is possible, the scale chosen is always greater than the maximum sized a in the data set i.e. all the scales used for encoding are greater than the largest variant a . To do so, we increase the scale by a *factor*, changing the scale from S to S' . The factor increments the scale in terms on interval sizes, so that we encode a bit using any encoding vertex that defines the variant Euclidian distance a . This can be visualized in Figure 9, where $S (< a)$ is extended for encoding purpose. Therefore, we can assume new scale S' , which is a large scale whose size is a multiple of an interval size.

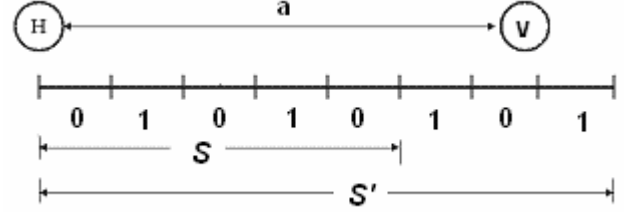


Figure 9. Variant a greater than (>) invariant scale S , S is changed to S' to make scale $> a$

Also, since we need to minimize distortions, we have to choose number of intervals p that minimizes the distortion. By extending the scale from S to S' , we are safe, and this representation can be seen in equation (2) and (3)

$$Scale\ S' = factor * S, \text{ where } factor \geq 1 \quad (2)$$

$$Interval\ Size = \frac{S}{p}, \quad p > 0, \quad p \sim \text{number of intervals} \quad (3)$$

In cases, where S is large, and the interval size is much greater than (\gg) a , any change to the value of a will perturb the positional values of vertex v by large amount, causing a lot of distortion. However, the number of intervals can be chosen sufficiently large to make sure that the interval size is less than the maximum value of a . We can still adopt a strategy in order to maximize our chances that the interval size is less than all variants a . Since this condition is to be met for all variant(s) a , we need to make sure that the smallest a (a_{min}) is greater than the interval size. In order to guarantee such a condition, for a fixed number intervals p , we have to choose a scale S such that the difference $|S - a_{min}|$ is minimized. For example, for a scale with two intervals and interval size equal to $\frac{S}{2}$, if $a_{min} > \frac{S}{2}$, we can

guarantee that all $a(s)$ are greater than the interval size. To guarantee such a condition, we chose a scale that is minimal of all possible values of S , defined by the Euclidian distances between cluster heads and the reference vertex. This might fail for a case of star shaped cluster, where there is only one cluster head, and S is larger than the variant a , since we choose a reference from a set of encoding vertices with maximum distance from the cluster head. Although, this helps us find a scale larger than all the variants a , but it might create a problem where the scale S is much larger (\gg) than variants a . To handle such a case, choose S based on a (reference) vertex that is nearest to the cluster head. This would make all variants ($a \geq S$). However, we change the scale from S to S' by using the factor as explained above in equations (2) and (3).

Based on the above discussion, we can conclude that during encoding, in order to avoid unnecessary distortion we can choose a scale that is the minimum of all set of scales S , defined by the Euclidian distances between cluster heads and the reference vertex. In order to make sure that the scale is greater than variants, we can increase its length by using a factor that increments its length by the interval size. The interval size is dependent on the number of intervals in which we divide the scale and try to minimize the distortion.

The reader should note that the scale is always defined in terms of number of intervals and the factor, since the scale length can vary for different clusters. We do not use different scales for different clusters, as this would make the data used for verification purpose data size dependent, thus making it storage inefficient. So, we use once scale that encodes all encoding vertices and tries to minimize distortion. The same scale is used for watermark extraction using the decoding strategy as described below.

2.5 Decoding Strategy

The decoding strategy is used during watermark extraction, and is similar to encoding. Since the scheme is blind, we do not use the original data set for watermark extraction. We find the order and the sequence of clusters, and within each cluster determine the encoded bit using the same scale that was used for encoding. The presence of at least one watermark is enough to verify the copyright claim. In order to identify the presence of a watermark, we use detection ratio (DR). This ratio (see equation 4) measures the similarity between original watermark w , and detected watermark w' .

$$\text{Detection Ration (DR)} \geq \text{Threshold, where DR} = \frac{\sum_{i=1}^{wsize} (w(i) * w'(i))}{wsize}, wsize \sim \text{watermark size in bits, } w(i) \sim i^{th} \text{ bit of original watermark } w, w'(i) \sim i^{th} \text{ bit of detected watermark } w' \quad (4)$$

DR is used during the detection process, and is measured on the basis of a threshold. We always assume threshold for detection ratio is '1', to cover the worst case of analysis.

3. PROBLEMS AND RESOLUTIONS

The following sub-sections analyze factors that result in problems and the possible resolutions for the same.

3.1 Encoding Based Error

The encoding scheme does a spatial substitution of vertices. In such a case, the Euclidian distance between vertices changes. Since the cluster formation is based on Euclidian distance, the clusters formed after encoding might differ from the original data set. As a result, the decoding process will result in loss of watermarks. This would result in error condition without attack on the system. In order to reduce these errors, we always make sure the distortions produced are minimized. This can be controlled by increasing the number of intervals during encoding (p in equation 3) i.e., adjusting the encoding scale. We also observe that since v as shown in Figure 8 moves towards the cluster head H , it does not move away from H . Since edges

between vertices are based on nearest neighbor property, there is likelihood that this will not change H as the nearest neighbor of v . Therefore, there is a higher chance it remains a member of the cluster.

3.2 Pathological Case

This is a case, where all clusters are of size '2' (see Figure 10). In such a case, we cannot use the options of merging these clusters with larger clusters. To handle this problem, we assume each cluster equal to the one vertex, and run the clustering algorithm on these assumed vertices. The Euclidian distance between two vertices in this case is equivalent to the minimum distance between two vertices of two different clusters. For example, in Figure 10, the Euclidian distance for vertex (v_1, v_5) and (v_2, v_3) is defined by the distance between vertices v_1 and v_3 . This would result in clusters as shown in Figure 11.

In cases, where two vertices form a cluster (e.g., $\{(v_1, v_5), (v_2, v_3)\}$ in Figure 11), we choose the cluster head as (v_1, v_5) , based on the maximum Euclidian distance between the vertices. The reference in this case is defined by the vertex that belongs to cluster head. This can be chosen from the two vertices $(v_1$ or $v_5)$, based on the minimum distance criteria used during cluster formation. As observed above, v_1 was used in minimum distance criteria for (v_1, v_5) and (v_2, v_3) . Therefore, it is chosen as the cluster head, and vertex v_5 is taken as the reference.

For other cases, where cluster size greater than '2', we represent the cluster-head by centroid of the vertices (In Figure 11 apply equation (1) to $\{v_4, v_9\}$). Once the centroid is determined, we find the reference based on the centroid of centroid of all the vertices. Once the reference is defined, we can find the order and the sequence of clusters used to encode.

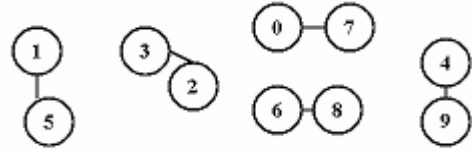


Figure 10. Clusters of Size ~ 2

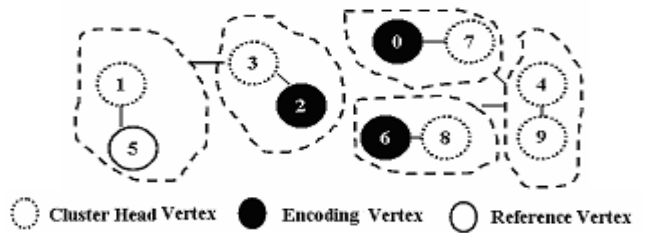


Figure 11. Clusters of Size (> 2) formed from clusters size ~ 2.

In order to encode, we use the idea presented in Sub-section 2.4 (see Figure 8), where clusters of size '2' are encoded using a reference. The encoded vertices in Figure 10 can be visualized in Figure 11.

3.3 Robustness Enhancement

The encoding is done by choosing a set of clusters in sequence. Loss of one cluster in this sequence might result in the loss of the entire watermark. To overcome this problem, we try to increase the hiding capacity per cluster, in order to reduce the sequence size. Another impact of doing so is the increase in hiding capacity of the scheme, which implies more replication of the watermark,

resulting in an enhancement in robustness. To achieve the same, we use the generalization of quantization index modulation [10], where we can encode more than 1 bit per encoding vertex, say b bits. In such a case, the scale has intervals labeled with values 0 to 2^b-1 . For example, as shown in Figure 12, ‘2’ bits of information can be encoded by labeling the intervals as {00, 01, 10, and 11}. To encode bit sequence ‘10’, we change the Euclidian distance (from a to a') between cluster head and encoding vertex such that it lies in an interval which represents the bit ‘11’. For the same interval size, it can be observed that the distortion introduced increases as we increase the bit information added per encoding vertex, since the Euclidian distance has to be modified by a larger magnitude. The increase in distortion can be dealt with by using smaller length intervals or in other words increasing the number of intervals used to divide the scale.

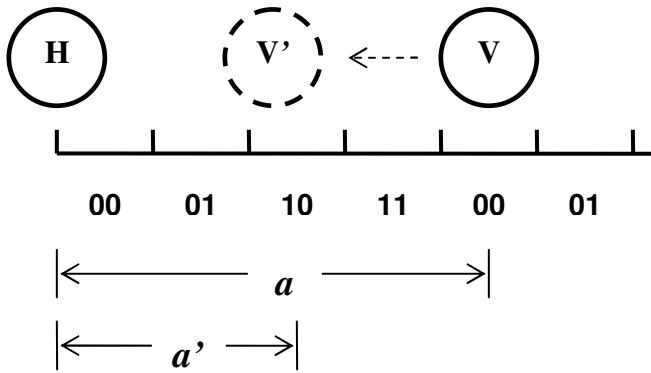


Figure 12. Generalization of QIM [10] to encode 2 bits per encoding vertex v

4. SCHEME ANALYSIS

The analysis for hiding capacity, time complexity and security is presented as follows:

4.1 Hiding Capacity Estimation

The hiding capacity can be estimated from the clusters that are used to encode the data. Each cluster defines a bit capacity which depends on the number of encoding vertices in the cluster. As observed in Sub-section 3.3, we can use a generalized case of quantization index modulation to encode more than one bit per encoding vertex. So, we can say that b bits are encoded per vertex (point) and this is defined as the *encoding factor*. The hiding capacity is a multiple of b in such a case, and is given in equation (5).

$$\text{Hiding Capacity} = \sum_{i=1}^{nc} bits_i, \quad nc = \text{number of clusters}, \quad bits_i =$$

$b \cdot nvc_i$, $b \sim$ encoding factor defined as bits per encoding vertex, $nvc_i \sim$ number of encoding vertices for i^{th} cluster (5)

As already observed in Sub-section 2.1, the best case of cluster is a single cluster with a single cluster head, and rest of the vertices as encoding vertices. For a model with n vertices, there can be $(n-2)$ encoding vertices, since one vertex from n vertices is used as a reference vertex and another vertex is used as the cluster head. Therefore, $b \cdot (n-2)$, defines the upper bound on hiding capacity of the scheme. The lower bound on the hiding capacity is ‘0’, where

each cluster’s hiding capacity is ‘0’. This can occur in cases where each vertex in a cluster is a cluster head e.g. a cube.

4.2 Time Complexity

The time complexity can be computed by analyzing the following steps:

1) *Cluster formation*: Using data structure such as KD-trees [1], we can compute nearest neighbors in $O(n \log n)$, for n vertices. Therefore, for n vertices, the complete process will take $O(n \log n)$ steps.

2) *Ordering*: Ordering efficiently k vertices per cluster requires $O(k \log k)$ steps, if we use the heap sort algorithm [12]. In case, there are nc number of clusters of maximum size t , ordering requires $O(nc \cdot t \log t)$. In addition, as $t = n/nc$, the running time will be $O(n \log(n/nc))$. Since $(n/nc) < n$, the ordering time complexity is less than $O(n \log n)$.

3) *Sequences*: Finding sequences requires nearest neighbor based search. Therefore, using the KD-trees [1], the running time of this step is $O(nc \log nc)$, where nc is the number of clusters formed. The maximum number of clusters (nc_{max}) that can be formed is $n/2$, assuming two vertices per cluster. As a result, $nc < n$, and it implies that $O(nc \log nc)$ is less than $O(n \log n)$.

4) *Encoding or Decoding*: Encoding and decoding takes place in $O(nv)$ steps, where nv is the number of encoding vertices. Since, $nv < n$, $O(nv)$ is less than $O(n \log n)$.

From the above observations, it can be inferred that the time complexity of algorithm is $(O(n \log n) + O(n \log(n/nc)) + O(nc \log nc) + O(nv)) \sim O(n \log n)$.

4.3 Security

It is trivial to trace the vertices (or points) used to hide information based on nearest neighbor based clustering, ordering, and sequence tracing mechanism. As a result, an adversary can launch attack on the encoding vertices. In order to make it difficult for the adversary to guess the encoding vertices, we can introduce randomness based on a secret key. Randomness can be introduced, as explained below:

1) *Randomness in sequence tracing process*: This is achieved by randomly choosing cluster (in step 1 of Table 1) from the ordered tuple of clusters. Since the cluster sequence is dependent on the first choice determined in step 1, randomness in this process induces randomness in other choices as well. Since, there can be $nc!$ different orders ($nc \sim$ number of clusters) used for encoding; the adversary will have to determine the choice from a search space of $nc!$.

2) *Randomness in vertex encoding process*: We need not encode all potential vertices. These vertices can be chosen randomly chosen. In this way, the adversary will have a search space of 2^{nv} , where nv is the number of encoding vertices.

3) *Randomness in cluster formation*: The scheme used nearest neighbor heuristic to form clusters. We can change the strategy by finding k^{th} nearest neighbor, and make k as a secret (part of the secret key used for watermarking). In addition, we can have rounds for randomly chosen k , where for every round we find randomly chosen clusters of size t ($\leq max$, where max is the maximum cluster size) As a result, the adversary will have to guess from a search space of $2^k \cdot 2^t$.

From the above discussion, we can conclude that the key to encode/decode the watermark should consist of seeds to random number generators for vertex selection, cluster formation, and sequence tracing. The approach presented in this paper is limited

to $k = 1$ and the impact of using the above cases will be taken up as a future initiative.

A common attack on a watermarking system is the addition of another watermark which overwrites a previous watermark. To overcome this problem, a standard resolution is to assume that the key used to encode and decode the watermark is secret and signed with the private key of a certifying authority. This signed key can be verified using the public key of the certifying authority. Any attempts to overwrite the watermark can be avoided, since the key used in this process will be assumed to be illegal, i.e. not signed by a certifying authority.

5. EXPERIMENTS AND RESULTS

The scheme has been implemented in C, and tested in UNIX and Windows operating system. We have used 3D mesh models [17], and watermarked their geometric information in order to show that our schemes are genetic in nature. The attacks are measured in terms of percentage of data set impacted. The metric used for the experiments are given as follows:

5.1 Metrics

The distortion produced due to addition of watermarks can be analyzed visually. However, it is a time consuming process to check for a large number of models and watermark. So, in order to measure distortion, we can use *signal to noise ratio* (SNR), see equation (6) [23] between original data (Data) and watermarked data (Data') for n vertices.

$$SNR(D, D') = 10 \log_{10} \sum_{i=1}^n (x_i^2 + y_i^2 + z_i^2) - 10 \log_{10} \sum_{i=1}^n ((x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2), n = \text{number of vertices, Data } D, \text{ Watermarked Data } D' \quad (6)$$

We measure the errors due to encoding based on *bit error rate* (BER), determined as a ratio of the total bits lost to the total bits encoded. Higher the rate more the chances of losing watermarks due to encoding.

In order to find the robustness of the technique, we use *watermark detection rate* (WDR) defined as the ratio of the total watermarked detected to the total watermarked embedded. The detection rate measures the percentage of watermarks detected. Higher the rate more is the chance of verifying the presence of the watermark i.e., more robust is the technique. The above metric is also an indicator for the false negative rate (FNR) or the probability of failure to detect the watermark. This is true since the presence of a single watermark makes WDR > 0 and FNR = 0.

The *hiding capacity* is defined in terms of the total encoded bits and the *encoding factor*, defined as number of bits per encoding vertex (point). We also estimate the *embedding rate* defined as the ratio of the hiding capacity to the total number of points.

5.2 Performance Analysis

This Sub-section analyses the performance using 3D models such as Stanford bunny, horse and Porsche car. To encode a 3D model, we *benchmark* the scale, such that the distortion and bit error rate induced due to encoding based errors are minimal. Table 2 gives an example of encoding analysis used during the benchmarking process of Stanford bunny model's scale. It indicates that distortion can be reduced (indicated by higher SNR) as mentioned in Sub-section 2.4, by decreasing the interval size i.e. dividing the

scale into larger number of intervals. In addition, it also shows the observation in Sub-section 3.1 - errors induced due to encoding can be reduced by minimizing the distortion. This can be observed by the reduction in bit error rate (BER) for higher SNR values.

Table 2. Encoding Analysis for Stanford bunny model using Watermark Size = 30, and varying scale with factor = 280, encoding factor ~ 1 bit/encoding vertex

Intervals	10^3	10^4	10^5	10^6	10^7
SNR	29.32 dB	52.1 dB	71.2 dB	91.8 dB	111 dB
BER	0.461778	0.3841	0.1532	0.0124	0

5.2.1 Hiding Capacity and Distortion Analysis

Different 3D models (see Figure 13) were subjected to embedding with watermarks of size ~ 30 bits. For an encoding factor of 1 bit per encoding vertex and benchmarked scales, it was observed (see Table 3) that the SNR is greater than (>) 100 dB. Visual inspection (see snapshots in Figure 14) for watermarked models was done, and they were found to be similar to the original models. The number of vertices used for encoding is from 30% to 50% of the original data set. Since we can use generalized QIM [10] as suggested in Sub-section 3.3, the encoding factor can be improved. Table 4 gives distortion analysis with encoding factor of 10 bits per encoding vertex and with different benchmarked scales (defined by factor and number of intervals). It is observed that even though the hiding capacity becomes 10 times the original, the watermarks are still imperceptible, since SNR > 100 dB. This is a significant improvement (see Table 4) as seen in the increase in the actual number of bits that can be encoded. The high capacity is also indicated by the embedding rate, where each model's vertex (point) encodes at least 3 bits.

Table 3. Encoding analysis for 3D Models with benchmarked scale (factor = 280 and number of intervals ~ 5×10^7 , BER ≤ 0.06), watermark size ~ 30 and encoding factor ~ 1 bit/encoding vertex

3D Model	Vertices	Encoding Vertices	Hiding Capacity	SNR
Bunny	35947	14841	1.75 KB	128.6 dB
Horse	48485	20060	2.45 KB	127.2 dB
Porsche	5247	2241	0.275 KB	126.3 dB

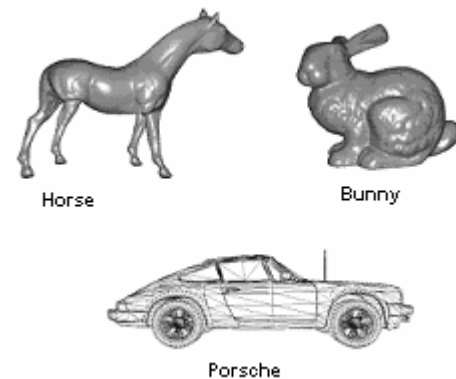


Figure 13. 3D Models

Table 4. Encoding Analysis for 3D Models different benchmarked scales (factor, number of intervals, BER ≤ 0.09), watermark size ~ 30 and encoding factor ~ 10 bits/encoding vertex

3D Model	Scale \sim (factor, No. of Intervals)	Increase in Hiding Capacity	Embedding Rate (bits/point)	SNR
<i>Bunny</i>	(150, 2^{22})	15.35 KB	3.85	118.7 dB
<i>Horse</i>	(200, 2^{20})	20.22 KB	3.83	106.4 dB
<i>Porsche</i>	(200, 2^{20})	2.55 KB	3.94	102.2 dB

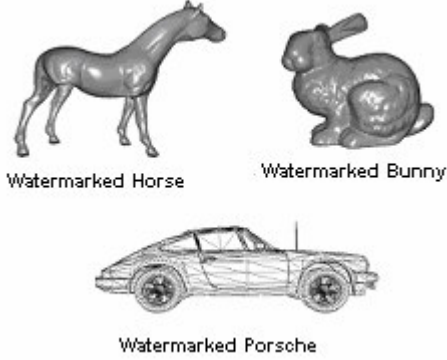


Figure 14. Watermarked 3D models

5.2.2 Robustness Analysis

Robustness analysis can be done on the basis of the following *attack models*. Based on the region in 3D models, we can define attacks as local or global. Global attacks can be formulated by impacting the vertices in the 3D models in a random fashion, whereas local attacks try to attack vertices situated in a sub-part only of the 3D model. For our experiments, we choose the following models of attacks, given in Table 5 and shown for the Stanford bunny in Figure 15. The local attacks were implemented by randomly choosing sub-parts of the 3D model. Cropping attack removes a sub-part of the data set. Simplification attack removes vertex information globally in a random fashion. Noise addition attacks add white Gaussian noise and are well suited to simulate impacts on lossy compression schemes that add noise and might remove the watermark. These attacks are done by randomly choosing vertices globally, and adding white Gaussian noise. Another kind of attack is adding noise in local regions such that it deforms the 3D model and changes the meaning. As a consequence, deformations can be realized as local meaning altering noise addition attacks. Uniform affine transformation commonly uses operations such as scaling, rotating, and translating which do not alter the meaning.

Table 5. Different Attack Categorization

Attack Type	Region based
<i>Cropping</i>	<i>Local</i>
<i>Simplification</i>	<i>Global</i>
<i>Noise Addition</i>	<i>Global</i>
<i>Noise Addition</i>	<i>Local</i>
<i>Uniform Affine</i>	<i>Global</i>

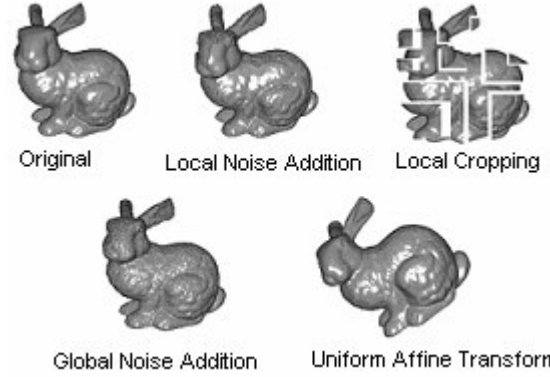


Figure 15. Snapshots of Attack Models using the Stanford bunny model

Watermark Detection Rates against Attacks: In order to observe robustness, we calculated the watermark detection rate (WDR) after subjecting the model to attacks as suggested above. The attacks were defined in terms of the percentage of vertices being subject to a given attack. The watermark size being searched is 30 bits. Since we do not depend on ordering and topology information, the scheme is 100% robust against reordering and re-triangulation attacks. It was found that the technique is 100% robust against uniform affine transformations (scaling, rotation and translation). The local noise attack was done using white Gaussian noise of 40 dB intensity. The related detection rates are given in Table 6, and snap-shots in Figure 16. Table 7 gives the detection rates for localized cropping attacks. Figure 17 gives snapshots of local cropping. In all the above cases, the watermark was detected.

Table 6. Watermark Detection Rate (WDR) for varying % of local 40 dB white Gaussian noise attacks, encoding factor ~ 1 bit/encoding vertex

3D Model	10 % Attack	20% Attack	40% Attack
<i>Bunny</i>	30.49 %	21.36 %	6.40 %
<i>Horse</i>	68.98 %	62.65 %	57.68 %
<i>Porsche</i>	38.73 %	30.14 %	29.95 %

Table 7. Watermark Detection Rate (WDR) for varying % of local cropping attack, encoding factor ~ 1 bit/encoding vertex

3D Model	30 % Attack	40% Attack	50% Attack
<i>Bunny</i>	8.36 %	6.50 %	4.10 %
<i>Horse</i>	44.41 %	37.34 %	29.21 %
<i>Porsche</i>	21.50 %	15.82 %	10.61 %

Global noise attacks were done using 40 dB white Gaussian noise, and global losses were carried out by simplification attacks. Compared to localized noise additions and cropping (sample losses), the global attacks lead to more percentage loss of watermarks. This can be observed by fixing the number of bits encoded per encoding vertex, and finding greater WDR(s) under higher percentage of attacks, as seen in localized attack Tables 6 & 7 when compared with global attack, given in Tables 8 & 9. For

example, WDR ~ '0' for 10% global attack, but it is not '0' for higher percentage (> 10%), in case of local attacks. The vulnerability of watermark loss during a global attack is more since clusters are impacted globally i.e. we attack comparatively a larger set of sequence of clusters. As a result, we observe a higher loss as compared to localized attack, wherein loss is local and lesser number of sequences of clusters are impacted globally.

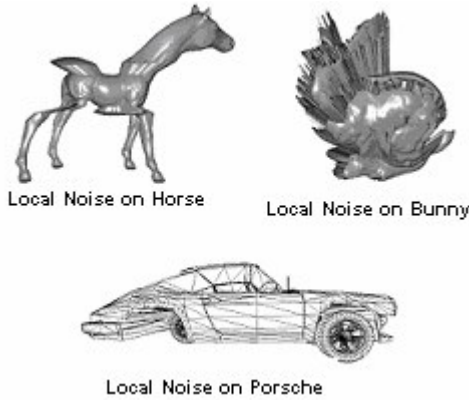


Figure 16. Snap-shot of local noise addition, watermark detection rate (WDR) is > 0 i.e. watermark detected

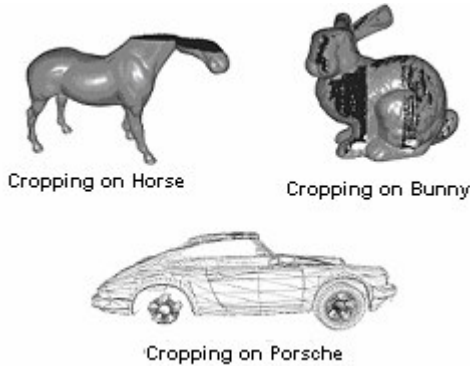


Figure 17. Snap-shot of local cropping attack, watermark detection rate (WDR) is > 0 i.e. watermark detected

Table 8. Watermark Detection Rate (WDR) for varying % of global 40 dB white Gaussian noise attack, encoding factor ~ 1 bit/encoding vertex

3D Model	1% Attack	5% Attack	10% Attack
Bunny	4.98 %	0	0
Horse	6.64 %	0.15 %	0
Porsche	28.16 %	1.41 %	0.14 %

Table 9. Watermark Detection Rate (WDR) for varying % of Simplification (global loss) attack, encoding factor ~ 1 bit/encoding vertex

3D Model	1% Attack	5% Attack	10% Attack
Bunny	6 %	0	0
Horse	8.17 %	0	0
Porsche	19.71 %	0	0

The robustness against global attacks can be improved by using generalized QIM [10] as suggested in Sub-section 3.3, wherein bits encoded per vertex is increased to 10 bits per encoding vertex. This is possible since watermarks are still imperceptible as observed during our hiding capacity and distortion analysis. This helps in increasing the hiding capacity since we can encode more bits per clusters i.e. for the same watermark size, the number of clusters used to store the watermark related bit information is comparatively less. This can be understood by observing Table 8 & 9 (global attacks with 1 bit/encoding vertex) compared with Tables 10 & 11 (global attack with 10 bits/encoding vertex), wherein WDR is comparatively more, indicating that watermarks for higher bit encoding factors, survive higher percentages of attacks. Also, by comparing results in Tables 6 & 7 with Table 10 & 11, since for higher % of attacks, WDR is comparatively higher for local attacks; we can conclude that robustness against local attacks is higher even at lower encoding factors, as compared to cases where we use higher encoding factors to improve the robustness against global attacks.

Table 10. Watermark Detection Rate (WDR) for varying % of global 40 dB white Gaussian noise attack, encoding factor ~ 10 bits/encoding vertex

3D Model	1% Attack	5% Attack	10% Attack
Bunny	56.09 %	16.16 %	4.15 %
Horse	53.48 %	17.92 %	6.06 %
Porsche	62.68 %	31.70 %	14.87 %

Table 11. Watermark Detection Rate (WDR) for varying % of Simplification (global loss) attack, encoding factor ~ 10 bits/encoding vertex

3D Model	1% Attack	5% Attack	10% Attack
Bunny	61.18 %	20.62 %	7.08 %
Horse	55.41 %	19.25 %	6.86 %
Porsche	59.75 %	26.34 %	10.48 %

5.2.3 Comparative Analysis

We need to compare this technique with mesh based blind watermarking approaches. As already mentioned in Sub-section 1.2, these approaches cannot be directly applied to point sampled geometry since they rely on topology information which can change due to attacks such as re-triangulation. Since our scheme does not rely on edge information, it is trivial to compare the robustness. However, we compare our approach with mesh based approach based on the memory usage, since the vertices remain the same, but there is a difference in the number of edges used. Therefore, in order to compare for reduction in memory consumption, we used % reduction in edge information. From Table 12, we can see that for different models the approach gives a significant reduction (> 67 %).

Table 12. Percentage Reduction in edges

3D Model	% Reduction in Edges
Bunny	98.81 %
Horse	88.48%
Porsche	67.50%

6. CONCLUSION AND FUTURE WORK

The paper presented a spatial blind technique to watermark 3D point samples geometry. The encoding scheme proposed is an extension of quantization index modulation that can achieve a high hiding capacity (embedding rate > 3 bits/point), and also maintains the imperceptibility of the watermark with reduced distortions (signal to noise ratio > 100 dB). Bounds on the hiding capacity of the scheme are also analyzed, with upper bound and lower bound calculated as $b*(n-2)$ and 0 respectively, where n is the number of points and b is the encoding factor (bits/encoding point). The time complexity of the scheme is estimated as $O(n \log n)$. The scheme is made secure by using randomness in the encoding process. For point clouds, the technique is shown to be robust against attacks such as uniform affine transformations (scaling, rotation, and translation), reordering, noise addition, and simplification and cropping. The scheme is more suitable for robustness against attacks such as uniform affine transformation, reordering, and localized cropping and noise addition as compared to global attacks of noise addition and simplification. Robustness against global attacks can still be improved by increasing the hiding capacity. The technique can be generalized to 3D meshes, and we can say that it provides robust watermarks against the above mentioned attacks and topology altering attacks (e.g. retriangulation) as well.

The scheme was found vulnerable to global attacks as compared to other attacks. As a future initiative, we would like to improve the robustness of our methodology for such kind of attacks.

7. ACKNOWLEDGEMENTS

The work is supported in part by US Army Research Office grant 48645-MA and National Science Foundation under Grant No. 0237954 for the project CAREER: Animation Databases.

8. REFERENCES

- [1] Arya, S., and Mount, D. M. *Computational Geometry: Proximity and Location*. In *The Handbook of Data Structures and Applications*, eds. D. Mehta and S. Sahni, Chapman & Hall/CRC, Boca Raton, 2005, 63.1-63.22
- [2] Autodesk - Maya: <http://www.autodesk.com>
- [3] Alface, P., and Macq, B. Blind watermarking of 3D Meshes Using Robust Feature Points Detection. In *Proceedings of International Conference on Image Processing 2005*, Volume 1, 11-14 Sept. 2005 Page(s):693 – 696.
- [4] Benedens, O. Geometry-Based Watermarking of 3D Models. *IEEE CG&A*, pp. 46-55, January/February 1999.
- [5] Benedens, O. Affine invariant watermarks for 3-D polygonal and NURBS based models. In *Proc. Information Security Workshop Wollongong*, Australia, 2000, pp. 20-21.
- [6] Benedens, O. Robust watermarking and affine registration of 3-D meshes. In *Proc. Information Hiding Noordwijkerhout*, the Netherlands, 2002, pp. 177-195.
- [7] Bors, A. Watermarking Mesh-Based Representations of 3-D Objects Using Local Moments. *Image Processing, IEEE Transactions on*, Volume: 15 Issue: 3 March 2006, Page(s): 687- 701.
- [8] Bors, A. Blind watermarking of 3D shapes using localized constraints. In *Proc. of 3D Data Processing, Visualization and Transmission*, 2004. 3DPVT 2004. Proc. 2nd International Symposium on, 6-9 Sept. 2004 Page(s):242 – 249.
- [9] Cayre, F., and Macq, B. Data hiding on 3-D triangle meshes. *Signal Processing, IEEE Transactions on*, Volume: 51 Issue: 4 Apr 2003, Page(s): 939- 949.
- [10] Chen, B., and Wornell, G. W. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. In *IEEE Trans. on Information Theory*, vol 47, pp. 1423-1443, May 2001.
- [11] Cotting, D., Weyrich, T., Pauly, M., and Gross, M. Robust Watermarking of Point-Sampled Geometry. In *Proceedings of International Conference on Shape Modeling and Applications 2004*, pp. 233-242, (SMI04, Genova, Italy, June 7-9, 2004).
- [12] Cormen, T.H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. Second Edition, ISBN-13: 978-0-262-03293-3.
- [13] Cox, I., Miller, M., and Bloom, J. *Digital Watermarking: Principles & Practice* (The Morgan Kaufmann Series in Multimedia and Information Systems), ISBN – 1558607145, 2001.
- [14] Fast Scan 3D <http://www.fastscan3d.com/download/samples>
- [15] Harte, T., and Bors, A. Watermarking 3D models. *Image Processing, 2002. Proceedings. 2002 International Conference on*, Volume 3, 24-28 June 2002 Page(s):661 - 664 vol.3.
- [16] Kanai, S., Date, H., and Kishinami, T. Digital watermarking for 3-D polygons using multiresolution wavelet decomposition. In *Proc. 6th IFIP WG 6.2 GEO-6 Tokyo*, Japan, 1998, pp. 296-307.
- [17] Large Geometric Models Archive: http://www.static.cc.gatech.edu/projects/large_models/
- [18] Praun, E., Hoppe, H., and Finkelstein, A. Robust mesh watermarking. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 49--56, August 1999.
- [19] Petitcolas, F., Anderson, R. J., and Kuhn, M. G. Information Hiding: A Survey. In *Proc. IEEE special issue on Protection of Multimedia Content*, 1999.
- [20] Ohbuchi, R., Masuda, H., and Aono, M. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE J. Select. Areas Commun.* vol. 16, pp. 551-560, 1998.
- [21] Ohbuchi, R., Mukaiyama, A., and Takahashi. Watermarking a 3D Shape Model Defined as a Point Set. In *Proceeding of CW 2004*: 392-399.
- [22] Uccheddu, F., Corsini, M., and Barni, M. Wavelet-Based Blind Watermarking of 3D Models. In *Proceedings of the 6th ACM MM & Security workshop 2004*, Pages: 143 – 154, 2004.
- [23] Zafeiriou, S., Tefas, A., and Pitas, I. Blind robust watermarking schemes for copyright protection of 3D mesh objects. *Visualization and Computer Graphics, IEEE Transactions on*, Volume 11, Issue 5, Sept.-Oct. 2005 Page(s):596 – 607.