3-D CONTENT PROTECTION TECHNIQUES

by

Parag Agarwal

APPROVED BY SUPERVISORY COMMITTEE:

_____

Dr. Balakrishnan Prabhakaran, Chair

_____

Dr. Bhavani Thuraisingham

_____

Dr. Sergey Bereg

_____

Dr. Guo Xiaohu

In Loving memory of my grand mother,

Mrs. Lakshmi Shah

3-D CONTENT PROTECTION TECHNIQUES

by

PARAG AGARWAL, B.Tech, M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2007

PREFACE

This dissertation was produced in accordance with guidelines which permit the inclusion as part of the dissertation the next of an original paper or papers submitted for publication. The dissertation must still confirm to all other requirements explained in the "Guide for the Preparation of Master's Thesis and Doctoral Dissertations at The University of Texas at Dallas". It must include a comprehensive abstract, a full introduction and literature review and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this dissertation to include as chapters authentic copies for papers already published, provided these meet type size, margin and legibility requirements. In such cases, connecting texts which provide logical bridges between different manuscripts are mandatory. Where the student is not the sole author of the manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the student's contribution to the work and acknowledging the contribution of other author(s). The signatures of the supervising Committee which precede all other material in the dissertation attest to the accuracy of this statement.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the guidance and support from my advisor Dr. Balakrishnan Prabhakaran. I am very grateful for his faith, encouragement, inspiration, and constructive criticism on my research work. Working with him has given me a tremendous exposure in writing research proposals, mentor students, applying research to business models, and constructive collaboration in research.

I would also sincerely thank Dr. Ravi Prakash for his teachings, support and constructive criticism, as it has taught me to be aggressive and rigorous in research. I really appreciate the learning in various fields due efforts from professors in my school. In particular, I would like to thank Dr. G. R. Dattatareya, Dr. Bhavani Thuraisingham, Dr. Sergey Bereg, Dr. Balaji Raghavachari, Dr. Kang Zhang, Dr. Hal Sudborough, Dr. S. Q. Zheng, Dr. R. N. Uma, Dr. Xiaohu Guo, Dr. Gopal Gupta, and Dr. Kamil Sarac.

Working in Multimedia and Networking lab has been a learning experience. Special thanks to Gaurav Pradhan, Ziying, Parin Shah, Yohan Jin, Ketaki Adi, Puneet Maheshwari, Manoj Pawar, Dr. Hui Li, Dr. Ming Li, Sameer Agrawal and Magesh Panchanathan for the tremendous help offered in terms of data sets, discussions, and multimedia resources. In particular, sincere thanks to my friends Kalpesh Sutaria, Ketaki Adi, Puneet Maheshwari, Venkat Krishnan, Sagar Naik, Rohan Mehta, Ketaki Adi, Tanushree Jana and Shivani Bansal for all the help offered in reviewing my research work.

All my family members have been instrumental in providing support emotionally and financially, during my PhD - my grand parents, parents, my wife Sapna, brother Anurag

and his family, Mahendra Kumar Gupta and his family, Shachin-Alka-Pulkit, Dr. Seema Agarwal, and everyone at Allahabad, New Delhi and Meerut.

My friends offered me great strength to pass this phase and I would like to thank all of them for their constant support – Mayank Rajawat, Anshoo Sharma, Chetan Tare, Gautam Bhuyan, Shaik Yunus Saleem, Ravi Kashyap, Prakash Baskaran, Shailendra Gaur, Iyer Subramanian, Prashant Pachouri, Karthik Anantharaman, Jahnvi Verma, Vineet Tyagi, Aravind Rawat, Rajiv Jayant, Ajay Sinha, Mukesh Singhal, Deepali Goel, Hemant Mohan, Sandeep Jain, Nishant Mittal, Anku Magotra, Avinash Agrawal, Snehal Chokhandre, Siddharth Swaroop, Shashidhar Gandham, Vaibhav Chakki, Siddan Gauda, Shreyas-Swapnil, Shambhu Suman, Dheeraj Sharma, and Vipul Jha.

November, 2007.

3-D CONTENT PROTECTION TECHNIQUES

Publication No. _____

Parag Agarwal, Ph.D.
The University of Texas at Dallas, 2007

Supervising Professor: Dr. B. Prabhakaran

3-D models can be represented as meshes and point clouds, whereas 3-D motion data is defined by time varying 3-D points for every human joint or as a matrix which is multivariate and multi-attribute. Representation, data processing operations and dimensionality of these data sets makes it difficult to reuse of existing content protection (copyright protection and tamper detection-correction) watermarking and finger printing based methods related to images, video, text and audio. Therefore, this dissertation addresses the issues related to the design of schemes for 3-D content protection. Copyright protection techniques require robust watermark to withstand data processing operations, whereas watermarking and finger printing schemes aim to accurately identify tampering due to data processing operations.

To handle the case of robust watermarks and build generic schemes for different representation of 3D contents, clustering based spatial robust blind watermarking mechanisms have been proposed. In order to encode watermark related bits for a given set of n points with $n!$ possible orders, we need to find an order. To find such an order, ordered groups of clusters of 3-D points are identified on the basis of time for motion data and proximity in 3-D space for 3-D models. Inside the clusters, 3-D based scalar quantities order

the points locally and bits can be encoded or decoded using proposed extensions to 3-D quantization index modulations. The schemes are analyzed for robustness against uniform affine transforms (scaling, rotation and translation), cropping and reordering. Comparatively, the schemes are less robust against randomized noise addition and simplification attacks. These can be improved by using benchmarking in case of motion data, and encoding more bits per point for 3-D models. The encoding schemes can be customized to achieve high hiding capacity, imperceptible watermarks and security.

For tamper detection of motion data, large imperceptible fragile watermarks are encoded inside time based clusters, using 3-point and 1-point quantization index modulation based bit encoding. The watermarks can accurately detect and classify attacks such as order reversal, uniform affine transforms and noise addition. This scheme is improved by encoding watermarks in the localized domain, and benchmarking the bit encoding scheme. Watermarks are not lost during semantic invariant operations (uniform affine transform and noise additions), meanwhile detecting attacks such as noise addition, affine transforms and order reversal. Further, tamper detection probable correction methodology has been proposed that combines fragile watermarking and finger-printing. These schemes identify shuffling attacks on rows, columns or their elements and correct them. Random noise addition attacks can be detected using fragile watermarks and corrected using interpolation.

For tamper detection in 3-D models, the skeleton of the 3-D model is authenticated against cropping and noise addition attack, using a customized version of the clustering based approach for 3-D models. Watermarks are not lost during progressive compression-decompression and uniform affine transform, when the skeleton stays unaltered.

TABLE OF CONTENTS

LIST OF FIGURES

Number Page

CHAPTER 1

CHAPTER 2

CHAPTER 5

CHAPTER 6

LIST OF TABLES

Number Page

CHAPTER 2

CHAPTER 3

CHAPTER 1

INTRODUCTION

1.1 Background

3-D Digital content such as 3-D models and 3-D motion capture data are readily being used in the entertainment industry (e.g. animations, video games, movies and commercials), engineering design (e.g. CAD, CAM, MCAD) and medical informatics (e.g. 3-D Images, biomechanical research). 3-D models can be created by using laser scanning devices or modeling tools such as AutoDesk's Maya, and can be represented as meshes (geometry and topology) and point clouds (geometry), see Figure 1.1. Motion capture data is collected using sophisticated motion capture facilities (e.g. VICON), wherein cameras capture the motion of humans that wear non-reflective suits covered with optical markers. The motion of markers is approximated to the motion of human joints giving us motion capture data represented as time varying translational and/or rotational information related to 19 human joints, see Figure 1.2. Since these data sets involve investment in terms of money, time and human effort preserving them is important. Loss due to theft and illegal usage result in their piracy or their corruption can result in loss of information important to medical science or patients being cured.

We can secure these data sets by placing them in secure servers and devise secure cryptographic application oriented protocols. Even copy control mechanisms can prevent illegal copying and usage of the data set. In spite of such secure mechanisms, we cannot guarantee that a secure system will never be compromised or a data could be tampered due to

user or machine related errors. Tampering of data can be detected using conventional error detection approaches such as checksums, cyclic redundancy check (CRC), MD5 etc. However, since 3-D data can be modified due to user related semantic invariant data processing (e.g. lossy compression or uniform affine transforms); the error detection may result in false alarms.

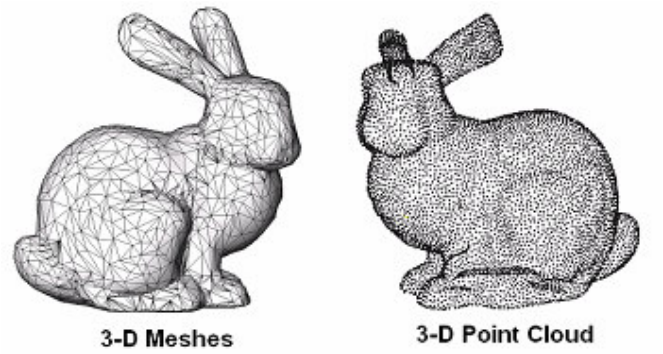


Figure 1.1. 3-D Model Representations

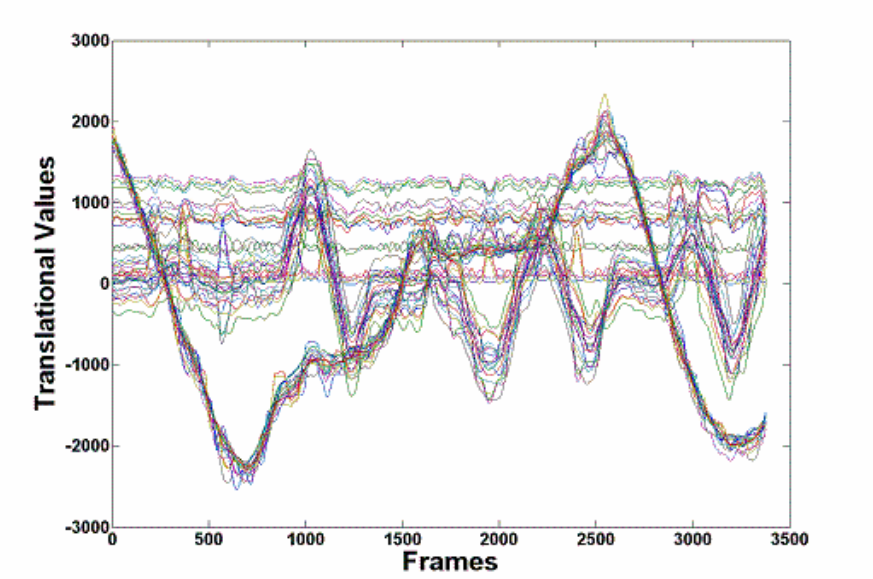


Figure 1.2. Motion Capture Data – Translational values of 19 joints

To counter the above problems, digital watermarking is one of the methods that is used to content protect digital media. The idea is to encode patterns *watermark* inside data. The presence of the watermark uniquely identifies the media and its owner, thus resolving the

copyright issue. Also, in case the meaning of the data set is tampered with, the loss of watermark will help as an evidence for tampering, thus authenticating its correctness. Another method to authenticate the data set is to generate signatures which are patterns finger-prints from the data set. Comparison of signatures related to original and tampered data indicate tampering.

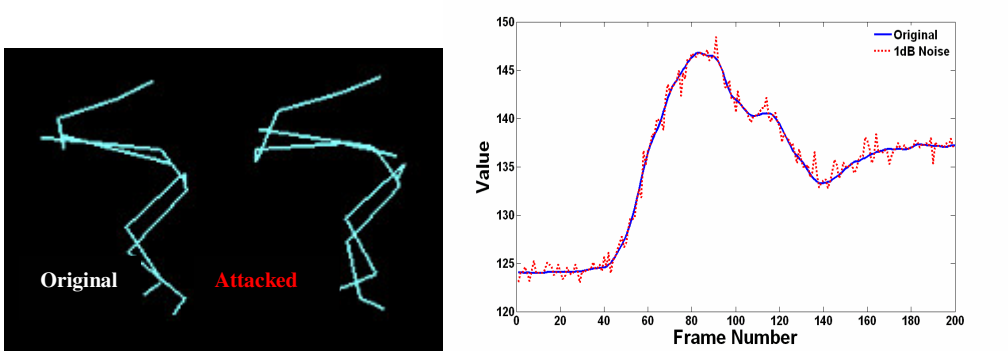


Figure 1.3. Tampering of Motion Data – Actions and motion signal

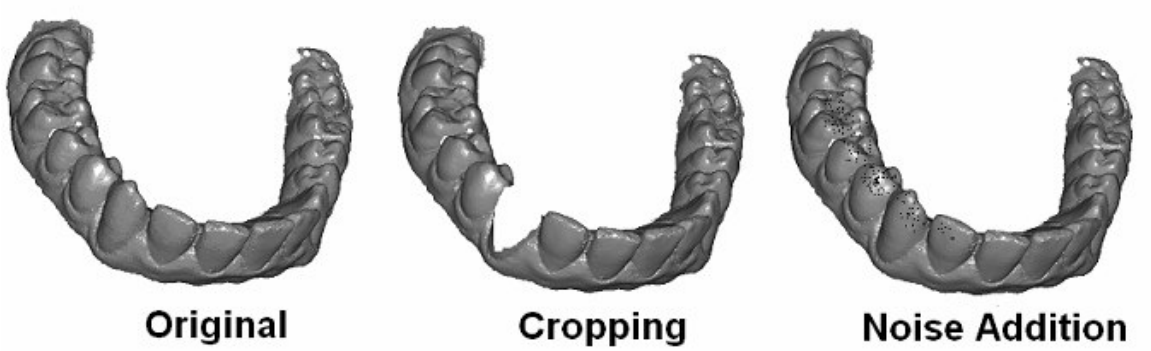


Figure 1.4. Tooth Models Being Tampered

Data processing varies from motion data to 3-D models and might lead to change in semantics of the data set as well. For example, noise added changes the meaning of motion (see Figure 1.3), while 3-D models (see Figure 1.4) under go cropping and noise addition attacks which changes their shape. Watermarks and finger prints can detect any alterations due to data processing attack. Such watermarks can be fragile if they detect even slightest change to the data-set, or semi-fragile, if they only detect semantic varying changes. Meanwhile, copyright protection watermarking techniques requires the watermarks to be

robust against all kinds of data processing operations. So, their role is opposite to watermarks in comparison to copyright protection application. Representation, data processing operations and dimensionality of 3-D data-sets makes it difficult to reuse of existing content protection (copyright protection and tamper detection-correction) watermarking and finger printing based methods related to images, video, text and audio. Also, as discussed in Chapters 2-6, the state of art in 3-D content protection related to 3-D models and motion data does not cater to all the issues that need be addressed while designing robust watermarking or tamper detection and correction techniques. Therefore, we need novel mechanisms for 3-D content.

1.2 Problem Statement

The dissertation primarily focuses on content protection techniques 1) Watermarking techniques for Copyright protection, and 2) Watermarking and finger printing techniques for tamper detection and correction, for 3-D content such as 3-D models and 3-D motion data. The design of such techniques has related issues, especially for watermarking based techniques. However, as mentioned above, the nature of the watermarks (robust/fragile/semi-fragile) can be based on the application – tamper detection or copyright protection. In addition, during tamper detection, we may even want to identify the attack and correct it. Watermarking techniques developed should be *generic* to cater to different formats for different data types. Also, since watermarks are series of 0s and 1s in a sequence, we need to find an order for encoding this information. Given n data samples, we have $n!$ orders, and out of these, we need to find an ordering that can solve the following problems during copyright protection and tamper detection. The following sub-section discusses these aspects in detail.

1.2.1 Related Problems - Copyright Protection and Tamper Detection

To solve the problem of tamper detection and robust watermarks, we need to make sure the following problems need to be addressed.

- ***Minimal Distortions*:** Watermark addition adds distortions which can be visualized which can alter the meaning of the data set. For example, the motion defined for normal walking can turn into a limping walking motion. Such loss of meaning is not expected when we add watermarks, therefore the technique should always be meaning preserving or in other words distortions should be minimized resulting in imperceptible watermarks.
- ***High Hiding Capacity:*** The addition of more watermarks the better it is for the mechanism we are going to develop. This is true since for tamper detection we can add more watermarks and easily localize the attack. For copyright protection, we replicate the watermark which in turn helps in robustness since loss of watermark in one sub-part of the data set can be compensated by another sub-part. More watermarks imply high hiding capacity which can result in more distortions due to watermark addition. As a consequence, there is a higher chance of losing the meaning of the data set and compromising the imperceptibility of the watermark. So we need to develop schemes that can have higher hiding capacity and still maintain the watermark imperceptibility criteria.
- ***Storage efficiency – Blind Watermarking Schemes*:** Watermarking schemes can be categorized on the basis of watermark extraction. Private watermarking schemes require the watermark extraction process. Developing private schemes means we need to store the original data which would result in storage inefficiency. Also,

private mechanisms do not make sense in tamper detection, since we need to store the original data which can be tampered as well. To solve this problem we need to develop blind watermarking mechanisms that do not require the original data set for verification. In cases of authentication finger printing based mechanisms the information extracted for this purpose should not be a function of the size of the data sets.

- ***Generic Nature:*** The 3-D contents are represented by different formats i.e. motion data is represented by local and global formats, and 3-D models are represented by meshes and point clouds. For each content type, the method should be applicable to different formats.

Properties such as hiding capacity and watermark imperceptibility have inverse relations. Since adding watermarks implies adding noise to data, higher hiding capacity implies more watermarks being added and result extra distortions, resulting in perceptible. Also, if we try to reduce the numbers of replications by reducing distortions, it brings down the robustness of the watermark against cropping attacks. So, building a scheme that achieves all of the above poses a great challenge.

1.2.2 Robustness of Watermarks for Copyright Protection

Data editing operations can be used to remove the watermarks that are meant for proof of ownership. At times, users operating on data and can unintentionally remove the watermarks e.g. lossy compression and decompression. So, we have to design a scheme with the aim to build robustness against intentional or unintentional operations, and these can be broadly classified as below:

- *Value Change Attack*: This occurs when watermarks are encoded in the absolute values of the data samples, and are lost when these values change. This can occur during noise addition due to lossy compression, interpolation of motion capture data, stylizing of 3-D models, or uniform affine transformations (scaling, rotation and translation) identified as orientation of the data set in 3-D space, or even format conversion (global to local) for motion capture data specifically.

- *Data Loss Attack*: This can occur when data samples are removed in the data set in a local or global manner. Localized attacks are termed as cropping and global are defined as simplification. Simplification attacks for 3-D models imply down resolution which occurs during progressive compression, and for motion capture data it implies curve simplification wherein the time varying rotational and translation information represented as curves is simplified.

- *Reordering Attack*: These attacks impact watermarks encoded using the order of occurrence of data in the files. For example, for motion capture data, it represents time or frames. However, for 3-D models it represents the vertices or points. For 3-D models represented as meshes, this can be also considered as topology change, since we change the connectivity of vertices in mesh representations of 3-D models, using re-meshing or re-triangulation. In case, the watermarking methodology uses the connectivity (topology) information to retrieve the watermark encoded in vertices (points), the extraction process will fail.

We need to make sure that only the indented person is able to detect the watermark. This is necessary as knowing the location of the watermark in the data set makes it vulnerable and

less robust. In other words, the privacy of the watermarks needs to be a secret and it helps induce *security* that helps in improving robustness.

1.2.3 Classification and Accuracy in Tamper Detection

As observed above, data sets can be subjected to different attacks. Attacks might alter the semantics of the data (e.g. deformation in shape of 3-D model, or change in actions related to motion capture data, or cropping or simplification of data). Also, some attacks do not alter the semantics e.g. lossy compression or uniform affine transforms. So, in order to authenticate the data set, any technique that is designed should be able to classify the attack that took place by detecting it with high accuracy, and if possible correct the error induced due to the attack. Classification helps since semantic invariant attacks cannot be identified as tampering always. Schemes that cannot distinguish between semantic invariant and variant attacks lead to false alarms. Also, classification of attacks tells us whether the attack can be corrected or not. For example, a cropping attack on 3-D model or motion cannot be reversed, or when actions of humans are reversed in time, this attack can be reversed by doing a reverse operation with respect to time. So, to achieve such a feature, we need to model the watermarks and fingerprints that are able to classify attack accurately and also correct them is possible.

1.3 Proposed Approaches

In order to solve the copyright problem, we propose clustering based bit encoding methods that arrange 3-D information into groups ‘clusters’ and try to find an order to encode the bits. The techniques differ from each other in terms of criteria for clustering, ordering criteria for bit encoding and the choice of bit encoding inside each cluster.

- **3-D Motion Data:** The technique tries to solve the problem by using divide and conquer by preserving watermarks at joint level, which in turn builds robustness at all the 19 human joint's motion. For every joint's positional (translational) information represented as points arranged in the order of time, we find clusters in the same order. Within every cluster, we find two sets of points – invariant set and encoding point set. Ordering for encoding inside each cluster is obtained using the invariant set and using a scalar quantity which is robust against uniform affine transforms. Once ordering is achieved, bits are encoded by perturbing the encoding points by using proposed bit encoding scenes that are extensions to 3-D quantization index modulation. Since, watermarks can be of any number of bits; these can use any number of clusters in the order of occurrence in time. Security is induced by observing random embedding points 'distances' between clusters, and also the randomness in encoding points. During the watermark extraction phase, we search for a given pattern of clusters and encoding points. The choice of quantization index modulations helps resolve the problem of watermark imperceptibility and high hiding capacity which is proven to be dependent on cluster size. The ordering and bit encoding schemes makes the watermarked bits 100% robust against uniform affine transforms, and this makes it also generic for local and global formats. Robustness against reordering is proven to be robust using larger sized clusters. Robustness against noise addition is controlled by using benchmarking, but the scheme is vulnerable to high intensity noise additions. Robustness against joint cropping attacks is 100%, but time based cropping is dependent on number of point used to encode the watermark, which must be twice, in order to guarantee robustness.

- **3-D Models**: The proposed approach is to build a generic solution for watermarking meshes and point clouds by choosing their geometry as the target for watermark embedding and extraction. The solution tries for form clusters of points (vertices). Inside each cluster, we identify a structure in order to identify points for encoding purpose. Since, we need to define an order to encode the watermark, a local ordering is defined for each cluster and clusters are ordered globally as well. The process of global ordering, groups the clusters in order to encode all the bits related to the watermark. For bit encoding, we use an extension of 3-D quantization index modulation by exploiting the structure identified for each cluster, and perturbing the encoding points. Since ordering and bit encoding are based on scalar quantities, it makes the scheme robust against uniform affine transforms. Also, proximity based grouping of clusters helps build robustness against localized attacks (noise addition and cropping). However, the scheme does suffer from global randomized attacks (noise addition and simplification). But, this can be dealt by using generalized version of quantization index modulation, which increases hiding capacity, resulting in smaller groups of clusters, reducing the chances of watermarks being destroyed. The bit encoding scheme is also benchmarked to reduce distortions and achieve the imperceptibility criteria.

The following tamper detection-correction involves watermarking and finger printing:

- **3-D Motion Data**: We have developed two approaches as mentioned below:
  1) *Tamper Detection using watermarks:* For every joint based translational information, we encode sequence of bits related to a watermark in a sequence of clusters arranged in the order of time. Each cluster contains four points, wherein

three points are used for 3-point encoding and the fourth point is used for 1-point encoding. For 3-point and 1-bit encoding, we encode the same bit by reusing a 3-D and 1-D quantization index modulation technique. During a uniform affine transform, the 1-D bits are flipped, while the 3-point bits stay unchanged, indicating the nature of the attack. Order reversal is easy to identify since we can always reverse the data in time and identify the watermark. All other attacks are identified as noise addition attacks. Since this scheme is fragile in nature, we better is by taking transform it by watermarking only local format representation. This makes the technique robust against semantic preserving uniform affine transforms, but it does capture attacks, where different joints are given different affine transforms. To differentiate between semantic altering and preserving attacks, we tend to find thresholds for the amount of noise that can be added to lose the watermark.

2) *Tamper detection and correction using watermarks and fingerprints*: The basic idea is to extract reference patterns (fingerprints) identified by row and column hash functions which take input parameters as row and column elements of the motion data matrix. By observing change in the values related to row and column function, we can detect and predict the attack pattern, and reverse it. The patterns (fingerprints) resulting from hash functions help in error detection by identifying the type of attack such as row shuffling, column shuffling, row element shuffling, column element shuffling and their combinations. Random attacks that change data element values are detected by change in fragile watermarks embedded in data elements. Finger prints help in solving the attack reversal, such as column or

row shuffling, whereas watermarking helps in reversing attacks such as column element or row element shuffling. The combination of these attacks can be corrected by combining watermarking and fingerprinting. Random attacks can be identified with change in watermark based tags related to the impacted data element, and can be corrected approximately by using interpolation.

- **3-D Models:** The technique tries to authenticate the shape of a 3-D mesh by identifying a skeleton defined using point or vertices that are invariant during progressive compression and decompression mechanisms. Watermarks are encoded in the points by customizing our watermarking algorithm for copyright protection for geometry. The difference lies in choosing clusters that are situated farthest from each other. The technique helps in building watermarks that do not change due to semantic invariant uniform affine transforms, and compression and decompression. But during tampering such as noise addition or cropping, watermarks are lost and we can authenticate the shape of the 3-D model.

1.4 Contributions

The contributions in the dissertation are described as follows:

- Clustering and bit encoding mechanisms to achieve spatial robust blind watermarking mechanism for copyright protection of 3-D models (point clouds and meshes) and 3-D motion capture data. The schemes are shown to have a high hiding capacity (depending on cluster size for motion data and number of bits encoded per encoding per encoding point in case of 3-D models) with controllable distortions i.e. watermarks imperceptible. The watermark is secure and robust against uniform affine transforms, noise addition, cropping and reordering. For 3-D models specifically,

watermarks have robustness against topology change and simplification attacks as well. The running time complexities are as follows: 1) Watermark embedding - O (n log n) for 3-D models and 3-D motion data 2) Watermark extraction – O (n log n) for 3-D models and O (n^2 log n) for 3-D motion data.

- Clustering based bit encoding to realize a fragile watermarking scheme that can detect tampering and classify attacks such as uniform affine transforms, noise additions and order reversal. The watermarks encoded are imperceptible, and are shown to be highly to change in bit information due to choice of bit encoding scale and watermark length.

- A localized transform based semi-fragile watermarking technique wherein imperceptible watermarks are not lost during semantic invariant operations such as uniform affine transforms and noise additions. Tampering can be detected when joints are given different uniform affine transforms or noise is added to change the semantics of the actions in a motion.

- Combination of fragile watermarking and finger printing method that can detect tampering such as shuffling attacks and random noise addition to motion data matrix. The technique is shown to reverse shuffling attacks only. The running time complexity of the scheme can be between O(n *log* n) to O (n!).

- Watermarking mechanism to authenticate the skeleton of 3-D models (meshes). The watermarks are shown to be semi-fragile, since the watermarks are not lost during progressive compression-decompression and uniform affine transforms, but are lost during shape altering attacks such as cropping and noise addition. This work was

done in collaboration with *Puneet Maheshwari*, who contributed by identifying border points that stay invariant during progressive compression mechanism.

Most of the schemes are generic i.e. they cater to all formats in case of motion data (global and local formats), and meshes and point clouds for 3-D models. The authentication for 3-D models is specific for meshes only.

1.5 Dissertation Outline

Rest of the dissertation is organized as follows:

- Chapter 2 and Chapter 3 cover the schemes for copyright protection of motion capture data and 3-models respectively.
- Chapter 4 is on fragile watermarking of motion capture data and an improvement on it using semi-fragile watermarking.
- Chapter 5 is based on a technique that combines watermarking and finger printing to in solve the problem of tamper detection and correction in motion data
- Chapter 6 deals with authentication mechanisms of 3-D models.
- Chapter 7 concludes and presents future work.

CHAPTER 2

ROBUST BLIND WATERMARKING OF 3-D MOTION DATA

Parag Agarwal and Balakrishnan Prabhakaran

Computer Science Department, EC 31

The University of Texas at Dallas

800 WEST CAMPELL RD.

Richardson, Texas 75080-3021

2.1 Introduction

The advent of Motion Capture (mocap) systems such as Vicon [2.45] has brought in applications like animation (games, films & TV, education). The above applications deal with motion analysis or reusability, and can benefit from having a large repository of 3-D human motions. In cases where data is stolen, we incur losses in terms of effort, time and money. Theft of motion data can lead to copyright issues. Digital watermarking is a solution to these problems, where the presence of a digital watermark helps in copyright protection. 3-D Mocap data streams consist of motion information related to human joints, and can be represented by their translational and/or rotational information. The translational and rotational information defines the attributes of the joints that vary with time, making the data multivariate. The process of encoding and decoding motion data samples poses problems related to watermarking scheme properties such as imperceptibility of the watermark, hiding capacity, security, robustness against data dependent attacks, and efficiency in terms of storage.

Problems related to imperceptibility and robustness are analyzed on the basis of distortions and watermark loss as follows:

Distortions in semantics of data: Addition of watermarks distorts the original data. The visibility of distortions is due to change in the semantics of the motion, and such as change will fail the imperceptibility criteria of the watermarking scheme. This can be visualized in Figure 2.1, where the human body appears distorted and the motion data signal appears changed from the original for perceptible watermarks are perceptible, as compared to the case where watermark is imperceptible.

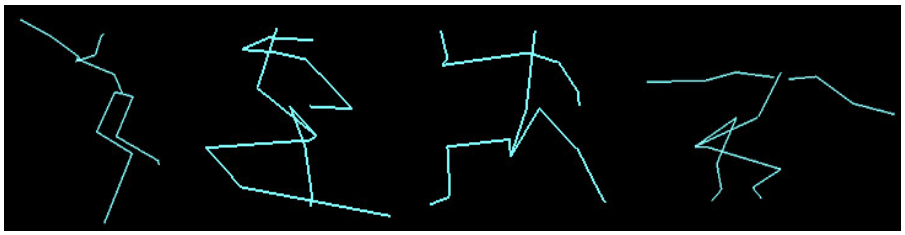


Figure 2.1. Snap-shots: Distortion due to watermarking

Watermark loss due to motion editing: Motion data can be subject to motion editing (warping) [2.10, 2.17, 2.18, 2.19, 2.24, 2.25, 2.35 and 2.44] operations done intentionally (adversary) or unintentionally (user – an artist) by using motion editing software such as Motion Builder [2.1]. Editing operations result in loss of the watermark, which defeats the purpose of robustness. The motion data editing attacks can be categorized as follows:

- *Sample Loss*: Operations such as cropping and curve simplification can result in loss of data elements that were associated with the watermark. Cropping results in loss of sub-parts of the motion data. Simplification can also be visualized as cropping of data elements at random locations. The cropped data might contain only a sub-part of the watermark or no watermark in the worst case, resulting in loss of the watermark.
- *Value change of samples*: Since watermark is associated with the data set values, any change in values of the motion data set alters the watermark. Value changes can take place as a result of additive noise (e.g. lossy compression), smoothing, affine transforms (translation, scaling, and rotation), and format conversions (global to local format in Vicon IQ [2.45]). Format conversion results in a change of co-ordinate systems, and thus, can be categorized under uniform affine transformations.
- *Reordering of samples*: The order of samples can be altered. For example, if we reverse the order of time series data, the action represented by the motion will change in

direction. Since the decoding process is order dependent, a change in the order creates problems in the decoding process.

It has been observed [2.13] that imperceptibility, hiding capacity, and robustness are in an inverse relation to each other. Watermark replication is a mechanism of achieving robustness. However, replication can induce more distortions resulting in loss of motion data semantics. As a result, watermark replication is restricted and this impacts the hiding capacity 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 secret and watermark for verification purpose. Therefore, it is necessary that the design of a scheme must be storage efficient i.e. blind. The scheme also needs to be secure, in order to guarantee that an adversary should not be able to locate the watermark and remove it.

Human motion comprises huge combinations of varying information, which could lead to different hiding capacity and imperceptibility factors. Finding a secure blind watermarking mechanism that is robust to attacks and incurs minimal distortions to the data set is a challenge.

2.1.1 Related Work and Issues

Different blind watermarking schemes have been developed for multimedia data such as images, audio, video and 3-D mesh objects. These can be categorized as spatial or transform domain techniques. Techniques based on spatial domain [2.3, 2.15, 2.43, 2.16, 2.48, 2.49,

2.50] operate on the data set without taking any transforms, as compared to transform domain techniques [2.14, 2.35, 2.46, 2.23] that use spread spectrum. Transform based techniques rely on frequency components, which reduces the hiding capacity. However, they are robust against noise addition, cropping attacks and produce less distortion.

Motion data is spatio-temporal and we can potentially use ideas related to other data such as video [2.4, 2.30, 2.37, 2.39, 2.41, 2.50, and 2.51] and audio [2.7, 2.9, 2.21, 2.23, 2.27, 2.28, 2.32, 2.33, 2.38, 2.40, 2.42, 2.43, 2.47, and 2.48] where information changes with time. Audio data can watermark using spatial or transform based methods, wherein we perturb the value information or the frequency component representing the transform. For video, we can watermark by encoding each frame by applying image based watermarking technique, or by perturbing the motion vectors defined by the difference between each frame. The fundamental problem in reusing the techniques for video is that the representation is different. However, audio data watermarking can be applied to motion data since it is multivariate i.e. we apply audio watermarking techniques for different variables (rotational or translational information) of the data that vary with time. The idea is discouraged since rotational and translational information change when we convert data formats i.e. local to global (change in co-ordinate system by using uniform affine transform), and this would require invariance for such a transform, which is not defined in audio based watermarking Applying image based watermarking techniques is not possible since images are represented as pixels, and image data does not vary with time, as in our case. Watermarking methodology for copyright protection of 3-D model might serve as a candidate, since motion data is time-varying 3-D data. Techniques for 3-D models [2.2, 2.6, 2.8, 2.11, 2.12, 2.16, 2.20, 2.26, 2.34, 2.35, and 2.49] perturb vertex (geometric) (and/or) topology information to encode and

decode the watermark. The topology information represents the connectivity between vertices. Based on topology information vertices can be chosen and perturbed. Another way is to take multi-resolution transforms and encode the watermark by perturbing the frequency components. These techniques are shown to be robust against uniform affine transforms, noise addition, cropping and simplification. The problem in applying these techniques to motion data is that these are not defined to solve problems related to time-varying 3-D information. Also, motion data consists on translational and rotational information for every human joint. We could at least solve the problem for translation information by assuming them as vertices and projecting them in 3-D space, and running the 3-D model watermarking algorithm. This would be possible if we the vertices were unique, which is not a case for motion data, as joints can occupy same space in time. In addition, we do not know how to take care of rotational information related to each joint.

Recent efforts to solve the problem related to watermarking mocap data have lead to frequency transform techniques that use discrete cosine transform [2.46] or multi-resolution analysis [2.22] per time varying attribute, in order to embed and extract the watermark. These techniques are basically private in nature where we require the original data to determine the presence of watermark. This makes the technique storage inefficient.

To summarize, spatio-temporal methods cannot solve the problem of watermarking mocap data due to representation and dimensionality problem. Although there are several blind watermarking methods for 3-D space, these do not define a way to watermark 3-D time varying data. So, we need a new methodology that could cater to the problem of watermarking mocap data that is robust, blind, and secure and offers high hiding capacity.

2.1.2 Contributions

To solve the above mentioned problems, the dissertation contributes by suggesting a methodology that embeds/extracts watermarks in the trajectory of the 19 human joint's that represent the motion data. For each such trajectory, clusters of 3-D points are identified on the basis of time. The watermark related bits are encoded/decoded in these clusters. The points within these clusters are fist ordered on the basis of scalar quantities that are calculated from positional information of 3-D points of the cluster. Finally, the bits can be encoded/decoded using proposed bit encoding schemes – '*Euclidian distance based encoding*' and or by reusing our existing technique Extended Macro Embedding procedure '*Projection based encoding*', proposed in [2.3]. The choice of bit encoding scheme gives us imperceptible watermarks, and security is induced by adding randomness in encoding process. The dissertation calculates the hiding capacity offered by the watermarking scheme based on embedding distance and cluster size. We also give the upper and lower bounds on the hiding capacity. The time complexity for watermark embedding and extraction is estimated as $O(n \log n)$ and $O(n^2 \log n)$ respectively. The dissertation gives theoretical and experimental analyses on the robustness against uniform affine transforms, noise additions, cropping and reordering attacks.

The technical contributions are identified as follows:

- Methodology that chooses human joint's trajectory for embedding/extracting watermarks within clusters of 3-D points, such that the watermarks are robust, blind, imperceptible, secure and result in high hiding capacity. The robustness is against uniform affine transforms, noise additions, cropping and reordering attacks.

- 3-D QIM method for bit encoding: For bit encoding process inside the cluster, the dissertation proposes an extension to 3-D Quantization index modulation (3-D QIM) – '*Euclidian distance based encoding*'.
- The technique is generic as it is applicable to existing formats due to its robustness against uniform affine transforms. This implies that a watermark survives inter-format conversions. It is more generic than our previous method [2.3] where encoding was being done by identifying clusters of triangle in 3-D, whereas we identify points and can encode either as triangles or just using them as points in 3-D space.

The problem of protecting motion capture data is related to protecting the content in the 3-D domain. Once this information is applied to 3-D characters and we generate an animation to obtain a video, the watermarked information can be lost as motion is now defined by frame playing in a video. Solving such a problem is out of the scope of our dissertation as it is a different domain and way of representing motion. So, we limit to the problem of embedding and extracting watermarking 3-D domain.

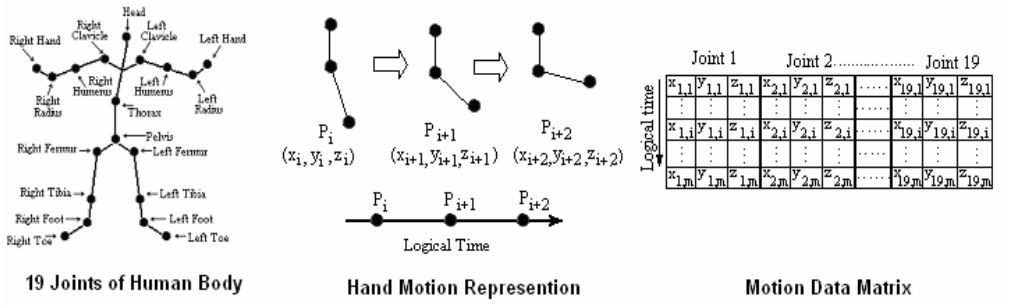


Figure 2.2. Motion Capture Data Representation

2.2 Scheme Design

Motion capture data consists of motion information about the 19 joints of the human body (see Figure 2.1). Every joint dataset consists of 3-D information described as positional

information (x, y, z) and rotational information (Euler angles) that varies in time. We can term these as points in 3-D space. As an example, we can visualize the motion of hand joint shown in Figure 2.2, where positional information changes for the joint due to its motion. This can be described as a trajectory of a joint in 3-D space, and the different joints considered gives us a combination of their trajectory information represented as a matrix of motion data, where the columns represents joint information (x, y, z) for 19 joints and rows represent the varying positional information with time.

Motion Capture data can be represented in different formats [2.31] such as Acclaim (.amc and .af combination, Biovision format (.bva and .bvh), and Motion Analysis (.trc and .htr), VICON [2.45] (.csv – comma separated value). The output of a motion captures system in translational data (3-D points) of human joints based on a world origin. This can be represented in a more simplified way by representing data based on segments or limbs. Each limb of human skeleton is made up of a set of joints. So, we can observe that motion information consists of data defined for a set of joints. This data can be broadly categorized into global and local data formats. Global data formats consist of translational and rotational information based on world origin. Local format consists of translational and rotational information based on a fixed origin, which is generally a stable joint such as the pelvis. It is to be noted that both these formats are based on translational information of the set of joints that describe the motion of the human skeleton. It is also possible to convert one format to another as this involves a change in the co-ordinate system. For example, VICON IQ [2.45] software allows exporting of motion capture data in (comma separated format) in local and global formats. Once exported, these are inter-convertible local data can be generated by localizing global data by fixing the origin to a stable joint such as the pelvis. In such

conversions the stable joint (origin) still preserves information related to translation and rotation with respect to world original, and we can use this information to reverse the local information to global information.

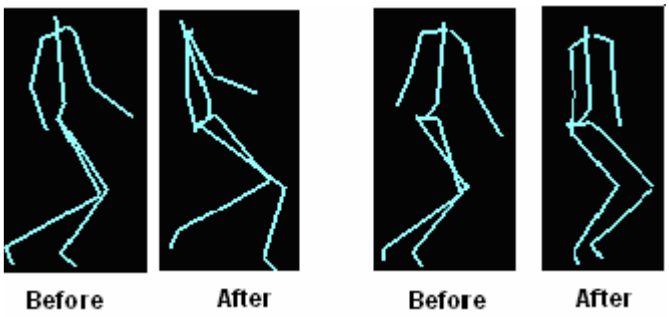


Figure 2.3. Snapshots of Change in Orientation Attacks due to uniform affine transform per joint

In order to watermark the motion data, we can choose to watermark the translation (and/or) rotational information related to the joints. As observed above, motion capture data is inconvertible into different formats (global to local and vice-versa), it would not be advisable to watermark rotational and translational data pertaining to a given co-ordinate system. So, our choice of watermarking is based on criteria that make it independent of any format conversions i.e. co-ordinate system. Therefore, we choose to watermark the trajectory of motion which does not change in time and in any co-ordinate system. To do so, we watermark the translation (positional) information of the motion data for a co-ordinate system (x, y, z). Rotational information (Euler angles) is derived using the translational information based on an origin. However, the reverse is not possible. So, watermarking the translational information itself watermarks the rotational information. Based on the discussion so far, since we choose to watermark the trajectory, for given 19 joints our problem is to come up with a scheme that can watermark the trajectory in 3-D space and the watermarks are invariant to format conversions.

We watermark the trajectory of each joint separately as it helps in building robustness against cropping attacks wherein joint information is cropped separately and used illegally. Since we try to watermark every joint, it gives us the ability to detect the watermark for every joint. So, the technique is robust against cropping attacks on a joint level. This makes it also robust against cropping attacks where a set of joints (e.g. right leg) are cropped, since we can prove the presence of watermark for every joint. Another idea for watermarking is to use a set of joints for watermarking by combining their data. However, this limits robustness against cropping attacks. Also, if joint motion is warped by changing their orientation using uniform affine transforms in 3-D space as seen in Figure 2.3, the relationship used to watermark a set of joints will no longer be valid since the motion information related to the data would have changed. However, as we will observe, since our bit encoding is robust against uniform affine transform, such kind of warping would not impact the watermark, thus, making it a better choice.

So, our problem reduces to watermarking each joint's 3-D information and the end result gives us the complete watermarked motion data. Watermarks can be described as information that can be a digital signature or any other information. This information has been converted into a series of 0s and 1s and it is this bit sequence that needs to be encoded during watermark embedding and decoded for watermark detection. The following sub-sections explain how the embedding and detection of watermark are done.

2.2.1 Watermark Embedding

In order to watermark the trajectory of the joint, we identify clusters of 3-D positional information, which can also be described as *points* in 3-D space as seen in Figure 2.4. This can be visualized in Figure 2.5, where we identify two clusters on the basis of time. The

choice of clustering on the basis of time is beneficial in the case of cropping attacks where a sub-part of the trajectory is cropped and re-used illegally. This is due to the nature of such cropping attacks, since points ordered in time get cropped together. Once the clusters are identified, we add sequence of bits that represent the watermark. Each cluster has a hiding capacity associated with it, which informs the number of bits we can encode. Since the number of bits in a watermark can be more than the hiding capacity per cluster, we can choose more than one cluster. These clusters are chosen on the basis of their occurrence in time, since cropping attacks take point in time and the watermark needs to be preserved during such an attack. Once the clusters are identified, we need to encode watermarks in the clusters as discussed below.

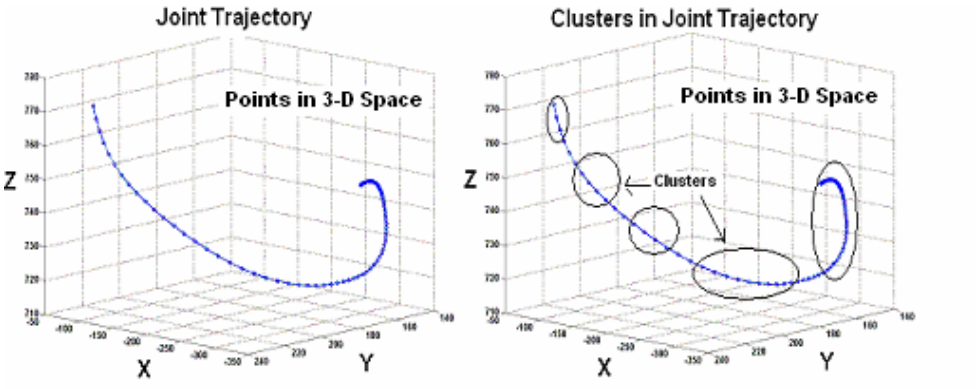


Figure 2.4. Clusters in trajectory of joint in 3-D space

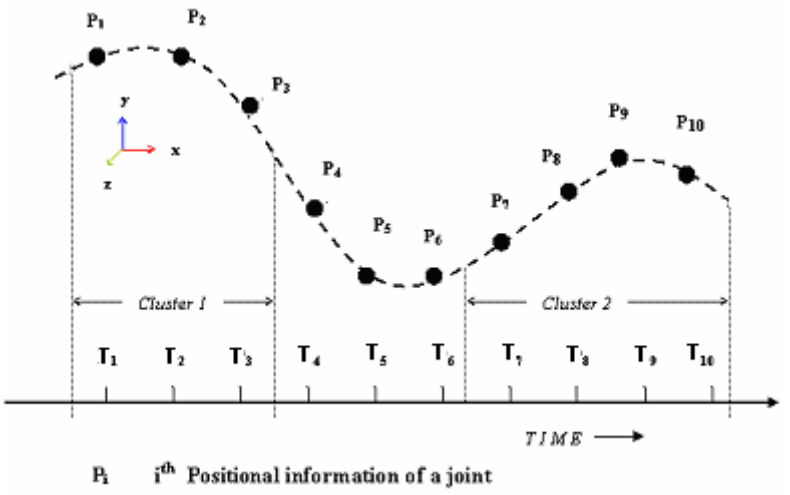


Figure 2.5. Clustering example for motion data showing joint 3-D positional information

2.2.1.1 Cluster Based Encoding. The idea of clustering is to use a divide and conquer strategy, where each cluster encodes bits that are robust against noise addition, uniform affine transformation and reordering attacks. The combined robustness of the clusters gives an overall robustness against these attacks. The problem of encoding in a cluster of size t (~ number of 3-D points) with B bits is done by projecting the 3-D points in 3-D space. For our method, since the encoding is to be robust against uniform affine transformation, we need at least three 3-D points [2.3]. The encoding is done by first identifying two point sets.

Definition 1. *Invariant points set:* 3-D points used as reference while encoding and their values are not perturbed.

Definition 2. *Encoding points set*: 3-D points which are used to hide information such that 1 bit is hidden per point.

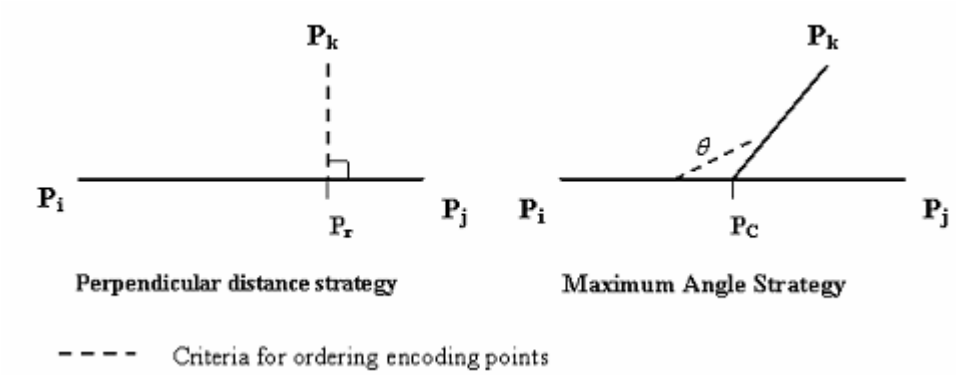


Figure 2.6. Scalar quantity used for ordering encoding points P_k using an invariant set $\{P_i, P_j\}$

Since we need at least three points to encode [2.3], two points stay invariant and one point is used to encode; the minimum size of the invariant set is 2. This also helps in maximizing the number of points used to encode bits in clusters. So, given a cluster of size t, we can encode maximum of ($B = t - 2$) bits. We observe that the choice of the invariant points is based on criteria that are invariant against uniform affine transformations, and reordering attack. This

choice can be based on a scalar quantity such as maximum or minimum Euclidian distance between the set of points, and any other such as the median of the Euclidian distances. So, we can generalize and say k^{th} largest Euclidian distance, in case of m pairs, where ($k < m$).

The next step is to identify the order in which the bits need to be encoded. This ordering should not be dependent on the order of occurrence of positional information in time as it would be vulnerable to reordering attack. So, we identify ordering criteria based on the 3-D information of the points belonging to the cluster.

The criterions are based on the following scalar quantities:

Perpendicular Distance: The Euclidian distance is described by the perpendicular dropped from the encoding point to the line joining invariant point set.

Maximum Angle: The maximum angle subtended by the line joining the centroid of the invariant points and the encoding point.

Figure 2.6 shows us this criterion, where for invariant set $\{P_i, P_j\}$ and encoding points represented by P_k. The perpendicular distance is the Euclidian distance $|P_kP_r|$, where P_r is the projection of point P_k on the line drawn between the invariant points P_i and P_j. The maximum angle is between the points P_i, P_C and P_k, where P_C represents the centroid of the points P_i and P_j. Once the order of points has been identified, we encode the respective bit in each encoding point (P_k). For encoding purpose, we modulate a scalar quantity using the encoding points such that the scalar quantity used for ordering and identifying the invariant set is not perturbed. This condition is necessary since we need to decode the bits representing the watermarks based on ordering of encoding points. This can be observed in the following discussion.

Projection based strategy: The scalar quantity used to encode the bit information is described by the perpendicular projection (P_r) of encoding point P_k on the line joining invariant set points P_i and P_j. The line is divided into equal number of intervals and labeled with 0s and 1s. The interval in which the projection lies represents the bit encoded. In case, we need to toggle the bit, we displace P_k parallel to the line joining the invariant points P_i and P_j such that the projection lies in an interval representing the bit to be encoded. This can be visualized in Figure 2.7, where the point P_k displaces to a position P_k' such that the projection P_r shifts to a position P_r' and the bit encoded changes from 1 to 0. It can be observed that since the encoding point is displaced parallel to the line, the criteria for ordering encoding point(s) i.e. perpendicular distance stays invariant.

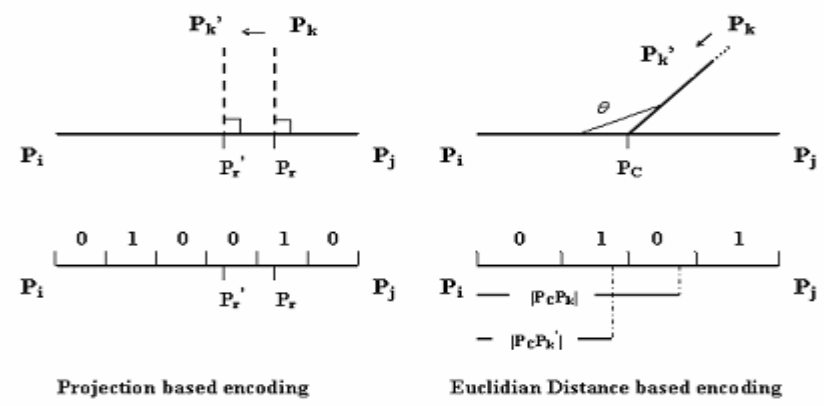


Figure 2.7. Bit Encoding schemes

Euclidian distance based strategy: The scalar quantity used to encode the bit information is the Euclidian distance (see Figure 2.7) between encoding point P_k and the centroid P_C of the invariant point set P_i and P_j. In order to encode, we visualize the Euclidian distance $|P_iP_j|$ as a scale divided into equal intervals and labeled as 0s and 1s. We measure the scalar quantity $|P_kP_C|$ against this scale and locate the interval and the bit it represents. In order to change the bit information (from 0 to 1) encoded by $|P_kP_C|$, we displace the position of point P_k on the

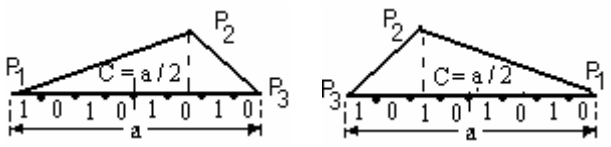
line joining P_C and P_k. It can be observed that since the encoding point is displaced along the line joining P_C and P_k, the criteria for ordering encoding point(s) (maximum angle) stays invariant.

Handling Pathological Cases: In some cases, there is a contention between points for invariant set. For these cases, we perturb some points to make sure only one pair is chosen. For example, the heuristic identifies the invariant set on the basis of maximum distant pair of points. In such a case, we can visualize contention in a cluster with size ~ 3, such that the points form an equilateral triangle or an isosceles triangle. There can be cases where points in the data set have same values i.e. points P_i and P_j belong to the same cluster and have equal values. For such cases, we have to choose among these points for encoding, which may result in contention. This situation can be avoided either by excluding these points from the encoding process or by perturbing them to achieve an encoding.

The projection based strategy has been used in *3-D model* watermarking in [2.16] and termed as a macro-embedding procedure MEP, whereas the Euclidian distance based strategy is our proposed method. In both cases, the scalar quantities change and this information is carried by the encoding point P_k or P_k'. Invariant point set can be subjected to reorder attack, in case the points are swapped in time. The Euclidian distance based encoding is robust against reordering of invariant points, since we not use the order of points in time, but use their positional information only. In our work [2.3], we extend the idea suggested in [2.16] such that it can handle reordering attack on invariant point set, as explained bellow:

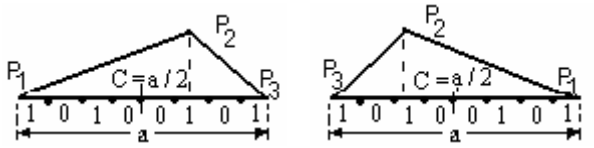
Extending MEP [2.3]: It can be observed that the invariant point set can be subjected to reorder attack, where positional information P_i and P_j are swapped in time. This may lead to loss of bit information stored using the ideas explained above. The idea related to projection

based encoding is made robust against such an attack by labeling the intervals as a palindrome of 1s and 0s. If bits corresponding to each interval are labeled from the center in opposite directions (see Figure 2.6) starting with 0 or 1, we get a palindrome of 0s and 1s. The advantage of a palindrome scheme [2.3] over the scheme suggested in [2.16] is that it is robust against reordering attack. For example, as shown in Figure 2.8, if the order of P_1 and P_3 is swapped, the encoded bit information flips from 0 to 1. However, as shown in Figure 2.9, the encoding does not change due to choice of the palindrome scheme as suggested in [2.3].



Before Attack After Attack

Figure 2.8. Macro Embedding Procedure [2.16]. Before reorder attack, for MEP bit = 0, and after Attack bit = 1



Before Attack After Attack

Figure 2.9. Extended Macro Embedding Procedure [2.3], no change in bit information

Comments on choice of bit encoding: The goal is to encode the maximum number of bits using n points in a cluster with minimal distortions such that the bits encoded are robust against uniform affine transforms and noise addition. Schemes such as least significant bit encoding, dither modulation or even quantization modulation that operate on absolute values for bit encoding are not robust against uniform affine transform. So, our choice is a bit encoding scheme where bits are preserved during uniform affine transform. Bit encoding in

schemes for 3-D meshes such as [2.2, 2.12, 2.20, 2.16, 2.43, and 2.49] are invariant to uniform affine transforms. So, we could apply these approaches per cluster by first finding a mesh for each cluster. However, the direct application of these approaches does not maximize the amount of bits based on the number of encoding points. To solve the problem, we consider two invariant points, and use the other points for encoding purpose. For every bit that is encoded, we use an extension of the projection based bit encoding from [2.16] technique and our proposed Euclidian strategy.

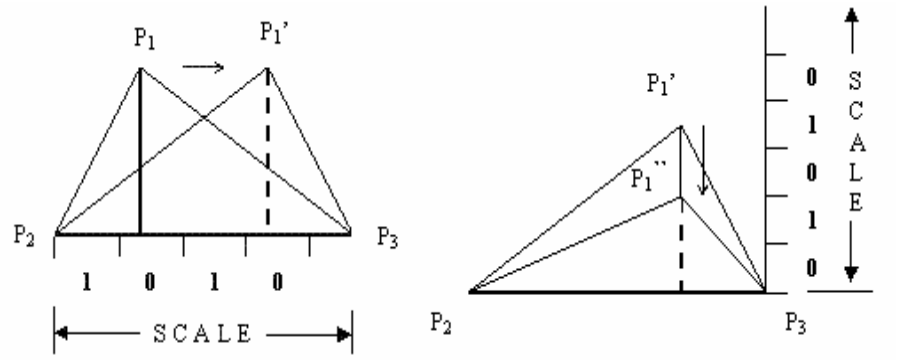


Figure 2.10. Representation of bit encoding [2.43] using independent scalar quantities in 3-D for points {P_1, P_2 and P_3}

Another method suggested in [2.43] is a variation of [2.16] approach and suggests encoding three bits per point by using two points as invariant. The basic idea is to encode bits in independent scalar quantities i.e. their modulation does not impact each other. For example, as seen in Figure 2.10, the projection of the point P_1 on the line $|P_2P_3|$ does not impact the modulation of the perpendicular projection on line. Since the points have to be ordered and then encoded, all the scalar quantities cannot be used for bit encoding. So, we have to fix one scalar quantity for ordering and then use the rest for encoding. Two scalar will be vulnerable to bit lost, since per point we are encoding two bits. So, our choice will be to use one scalar quantity for encoding purpose. Also, the invariant points are vulnerable to reordering attacks

since we still might be using projection based method to encode. So, the bit encoding suggested in [2.43] does not solve our problem, but the idea of independent scalar quantities helps us design our bit encoding scheme. Therefore, our extensions to 3-D QIM are more justified.

2.2.1.2 Choice of Scale – Benchmarking: The scale used for encoding purpose is defined by the number of intervals; we divide the Euclidian distance between the invariant points set. The definition of the scale used for encoding is restricted to the number of intervals, since the scale length varies with the Euclidian distance between invariant points. The idea of using quantization index modulation [2.15] helps in controlling distortions. This can be understood from the Figure 2.4, where we divide the Euclidian distance into equal intervals. The displacements for the point are the interval length that results in change in the value of positional information of encoding points. Smaller displacements lead to smaller distortions i.e. add less noise to the data set, and thus do not impact the semantics of the motion data. The drawback of using very small intervals is that they are vulnerable to noise addition. Noise additions can lead to change in the positional information of encoding points, which in turn changes the bit encoded since either the projection changes or the Euclidian distance changes. Also, as we increase the size of the interval length, we add robustness against noise addition attacks. However, as observed, larger intervals can add more distortion during watermark addition. So, as a comprise between robustness against noise addition and distortion, we have to *benchmark* our system by finding an approximate threshold on scale '*number of intervals*' used for encoding such that the semantics of the data stay unaltered and the bit representing the watermark stays unchanged during certain levels of noise addition attacks. Therefore, the choice of a benchmarked QIM scale can help us design a system

which is robust against meaning preserving noise addition and also conserves the imperceptibility criteria of the watermarking system.

2.2.1.3 Security Enhancements: The clusters can be chosen in a random fashion, where randomness is a secret based on a key used for watermarking. As a consequence, the clusters are separated by a random number of points or samples, termed as *embedding distance*. The use of random embedding distance reduces the chance of attacks, since the adversary finds it difficult to search for watermarks. In addition, we make the scale and encoding points per cluster, a secret as well. The scale is fixed as suggested in Sub-section 2.1.2, but the encoding points can be chosen randomly. Randomness is maintained using random numbers generated using a random number generator using a seed. The secrecy is maintained by keeping the seed secret. So, our key is a combination of the scale and seeds to random number generators that define patterns for clustering.

2.2.2 Process Summary and Watermark Extraction

The complete process of watermark embedding and extraction is summarized in Figure 2.11. The same key and watermark are used for embedding and extraction of watermark. The scale used for bit encoding and decoding is the same. The watermarked motion data is vulnerable to an attack which is represented by the unreliable channel. It is to be noted that during watermark embedding and extraction, we use a key to find patterns of clusters and embedding distances between them. This key is secret for security reasons. Since our scheme is blind in nature, in order to detect a watermark, we need to identify sequence of clusters in the watermarked motion data, and try to decode the bits related to the watermark. From each cluster, we decode the bit information and compare it with the original watermark

bit by bit, as shown in equation (2.1). For our scheme design, threshold is equal to '1'.

Detection Ratio (DR) ≥ Threshold, where DR = $\frac{\sum_{i=1}^{wsize}(w(i) * w'(i))}{wsize}$, $wsize$ ~ watermark size in bits, $w(i)$ ~ i^{th} bit of original watermark w, $w'(i)$ ~ i^{th} bit of detected watermark w' (2.1)

The extraction process is very simple in case there no cropping attacks. In the event of cropping attacks, we need to search for the presence of clusters in remaining data set.

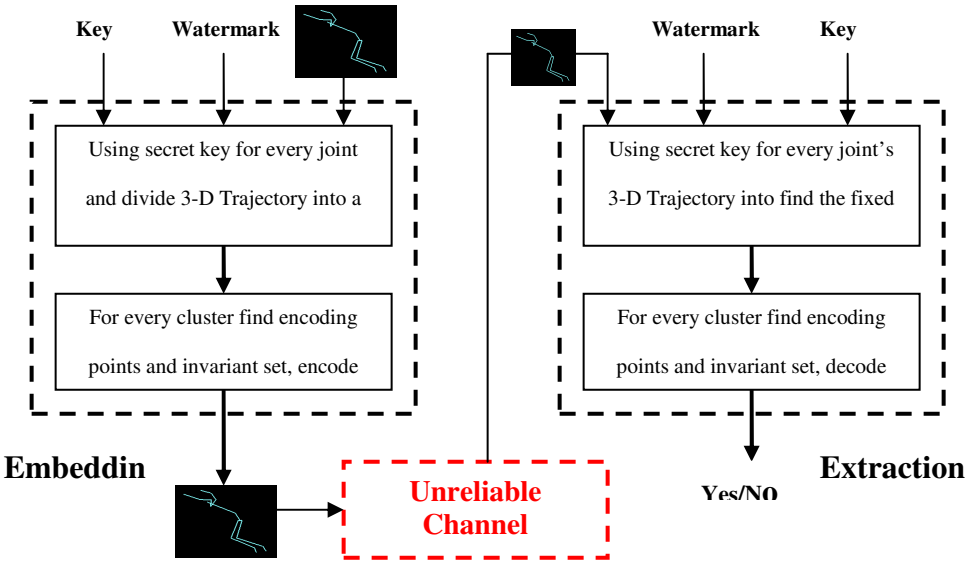


Figure 2.11. Watermark Embedding and Extraction Process

2.3 Scheme Analysis

The following subsection analyzes the watermarking scheme in terms of the properties of watermarking system.

2.3.1 Security Analysis

Security of the scheme relies on the ability of an attacker to trace the presence of watermark and remove the watermark. In the worst case, the adversary tries to attack all the points in the

data set. However, as designers of the algorithm, we aim to make it difficult for an attacker to search for watermarks. To do so, the idea of embedding distance has been introduced in Sub-section 2.1.3, where the clusters that carry the bit information are separated by random number of points. The randomness is determined on the basis of a secret which is used during encoding and decoding process. Due to this randomness, we can see that for m number of positional information varying in time, the search space for the adversary is 2^m. To enhance the scheme, we can exploit 3-D QIM where the scale used for encoding is also a secret. In such a case, if k defines the possible number of scale, the search space for scale is 2^k. In addition, per cluster we choose randomly to encode a bit in the ordered set of encoding points. So, given ne encoding points, randomness results in a 2^{ne} search space. Therefore, the total search space for the adversary will be $2^{m+k+\mathrm{ne}}$. As observed, the idea of randomization helps in inducing security as it increases the search for the adversary. Randomization can be different depending on the methodology adopted to watermark a data set. In our case, we introduce randomized embedding distances to induce security. For a technique such as [2.35], it could be random selection of patches for bit encoding. Thus, embedding distances is a choice, we make to induce security and it depends on our method. A common attack on a watermarking system is addition of another watermark which overwrites a previous watermark. Such attacks lead to ambiguity in the claim of the watermark. To overcome this problem, a common 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.

2.3.2 Imperceptibility Analysis

The imperceptibility criteria of the watermarking scheme depends on the distortions added during encoding phase. Distortions added also depend on number of points we use to encode and the scheme used for bit encoding. As observed in the analysis of hiding capacity (see Sub-section 3.3), larger cluster sizes lead to higher hiding capacity which intends to increase distortion. Also, the use of embedding distance reduces the number of clusters which in turn reduces the hiding capacity, resulting in lesser distortions. Distortion can also be controlled by the choice of scale used for bit encoding. As observed in Sub-section 2.1, the use of 3-D QIM helps us in designing schemes that add watermarks without altering the semantics of the data set. This is due to the choice of the interval length per scale defined using the number of intervals.

2.3.3 Time Complexity Analysis

The running time complexity for the encoding process mainly depends on the encoding process in the cluster. This is due to the fact that clusters occur in a fixed pattern; this will be O(1). To encode, we have to first an order among points based on scalar quantities depending on 3-D information. Ordering requires sorting which occurs in O(n log n) using an efficient algorithm such as heap sort. The bit encoding process takes O(n). Therefore, the sorting process dominates the procedure and the running time complexity is O(n log n). Since watermark extraction requires a similar process, it will take O(n log n). However, during cropping, we will have to find clusters using a brute force method with O(n) trials. Therefore, for extraction during cropping, our method takes O(n^2 log n).

2.3.4 Hiding Capacity Analysis

The hiding capacity is estimated as a rate, which is the ratio of the total number of bits encoded to the total number of points in the data set. For our scheme, we assume that one bit is embedded per encoding point. For the data set *D*, we can derive the embedding rate ER(*D*) using equation (2.2), which is the ratio of the total number of bits encoded in all the clusters to the total number of points *m*.

$$\mathrm{ER}(D) = \frac{\sum_{i=1}^{n} Bits_i}{m}, \; n \sim \text{total number of clusters}, \; Bits_i \sim \text{bits encoded in } i^{\text{th}} \text{ cluster} \qquad (2.2)$$

The constant factors in our case are the number of points *m,* dimensionality '*3*', and the embedding distance range [ED_{min} ED_{max}]. In order to minimize or maximize embedding rate, we need to find the cluster size that minimizes or maximizes ER(*D*) for any given set of embedding distances. To do so, we use a model where the embedding distance is fixed. The total number of clusters *n* of fixed size (*CSize ~ total number point per cluster*) will be separated by *n-1* embedding distances of fixed size *ED*. We calculate the relation between the total number of samples *m*, cluster size *CSize*, and embedding distance in equation (2.3).

$$m = n * CSize + (n - 1) * ED \qquad (2.3)$$

$$n = \frac{(m + ED)}{CSize + ED} \qquad (2.4)$$

From equation (2.3), we can derive the number of clusters as a function of *m*, *ED* and *CSize*, as shown in equation (2.4). Since *n* known, we can estimate ER(D) (see equation 2.5) using number bits encoded per cluster Bits(*CSize*), number of points *m*. Since the number of invariant points per cluster is '2', we can say that Bits(CSize) is equal to '*CSize* -2'. Therefore, we can substitute *n* and Bits(*CSize*) in equation (2.5) to derive equation (2.6) that

gives us the embedding rate in terms of cluster size, number of points *m* and embedding distance *ED*.

$$\text{ER}(D) = \frac{Bits(CSize) * n}{m} \tag{2.5}$$

$$\text{ER}(D) = \frac{CSize - 2}{m} * \frac{(m + ED)}{CSize + ED} \tag{2.6}$$

Equation (2.6) can be expressed as equation (2.7), since *ED* and *m* are fixed. This is minimized, if *CSize* is equal to 3, and maximized if CSize is equal to *m*.

$$\text{ER}(D) = \frac{CSize - 2}{CSize} = 1 - \frac{2}{CSize} \tag{2.7}$$

Using equation (2.7), we get upper bound on embedding rate $\text{ER}_{max}(D)$ (see equation 2.8), where the embedding distance is not present, since we use up all the data points in the cluster. However, for minimizing the embedding rate, we can assume a fixed embedding distance, and equation (2.6) gives us the lower bound, expressed as equation (2.9).

$$\text{ER}_{max}(D) = \frac{CSize - 2}{CSize} = 1 - \frac{2}{m} \tag{2.8}$$

$$\text{ER}_{min}(D) = \frac{1}{m} \cdot \frac{(m + ED)}{3 + ED} \tag{2.9}$$

From the above discussion, we can conclude that for a fixed embedding distance, the embedding rate increases with cluster size. The upper bound on embedding rate depends on the largest cluster size *m*, and the lower bound depends on the smallest cluster size '3'.

2.3.5 Robustness Analysis

In this dissertation, we consider two kinds of attack – semantics preserving and semantics altering. Semantics preserving attacks relate to the idea of removing the watermark while preserving the semantics of the motion data. Semantics altering attacks change the semantics

of the data set to derive another motion with a different semantics. Generally for copyright mechanisms, we assume that the technique should be at least robust against semantics preserving attacks since the copyright is related to the semantics of the data set. The following discussion classifies the attacks and discusses the robustness of our scheme that can be dependent on watermark size, watermark replications; number of points used for encoding and cluster size.

Uniform affine transformation: As discussed in Section 2.1, we can model this attack as format conversion where we do a change in co-ordinate system (local to global format and vice-versa). Such format conversion is semantics preserving in nature. However, if we apply different affine transform for each joint uniformly then the joints have a non-uniform affine transformation, which does alter the semantics of the data set. Format conversion (uniform affine transform) is applied to all the data points, implying that they impact all the encoded watermarks and replication of watermarks does not help to build robustness against the same. So, watermark length (size) is not a criterion to define robustness against such an attack.

To build robustness against such attacks, we make sure that robustness is provided for every bit that is encoded. So, the bits are robust against uniform affine transformation attacks due to the encoding strategy in the cluster based encoding process. The choice of the invariant set, order of encoding points and the bits to be encoded is based on proportionality of scalar quantities measured using a set of points. Since the proportionality '*comparison*' of these scalar quantities is unaltered during a uniform affine transformation, the bits encoded stay unchanged. As a result, during the watermark extraction process, bits related to the watermark can be decoded. Therefore, our methodology is robust against uniform affine transforms that can impact all the points for every joint's trajectory.

Cropping Attack: In this attack, we can crop the motion data matrix vertically or horizontally or by a combination of both. Vertical cropping preserves the semantics of the joint data, since we remove the complete information (trajectory in 3-D space) of a subset of joints. Since the watermark is being encoded per joint information, the scheme is not vulnerable to cropping attack wherein the entire joint information is cropped and use illegally. However, during horizontal cropping we remove points ordered in time, and this could alter the semantics, since we alter the trajectory of the joint in 3-D space. Cropping attacks are localized in nature i.e. they impact only the cropped set of data points. So, a scheme that tries to maximize the hiding capacity by replicating the watermark will have better robustness against the same. For a fixed hiding capacity, larger sized watermarks are more vulnerable to be impacted by cropping since they occupy more number of clusters as compare to a smaller watermark. In the following discussion, we analyze the robustness on the basis of cropping size.

In order to make sure that we are able to detect the watermark in the cropped part, the size C of the cropped region should be large enough and must have watermarks encoded in the same. This can be determined from C_{min} which is the number of points used to hide the watermark, and this can be the minimal cluster size ($CSize_{\min} \sim C_{min}$) that can hide all the watermark bits. In case the watermark has been replicated r times, we can say that the number of cropping instances for minimum size cropping is r. Of course, we can take larger values of C, but we are interested in estimating robustness for minimal size cropping. For a joint data with m points, the number of possibilities for cropping data with minimum cropping size C_{min} is '$m - C_{min} + 1$'. From these observations, we can find the probability of detecting the watermark, given C_{min}, m and r replications (see equation 2.10).

$$\text{Prob}_{\text{detection}} = \frac{r}{m - C_{\min} + 1} \tag{2.10}$$

We can improve $\text{Prob}_{\text{detection}}$, by replicating more (increase r) by increasing the hiding capacity. As observed in Sub-section 3.3, hiding capacity increases with cluster size, but since we restrict cluster size equal to C_{min}, the hiding capacity is limited in our case. In cases, where the cropped points are less than C_{min}, the $Prob_{detection}$ is 0. To maximize this probability, we need to choose a minimal cropping size $2*C_{min.}$

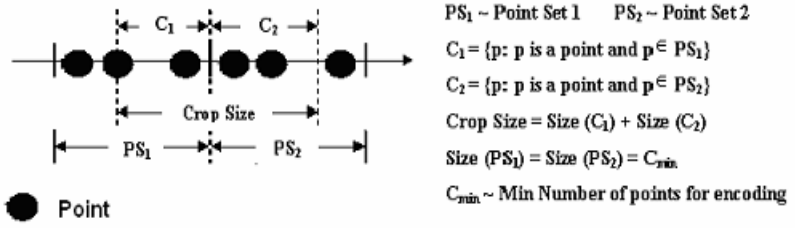


Figure 2.12. Cropping Scenario for a set of motion data points

This can be determined using Figure 2.12, where it is shown that data cropped consists of two set of points C_1 and C_2. These two sets belong to two different point sets PS_1 and PS_2, and it is assumed that they are of size C_{min} i.e. each comprises of a watermark. In case, we crop data and get C_1 or C_2 or both and whose size is not equal to their respective point sets PS_1 and PS_2, then we will not be able to detect the watermark. In order to do so, we need to make sure that either C_1 or C_2 is equal to C_{min}. This proves the fact the minimum cropping size required to maximize the probability of detection is $2*C_{min}$.

Re-ordering attack: This attack is done either by completely inverting the human action in time, or by randomly picking two neighboring points and swapping them. In the first case, we reverse the actions and change the semantics. However, few cases of random picking and swapping might not alter the semantics to such a great extent. So, reordering can impact all the points, in case inversion of motion information, or uniformly spread out among a set of

points. Since these attacks can impact all the points, watermarks size can play a role in defining robustness against such an attack, since larger sized watermarks are more spread out among a larger set of clusters.

It can be observed that robustness against order inversion can be trivially be dealt with by inverting the order and detecting the watermark. However, the case of random selection and reordering has to be dealt. Reordering attacks are not a problem when it occurs within the same cluster. This comes from the fact our scheme uses ordering within a cluster that is not dependent on time, but on the orientation of points in 3-D space. Reordering comes into picture when it impacts two different clusters. In this case, points from different clusters get reordered. It can be observed from a simple probabilistic equation that the choice of larger clusters can increase the robustness against such attacks. In equation (2.11), we see the probability of choosing two points from the same cluster, where the clusters of size *CSize* are made up from *m* points of joint motion information. This probability is further being simplified by expressing number of clusters (*NCluster*) as $\frac{m}{CSize}$ (see equation 2.12). Equation (2.13) gives us the formula for calculating the probability in terms of *m* and *CSize*. This implies that for larger cluster size the probability of failure for inter-cluster reordering attack is low. The maximum value for this probability is obtained for *CSize* ~ *m*, where we can assume 100% robustness.

$$\text{Prob}_{\text{Choice}} = \frac{\binom{CSize}{2}}{\binom{m}{2}} * \left(\frac{1}{NCluster}\right), \; NCluster \sim \text{Number of Clusters} \tag{2.11}$$

$$\text{Prob}_{\text{Choice)}} = \left(\frac{CSize * CSize - 1)}{m * m - 1} * \frac{m}{CSize}\right. \tag{2.12}$$

$$\text{Prob}_{\text{Choice}} = \frac{CSize - 1}{m - 1} \tag{2.13}$$

Noise Addition: In this attack, we add noise that can be due to smoothing resulting from linear interpolation (see equation 2.14 & 2.15) or possibly by adding another watermark. These can impact any subset of points, and impact the bits encoded per clusters. So, robustness against such an attack should depend on the nature of bit encoding scheme to preserve a bit of a watermark.

$$\text{Smooth1 (q (i))} = \frac{q(i-1) + q(i) + q(i+1)}{3}, \; q(i) \sim i^{th} \text{ instance of scalar q} \tag{2.14}$$

$$\text{Smooth2 (q (i))} = \frac{q(i-1) + q(i+1)}{2}, \; q(i) \sim i^{th} \text{ instance of scalar q} \tag{2.15}$$

Bit encoding is done using 3-D quantization index modulation. For QIM, we have a scale divided into equal length intervals. This division is based on number of intervals which defines the interval length.

As pointed out in Sub-section 2.1.2, the choice of the scale is based on the number of intervals since the interval length can vary. Also, the interval length determines the robustness against noise addition operations and the distortion added due to encoding. By robustness, we imply that addition of noise does not alter the scalar quantity to move out of the interval that signifies the bit encoded. For example, in the projection scheme (see Figure 2.4), the noise should not push the projection into another interval which toggles the bit encoded. Similarly for the Euclidian distance based approach, the addition of noise does not change the Euclidian distance so that we end up losing the bit information. The use of QIM helps in designing schemes by choosing a scale, such that the interval length restricts loss of bit information due to noise addition. The scale we use for the bit encoding scheme is robust against only a certain level of noise addition and can be lost as the intensity of noise is

increased. Therefore, in such cases with added distortion due increase noise we can observe an increase in bits loss.

From the above discussion, it can be concluded that the scheme builds robustness on a bit level in case of uniform affine transforms, vertical cropping, lower intensity noise additions and intra-cluster reordering attacks. Such kind of robustness helps achieve robustness for watermarks of any length. Robustness against inter-cluster reordering can be improved by choosing to encode with larger cluster size, and it also helps in improving watermark replication due to a better hiding capacity. However, during an increase in noise addition attacks, larger sized watermark will be more vulnerable to loss. Also, during time based cropping, the length of the watermark determines the maximum cropping size as it depends on twice the number of points that contain the watermark. These results will be verified in the following Sub-section 2.4.3.4.

2.4 Experiment Analysis and Discussion

The evaluation of this algorithm was done in MATLAB for watermark imperceptibility, choice of watermarking parameters, robustness against attacks and data hiding capacity. The experiments for data hiding capacity are generic. However, for robustness and imperceptibility, we use 100 different motion data files with varying number of frames (300 frames/second to 4286 frames per second captured at 120 frames/sec for 19 joints) collected from the motion capture laboratory at University of Texas at Dallas. Some of the snap-shots of the datasets used are given in Figure 2.13. For all the experiments, we embed a watermark of size 30 bits and watermark the trajectory of the 19 joints of the human motion the different files. The size of the watermark is changed in case of robustness analysis where watermark

size is a factor is the analysis. The following Sub-sections give details of the metrics used, attack models used for robustness, and performance analysis.

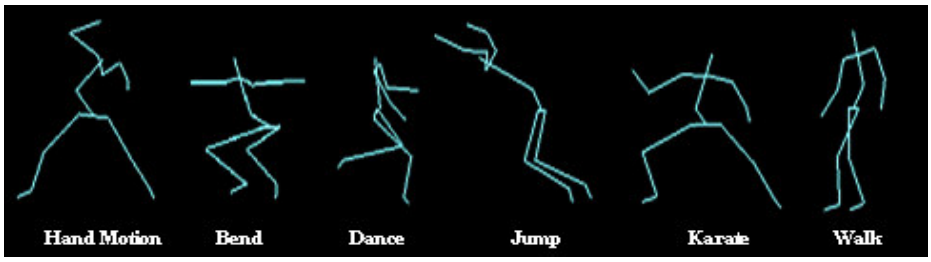


Figure 2.13. Snap-shots of some motions used for analyzing performance

We have used several metrics to evaluate the system. For imperceptibility and distortion due to attacks, we use the following:

Distortion Metric: This can measure the average displacement of joint position due to watermark or noise addition, which can be measured as Euclidian distance between joints positions P and P' (see equation 16, for original matrix M and modified matrix M').

$$\text{Distortion} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} Euclidian(P_{i,j}, P'_{i,j})}{n * m}$$

, where for i^{th} frame and j^{th} joint ($P_{i,j}$ is positional data for M) and ($P'_{i,j}$ is positional data for M'), m ~ number of frames, n ~ number of joints. (2.16)

This metric is useful in evaluating the distortion produced due to watermark addition, with lower values indicating more imperceptibility.

In order to evaluate for robustness we use the following metrics.

Bit Error Rate (BER): This is measured as the ratio of the number of bits flipped to the expected number of bits encoded in a given data set. Higher the rate, less robust is the scheme against a given attack.

Watermark Detection Rate (WDR): This is measured as the ratio of the number of watermarks detected to the total number of watermarks encoded. Higher values indicate better robustness.

2.4.1 Results and Discussion

The following Sub-sections discuss the experimental results.

2.4.1.1 Benchmarking Watermarking Scale (Distortion Vs Robustness): As explained in Sub-section 2.2.1.2, the distortion induced due to bit encoding is dependent on the interval size of the scale used during the bit encoding process. The size of intervals is dependent on the number of intervals by which we divide the scale. Therefore, to make the watermarks imperceptible, we choose a scale (number of intervals) that minimizes distortion. Figure 2.14 illustrates that as we increase the number of intervals, distortion is reduced for different motion files. So, the choice of an appropriate scale makes the watermark imperceptible. As a result, as shown in Figure 2.15 and Figure 2.16, the joint trajectory and action which had distortions due to perceptible watermarks will no longer have distortions and the watermark is imperceptible.

Also, the choice of the scale impacts robustness against noise addition. Figure 2.14 illustrates this fact, since scales with lesser number of intervals (larger interval size) are more robust (smaller bit error rates) as compared to scales with larger number of intervals (smaller interval size). Figure 2.14 shows us that larger interval sizes are robust but cause more distortion, for example number of intervals ‘50’ causes more distortion but has less bit error rates for varying noise attacks as compared to number of intervals ‘1000’.

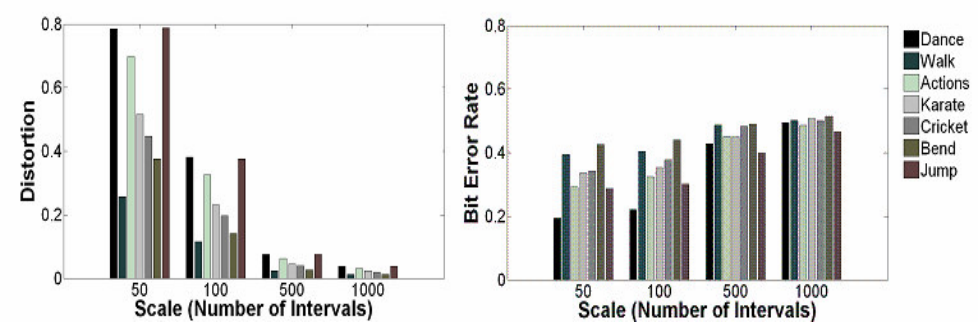


Figure 2.14. Robustness and Distortion for varying number of intervals for different motion files

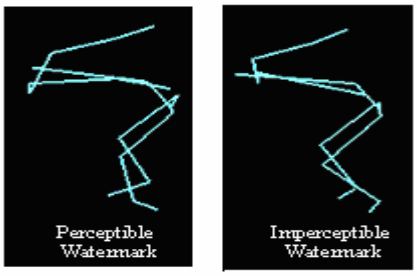


Figure 2.15. Benchmarking removes distortions in action

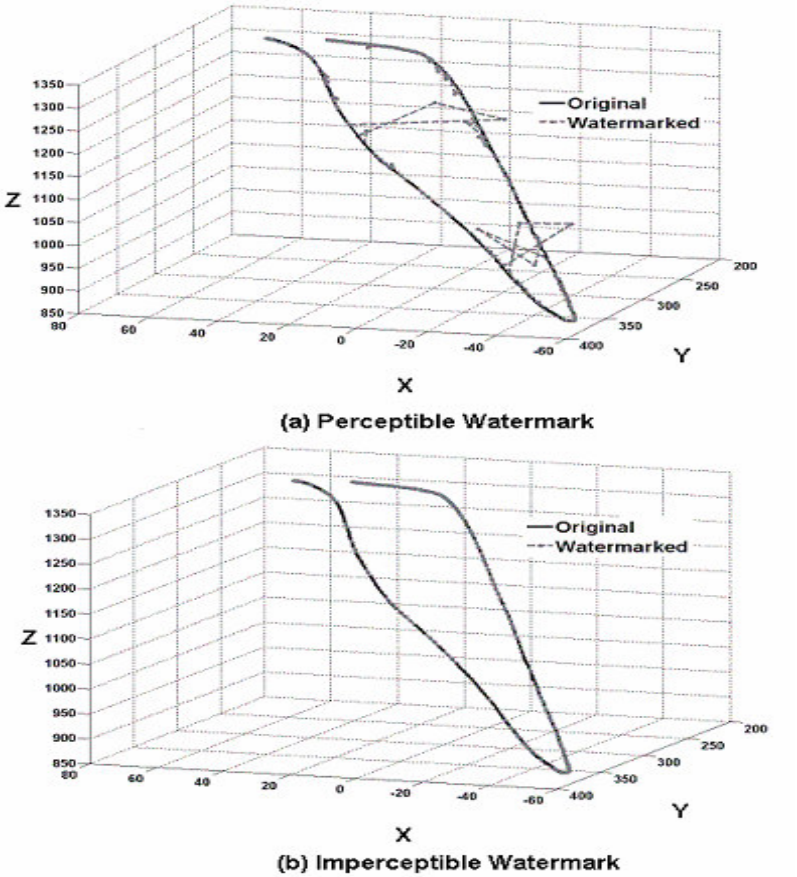


Figure 2.16. Benchmarking removes distortions in 3-D trajectory

Based on the above discussions, in order to choose a scale for encoding that tries to bring down the bit error rate and minimize distortion as well. So, we need to find a benchmark 'number of intervals' that will distort the data set by an amount that does not impact the semantics of the data set.

Different motion data such as shown in Figure 2.13 have different semantics. As a result, for similar scales being used the perceptible distortions in each motion will vary. Therefore, the scale (number of intervals) at which the watermarks are imperceptible is different. To prove this fact, we used a range of scales to add watermarks to the motion data files, and visually inspected the motion data. It was observed that different motion files result in different scales, and this can be observed in Table 2.1 that gives us approximate thresholds

Table 2.1. Benchmarking Scales for different Motion Data

Motion Type	*No. of frames*	*No. of intervals*
Walk	1315	80
Jump	331	60
Karate	262	50
Actions	1829	40
Dance	3378	40
Cricket	258	100
Bend	323	80

2.4.1.2 Comparison and Choice of Bit Encoding Scheme. The bit encoding scheme used for robustness analysis is Euclidian distance based encoding. This can be understood from Figure 2.17, which shows robustness analysis in terms of attacks on watermarks. The bit error rate observed for Euclidian distance based encoding is better than projection based encoding (extended MEP [2.3]). Although both the scheme are robust against reordering

attacks for same scale, watermark size encoded and attack, Euclidian distance based encoding performs better and hence is our choice for robustness analysis.

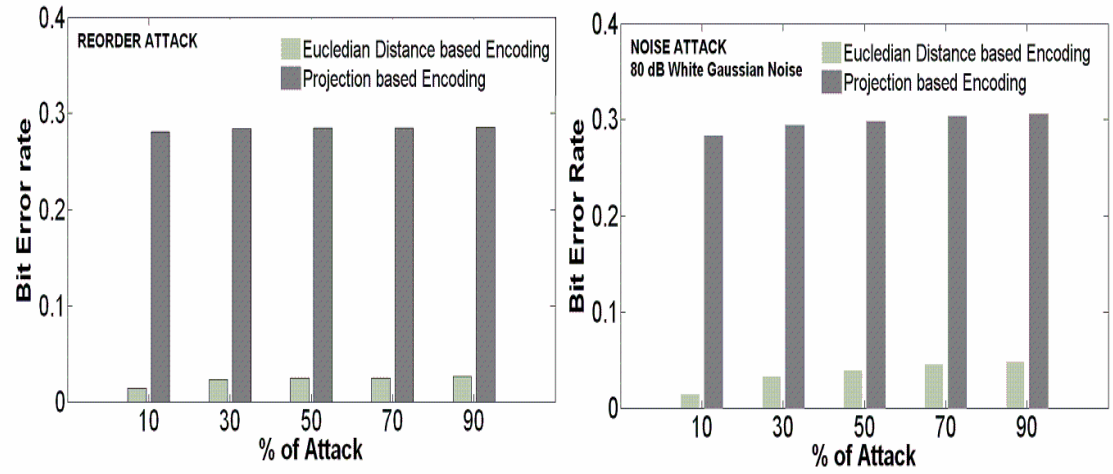


Figure 2.17. Robustness analysis for comparing bit encoding schemes (Euclidian distance based encoding Vs projection based encoding [2.3]), for same cluster size 50, watermark size 32 and scale (number of intervals ~ 32768.)

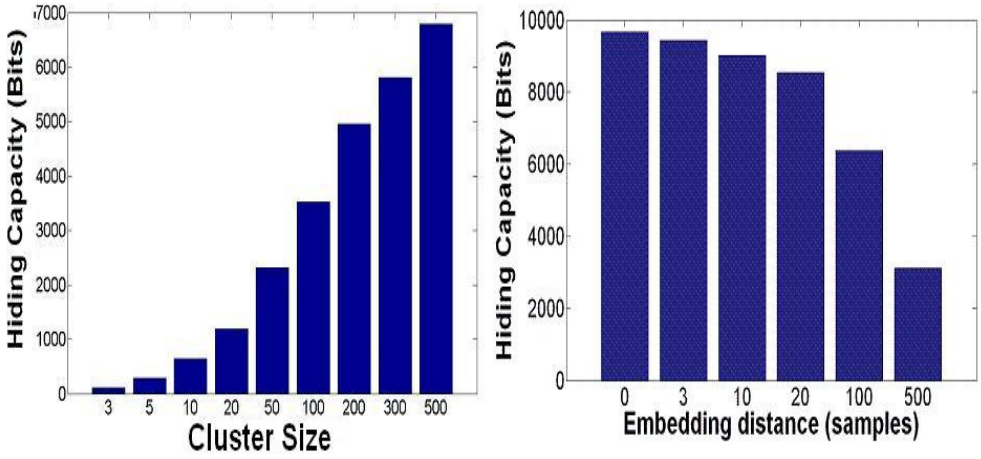


Figure 2.18. Hiding Capacity based on Cluster Size and Embedding distance

2.4.1.3 Hiding Capacity. The experiments were done for fixed 10000 samples, and hiding capacity measured as bits instead of embedding rate since the data set is fixed. In order to measure the impact of cluster size, we vary the embedding distance randomly for the range [0 500]. For embedding distance impact, we randomly vary the cluster size for the range [3 500]. Figure 2.18 verifies the results in Sub-section 2.3.4, since the hiding capacity is

increases with the cluster size and is minimum for cluster size ‘3’. On the contrary (see Figure 2.18), with increasing embedding distances the hiding capacity drops.

2.4.1.4 Robustness Analysis. Robustness is analyzed as follows:

Uniform Affine transforms: The technique shows 100% robustness against uniform affine transformation (translation, scaling and rotation). Since we embed watermarks on a per joint basis, the technique is also robust against an attack where uniform affine attacks vary across different joints. Therefore, the scheme is robust against (semantics preserving) format conversions and different (semantics altering) non-uniform attacks on joints. This can be observed in Figure 2.19, where the orientation of joint trajectory is changed by applying combinations of uniform affine transforms, and all watermarks are detected such that the bit error rate it 0.

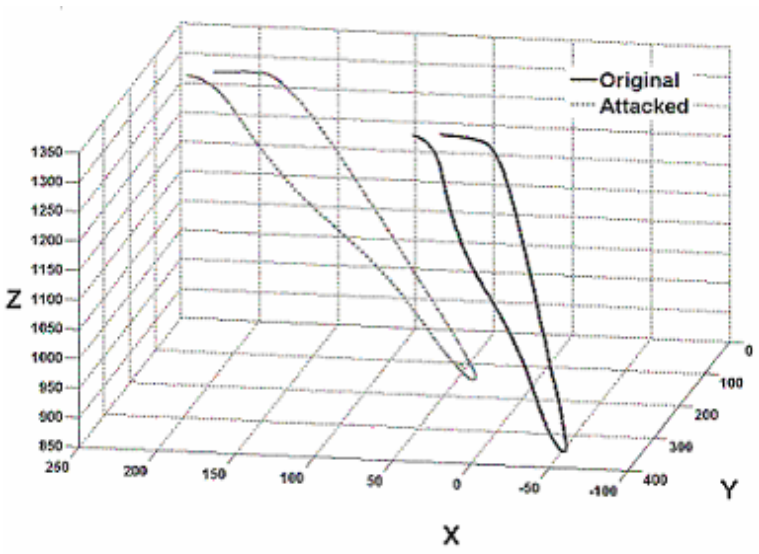


Figure 2.19. Snap-shots of uniform Affine transforms on 3-D trajectory of a joint

Cropping Attacks: Since we watermark each joint motion information (trajectory in 3-D space), our scheme is robust to semantics preserving vertical cropping. As suggested in the attack model, horizontal or time based cropping attacks are semantic altering. Robustness

against such an attack depends on the cluster size as discussed above. This is observed in Figure 2.20, where a 3-D trajectory is cropped and translated. Since clusters are still present in the cropped portions, we can detect the watermark.

Figure 2.21 gives an analysis for robustness against horizontal (time based) cropping, where watermark is assumed to be embedded in a cluster and the data is being attacked by varying the cropping size. Since all the bits can be encoded in a minimum sized cluster ('5' in this case), than for a given cropping size, the choice of clusters larger than this minimum cluster size will result in smaller watermark detection rates. Therefore, the minimum cluster size used to encode the watermark maximizes the robustness against different cropping sizes. Also, for cropping size greater than a cluster, we find the watermark detection rate is more than '0' and becomes maximized for values where cropping size is twice the size of the clusters. Therefore, the cropping is limited by twice the data set size (number of points i.e. minimum sized cluster in this case) required to encode the watermark, which verifies our relation as suggested in Sub-section 2.3.5

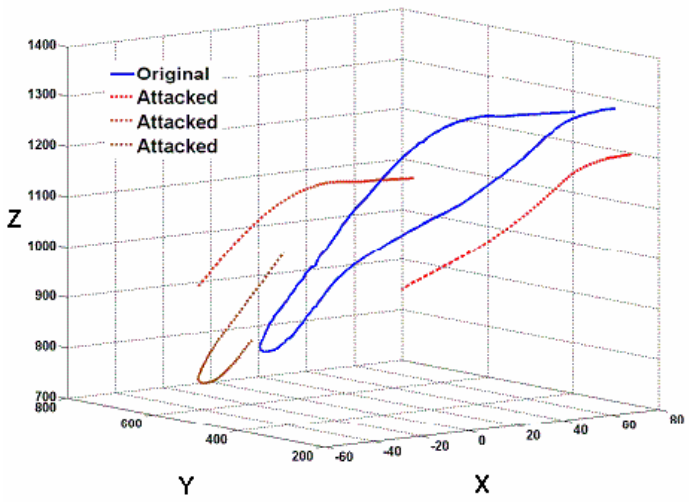


Figure 2.20. Cropping Attack Analysis (Snap-shot of trajectory)

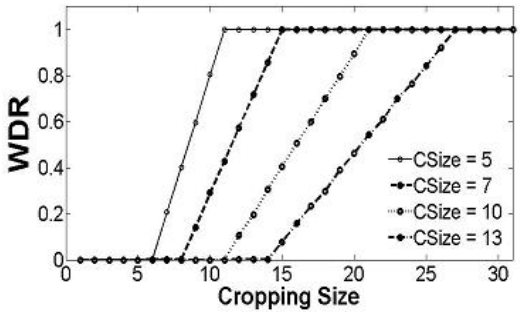


Figure 2.21. Cropping Attack Analysis – Watermark Detection Rate for varying cropping size

Noise addition attacks were done by adding second watermark; white Gaussian noise or smoothing attacks (see equation 2.14 & 2.15). As suggested in Sub-section 2.3.5, from Figure 2.22, we can observe that robustness against noise addition decreases with the increase in intensity (e.g. Gaussian noise 10 dB to 1dB) of noise addition as shown by relative comparison in distortions and bit error rates. This can also be observed for watermarking attack (see Table 2.2) where higher intensity noise attacks achieved using scales with lesser number of intervals results in high bit error rates.

Table 2.2. Bit Error Rate Vs Distortion for varying Scale (Number of intervals) 2nd Watermark Attack

Number of intervals	50	100	500	1000
Distortion	1.1843	0.5919	0.1191	0.0593
Bit Error Rate	0.4002	0.2482	0.0498	0.025

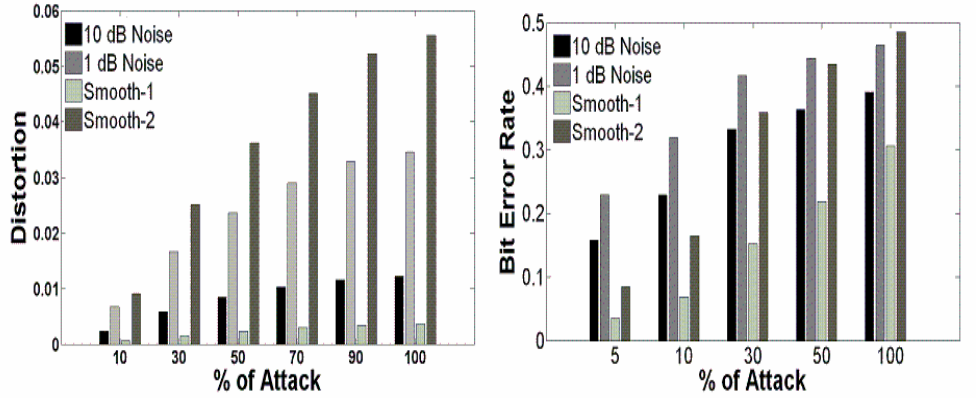


Figure 2.22. Distortion and Robustness analysis for noise addition and reordering attacks

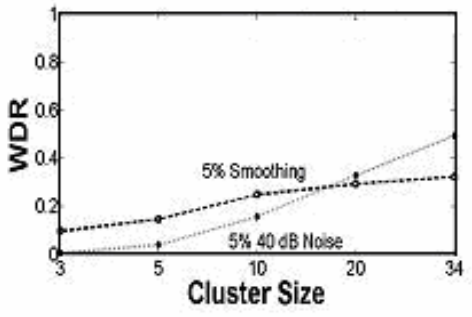


Figure 2. 23. Watermark Detection Rate (WDR) Vs Cluster Size
Robustness for cluster varying – Noise Addition, watermark size = 30

The distortion (see Table 2) caused during second watermark addition attack due to lesser number of intervals is more, since for the proposed watermarking scheme, noise added is dependent on the interval size which large in this case.

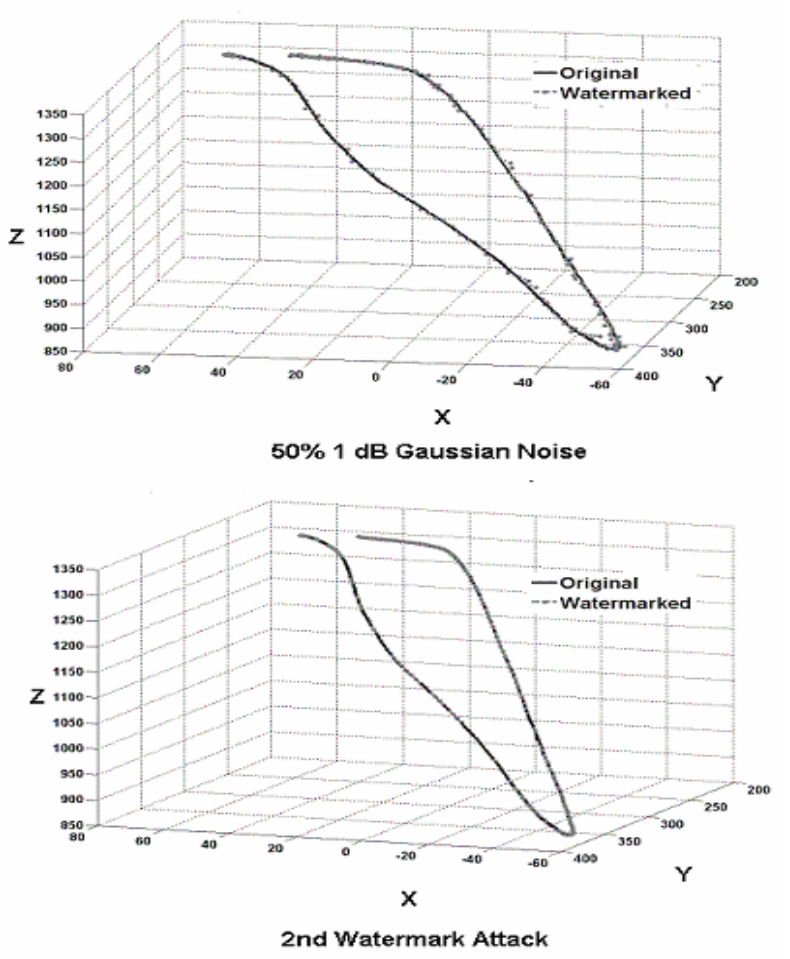


Figure 2.24. Snap-shots of variations of noise attacks on 3-D trajectory

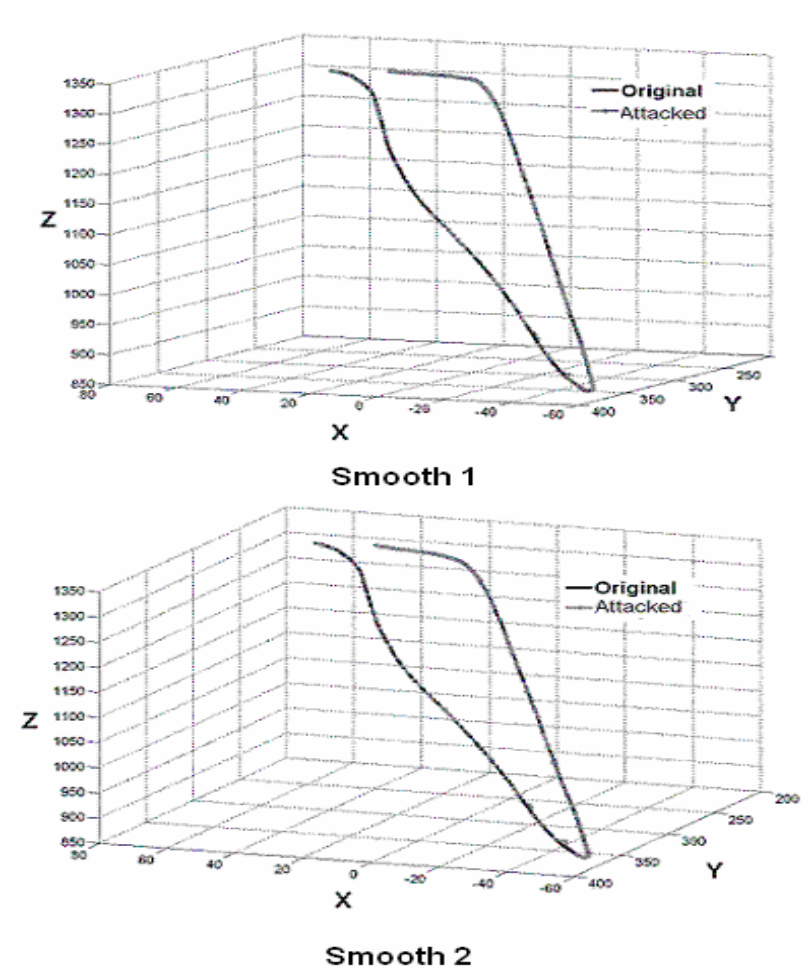


Figure 2.25. Snap-shots of variations of noise attacks on 3-D trajectory

Figure 2.24-2.25 gives snap-shots of noise attacks on 3-D joint trajectory, where watermarks were detected successfully. Robustness against noise addition attacks can be improved by choosing larger cluster sizes as they help increase the hiding capacity and we can replicate more. This is observed in Figure 2.23, where the watermark detection rate improves with increasing cluster size for fixed watermark of length '30'.

We tested the technique for *reordering attacks* by choosing random neighboring points in a trajectory and swapping them. An improvement in the robustness against reordering attacks is also observed as given in Figure 2.26, and this can be accounted due to the increase in hiding capacity and better robustness for larger cluster sizes against inter-cluster reordering

attack as suggested in Sub-section 2.3.5. Based on this idea, we can choose a cluster size that improves robustness against reordering attack and other attacks as well. As seen in Figure 2.27, the scheme is highly robust against reordering attacks since bit error rate is less than 0.03 even after 100% points are subjected to attack. A snap-shot of reordering attack can be observed in Figure 2.28.

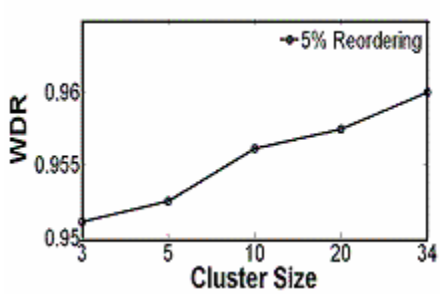


Figure 2.26. Watermark Detection Rate (WDR) Vs Cluster Size
Robustness for cluster varying – Reorder Attack, watermark size = 30

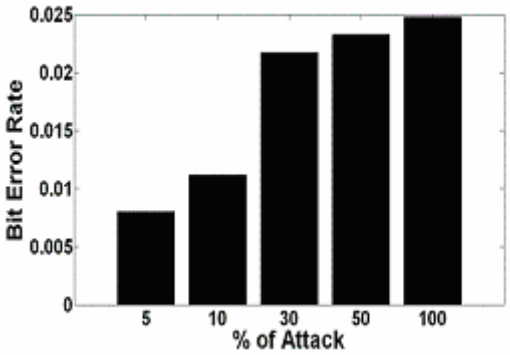


Figure 2.27. Reordering attack analysis - bit error rate

Interpretation from Robustness Analysis: Based on the above observations, we can conclude that robustness against uniform affine transforms, vertical cropping, and reordering is more as compared to noise additions and time based cropping attacks. The high robustness makes the watermark length less dependent of such attacks, as shown in Figure 2.29, where the watermark detection rate stays above 90% for varying watermark size. However, time based attacks are dependent on the cropping size (see Figure 2.21) and this should be twice the

number of points for encoding the watermark. Therefore, the length of the watermark we can encode will depend on the cropping size. Robustness against noise addition attacks can be modeled by benchmarking the scale, but larger intensity attacks that add more distortions will result in the loss of the watermarks. Larger cluster sizes can help improve robustness against attacks since we can replicate more data and it also builds robustness against reordering attacks. Therefore, larger watermark sizes are vulnerable to loss during higher intensity noise addition attack. The above result about watermark length is consistent with our discussion in Sub-section 2.3.5, and can also be seen observed in Figure 2.29.

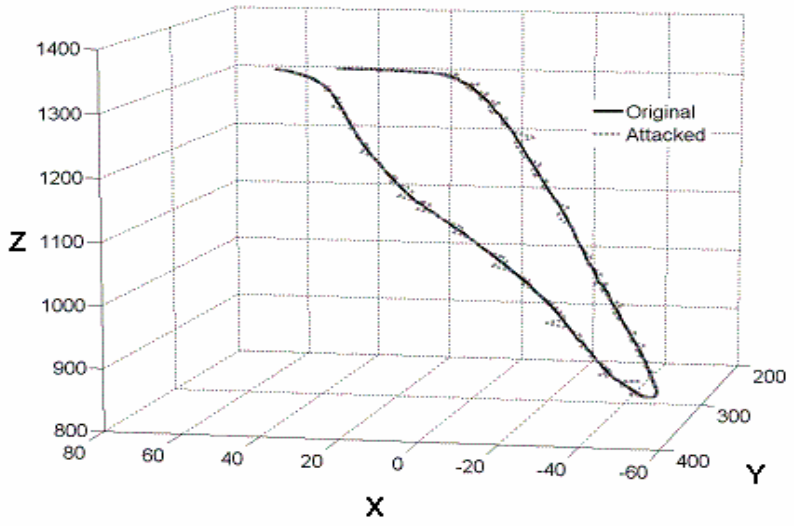


Figure 2.28. Reordering attack analysis – 3-D trajectory of joint

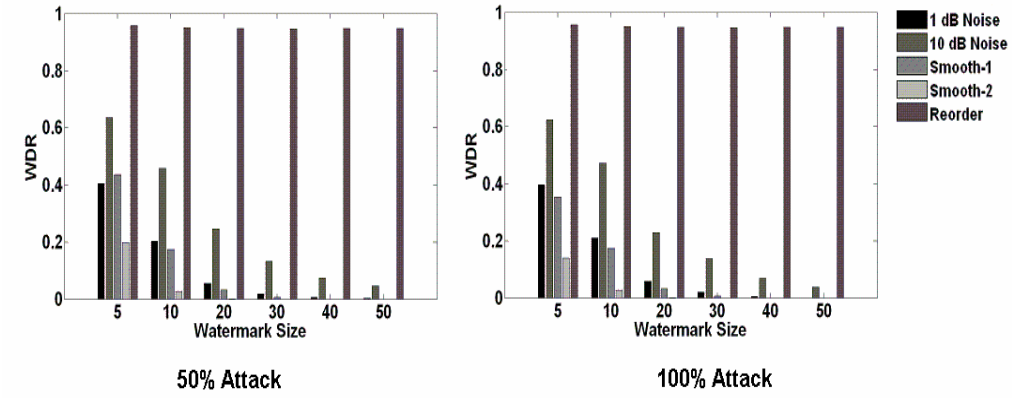


Figure 2.29. Robustness analysis for watermark size and percentage of different attacks

2.5 Summary

The dissertation presented a blind robust watermarking algorithm for copyright protection of 3-D motion capture data. The scheme divides the time varying positional information into clusters separated by secret embedding distances that enhance the system's security. Watermarks encoded in these clusters are shown to be robust against attacks such as uniform affine transformation, noise addition, reordering, and cropping. The bit encoding is done using proposed extensions to 3-D quantization index modulation, resulting in imperceptible watermarks. The upper and lower hiding capacity offered by the system is estimated in terms of the parameters of the system (cluster size and embedding distance), and it is shown that the hiding capacity increases with cluster size for fixed embedding distance. Experiments show us the credibility of the theoretical estimates, with imperceptibility being achieved with benchmarking of the watermarking scale. Analysis on robustness proves the technique to be highly robust against uniform affine transforms, vertical cropping, and reordering attacks as compared to noise addition and time based cropping attacks. The watermark length is dependent on the size of cropping. Although benchmarking improves robustness against noise addition attacks, the robustness decreases with the increase in intensity of noise added. Since the scheme is robust against uniform affine transforms, it makes it generic and applicable to different motion data formats. The time complexity for watermark embedding and extraction is estimated as $O(n \log n)$ and $O(n^2 \log n)$ respectively.

BIBLIOGRAPHY

[2.1] AUTODESK - Motion Builder - www.autodesk.com

[2.2] Alface, P., and Macq, B. 2005. Blind watermarking of 3-D Meshes Using Robust Feature Points Detection. In Proceedings of International Conference on Image Processing 2005, Volume 1, 11-14 Sept. 2005 693 – 696.

[2.3] Agarwal, P., Adi, K., and Prabhakaran, B. 2006. Robust blind watermarking mechanism for motion data streams. In *Proceedings of ACM Multimedia and Security Workshop*, Geneva, Switzerland, September 26-27, 2006.

[2.4] Bhattacharya, S.,Chattopadhyay, T., and Pal, A. 2006. A Survey on Different Video Watermarking Techniques and Comparative Analysis with Reference to H.264/AVC. Consumer Electronics, 2006. ISCE '06. 2006 IEEE Tenth International Symposium on 2006, 1 – 6.

[2.5] Bruderlin, A., and Williams, L. 1995. Motion signal processing. In Computer Graphics(Aug. 1995),. Proceedings of SIGGRAPH 95, 97—104

[2.6] Benedens, O. 2002. Robust watermarking and affine registration of 3-D meshes. 2002. In Proc. Information Hiding Noordwijkerhout, the Netherlands, 2002, 177-195.

[2.7] Byeong-seob, K., Nishimura, R., and Suzuki, Y. 2005. Time-spread echo method for digital audio watermarking. Multimedia, IEEE Transactions on, Volume: 7, Issue: 2 April 2005, 212- 221.

[2.8] Bors, A. Watermarking Mesh-Based Representations of 3-D Objects Using Local Moments. 2006. Image Processing, IEEE Transactions on, Volume: 15 Issue: 3 March 2006, 687- 701.

[2.9] Bodo, Y., Laurent, Y. N., and Dugelay, J-L. 2003. Watermarking video, hierarchical embedding in motion vectors. In Proceedings of Image Processing, 2003. ICIP 2003. Proc. International Conference on Volume 2, 14-17 Sept. 2003, 739-42 vol.3.

[2.10] Choi, K. J., and Ko, H.S. 2000. Online motion retargeting, Journal of Visualization and Computer Animation. Proceedings. Seventh Pacific Conference on Computer Graphics and Applications v. 11 n. 5 2000, 223-235.

[2.11] Cotting, D., Weyrich, T., Pauly, M., AND GROSS, M. 2004. Robust Watermarking of Point-Sampled Geometry. 2004. In Proceedings of International Conference on Shape Modeling and Applications 2004, 233-242.

[2.12] Cho, J.-W., Prost, R., and Jung, H.-Y. 2007. An Oblivious Watermarking for 3-D Polygonal Meshes Using Distribution of Vertex Norms, IEEE Trans. on Signal Processing, Vol 55, No. 1, 142-155.

[2.13] Cox, I., Miller, M., and Bloom, J. 2001. *Digital Watermarking: Principles & Practice* (The Morgan Kaufmann Series in Multimedia and Information Systems)

[2.14] Cox, I., Kilian, J., Leigton, F., and Shamoon, T. 1996. Secure Spread Spectrum Watermarking for Multimedia. In *IEEE Trans. Image Processing*, 1996, vol. 6, pp. 1673-1687.

[2.15] Chen, B., and Wornell, G. W. 2001. 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

[2.16] Cayre, F., and Macq, B. 2003. Data hiding on 3-D triangle meshes. In Signal *Processing, IEEE Transactions on*, Volume: 51 Issue: 4 Apr 2003 , Page(s): 939- 949

[2.17] Gomes, J., Velho, L., DE Silva, F. W., and Goldenstein, S. K. 2000. *Motion Processing using Variable Harmonic Components* Computer Animation 2000, 62-70.

[2.18] Golam, A., and Wong, K. C. 2000. Dynamic Time Warp Based Framespace Interpolation for Motion Editing. Graphics Interface. pp. 45-52.

[2.19] Jeong, I.-K., Park, K.J., Baek, S. M., and Lee, I. 2001. Implementation of a motion editing system, Proceedings of the seventh international conference on virtual systems and multimedia, 2001, 761-769.

[2.20] Harte, T., and Bors, A. 2002. Watermarking 3-D models. Image Processing. 2002. Proceedings. 2002 International Conference on, Volume 3, 24-28 June 2002 661 - 664.

[2.21] Huang, J., Wang, Y., and Shi, Y. 2002. A blind audio watermarking algorithm with self-synchronization. In Proceedings of IEEE International Conference on Circuits and Systems, 2002, vol. 3, 627-630.

[2.22] Kim, T., Lee, J., and Shin, S. 2000. Robust Motion Watermarking based on Multiresolution Analysis. In *Proc. EUROGRAPHICS*, Vol. 19 No. 3, pp. 189-198, 2000

[2.23] Kim, H. J. 2003.Audio Watermarking Techniques, In *Pacific Rim Workshop on Digital Steganography*. Kyushu Institute of Technology, Kitakyushu, Japan, July, 2003

[2.24] Lim, I.S., and Thalmann, D. 2002. Construction of Animation Models out of Captured Data. Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2002). August, 26-29.

[2.25] Lee, J., and Shin, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. Proceedings of SIGGRAPH 99, 39--48.

[2.26] Lee, S. H. 2004. 3-D mesh watermarking using projection onto convex sets. Image Processing, 2004. ICIP '04. 2004 International Conference on, 24-27 Oct. 2004 Volume: 3,On page(s): 1577- 1580 Vol. 3

[2.27] Lu, C. S., Marl Lio, H. Y., and Chen, L. H. 2000. Multipurpose Audio Watermarking. In Proceedings of 15th Int. Conf. on Pattern Recognition, Barcelona, Spain, Vol. III, pp. 286-289, 2000

[2.28] Li, W., Xu, X., and Lu, P. 2006. Localized audio watermarking technique robust against time-scale modification. Multimedia, IEEE Transactions on, Volume 8, Issue 1, Feb. 2006, 60 – 69.

[2.29] Lemma, A.N., Aprea, J., and Oomen, W.; van de Kerkhof, L. 2003. A temporal domain audio watermarking technique Signal Processing, IEEE Transactions on [2.see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] Volume 51, Issue 4, April 2003, 1088 - 1097

[2.30] Lu, C-S, Chen, J-R., Liao, H.-Y.M.; and Fan K-C. 2002. Real-time MPEG2 video watermarking in the VLC domain. In Pattern Recognition, 2002. Proceedings. 16th International Conference on Volume 2, 11-15 Aug. 2002 552 - 555 vol.2.

[2.31] Menache, A. 2000 .Understanding Motion Capture for Computer Animation and Video Games, Morgann Kaufman, ISBN - 0-12-4900630,

[2.32] Malik, H., Khokhar, S., and Rashid, A. Robust audio watermarking using frequency selective spread spectrum theory; Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on Volume 5, 17-21 May 2004, 385-8 vol.5

[2.33] Mansour, and Tewfik, M. A. 2001. Time-scale invariant audio data embedding. In proceedings of International Conference on Multimedia and Expo, 2001.

[2.34] Uccheddu, F., Corsini, M., and Barni, M. 2004. Wavelet-Based Blind Watermarking of 3-D Models. 2004. In Proceedings of the 6th ACM MM & Security workshop 2004, Pages: 143 – 154.

[2.35] Ohbuchi, R., Takahashi, S., and Miyazawa, T. 2001. Watermarking 3-D polygonal meshes in the mesh spectral domain. In *Proc. Graphics Interface*, ON, Canada, 2001, pp. 9-17

[2.36] Parent, R. 2001. Computer Animation - Algorithms and Techniques (The Morgan Kaufmann Series in Computer Graphics), 2001, ISBN - 1-55860-579-7.

[2.37] Pankajakshan, V., Doerr, G., and Bora, P. K. 2006. Assessing Motion Coherency in Video Watermarking. In Proceedings of the 8th ACM Multimedia and Security Workshop.

[2.38] Petrovic, R. 2001. Audio signal watermarking based on replica modulation. In Proceedings of International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Service, 2001, vol. 1, 227-234.

[2.39] Su, K., Kundur, D., and Hatzinakos, D. 2005. Statistical invisibility for collusion-resistant digital video watermarking. In Multimedia, IEEE Transactions on , vol.7, no.1, 43-51, Feb. 2005.

[2.40] Wu, C., Su, C., and Kuo, C. 2002. Robust and efficient digital audio watermarking using audio content analysis. In Proceedings of Security and Watermarking of Multimedia Contents, SPIE, 2002, vol. 3971, 382-392.

[2.41] Xiao, S., Lu, Z., Zou, F., and Ling, H. 2006. Video Watermarking Algorithm Based on Balanced Multiwavelets Signal Processing, The 8th International Conference on Volume 1, 16-20.

[2.42] Wang, X.-Y.,and ZHAO, H. 2006. A Novel Synchronization Invariant Audio Watermarking Scheme Based on DWT and DCT. Signal Processing, IEEE Transactions on, Volume 54, Issue 12, 4835 – 4840.

[2.43] Wang, C., AND Cheng, Y. 2005. An Efficient Information Hiding Algorithm for Polygon Models. In *EuroGraphics* 2005, Volume 24 Page 591 - September 2005

[2.44] Witkin, A., and POPOVIC, Z. 1995. Motion warping. In SIGGRAPH 95 Proceedings, Annual Conference Series, 105--108.
[2.45] VICON. www.vicon.com

[2.46] Yamazaki, S. 2004. Watermarking Motion Data, In *Proc. Pacific Rim Workshop on Digital Steganography* (STEG04), pp.177-185, Nov 2004

[2.47] Yeo, I., and Kim, B. 2003. Modified patch-work algorithm: A novel audio watermarking scheme. In proceedings of IEEE Transactions on Speech and Audio Processing, vol. 11, 2003.

[2.48] Yeo, I., Kim, B. 2003. Modified patch-work algorithm: A novel audio watermarking scheme. In *IEEE Transactions on Speech and Audio Processing*, vol. 11, 2003

[2.49] Zafeiriou, S., Tefas, A., and Pitas, I. 2005. Blind robust watermarking schemes for copyright protection of 3-D mesh objects. In *Visualization and Computer Graphics*, *IEEE Transactions on*, Volume 11, Issue 5, Sept.-Oct. 2005 Page(s):596 – 607

[2.50] Zhang, J., Li, J., and Zhang, L. 2001. Video Watermark Technique in Motion Vector. In Proceedings of sibgrapi, p. 179, XIV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'01), 2001.

[2.51] Zhang, J., Ho, A.T.S., Qiu, G., and Marziliano, P. 2007. Robust Video Watermarking of H.264/AVC. Circuits and Systems II: Express Briefs, IEEE Transactions on [2.see also Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on], Volume 54, Issue 2, Feb. 2007, 205 – 209

CHAPTER 3

ROBUST BLIND WATERMARKING OF POINT SAMPLED GEOMETRY

Parag Agarwal and Balakrishnan Prabhakaran

Computer Science Department, EC 31

The University of Texas at Dallas

800 WEST CAMPELL RD.

Richardson, Texas 75080-3021

3.1 Introduction

3-D objects have found applications in CAD/CAM, animations, movies, video games, and medical informatics (3-D Images). 3-D objects can be created from software such as Maya [3.2], or derived using 3-D data acquisition devices [3.14, 3.17]. 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 [3.13, 3.19]. Watermarking achieves protection by hiding information (watermark) inside digital medium (3-D object in our case). The presence of the watermark verifies the copyright.

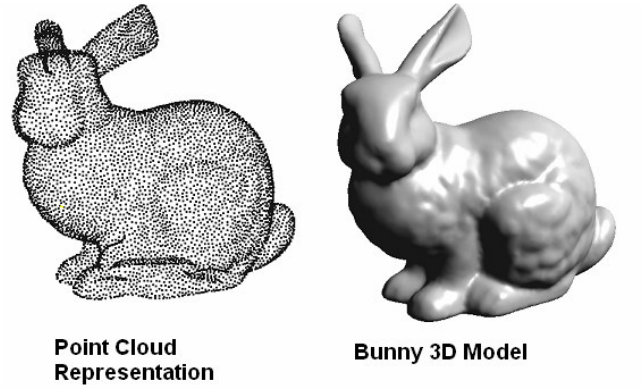


Figure 3.1. Point Representation of Stanford Bunny Model

Point sampled geometry consists of 3-D geometry (see Figure 3.1 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 3-D models, such as 3-D mesh and 3-D point sampled geometry. This comes from the fact that generic schemes operate only on geometric information of 3-D meshes and 3-D point sampled geometry. We also observe that in case of 3-D meshes, geometry based encoding is robust against connectivity altering attacks, since the encoding is independent of this information. Therefore, this dissertation focuses on generic schemes of watermarking of

point sample geometry representations of 3-D 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*: 3-D models can be represented in different file formats (e.g. polygon format - .ply, virtual reality format - .VRML) and the 3-D points are given identified as a (order) sequence. The sequence numbering (ordering) can be changed, but it will not alter the 3-D 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 3-D 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 3-D 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 3-D 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 3-D 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 3-D 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 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 3-D points. As mentioned above, we have to make the scheme independent of the order of 3-D points given in the file describing the 3-D models. For n given 3-D points, we can have $n!$ number of orders. Since 3-D 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.

3.1.1 Related Work and Issues with State of Art

Watermarking techniques [3.13, 3.19] developed for other kinds of multimedia such as audio, video, and images data are not generalized enough to be applicable to 3-D point clouds. The state of art in watermarking point clouds [3.11, 3.21] 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 [3.22] for 3-D meshes that rely only on geometry information for watermarking 3-D 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 3-D point clouds might be solved by using solutions related to other representations of 3-D objects. 3-D 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 3-D meshes. These techniques are based on spatial information or some kind of transform. Spatial techniques [3.3, 3.4–3.9, 3.15, 3.20, 3.23] operate on the data, whereas transform techniques [3.16, 3.18, 3.19, 3.21] apply mathematical techniques to derive information from the original data. Informed or private watermarking scheme for 3-D meshes [3.4 – 3.6, 3.16, 3.18, 3.20, 3.21] are not suitable for our scheme. Some of the blind techniques for 3-D meshes operate on the geometry (vertex or point) and topological (edges) information of the 3-D 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 3-D mesh schemes to 3-D point sampled

geometry, by representing the 3-D point sampled geometry as a 3-D 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 dissertation 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.

3.1.2 Contributions

The dissertation 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 3-D 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) [3.10]. 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 3-D meshes, it is robust against all the above mentioned attacks and also, topology change (e.g. re-triangulation) 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 [3.20], and have readily been used for 3-D meshes in [3.3, 3.7, 3.8, 3.9, 3.15]. However, as mentioned above, ideas related to 3-D meshes are not directly applicable to 3-D 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.

3.2 Scheme Design

Point clouds can be visualized as set of spatial information represented as geometrical information in the form 3-D Points or vertices with positional information (x, y, z). Unlike 3-D 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 [3.10]. 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.

3.2.1 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 3.2, where set of vertices that are not ordered in 3-D 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 (≥) 2, whereas an encoding vertex has degree equal to 1. The graph has two levels: first level defined by the connectivity between vertex 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 3.2, 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.

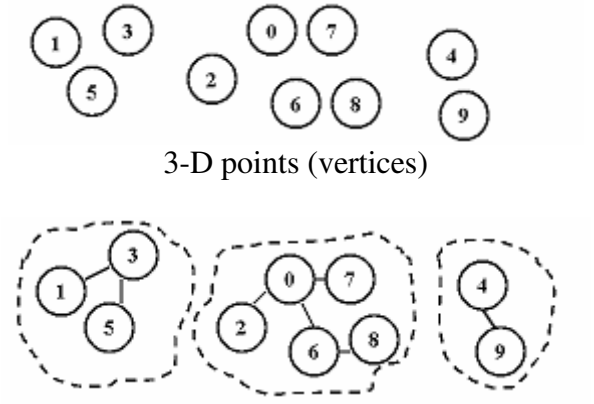


Figure 3.2. 3-D points and their cluster formations

For each cluster, we find a reference vertex. In cases, where the number of cluster heads is greater than or equal to (≥) 2, this reference is the centroid of the cluster heads, and is referred as *r* (see equation 3.1).

$$x_r = \frac{1}{nh}\left(\sum_{i=1}^{nh} x_i\right),\ y_r = \frac{1}{nh}\left(\sum_{i=1}^{nh} y_i\right),\ z_r = \frac{1}{nh}\left(\sum_{i=1}^{nh} z_i\right),\ \text{where } nh \sim \text{number of cluster heads} \quad (3.1)$$

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 3.3, where reference is vertex v_i, and cluster head is vertex v_0.

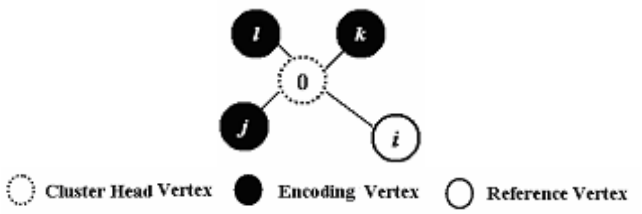


Figure 3.3. Best Case example for encoding, bits encoded ~ 3

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 3.3), 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 3.2.4) 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 3.4, where the example given in Figure 3.2 shows cluster of size '2', being merged to the cluster next to it, using its reference *r*. The net output of the process in observed in Figure 3.5, 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, 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 3.4.

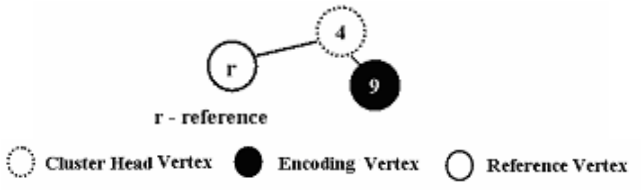


Figure 3.4. Merging Process for cluster size = 2

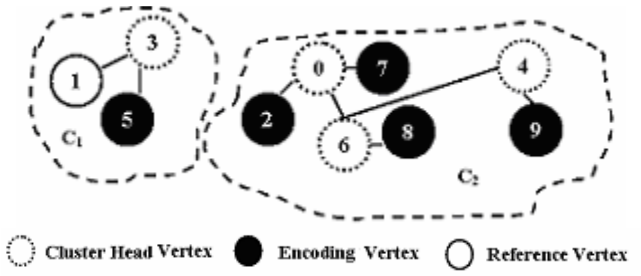


Figure 3.5. Clusters (C_1 and C_2) for Figure 3.2. with encoding vertex, cluster head vertex and reference vertex

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 3.2, Figure 3.5 shows that we have two clusters C_1 and C_2, with bits to be encoded in $\{v_1\}$ for C_1, and $\{v_2, v_7, v_8,$ 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.

3.2.2 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 <degree, bits, greatest angle>. 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 <*Distance*, *Angle*>. 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) $<H_i, H_j \ldots H_k>$ (such that $H_i \prec H_j \ldots \prec H_k$). For every cluster head H_i, we have ordered set of encoding vertices defined by encoding vertex set tuple $(EVST_i) = <v_m, v_n \ldots v_t>$, such that $v_m \prec v_n \ldots \prec v_t$.

- Inter-Cluster Ordering: The clusters are ordered by the tuple <number of vertices, distance>. The distance is defined by the maximum Euclidian distance between cluster heads. For cases where number of cluster heads is '1' (as in Figure 3.3), this distance is given by distance between the reference vertex and the cluster head. The output of this process is a tuple of clusters $(T) = <C_i, C_j \ldots C_l>$ such that $C_i \prec C_j \ldots \prec C_l$.

Figure 3.6 shows the example (presented in Figure 3.2) 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 3.7, 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 3-D 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.

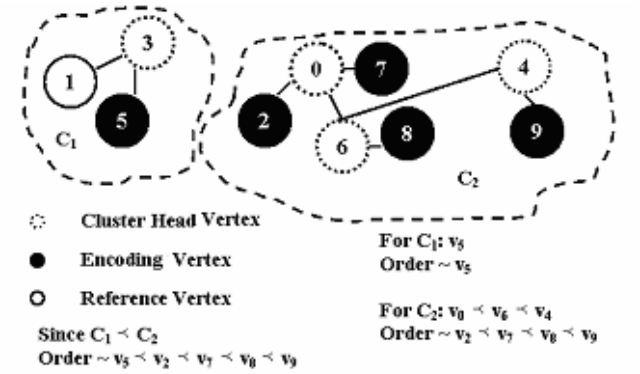


Figure 3.6. Ordering of Points (vertices) for Figure 3.2

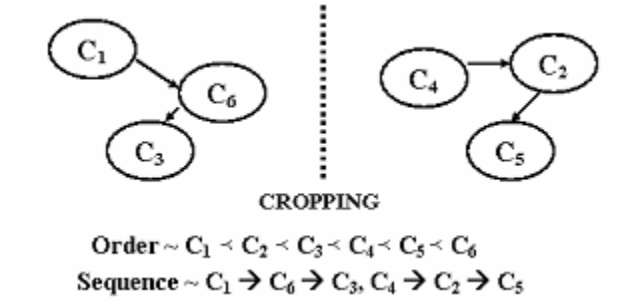


Figure 3.7. Cropping of 3-D Model separates clusters

3.2.3 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 3.1, 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 3.7, the sequence tracing steps results in a tuple $<C_1, C_6, C_3>$, 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 3.7, when cropping occurs, the watermarks

are conserved in the cluster sequence <C_1, C_6, C_3> and <C_4, C_2, C_5>. 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 3.1. Sequence Tracing Steps

Step1 :	1: Bits = Watermark size (*WSize*) 2: *C* = first member in the tuple *T*, where *T* = <C_i, C_j… C_l> 3: New Tuple (*T*) = <C_j… C_l>, Derived by removing C_i from the tuple. 4: Watermark Tuple (*WT*) = < C_i >
Step 2 :	While *(Bits > 0 AND Not Empty (*T*))* Begin *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*)* = <C_j… C_l>, *Derived by removing* C_n *from the tuple.* *3: New Watermark Tuple* <WT> = < C_i,.. C_n> *Derived by adding* C_n *4: Bits = Bits – Number of bits in* C_n 5: C_j = C_n End

3.2.4 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 [3.10].

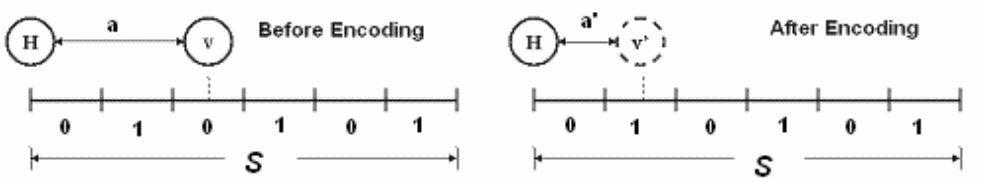


Figure 3.8. Logical representation of bit encoding using scale *S*, encoding vertex *v* and cluster head *H*. Encoding bit 1 changes vertex *v* to *v'*, and Euclidian distance from *a* to *a'*

Encoding is explained in Figure 3.8, 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 3.8, 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*.

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 3.8). 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 3.8. During the encoding process, *S* is divided into intervals (see Figure 3.8). 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 3.8 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 3.9, 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.

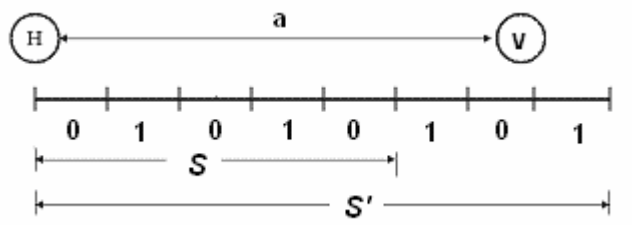


Figure 3.9. Variant *a* greater than (>) invariant scale *S*,
S is changed to *S'* to make scale > *a*

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 (3.2) and (3.3)

$$Scale\ S' = factor * S, \text{ where } factor \geq 1 \tag{3.2}$$

$$Interval\ Size = \frac{S}{p},\ p > 0,\ p \sim \text{number of intervals} \tag{3.3}$$

In cases, where *S* is large, and the interval size is much greater than (>>) *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 (>>) 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 (3.2) and (3.3).

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.

3.2.5 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 3.4) measures the similarity between original watermark w, and detected watermark w'.

Detection Ration (DR) $\geq$ Threshold, where DR = $\frac{\sum_{i=1}^{wsize}(w(i) * w'(i))}{wsize}$, $wsize$ ~ watermark size in bits, $w(i)$ ~ i^{th} bit of original watermark w, $w'(i)$ ~ i^{th} bit of detected watermark w' (3.4)

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.

3.3 Problems and Resolutions

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

3.3.1 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 3.8 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.

3.3.2 Pathological Case

This is a case, where all clusters are of size '2' (see Figure 3.10). 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 3.10, 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 3.11.

In cases, where two vertices form a cluster (e.g., {(v_1, v_5), (v_2, v_3)} in Figure 3.11), 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 3.11 apply equation (3.1) 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.

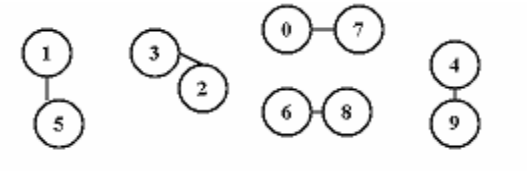


Figure 3.10. Clusters of Size ~ 2

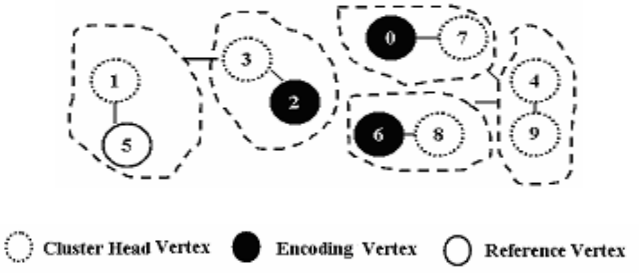


Figure 3.11. Clusters of Size (> 2) formed from clusters size ~ 2.

In order to encode, we use the idea presented in Sub-section 3.2.4 (see Figure 3.8), where clusters of size '2' are encoded using a reference. The encoded vertices in Figure 3.10 can be visualized in Figure 3.11.

3.3.3 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 [3.10], 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 3.12, '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 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.

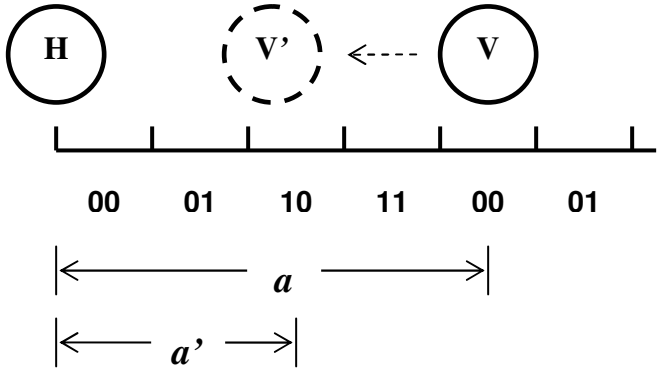


Figure 3.12. Generalization of QIM [3.10] to encode 2 bits per encoding vertex *v*

3.4 Scheme Analysis

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

3.4.1 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 (3.5).

$$\text{Hiding Capacity} = \sum_{i=1}^{nc} bits_i \text{ , } nc = \text{number of clusters, } bits_i = b*nvc_i, \; b \sim \textit{encoding factor}$$

defined as bits per encoding vertex, $nvc_i \sim$ number of encoding vertices for i^{th} cluster (3.5)

As already observed in Sub-section 3.2.1, 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**(*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.

3.4.2 Time Complexity

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

1) *Cluster formation*: Using data structure such as KD-trees [3.1], 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 [3.12]. In case, there are *nc* number of clusters of maximum size *t*, ordering requires O (*nc. 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 [3.1], 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)) ~ O ($n \log n$).

3.4.3 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 3.1) 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 ~ 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 $2^k.2^t$.

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 dissertation 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.

3.5 Experimental Analysis and Discussion

The scheme has been implemented in C, and tested in UNIX and Windows operating system. We have used 3-D mesh models [3.17], 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:

3.5.1 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 (3.6) [3.23] 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 = number of vertices, Data D, Watermarked Data D' (3.6)

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 loosing 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* in 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.

3.5.2 Performance Analysis

This Sub-section analyses the performance using 3-D models such as Stanford bunny, horse and Porsche car. To encode a 3-D model, we *benchmark* the scale, such that the distortion and bit error rate induced due to encoding based errors are minimal. Table 3.2 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 3.2.4, by decreasing the interval size i.e. dividing the scale into larger number of intervals.

Table 3.2. 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

In addition, it also shows the observation in Sub-section 3.3.1 - 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.

3.5.2.1 Hiding Capacity and Distortion Analysis

Different 3-D models (see Figure 3.13) 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 3.3) that the SNR is greater than (>) 100 dB. Visual inspection (see snapshots in Figure 3.13) 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 [3.10] as suggested in Sub-section

3.3, the encoding factor can be improved. Table 3.4 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 3.4) 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 3.3. Encoding analysis for 3-D Models with benchmarked scale (factor = 280 and number of intervals ~ 5 x10^7, BER ≤ 0.06), watermark size ~ 30 and encoding factor ~ 1 bit/encoding vertex

3-D 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

Table 3.4. Encoding Analysis for 3-D Models different benchmarked scales (factor, number of intervals, BER ≤ 0.09), watermark size ~ 30 and encoding factor ~ 10 bits/encoding vertex

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

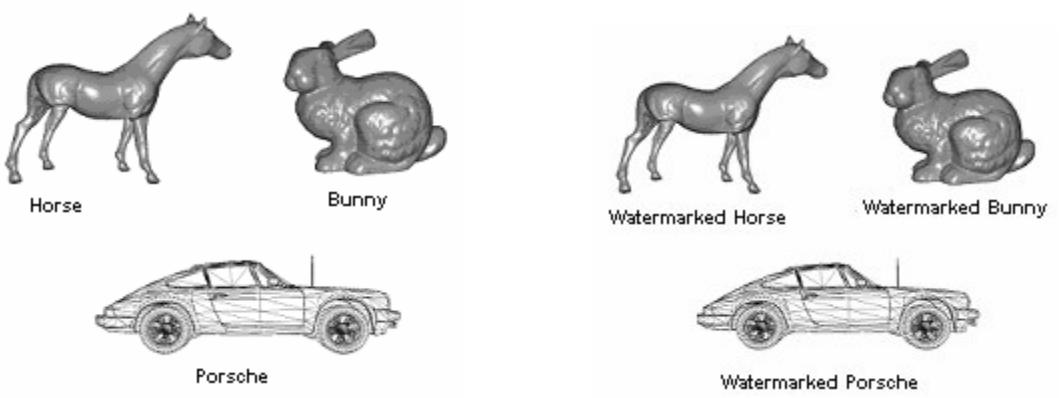


Figure 3.13. 3-D Models

3.5.2.2 Robustness Analysis

Robustness analysis can be done on the basis of the following *attack models*. Based on the region in 3-D models, we can define attacks as local or global. Global attacks can be formulated by impacting the vertices in the 3-D models in a random fashion, whereas local attacks try to attack vertices situated in a sub-part only of the 3-D model. For our experiments, we choose the following models of attacks, given in Table 3.5 and shown for the Stanford bunny in Figure 3.14. The local attacks were implemented by randomly choosing sub-parts of the 3-D 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 3-D 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 3.5. Different Attack Categorization

Attack Type	Region based
Cropping	*Local*
Simplification	*Global*
Noise Addition	*Global*
Noise Addition	*Local*
Uniform Affine	*Global*

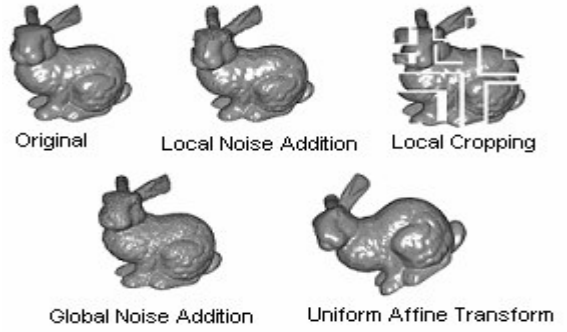


Figure 3.14. Snapshots of Attack Models using the Stanford bunny model

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 3.6, and snap-shots in Figure 3.15. Table 3.7 gives the detection rates for localized cropping attacks. Figure 3.15 gives snapshots of local cropping. In all the above cases, the watermark was detected.

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 3.6 & 3.7 when compared with global attack, given in Tables 3.8 & 3.9. 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.

Table 3.6. Watermark Detection Rate (WDR) for varying % of local 40 dB white Gaussian noise attacks, encoding factor ~ 1 bit/encoding vertex

3-D Model	30 % Attack	40% Attack	50% Attack
Bunny	30.49 %	21.36 %	6.40 %
Horse	68.98 %	62.65 %	57.68 %
Porsche	38.73 %	30.14 %	29.95 %

Table 3.7. Watermark Detection Rate (WDR) for varying % of local cropping attack, encoding factor ~ 1 bit/encoding vertex

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

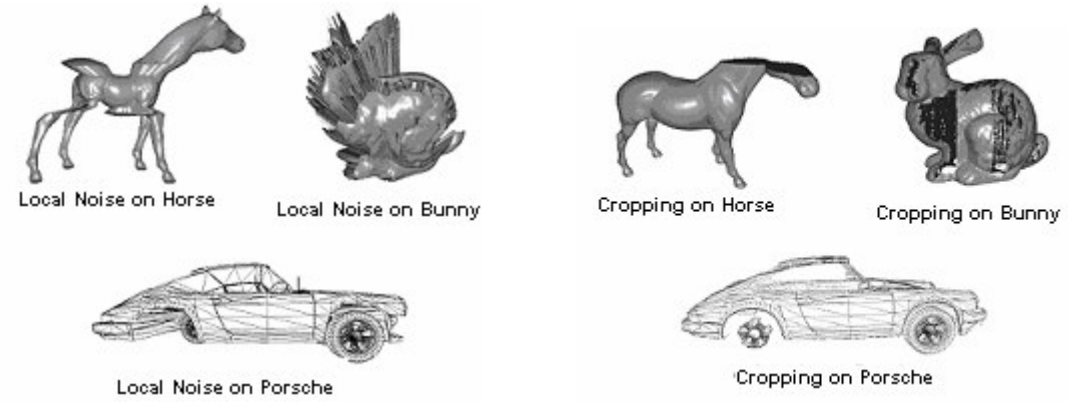


Figure 3.15. Snap-shot of local noise addition and cropping, watermark detection rate (WDR) is > 0 i.e. watermark detected

Table 3.8. Watermark Detection Rate (WDR) for varying % of global 40 dB white Gaussian noise attack, encoding factor ~ 1 bit/encoding vertex

3-D 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 3.9. Watermark Detection Rate (WDR) for varying % of Simplification (global loss) attack, encoding factor ~ 1 bit/encoding vertex

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

Table 3.10. Watermark Detection Rate (WDR) for varying % of global 40 dB white Gaussian noise attack, encoding factor ~ 10 bits/encoding vertex

3-D 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 3.11. Watermark Detection Rate (WDR) for varying % of Simplification (global loss) attack, encoding factor ~ 10 bits/encoding vertex

3-D 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 %

The robustness against global attacks can be improved by using generalized QIM [3.10] 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 3.8 & 3.9 (global attacks with 1 bit/encoding vertex) compared with Tables 3.10 & 3.11 (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 3.6 & 3.7 with Table 3.10 & 3.11, 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 3.12. Percentage Reduction in edges

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

3.5.2.3 Comparative Analysis

We need to compare this technique with mesh based blind watermarking approaches. As already mentioned in Sub-section 3.1.2, 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 3.12, we can see that for different models the approach gives a significant reduction (> 67 %).

3.6 Summary

The chapter presented a spatial blind technique to watermark 3-D 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 3-D meshes, and we can say that it provides robust watermarks against the above mentioned attacks and topology altering attacks (e.g. re-triangulation) as well.

BIBLIOGRAPHY

[3.1] 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

[3.2] Autodesk - Maya: http://www.autodesk.com

[3.3] Alface, P., and Macq, B. Blind watermarking of 3-D 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.

[3.4] Benedens, O. Geometry-Based Watermarking of 3-D Models. IEEE CG&A, pp. 46-55, January/February 1999.

[3.5] Benedens, O. Affine invariant watermarks for 3-D polygonal and NURBS based models. In Proc. Information Security Workshop Wollongon, Australia, 2000, pp. 20-21.

[3.6] Benedens, O. Robust watermarking and affine registration of 3-D meshes. In Proc. Information Hiding Noordwijkerhout, the Netherlands, 2002, pp. 177-195.

[3.7] 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.

[3.8] Bors, A. Blind watermarking of 3-D shapes using localized constraints. In Proc. of 3-D Data Processing, Visualization and Transmission, 2004. 3-DPVT 2004. Proc. 2nd International Symposium on, 6-9 Sept. 2004 Page(s):242 – 249.

[3.9] 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.

[3.10] 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.

[3.11] 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).

[3.12]Cormen, T.H., Leiserson, C. E., Rivest, R. L, and Stein, C. Introduction to Algorithms. Second Edition, ISBN-13: 978-0-262-03293-3.

[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.

[3.14] Fast Scan 3-D http://www.fastscan3-D.com/download/samples

[3.15] Harte, T., and Bors, A. Watermarking 3-D models. Image Processing. 2002. Proceedings. 2002 International Conference on, Volume 3, 24-28 June 2002 Page(s):661 - 664 vol.3.

[3.16] Kanai, S., Date, H., and Kishinami, T. Digital watermarking for 3-D polygons using mutltiresolution wavelet decomposition. In Proc. 6th IFIP WG 6.2 GEO-6 Tokyo, Japan, 1998, pp. 296-307.

[3.17]Large-Geometric-Models-Archive,
http://www.static.cc.gatech.edu/projects/large_models/

[3.18] Praun, E., Hoppe, H., and Finkelstein, A. Robust mesh watermarking. In Computer Graphics (Proceedings of SIGGRAPH 99), pages 49--56, August 1999.

[3.19] Petitcolas, F., Anderson, R. J., and Kuhn, M. G. Information Hiding: A Survey. In Proc. IEEE special issue on Protection of Multimedia Content, 1999.

[3.20] 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.

[3.21] Ohbuchi, R., Mukaiyama, A., and Takahashi. Watermarking a 3-D Shape Model Defined as a Point Set. In Proceeding of CW 2004: 392-399.

[3.22] Uccheddu, F., Corsini, M., and Barni, M. Wavelet-Based Blind Watermarking of 3-D Models. In Proceedings of the 6th ACM MM & Security workshop 2004, Pages: 143 – 154, 2004.

[3.23] Zafeiriou, S., Tefas, A., and Pitas, I. Blind robust watermarking schemes for copyright protection of 3-D mesh objects. Visualization and Computer Graphics, IEEE Transactions on, Volume 11, Issue 5, Sept.-Oct. 2005 Page(s):596 – 607.

CHAPTER 4

WATERMARKING BASED TAMPER DETECTION OF 3-D MOTION DATA

Parag Agarwal and Balakrishnan Prabhakaran

Computer Science Department, EC 31

The University of Texas at Dallas

800 WEST CAMPELL RD.

Richardson, Texas 75080-3021

4.1 Introduction

The advent of Motion Capture systems [4.14] has brought in applications like animation (games, films & TV, education), and life sciences (biomechanical research, gait analysis, rehabilitation, posture, balance and motion control, sports performance). The above applications deal with motion analysis or reusability, and can benefit from having a large repository of 3-D human motions. In cases, where data is tampered, its integrity is lost, and we incur losses in terms of accuracy, effort, time and money. Tampering can be avoided by a data authenticating mechanism, such as fragile watermarking. Fragile watermarking can be achieved by embedding a watermark inside a target data. Tampering is recognized whenever during an extraction process if sub-part of the embedded watermarks is found corrupt.

3-D Mocap data streams consist of motion information related to human joints, and can be represented by their translational and/or rotational information. Developing a fragile watermarking technique for motion data poses the following challenges:

- **Distortions in meaning of data:** Addition of watermarks distorts the original data. This changes the meaning of the data set. The visibility of distortions due to change in the meaning of motion data will fail the imperceptibility criteria of the watermarking scheme.
- **Accuracy of detection:** Data set can be attacked using motion editing operations [4.1], such as noise addition, reordering, and uniform affine transformations. As a result, the meaning of the data set at can change different locations. The watermarking methodology should be able identify the attack, and detect the change accurately.

Accuracy can be achieved by replicating the watermark at different locations in the data set. However, replication can induce more distortions resulting in loss of meaning of the data. Distortions will discourage the replication of the watermark, and eventually impact the

accuracy of the technique. In addition, the fragile watermark technique needs to be storage efficient. Storage efficiency can be achieved by reducing the information required to verify the presence of watermark. This can be achieved by blind watermarking mechanism, which uses the original watermark and key to embed and extract the watermark. Also, since motion data has different formats (global and local), the technique should generic enough to be applicable to both of them. To the best of our knowledge, there is not a single technique that can solve all the problems for 3-D motion data.

4.1.1 Related Work

Authentication based watermarks [4.10] can be classified as fragile watermark or semi-fragile watermarks. Fragile watermarks cannot withstand any alteration of the data. However, semi-fragile watermarks can withstand meaning preserving changes, while not tolerating meaning altering changes. These techniques [4.10] can be described as spatial and transform domain. Transform domain operates on the frequency components of the subset of the data, as compared to the spatial techniques that operate on the original data. Several authentication based approaches using watermarks (fragile and semi-fragile) have been proposed for images [4.4, 4.5, 4.6 and 4.8], audio [4.3], video [4.11] and 3-D models [4.7, 4.9 and 4.12]. However, none of the approached are generalized enough to be applied to motion data. To the best of our knowledge there is no work done in this regards for fragile watermarking motion data streams.

A non-watermarking method [4.13], explains a scheme of extracting reference code, which is used to detect tampering. However, this technique requires extra storage, which makes the scheme not scalable. Therefore, it is eminent that we need a novel scheme to tamper proof motion data.

4.2 Scheme Design

We propose to watermark each joint data of motion data matrix separately, since alteration to the data-set can be tracked on a per joint basis. Encoding inside each joint is done by visualizing it as clusters of 4 points. It can be observed that D_i can be represented as point (P_i) in 3-D space (see Figure 4.1). Since D_i (s) are already totally ordered with respect to time, we can say that points P_i and P_j $(i < j)$ are adjacent to each other. We can identify non-intersecting cluster of points whose sizes are multiples of four. Bits are embedded inside the clusters by identifying four points at a time. For each of the four points given, the same bit is encoded using two set approach, where in fist set we have one point, and the other set the remaining three points. One point and three point encoding is done using an extension of quantization index modulation [4.2]. The following sections describe the technique in detail.

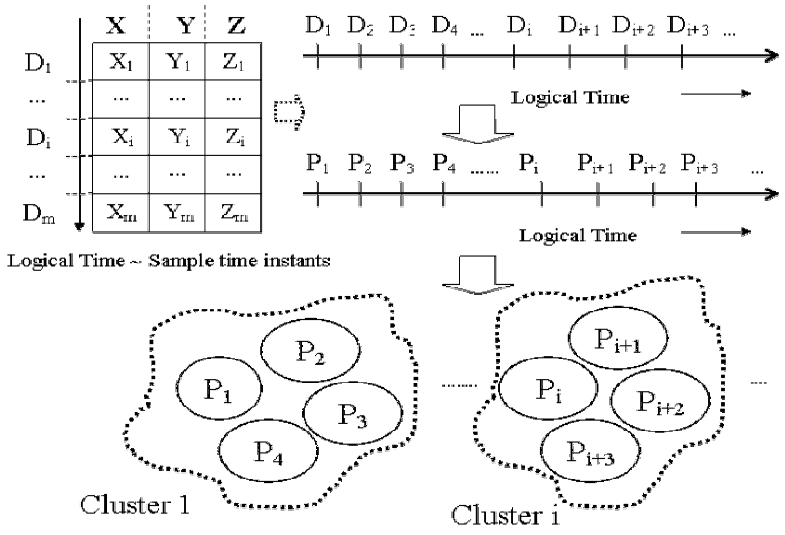


Figure 4.1. Visualization of 3-D motion data as 4 point clusters

4.2.1 Cluster Based Encoding

Each cluster is of size $4k$ points or samples (where $k_{min} = 1$), and can encode *'k'* bits. For simplicity we assume cluster size = 4. These bits are encoded in the sequence (or order) in

which the points occur. For example, to encode each bit we use four points $\{P_i, P_{i+1}, P_{i+2}, P_{i+3}\}$, where first point subset $\{P_i,\}$ is taken for 1-point encoding, and the subset is used for $\{P_{i+1}, P_{i+2}, P_{i+3}\}$ 3-point encoding.

We can visualize this situation in Figure 4, where points $\{P_1, P_2, P_3, P_4\}$, represented as vectors are shown. The following explanation is used to abstract the idea of encoding for 3-point and 1-point.

3-Point Encoding: It can be observed that affine transformation preserves *proportions* of lines. The lines can be described as scalar quantities q_1 and q_2, and the ratio (q_1/q_2) must be invariant to affine transformations. The scalar q_1 and q_2 can be realized using three points represented as two vectors, as shown in Figure 4.4. When we consider three points $(P_2, P_3$ and $P_4)$, we have can have two vectors such that one point is common among these vectors. The magnitudes of these vectors give us two scalars (*Euclidian distances between the points*) whose ratio stays invariant to affine transformations.

$$B\,(q_1, q_2) = bit, \text{ where } bit = \{0, 1\}. \tag{4.1}$$

In order to encode the bit information inside a data set, we use the function '$B\,(q_1, q_2)$' whose inputs are the two scalar quantities. A bit can be encoded by observing the output of the equation (4.1). In case, a required bit condition is not met, as shown in Figure 4.2), where the expected bit = 1. To handle this case, we substitute the point P_3 by another point to encode the expected bit, resulting in change in scalar quantity $|P_2P_3|$.

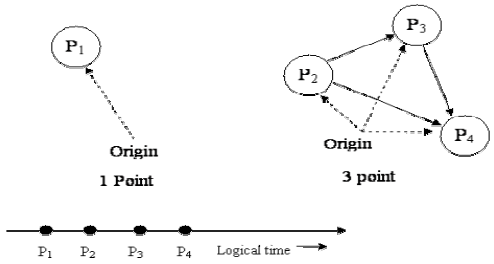


Figure 4.2. Vector representations of points to be encoded

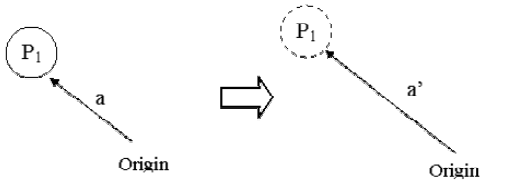


Figure 4.3. 1-point Encoding

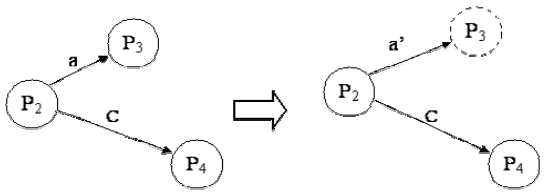


Figure 4.4. 3-point Encoding

Handling Pathological Cases: There can be cases where points in the data set have same values i.e. points P_i and P_j belong to the same cluster and have equal values. For such cases, we have to choose among these points for encoding, which may result in contention. This situation can be avoided either by excluding these points from the encoding process or by perturbing them to achieve an encoding.

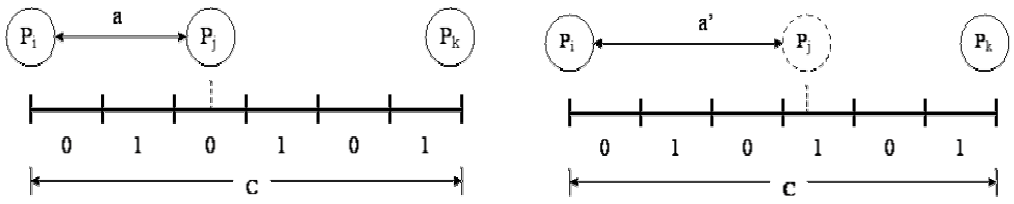
A single point P_1 in 3-D space can be visualized as a vector (see Figure 4.3) which results in only one scalar – magnitude (in Figure 4.3, $q = |P_1|$), which is variant to affine transformation.

1-Point Encoding: In this encoding scheme a bit is encoded inside a single point by subjecting values x, y and z to be perturbed in order to encode '1' or '0'.

$$Bit\ (q) = b,\ b = \{0, 1\}. \tag{4.2}$$

A bit can be encoded by observing the output of the equation (4.2). In case a required bit condition is not met, as shown in Figure 4.3, where the expected bit = 1. To handle such a case, we substitute the point P_1 by another point to encode the expected bit, resulting in change in scalar quantity $|P_1|$. The functions represented in equations (4.1 and 4.2) can be implemented using a bit encoding scheme, which uses quantization index modulation (QIM) [4.2]. QIM has also been customized to watermark 3-D Models for tamper detection [4.7, and 4.9]. The customization of QIM in our case is explained as follows:

Bit Encoding Scheme. As shown in Figure 4.5 (a & b), we take two points (P_i and P_k), the Euclidian distance '*C*' between them stays invariant. However, the distance '*a*' between P_i and P_j is variant, and changes due to encoding as explained below.



(a)Before change in bit (b) After change in bit

Figure 4.5. Logical representation of bit encoding applied to points P_i, P_j, and P_k

The scalar '*C*' can be divided into '*p*' number of equal intervals, where $p_{min} = 2$. The interval set of 0(s) is identified by S_0 and 1(s) by S_1, (S_i: $i \in \{0, 1\}$). Let *Pos* (P_j): position of P_j on side *'C'*. The bit '*i*' can be determined based on the position of P_j on side '*C*' using the following rules:

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

As observed in Figure 4.5 (a&b), which is a case, where the bit to encode is '1', but P_j lies on an interval where the corresponding bit is '0'. As a consequence, we need to shift the position of P_j, which results in change of the scalar $|P_i P_j|$ from (a to a').

The above explanation assumes scalars as length of vectors represented in Figure 4.2, and can be customized for 1-point and 3-point encoding as shown below:

Customization for 1-Point Encoding: We can visualize P_i as the '*origin*' in Figure 4.3, and P_k as an assumed point for which scalar 'C' is fixed. Point P_1 is equivalent to point P_j, and its scalar representation is 'a'. The function *Bit* (q) is implemented by assuming ($q = a$) and the bit information is determined by *Pos* (P_j).

Customization for 3-Point Encoding: We can visualize P_i as the P_2 in Figure 4.4, and P_k as P_4. The function *Bit* (q_1, q_2) is implemented by assuming have '$q_1 = C$' and '$q_2 = a$', and Pos (P_j) determines the output of the function.

In both the cases, change in P_j (P_1 for 1-point and P_3 for 3-point) is determined by the function Pos (P_j). For 3-point technique, we observe that the ratio $q_1 = |P_i P_j|$, and $q_2 = |P_i P_j|$, which implies that once the bit is encoded it is invariant to affine transformation, since (q_1/q_2) is invariant. The intervals on the scalar 'C' have significance in making the technique sensitive to tampering, and imperceptibility of the watermark.

The following ideas are important in customizing the scheme to data sets, in order to control distortions and accuracy of tamper detection.

Controlling Distortions. Smaller interval size implies that P_j has to be displaced by a smaller amount during the bit encoding process. This will reduce the distortion during embedding process, which would make the watermark more imperceptible. During encoding, we need to choose an interval size, and number of such intervals to define the scale along

which a single point is encoded. The size of the scale is given by the equation (4.3).

$$Scale = Interval\ Size * Number\ of\ intervals \tag{4.3}$$

For 3-point encoding the scale is determined by an invariant side $|P_iP_k|$, and the interval size can be determined from equation (4.3), by dividing the scale by the number of intervals. In case of 1-point encoding, we derive the scale based on the choice of interval size and number of intervals, and this choice is explained as follows:

Interval Size Selection: In cases, where the interval size is greater than scalar quantities related to points, the points would be in the first interval only, and would have a bit '0' (see Figure 4.5(b)). Since bits can be either '0' or '1', in cases '1' is required to be encoded, distortion will become more probable. Therefore, in order to avoid such a situation, the size (see equation 4.4) should be less than or equal to the minimum size of a scalar quantity of all the points being encoded.

$$Interval\ Size \leq Min\ (|Qi|),\ Qi \in\ Data\ Set. \tag{4.4}$$

Number of Intervals Selection: In cases, where scalar quantities related to points are greater than the scale, the bit related to them is always the same. Since bits can be either '0' or '1', in cases '1' is required to be encoded, distortion will become more probable. Therefore, in order to avoid such a situation, the size of the scale should be greater than the largest size of the scalar quantities. To guarantee the same, the number of intervals can be determined by equation (4.5).

$$\text{Number of intervals} > \frac{Max(|\ Qi\ |)}{IntervalSize},\ Qi \in\ Data\ Set. \tag{4.5}$$

Since we are encoding, the maximum or minimum scalar quantities, as given in equations (4.4) and (4.5) might change. As a result, decoding might give us different bits, resulting in

erroneous condition. To avoid this situation, for equation (4.5), number of intervals is infinitely large, implying that the scale is be infinity (a very large number for practical usage). The interval size in equation (4.4) can be based on a point, where encoding is not enforced. To guarantee that any other point during encoding does not substitute this point, we always increment the scalar quantities for encoding. Also, if there are several similar points that are fit for interval size criteria, we take only the first occurrence. The choice of this point has the added advantage of tamper detection. Since in cases, where this point is perturbed, the interval size is changed, resulting in flipping of encoded bits for other points. This fact will act as a discouraging factor for the adversary to change this point.

The above factors will help tailor the encoding scheme to adapt to the data set given, and would be beneficial in reducing the distortions.

Accuracy in Tamper Detection. Any slight displacement to point P_j (see Figure 4.5) to another interval might toggle the bit encoded. In cases, where interval size is large, the probability is less since a larger displacement is required to change the bit. This could result in false negatives, and can be avoided by reducing the size of intervals. Therefore, smaller interval lengths can test any changes to the bit encoded in 1-point or 3-point scheme. However, in cases, intervals are smaller; any change resulting from tampering may cause the point P_j to be shifted into an interval which has the same bit. As a result, we cannot detect tampering which leads to false negatives. Such effects are more probable if the interval size is reduced, since the probability of presence in '*n*' intervals is '*1/n*' which decreases as '*n*' increases.

Watermark Size Based Probabilistic Model: In cases where all the bits do not change, a watermark of size '*WSize*' can be detected. The probability of detection can be expressed as

'p^{WSize}', where 'p' is the probability that bit has not been flipped. The probability of failure to detect a watermark (P_f) can be expressed as equation (4.6).

$$P_f = 1 - p^{WSize} \tag{4.6}$$

Since ($p < 1$), from equation (4.6), it can be inferred that the probability of failure is least when watermark size is equal to '1'. In addition, we also observed that by increasing the size of the watermark size, we increase the likelihood to detect a watermark. This is a positive sign for tamper detection, since loss of watermark implies tampering, and this can be concluded with a high probability. Once it is confirmed the watermark has been tampered, we can localize the search to the bit that has changed. As shown above, the change in bit information depends on the false negatives during an attack, since the shift in point location could result in the same bit. Therefore, from the above discussion it can be concluded that although the false negatives might be present due the small sized intervals, we can increase our chances of tamper detection by choosing a watermark of sufficient size. We can also conclude that larger the number of scalar quantities used to encode a bit, more accurate is the detection. Since the number of scalar quantity corresponds to the length of the watermark.

4.2.2 Decoding and Verification

The decoding process is similar to encoding process. Verification is done by identification change in the bit information. A change in the bit information (1-point or 3-point encoding) is reflected as a compliment of the bit. This helps in identifying the location of the attack.

In order to identify the attacks, the following rules can be followed:

Affine attack rule: An affine attack is identified by a change in the 1-point bit, and no change in the 3 point bit. This is because the 3-point encoding is robust to affine transformation, as compared to 1-point encoding.

Noise addition rule: These attacks can be identified by change in both 1-point and 3-point encoding scheme.

Reversing order attack rule: In this case the order of occurrence of points is reversed in logical time. This can be identified by detecting information in the reverse time. In addition, it will change the bit information for 1-point and 3-point encoding as well.

Combined attack rule: The above mentioned attacks can be launched at the same time, resulting in all indications.

4.3 Extensibility to Semi-Fragile Watermarks

Fragile watermarking schemes detect tampering and localize the tampered part of the data set. In our case, we are trying to classify attacks and detect tampering. In case motion data is given uniform affine transform or noise is added due to lossy compression, fragile watermarks are lost. As a result, it would give us false indications, even though the semantics of the data set is unaltered. The following discussion explains a watermarking scheme that is robust against user related operations, but tries to identify attacks. The scheme suggested can be converted into a semi-fragile watermarking scheme by using motion data information in the localized domain. In the localized domain we transform the transitional information to a local co-ordinate system such that the watermarks are embedded per joint information in the order of time. For every joint information (x_t, y_t, z_t) at time 't', we localize this information based on reference joint information (x_r, y_r, z_r), which acts as the origin to this co-ordinate system. Such a joint can be the pelvis joint, or any other joint. In case, uniform affine transformations are applied, it does not change the localized information, thereby resulting in robustness against uniform affine transformations. Once the transformation is done, we embed the watermark per joint data in the localized geometry information. To encode and

decode bits we use the 1-point encoding scheme. Since, in this approach we take local transform, we achieve robustness against uniform affine transforms. In case, different affine transforms are given to each joint, watermark are lost indicating as attack. The methodology suggested above does not cater to the problem of compression schemes [4.21] for Mocap data that can be lossy in nature. Since lossy schemes can add noise to the data set, the watermark-based bits might flip. In order to make the watermark robust to noise addition, we can make the interval size larger, since larger interval size would imply that the scalar values would have to be changed by a larger amount in order to shift the point to an interval where the encoded bit will flip. As a result, larger interval sizes will be robust to noise additions. However, since the bit is not changed due to noise addition, we cannot detect the tampering. Also, in order to reduce distortion, we prefer smaller interval sizes. Therefore, factors such as robustness, tampering and distortions make it difficult to come up a scheme that caters to lossy compression. Since the encoding entirely depends on the scale, we need to search for a scale that makes the scheme robust to noise addition, minimize distortion and can still detect tampering with a high accuracy. We assume that noise added will change bits if they change the semantics of the data set. From this assumption and the above observation about robustness, we can conclude that certain interval sizes or number of intervals will result in more robustness to noise and tampering can be observed when semantics change. This can be explained as follows: These noise values correspond to the lossy compression schemes. Since the semantics are not altered due to addition of noise, the semantics are conserved for these interval sizes or given number of intervals. If noises levels are increased beyond semantics conserving noise observed for lossy compression, we assume that tampering takes place. Therefore, we can say that for a given interval size or number of intervals we can detect

tampering that are robust to lossy compression and minimizes the distortion as well. The challenge is to find such the number of intervals or interval sizes that achieve our goal. We can make this search linear O (n) or logarithmic O (log n) by doing a binary search.

4.4 Experiments and Results

The watermarking scheme has been implemented in Matlab 7.0.4, and applied to data samples that were collected from University of Texas at Dallas Motion Capture lab (Vicon [4.14]). The experiments were done on a motion matrix, which is a dance sequence with (4286 frames captured at 120 frames/sec for 19 joints = 81434 points or samples). The model is justified for performance analysis, since it consists of joints moving in varying directions at different time intervals, thus giving diversity in motion data. Other motion data such as karate actions, t-pose, exercise, and walk are used to analyze accuracy and distortions.

The experiments in this section increase the interval size or increase the number of intervals. Both operations imply the same purpose. Measurements for distortion are taken as mean value of the following metrics.

Signal to Noise Ratio (SNR) can measure the distortion produced. Higher the distortion less imperceptible is the watermark. SNR for 3-D motion data is calculated in equation (4.7) (RMS ~ root mean square).

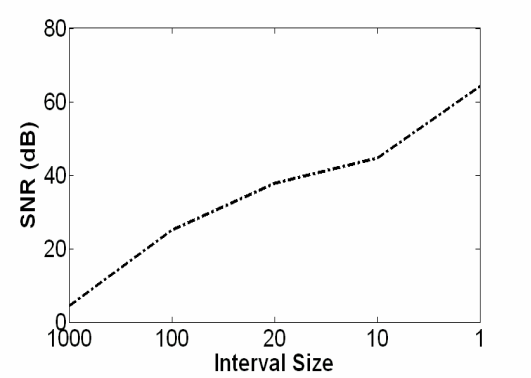
$$SNR\ (M, M') = 20\ Log_{10}\ \frac{RMS(M)}{RMS(M - M')} \tag{4.7}$$

Accuracy during tampering is detected using the mean values of the following metric:

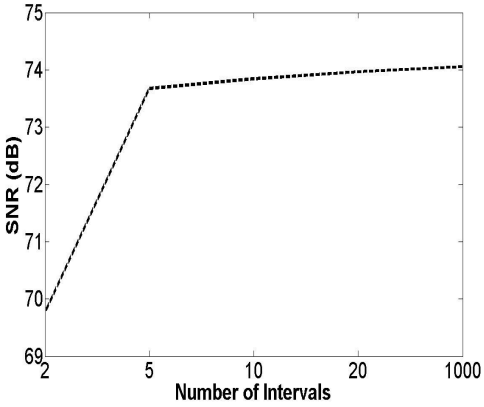
Detection Rate (DR) can be a measure of the accuracy with which tampering is detected. It is defined as the ratio of number of error detected to the total number of possible errors. DR is primarily equivalent to the probability of failure (P_f), as given in equation (6), as it measures the failure to detect the watermark.

4.1 Performance Analysis

The following subsections give a performance analysis of the scheme for different parameters of the scheme.



4.6 (a) 1-Point Encoding



4.6 (b) 3-Point Encoding

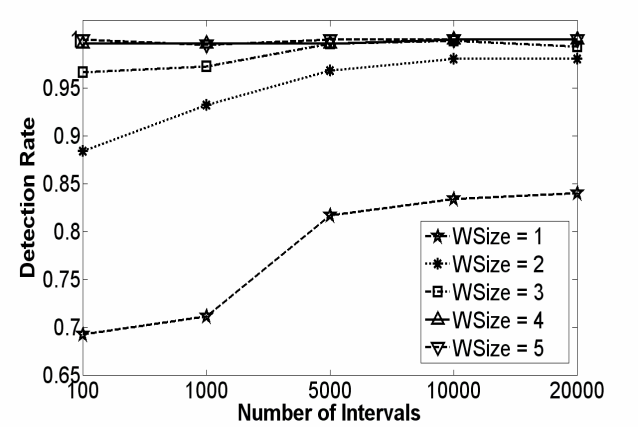
Figure 4.6. Impact of interval Size on distortions due to 1-Point encoding

Distortion Analysis. It can be observed from Table 1 that for different motion types we have distortion > 100 dB. In order to analyze distortion for 1-point and 3-point encoding, we analyze them separately as follows:

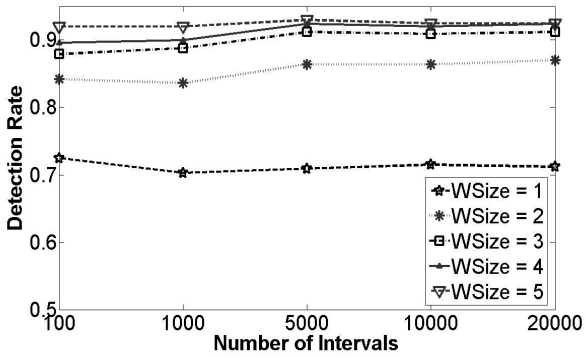
1-Point Encoding: The data being subject to watermark varies in the range [1 900]. An attempt to encode a watermark inside it sees a distortion maximized (see Sub-section 4.4.1) when interval size is '>' 900. This can be observed in Figure 4.6(a), where the interval size = 1000 has maximum distortion. By increasing the interval size the distortion decreases.

3-Point Encoding: As observed in Sub-section 4.4.1 reduction in interval length results in reduction in distortion (see Figure 4.6(b)), and it becomes consistent for the range (74 to 75 dB), after the number of intervals is increased '>' 5.

Accuracy Analysis for Encoding. Affine transformation and noise addition attacks are uniformly carried out on the data set. It can be observed from Table 4.1 that for different motion types, the scheme gives high detection rate. This Sub-section also proves the claims presented in Sub-section 4.4.1, which show that larger sized watermark help increase the accuracy of tamper detection. Also detection ability increases by reduction in interval size, which can be done by increasing the number of intervals. Figure 4.7 show the results for tampering based on noise, and affine transformation. The general trend in both the graphs shows us that increase in watermark size increases the accuracy (high value of detection rate) of tamper detection. We also carried out reverse ordering attacks and detected them with 100 % accuracy.



4.7 (a) 1 dB Noise Attack



4.7 (b) Affine Attack

Figure 4.7. Detection rate Vs Number of intervals for varying watermark size (WSize)

Robustness for Semi-fragile nature: For uniform affine transformation, we subjected the motion data files to format conversion from local to global [4.14]. The results showed that bit error rate was 0, which showed 100 % robustness to uniform affine transformation. As mentioned above (in attack models) for non-uniform affine attacks are modeled as noise additions, we can model this as noise addition as follows. For noise addition, attacks we analyze the bit error rates for different intensity of noise measured in decibel. Figure 4.5 shows us that the bit error rate decreases with the increase in interval size by reducing the number of intervals. This is consistent with Sub-section 4.5 which states that larger interval sizes will observe less bit errors as compared to smaller interval sizes, which shows us that they are more robust to noise additions. It also shows us that the as the noise intensity is reduced the bit error rate falls and is the least for the scale with the smallest number of intervals ‘1000’. Therefore, in order to choose a scale which is sensitive to noise addition, our choice is related to number of intervals for which bit error rate maximizes, and this case it is ‘10000’.

Table 4.1. Analysis of Sample Motion files Encoding, Watermark Size = 5

Motion Type	# Samples	SNR (dB)	1dB Noise (DR)
Dance	19000	86.002	0.99
Walk	17594	112.1351	0.99352
Karate 1	10928	104.3759	0.99631
Karate 2	11039	112.2567	0.99828
Exercise	19000	104.4280	1
TPose	19000	104.4280	1

Table 4.2. Percentage increase in Distortion for varying noise attack and different scales

# of Intervals	.1 dB Noise	1 dB Noise	10 dB Noise	20 dB Noise	50 dB Noise
NInt = 1000	0.10325	0.092848	0.032673	0.010199	0.00030458
NInt = 5000	0.21392	0.18515	0.066839	0.020027	0.00061366
NInt = 10000	1.3441	1.2059	0.37157	0.10718	0.0032908
NInt = 50000	3.0996	2.7637	0.73367	0.20528	0.0062431
NInt = 100000	8.5344	7.3119	1.8184	0.4681	0.012698

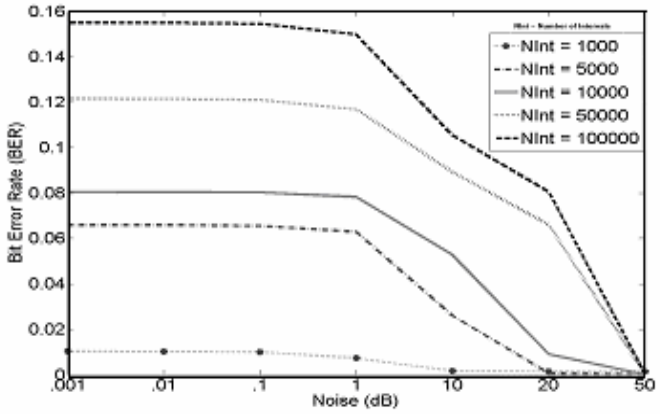


Figure 4.8. Bit Error Rate for varying Scale (Number of Intervals – NInt) and Noise Attack

From the above discussion, we can say that scales with larger number of intervals are more sensitive to distortion, and have a higher bit error rate, and try to minimize distortion during the encoding phase. This makes the scales with larger number of intervals more suitable for error detection. But this model is more suitable when the threshold is ~ 0. However, as we choose a larger threshold, the tolerance level for distortion can be increased, and we will chose a different scale instead of a very sensitive scale. It can be noted that distortions vary from one motion data file to the other, so the threshold choice and change in semantics can

also vary. This can be observed in Tables 4.4- 4.11 where for different scales, distortions are different and so is the percentage increase in distortions. Based on these ideas, we can benchmark the system for the kind of robustness and distortion it can handle, resulting in different data dependent semi-fragile schemes.

4.5 Summary

In order to handle the problem of tamper detection, the dissertation identified a fragile watermarking scheme that could classify and detect with high accuracy attacks such as uniform affine transforms, order reversal and noise addition. The scheme was improved in order to handle false alarms against uniform affine transforms and noise additions that do not alter the semantics of motion. However, these semi-fragile watermarks are shown to identify high intensity noise additions and order reversal attacks.

Since we were watermarking the positional information by considering points in a joint, we were watermarking each joint's trajectory, and this made our technique applicable to different format representations (global and local). Also, with semi-fragile watermarks the technique is still generic as we can always convert global to local formats without losing the watermark and carry out tamper detection procedure.

BIBLIOGRAPHY

[4.1] Autodesk – Motion Builder, www.autodesk.com

[4.2] B. Chen and G. W. Wornell, “Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," IEEE Trans. on Information Theory, vol 47, pp. 1423-1443, May 2001

[4.3] C. S. Lu, H. Y. Mark Liao, and L. H. Chen, "Multipurpose Audio Watermarking", Proc. 15th Int. Conf. on Pattern Recognition, Barcelona, Spain, Vol. III, pp. 286-289, 2000

[4.4] D. Kundur and D. Hatzinakos, "Digital watermarking for telltale tamper proofing and authentication", Proceedings of the IEEE, vol. 87, no. 7, pp. 1167--1180, July 1999

[4.5] E.T. Lin and E.J. Delp, "*A review of fragile image watermarks*," in Proc. of ACM Multimedia & Security Workshop, Orlando, 1999, pp. 25--29.

[4.6] E. T. Lin, C. I. Podilchuk, and E. J. Delp, "Detection of Image Alterations Using Semi-Fragile Watermarks," Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents II, Vol. 3971, January 23 - 28, 2000, San Jose, CA

[4.7] F. Cayre, O. Devillers, F. Schmitt and H. Maître, Watermarking 3-D Triangle Meshes for Authentication and Integrity, INRIA Research Report RR-5223, Jun. 2004

[4.8] J. Fridrich, M. Goljan, and A. C. Baldoza, "New fragile authentication watermark for images," in Proc. IEEE Int. Conf. Image Processing, Vancouver, BC, Canada, Sept. 10--13, 2000.

[4.9] H.T. Wu and Y.M. Cheung, A Fragile Watermarking Approach to 3-D Meshes Authentication, Proceedings of the 7th Workshop on Multimedia & Security (ACM'05), pp. 117-123, 2005.

[4.10] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Mathew Miller, Digital Watermarking: Principles & Practice (The Morgan Kaufmann Series in Multimedia and Information Systems)

[4.11] Minghua Chen; Yun He; Lagendijk, R.L., "A fragile watermark error detection scheme for wireless video communications," *Multimedia, IEEE Transactions on*, vol.7, no.2pp. 201- 211, April 2005

[4.12] M.M.Yeung and B.-L.Yeo, "Fragile watermarking of three dimensional objects," Proc. 1998 Int'l Conf. Image Processing, ICIP98, volume 2, pp. 442--446. IEEE Computer Society, 1998.

[4.13] P. Agarwal, K. Adi, B. Prabhakaran, “SVD-based Tamper Proofing of Multi-attribute Motion Data”, Proc. of The 12th International conference on Distributed Multimedia Systems (DMS) ‘06

[4.14] Vicon, http://www.vicon.com

CHAPTER 5

TAMPER DETECTION AND CORRECTION IN 3-D MOTION DATA

Parag Agarwal and Balakrishnan Prabhakaran

Computer Science Department, EC 31

The University of Texas at Dallas

800 WEST CAMPELL RD.

Richardson, Texas 75080-3021

5.1 Introduction

Sophisticated motion capture facilities (that were actually developed for helping cartoon animators) aid in mapping the complex human motion in the three dimensional (3-D) space. These facilities require a participant to wear specialized markers on them. Motion tracking systems then track these specialized markers and map the participant's motion in the 3-D space. There are different types of these markers that can be used such as reflective markers, magnetic sensors, and RFID-based (Radio Frequency Identification) markers. The types of motion tracking systems also vary based on the type of markers used – Infrared-based cameras for reflective markers, magnetic systems using a centralized transmitter and a set of sensor relays, and RFID trackers. Figure 5.1 shows the reflectors on the participant's body mapped in the 3-D space. As the participant keeps moving, the cameras track the body movement and map the body joints/segments in the 3-D space. Such a 3-D mapping of the human body motion helps in analyzing or comparing the motions. For instance, Figure 5.2 shows the comparison of the movement of the foot segment for a similar motion performed by two different participants.

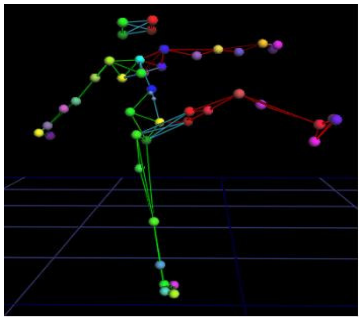
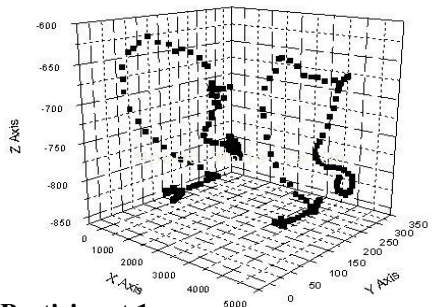


Figure 5.1. Reflectors Mapping in 3-D Space

Motion capture system has found different applications in medical fields such as biomechanical research, rehabilitation, posture, balance and motion control, and sports

performance. The above applications deal with motion analysis or reusability, and can benefit from having a large repository of 3-D human motions. In cases, where data is tampered, its integrity is lost and we incur losses in terms of accuracy, effort, time and money.



Participant 1 **Participant 2**

Figure 5.2. Foot Segment movement of 2 different participants for the same motion

Mocap data is multi-attribute, and can be described as $M_{m \times nc}$ matrix (with $nc \sim 57$ columns and m can be any number of frames, see Figure 5.3 for human body described in terms of joints whose motion results in motion data expressed as a matrix in Figure 5.4) (for the comma separated value (*.csv*) format of Vicon IQ [5.22]), with columns representing different attributes such as rotational and positional data of skeleton joints, and rows representing the changing values over time. Adversaries can use motion editing techniques [5.22] such as to tamper data. The attacks on a matrix M can be categorized as follows:

- **Column or Row Shuffling:** Either Columns or rows are shuffled.
- **Column or Row Element Shuffling:** In such kind of attack, elements of a given column or a row are shuffled.
- **Combinations:** These can be combination of the above mentioned attacks e.g. row-column shuffling.
- **Random Attacks:** Any other kind of attack that is not detectable is termed as random noise attack, wherein elements of the matrix seem to have changed.

The above attack patterns will result in joint motions following different patterns from their original. Since human body motion is described by motion information of joints, it will change as a consequence of the above attacks. The tampering methodology should be capable of verifying and tracing the pattern of such kind of attacks. Also, since motion data has different formats (global and local), the technique should generic enough to be applicable to both of them.

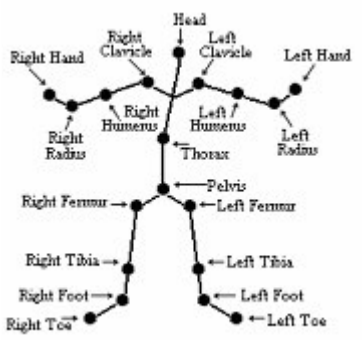


Figure 5.3. Human Body with 19 joints

Joint 1 Joint 2..... Joint 19

$x_{1,1}$	$y_{1,1}$	$z_{1,1}$	$x_{2,1}$	$y_{2,1}$	$z_{2,1}$	.....	$x_{19,1}$	$y_{19,1}$	$z_{19,1}$
⋮	⋮	⋮	⋮	⋮	⋮	.....	⋮	⋮	⋮
$x_{1,i}$	$y_{1,i}$	$z_{1,i}$	$x_{2,i}$	$y_{2,i}$	$z_{2,i}$	.....	$x_{19,i}$	$y_{19,i}$	$z_{19,i}$
⋮	⋮	⋮	⋮	⋮	⋮	.....	⋮	⋮	⋮
$x_{1,m}$	$y_{1,m}$	$z_{1,m}$	$x_{2,m}$	$y_{2,m}$	$z_{2,m}$	.....	$x_{19,m}$	$y_{19,m}$	$z_{19,m}$

Figure 5.4. Motion Data Representation with positional information

5.1.1 Issues with State of Art – Related Work

Tampering can be avoided by using tamper detection mechanism, such as fragile watermarking. Fragile watermarking for different kind of multimedia such as images, video, audio, 3-D models and 3-D motion Capture data [5.5, 5.7-5.11, 5.13-5.14, 5.16, 5.19, 5.21] can be achieved by embedding a watermark inside a target data. Watermarks are patterns that can be information such as digital signatures, names etc. For encoding purpose, we can convert them into a series of bits 0s and 1s. The fragile watermarking algorithms encode the bit information using a key. During the watermark extraction process, we extract the

watermark using the same key and try to compare the extracted watermark with the original watermark. Tampering is recognized whenever during the watermark extraction process, a sub-part of the embedded watermark is found corrupt. The sub-part points to the spatial location of corruption and serves as an evidence for tampering. Another way of detecting tampering is by taking a fingerprint using a hash function such as simple checksum, SHA (secure hash algorithm), message digest and comparing it with an original reference fingerprint, or parity bit checking to detect tampering. The problem with such approaches is that they tell us that attack occurred, but we cannot identify the nature of the attack and restore the data to its original value. In short, the above suggested methods help us detect the error but they do not help us reverse the attack. So, we need a technique that can identify attacks and also reverse them consecutively.

Error induced in the data can be corrected by forward error correction (FEC) [5.2, 5.4] that adds error correction information and uses it to remove errors. FEC has been used by networking protocols at different layers of the protocol stack. Generally, data is divided into packets or frames and sent across a lossy network. Information related to a packet or frame is added to a k^{th} packet of frame that follows it. In case of a loss, the data can be reconstructed to achieve a desirable quality. In such kind of framework, we increase the size of packets or frames to achieve reliability, but we cannot guarantee that the quality of the data to be same as the original. The problem with this approach is that it adds additional information to packets or frames, thus consuming more network bandwidth. Another method that can be applied is interleaving [5.4], where extra information is not required to do error correction. Data elements can be interleaved between different frames or packets. During a loss at the receiver side, the interleaving pattern is reversed and the loss is uniformly spread out, and the

data is reconstructed by using techniques such as interpolation. The negative side of interpolation is that the quality of the data can be compromised, since interpolation are approximate values. In order to apply forward error correction or interleaving, we have to take each joint information and apply the same. However, we will have to compromise the quality of motion data i.e. trajectory of a joint might change, in case of shuffling attacks. Also, we need to quantify the amount of information that needs to be generated during the application of forward error correction for each joint data. Given, m frames for a joint, we need to divide it into k blocks and then generate $k*p$ amount of information, where p ($< m/k$) is a constant. So, the amount of information we need to correct the data for n joints is $nc*k*p \sim O(nc*m) \sim O(n^2)$, which is similar to the order of matrix information. Therefore, we need a mechanism to quantify and minimize the information required to use forward error correction. Also, we would like to maintain the quality of the data when the error correction is done. Another problem that could potentially confuse forward error correction method is that when columns are shuffled, joint data is shuffled, for example – left hand joint information is exchanged for right hand joint information. Since forward error correction mechanism assumes that the columns are not shuffled, the information to correct errors can be used. However, this is not true in our case, and the forward error correction cannot be applied. For interleaving, we interleave joint data assuming it a continuous stream of information varying with time and in a given order, and assume that during the reverse process of interleaving at receiver the order of data elements has not changed. However, due to a row shuffling attack, this order changes and we cannot apply reverse interleaving for the same reason.

Recent advances in watermarking [5.1, 5.3, 5.15, 5.17] for error correction in images and videos hide error concealment as watermarks in the data set. These techniques do a frequency transform and use the medium range frequency components to hide the error concealment information. In the event of an error, where the quality of image of video drops, we can improve the same by retrieving the error concealment information and improving the quality of the data accordingly. It is difficult to apply these ideas to motion capture data since the representation is different. Since motion capture data consists of time varying information (x, y and z for human joints), we might like to apply a watermarking based error concealment scheme applicable to a time-series data. Audio data is time-series data and techniques such as [5.22, 5.23] use watermarking for error correction, wherein data divided into packets and sent over a lossy channel. The data packets hide information (represented as watermarks) about other data packets. In case, some packets are lost, the packets that are received successfully are use to rover the error induced due to lost packets based on the hidden information. These techniques can be applied to motion capture data by taking joint information (varying x, y and z) dividing them into groups (clusters) ordered on the basis of time. One group hides information about another group. Loss in one group information can be regained from another group using the watermarking embedded. Such techniques will fail in the event of shuffling attacks of elements between different groups. Therefore, direct application of these techniques is not possible, but as we develop our technique, we will see that watermarking does help in our methodology.

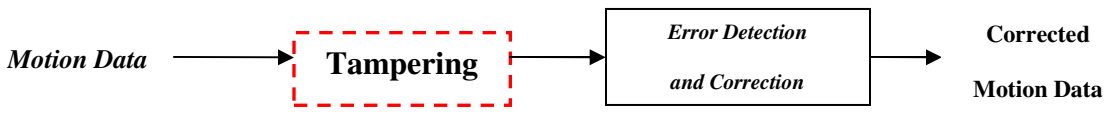


Figure 5.5. Illustration of problem statement

To summarize our problem statement, motion capture data can be subject to tampering due to shuffling and random attacks. The state of art of error detection and correction is not able to detect the attack patterns and reverse them and even given bounds on the possibility of solving such a problem. So, in order to detect errors and correct them in a motion data matrix (see Figure 5.5), we present a method that uses watermarking and multiple hash functions fingerprints to solve our problem.

5.1.2 Overview and Contributions

The dissertation mainly contributes by showing that fingerprinting and watermarking mechanisms can be combined to solve the problem related to detection and correction of errors in a motion capture data repository. The mechanisms proposed for fingerprinting and watermarking are simple to implement and give us theoretical bounds on the information required and the accuracy for error detection and correction. The basic idea is to extract reference patterns (fingerprints) identified by row and column hash functions which take input parameters as row and column elements of the motion data matrix. By observing change in the values related to row and column function, we can detect, predict the attack pattern and reverse it. The patterns (fingerprints) resulting from hash functions help in error detection by identifying the type of attack such as row shuffling, column shuffling, row element shuffling, column element shuffling and their combinations. Random attacks that change data element values are detected by change in watermarks embedded in data elements. Finger prints help in solving the attacks reversal such as column or row shuffling, whereas watermarking helps in reversing attacks such as column element or row element shuffling. The combination of these attacks can be corrected by combining watermarking and

fingerprinting. Random attacks can be identified with change in watermark based tags related to the impacted data element, and can be corrected approximately by using interpolation. Since we use information related to each row and column and use it to reverse the attack, we can categorize our methodology as forward error correction where we use O (n) space. The time complexity varies from o($n \log n$) to O(n!) depending on the attack being detected and reversed. Since the technique is applicable to a matrix it is applicable to different motion data formats (global or local) as well.

The methodology suggested in this dissertation derives its motivation from our previous approach, suggested in [5.20], wherein we took the singular value decomposition of the motion data matrix. The resulting first right and left singular vectors due to SVD are used as patterns (fingerprints) whose components are function values as suggested in this dissertation. The fundamental difference between the approaches is that our model is more generic as compared to the SVD approach. The generic nature of our method comes from the fact that the mathematical function can be implemented using different techniques like mean, standard deviation and by using the first and right singular vectors of SVD. In addition to pattern based comparison, we use watermarking to detect random attacks, column element shuffling and row element shuffling attacks. However, in case of [5.20], watermarking was never used and the system assumed such attacks as random attacks.

5.2 Scheme Design

The tamper detection methodology is applied on MoCap data (.csv – comma separated values format) acquired from Vicon IQ [5.24] (*120* frames/sec). As discussed earlier, this data can be expressed as a matrix *M*, where columns represent the joint positional information of the human skeleton, varying values per frame (row). The dissertation develops a tamper

detection technique to handle such attacks. The fundamental ideas behind the scheme are as follows:

5.2.1 Tamper Detection-Correction Methodology

The scheme depends on the idea of hashing functions and watermarking of data elements. We define two *hash functions* F_i related to each row *i,* and G_i related to column *i.* The inputs to function F_i are elements of the i^{th} row. Similarly, input to the function G_j is the elements of j^{th} column. These hash functions are mathematical formulas such as variance, checksum etc, and their output is a number. So, given a motion data matrix *M*, we have *m* values from function F_i ($1 \leq i \leq m$). We define a tuple from these values FT = < F_1, F_2, $F_{3...}$ F_m>, similarly, we have a tuple defined from the function *G* for *nc* columns GT = <G_1, G_2, $G_{3...}$ G_{nc}>. Since the tuple FT and GT uniquely identify the row and column information using hash functions, we can say that these are *fingerprints* of the motion data matrix. Watermarking of data elements is carried out by embedding tags identifiers (*id*) to uniquely identify data elements. The following Sub-sections explain the mechanics for using hash functions and watermarking based tag identifiers to identify attacks (row shuffling, column shuffling, row element shuffling, and column element shuffling, combinations and random attacks) and possibly reverse them.

1) Row or Column Shuffling:

During a row shuffling attack, the column elements do not change. This can be observed in Figure 5.6, where the rows 1 to 3 are shuffled and the column elements are unaltered. So, we can say that GT will be unaltered. However, tuple representing the row FT will change such that the order of occurrence of values of the function values F will be shuffled in the same order as we shuffle the rows 1 to 3. We can recognize that a row shuffling attack occurred by

observing that GT did not change while FT changed. The pattern of shuffling can be identified by ordering the values of FT' (the changed FT) and FT, and mapping the index of the ordered values for FT and FT'. Due to the attack, values remain the same, but the index changes. Hence, we are able to identify the pattern and map the rows back and restore the dataset. For our example, when the values are ordered, we can observe that indexes (1, 2, 3) of the ordered FT's are mapped to indexes (3, 1, 2) of FT. Therefore, by following this simple procedure, we are able to detect the attack and reverse it.

Similarly, we can identify a column shuffling attack, due to the symmetrical nature of the matrix. This can be observed by a change in GT to GT', wherein the function values in GT are shuffled in the same order.

An important property observed is that by shuffling rows, we do not change the elements of the columns, resulting in unchanged GT. We can generalize this property and term it as row tampering property.

Row Tampering Property: Mathematically, if A is modified to B using row tampering, then $A_{mxnc} \neq B_{mxnc}$ such that $\bigcup_{k=1}^{m} a(k, i) = \bigcup_{k=1}^{m} b(k, i)$, for all i $(1 \leq i \leq nc)$, and $\bigcup_{k=1}^{nc} a(j, k) \neq \bigcup_{k=1}^{nc} b(j, k)$, for j $(1 \leq j \leq m)$.

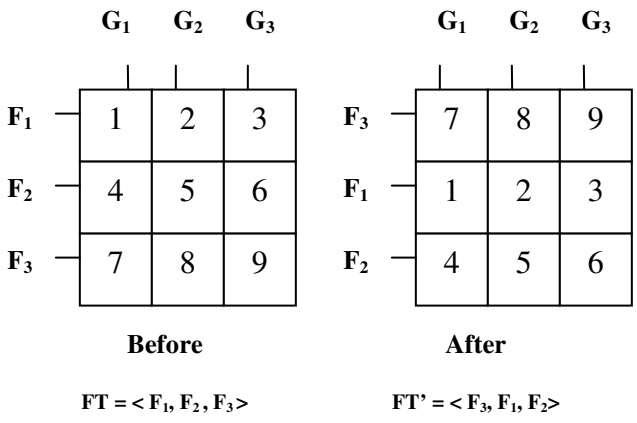


Figure 5.6. 3x3 Matrix attacked by shuffling rows 1, 2, and 3

A similar observation is case of columns where rows do not change i.e. FT does not change. So, we have a column tampering property to identify the consistency in FT.

Column Tampering Property: Mathematically, if A is modified to B using column tampering, then $A_{mxnc} \neq B_{mxnc}$ such that $\bigcup_{k=1}^{nc} a\ (i,\ k) = \bigcup_{k=1}^{nc} b\ (i,\ k)$, for all i $(1 \leq i \leq m)$, and $\bigcup_{k=1}^{m} a\ (k,\ j) \neq \bigcup_{k=1}^{m} b\ (k,\ j)$, for j $(1 \leq j \leq nc)$. So, we can say that the order of shuffling rows does not alter the order in which the columns are shuffled and vice-versa.

2) Row or Column Element Shuffling

During row element shuffling attack as shown in Figure 5.7, we tend to shuffle elements of a given row. As a result, we do not alter the row elements during row element shuffling. In this case, the column tampering property is valid, wherein the row elements do not change. Therefore, during row element shuffling, the tuple FT is conserved. However, the tuple GT changed to GT', since we get new values from the function G for each column.

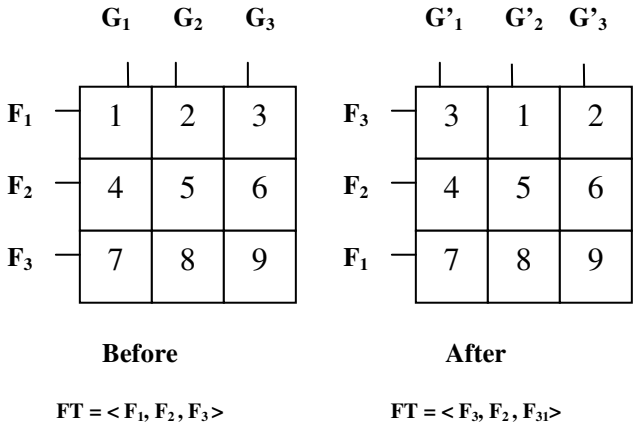


Figure 5.7. 3x3 Matrix attacked by shuffling elements of row 1

It can be observed that only the columns change, and the resulting values of G for each column is different. Therefore, it becomes difficult to trace the order of shuffling of the row

elements, by ordering the values of $G_i(s)$ in GT' and comparing them to GT. In order to find the correct order, we have to keep trying different permutations of ordering the row elements, in order to get the correct ordering in which the row elements were shuffled. In our case with 3 x3 matrix, we have a permutation of 3!.

The situation becomes more complex, when more than one row is shuffled. So, for every row, we can have 3! orders, and in this case we will have 3.3! different orders. A similar situation is observed for column element shuffling, wherein the GT remains the same due to row tampering property, since columns elements are unaltered and FT changes to FT'. In case of row element or column element attack, we need to find the correct order from *nc*! or *m*! different permutations. So, the time complexity for reversing such an attack is O(*nc*!). Therefore, once a row element or column elements shuffling attack is detected, we need to reverse the attack by using a brute force method that requires an exponential time algorithm.

We present a watermarking based method that can solve the problem for row element shuffling.

- *Element Tags ID(s) using Watermarking*: For simplicity, we first try to solve the problem for row element shuffling, since the number of elements is less, and later show the applicability for column element shuffling. Each element in a row is watermarked with a unique tag *id*. This tag *id* is a watermark that can be a number that is in the range 0 to 2^b -1, such that the *id* (watermark) can be represented using *b* bits. The technique we use to encode *b* bits in the element value is quantization index modulation [5.3], by modulating the value of the element based on a scale. As seen in Figure 5.8, the scale is divided into equal number of intervals, and each interval is labeled with bit equivalent of numbers in the range 0 to 2^b -1. The element value is measured on the scale and if it lies in the interval that represents the id

to be encoded, it is not modulated. Otherwise, we modulate the value such that it lies in an interval, representing the bit information. Since the perturbation in value might impact the semantics of motion data, the distortion induced should be minimized. As observed, modulations are proportional to interval size, and can be controlled by choosing intervals with small size. The choice of the scale can be the maximum size of an element value, and the interval size depends on number of intervals which can be fixed to minimize the distortion induced.

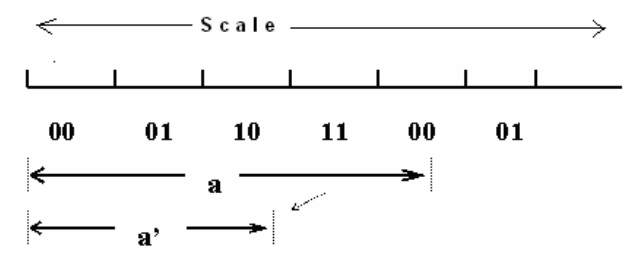


Figure 5.8. 2 bit encoding for the absolute value |a| of the data sample (value changes from a to a' due to modulation)

Once tagged with unique *id*(s), the order of elements in a given row can be determined on the basis of the unique *id*(s). So, in the event of element shuffling, we can find which elements have been impacted by decoding the *id*(s), and reversing the order, on the basis of the unique *id*(s). Since, for row element shuffling, we only need 57 tag *id*(s), we can easily identify the change in order of occurrence of elements and placing them in a correct predefined order. The problem of identifying the correct order is brought down from 57! to 57 only, which is a significant improvement. It is hard to solve the problem when we deal with column element shuffling attacks. The reason is the limited supply of unique *id*(s) that can be assigned to the column elements. This occurs due to the maximum number of bits that can be encoded in the element, since the scale cannot be divided into an interval which is infinitesimally small i.e.

equivalent to zero. Due to this limitation, we see that *id*(s) in columns reoccur during the encoding process. In case of an attack, where elements with same *id* are shuffled, such an attack could lead to a similar situation, wherein we had to determine the correct arrangement of elements with same *id*(s). Also, in case elements with same *id* are shuffled with elements with different *id*(s), it is still hard to find the correct index of the elements with unique *id*(s), since we have to choose from a set of indices where elements with same tag *id*(s) are stored. For example, if the elements with same tag id are stored at 2^{nd} and 5^{th} position, but it is changed to 3^{rd} and 6^{th} position. It is hard to find, which element should go back to 2^{nd} or the 5^{th} position, since their *id*(s) are the same. Mathematically, for m elements in a column, and given maximum b bits to encode, every tag *id* reoccurs $(m/2^b)$ times. This problem can be solved by checking all the permutations in $(m/2^b)!$ trials. Therefore, we see that using exponential brute force methods problem reoccurs due to limitation of the number of bits that can be watermarked. However, we do bring down the calculations involved in the process, from $O(m!)$ to $O(m/2^b!)$, but it does not change the nature of time complexity. In other words, the tighter bound is $m/2^b!$, but the worst case behavior is $O(n!)$. We will take up this problem as a future initiative.

An important aspect to note here is that the number of elements in a row are 57, whereas every column has m elements which can be larger than 57. Due to the limitation in the tag *id*(s) as suggested above, it is easier to tag the row elements uniquely as compared to column elements. Therefore, our policy is to choose tags for row elements first. Since these tags are shared with column elements, an elements is a member of row and column, we suggest an idea to uniquely tag column elements and minimize on reoccurring tags.

Since an element can be a member of a row and column as well, we need to share the same *id*

in both row and column. In case, every row has the same order of *id*(s), then every column element has the same *id*. For example, every first row element is an element of the first column, and in case, it is labeled with the same unique *id*, all the column elements carry the same *id*. Thus, making it impossible to distinguish the elements based on the *id*(s) encoded in them. In order to make sure that *id*(s) are not the same, we make sure that at least 2^b elements carry unique *id*(s). This is possible by choosing 2^b unique order of *id*(s) for 2^b contiguous rows. For example, for 2^2 (4) tags, we can encode 4 row elements with unique order of tag *id*(s) shown in Figure 5.9. We can repeat the pattern. As observed the row elements have unique *id*(s), meanwhile column elements see a repetition in tag *id*(s).

1	2	3
1	2	3
1	2	3

(a) Tag *id(s)* not unique for column elements

1	2	3
3	1	2
2	3	1

(b) Tag *id(s)* unique for column elements

Figure 5.9. Uniqueness in column element tags due to ordering Scheme

Watermarking based tag *id* mechanism can be applied to reverse attacks such as row element shuffling. However, applying tag *id* to solve the problem of row or column shuffling will pose problems since during row shuffling attacks it is not possible to distinguish between rows that are swapped (shuffled) and have the same tag ids per row element. Therefore, to be safe, fingerprints can help us solve the problem of row-column shuffling.

3) Combination of Attacks

The following discussion covers the attack that occurs as combinations of shuffling attacks:

In the event of combination of row-column shuffling, we apply the detection mechanism for

FT and GT. For example in case of Figure 5.10, we have combination of the row and column attacks and the steps taken in the previous examples to detect shuffling can be used to identify row and column shuffling and reverse it. Therefore, we can say that different shuffling row and column pattern of attacks can occur, but since row and column elements do not change during these attacks the function values obtained do not change. So, we need to detect the change in order of these values in GT and FT, by ordering GT' and FT and mapping the indexes.

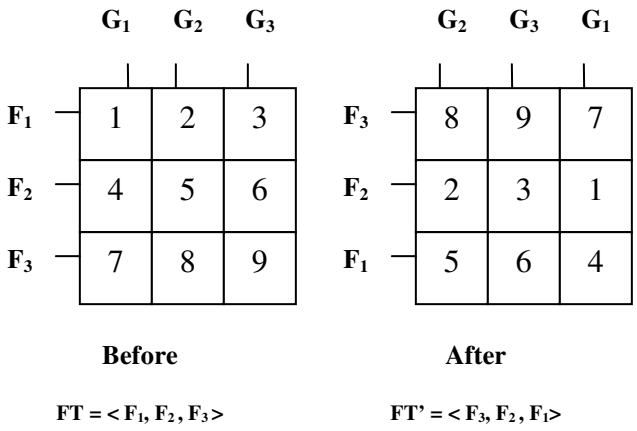


Figure 5.10. 3x3 Matrix attacked by shuffling rows 1, 2, 3 and columns 1, 2, 3

During row shuffling, we see a change in FT', and it can be ordered and made equivalent to FT. Therefore, this will act as an indication for row shuffling attack. Since, during a row element shuffling, GT will not be equal to GT' and also the ordered elements of GT'. So, we can concur that there is a combination of row and row element shuffling attack and use the corresponding attack reversal process. Based on the above idea, by symmetry we can identify column element shuffling and column shuffling attacks. For such a case, ordered GT' will be similar to GT, and FT and FT' will not be same even when FT' is ordered.

In the event of row element and column shuffling, we end up impacting only FT, since GT

remains same. Since during row element shuffling FT will not be equal to ordered FT' or FT as such, we first to make sure that the row element shuffling is reversed. Once this is achieved, column shuffling can be removed. Similarly**,** by symmetry, we can identify row shuffling and column element shuffling attacks and remove them. In case of combination of row-column element shuffling attack, we can identify it as a random attack as discussed below:

3) Random Attack

The idea to use FT and GT to detect random will fail, since we perturb values of data elements such that all the rows and column function values are changed, resulting in ambiguity, as observed in Figure 5.11, which results in change in function values for F and G. So, a simple method to overcome this attack is to check for tag *id*(s), rather than checking for function values. Since the tag *id*(s) are watermarks, they are lost due to noise addition attacks.

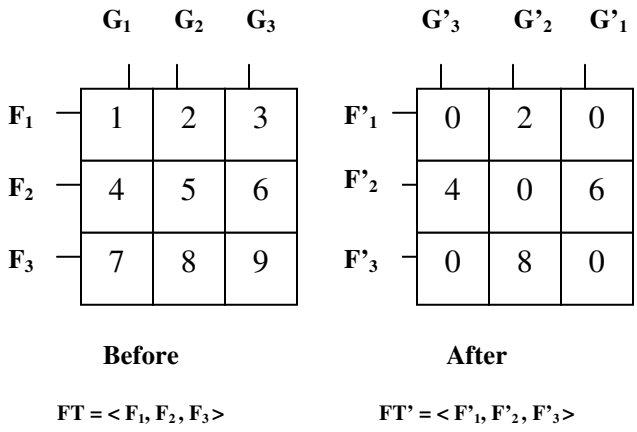


Figure 5.11. 3x3 Matrix attacked randomly resulting change in all function values

Since the encoding of watermarks uses small interval size to control distortion, it makes the scheme vulnerable to loss of bit information i.e. a small perturbation in value shifts the interval in which the value lies (see Figure 5.9). In cases, where the changed tag *id* is similar

to a neighboring elements tag *id*, we can distinguish by comparing with a predetermined tag *id* encoding pattern. It can also happen that noise added does not change the semantics of motion, but we see change in tag *id*(s). However, it is difficult to design a mechanism that exactly tells when an action in motion changes due to noise addition i.e. the metric should validate change in motion according to human perception. Finding such a metric is out of the scope of this dissertation, and we limit to finding the point where it would indicate that the random attack occurred.

To summarize, random attacks are preferably detected using tag *id*(s). In cases, where tags *id*(s) are lost but the motion semantics did not change due to change in element value, we can detect such an event by benchmarking our system. During random attacks, we do not know by how much the value changed, such a change is impossible to reverse. But we can use interpolation mechanism, wherein for elements that have been modified are not used and rather interpolated from neighboring elements. This would see result in the degradation of quality, but helps in filling out gaps created due to random attack.

5.2.2 Synopsis of Technique

The methodology requires patterns (fingerprints) and watermarked data elements in order to carry out error detection and correction procedure. So, whenever we add a motion data file to the repository, we tag the elements and extracts patterns (fingerprints) afterward, as seen in Figure 5.12. Tag addition is done before pattern extraction as watermarks add noise, and the patterns might change, if extracted before. So, we first add tag *id*, and once we are sure that the data elements will not be perturbed further, we extract the patterns. The error detection and correction process is shown in Figure 5.13, where we extract the tag *id*(s) and patterns from the tampered motion data matrix and subject it to our algorithm.

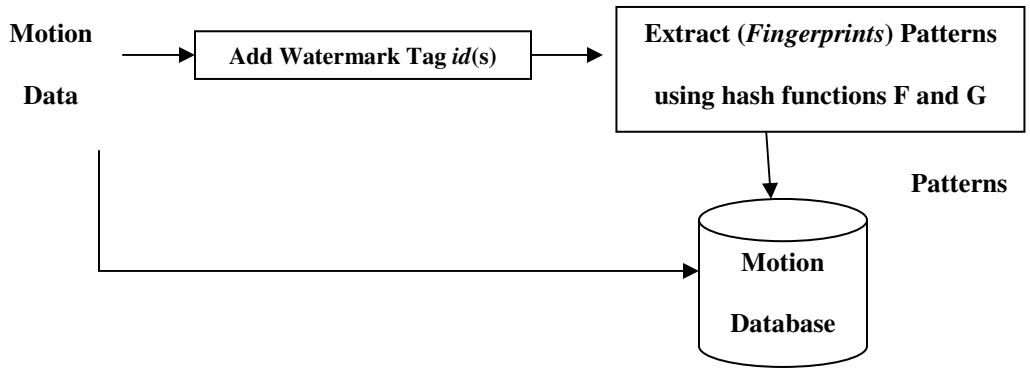


Figure 5.12. Pattern extraction and Watermark Tag *id* addition process

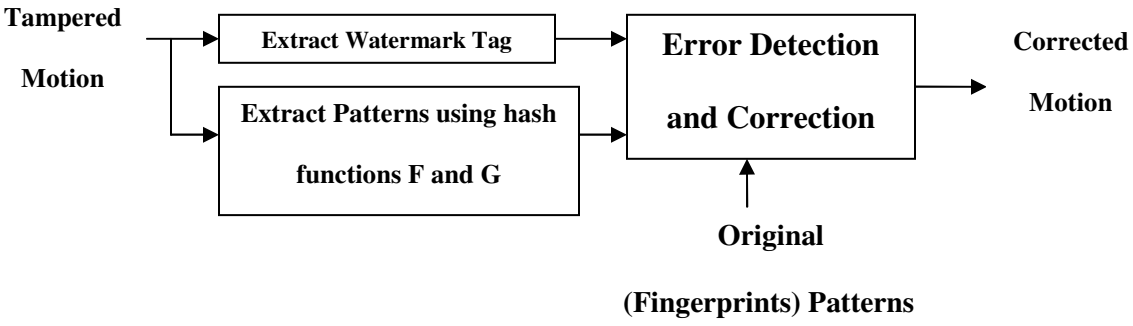


Figure 5.13. Error Detection and Correction Process

Given (fingerprints or patterns) FT and GT from original matrix, and FT' and GT' patterns from the tampered matrix, the steps required for tamper detection and attack inversion are given in Algorithm as given below.

Algorithm for Error Detection and Correction

1: ***If*** (FT = FT' AND GT = GT')

2: No attack

3: ***Else if*** (FT = FT' AND GT != GT')

4: *If*(Ordered(GT') = GT)

5: *Column Shuffling Attack*

6: Reverse attack by comparing indexes of GT and Ordered(GT')

7: *Else*

8: *Row element Shuffling Attack*

9: Reverse attack by using watermark tag id(s)

10: *End*

11: ***Else if***(GT = GT' AND FT != FT')

12: *If*(Ordered(FT') = FT)

13: *Row Shuffling Attack*

14: *Reverse attack by comparing indexes of* FT *and Ordered(*FT'*)*

16: *Else*

17: *Column element Shuffling Attack*

18: *Reverse attack by finding a correct permutation after applying watermark tag id(s)*

19: *End*

20: ***Else if*** (Ordered(FT') = FT AND Ordered(GT') = GT)

21: *Combination of row and column shuffling*

22: Reverse attack by *comparing indexes of* GT *with Ordered(*GT'*).*
and FT *with Ordered(*FT'*)*

23: ***Else if*** (Ordered (FT') = FT AND Ordered (GT') != GT)

24: *Combination of row shuffling and row element shuffling attack*

25: Reverse row shuffling attack by comparing indexes of FT *with Ordered (*FT'*)*

26: *Reverse row element shuffling by finding a correct permutation after applying watermark tag id(s)*

27: ***Else if*** (Ordered (GT') = GT AND Ordered (FT')! = FT)

28: *Combination of column shuffling and column element shuffling attack*

29: Reverse column shuffling attack by comparing indexes of GT *with Ordered (*GT'*)*

30: *Reverse row element shuffling by finding a correct permutation after applying watermark tag id(s)*

31: ***Else if*** (GT' = GT AND Ordered (FT')! = FT AND FT' ! = FT)

32: *Combination of row shuffling and column element shuffling attack*

33: Reverse column element shuffling attack *by finding a correct permutation after applying watermark tag id(s), followed by reversal of row shuffling attack*

34: ***Else if*** (FT' = FT AND Ordered (GT')! = GT AND GT' ! = GT)

35: *Combination of column shuffling and row element shuffling attack*

36: Reverse row element shuffling attack *by finding a correct permutation after applying watermark tag id(s), followed by reversal of column shuffling attack*

37: ***Else***

38: *Random Attack*

39: *Trace elements by finding change in tag id(s)*

40: ***End***

5.3 Analysis

The technique proposed in this dissertation uses combination of watermarking and pattern (fingerprints) based error detection and correction. This is possible since watermarking based tag *id*(s) cannot solve the problem related to error detection in case of shuffling attacks. However, they help in detecting random attacks. During error correction, shuffling attacks such as row, column, row-column can be resolved using patterns only. However, watermark tags *id*(s) help solve the problem related to element shuffling, which patterns cannot. Therefore, the techniques compliment each other.

The scheme is generic, since the hash function F and G can be implemented using techniques

such as checksum, cyclic redundancy check (CRC), SHA, mean, standard deviation, or variance. Therefore, techniques such as CRC that could not point out an attack and remove it can now be used to do the same with the proposed generic model. The choice of hash function is to make sure that the output patterns (fingerprints) are unique values for columns and rows with different element. Since we use side information (fingerprints) to correct errors, we can say that we have developed a signature based scheme that can do forward error correction. Due to applicability of the technique to a matrix it can be applied to different formats of motion data (global and local). We now evaluate our method based on complexity and compare it with our previous methodology [5.20].

1) Space and Time Complexity

Since we require the original tuple for comparison and attack inversion, we need to store the tuples F and G. So, we need O (m+*57*) ~ O (n) of storage with every matrix of size (m x *57*). The analysis for time complexity can be done in terms of tamper detection and correction.

During the watermarking process, we need O (57*m) (~ O(n)) steps, which are equal to the number of element tag *id*(s). Since we have to extract patterns, we need O(m + *57*) (~ O(n)) steps for extracting the tuples F and G. Therefore, during the process of adding motion capture file to our repository, we would require O(n) steps. In order to detect errors and correct them, we first extract watermarks based *id*(s) using O(n) steps, and then extract function (reference pattern) values using O (n) steps. To detect a shuffling attack, we need to sort the data and this will take O(n *log* n), if we used heap sort. Using the idea of watermark based tag *id(s)*, we can solve the problem of row element shuffling in linear time O(n), since we can realign the element by ordering the elements again based on a predefined order. However, the column element shuffling attacks as observed is still exponential. The

maximum computation involved is for column element shuffling attack reversal, and it is O (n!). Therefore, for tamper detection and correction due to shuffling, we can say that our algorithm requires minimum o(n log n + n) ~ o(n log n) and maximum O (n *log* n + n!) ~ O (n!) time. In case of random attacks, we need to identify the attacked data elements and use interpolation to correct them and this can be done in O(57*m) ~ O(n). Since, we try to determine shuffling attack before we decide on a random attack, we can say that the time complexity requires at least O(n *log* n) for tamper detection, and can be corrected in o(n) to O(n!). To summarize, the complete process of tamper detection and correction for all attacks will take o(n *log* n) to O(n!).

***2) Comparison with* [5.20]**

As already shown in Sub-section 5.1.1, the state of art related to fingerprinting, watermarking, forward error correction, and interleaving cannot solve the problem related to error detection and correction in motion capture data. Therefore, we suggested a new method that combines fingerprinting and watermarking to solve the problem. This method draws its inspiration from our previous method [5.20] based on Singular value decomposition (SVD) [5.12], and this Sub-section does a comparison analysis with the same.

In our previous approach [5.20], we use first left and right Eigen vectors of the singular value decomposition of the motion data matrix, in order to derive FT and GT respectively. So, our approach generalizes our previous approach, and we can implement the function using different mathematical formulas as explained above. The technique was capable of detecting and reversing row shuffling, column shuffling and row-column shuffling attacks. However, [5.20] does not solve the problem of reversing row element shuffling and column element shuffling attacks, since it assumed these attacks as random attacks and treated them

accordingly. Random attacks were solved using patterns only, which as observed in our case fails, and must be solved using watermarking based tags. In our case, we solve the problem of reversing row element shuffling attacks using watermark tags in O (n) and column element shuffling in O ($n!$). Random attacks can be detected and fixed using interpolation, and this guarantees better robustness and accuracy than [5.20].

We can use SVD to detect and correct errors by using right and left singular Eigen vectors. However, it can be observed that the time complexity for SVD is O (n^3), as compared to directly calculating F and G using functions such as mean or variance in O (n) time. Based on this observation, we prefer not to use SVD for our methodology.

3) *Extensibility*

The ideas suggested here are applicable for detecting and correcting tampering in data that is represented as matrix. For example, if we take relational databases, where a table is kept is equivalent to a matrix, column swapping attacks can change information as every data element should occur in its column e.g. a student record with grades in two courses are swapped. Row swapping will not occur in this case, since order of rows is of not importance for the correctness of the table. Of course, the table can be subject to row or column element swapping, and it would be easy to detect the attack, but impossible to reverse. The impossibility comes from the fact we cannot watermark data with absolute values like name composed of alphabets. So, for such a case, we can only say that the database was corrupted. Random attack identification is again not possible since we cannot watermark always. To summarize, our method can be extended to databases for detecting and reversing column shuffling. However, since we cannot watermark every data element, detection of row and column element is possible, but the correction is not possible.

5.4 Performance Analysis

All the experiments are carried out 100 different motion data files, and tampering was visualized with a java tool written with SUN's OPENGL and it can visualize motion data in (.csv – comma separated value). This data was obtained in a *.csv* format created by Vicon IQ [5.24] Mocap system (Motion Capture facility at University of Texas at Dallas). The data (represented by 32 bit floating point) consists of joint information of a skeleton, expressed in terms of translational (co-ordinates) data. The following sub-sections give analysis on the patterns used for identifying attacks. Later, we find the probability of failure in order to measure the accuracy against these attacks. In order to analyze the distortion, we used signal to noise ratio defined by equation (5.1).

$$SNR\,(M, M') = 20\,Log_{10}\,\frac{RMS(M)}{RMS(M - M')} \quad M \sim \textit{original data matrix, } M' \sim \textit{encode data matrix,}$$

SNR ~ signal to ratio, RMS – Root Mean Square (5.1)

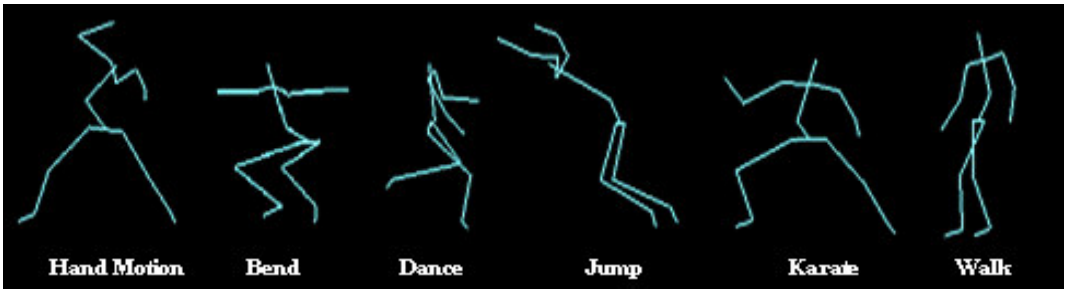


Figure 5.14. Snap-shots of some motion used for experiments

5.4.1 Distortion Analysis

During the tag *id* encoding process, we used 10 bit *id*(s) and with scale value that is larger than all elements. We used number of intervals 2^{10}, and observed that the signal to noise ratio varied from 95 to 120 dB. Such high signal to noise ratios indicated that the motions were similar after encoding and the tag system could be used for reverse element shuffling attacks.

We used a visualizer to validate the similarity in motion and it was found that addition of fragile watermark does not alter the motion. The analysis of attack pattern is done as shown in the next Sub-section.

5.4.2 Analysis of Attack Patterns

In order to analyze our approach for using row and column based hash functions, we do attacks such as row shuffling, column shuffling and row-column shuffling. Figure 5.15 gives visual examples of tampering done. We can see in Figure 5.16 that column mean values change, but the mean values remain unaltered.

We give examples wherein the mathematical function F and G are implemented as mean of row or column elements respectively. Row tampering attack only (row shuffling or column element shuffling) is observed in Figure 5.17, where we see that mean column values do not change, but the error is induced due to change in mean row values. Similarly for column tampering attack only (column shuffling or row element shuffling).

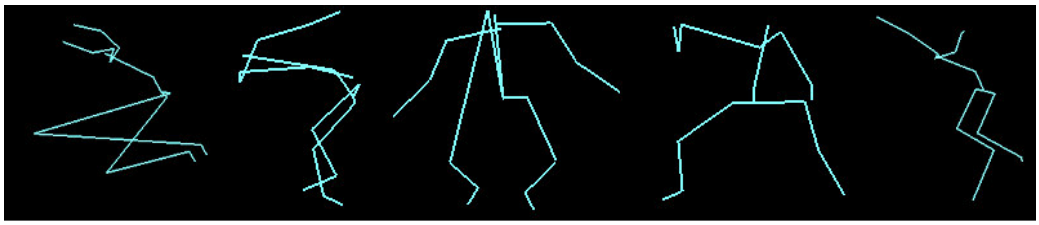


Figure 5.15. Snap-shot: Tampering of Actions

The row and column attack is observed in Figure 5.18, where error is induced in row and column mean values and is used as an evidence for an attack. In all the above cases, we can reverse the attack if these are cases of shuffling attacks. However, if the values have changed, inversion is not possible as we do not have an idea by how much the value was perturbed. In such a case, we have to use watermark tag *id*(s) to identify change in data elements. The next sub-section lists the accuracy of our method using different mathematical functions.

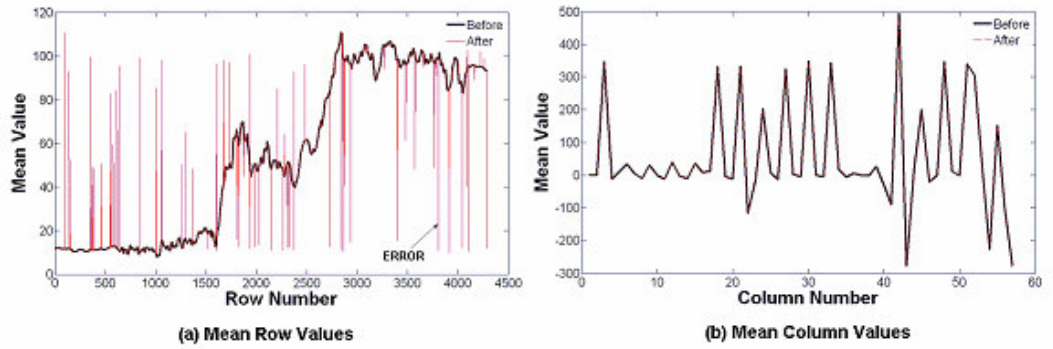


Figure 5.16. Row tampering Attack observed using mean row and column values

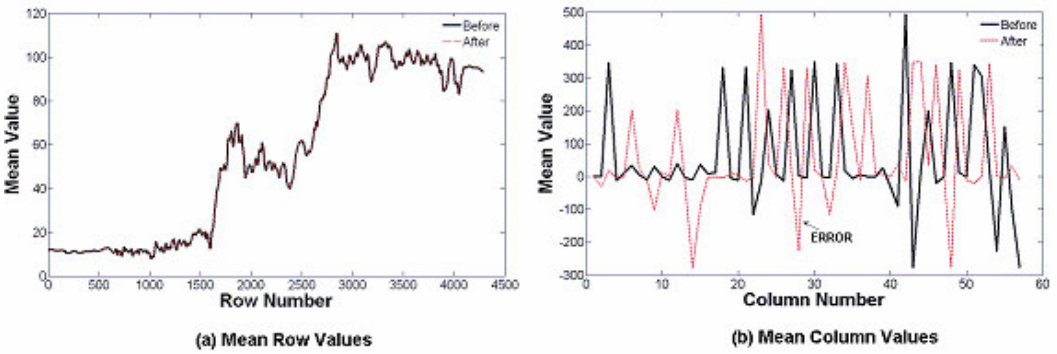


Figure 5.17. Column tampering Attack observed using mean row and column values

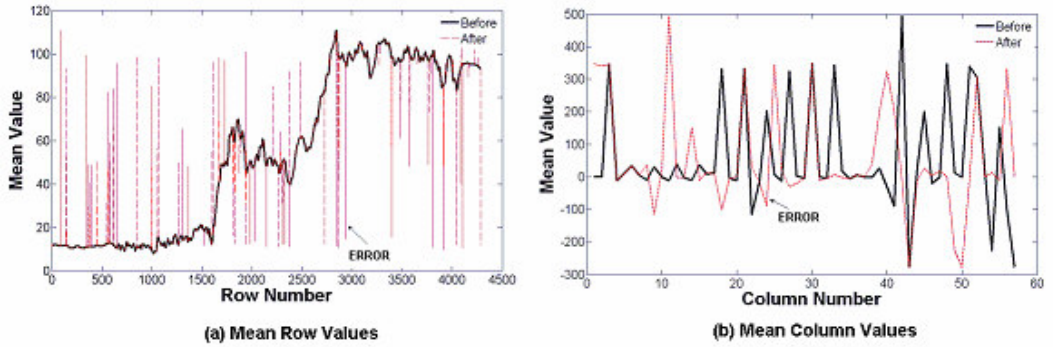


Figure 5.18. Row-Column Attack observed using mean row and column values

5.4.3 Accuracy of Hash Functions

We implemented the hash functions G and F using different mathematical formulas such as standard deviation, and mean. The accuracy of detecting and reversing row shuffling, column shuffling and row-column shuffling attacks is 99.99%. Row element shuffling attacks were detected with 99.99% accuracy, and the tag *id*(s) were successfully decoded, and the attack

was reversed. In case of column shuffling element attack, it is identified in with 99.99% accuracy where elements with same *id*(s) are not shuffled with each other. However, for cases where same *id* elements are shuffled, we had to resort to brute force method to reverse the attack. In case of random attack, we were able to identify the attack with 99.99% accuracy, and add the lost element by interpolation. The high accuracy of detecting shuffling attack and reversing specific row, column and row-column attacks implies that the hash functions results in unique values (fingerprints or patterns) for different row and columns. Therefore, these functions are well suited for implementing tamper detection techniques for motion data. For random attacks, we observed that the idea of finding out the element that has been attacked fails if we use comparison of function values. This occurred for cases, where we randomly changed the matrix elements values and observed the accuracy to be 50% on the average. To overcome this problem, we used change in tag *id* as indicator for random attack detection. It was observed that we could successfully observe change in tag *id* information, giving us an accuracy of 99.99 %.

5.5 Summary

The dissertation proposed a method that uses pattern (fingerprinting) and watermarking to detect and correct errors induced in motion capture data represented as a matrix. The patterns are created using hash functions that have their inputs as row and column elements of the motion data matrix. The elements are watermarked with unique tags. The fingerprints can detect and categorize shuffling attacks – row, column, row-column, row element and column element and their combinations. Finger prints help in solving the attacks reversal such as column shuffling and element shuffling, whereas watermarking helps in reversing attacks such as column element or row element shuffling. Random attacks that change value of

elements are detected using the change observed in the tags related to a tampered data element. Once detected they can be removed by substituting the element with an interpolated data element. Experiments show that extracted patterns are able to detect the type of attacks with 99.99 % accuracy. Also, once detected attacks can be reversed on the basis of their type. Analysis shows that the proposed method uses O(n) space to detect and correct errors, and the time complexity of detection and correction process varies from o($n \log n$) to O(n!). It can be concluded from the dissertation that combination of pattern and watermarking based error detection and error correction are better than methods that use only watermarking, forward error correction and interleaving techniques. Since the technique is applicable to a matrix it is applicable to different motion data formats (global or local) as well.

BIBLIOGRAPHY

[5.1] A. Yilmaz and A. A. Alatan, “Error concealment of video sequences by data hiding,” in Proc. Int. Conf. Image Processing, vol. 3, Sept. 2003, pp. 679-682.

[5.2] Clark, George C., Jr., and J. Bibb Cain. Error-Correction Coding for Digital Communications. New York: Plenum Press, 1981. ISBN 0-306-40615-2.

[5.3] C.-S. Lu, “Wireless multimedia error resilience via a data hiding technique,” in Proc. Int. Workshop Multimedia Signal Processing Virgin Islands, Dec. 2002, pp. 316-319.

[5.4] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," IEEE Network Magazine, Sept./Oct. 1999

[5.5] C-P Wu, C.-C. Jay Kuo, Fragile Speech Watermarking for Content Integrity Verification (2002), citeseer.ist.psu.edu/wu02fragile.html

[5.6] Chen and G. W. Wornell, “Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. on Information Theory*, vol 47, May 2001, pp. 1423-1443

[5.7] C.S. Collberg, and C. Thomborson,"Watermarking, Tamper-Proofing, and Obfuscation—Tools for Software Protection," IEEE Trans. Software Eng., Aug. 2002, pp. 735-746.

[5.8] C.Kailasanathan, R.Safavi Naini, and P.Ogunbona, "Fragile Watermark on Critical Points", International Workshop on Digital Watermarking Seoul, Korea, (Springer-Verlag), November 21-23, 2002

[5.9] D. Kundur and D. Hatzinakos, "Digital watermarking for telltale tamper proofing and authentication", Proceedings of the IEEE, vol. 87, no. 7, pp. 1167--1180, July 1999

[5.10] E.T. Lin and E.J. Delp, “A review of fragile image watermarks”, in Proc. of ACM Multimedia & Security Workshop, Orlando, 1999, pp. 25—29

[5.11] F. Cayre, O. Devillers, F. Schmitt and H. Maître, “Watermarking 3-D Triangle Meshes for Authentication and Integrity”, INRIA Research Report RR-5223, Jun. 2004

[5.12] H. Golub and C.F.V.Loan. Matrix Computations. The Johns Hopkins University Press, Baltimore, Maryland, 1996

[5.13] H.T. Wu and Y.M. Cheung, “A Fragile Watermarking Approach to 3-D Meshes Authentication”, Proceedings of the 7th Workshop on Multimedia & Security (ACM'05), pp. 117-123, 2005.

[5.14] I. Cox, M. Miller, J. Bloom, “Digital Watermarking: Principles & Practice” (The Morgan Kaufmann Series in Multimedia and Information Systems)

[5.14] J. Lee and C. S. Won, “A watermarking sequence using parities of error control coding for image authentification and correction,” IEEE Trans. Consum. Electron. vol. 46, no. 2, pp. 313-317, May 2000.

[5.15] J. Fridrich, M. Goljan, and A. C. Baldoza, "New fragile authentication watermark for images," in Proc. IEEE Int. Conf. Image Processing, Vancouver, BC, Canada, Sept. 10--13, 2000.

[5.16] K. Munadi, M. Kurosaki, and H. Kiya, “Error concealment using digital watermarking technique for interframe video coding,” in Proc. Int. Tech. Conf. Circuits/Systems, Computers, and Communications, Jul. 2002, pp. 599-602.

[5.17] L. L. Peterson, B. S. Davie, Computer Networks: A Systems Approach, 3rd Edition (The Morgan Kaufmann Series in Networking)

[5.18] M. M. Yeung and B.-L. Yeo, "Fragile watermarking of three dimensional objects," Proc. 1998 Int'l Conf. Image Processing, ICIP98, volume 2, pp. 442--446. IEEE Computer Society, 1998.

[5.19] P. Agarwal, K.Adi, B. Prabhakaran, “SVD-Based Tamper Proofing Of Multi-Attribute Motion Data” , To appear in Proc. of The 12th International conference on Distributed Multimedia Systems (DMS) ’06, pp-46-52

[5.20] P. Agarwal, and B.Prabhakaran, "Tamper Proofing of 3-D Motion Data Streams", Proc.of Multimedia Modeling Conference, Jan 2007, pp – 731 740

[5.21] S., Cheng, H. Yu, and Z. Xiong "Error concealment of MPEG-2 AAC audio using modulo watermarks," Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on Publication Date: 2002 Volume: 2, On page(s): II-261- II-264 vol.2.

[5.22] S Cheng,; Yu, H. H.; Xiong, Z; Method and apparatus for audio error concealment using data hiding, United States Patent, No. US 7047187B2, May 2006.

[5.23] Vicon, http://www.vicon.com

CHAPTER 6

AUTHENTICATION OF 3-D MESHES

Puneet Maheshwari, Parag Agarwal and Balakrishnan Prabhakaran

Computer Science Department, EC 31

The University of Texas at Dallas

800 WEST CAMPELL RD.

Richardson, Texas 75080-3021

6.1 Introduction

Digital Watermarking has found applications in authenticating 3-D models by using the loss of the watermark as an evidence for tampering of the 3-D shape of the data. There are many ways of representing 3-D models but the most commonly used method is triangulated meshes. 3-D meshes can be represented using geometric information defined by a set of 3-D vertices (points), and the topological information defined as the connectivity (edge) between vertices. 3-D meshes can be compressed by using algorithms such Compressed Progressive Meshes (CPM) [6.9], Progressive Forest Split (PFS) compression [6.11], and Progressive Mesh (PM) [6.7], wherein a 3-D model can be represented by different levels. The first level being the base mesh and further data is added to achieve further refinements till we get the original 3-D model back. Since authentication based watermarks need to be present in every refinement starting from the base mesh, it becomes a necessary criteria to achieve such presence. The state of art in 3-D authentication does not cater to such a criteria, and it is imperative that when a 3-D model is subjected to compression [6.8], and watermarks embedded in the geometric or topological information of the 3-D mesh will be lost without actual tampering taking place. This will contradict the idea of tamper detection. In addition, since other user level operations such as uniform affine transforms (rotation, scaling and translation) can be done, making the watermarks robust against the same is also a necessary criterion.

The purpose of this dissertation is to suggest a methodology that identifies the 3-D points (vertices) that are present in every refinement resulting from the CPM [6.9], PM [6.7] and PFS [6.11] algorithms. Since these points (vertices) define the shape of the object, it is also necessary that during watermarking we do not lose the shape of the 3-D object due to the

encoding. Therefore, reduction in distortion achieving imperceptibility of the watermark is also necessary.

6.1.1 Related Work and Issues

Techniques for other kinds of data [6.3] such as images, video and audio is not directly applicable for 3-D models, since the representation of 3-D models is in higher dimensions. The 3-D model watermarking must be robust against uniform affine transformations and compression mechanisms. These operations are user level operations, and we should not loose the watermarks in such cases. Some of the techniques for authenticating 3-D models [6.5, 6.13, and 6.14] are fragile in nature i.e. identify the smallest change to the shape of the model. Other techniques are robust (Semi-fragile) against uniform affine transforms, but not robust against compression mechanism i.e. the work done on authenticating 3-D models does not identify vertices (points) that will stay invariant to the refinements introduced by CPM [6.9], PM [6.7] and PFS [6.11]. Therefore, we have to design a semi-fragile [6.11] methodology that makes the technique robust against progressive compression mechanisms.

6.2 Scheme Design

The idea behind our algorithm is to select those set of vertices which are never used for compression. Before selecting this set of vertices, we will briefly describe the generic nature of some popular mesh compression algorithms such as CPM, PM and PFS. Various approaches have been suggested to send 3-D models over a network. Most commonly used methods are non-progressive approach, fine grain approach and group refinements. In [6.8, 6.10], group refinement approach is being used. In PM (Progressive Meshes) [6.7], a crude model is transmitted first and then refinement is made progressively through a series of vertex splits, which is inverse of the edge collapse operations (As shown in Figure 6.1).

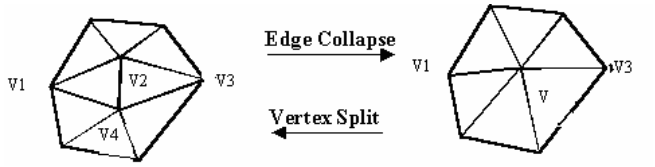


Figure 6.1. Vertex Split and Edge Collapse

The CPM (Compressed Progressive Meshes) [6.9] approach eliminates the overhead of downloading full resolution model which is required in the progressive refinement technique. It uses a technique, which refines the topology of the mesh in batches, which each increase the number of vertices by up to 50 percent. Less than an amortized total of 4 bits per triangle encode where and how the topological refinements should be applied. It estimates the position of new vertices from the positions of their topological neighbors in the less refined mesh using a new estimator that leads to representations of vertex coordinates that are 50 % more compact than previously reported progressive geometry compression techniques.

Progressive mesh (PM) [6.7] representation introduces a scheme for storing and transmitting arbitrary triangle meshes. This efficient, lossless, continuous-resolution representation addresses several practical problems in graphics: smooth geomorphing of level-of-detail approximations, progressive transmission, mesh compression, and selective refinement. In addition, it presents a new mesh simplification procedure for constructing a PM representation from an arbitrary mesh. The goal of this optimization procedure is to preserve not just the geometry of the original mesh, but more importantly its overall appearance as defined by its discrete and scalar appearance attributes such as material identifiers, color values, normals, and texture coordinates.

Progressive Forest Split (PFS) [6.11] representation is an adaptive refinement scheme for storing and transmitting manifold triangular meshes in progressive and highly compressed

form. The PFS format shares with PM and other refinement schemes the ability to smoothly interpolate between consecutive levels of detail. However, it achieves much higher compression ratios than PM by using a more complex refinement operation which can, at the expense of reduced granularity, be encoded more efficiently. A forest split operation doubling the number n of triangles of a mesh requires a maximum of approximately $(3.5.n)$ bits to represent the connectivity changes, as opposed to approximately $(5 + log_2 n\ )n$ bits in PM. It describes algorithms to efficiently encode and decode the PFS format. It also shows how any surface simplification algorithm based on edge collapses can be modified to convert single resolution triangular meshes to the PFS format. The modifications are simple and only require two additional topological tests on each candidate edge collapse.

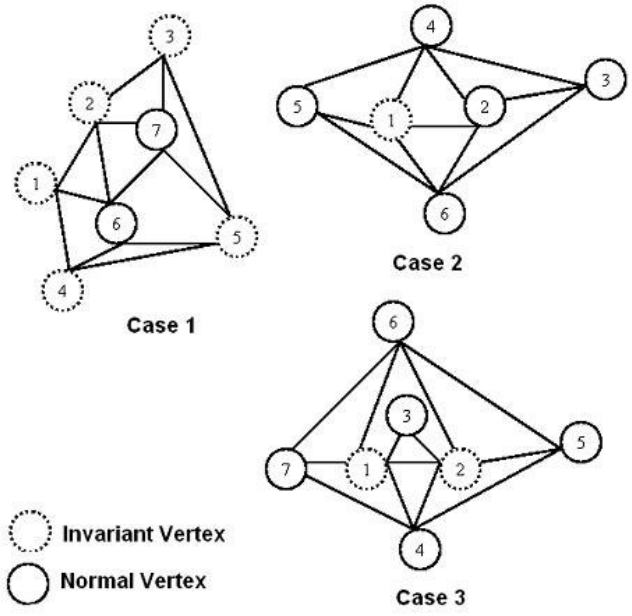


Figure 6.2. Cases for choosing Invariant Vertex Set

In all these compression techniques [6.7, 6.9, 6.11], the basic operations used for compression and decompression are edge collapse and vertex split. However, for collapsing of the edges, there are certain restrictions which must be followed.

1. At most two vertices may be collapsed into one. As shown in the case 2 of Figure 6.2 either edge (v_1, v_2) or edge (v_2, v_3) can be collapsed.

2. As show in the case 3 of Figure 6.2, for each edge e = (v_1, v_2) that will be collapsed and any vertex v_6 that is connected by an edge to both v_1, and v_2, the triple (v_1, v_2, v_6) must define a valid triangle.

6.2.1 Invariant Vertex Identification

Since all edges cannot be collapsed by the above compression algorithms, there remain a certain number of vertices corresponding to these edges which are never used by the compression algorithms CPM [6.9], PM [6.7] and PFS [6.11]. Now, we will elaborate on each such set of vertices.

1) *Boundary vertices of the 3-D model*. These vertices cannot be used by the compression algorithms because these are critical vertices. We define critical vertices as those set of vertices which influence the shape of 3-D model. These vertices can be determined using a standard convex hull algorithm. (In our case, we use quick hull algorithm [6.2] to find such vertices). In the case1 of the Figure 6.2 all the boundary vertices are marked with dotted circle. These set of vertices form the convex hull of all the given vertices in the 3-D model. These vertices can be used for watermarking.

2) *Neighboring vertices to split vertex* will never be used by the compression algorithms. Let us illustrate this by an example using case 2 of Figure 2. Suppose edge (v_2, v_3) is collapsed. Then, we can never use vertex v_1 for the compression. This is mainly due to two reasons. First, edge (v_1, v_2) cannot be collapsed anymore due to the constraint of compression. Next, the remaining set of vertices with which vertex v_1 form the edges are boundary vertices. We mentioned above that boundary vertices are never used for compression. So, vertex v_1 is

never used for compression. These vertices are computed after each refinement of CPM [6.9], PM [6.7] and PFS [6.11]. But it does not take extra time to compute these vertices as they are marked when edge collapse operation is done.

3) *Vertices of edges which do not form simple triangle*. In case 3 of Figure 6.2, we illustrate such set of vertices. Edge (v_1, v_2) does not form a simple triangle, so it cannot be collapsed. All the edges of vertices v_1 and v_2 except the edge (v_1, v_2) are with the boundary vertices. So, we can never use vertices v_1 and v_2 for the compression. We can calculate these vertices from the data of 3-D models. The data for 3-D model stores all the vertices and faces. We first sort all the faces (triangles) according to the label of vertices. We now check every two consecutive faces. If any two consecutive triangles have two of its vertices common, we can say that these two vertices form the complex triangle. In this way, we calculate all the vertices that do not form the simple triangle. These pair of vertices cannot be used by compression algorithm.

6.2.2 Watermarking

The above ideas can be integrated into watermarking algorithms resulting in enhancements that give robustness against compression schemes. We can visualize this in Figure 6.3 wherein our scheme gives input to watermarking algorithms. Since the purpose of our scheme is to identify invariant vertices, we need watermarking algorithms that watermark geometric information only. Authentication methodologies such as [6.5, 6.13, and 6.14] that use topological and geometrical information for watermarking can be used for our purpose. However, the output of our algorithm is a set of invariant vertices (geometrical information). So, we need to generate connectivity (topological) information using the geometry (vertices). This can be done by triangulation of the vertices by choosing pairs of

vertices as seeds, wherein each pair has vertices which are maximum distance apart. Once triangulation is achieved, we run the algorithms suggested in [6.5, 6.13, and 6.14] to watermark the data set. However, one should take care that during the encoding process the connectivity stays consistent since this information is used during the watermarking stage, and without an attack, extraction of all the embedded watermarks is possible. Since one cannot guarantee that due to encoding the connectivity will not change, these techniques might not prove to very promising in our case. To solve this problem, we can resort to algorithms that only use geometrical information for watermarking, as suggested in the next sub-section.

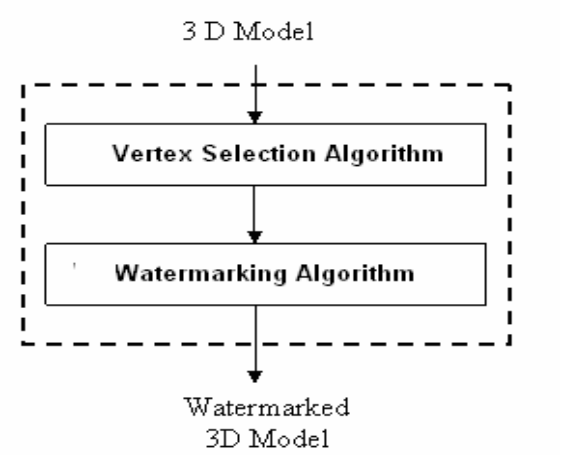


Figure 6.3. Enhancing the Watermarking Algorithm

Geometry based Watermarking: The scheme suggested in [6.1] watermarks the geometry of the data set i.e. uses only the vertex information. This algorithm is spatial, blind in nature and robust against uniform affine transformations. Therefore, we can have watermarks that are robust against uniform affine transforms and progressive compression techniques. However, any attack on invariant vertices results in loss of the watermark. As already shown above, shape defining border vertices are invariant to compression scheme, they are used to

watermark and help in the authentication of the 3-D model shape. Other vertices can also be used to watermark and can serve as tampering indicators.

Customization of Geometry Based Watermarking [6.1]: In the watermarking scheme suggested in [6.1], we first find clusters of 3-D vertices by running a nearest neighbor heuristic. Groups of such clusters are identified based on proximity, such that each group can embed a given watermark of particular size. Inside each such group, we can find an order to encode and decode the bits related to the watermark. Ordering is done by first ordering the clusters, and then ordering vertex information per cluster. After this process, we encode the bit related information in watermark bit sequence and the order of vertices. The encoding scheme is an extension of 3-D (QIM) quantization index modulation, and enhances the hiding capacity and the watermark imperceptibility. During the watermark extraction process, we subject similar grouping and ordering, and use 3-D QIM to decode the bits. As observed in [6.1], the watermarks are robust to cropping or localized noise addition attacks, since groups are identified in a localized manner. Since the scheme suggested in [6.1] deals with watermark robust against cropping, we need to customize it, as the purpose of our watermarks is for authentication. To do so, we do not resort to localized group, but assume one group. This group has all the clusters and encodes the watermark. To make sure that localized cropping or noise addition operations are identified, we chose an order among clusters such that the cluster in sequence are situated farthest from each other i.e. cluster j is farthest from cluster j+1. For example, cluster C_1 and C_2 are near to each other, as compared to clusters C_3 and C_4 and the watermark is encoded in the cluster pairs $\{C_1, C_2\}$ and $\{C_3, C_4\}$. In case cropping occurs, watermarks are preserved and we will still be able to identify the watermarks. To avoid such a problem, we

adapt to the choice of choosing clusters in sequence that are situated farthest. So, if we pick C_1 and C_3 to encode, there is a higher chance for finding the cropping attack. Noise addition attacks can be identified by customizing the bit encoding scheme. This is done by making the bits encoded per cluster sensitive to loss due to noise addition. To achieve the same, the scale used for bit encoding is divided into smaller intervals, since in QIM [6.4]; smaller intervals are more vulnerable to change in bit information.

6.3 Analysis

The following subsections analyze the scheme for time complexity and possible applications for copyright protection.

6.3.1 Time Complexity of Invariant Vertex Selection

The complexity of invariant vertex selection is analyzed as follows:

1) For the vertices of the first type, we need to compute convex hull. It takes O ($n \log n$) to compute the convex hull [6.2]. The space complexity is the number of vertices obtained on the convex hull.

2) The second set of vertices is computed after each refinement. These set of vertices vary according to the compression scheme used. If s is the number of split vertices in a refinement and d is the maximum degree for a vertex, then the complexity for processing these set of vertices is O ($s*d$).

3) In computing the third set of vertices, we first sort all the faces (triangles) according to the label of vertices. This takes O ($n \log n$) and to do the checking between two consecutive faces, it takes O (n) time. So, the overall complexity for computing this third set of vertices is O ($n \log n$).

The overall complexity of the algorithm is also O ($n \log n$).

6.4 Experiments and Analysis

Experiments for our algorithm are done using the 3-D models given in Figure 6.4. Machine used for experiment was Pentium 4, 1.69 GHz, 512 MB RAM. In all the experiments, we assume that the watermarks are embedded in the model after the compression. We have used different types of compression schemes like CPM [6.9], PM [6.7] and PFS [6.11] for our experiments. In order to verify the presence of watermarks, we implemented a spatial vertex based watermarking algorithm [6.1], and verified the presence of the watermarks in all the refinements during compression and decompression process. The set of vertices to be watermarked are the invariant vertices we obtain using our algorithm. Also, the choice of the watermarking scheme [6.1] is invariant to uniform affine transformations (scaling, translation and rotation). Therefore, we can develop a semi-fragile watermarking scheme robust against uniform affine transformations and progressive compression as well.

Table 6.1. Original Vertices before Compression and Decompression

S. No.	3-D Model	Vertices
1.	*Bunny*	35947
2.	*Hand*	327323
3.	*Horse*	48485
4.	*Elephant*	19757
5.	*Dragon*	437645

Since we want to show that watermarks are present in all the refinements, we will first decompress the mesh as shown in Figure 6.5. To verify that our algorithm is robust against compression, we compress the mesh for the second time. Even after the series of decompression and compression, watermarks are found to be preserved in this compressed mesh. Table 6.2 shows that watermarked vertices are preserved even after we perform

repeated compression and decompression. In addition, we see that the number vertices to encode are less since we consider vertices from the base mesh set only.

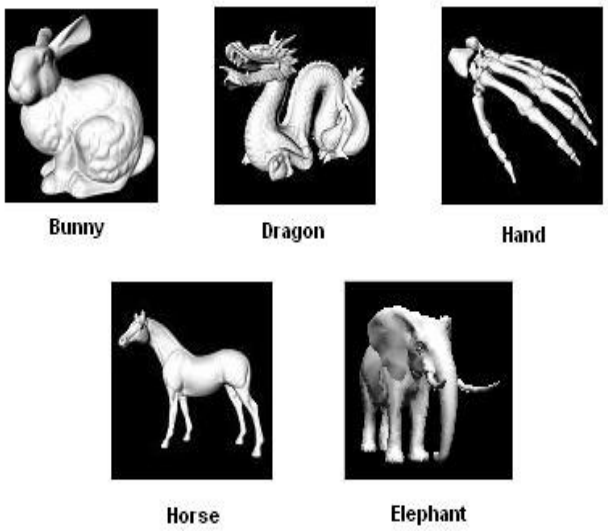


Figure 6.4. 3-D Models used for Experiment

Table 6.2. Vertices that are watermarked and vertices that are preserved after series of compression and decompression

3-D Model	Total Number of Vertices that can be watermarked	Vertices watermarked After Compression	Number of Vertices Watermarked after decompression
Bunny	158	67	67
Hand	3876	1232	1232
Horse	267	134	134
Elephant	103	48	48
Dragon	2345	978	978

Table 6.3. Vertices observed during Compression and Decompression using Compressed Progressive Mesh (CPM)

3-D Model	Vertices After Compression	Vertices After Decompression	Vertices After Second Compression	Total Number of Vertices to be Watermarked
Bunny	470	34834	468	158
Hand	14486	324444	14408	3876
Horse	960	45472	946	267
Elephant	376	18824	373	103
Dragon	8752	432648	8701	2345

Table 6.4. Vertices observed during Compression and Decompression using Progressive Forest Split Compression (PFS)

3-D Model	Vertices After Compression	Vertices After Decompression	Vertices After Second Compression	Total Number of Vertices to be Watermarked
Bunny	868	34987	843	158
Hand	26986	325345	25678	3876
Horse	1765	46492	1745	267
Elephant	821	18927	856	103
Dragon	1543	434659	1432	2345

Table 6.3, 6.4 and 6.5 gives analysis for different 3-D models when they are subjected to compression and decompression. We observe that total number of vertices after compression is different in CPM [6.9], PM [6.7] and PFS [6.11]. But the total number of invariant vertices is same irrespective of the compression scheme.

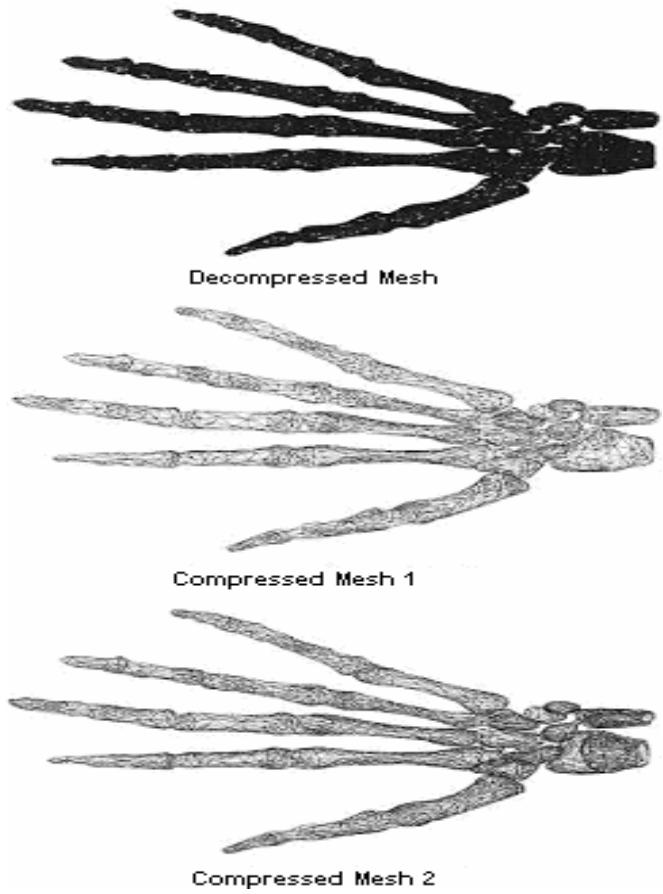


Figure 6.5. Compressed and Decompressed snapshots of Hand 3-D Model

6.5 Summary

A 3-D mesh authentication schemes were developed by using the skeleton of a 3-D mesh as a means to authenticate the shape of the model. The watermarking scheme used for authentications shows that watermarks are preserved during progressive compression-decompression and uniform affine transforms i.e. cases where the skeleton does not change. However the scheme identified noise addition and cropping attacks with simple customization of the watermarking scheme.

BIBLIOGRAPHY

[6.1] Agarwal, P. and Prabhakaran, B. Robust blind watermarking of Point Sampled Geometry. To Appear in Proceedings of 9th Workshop on Multimedia & Security 2007

[6.2] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The Quickhull algorithm for convex hulls. ACM Trans. Math. Softw. 22(4):469-483, December 1996.

[6.3] Cox, I., Miller, M., Bloom, J. Digital Watermarking: Principles & Practice (The Morgan Kaufmann Series in Multimedia and Information Systems), ISBN – 1558607145, 2001.

[6.4] Chen, B., and Wornell, G. W. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. IEEE Trans. on Information Theory, vol 47, pp. 1423-1443, May 2001.

[6.5] Cayre, F., Devillers, O., Schmitt, F., and Maître, H. Watermarking 3-D Triangle Meshes for Authentication and Integrity. INRIA Research Report RR-5223, June 2004.

[6.6] Cayre, F., and Macq, B. Data hiding on 3-D triangle meshes. IEEE Transactions on Signal Processing, vol. 51, Issue: 4, Apr 2003, pp. 939- 949.

[6.7] Hoppe, H. Progressive Meshes. In Proceedings of SIGGRAPH 96, 1996, pp. 99-108.

[6.8] Peng, J., Kim, C.-S., and Jay Kuo, C.-C. Technologies for 3-D mesh compression: A survey. Journal of Visual Communication and Image Representation, Volume 16, Issue 6, December 2005, Pages 688-733.

[6.9] Pajarola, R., and Rossignac, J., Compressed progressive meshes. IEEE Trans.Visual.Comput. Graphics, vol. 6, Jan. 2000, pp. 79-93.

[6.10] Touma, C., and Gotsman, C. Triangle Mesh Compression. In Proc.Graphics Interface '98, pp. 26-34, 1998.

[6.11] Taubin, G., GueÂziec, A., Horn, W., and Lazarus, F. Progressive Forest Split Compression. In Proc. SIGGRAPH '98, pp. 123-132, 1998.

[6.12] Uccheddu, F., Corsini, M., and Barni, M. Wavelet-Based Blind Watermarking of 3-D Models. In Proceedings of the 6th ACM MM & Security workshop 2004, Pages: 143 – 154, 2004.

[6.13] Wu, H.T., and Cheung, Y.M. A Fragile Watermarking Approach to 3-D Meshes Authentication. In Proceedings of the 7th Workshop on Multimedia & Security (ACM '05), pp. 117-123, 2005

[6.14] Yeung, M.M., and Yeo, B-L. Fragile watermarking of three dimensional objects. In Proc. of Int'l Conf. Image Processing, ICIP98, Volume 2, pp. 442-446. IEEE Computer Society, 1998

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Most of the methods proposed in the dissertation for different 3-D content type are applicable for their different format representations. However, for 3-D model's authentication, the technique is applicable to their mesh representations only.

7.1.1 Robust Watermarking Schemes for Copyright Protection

Clustering based spatial blind watermarking approaches to embed and extract watermarks for 3-D models and 3-D motion data were proposed. These techniques are shown to have a high hiding capacity, secure and also try to minimize impact on semantics of the data set. For 3-D models, the schemes are generic and can be applied to point clouds and meshes with a time complexity of O($n \log n$) for watermark embedding and extraction; meanwhile for 3-D motion data these are applicable for a wide range of formats and have a time complexity of O($n \log n$) for watermarking embedding and O($n^2 \log n$) for watermark extraction. The design of the watermarking scheme for 3-D models makes it highly robust against cropping, local noise addition, reordering and uniform affine transforms; meanwhile robustness against global attacks (simplification and noise addition) is comparatively less. For 3-D motion data, the schemes are highly robust against uniform affine transforms, cropping, and reordering. However, the scheme is less against global noise addition attacks and even horizontal cropping attacks where the number of points cropped have lesser points than the twice the

watermarked points. In both the data types, extensions to 3-D quantization index modulation have been proposed and used to do bit encoding and decoding.

7.1.2 Watermarking and Fingerprinting for Tamper Detection and Correction

In order to handle the problem of tamper detection, the dissertation identified a fragile watermarking scheme that could classify and detect with high accuracy attacks such as uniform affine transforms, order reversal and noise addition. The scheme was improved in order to handle false alarms against uniform affine transforms and noise additions that do not alter the semantics of motion. However, these semi-fragile watermarks are shown to identify high intensity noise additions and order reversal attacks. Also, combination of finger printing and fragile watermarking schemes handle attacks on motion data matrix. Attacks such as shuffling and random noise addition can be detected with high accuracy. Meanwhile shuffling attacks can be corrected; noise addition attacks are corrected using interpolation.

3-D mesh authentication scheme were developed by using the skeleton of a 3-D mesh as a means to authenticate the shape of the model. The watermarking scheme used for authentications shows that watermarks are preserved during progressive compression-decompression and uniform affine transforms i.e. cases where the skeleton does not change. However the scheme identified noise addition and cropping attacks with simple customization of the watermarking scheme.

7.2 Future Work

7.2.1 Improving the Accuracy during Authentication of 3-D content

3-D model authentication is one of the problems that we solved by authenticating the skeleton of the 3-D model represented as a mesh. However, 3-D models have other representations such as point clouds and parametric curves and the technique should be able

to authenticate these representations as well. In addition, skeleton of the model cannot always be the best way for shape authentication. This can be understood from the fact that change in shape can be at different resolutions. For example, when we perceive at a shape closely, tampering is more visible. So, defining a scheme that identifies tampering by defining multi-resolutions on the basis of human perception is important.

We have proposed a semi-fragile watermarking scheme that does not alter watermarks during semantic preserving lossy compression. This was done by customizing the scale used for encoding/decoding. However, noise levels can be increased and still watermarks can be lost, as seen in Chapter 2. So, the idea of such semi-fragile watermarks will not work in all cases, since human perception for change in actions in motion need to be correlated as well. So, we need to design a watermarking or finger-printing scheme that can exactly tell us when the motion data changes.

7.2.2 Improving the Robustness of watermarks for copyright protection

Robust watermarks needs to withstand attacks of all nature including localized and global attacks, and uniform affine transformations. It is a known fact in the watermarking field that techniques can be built to be robust against uniform affine transforms, but achieving robustness against local and global attacks consecutively is hard. The future efforts will be oriented towards developing techniques that achieve robustness against all categories of attacks. Since clustering improves robustness against localized attacks and frequency transform attacks achieve robustness against global attack, a technique is required that exploits the benefits of both the methodologies.

VITA

Parag Agarwal was born in March 14th 1976 in Meerut India. He did his schooling from St. Mary's Academy, Meerut in 1994 and joined National Institute of Technology, Calicut to follow up Bachelor's of technology in Computer Science in 1995. After completing his bachelors in 1999, he worked for two years in the Indian IT industry for companies like Cognizant and Nuntius Systems on projects related to Telecom and Web-services. In 2002, he completed his MS in Computer Science from University of Texas at Dallas, and went back to India to work for Motorola India in 2003. In Spring 2004, he rejoined University of Texas at Dallas to follow-up the PhD program in computer science. Apart from academia, Parag has interests in arts dealing with abstract expression in cartoons and photography.