ANIMATED MESH SIMPLIFICATION BASED ON SALIENCY METRICS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ahmet Tolgay

July, 2008

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Tolga K. Çapın (Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Bülent Özgüç

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. A. Aydın Alatan

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

ANIMATED MESH SIMPLIFICATION BASED ON SALIENCY METRICS

Ahmet Tolgay
M.S. in Computer Engineering
Supervisors: Assoc. Prof. Dr. Uğur Güdükbay
and Asst. Prof. Dr. Tolga K. Çapın
July, 2008

Mesh saliency identifies the visually important parts of a mesh. Mesh simplification algorithms using mesh saliency as simplification criterion preserve the salient features of a static 3D model. In this thesis, we propose a saliency measure that will be used to simplify animated 3D models. This saliency measure uses the acceleration and deceleration information about a dynamic 3D mesh in addition to the saliency information for static meshes. This provides the preservation of sharp features and visually important cues during animation. Since oscillating motions are also important in determining saliency, we propose a technique to detect oscillating motions and incorporate it into the saliency based animated model simplification algorithm. The proposed technique is experimented on animated models making oscillating motions and promising visual results are obtained.

Keywords: simplification; animation; mesh saliency; deformation.

ÖZET

CANLANDIRILAN MODELLERİN DİKKAT ÇEKİCİ BÖLGE ÖLÇEĞİNE DAYALI OLARAK BASİTLEŞTİRİLMESİ

Ahmet Tolgay
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Doç. Dr. Uğur Güdükbay
ve Yrd. Doç. Dr. Tolga K. Çapın
Temmuz, 2008

Üç boyutlu bir poligonsal modeldeki görsel açıdan önemli bölgeler dikkat çekici bölge kavramı ile tanımlanır. Bu kavramı kullanan poligonsal model basitleştirme algoritmaları sabit bir üç boyutlu model üzerindeki dikkat çekici detayları koruyarak çalışır. Bu çalışmada hareketli üç boyutlu modeller üzerinde basitleştirme için kullanılacak olan bir dikkat çekici bölge metriği ortaya koyulmaktadır. Bu metrikte durağan modeller için geliştirilmiş olan metriğe ek olarak dinamik bir modeldeki hızlanma ve yavaşlama bilgisi kullanılmaktadır. Bu, animasyondaki görsel açıdan önemli ve dikkat çekici özelliklerin korunmasını sağlar. Harmonik ve periyodik salınım türü hareketler bir canlandırma sürecinde önemli ölçüde göze çarpmaktadır. Bu tez çalışmasında hareket eden poligonsal modellerde bu tür hareketleri algılayabilen ve çarpıcı bölge tespitinde kullanabilen bir model basitleştirme yöntemi geliştirilmiştir. Geliştirilen model basitleştirme yöntemi harmonik ve periyodik salınım türü hareket yapan modeller üzerinde kullanılmış ve görsel kalite açısından başarılı sonuçlar elde edilmiştir.

Anahtar sözcükler: Model basitleştirme, dikkat çekici bölge ölçümü, salınım hareketi algılama, dinamik model animasyonu.

Acknowledgement

I would like to gratefully acknowledge the support and supervision of Assoc. Prof. Dr. Uğur Güdükbay and Asst. Prof. Dr. Tolga K. Çapın who allowed me to have the opportunity of pursuing this research. With their support and motivation this work has become possible.

I want to thank my parents, their parents and every member of my family for their love, patience, encouragement and everything they did during my study. Special thanks goes to my aunt Nurhan for her endless moral and logistics support. In addition, I greatly appreciate the motivating efforts of my old friends.

I would like to express my gratitudes to my jury members Prof. Dr. Bülent Özgüç, Prof. Dr. Özgür Ulusoy, and Assoc. Prof. Dr. A. Aydın Alatan for their valuable comments and suggestions.

We used public sample data of Animazoo and StockMoves generic free motion data of Motek Entertainment. Their sample BioVision hierarchy files allowed us to find specific motions among a huge set. The horse gallop animation is taken from Jovan Popović `http://people.csail.mit.edu/sumner/research/deftransfer/data.html#download`.

Contents

List of Figures

Chapter 1

Introduction

Continuous increase in the visual quality and complexity of 3D meshes has led the problem of simplifying them while preserving their visual properties. As they contain more information, we need more processing power for visualization using previous techniques. However in the current time frame, processing capabilities are limited in terms of memory and instructions per second even if we consider parallel computing systems. These limitations bring the need for new techniques that are able to reduce the information so that it is more feasible in terms of processing speed while quality of mesh is as close as possible to the original.

Since this necessity is as old as the first meshes ever used in industry, this problem has been studied extensively and many techniques have been developed. For an in depth survey, see [7]. Most of these studies have focused on techniques that simplify single static meshes. In this thesis, the main concern is animated mesh sequences that are composed of a number of meshes deforming in time. This kind of data has additional information that should be taken into consideration, such as the speed and acceleration that different parts of the mesh have and specific kinds of patterns about the motion, such as oscillations. Connectivity is preserved during this deformation. Otherwise, acceleration information of the mesh would be lost. There are many applications using these kinds of animations, which build the motivation behind this study. Skinned meshes using the motion data coming from an articulated skeleton either captured or created by an artist are common in virtual environments and video games. Our method can be applied to these meshes or any complete set of mesh data with preserved connectivity. The goal is reaching a predefined number of vertices for all frames having the maximum visual quality and minimum error. For this reason, a view-independent error metric for the saliency-based simplification of animated meshes is proposed.

The proposed error metric is composed of two parts that are used to define the important regions of the sequence:

- The mesh saliency metric developed by [16]. This part works on static meshes and it is calculated for every single mesh independent of each other.
- The animated mesh saliency metric that is proposed by this thesis, which depends on the speed and acceleration information contained in an animation.

Evidently, the main concern here is detecting the salient parts of meshes. This is currently an open problem that involves many disciplines including psychology, psychophysics and neuroscience. The main contribution of this work is addressing this problem and proposing algorithms towards building a solution base. After detecting salient parts using the proposed metric, those parts are preserved during the simplification process. An extended version of the QSlim [26] algorithm for the simplification of static meshes is used by including the saliency information derived from the deformation which constrains simplification in deforming areas. This method targets generic deformable meshes and is closely related to the algorithms presented in [1, 6].

The organization of this thesis is as follows. Chapter 2 provides background information on mesh simplification and saliency, for both static and animated meshes. In chapter 3, saliency metric for static meshes and our animated mesh saliency metric are presented. Afterwards details about our oscillation detection algorithm are given. Chapter 4 provides simplification results of our metric and two different approaches to use it. Finally chapter 5 concludes.

Chapter 2

Background

Previous work about this study can be classified in two parts: mesh simplification and saliency. These concepts were combined recently by the work of [16]. In this chapter, related work about mesh simplification and saliency will be explained. Afterwards, studies about saliency-based metrics and animated meshes will be discussed.

2.1 Mesh Simplification

The idea that we benefit from simple meshes was initiated with using different resolutions of static meshes for different viewing distances [19]. Since then, many methods have been developed, which can be categorized according to several criteria, such as view dependence, metric being used, change in topology and the mechanism used as explained with detail in the survey by [7] and [8]. Q-Slim is one of the popular algorithms that uses iterative edge contractions (see Figure 2.1) with the help of quadric error metric (QEM) [26]. Basically, it works in 5 steps:

1. find the quadric matrices for every vertex,
2. select the pairs that can be contracted,

3. find the target vertex for each pair and the cost for that contraction,

4. insert all the pairs in a min-heap,

5. remove the minimum cost pair by contracting them and compute the new costs related to that pair.

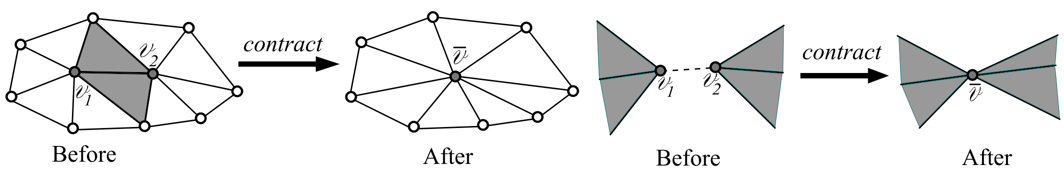

Figure 2.1: Edge contraction. Vertices v_1 and v_2 (connected on left and unconnected on right) are contracted to $\bar{v}$

Here, cost is the error introduced by contraction and calculated by summing the squared distances between the new vertex and the planes of the contracted vertices v_1 and v_2. It has been enhanced for simplifying meshes with geometry and other additional attributes like color, texture, etc. and the area of adjacent faces were incorporated to the algorithm for improving edge cost metric [27, 15]. The original algorithm has a good balance between speed and space, but numerous innovative algorithms focused on QEM have been introduced; either advancing it or using it in a different simplification procedure. A topology preserving variation was introduced [10] and later implemented using QEM [39].

Progressive meshes [14] are used to maintain levels-of-detail of static mesh models. The method tracks the edge collapse and vertex-split operations to move from finer to coarser level of detail and vertex-split operations to get to the finer level of detail from the coarse mesh. Lindstorm et al. [31] use the quadric error metric for selecting the representative vertex of a cluster minimizing the error, using the fact that vertex clustering is equal to the contraction operation applied on every vertex of a cluster at the same time. Shaffer and Garland [11] use dual quadric metric, which is a measure of distance of a plane to a set of points just as the quadric metric is about the distance of a point to a set of planes. Their algorithm is a two-pass process. In the first pass, the model is quantized using a grid and surface information is calculated in the form of dual quadrics. In the second pass, the original vertices are clustered using a BSP tree constructed by the surface information in the first pass.

Zhang and Turk [12] use a visibility measure for guiding the priority of edges to be contracted. In their work, it is considered if some edge is not visible enough from an evenly distributed number of camera viewpoints, then that edge does not cause much error in terms of visible quality when contracted. Hong and Kaufman [40] use a similar decimation algorithm using a different error metric on tetrahedral meshes. Their error metric uses the gradient magnitude, density value and the volume for calculating error cost associated with the decimation. Wu et al. [43] add global geometry features of the mesh as a weight coefficient for the edge costs of QEM. Their geometry feature is based on the crease angle of an edge stating that the smaller the crease angle, the more important the geometry feature is.

Ho et al. [38] use Q-Slim with the help of a user for selecting the regions to remain detailed. They have a system composed of two stages: weighting and local refinement. In the weighting stage, the user selects an important region that causes delaying the collapses in that region. After this stage is completed with satisfactory result, the user selects a new set of vertices in the local refinement stage. The system splits these vertices and collapses some low cost edges in order to preserve the polygon count. This system is specifically successful when the polygon count is very low. DeCoro et al. use QEM for calculating the optimal representative vertex position of a cluster for their real-time simplification algorithm [5]. They utilize the GPU for increased performance having an average of 20 fold increase in the throughput compared to CPU. Cook et al. [35] use stochastic simplification in order to simplify excessively complex scenes which otherwise can not be rendered using current resources in terms of hardware and software.

2.2 Concept of Saliency

Saliency, which characterizes the level of significance of the subject being observed, has been a subject of cognitive sciences for more than 20 years. It is closely related to many disciplines like artificial intelligence, neuroscience, psychology and recently computer graphics. In this section, recent studies about saliency and perception related to computer graphics or mesh simplification will be mentioned briefly.

Reddy [28] uses the models of visual perception, including vision metrics such as visual spatial frequency and contrast, to optimize the visual quality of rendering of flythrough in a scene. Luebke and Hallen propose a perceptually-driven rendering framework which evaluates local simplification operations according to worst-case contrast gratings and worst-case spatial frequency of features they can induce in the image. In their work, contrast grating is a sinusoidal pattern that alternate between two extreme luminance values and worst-case one is a grating with the most perceptible combination of contrast and frequency possibly induced by a simplification operation [9]. They apply the simplification only if a grating with that contrast and frequency is not expected so they do not get any perceptible effect which results a high fidelity model. A set of experiments have been performed using three groups of tasks for measuring visual fidelity [4]. These tasks are naming the model, rating likeness of the simplified model compared to a standard one using a 7-point scale and choosing the better model among two equally simplified models using Q-Slim and V-clust [21]. The results of these experiments and some automated fidelity measures ([24], [30]) show that automated tools are poor predictors of naming times but very good predictors of ratings and preferences.

Williams et al. extend the perceptual simplification framework by [9] to models with texture and light effects [29]. Howlett et al. use an eye tracker to identify salient regions and the fixation time on those regions of models and modify Q-Slim to simplify those regions with a weight value [36]. As a result of experiments similar to [4] it is shown that the modified Q-Slim performs better on natural objects but not on man-made artifacts which indicate that saliency detection is an important property with promising results. A saliency metric that does not make use of eye trackers is developed by [16] and used in our study. This metric heavily depends on the curvature information of surface model and gives results with higher fidelity compared to Q-Slim. It is shown that visual attention can be directed by increasing the saliency at user selected regions using geometric modification [41]. With a weight change in the center-surround mechanism, they modify mean curvature values of vertices by using bilateral displacements and verify that the change increases user attention by using eye trackers.

Another saliency metric and a measure for degree of visibility is proposed by [25]. Their saliency metric makes use of the Jensen-Shannon divergence of probability distributions [18] by evaluating the average variation of JS-divergence between two polygons yielding similar results as [16]. A saliency map for selective rendering which makes use of colors, intensity, motion, depth, edges, and habituation which refers to saliency reduction over time as the object stays on screen is developed using GPU [32]. Their saliency map is based on the model suggested by [23].

Saliency is also used in viewpoint selection criteria [17]. In their study, some viewpoints are selected among a sample point set forming the vertices of a graph on the bounding sphere of an object. The graph is partitioned according to the degree of similarity between its edges and the selected viewpoints represent partitions with the most similar edges. Then, they are sorted using [16] to select the most salient one. The mesh saliency metric is improved using Morse theory [42]. In this work, two main disadvantages of [16] are pointed out. One is the Gaussian-weighted difference of fine and coarse scales can result in same saliency values for two opposite and symmetric vertices because of the absolute difference in the equation (see Equation 3.1) and the other one is combining saliency maps at different scales makes controlling the number of critical points difficult. Therefore, instead of the Gaussian filter they use a bilateral filter and define the saliency of a vertex as the Gaussian-weighted average of the scalar function difference between its neighboring vertices and the vertex itself.

A recent work uses database of objects to measure the distinctiveness of different regions of an object [33]. It is based on the idea that if a region has the most unique shape which is used to differentiate the object from other objects, then that region is an important part of the object. It works by selecting several random points which are centers of overlapping spheres over the surface and generating shape descriptors from the surfaces covered by those spheres. Next, a measurement of how distinctive is each region with respect to a database of multiple object classes is taken and if the best matches of a region are all from that objects own class then that region is distinctive. Although a database is required, it gives better results than [16] in terms of simplification quality.

2.3 Animated Mesh Simplification

Simplifying animated meshes includes the same problems as static meshes, but animation increases the data to be processed heavily. This topic has not been much studied until the Deformation Sensitive Decimation (DSD) algorithm [1]. It works on an animation with k different example poses which have the same connectivity. When a model is simplified using Q-Slim and then animated, the deforming regions can be low quality since the algorithm cannot know those regions in advance. To solve this problem, they simply assign the total cost of an edge over every pose so that a deforming region which has high costs for some poses is not much simplified because of the other poses and only the edges which have low cost totally get simplified first.

Based on [22], Brown et al. propose a visual attention based LOD management technique [34]. Their framework uses a visual attention model to predict eye movements and then attaches importance values to different regions of scene using these predictions. The visual model is composed of size, position, rotational and translative motion information and luminance of objects in the scene during an animation. The triangle budget is spent non-uniformly giving high detail to important objects detected using the attention model.

In another work [6], simplification of articulated body animation is studied. An approximate probability distribution is required instead of a set of example poses providing pose independency. All vertices are transformed into a frame of reference and the error costs of transformed vertices are summed with respect to that frame of reference. Any articulated and skinned mesh can be handled by this algorithm. Simplifying only a static pose of an articulated figure is proposed by [20]. Dynamic mesh simplification methods have been proposed by [2, 3] where the low frequency affine transformation information is separated from the high frequency deformation information between mesh models at subsequent poses. This method focuses on run-time simplification with an auxiliary set of data structures such as T-Directed Acyclic Graph [2], pre-computed offline. For a better quality simplification of highly deformed regions, a Deformation Oriented Decimation (DOD) algorithm is proposed by [13]. It incorporates deformation information into the cost metric by adding a deformation cost ξ to the DSD cost [1],which is composed of three components.

$$\xi_{ij} = w_{ij} * \bar{A}_{ij} * \left(\overline{\Delta} l_{ij}\right)^2 \tag{2.1}$$

Here ξ_{ij} is the cost that is added for contracting vertices v_i and v_j along all the frames. $\overline{\Delta} l_{ij}$ is the average length change that contributes as the deformation measurement. It is the sum of edge length changes between two consecutive frames for the entire animation. $\bar{A}_{ij}$ is the total area of triangles that share the edge between v_i and v_j along all frames. w_{ij} is a weight representing the normalized maximum deformation and helps to preserve the edge with irregular deformations. This weight is the result of the fact that we pay more attention to unexpected sudden deformations.

Sumner et. al propose a new way of transferring deformation from a source mesh sequence to a target mesh [37]. They specify a set of marker vertices manually for building correspondence between the source and target meshes. Then the set of transformations of source mesh are computed and applied on the target mesh.

Despite all these innovations, the problem of finding salient regions over an entire animation is still an open problem. Saliency metrics developed for static meshes, such as [33] or [42], can also be used on dynamic meshes, but they do not use any specific information about the motion. DOD is already an improvement over DSD and it focuses on deformation, not saliency. Therefore, we propose a new animation saliency metric that uses static mesh saliency and the motion information available in the animation. We developed an experimental oscillation detection algorithm because we suppose oscillating motions increase the saliency of the region performing that kind of motion.

Chapter 3

Saliency of Animated Meshes

In this thesis, we propose a new, view-independent animation saliency metric that estimates the visual importance of a vertex in a dynamic mesh during the course of an animation. It is view-independent to make it suitable for a general-purpose algorithm. Our work extends static mesh saliency metric, as proposed by Lee et al. [16], to animated meshes. Lee et al. aim to find a metric to define the visually important parts of an input static mesh that could be used in mesh simplification and viewpoint selection. The static mesh saliency of a vertex v is calculated as follows:

$$S(v) = |G(C(v), \sigma) - G(C(v), 2\sigma)| \tag{3.1}$$

where $S(v)$ is the mesh saliency and $G(C(v), \sigma)$ is the Gaussian-weighted average of the mean curvature at vertex v at a distance σ.

In other words, mesh saliency is computed by calculating curvature on each vertex of the mesh, finding the absolute difference between Gaussian-weighted averages of these curvatures at fine and coarse scales, and repeating this process to combine different saliency values found for different standard deviations of the Gaussian filter. This saliency value relies on the intuition that regions with different curvature characteristics from their neighborhood are attractive to human vision. Lee et al. have also demonstrated a mesh simplification solution using this saliency value, where the least salient regions are simplified first. The experimental results show that saliency-based simplification of static meshes provides perceptually better results compared to earlier methods.

An example result from our saliency based simplification is shown in Figure 3.1. Except the feet, most salient parts of the horse are grouped on its head because of the frequent change of curvature there. Its feet (see Figure 3.2) are highly salient due to animated activity but since they do not contain much detail, weight is given to static saliency in this sample. Its head being salient causes significant decrease in the number of polygons on its neck and back.

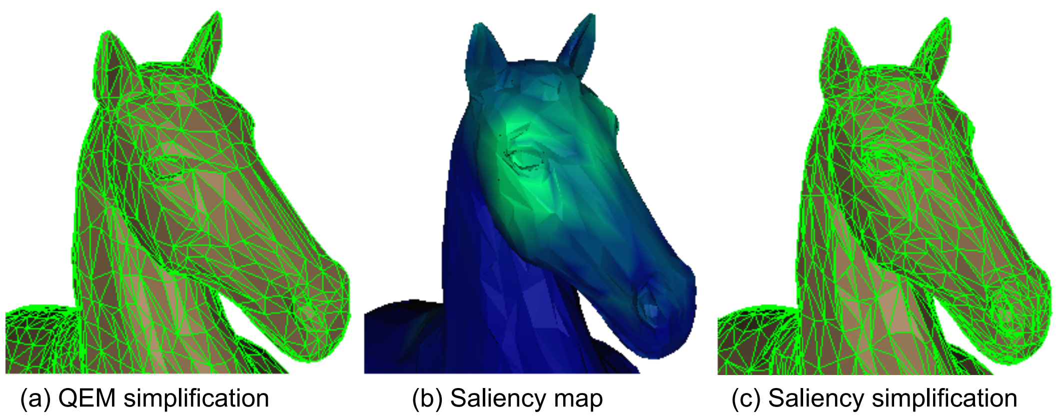

Figure 3.1: Comparison between Q-Slim and saliency metrics. (a) is a model simplified to 50% using Q-Slim, (b) is the saliency map of the same model and (c) is the same model simplified to 50% using saliency map.

Our work extends this saliency metric to animated meshes, which determine the priority to be given to each region of the mesh. The proposed animation saliency metric can be used for animated mesh simplification, compression, viewpoint selection, etc. We demonstrate the application of this metric for the simplification of animated mesh sequences.

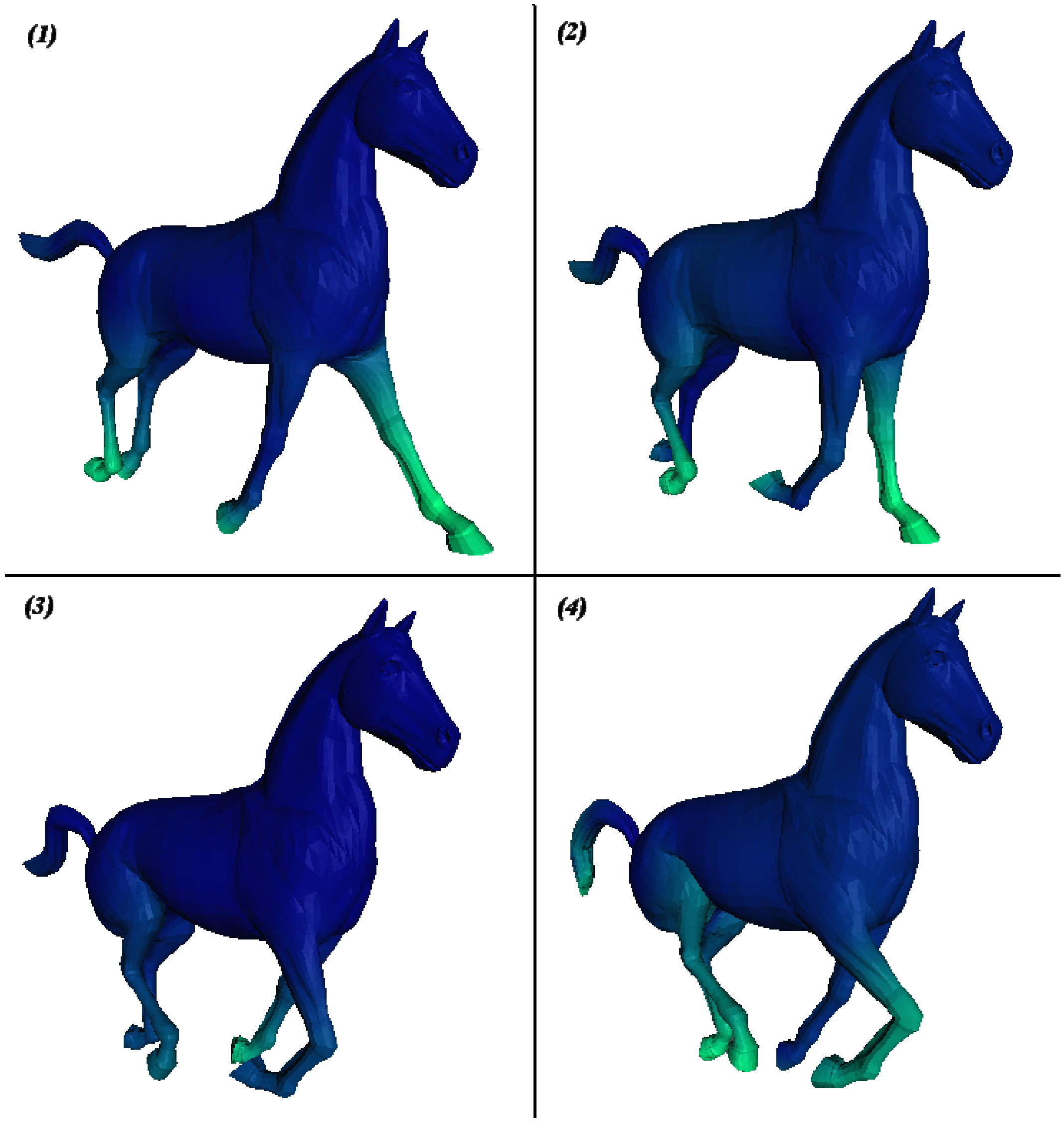

Figure 3.2: Four successive frames with salient regions shown in green.

One solution to the problem of finding salient regions in an animated mesh is to compute a static saliency map for each frame of the animation, then averaging of the saliency for the complete sequence, and finally simplifying the mesh with these average saliency values. Alternatively, a small number of important regions can be computed for the whole animation using a combination of the static mesh saliency and other cues related to animation. Our solution follows the second approach. This significantly decreases the computational and storage requirements as compared to computing and storing a different saliency map for each frame.

3.1 Animated Mesh Models

For labeling the vertices of our mesh sequence with n frames, each having m vertices, the following notation is used:

$$V_1^1, ..., V_m^1, ..., V_1^n, ..., V_m^n$$

We assume that animated meshes that we work on can have several kinds of animations such as object transformations, local mesh deformations, or any combinations of these. We assume that transformations are affine and can be of any nature - such as simple linear animations (e.g., by keyframe animation with linear interpolation), highly dynamic motions (e.g., motion captured character animation), periodic movements (e.g., walking), or scaling in any axis combination. Similarly, we place no constraints on the type of deformation of an object - it could have a small-scale deformation (such as a flag flapping with the force of wind, or creases formed in cloth animation), or medium-to-large scale (such as an object morphing into another object). We assume that the number of vertices and the connectivity of the original mesh do not change throughout the animation, which is the case for most deformable and skinned mesh animations.

For being able to work on different kinds of animations performing motions we want, we used motion capture data. For transferring the motion data a skeleton and envelopes around the skeleton segments were used. The segments which get the motion from the capture data are effective on the vertices around them and on the vertices at the neighboring segments' tips. Envelopes define the level of affection which helps lowering the effect as we go from the middle to the tip of the segment. However, during the motion transfer to our humanoid model, many regions were unnaturally deformed due to effects of several segments of the underlying skeleton. Around highly deformed parts of the model like hip or neck, where more than one skeleton segment are effective on the motion, it is hard to define the boundary curve that indicates the end of affection for a segment. With that many controls it becomes very hard to characterize the motion effecting elements of every vertex over a highly complex model. For this reason, we have used these models for only detecting periodic motions which is actually the only reason for their existence.

3.2 Perceptual Model

Since we are to simplify an animated mesh sequence, we need a new metric for defining salient regions because different types of motion change the focus of attraction on the animation. A 3D object with moving parts would have salient points in those regions, because human visual system is object-bound: we tend to track and follow moving objects in a scene, and we are also able to compensate for moving viewpoint while tracking the moving object with the help of our vestibulo-ocular reflex.

Our method finds salient regions in an animation in two stages. The first stage finds the magnitude of motion using the positions of vertices at each frame in the world coordinate system. We calculate the Euclidean distances for all vertices in all frames as follows:

$$\mathbf{d}_{ij} = \left\| \mathbf{V}_j^i - \mathbf{V}_j^{i-1} \right\|, \tag{3.2}$$

where $\mathbf{V}_j^i$ is the global position of the j^{th} vertex of i^{th} frame, $\mathbf{V}_j^{i-1}$ is the global position of the j^{th} vertex of $(i-1)^{th}$ frame, and $i \in 2 \ldots n$.

Then, we normalize these distances for each frame by dividing all of them to the maximum one and compare the relative movement of each vertex between two consecutive frames. We can detect most of the simple motions (mesh deformation, no global transformation) successfully by this way (see Figure 3.2).

At this point, our contraction cost for an edge formed by vertices v_x and v_y of frame i is in the form below:

$$C_{xy}^i = QEM_{xy}^i + \frac{\left(\delta_x^i + \delta_y^i\right) \times w_a + \left(S\left(v_x^i\right) + S\left(v_y^i\right)\right) \times w_s}{2 \times Bf} \tag{3.3}$$

where QEM_{xy} is the quadric error for that edge, δ_x^i is normalized distance between vertices v_x^i and v_x^{i-1}, $S(v_x^i)$ is the static saliency of vertex v_x^i. w_a and w_s are the user-defined weight coefficients for animated and static saliency metrics, respectively. *Bf* is the balancing factor, which is explained in Chapter 4.

Briefly, we find the moving regions and mark them as salient as faster they move in this stage. For many cases a small region is salient; however there are some cases in which nearly the entire object is moving at the same speed and some little regions are moving slower causing the whole object look salient. In such cases we see that whole object is marked as salient incorrectly since it is uncommon that most parts of an object draw attention simultaneously. Consequently, if the animated mesh has smaller parts with very small acceleration compared to a larger accelerating component, they should be expected to be salient because of their relative motion to the rest of the object. For that reason, we define a threshold (τ, ρ) with τ being the threshold saliency value and ρ being the threshold ratio. It is interpreted as follows. If more than $\rho\%$ vertices of a frame have a saliency value higher than τ, then change all the saliency values τ with 1-τ. This procedure corrects the possibly wrong saliency information that may be introduced (see Figure 3.3).

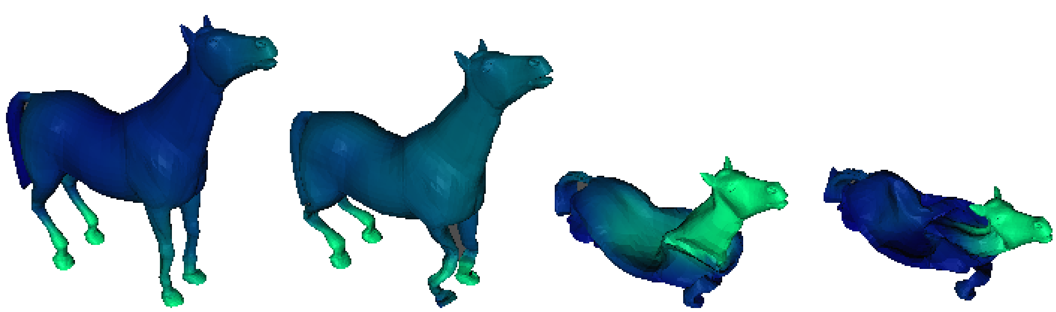

Figure 3.3: With the help of threshold value $(0.5, 70)$ we correct the animation saliency values on this collapsing horse model. Note that lighter regions are more salient.

In the second stage, we detect a special oscillation motion by using the distance information extracted in the first part. Since we have the distance information at each frame and the time passed between each frame is equal, that gives us the velocity information in a different scale. At this stage, we use the information of direction change. Direction change happens if the angle between two segments connecting three vertices in frames $i-1$, i and $i+1$ is not equal to 180 degrees.

Since we do not have direction information in the first stage, the graph shows the motion as if it is happening on a line. If we can obtain the direction changes on the timeline, we can deduce the signs of an oscillating move at that vertex. In this way, we can mark the vertices with a higher saliency compared to a moving part with a similar velocity. In order to detect a regular direction change pattern, we need a model with that kind of motion. To this end, we transferred several motion capture data to our model. It is also possible to work on raw motion capture data for detecting specific motions but that would be relevant for only the animations that use those motion capture data.

Measuring the direction change between frames is relatively easy. The first frame has no direction; first two frames form a direction but they cannot have any direction change; therefore, only at the third frame there can be a measurable direction change. Beginning from the third frame, we calculate the angle between the two line segments formed by vertices of three consecutive frames as in Figure 3.4.

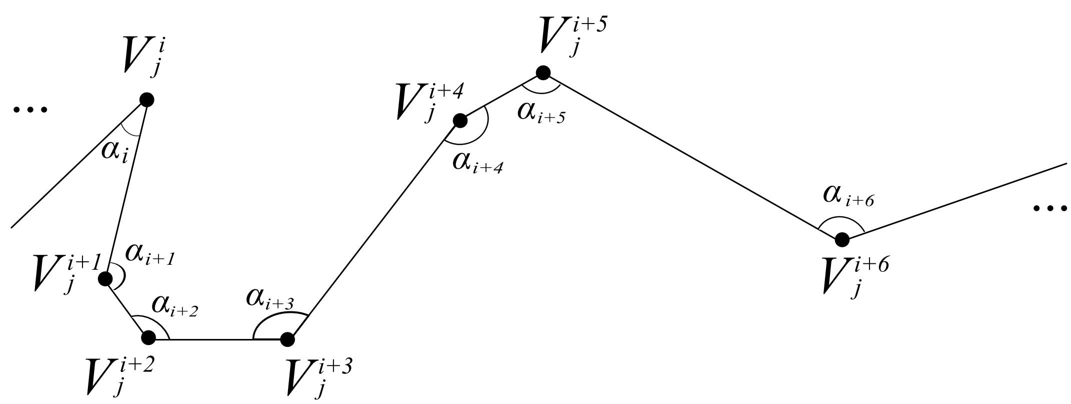

Figure 3.4: The angles between segments to measure direction change between frames.

We define a significant direction change as an angle smaller than 90 degrees but not every oscillating motion has to have vertices having a 90 degrees of change during the course of motion so this angle threshold is subject to change according to the motion or oscillation. After identifying the frames that have direction change smaller than the threshold, we determine how distant two consecutive frames that have similar direction change. We mark each frame with 1 if it has a direction change and 0 if not. Let the number of frames which do not have direction change between two 1-marked frames be σ_i. That σ series give us the information we need since we are looking for a sequence of similar numbers as the indication of regular direction change behavior.

Figure 3.5 shows a sample of how far each two consecutive frame labeled as 1 is generated. The idea is if most of the 1-frames are at similar distance to each other, then we have a regular motion pattern at that vertex. Therefore, we find the variance of that distance list for each vertex and mark the vertices that have specific variances. Since we do not have many motion samples, we have to find those specific variances manually by checking the variances at the regions that do the specific motion and updating the variance control in our implementation.

0	1	1	0	0	0	1	0	1	1	0	1	1	1	0	0	0	1

Figure 3.5: Frames having a direction change smaller than the threshold are labeled with 1.

Figure 3.6 gives the oscillation detection algorithm. It measures the variance of the distance between two direction changes seeking for a regular window length change in the array $unchangedWindow$. The complexity of this algorithm is $O(mn)$ where m is the number of vertices in the model and n is number of frames. Thus, it does not bring too much processing overhead.

[1] **for** each vertex $\mathbf{V}_j^i$
  //here $unchangedAngleWindow$ is the window length specifying the distance between two angle changes happening with an angle smaller than the threshold
[2] $unchangedAngleWindow = 0$
[3] $currentSlot = 0$
[4] **do for** i = 3 to n (for n total frames)
[5]   **if** $\mathbf{V}_j^i.angleChange > angleThreshold$
    //meaning the direction has not changed
[6]   **then** $unchangedAngleWindow \leftarrow unchangedAngleWindow + 1$
[7]   **else**
    //then it is the end of $unchangedAngleWindow$ therefore close the window, empty it and save it in the array's next slot
[8] $unchangedWindow[currentSlot] \leftarrow unchangedAngleWindow$
[9] $unchangedAngleWindow = 0$
[10] $currentSlot \leftarrow currentSlot + 1$
[11] find the variance of data in unchangedWindow
[12] **if** variance is within limits
[13] then mark every frame of current vertex
[14] **else**
[15] unmark every frame of current vertex

Figure 3.6: The oscillation detection algorithm

Chapter 4

Results

We use QSlim to simplify the animated model by integrating the proposed saliency metric and the quadric error metric it uses [26]. With different weights, we can get a good simplification in different sequences. For a model with a dominating static saliency, we can decrease the animated saliency weight by decreasing the coefficient w_a(see Equation 3.3). For a model without significant regions having static saliency, we can increase the weight of animated saliency. The effect of weights become more clear as we further simplify the model. The regions having salient vertices are shown in green in animated saliency maps (see Figures 4.1 (b) and 4.2).

Figure 4.1 compares the QSlim quadric error metric applied to each frame individually to our animated mesh saliency metric for mesh simplification. Since the fastest moving parts are the model's feet they have high saliency for most of the frames. In Figure 4.1 (c), the upper part has more polygons making it high fidelity and less error prone in case of deformation. There are two ways of incorporating the saliency values into the quadric error metric:

1. We could set the errors before any simplification is made. In this case, the contraction costs specifying the order of contractions are calculated using our saliency metric (see Equation 3.3) and inserted to the heap before simplification starts. No modification or calculation related to saliency is

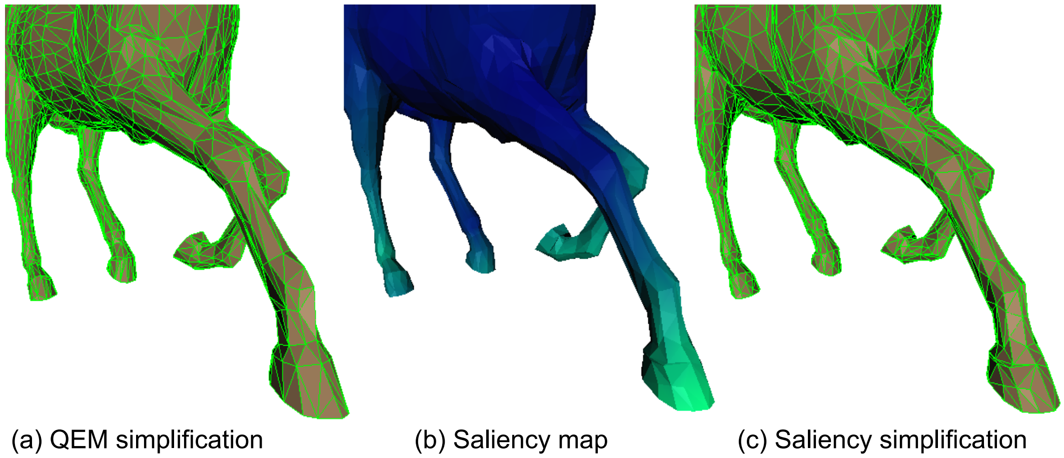

Figure 4.1: Comparison between Q-Slim and our animation saliency metric. Both models are simplified to 25%.

made during the simplification.

2. We could add the saliency values during the simplifying process. In this case, no calculations or changes are made before the simplification. Since the saliency values are available for each vertex, we can use them as we do the contractions. We add the saliency metric to QEM metric iteratively before each contraction. Since the saliency values are added one-by-one after each contraction, they accumulate after some time causing some vertices to be highly salient. This leads to contractions happening around those salient regions most of the time, which cause an imbalanced order of simplification. We divide all the saliency values with a balancing factor to prevent this.

We tried both approaches and observed that the second approach results in a better model in terms of both distance from original and perception (see Figures A.1 and A.5, A.3 and A.10). The comparisons indicating the distance between original and simplified models for QEM and our metric are listed as graphs in appendix. The first four graphs are generated using first approach and the rest are results of the second one. Although it is affordable, we get worse result compared to QEM either way because it is designed to achieve the least geometric error and since we modify the order of contraction for a better perceptual result, we get more geometric error inevitably. The Hausdorff distance graph in

Figure A.2 has many spikes since it measures the most distant parts but we get significantly better results using the second approach as can be seen in Figures A.6 and A.8. The effect of different weights for animation and static saliency values is not very significant in terms of distance from original (see Figures A.11 and A.12, A.5 and A.9). The reason is at these level of contractions (25% and 50%) simplifications are not enough to cause severe deformations allowing different weights change the contraction distribution safely.

In Figures 4.3 and 4.4, there are some vertices mostly on legs marked as oscillating, which are actually not oscillating like the regions we want to identify. It is generally because those vertices also do little oscillations that have similar variation patterns which are hardly attracted by our attention. In some other animation those kinds of little oscillations may be salient depending on the viewpoint but our algorithm is view independent. In order to eliminate those false positives, a classification using relatively unique properties like the angle change pattern or distance graph of vertices can be made. However, most of the target regions (the hands and feet in Figure 4.3 and the hands in Figure 4.4) are correctly marked.

We tested our algorithm on cloth simulations. For motions under gravitational and wind forces, we observe that in many cases cloth behavior is too random for finding a regular oscillation pattern. Figure 4.5 shows the result of one sample data that we could generate acceptable oscillation. In this example, four vertices are fixed and the rest of the cloth moves under gravitational force. The cloth is not so stiff thus it can generate oscillating motion.

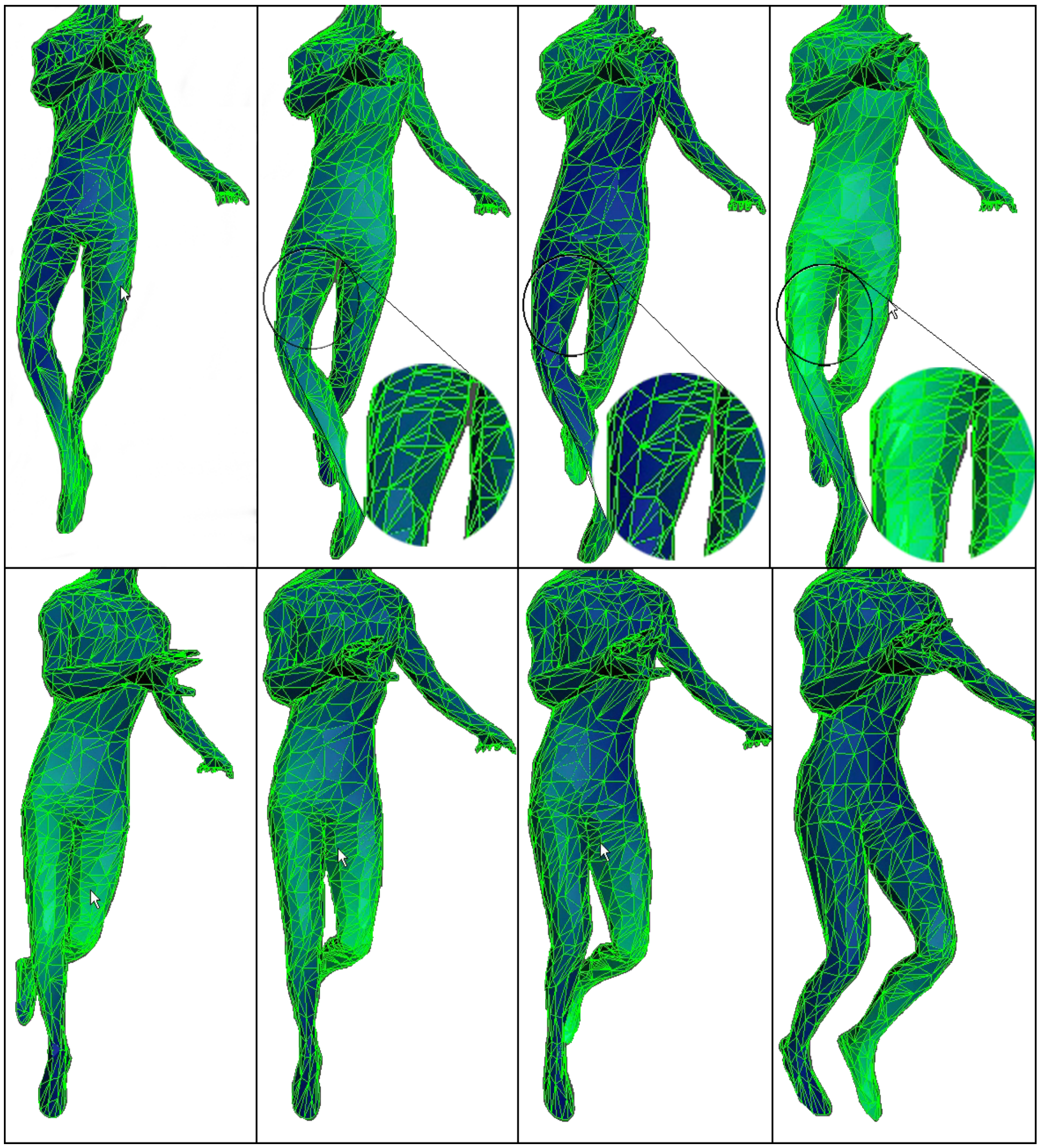

Figure 4.2: The results of our saliency-based dynamic mesh simplification. Here the dancer model is simplified to 18.5 % of its original complexity. The subfigures are several sample frames from the animation. Note that the magnified regions in the top row show the salient regions marked as green at the right subfigure and non-salient regions at the middle subfigure left as blue.

Figure 4.3: The results of our oscillating motion detection algorithm. The direction change threshold angle is taken as 100 degrees and we mark the vertices having variance less than 2. Green is an indicator of the salient regions; red is an indicator of an oscillating motion.

Figure 4.4: The results of oscillating motion detection on another motion capture sequence. The direction change threshold angle is taken as 90 degrees and we mark the vertices having variance between 2 and 6.

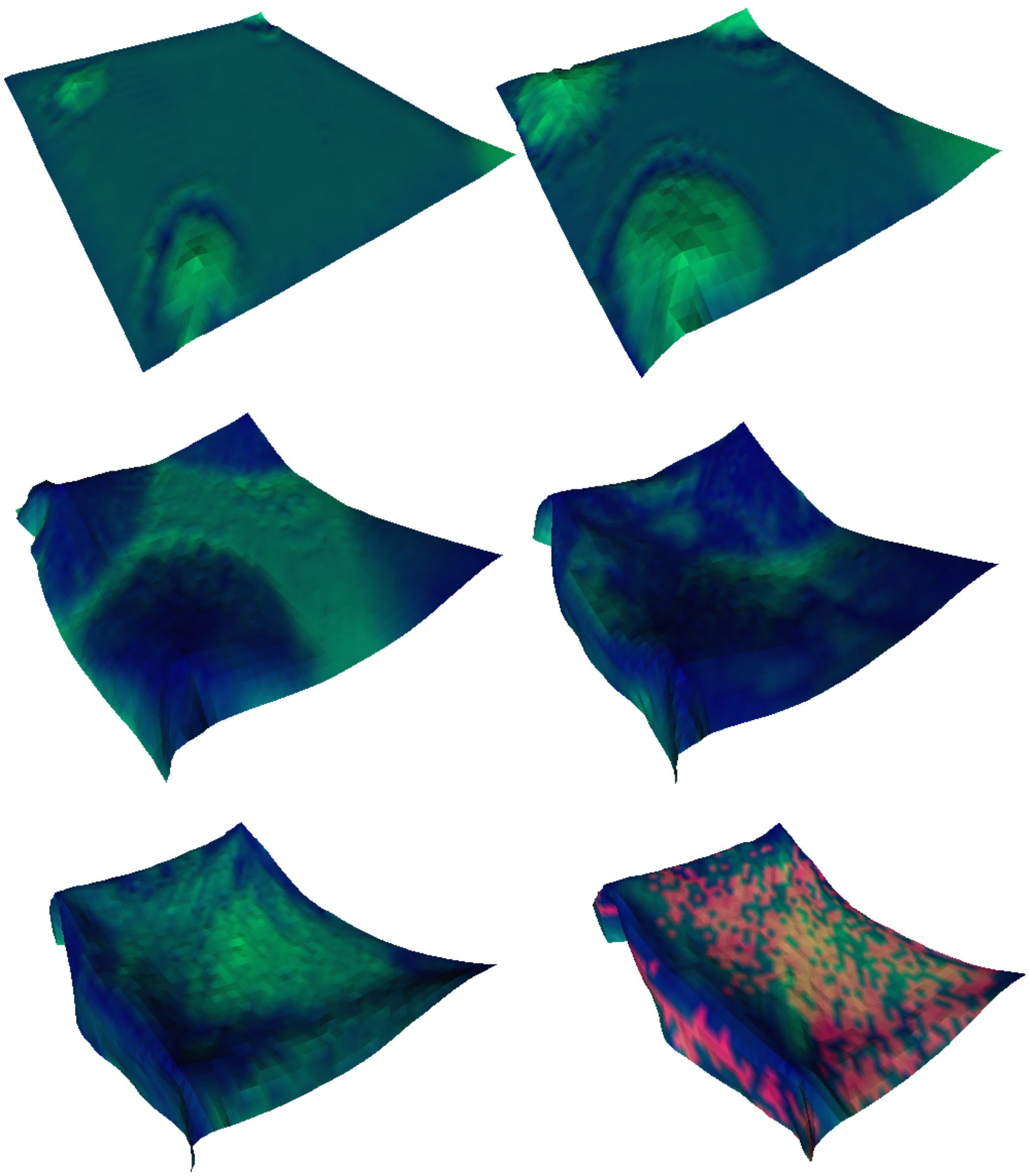

Figure 4.5: The result of oscillation detection on cloth simulation. First five frames show the salient regions based on our animation saliency metric. Direction change threshold angle is 90 degrees and vertices having variance between 20 and 50 are marked as oscillating at the last frame. There are four fixed vertices holding the cloth. They can be noticed clearly in the second frame as they are in the middle of four salient regions near the four corners of cloth. The left two vertices cause left part oscillate separately and middle regions of that part are marked as well.

Chapter 5

Conclusion

We have proposed a saliency metric for finding visually important regions in an animated mesh and used it for simplifying animated meshes. The proposed solution provides a better estimate of the most important regions of a mesh.

We think that the straight logic of the more dynamic the motion is, the more the saliency will be is not enough to determine the saliency measure. Moreover, the eye cannot catch very fast motions causing a decrease in the saliency level in some speeds higher than a threshold depending on many factors like viewpoint, light conditions, eye performance, etc. For years numerous researches have been done in psychology and psychophysics to find out the factors that draw human attention. For a detailed review see [22]. We see that many parameters exist in a correct model of attention and we think it can never be as simple as finding faster moving regions. Thus, we develop an algorithm for detecting the motions that have a harmonic nature. On top of these there are other parameters like color, contrast, viewpoint, etc.

For future work, we are planning to use more sample motions to have a precise definition on what will be the limit for variation and angle change threshold. For that, preparing a user interface that enables users to select the salient regions on an animation would greatly improve our metric because of more learning data. Moreover, the detail capture ability of human visual system is not linear. We cannot capture the details of fast movements. Depending on the viewing distance and the object detail, we can reduce the saliency value if the motion has a higher frequency/speed than a limit. The static saliency information can be used to decide the level of object detail and the parts exceeding threshold can have low saliency exploiting the visual degradation of our eyes. One great improvement to our oscillation detection algorithm would be adding it the capability of finding the oscillating part of the animation. It examines all the frames and finds oscillation in the entire course of animation therefore the cases where several small parts contain different kinds of periodic motion are not handled currently.

Bibliography

[1] MOHR A. and GLEICHER M. Deformation sensitive deformation. Technical report, University of Wisconsin, Madison, 2003.

[2] SHAMIR A., PASCUCCI V., and BAJAJ C. Multiresolution dynamic meshes with arbitrary deformations. In *IEEE Visualization Conference*, pages 423–430, 2000.

[3] SHAMIR A., PASCUCCI V., and BAJAJ C. Temporal and spatial level of detail for dynamic meshes. In *ACM Symposium on Virtual Reality Software and Technology*, pages 77–84, 2001.

[4] WATSON B., FRIEDMAN A., and MCGAFFEY A. Measuring and predicting visual fidelity. In *ACM Computer Graphics (Proceedings of SIGGRAPH '01)*, pages 213–220, New York, NY, USA, 2001.

[5] DECORO C. and TATARCHUK N. Real-time mesh simplification using the GPU. In *ACM Symposium on Interactive 3D Graphics (I3D)*, volume 2007, pages 161–166, 2007.

[6] DECORO C. and RUSINKIEWICZ S. Pose-independent simplification of articulated meshes. In *ACM Symposium on Interactive 3D Graphics*, pages 17–24, 2005.

[7] LUEBKE D. A developers survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[8] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., and WATSON B. *Level of Detail for 3D Graphics.* Morgan Kaufmann, 2002.

[9] LUEBKE P. D. and HALLEN B. Perceptually-driven simplification for interactive rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 223–234, London, UK, 2001. Springer-Verlag.

[10] TAMAL K. D., HERBERT E., SUMANTA G., and DIMITRY V. N. Topology preserving edge contraction. *Publications de L'institut Mathmatique (Beograd) (N.S.)*, 66:23–45, 1999.

[11] SHAFFER E. and GARLAND M. Efficient adaptive simplification of massive meshes. In *IEEE VIS '01: Proceedings of the Conference on Visualization*, pages 127–134, Washington, DC, USA, 2001.

[12] ZHANG E. and TURK G. Visibility-guided simplification. In *IEEE VIS '02: Proceedings of the Conference on Visualization*, pages 267–274, Washington, DC, USA, 2002.

[13] HUANG F., CHEN B., and CHUANG Y. Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 53–62, Aire-la-Ville, Switzerland, 2006.

[14] HOPPE H. Progressive meshes. In *ACM Computer Graphics (Proceedings of SIGGRAPH '96)*, pages 99–108, 1996.

[15] HOPPE H. New quadric metric for simplifying meshes with attributes. In *IEEE VIS '98: Proceedings of the Conference on Visualization*, pages 59–66, 1998.

[16] LEE C. H., VARSHNEY A., and JACOBS D. W. Mesh saliency. *ACM Transactions on Graphics*, 24(3):659–666, 2005.

[17] YAMAUCHI H., SALEEM W., YOSHIZAWA S., KARNI Z., BELYAEV A., and SEIDEL H.-P. Towards stable and salient multi-view representation of 3D shapes. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, page 40, 2006.

[18] BURBEA J. and RAO C. On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory*, 28(3):489–495, 1982.

[19] CLARK H. J. Hierarchical geometric models for visible surface algorithms. *Communications of ACM*, 19(10):547–554, 1976.

[20] HOULE J. and POULIN P. Simplification and real-time smooth transitions of articulated meshes. In *Graphics Interface*, pages 55–60, 2001.

[21] ROSSIGNAC J. and BORREL P. Multi resolution 3D approximations for rendering complex scenes. In *Geometric Modeling in Computer Graphics*, pages 455–465. Springer-Verlag, 1993.

[22] WOLFE J. *Visual Search*, pages 13–73. Psychology Press Ltd., 1998.

[23] ITTI L. and KOCH C. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[24] BOLIN R. M. and MEYER W. G. A perceptually based adaptive sampling algorithm. In *ACM Computer Graphics (Proceedings of SIGGRAPH '98)*, pages 299–309, New York, NY, USA, 1998.

[25] FEIXAS M., SBERT M., and GONZLEZ F. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception*.

[26] GARLAND M. and HECKBERT P. Surface simplification using quadric error metrics. In *ACM Computer Graphics (Proceedings of SIGGRAPH '97)*, pages 209–216, 1997.

[27] GARLAND M. and HECKBERT P. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE VIS '98: Proceedings of the Conference on Visualization*, pages 263–269, 1998.

[28] REDDY M. Perceptually optimized 3D graphics. *IEEE Computer Graphics and Applications*, 21(5):2–9, 2001.

[29] WILLIAMS N., LUEBKE D., COHEN D. J., KELLEY M., and SCHUBERT B. Perceptually guided simplification of lit, textured meshes. In *I3D '03: Proceedings of the ACM Symposium on Interactive 3D Graphics*, pages 113–121, 2003.

[30] CIGNONI P., ROCCHINI C., and SCOPIGNO R. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17:167–174, jun 1998.

[31] LINDSTROM P. Out-of-core simplification of large polygonal models. In *ACM Computer Graphics (Proceedings of SIGGRAPH '00)*, pages 259–262, 2000.

[32] LONGHURST P., DEBATTISTA K., and CHALMERS A. A GPU based saliency map for high-fidelity selective rendering. In *Afrigaph '06: Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 21–29, 2006.

[33] SHILANE P. and FUNKHOUSER T. Distinctive regions of 3D surfaces. *ACM Trans. Graph.*, 26(2):7, 2007.

[34] BROWN R., COOPER L., and PHAM B. Visual attention-based polygon level of detail management. In *GRAPHITE '03: Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 55–ff, 2003.

[35] COOK L. R., HALSTEAD J., PLANCK M., and RYU D. Stochastic simplification of aggregate detail. *ACM Transactions on Graphics*, 26(3):79, 2007.

[36] HOWLETT S., HAMILL J., and O'SULLIVAN C. An experimental approach to predicting saliency for simplified polygonal models. In *APGV '04: Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization*, pages 57–64, 2004.

[37] POPOVIC J. SUMNER R. Deformation transfer for triangle meshes. *ACM Transactions on Computer Graphics*, 23(3):399–405, 2004.

[38] HO T., LIN Y., CHUANG J., PENG C., and CHENG Y. User-assisted mesh simplification. In *VRCIA '06: Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, pages 59–66, 2006.

[39] KANAYA T., TESHIMA Y., KOBORI K., and NISHIO K. A topology-preserving polygonal simplification using vertex clustering. In *GRAPHITE '05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 117–120, 2005.

[40] HONG W. and KAUFMAN A. Feature preserved volume simplification. In *SM '03: Proceedings of the eighth ACM Symposium on Solid Modeling and Applications*, pages 334–339, 2003.

[41] KIM Y. and VARSHNEY A. Persuading visual attention through geometry. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):772–782, 2008.

[42] LIU Y., LIU M., D., and RAMANI K. Salient critical points for meshes. In *SPM '07: Proceedings of the ACM Symposium on Solid and Physical Modeling*, pages 277–282, 2007.

[43] WU Y., HE Y., and CAI H. Qem-based mesh simplification with global geometry features preserved. In *GRAPHITE '04: Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 50–57, 2004.

Appendix A

Mesh Simplification Distance Graphs

The graphs in Figures A1, A2, A3 and A4 show the mean distances between the original and simplified models. For these four graphs saliency metrics are added before simplification. Distance measurements are taken with Metro tool using subdivision sampling [30]. Forward distance is the distance from original model to the simplified one. Backward distance is the distance from simplified model to the original.

The graphs from A.5 to A.12 are created by simplification done by saliency values propagated during the simplification. Since our saliency values are added along with the contractions, they accumulate on the vertex they get contracted with. Therefore some early contractions get heavy in terms of error value causing an imbalanced saliency distribution. For decreasing this effect we reduce the values using simple division so that the accumulation is little from the beginning. For some models we divide the saliency values by 1,000 (Figures A.7, A.8, A.11, A.12) and for Figure A.10 we divided by 10,000.

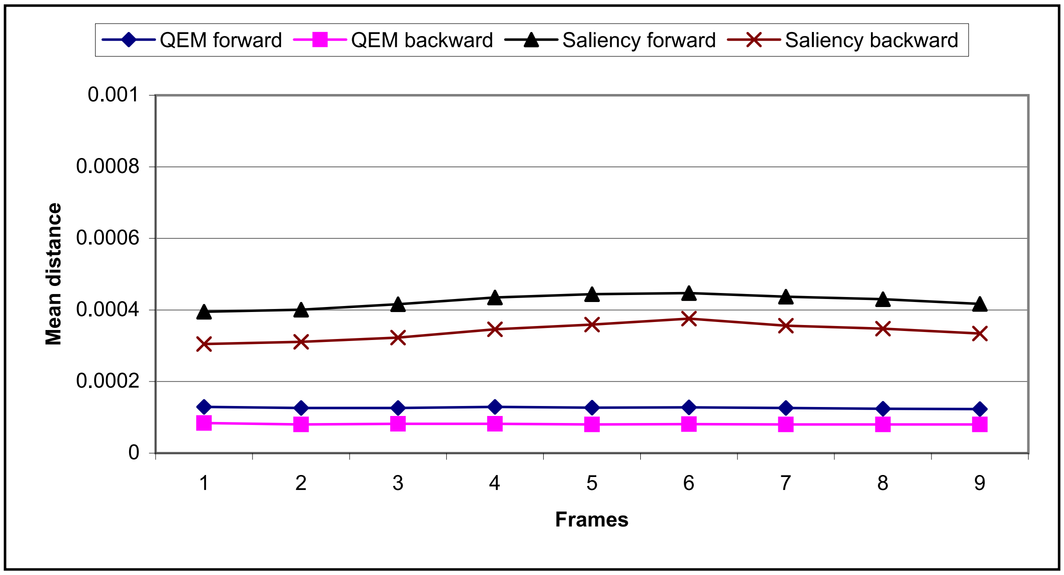

Figure A.1: The mean distance graph for the horse gallop animation simplified to 50%. Here w_s is 3 and w_a is 0.5.

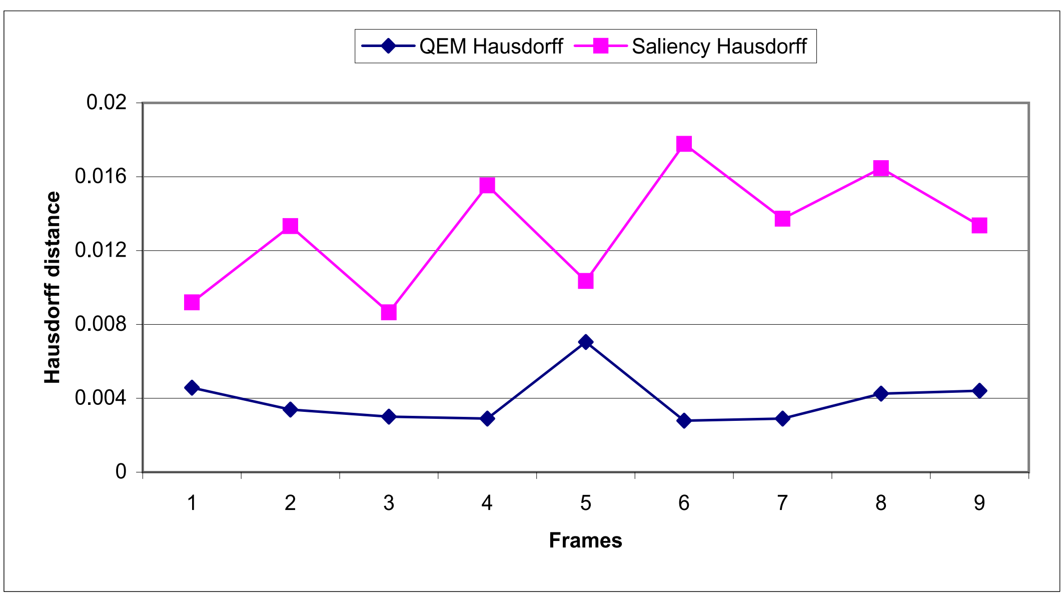

Figure A.2: The Hausdorff distance graph for the horse gallop animation simplified to 50%. Here w_s is 3 and w_a is 0.5.

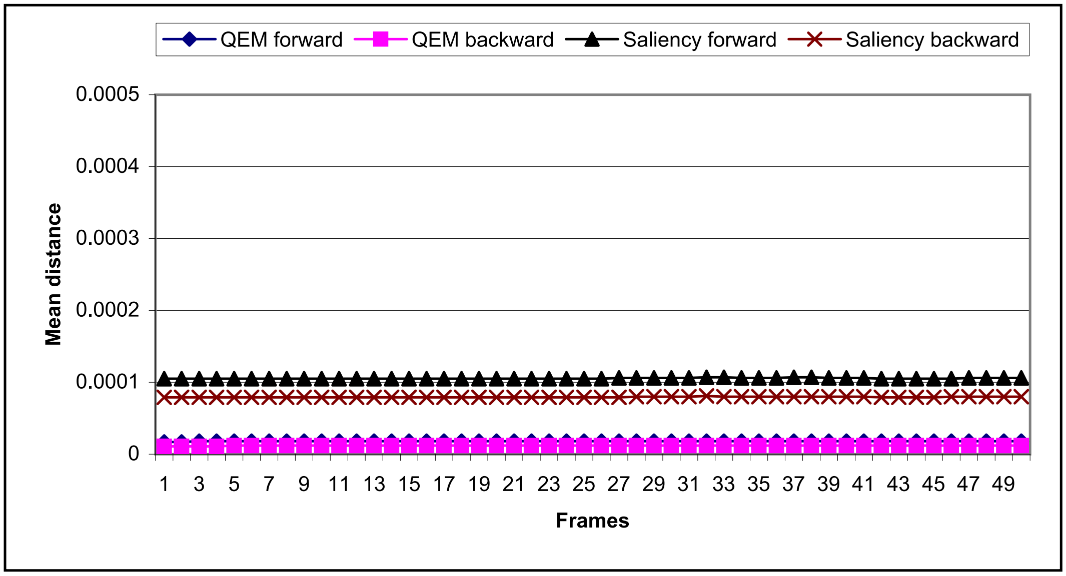

Figure A.3: The mean distance graph for the dance animation simplified to 50%. Here w_s is 3 and w_a is 0.5.

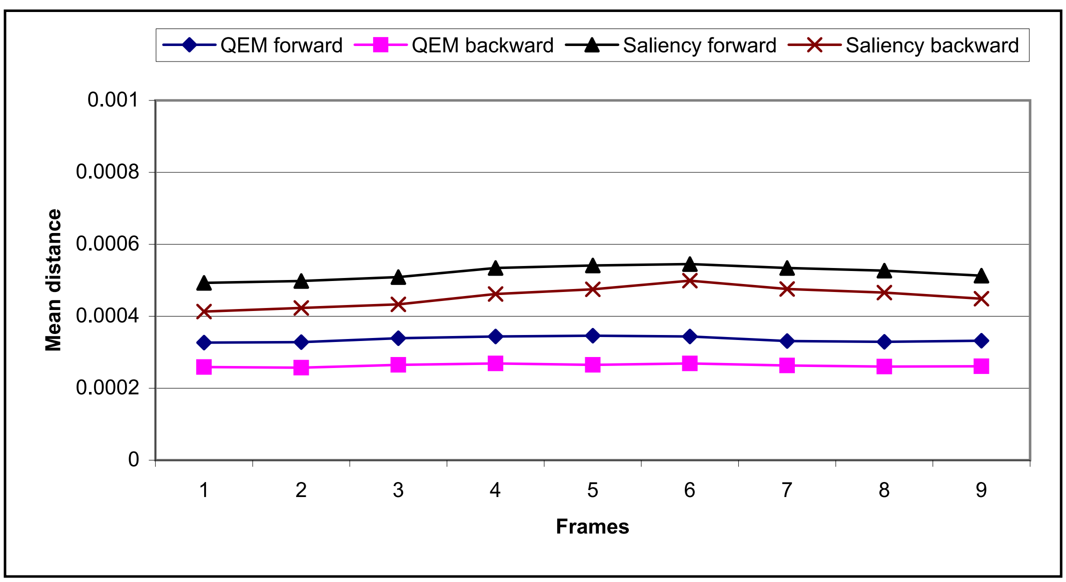

Figure A.4: The mean distance graph for the horse gallop animation simplified to 25%. Here w_s is 3 and w_a is 0.5.

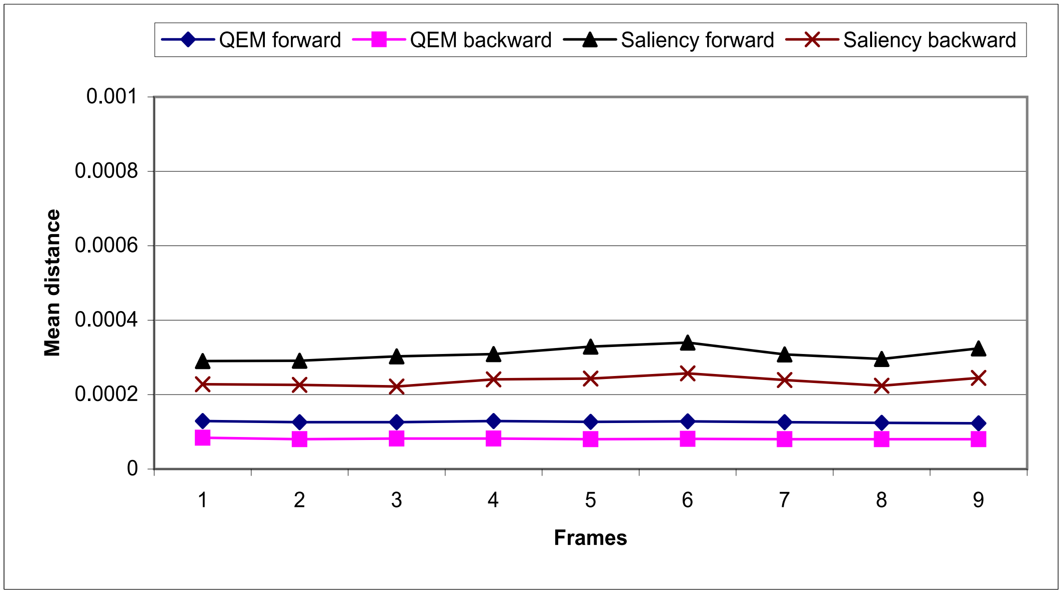

Figure A.5: The mean distance graph for the horse gallop animation simplified to 50%. Here w_s is 0.5 and w_a is 0.5.

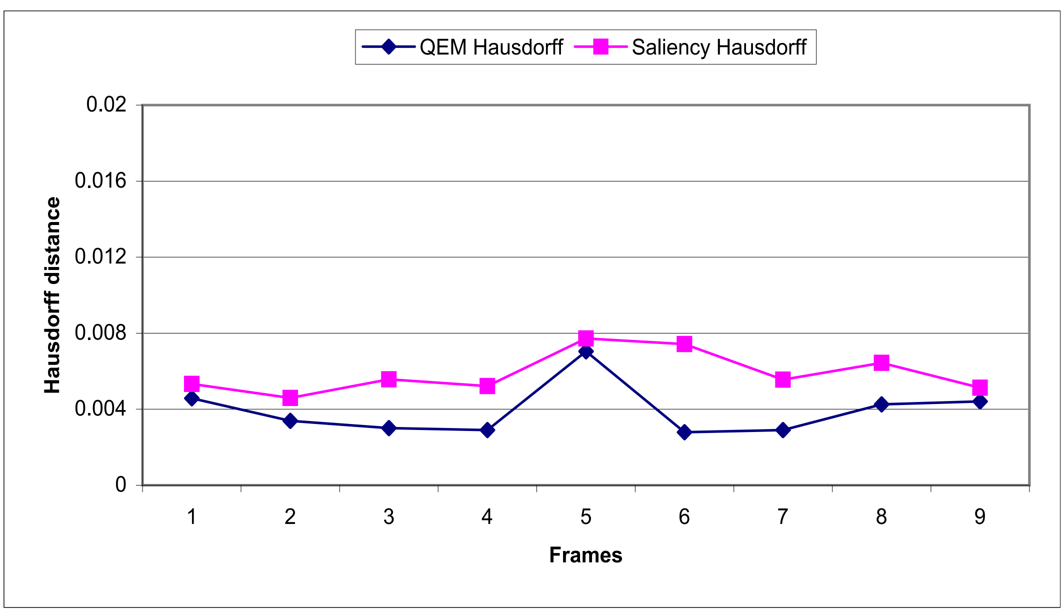

Figure A.6: The Hausdorff distance graph for the horse gallop animation simplified to 50%. Here w_s is 0.5 and w_a is 0.5.

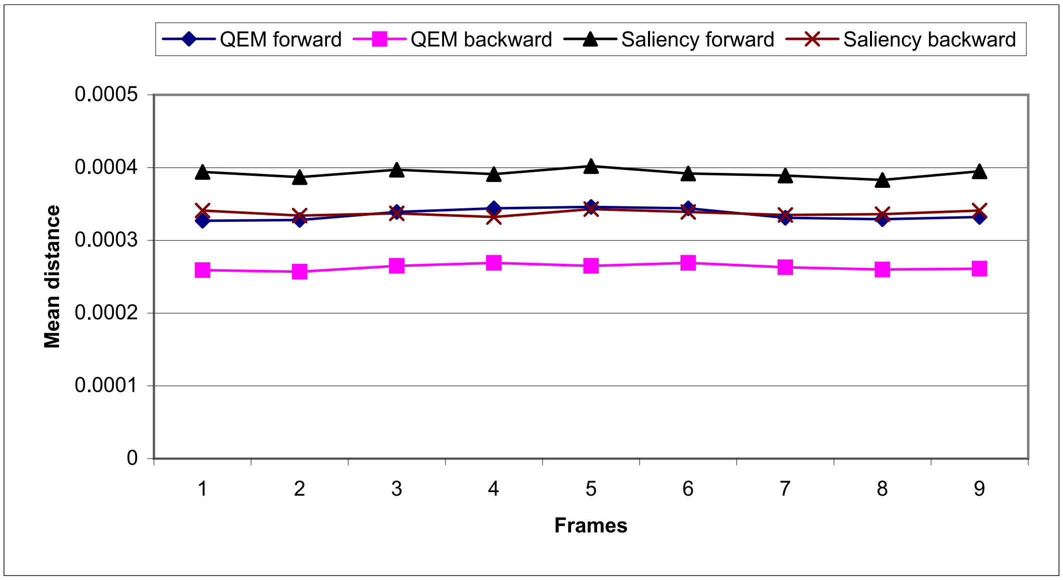

Figure A.7: The mean distance graph for the horse gallop animation simplified to 25%. Here w_s is 0.5 and w_a is 0.5. To prevent imbalanced propagation combined saliency values are divided by 1000.

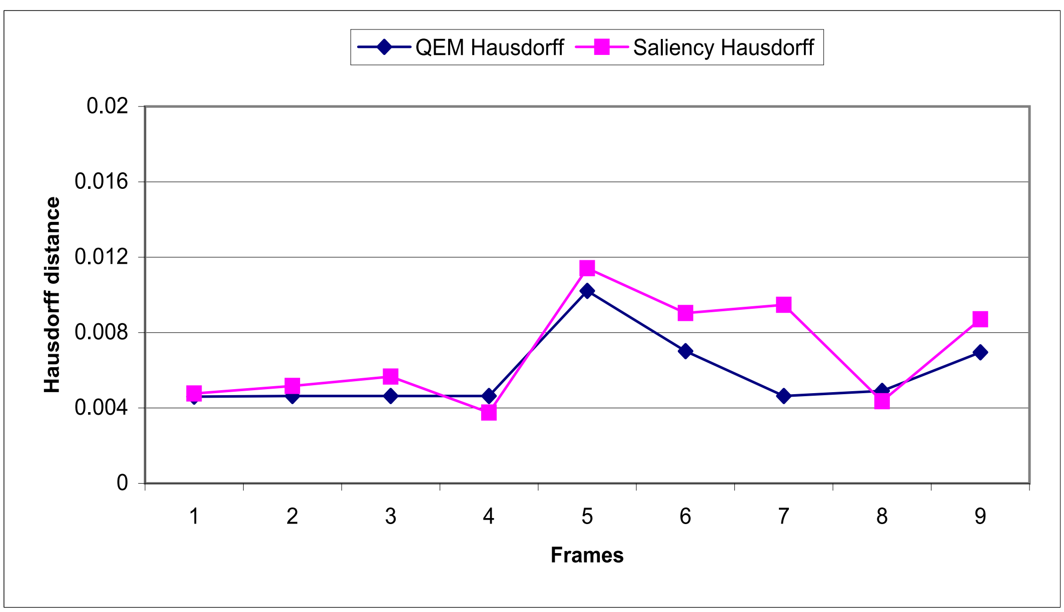

Figure A.8: The Hausdorff distance graph for the horse gallop animation simplified to 25%. Here w_s is 0.5 and w_a is 0.5. To prevent imbalanced propagation combined saliency values are divided by 1000.

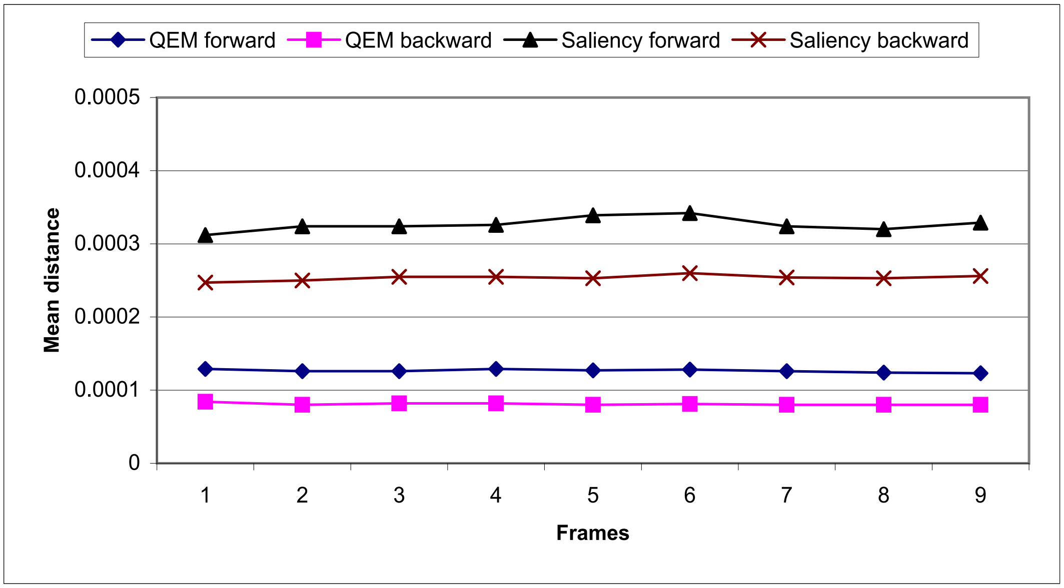

Figure A.9: The mean distance graph for the horse gallop animation simplified to 50%. Here w_s is 2.0 and w_a is 0.1.

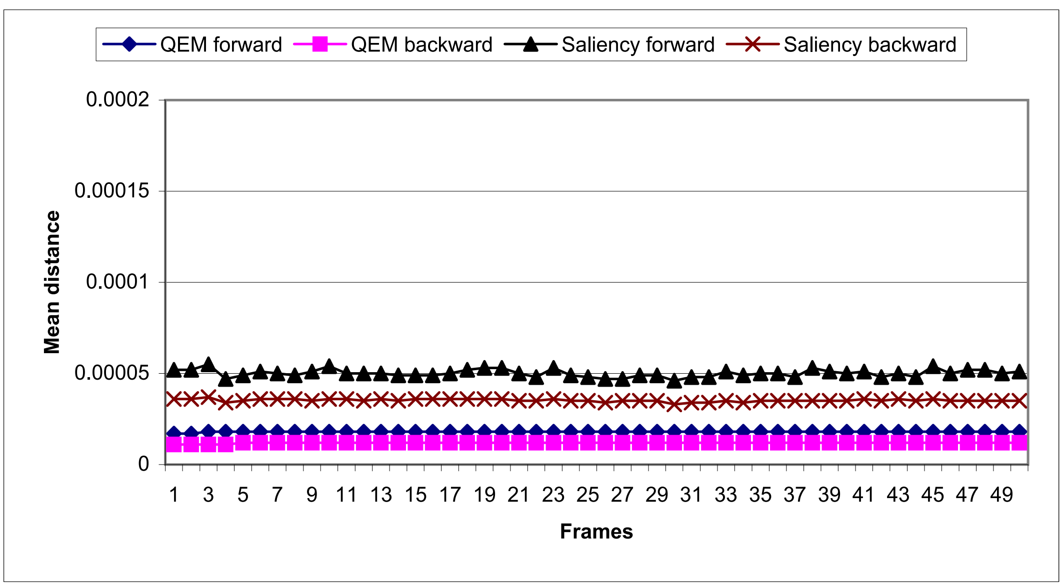

Figure A.10: The mean distance graph for the dance animation simplified to 50%. Here w_s is 0.5 and w_a is 0.5. To prevent imbalanced propagation combined saliency values are divided by 10000.

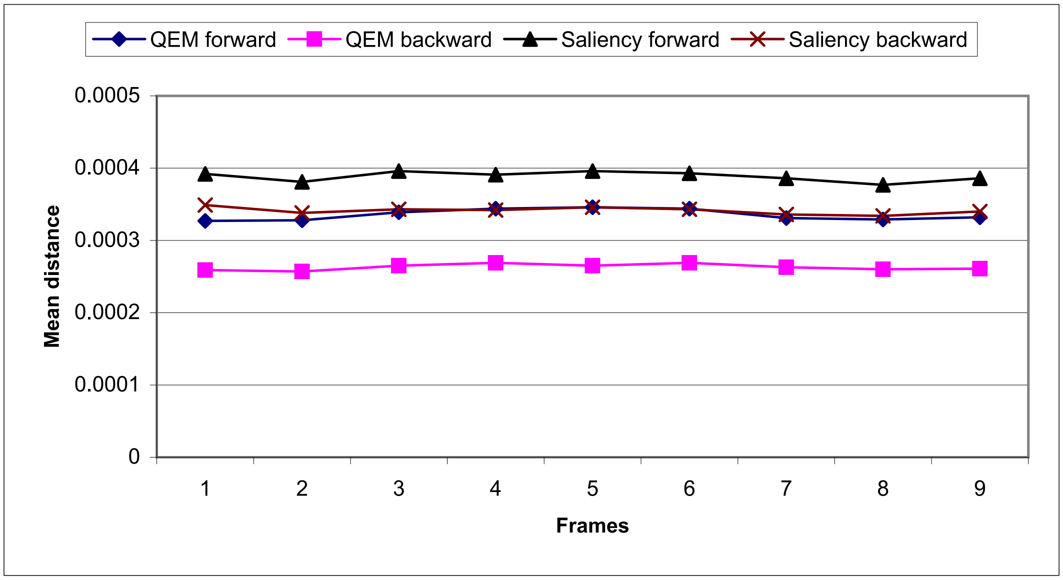

Figure A.11: The mean distance graph for the horse gallop animation simplified to 25%. Here w_s is 0.8 and w_a is 0.2. To prevent imbalanced propagation combined saliency values are divided by 1000.

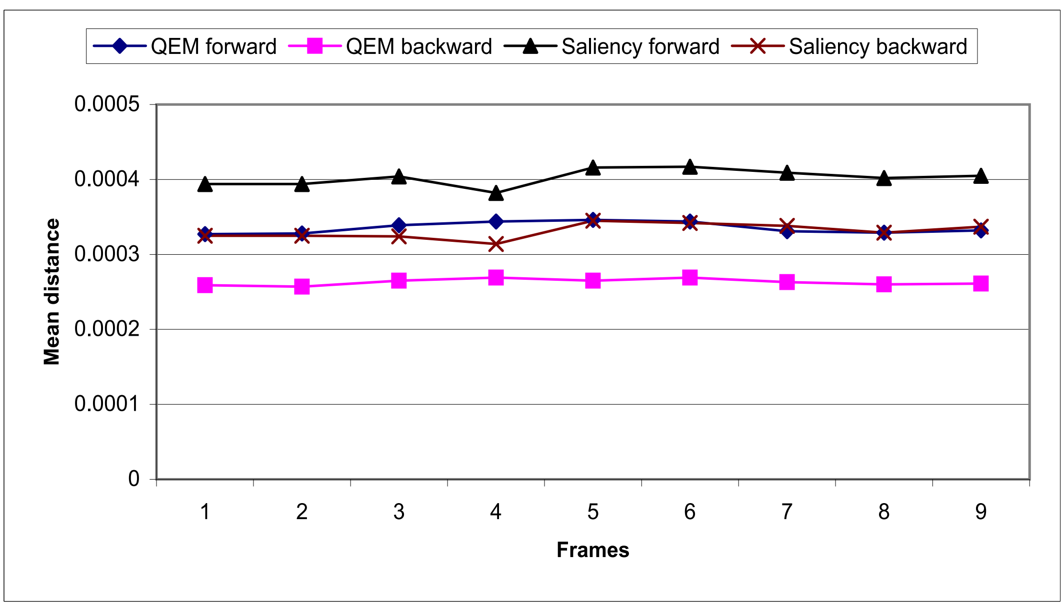

Figure A.12: The mean distance graph for the horse gallop animation simplified to 25%. Here w_s is 0.2 and w_a is 0.8. To prevent imbalanced propagation combined saliency values are divided by 1000.