Adaptive Online Transmission of 3D TexMesh Using Scale-space and Visual Perception Analysis

Irene Cheng, *Student Member, IEEE,* and Pierre Boulanger, *Member, IEEE*

***Abstract* — Efficient online visualization of 3D mesh, mapped with photo realistic texture, is essential for a variety of applications such as museum exhibits and medical images. In these applications synthetic texture or color per vertex loses authenticity and resolution. An image-based view dependent approach requires too much overhead to generate a 360° display for online applications. We propose using a mesh simplification algorithm based on scale-space analysis of the feature point distribution, combined with an associated visual perception analysis of the surface texture, to address the needs of adaptive online transmission of high quality 3D objects. The premise of the proposed textured mesh (TexMesh) simplification, taking the human visual system into consideration, is the following: given limited bandwidth, texture quality in low feature density surfaces can be reduced, without significantly affecting human perception. The advantage of allocating higher bandwidth, and thus higher quality, to dense feature density surfaces, is to improve the overall visual fidelity. Statistics on feature point distribution and their associated texture fragments are gathered during preprocessing. Online transmission is based on these statistics, which can be retrieved in constant time. Using an initial estimated bandwidth, a scaled mesh is first transmitted. Starting from a default texture quality, we apply an efficient Harmonic Time Compensation Algorithm based on the current bandwidth and a time limit, to adaptively adjust the texture quality of the next fragment to be transmitted. Properties of the algorithm are proved. Experimental results show the usefulness of our approach.**

***Index Terms*— Bandwidth adaptation, feature extraction, LOD, perceptual evaluation, scale-space analysis, texture reduction.**

I. INTRODUCTION

EFFICIENT bandwidth utilization and optimal transmission quality are two main objectives of online applications when transmitting 3D models. Our strategy is to transmit the mesh data first, and then achieve best-effort texture quality by adapting to the current bandwidth and a specified time. Although a historic average can be used to estimate current bandwidth [6], this approach can cause unacceptable over or under estimation because of bandwidth fluctuations. An optimal bandwidth monitoring approach provides a more accurate estimation but has to sacrifice a portion of the transmission time [34]. Our solution is to use an adaptive approach, which does not need to sacrifice transmission time for bandwidth estimation, while efficiently adjusting the quality of the fragments not yet transmitted.

In order to optimize bandwidth, a multi-scale incremental simplification approach is most suitable for online applications. Multi-scale allows a coarse version to be refined incrementally. An incremental approach is possible if the set of vertices in a coarse version is a subset of a fine version. If a simplification process involves relocation of mesh vertices or texture coordinates, then an entirely new version, instead of the vertices added to the previous version, has to be transmitted in a refining process [4][13][27][31][16][33]. We apply vertex removal and hole filling without affecting the 3D point and associated texture coordinates.

In recent years, researchers started to incorporate color and texture into their mesh simplification models. When texture is mentioned in the literature, it often refers to per vertex color, synthetic or animated texture [28][30]. Per vertex color limits the texture resolution to that of the mesh. Synthetic texture using simple patterns and small images can speed up interactivity in online applications because patterns can be duplicated or estimated based on available neighboring data. Experimental results show that this technique has better quality and higher compression factor than MPEG [8]. But this technique is not suitable for applications requiring photo-realistic texture, such as the real life texture on museum exhibits. When the viewer expects to see an authentic artifact, mapping synthetic patterns [29] is not applicable. Another approach is to use image-driven simplification methods [21] to display textures using images from multiple view. However, rendering the entire model for every edge in every viewpoint for different scales is expensive, even with hardware-accelerated rendering. The high-resolution texture image mapping technique used in our TexMesh model is different from the per vertex color [11][14][28][29], which requires color interpolation between vertices. For applications requiring real life texture, interpolating colors or estimating texture pattern is not acceptable. The non-interpolated texture, captured together with the geometry during the data acquisition stage, can have resolution much higher than the mesh in our TexMesh model. Reduced versions of photo-realistic texture images, with resolution up to millions of pixels, are suitable for displaying on small monitors or high definition screens in reality centers. Photo-realistic texture is used in the compressed texture maps [35], but their effort is on recovering geometry from texture patches retrieved from multiple photographs. A distance-based technique is applied to photo-textured terrain [20]; however, color interpolation between pixels is necessary to avoid blocky appearance of terrain texture.

The joint geometry/texture progressive coding method [24] applies wavelet transform to encode the mesh and texture data for transmission, but the method is not adaptive to fluctuating bandwidth. Wavelets were also used to create space optimized texture maps, which did not require any on-chip compression

support [2]. In our method, we apply scale-space filtering and zero-crossing detection to extract feature points. Each scale is associated with a default texture quality, and the quality of each fragment is allowed to deviate within a limit based on the number of feature points in it. The quality is later readjusted according to current bandwidth. By relating scale to viewing distance, automatic selection of a corresponding simplified textured mesh is possible. Another advantage of using scale-space analysis is its capability of handling both local and global smoothing, which is controlled by the filter window size and the standard deviation parameter σ. Many simplification techniques are restricted to smoothing in the local neighborhood. For example, the simplification envelopes algorithm [10] is based on the principle that each vertex is displaced within $\pm\varepsilon$ to the extent constrained by surface curvature. Avoidance of self-intersections prevents drastic simplification.

Ever since the introduction of scale-space filtering in 1983 the technique has aroused intense research interest [32], but previous applications focused on image deformation, vortex tracking, noise elimination and segmentation in 2D images [1][2][3][17][19]. As far as we know, scale-space filtering has not been used in the literature to compute feature point distribution, and thereby determine texture quality adaptively. In recent research, we discussed an approach for variable compression of texture based on model complexity [5]; and discussed the relative importance of texture and mesh in human perception [25]. In the current paper we address the issue of adaptive quality adjustment taking into account bandwidth fluctuations within a given time period, and visual perception. The work is based on the observation that texture quality has more impact in regions with high feature density, where visual depth and contrast are crucial. Our technique for detecting small changes *vs.* major variations is based on scale-space filtering. Note that one major difference between wavelets and our scale-space approach is that wavelet scales up or down by a factor of 2; by contrast, in scale-space the scales at which changes occur depends on surface property and is not fixed beforehand.

In this work, we extend our basic textured mesh (TexMesh) model to incorporate the adaptive capability to transmit 3D objects. We use scale-space analysis and zero-crossing detection to extract feature points at different levels of details (LOD). A scale map and a fragment map are generated based on surface property and feature point distribution. The TexMesh model applies a dynamic strategy and adapts to the current bandwidth when computing texture quality of the next transmitted fragment. We apply a Harmonic Time Compensation Algorithm to ensure optimal use of the time limit and bandwidth. By splitting the texture into fragments, distributed transmission and parallel processing is possible.

The remainder of this paper is organized as follows: Section II discusses the impact of geometry and texture resolution on visual fidelity, and why we focus on texture quality given limited bandwidth. Section III gives an overview of the online adaptive strategy. Section IV gives a summary of our basic TexMesh model, explaining how a modified version of the Gaussian filtering and Laplacian can be used to extract feature points, and why the TexMesh model is chosen for bandwidth adaptation. Section V explains how fragment map is generated, how texture quality is assigned to each fragment, and how visual depth on high and low feature density surfaces is perceived by the human visual system. Section VI analyzes the adaptive transmission strategy. Section VII integrates feature extraction, simplification at multiple scales, bandwidth adaptation and visual perception into the TexMesh framework. Section VIII concludes the work and outlines future directions.

II. TRADE-OFF BETWEEN GEOMETRY AND TEXTURE RESOLUTION

Perceptual evaluation experiments were conducted to relate geometry and texture resolution to visual quality of 3D objects. If real life texture is not essential, selecting a suitable texture pattern can hide geometry deficiency and texture artifact caused by quantization — visual masking [12]. But for applications using real life texture, altering the pattern is not an option.

To avoid visual masking, plain texture was used in perceptual experiments so that the judgments are based on the geometry and texture resolution, and is not affected by the complexity of the texture pattern [25]. Experimental result shows that after reaching an optimal mesh density, increasing geometry has little effect on visual fidelity (Fig. 2.1). However, higher texture resolution continued to improve quality (Fig. 2.2).

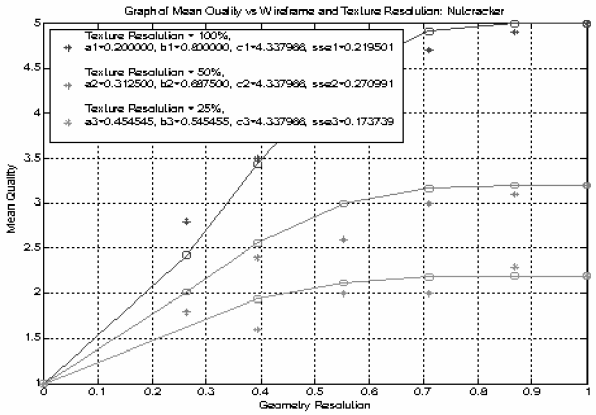

Fig. 2.1: Perceptual experiments show that visual quality relates to geometry resolution exponentially [25].

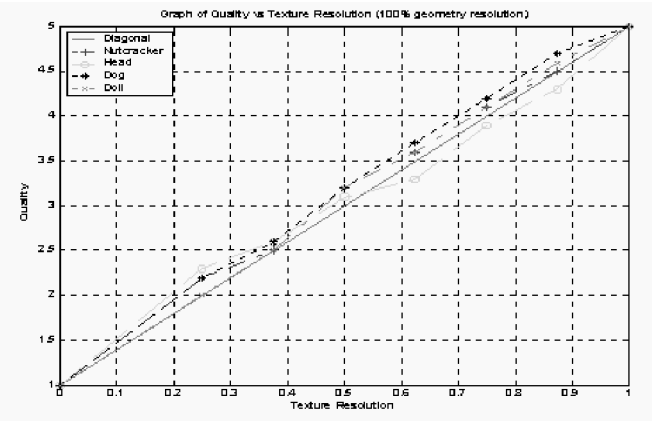

Fig. 2.2: Perceptual experiments show that visual quality relates to texture resolution linearly [25].

High quality texture images impose visual depth to the human visual system through many cues, including linear perspective, texture gradient, shade and contrast. The perception of depth can be induced effectively by high resolution texture images even when mapped onto highly simplified meshes [26]. Given limited network resources, it is therefore beneficial to first transmit the mesh, which is kept small, and adaptively optimize the remaining bandwidth to maximize texture data. The goal is to achieve the best-effort overall visual fidelity.

III. OVERVIEW OF ADAPTIVE STRATEGY

Our strategy for adaptive online transmission of 3D objects has several components, which are shown in Fig. 3.1. The components are divided into two groups: preprocessing is described in the upper group, and the online process is described in the lower group. Feature point distribution is denoted by a value $\in [0,1]$, and is mapped onto a compression scale. For example, in the current implementation, we use the JPEG compression scale [0,100], i.e. $[0,1] \rightarrow [0,100]$. While wavelet coding applies to the entire image and is geometry-independent, our approach supports variable quality determined by the density of surface structures. Note that we use JPEG for convenience and wide support on the web and in JAVA; however, standards such as JPEG2000 can be used as well in the future to code fragments.

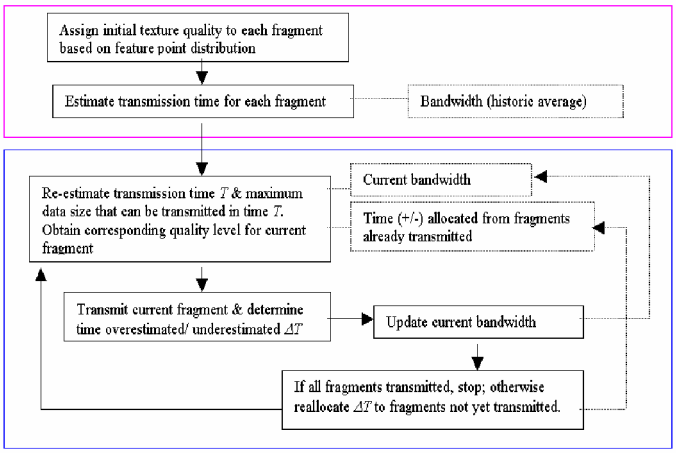

Fig. 3.1: Summary of the adaptive online transmission strategy.

IV. BASIC TEXMESH MODEL

In this paper, we extend our basic model to address the issue of online adaptive transmission taking visual quality into account. A summary of the basic model is given in this section for completeness. Interested readers can refer to [7] for additional details.

A. Level of details (LOD) in scale-space

The Gaussian filter is an efficient smoothing tool in computer vision. However, scale-space filtering had mainly been applied in 2D images and only recently has this technique been used in computer graphics, with limited applications in 3D visualization. We use scale-space filtering (SSF) for adaptive on-line 3D TexMesh simplification and transmission. Traversal between different scales is achieved by varying the standard deviation parameter σ. The higher the value of σ the more is the smoothing. SSF is based on locating the zero-crossings of a signal at multiple scales. Zero-crossings are used to detect the degree of persistence of a structure (feature) in a 3D model. Minor structures tend to diminish as σ increases, and only major structures survive at higher scales. When using a small window size, SSF eliminates signal noise in the local region. By using a bigger window size, the filtering or averaging effect covers a larger surface. Fig. 4.1 is an example of global smoothing using a window size of 201 on a signal of 256 values. To obtain global smoothing, the window size has to be at least twice the standard deviation computed from the sample space (covering at least 97.7% of the sample data in a normal distribution). If a smaller window size is used, smoothing will be restricted and converge before reaching the bottom scale in Fig. 4.1.

Theoretically, 100% global smoothing will end up with a monotonous surface losing all the surface features. When considering the human visual system, this is not necessary because human vision is insensitive to details beyond a certain distance. For a perceivable object, the filter window can be smaller than twice the standard deviation to save computation time. In our experiments, we applied a window size of 1.4 times the standard deviation. We found that this window size provides sufficient simplification for objects placed at a distance close to infinity in the virtual world. Further simplification beyond this point by using a bigger window is not necessary.

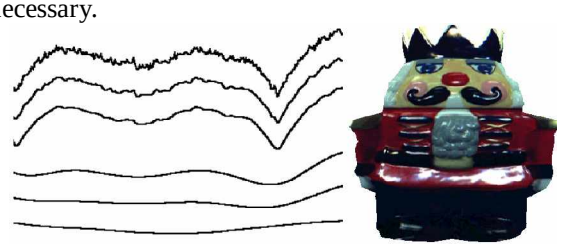

Fig. 4.1: An example of scale-space filtering applied to 256 signal values (left) extracted near the bottom of the Nutcracker model (right). Six scales S_i, increasing from top to bottom, are shown on the left. The original signal S_0 is at the top. Note that local variations (fine details) in the original signal are gradually removed and the scaled signal becomes smoother.

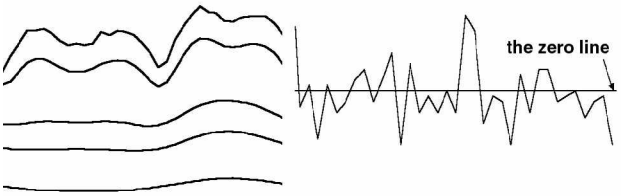

Fig. 4.2: 36 signal values were extracted from the Nutcracker model: (Left) Starting from top to bottom are the original signal S_0 and four scaled signals. They have 18, 8, 6, 4, and 2 zero crossings respectively. (Right) The 18 zero crossings in the original signal S_0 are illustrated as intersections on the horizontal line.

The zero-crossings at different scales can be computed by applying the second derivative of the Gaussian (called Laplacian-of-Gaussian or LoG). 18 feature points are identified in the original signal (Fig. 4.2, right). By increasing σ, the number of feature points decreases from 18 to 2 as reflected by the increasing smoothness of the scaled values (Fig. 4.2, left).

B. Spherical approach on scanned range data

Modern laser scanners detect depths and generate 3D vertices in the form of point clouds. Fig. 4.3a shows a 6-inch dog object. The generated point cloud (Fig. 4.3b) is then triangulated (Fig. 4.3c) and mapped with the scanned texture (Fig. 4.3d) to form the texture mapped 3D model.

We achieve SSF of a 3D model as follows: First note that the data acquired (Fig. 4.3b) can be represented as $R_x(\alpha,y)$; where α is the angle on a horizontal plane around the y-axis of rotation of an object, y is the vertical location, and R_x denotes the distance to the surface of an object for a given (α,y) pair. SSF for a 3D model is thus similar to a 2D image $I(x,y)$, for the simplified mesh representation considered here, with $I(x,y)$ replaced by $R_x(\alpha,y)$. Also, the appropriate scaling along the horizontal and vertical directions can be significantly different, depending on the variance of the sample points for a given region. Thus, SSF in 3D can be summarized by the following equations:

$$w_G(\alpha,y)=\begin{cases}\dfrac{1}{\sigma\sqrt{2\pi}}e^{-\frac{\phi\alpha^2}{2\sigma^2}-\frac{\psi y^2}{2\sigma^2}} & (\alpha,y)\in W\\ 0 & elsewhere\end{cases} \quad (1)$$

$$R_x * S(\alpha,y)=\int_{-t}^{t}\int_{-t}^{t} R_x(\alpha+u,y+v)w(u,v)dudv \quad (2)$$

$$w_{LoG}(\alpha,y)=\begin{cases}-\dfrac{1}{\pi\sigma^4}\left[1-\dfrac{\phi\alpha^2+\psi y^2}{2\sigma^2}\right]e^{-\frac{\phi\alpha^2+\psi y^2}{2\sigma^2}} & (\alpha,y)\in W\\ 0 & elsewhere\end{cases} \quad (3)$$

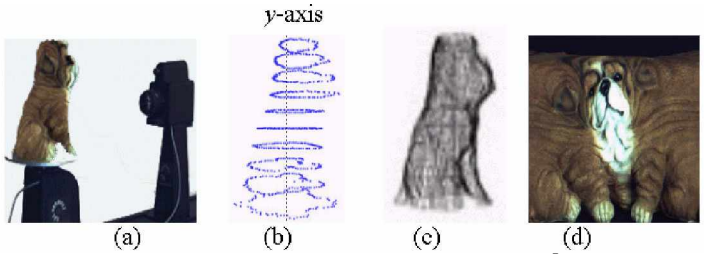

Fig. 4.3: From left to right: Scanning a dog object with Zoomage® 3D scanner, 3D points from scanning, triangulated mesh from 3D points, and scanned texture of the dog object.

Here $w_G(\alpha,y)$ represents the weight at pixel (α,y), R_x*S represents the smoothed image, and w_{LoG} is the Laplacian of Gaussian function. The weights $w(u,v)$ are computed by using a square window W of length $2t+1$. In the discrete case, *e.g.* with a real image, we actually use summation instead of integrals, and normalize the Gaussian weights so that the sum of all the weights equals 1.

For uniform sample points, ϕ and Ψ equal 1, but for irregular sampling, ϕ and Ψ are used to accommodate the variable inter-sample distance. Note that in the actual implementation we use two passes of 1-D filters, since all the filters discussed above are separable. The vertices are first smoothed along the x-axis and then the resulting values are filtered once more along the y-axis. Fig. 4.5 shows the face features, of a head model, change towards a smoother spherical surface when going from lower to higher scales (left to right). The original mesh contains 1,872 vertices and 3,672 faces. The other five meshes are generated at increasing scales by removing less important feature points (Fig. 4.4):

Scale i	# of vertices removed	# of faces in mesh
5	685	2226
7	744	2108
10	824	1948
15	985	1624
20	1196	1190

Fig. 4.4: Five simplified versions of the head model

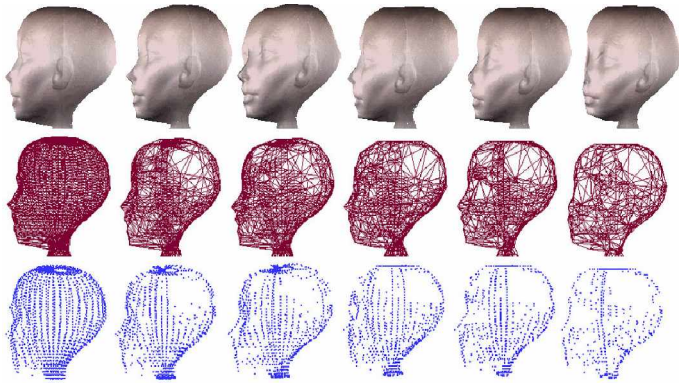

Fig. 4.5: Increasing scales Si from left to right. (Top) 3D mesh with texture, (middle) 3D mesh and (bottom) feature points extracted at scale S_i.

C. Enhancing online transmission and visualization using the TexMesh model

Although many state-of-the-art mesh simplification techniques have been developed in the last decade, levels-of-detail generation for efficient online visualization can be enhanced, by applying scale-space filtering, in the following areas:

1. Reduce geometric data to be transmitted — Earlier techniques often require relocation of vertices during the simplification. Instead of transmitting the inserted vertices, the whole set of vertices representing the entire level needs to be transmitted during refinement. Our scale-space filtering simplification technique does not require vertex relocation. Vertices at a coarse level are a subset of those at a finer level. Only inserted vertices need to be transmitted during refinement.
2. Avoid fragment repartitioning — Since our scale-space simplification technique does not require vertex relocation, feature points associated with texture fragments remain unchanged across different levels-of-detail. By contrast, if a simplification technique requires vertex relocation, feature points have to be redistributed to the appropriate fragments when changing from one level to another.
3. Avoid texture, color and face normal adjustment — Simplification techniques requiring vertex relocation also require texel repositioning, or color and face normal re-computation if the technique employs color per vertex, color interpolation and shading. Our TexMesh model is

designed for 3D mesh mapped with high-resolution real texture. Texel repositioning is not necessary.

4. Enhance visual quality based on human perception — The traditional approach to assess the performance of simplification techniques is based on geometric metrics. In recent years, perceptual metrics have received more attention from researchers [4][11][18][25][26] for two main reasons:
   i) Two geometrically different meshes can appear visually similar, and therefore assessment based solely on geometric metrics may not be accurate.
   ii) The visually fidelity of 3D objects is ultimately determined by human observers, and thus evaluation using perceptual metrics is believed to be more consistent with the human visual system.

   When applying scale-space filtering, 3D surface structures are ranked based on their dimension. Big global structures are inserted before fine smaller details during refinement. This property is consistent with how the human visual system perceives objects moving from far to close.
5. Take care of global as well as local smoothing — Many simplification methods examine local, ignoring global surface property. This may result in removing large noticeable structures from the 3D surface, instead of removing less important small details at an early stage of simplification. For example, techniques merging nearly coplanar neighboring triangles will simplify a surface from (a) to (b), because the faces adjacent to vertex *V* represent a more coplanar surface than the smaller local structures. However, simplifying from (a) to (c) is more consistent with how the human visual system perceives a change from close to far.

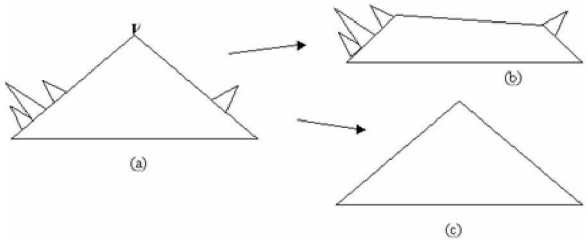

Fig. 4.6: An example of simplification technique considering only local surface curvature, i.e. (a) to (b), and that taking into account of global structures i.e. (a) to (c).

6. Allow drastic simplification — Another advantage of using scale-space filtering is its capability to control the degree of smoothing, by changing a parameter σ. Many simplification techniques restrict drastic simplification. They are good for surface reconstruction, or noise elimination, but too conservative for level-of-detail generation. When an object appears far away from the viewer, much of the detail, including genus and insignificant topology, do not need to be preserved.
7. Suppress mesh data which do not improve visual quality — As explained in item 5 above, feature points representing smaller surface structures are inserted after those representing larger structures, ensuring the global shape of a 3D TexMesh is in place before fine local details. The noticeable difference generated by the inserted structures is decreasing during the refinement process. Taking advantage of this characteristic, progressive refinement can be performed more efficiently by assessing the perceptual impact of the inserted vertices. Only refinement, which has significant visual impact on human perception, is carried out. Mesh refinement is terminated when further vertex insertion will not improve visual quality, releasing bandwidth to other processes.
8. Evaluation based on original surface property avoiding cumulative error — When selecting the next edge or vertex to remove, the simplification criteria used in most methods choose the candidate, which will generate the minimum deviation from the previous simplified surface. For example, in progressive meshes, the minimum energy cost in the neighborhood is affected by an edge collapse operation, and has to be recomputed. The simplified surface then affects the choice of the next collapse operation and the vertices to remove. In our method, the order of vertex removal follows the priority predetermined by applying scale-space filtering on the original 3D surface. The original 3D surface, not the simplified versions, determine the priority of vertices removed or inserted.

Bandwidth fluctuation is a major challenge in online visualization of 3D TexMesh. Since an exact bandwidth is not available before transmission. The amount of data to transmit, and thus the quality of the 3D TexMesh, has to be predicted. Over or under bandwidth estimation often results in inefficient use of limited resources. The goal of our adaptive approach is to absorb the fluctuations periodically by dividing texture data into fragments and altering the quality of the fragments not yet transmitted based on actual bandwidth, instead of cumulating the time deviation until the very end of the transmission.

While absorbing the time surplus (or deficit), by increasing (or decreasing) the fragment data size, the challenge is how to assign different qualities to these fragments so that a best-effort overall TexMesh quality can be maintained given limited resources. Visual quality can be affected by many factors, based on psychophysical experiments conducted in the past. Contrast induced by the surface property is an important visual cue to predict the resulting quality of 3D TexMesh. In order to represent three-dimensional real world objects on a two-dimensional display device, it is essential to impose the perception of depth and contrast on the Human Visual System. A rough surface requires more contrast then a flat surface in order to highlight the surface property. The smoothness of a mesh surface is dictated by its underlying geometry, which can be predicted by the feature point distribution. If the same texture quality is assigned to the entire texture without taking into consideration depth and contrast representation, a plain surface may have excessive quality, leaving insufficient bandwidth to more complex surfaces, and degrading the overall visual fidelity of the 3D object. Feature point distribution based on scale-space analysis is therefore used in our approach to determine the relative quality of texture fragments.

V. VARIABLE TEXTURE QUALITY BASED ON FEATURE POINT DISTRIBUTION

It was suggested that 3D mesh simplification has come to maturity [22]. However, previous simplification methods focused mainly on geometry without integrating real texture, bandwidth and perception in a coherent manner. Many studies emphasized the importance of millions of triangles in order to present fine surfaces, but ignored high resolution real texture which has been shown through user evaluations to have more impact on perceptual quality in 3D visualization [25]. Due to bandwidth fluctuation, adaptive texture quality is essential for online transmission. In our approach, statistics on feature points collected through preprocessing are used for efficient transmission of geometry and texture data given limited network resources.

Feature points are defined as a set of vertices, which can best represent the geometry of a 3D model. In Fig. 4.5, for example, the original head model contains 1872 feature points (scale S_0). After removing 1196 vertices, it is represented by 676 feature points at scale S_{20}. At any scale S_i, feature points are detected by applying LoG. Vertices creating zero crossing are recorded as feature points and assigned the value *i*. Each feature point is then represented by three components: *(i, (tx,ty), (gx,gy,gz))*. The second and third components are the 2D texture and 3D vertex coordinates, respectively. Based on scale-space theory, the number of feature points (structures in the sample space) decreases as scale level increases. This concept best describes how objects are perceived by the human visual system when they move from close to far.

A. Fragment map and geometry-driven texture quality

The texture image of a 3D model can be transmitted as one block or a collection of sub-blocks. The advantage of using sub-blocks is to facilitate distributed transmission and apply variable qualities to different texture regions as explained below. The main concern is whether the additional headers and meta-data will increase the overall volume of data that needs to be transmitted. In this section, we will show that sub-dividing into smaller blocks of optimal dimension does not increase the overall volume for high-resolution texture images. Instead, the sub-block approach helps to fully utilize the available bandwidth.

The texture image of a 3D model is fragmented into N_x*N_y pieces after determining the optimal size of a fragment. To apply JPEG compression efficiently, keeping in mind the size of macroblocks, the optimal dimension of a fragment is chosen as a multiple of 16. The entire texture is also adjusted so that there is no partial fragment. For example, a texture image with dimension 4800*1600 pixels, can be divided into 7,500 fragments of size 32*32 pixels. A fragment map contains fragments arranged in a matrix with N_y rows and N_x columns. Since each 3D vertex is associated with a 2D texel, it is possible to distribute the vertices into the N_x*N_y fragments. We used five texture patterns (Fig. 5.1) to compare the fragmented and non-fragmented sizes for different qualities using the Intel JPEG compression library. Each fragment has a dimension of 16*16 pixels for the 256^2 resolution, and 32*32 pixels for higher resolutions. The initial result showed that the sum of the fragments Σf was less than the size of the single non-fragmented JPEG file F of equal quality, for images of resolution greater than 256^2 pixels, when using 60%, 50%, 40% and 20% qualities. For high resolution images, it is therefore advantageous to transmit individual fragments to the client site before recombining and rendering.

(a) (b) (c) (d) (e)

Texture image	Resolution (pixels)	JPEG Quality	F (KB)	Σf (KB)
(a)	1024^2	60%	566	293
	256^2	20%	29	44
(b)	1024^2	60%	305	222
(c)	1024^2	60%	400	253
(d)	1024^2	60%	494	269
(e)	1024^2	40%	383	226
		50%	423	237
		60%	458	249
	512^2	40%	111	62
		50%	122	66
		60%	130	70
	256^2	20%	25	42
		60%	56	47

Fig. 5.1: Initial experimental results show that sum of sub-blocks of optimal size is less (for high resolution texture) than the size of a corresponding non-fragmented image.

To verify the initial result, five different images of Mount Rushmore at resolution $R = 256^2, 512^2, 1024^2, 2048^2$ and 4096^2 pixels (Fig. 5.2) were used. The resolution is original without interpolation because our analysis on sixteen interpolated texture images showed that the fragmentation approach performed even better if interpolated images are used. Four versions of each Mt. Rushmore image containing number of fragments $n = 2^2, 4^2, 8^2$ and 16^2 respectively were generated. Each version was then tested using quality Q = 100%, 90%, 80%, 60%, 40%, 20% and 10%. The observation is that, when $n \leq 256$ and each fragment size $\geq 128^2$ pixels, the sum of the fragment files Σf was smaller than the single file F of equal JPEG quality. When $\Sigma f < F$ is true at a quality level, it is also true at lower quality levels.

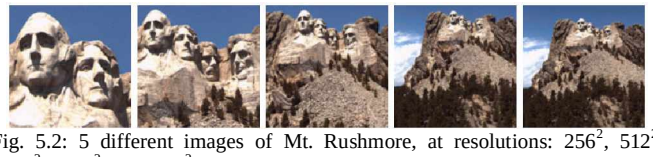

Fig. 5.2: 5 different images of Mt. Rushmore, at resolutions: 256^2, 512^2, 1024^2, 2048^2 and 4096^2 pixels respectively, were analyzed in the experiments.

In Fig. 5.3 and Fig. 5.4, it is observed that at 100% quality when fragment size $\geq$ 128 pixels and the number of fragments $\leq 16^2$, fragmentation does not increase the transmitted data. When increasing the number of fragments beyond 16^2, both the JPEG decoding time and data size are in favor of a single non-fragmented file. When restricting the number of fragments to a maximum of 16^2, the difference in decoding time is within a second for lower resolution images, and is within 2 seconds for images at a resolution of 2048^2 and 4096^2 pixels. This statistics is based on the performance of a Dell desktop computer, using Windows2000 at 1.8 GHz., with 40 GB disk space and 512MB RAM.

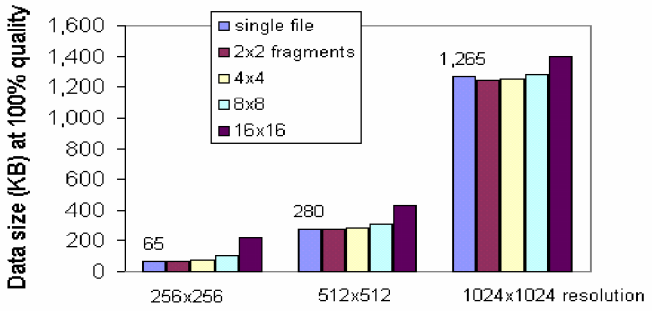

Fig. 5.3: Data size of three Mt. Rushmore images at resolution 256^2, 512^2 and 1024^2 pixels. At each resolution, the single file size (non-fragmented) is compared with the sum of fragment with n = 2^2, 4^2, 8^2 and 16^2 respectively.

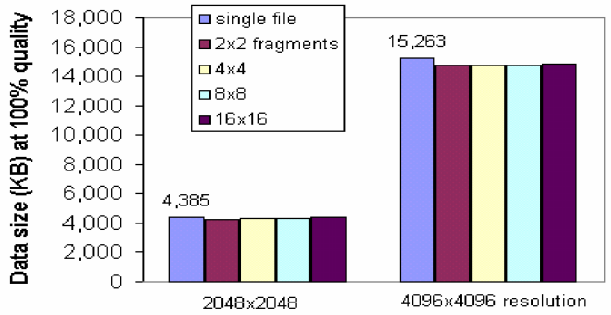

Fig. 5.4: Data size of two Mt. Rushmore images at resolution 2048^2 and 4096^2 pixels. At each resolution, the single file size (non-fragmented) is compared with the sum of fragments with n = 2^2, 4^2, 8^2 and 16^2 respectively.

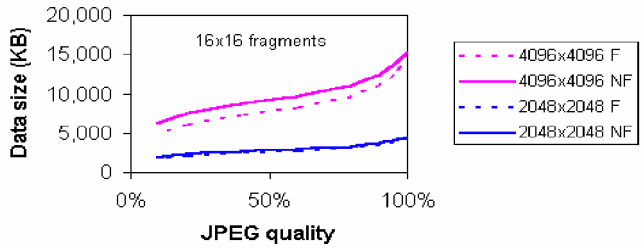

Fig. 5.5: Using images of Mt. Rushmore — To compare size of non-fragmented (NF) and fragmented (F) data at resolutions 4096^2 and 2048^2 respectively. Number of fragments is 16^2.

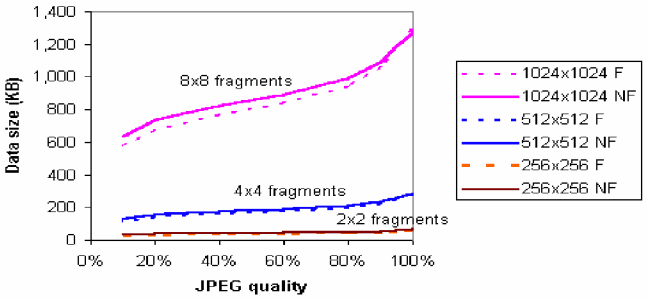

Fig. 5.6: Using images of Mt. Rushmore — To compare the size of non-fragmented (NF) and fragmented (F) data at resolutions 1024^2, 512^2 and 256^2. Number of fragments is 8^2, 4^2 and 2^2 respectively.

We confirmed this finding again by using fifteen more images of different resolutions, and by applying various qualities to each image. The results are consistent. Fig. 5.5 and Fig. 5.6 compare data sizes at different qualities. It can be seen that when dividing into fragments of optimal size, the sum of fragments is not bigger than the single non-fragmented file of equal JPEG quality.

Based on the above finding, we keep the number of fragments n at 16^2 or less when applying our adaptive transmission strategy. Note that there is a tradeoff between using 16^2 fragments or less. The advantage of using fragments is for periodic bandwidth adaptation, so that the time deviation at the end of the transmission can be reduced. For this reason, more fragments are better. However, the JPEG coding/decoding time is proportional to the number of fragments. Based on our experimental analysis, online transmission of high resolution images can benefit from fragmentation if $n \leq 16^2$. For example, when an image of 10.97M pixels is divided into 16^2 fragments, the additional coding/decoding time using the Dell desktop mentioned above is about 5 seconds, which is only 5% of the transmission time if we assume a network speed of 100KB/second. When using 8^2 fragments, the deviation from a given time limit increases from 1% to within 2.5%, but the coding/decoding time is reduced to within 2 seconds. In addition to bandwidth adaptation, fragmentation provides efficient support for multiple data repositories, multiple routes dispatch and parallel processing.

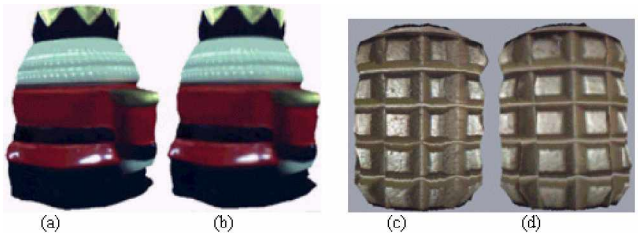

Fig. 5.7: A snap shot of the nutcracker 3D model view from the back (a and b), and the military grenade model (c and d), with original texture quality (a and c), and half of the original texture quality (b and d).

Since the human visual system is less sensitive to details far away, the texture quality Q_i at each scale S_i needs to increase only when i decrease towards a finer model. Given a viewing distance, the corresponding S_i and Q_i are selected. Instead of applying uniform quality to all fragments, a variable approach is used so that texture quality of each fragment (x,y) varies depending on the feature point density associated with it. We use the following grenade and nutcracker models to illustrate how the distribution of feature points affects the perceptual quality of texture. In Fig.5.7c & d, the grenade has square structures on the surface, and therefore the feature point distribution is higher than the back of the nutcracker (Fig.5.7a & b), which is comparatively flat. Note that even if the texture quality is reduced to half, there is no significant perceptual degradation on the nutcracker. However, the grenade on the right (Fig.5.7d) shows noticeably lower perceptual quality. While the difference in quality of the grenades is obvious, the degraded shiny patch under the belt of the nutcracker (Fig.5.7b) is not noticeable. Based on this finding we adopt a variable approach by applying different qualities on texture fragments depending on the feature point distribution, instead of applying a uniform quality to all fragments. Furthermore, the variable qualities are adjusted adaptively based on the current bandwidth. An adaptive approach is necessary when transmitting data on the Internet because bandwidth fluctuations, which can adversely affect the expected quality of service (QoS).

Before explaining how variable qualities are assigned to different fragments, we define the following notation:

S_i – Scale i , *i.e.*, $0 \le i \le n$ where S_0 is the original signal and S_n is the scale at infinity.

ΔQ – Texture quality tolerance limit for each scale controlled by an upper and a lower bound. Analogous to the depth of field in photography, outside which an object is out of focus, ΔQ is the tolerance range when displaying 3D objects at a given distance. Given a viewing distance, the human visual system finds this range of qualities satisfactory.

Q_i – Default texture quality associated with scale S_i.

(x,y) – x and y are the coordinates in the $N_x * N_y$ fragment map.

$q_i(x, y)$ – Texture quality of fragment (x,y) at S_i.

$f_i(x,y)$ – Number of feature points in fragment (x,y) at scale S_i.

$f_i^{\max}$ – The maximum number of feature points in a fragment at scale S_i.

$f_i^{\min}$ – The minimum number of feature points in a fragment at scale S_i.

$\eta_i(x,y)$ – Normalized value of $f_i(x,y)$.

$\bar{\eta}_i$ – The mean of the normalized values $\eta_i(x,y)$ at scale S_i.

Γ – Feature point distribution threshold, *i.e.* $0 \le \Gamma \le 1$, above which $q_i(x, y) > Q_i$, and below which $q_i(x, y) < Q_i$.

$d_i(x,y)$ – Data size of fragment (x,y) at S_i with quality $q_i(x, y)$.

D_i – Data size of all fragments at S_i:

At a given scale i, $f_i(x,y)$ is normalized as:

$$\eta_i(x,y) = \frac{f_i(x,y) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \qquad (4)$$

The texture quality $q_i(x, y)$ of fragment (x,y) at scale S_i is computed as:

$$Qi + (\eta_i(x,y) - \Gamma)\,\Delta Q \qquad (5)$$

In the current implementation, threshold $\Gamma = \bar{\eta}_i \in [0,1]$. Fragments with $\eta_i(x, y) = \Gamma$ are assigned quality Q_i. If $\eta_i(x, y) > \Gamma$, the fragment texture quality is higher than Q_i. If $\eta_i(x, y) < \Gamma$, texture quality is lower than Q_i. ΔQ controls the deviation (+/-) from Q_i constrained by an upper and a lower bound. Regions on the 3D model surface with higher feature point population are displayed with higher quality, and less populated regions are displayed with lower quality texture. By increasing or decreasing Γ, the overall quality of the texture image can be decreased or increased, along with the data size. For each model texture, a lookup table is used to record D_i, $d_i(x,y)$ and $q_i(x, y)$ for a range of Γ. Given a time limit and the current bandwidth, the appropriate D_i, and the associated $d_i(x,y)$ are selected as the reference data size to transmit the first fragment. Details of this adaptive strategy will be discussed in the next section.

B. How feature point distribution relates to visual depth

The sensation of reality in a virtual world occurs because of visual depth perception. One of the many cues for visual depth perception is shade and sharpness [23], which is represented by changing spatial frequency and contrast, on the texture surface. The sensation of changing spatial frequency and contrast varies depending on the brightness adaptation of the human visual system. We first look at how visualization is affected by different levels of brightness, and then examine how the underlying geometry affects spatial frequency and color constancy on texture surface.

Brightness adaptation is good if the Weber ratio $\Delta I/I$ is small. ΔI denotes the increment of illumination discriminated 50% of the time with background illumination I. By plotting the log of Weber's ratio against log I, it is observed that brightness discrimination is poor at low levels of illumination, and improves significantly as background illumination increases [15]. The effect of illumination on vision depends not only on the light source but also the reflectance of the object. Under normal light source, visual discrimination is better on bright rather than dark texture pattern. In our perceptual experiments, we apply the worst scenario and assume visual sensitivity is high. We maintain a reasonable brightness level on the texture so that visual quality between visual stimuli can be easily discriminated. After filtering out the brightness factor, we can focus on how spatial frequency and color constancy of a given texture change depending on the underlying geometry, and discuss how visual quality is affected by these changes.

When discussing spatial interaction in the visual system, it pointed out that the brightness at any pixel depends on the relationships among the intensities in the visual field [9]. In other words, the brightness at each pixel is determined by the intensity distribution in a region. Intensity distribution in a region is induced by the underlying geometry represented by feature point distribution. Using feature point distribution to guide texture reduction is thus consistent with the human visual system. Furthermore, Cornsweet applied a physiological explanation of brightness constancy [9]: sensitivity is close to zero when spatial frequency is null (a uniform or homogeneous field). That is, for two zero frequency fields, with one more intense than the other, the resulting brightness would be almost identical. A flat surface generates a low frequency field, which explains the nearly indistinguishable qualities of the two nutcracker versions (Fig. 5.7a & b). High frequency field exists on surfaces with high feature density because visual depth, shade and contrast are more prominent in these regions. High frequency is more affected by quantization, than low frequency, in the texture reduction process resulting in the more obvious degradation on the grenade (Fig. 5.7c & d). Note that the textures of the nutcracker and grenade have simple texture pattern. If the textures have more complex design, texture masking may hide the degradation or make it less obvious.

Given a texture pattern and a limited bandwidth, we address the question of: which region (with high or low feature distribution) should be chosen to assign a lower quality, in order to optimize bandwidth and maintain satisfactory overall visual fidelity? In our TexMesh model, higher quality is

assigned to higher feature density fragments based on the perceptual analysis given above. Higher qualities in these fragments also provide a cushion effect, to prevent the quality dropping too much below the default quality Q_i during the bandwidth adaptation process, causing significant visual degradation.

VI. ADAPTIVE BANDWIDTH MONITORING AND TEXTURE QUALITY DETERMINATION

Due to fluctuations, current bandwidth has to be monitored periodically in order to maintain a good estimate of the data size that can be transmitted in a specified time T_0. To minimize the overall discrepancy, we reallocate the time surplus/deficit cumulated to the fragments not yet transmitted. The n (N_X*N_Y) fragments are pre-sorted in decreasing $\eta_i(x, y)$ values, *i.e.*, from 1 to 0,

$F_{list} = \{ F_1, ..., \bar{F}, ..., F_n \}$, *i.e.*, $\bar{F}$ has quality Q_i.

The first fragment to be transmitted is $\bar{F}$ with quality Q_i.

Based on a time limit T_0 and a historic bandwidth average β_0, we estimate maximum data size to be transmitted as:

$$D_1 = T_0 * \beta_0 \qquad (6)$$

Where: β_k is the current bandwidth (KB/sec.) recorded after k fragments are transmitted, *i.e.*, $0 \le k < n$. β_0 is the historic average bandwidth before transmission.

T_k is the time left after k fragments are transmitted. T_0 is the original time limit (seconds), and D_{k+1} is the maximum data size that can be transmitted given β_k and T_k.

The fragment list F_{list}, best matching D_1, is selected from the lookup table. Size of $\bar{F}$ is used to estimate the transmission time of the first fragment:

$$est_1 = T_0 * \frac{d_1}{D_1}, \text{ or } est_1 = \frac{d_1}{\beta_0} \qquad (7)$$

Where: d_k represents the data size of the k^{th} fragments, est_k is the estimated time required to transmit fragment k.

We estimate the transmission time est_g for all the remaining fragments, *i.e.*, $2 \le g \le n$: $est_g = T_0 * \frac{d_g}{D_1}$ (8)

After d_1 is transmitted, we have the updated bandwidth β_1 based on the time act_1 recorded when transmitting d_1:

$$\beta_1 = \frac{d_1}{act_1} \qquad (9)$$

Where act_k is the actual time needed to transmit fragment k.

The next fragment is selected as follows: (a) The leftmost fragment in F_{list} if $\beta_1 \le \beta_0$, and

(b) The rightmost fragment in F_{list} if $\beta_1 > \beta_0$.

Let: ΔT_k = Difference between estimated and actual transmission time for k^{th} fragment; *i.e.*, $est_k - act_k$

Δt_k be the cumulated compensating time (+/-) allocated to the k^{th} fragment from the previous k-1 fragments (refer to Algorithm 1 below), and wt_f be the weight applied to the f^{th} fragment when allocating ΔT_{k-1}, *i.e.*, $k \le f \le n$.

(a) If the actual bandwidth is lower than the estimated one, loss of time ΔT_1 has to be compensated when transmitting the remaining n-1 fragments, so that each remaining fragment has to share a portion of ΔT_1. Instead of the initial est_2 computed in Equation (8), the 2^{nd} fragment has $wt_2 * \Delta T_1$ seconds less, where wt_2 is the assigned weight. We regain the time by transmitting the leftmost fragment in F_{list} with reduced quality.

(b) Similarly, if the actual bandwidth is greater than the estimated one, the gained time ΔT_1 is allocated to the remaining n-1 fragments, so that each remaining fragment can have additional time. Instead of the initial est_2, the 2^{nd} fragment has $est_2 + wt_2 * \Delta T_1$ seconds. We adjust the time by transmitting the rightmost fragment in F_{list} with increased quality.

Based on the revised est_2, we compute: $d_2 = \beta_1 * est_2$; and then obtain corresponding quality for the 2^{nd} fragment from the lookup table using d_2. In general, after k-1 fragments are transmitted:

$$\Delta T_{k-1} = est_{k-1} - act_{k-1} \quad (10), \text{ and}$$

$$\Delta t_k = \Delta t_k + \Delta T_{k-1} * wt_k \quad (11)$$

The computation of weight wt_k is explained in Algorithm 1.

$$est_k = est_g + \Delta t_k \quad (12)$$

$$\beta_{k-1} = \frac{d_{k-1}}{act_{k-1}} \quad (13), \text{ and, } d_k = \beta_{k-1} * est_k \quad (14)$$

The quality for the k^{th} fragment is obtained from the lookup table based on d_k.

Bandwidth fluctuation has a larger impact on the quality if ΔT_k has to be shared by a smaller number of fragments. Therefore fragments with quality $\cong Q_i$ are transmitted last to allow more flexibility for adjustment within the control limit ΔQ. Once the transmission is started, the quality of a fragment is self-adjusted depending on the updated bandwidth.

A. *Harmonic Time Compensation Algorithm*

Since later fragments have to share all preceding allocations, Algorithm 1 assigns decreasing weights $(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, ...)$ to fragments $(k+1)^{th}$ to n^{th}, when reallocating ΔT_k.

Algorithm 1 – Harmonic Time Compensation

After transmitting k^{th} fragment,

div=2;

$$\varsigma_k = \sum_{j=2}^{n-k+1} \frac{1}{j} \approx \ln(n-k+1);$$

for ($i = k+1;\ i \le n;\ i++$) {

$$wt_i = \frac{1}{div * \varsigma_k} \ ;$$

$\Delta t_i += \Delta T_k * wt_i$; // Allocate to remaining fragments

div^{++};

}

There are two questions we have to address:

(1) How efficient is the algorithm with respect to bandwidth optimization in a given time?

(2) How does the adaptive approach affect the perceptual quality?

To prove the efficiency of the algorithm, we define Π as the time surplus (or deficiency) with respect to the limit T_0. Π is the result of three errors: estimation error E_{est}, allocation error E_{alloc}, and compensation error E_{comp}. In Theorem 1 we establish the upper and lower bound of Π (Proof: See Appendix A).

Theorem 1: Π is bounded by: $\Delta T_n + (\Delta T_{n-1}/2) + (1.088 + \ln|\ln(n)|)\Lambda$

Where Λ is defined as the average difference between the estimated and actual transmission time for n-1 fragments, *i.e.*, $\Lambda = \frac{\sum_{j=1}^{n-1} \Delta T_j}{n-1}$.

The upper and lower bounds in Theorem 1 are verified by experimental results in the next section. We will show that our adaptive approach does not have an adverse effect on perceptual quality for reasonable bandwidth fluctuations.

B. *Experimental results*

Let n = 256. Applying Theorem 1, we obtain:

$$E_{comp} \leq |2.8\Lambda|$$

Since Λ is the average deviation over the entire transmission period, it is expected to be small. The other two components of Π: estimation error E_{est} (ΔT_n) and allocation error E_{alloc} ($\Delta T_{n-1}/2$), can be minimized by using sufficiently small data size for the last two fragments.

In order to see how Π responds to bandwidth fluctuation, we implemented a bandwidth monitor, and extracted three bandwidth sample sets from an Ethernet connection on different days and different times (Fig. 6.1). We then vary the value of β_0 below and above the average of the sample set within a reasonable range. The test file is 418KB with 256 fragments.

Bandwidth sample set	Actual bandwidth avg. (KB/sec)
1	41.68
2	45.64
3	42.07

Fig. 6.1: Bandwidth samples recorded from an Ethernet connection

From Fig. 6.2 one can observe that Π is minimum, when Λ is close to zero. Similar trends were obtained from Samples 2 and 3. By keeping the n^{th} and $(n\text{-}1)^{th}$ fragments sufficiently small, as in our experiments, the deviation from the time limit is within 1%. By comparison, the last column shows the discrepancy in percentage of time limit, should historic average bandwidth be used in a static approach.

β_0 (KB/sec)	Λ (sec)	Surplus/deficit (+/-)% of limit	Time Limit (sec)	Static %
20	0.038	0.399	20.9	52.0
23	0.031	0.346	18.17	44.8
26	0.023	0.280	16.07	37.6
29	0.015	0.195	14.41	30.4
32	0.011	0.142	13.06	23.2
35	0.003	0.010	11.94	16.0
38	-0.001	0.010	11	8.8
41	-0.004	-0.027	10.19	1.6
44	-0.008	-0.034	9.5	-5.5
47	-0.008	-0.058	8.89	-12.0
50	-0.012	-0.058	8.36	-19.8
53	-0.012	-0.090	7.88	-27.2

Fig. 6.2: Experimental results show that the Harmonic Time Compensation Algorithm had less than 1% deviation for a given time limit. β_0 was used as the initial bandwidth estimation (Sample set 1).

To see how variable quality affects the overall visualization, we used β_0 = 32 and 50, together with bandwidth sample 1 of average 41.68 (KB/sec), and applied to the dog texture. The original texture has quality Q_i = 80% and ΔQ_i is [40%, 100%]. Fig. 6.3 shows that the perceptual quality is maintained, after applying variable qualities to fragments adaptively in order to satisfy the time limit. Given the estimated texture in the middle, actual quality is increased in case of underestimation of actual bandwidth (left), and actual quality is decreased for overestimation (right).

Fig. 6.3: (Middle) Initial estimated texture (middle), (left) increased quality resulted from underestimation, and (right) decreased quality resulted from overestimation of bandwidth.

We performed fifteen more simulations using a wider range of bandwidths, including dial-up and high-speed networks. Experimental results are consistent with earlier findings. Results from seven higher speed networks are plotted in Fig. 6.4 and five from lower speed networks are plotted in Fig. 6.5. It can be seen that, when the initial estimation is within a reasonable range of the bandwidth average, the overall deviation is less than 1% of the given time limit. The deviation gets smaller when the initial estimation β_0 approaches the actual average bandwidth of the entire transmission period. Since the initial estimation is based on historic average, it is not expected to be very different from the actual average for a reasonably long transmission period. Also, note that when networks have similar average and if a network is relatively stable, i.e. 3.31 KB/sec. as shown by (a), (b), (c) and (d) in Fig. 6.5, the surplus/deficit curves follow a similar trend. On

the other hand, although the bandwidth average of 111.94 and 110.89 KB/sec are close (Fig. 6.4), the surplus/deficit curves have different trends due to unstable bandwidth fluctuations.

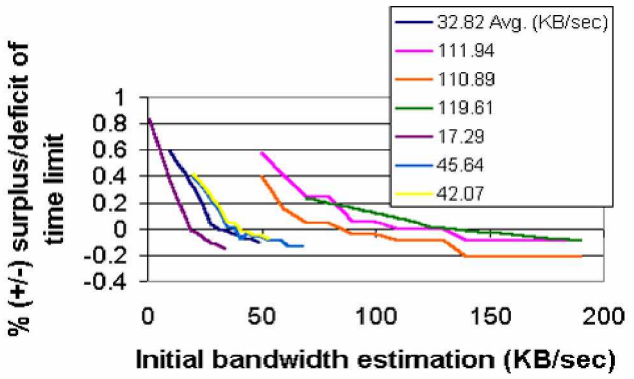

Fig. 6.4: Simulation results of seven higher speed networks — By applying different initial estimated bandwidth B_0, the final deviation from a given time limit is with 1%.

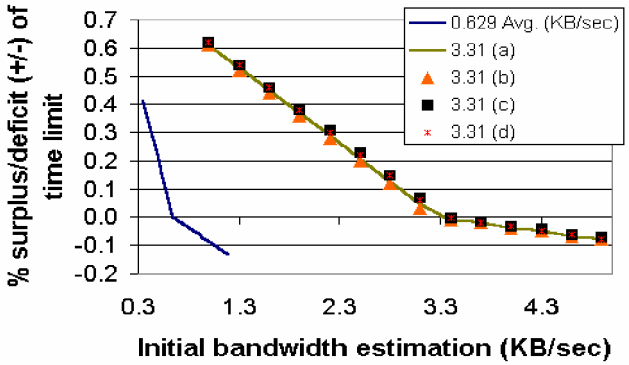

Fig. 6.5: Simulation results of five lower speed networks — By applying different initial estimated bandwidth B_0, the final deviation from a given time limit is with 1%.

VII. INTEGRATING FEATURE EXTRACTION, SIMPLIFICATION AT MULTIPLE SCALES, BANDWIDTH ADAPTATION AND PERCEPTUAL QUALITY INTO THE TEXMESH FRAMEWORK

On-line 3D visualization is an expanding area of multimedia research covering graphics, imaging, visual perception and network transmission. Adaptation to available bandwidth to achieve satisfactory transmission time and visual quality is one of the main objectives in 3D simplification (both mesh and texture). We use SSF analysis to integrate texture reduction with mesh simplification. The approach can be summarized as follows:

Preprocessing

(a) Perform a SSF analysis of the 3D object and identify regions of strong persistent structures *vs.* regions of small surface variations, at different scales. Generate a priority list of feature points from strong to weak persistence.
(b) Divide the model texture into fragments of optimal size. Generate a fragment map for each scale S_i by distributing the feature points onto corresponding texture fragment.
(c) For each scale S_i use a lookup table to record the individual and total fragment size, and associated qualities for a range of Γ.

Runtime processing

(a) Based on the viewing distance, obtain the corresponding scale S_i. Use the initial bandwidth (historic average) B_0, and T_0 to locate the closest matching set of texture fragments in the lookup table. T_0 is the given time limit minus the time required to transmit the geometric data (mesh).
(b) Compute the texture quality of each fragment using the adaptive approach and transmit the texture fragments.
(c) Client site recombines fragments and renders the texture-mapped 3D model.

VIII. CONCLUSION AND FUTURE WORK

The adaptive TexMesh model proposed in this paper applies scale-space analysis to extract feature points at different scales. Quality of a texture fragment is determined by the underlying geometry, and the current bandwidth in a given time period. We apply LoG to detect zero crossings at each scale and generate statistics during preprocessing. These statistics are then used during runtime for efficient extraction and transmission of texture data. Perceptual quality is evaluated implicitly based on the feature point distribution. Experimental results show that this adaptive approach utilizes bandwidth more efficiently, and thus provides better control on QoS for online transmission of 3D objects.

In the current implementation, we use the Intel JPEG Library. We plan to combine the advantages of both the wavelet and TexMesh approaches to develop a better coding technique for transmitting textured meshes. In future work, we will examine further into the relationship between visual fidelity and surface property, and perform experiments with more 3D models. In addition to feature point distribution, we will also look into other visual discrimination predictors, and incorporate them into our TexMesh framework.

References

[1] D. Bauer and R. Peikert, “Vortex tracking in scale-space,” Eurographics-IEEE TCVG Symposium on Visualization, 2002.
[2] L. Balmelli, G. Taubin and F. Bernardini, “Space-optimized texture maps,” Eurographics, 2002.
[3] P. Boulanger, O. Jokinen and A. Beraldin, “Intrinsic filtering of range images using a physically based noise model,” VI 2002, Calgary, Canada.
[4] D. Brodsky and B. Watson, “Model simplification through refinement”, Proc. of Graphics Interface 2000.
[5] I. Cheng, “Efficient 3D Object Simplification and Fragmented Texture Scaling for Online Visualization,” IEEE International Conference on Multimedia, 2003.
[6] I. Cheng, A. Basu, Y. Zhang and S. Tripathi, "QoS Specification and Adaptive Bandwidth Monitoring for Multimedia delivery," Proc. IEEE EUROCON, 2001.
[7] I. Cheng and P. Boulanger, “Scale-Space 3D TexMesh Simplification,” IEEE Int'l Conference on Multimedia, 2004.
[8] D. Cohen-Or, Y. Mann, S. Fleishman, “Deep Compression for Streaming Texture Intensive Animations”, Siggraph, August 8-13, 1999.
[9] T. Cornsweet, “Visual Perception”, Stanford Research Institute, Academic Press Inc. 1970
[10] J. Cohen, A. Varshney, D. Manocha, G. Turk and H. Weber, “Simplification Envelopes”, Proc. Siggraph 1996.
[11] J. Cohen, M. Olano and D. Manocha, “Appearance-Preserving Simplification”, Siggraph 1998.
[12] J. Ferwerda and S. Pattanaik, “A model of visual masking for computer graphics,” ACM SIGGRAPH 1997, pp. 143-152
[13] M. Garland and P. Heckbert, “Surface Simplification sing quadric error metrics”, Siggraph 1997 p209-216.
[14] M. Garland and P. Heckbert, “Simplifying Surfaces with Color and Texture using Quadric Error Metrics”, IEEE Visualization 1998.

[15] R. Gonzalez and R. Woods, "Digital Image Processing," second edition, 2002 Prentice Hall
[16] H. Hoppe, "Progressive meshes" Proceedings of SIGGRAPH 1996, L.A.
[17] A. Kuijper and L. Florack, "Logical filtering in scale space", Institute of Information and Computing Sciences, Utrecht University, Technical Report, 2001.
[18] D. Luebke and B. Hallen, "Perceptually driven simplification for interactive rendering", 12th Eurographics Workshop on Rendering Techniques, London, UK 2001.
[19] T. Lindberg, "A scale selection principle for estimating image deformations," ICCV, Cambridge, MA, 1995.
[20] P. Lindstrom, D. Koller, L. Hodges, W. Ribarsky, N. Faust and G. Turner, "Level of Detail Management for Real-Time Rendering of Phototextured Terrain", TR-95-06, Graphics, GeorgiaTech, GA.
[21] P. Lindstrom and G. Turk, "Image-driven simplification", ACM Transaction On Graphics, 2000.
[22] D. Luebke, "A Developer's Survey of Polygonal Simplification Algorithms", IEEE CGA, May/June 2001.
[23] S. Nagata, "How to reinforce perception of depth in single two-dimensional pictures," Proc. Of the SID, Vol. 25/3, 1984
[24] M. Okuda and T. Chen, "Joint Geometry/Texture Progressive Coding of 3D Models", IEEE Int'l Conf. On Image Processing, Vancouver, Sep. 2000.
[25] Y. Pan, I. Cheng, and A. Basu, "Perceptual quality metric for qualitative 3D scene evaluation," IEEE Transactions on Multimedia, April 2005, Vol.7, No.2.
[26] H. Rushmeier, B. Rogowitz and C. Piatko, "Perceptual issues in substituting texture for geometry," Proceeding of SPIE 2000 Vol.3935, p372-383.
[27] E. Shaffer and M. Garland, "Efficient Adaptive Simplification of Massive Meshes", IEEE Visualization 2001.
[28] M. Soucy, G. Godin and M. Rioux, "A texture-mapping approach for the compression of colored 3D triangulations", The Visual Computer (1996) 12: 503-514.
[29] P. Sander, J. Snyder, S. Gortler and H. Hoppe, "Texture mapping progressive meshes", Siggraph 2001.
[30] G Turk, "Generating texture on arbitrary surfaces using reaction-diffusion", Siggraph, July 1991.
[31] G Turk, "Re-tiling polygonal surfaces", Siggraph, 1992.
[32] A. Witkin, "Scale-space filtering," International Joint Conference on AI, 1983, pp. 1019-1022.
[33] J. Xia, J. El-Sana and A. Varshney, "Adaptive Real-Time Level-of-detail-based Rendering for Polygonal Models" IEEE Trans. on Visualization and CG, June 1997.
[34] Y. Yu, I. Cheng and A. Basu, "Optimal adaptive bandwidth monitoring," IEEE Trans. on Multimedia, September, 2003.
[35] Y. Yu, A. Ferencz and J. Malik, "Compressing Texture Maps for Large Real Environments", Siggraph 2000 Sketch.

Appendix A - Proof of Theorm 1:

The time deviation ΔT_k caused by the k^{th} fragment can be expressed as $\Lambda + \varepsilon_k$ where Λ is the average time deviation. Let ς_k be $\ln(n-k+1)$, as defined in Algorithm 1. After 1^{st} fragment is transmitted, ΔT_1 is allocated to the remaining n-1 fragments as follows:

$\Delta T_1 = \Delta T_1 * (1/(2*\varsigma_1) + 1/(3*\varsigma_1) + \ldots + 1/(n*\varsigma_1))$

After 2^{nd} fragment, ΔT_2 is allocated to the remaining n-2 fragments as follows:

$\Delta T_2 = \Delta T_2 * (1/(2*\varsigma_2) + 1/(3*\varsigma_2) + \ldots + 1/((n-2)*\varsigma_2) + 1/((n-1)*\varsigma_2))$

…..

In the last two allocations,

$\Delta T_{n-2} = \Delta T_{n-2} * (1/(2*\varsigma_{n-2}) + 1/(3*\varsigma_{n-2})$; allocated to $(n-1)^{th}$ and n^{th} fragments,

$\Delta T_{n-1} = \Delta T_{n-1} * (1/(2*\varsigma_{n-1}) + 1/(2*\varsigma_{n-1})$; allocated to n^{th} fragment.

Since there is no other fragment after n, the n^{th} fragment has to share 100% of ΔT_{n-1}. est_n is revised by adding the cumulated compensating time Δt_n applied to the n^{th} fragment, and we compute Π, the overall surplus/deficit, as follows:

$\Pi = est_n + \Delta t_n - act_n$

$= d_n / \beta_0 + (\Delta T_1/(n*\varsigma_1) + \Delta T_2/((n-1)*\varsigma_2) + \ldots + \Delta T_{n-2}/(3*\varsigma_{n-2}) + \Delta T_{n-1}) - d_n / \beta_n$

$= (d_n / \beta_0 - d_n/\beta_n) + (\Delta T_1/(n*\varsigma_1) + \Delta T_2/((n-1)*\varsigma_2) + \ldots + \Delta T_{n-2}/(3*\varsigma_{n-2}) + \Delta T_{n-1}/(2*\varsigma_{n-1})) + (\Delta T_{n-1}/2)$

We define $\Pi = E_{est} + E_{comp} + E_{alloc}$

E_{est} – Estimation Error $\Delta T_n = (d_n / \beta_0 - d_n/\beta_n)$ caused by the discrepancy between the historic average and the actual bandwidth for n^{th} fragment. Note that the 1^{st} d_n is before adjustment of Δt_n, and the 2^{nd} is after.

E_{comp} – Compensation Error $\Delta t_n = (\Delta T_1/(n*\varsigma_1) + \Delta T_2/((n-1)*\varsigma_2) + \ldots + \Delta T_{n-2}/(3*\varsigma_{n-2}) + \Delta T_{n-1}/(2*\varsigma_{n-1}))$ allocated from fragments 1 to n-1, shared by the n^{th} fragment.

E_{alloc} – Allocation Error $(\Delta T_{n-1}/2) =$ $(d_{n-1} / \beta_0 - d_{n-1}/\beta_{n-1})/2$, incapable of allocating further.

E_{comp} can be further analyzed by splitting ΔT_i into ε_i and Λ:

$E_{comp} = (\varepsilon_1/(n*\varsigma_1) + \varepsilon_2/((n-1)*\varsigma_2) + \ldots + \varepsilon_{n-2}/(3*\varsigma_{n-2}) + \varepsilon_{n-1}/(2*\varsigma_{n-1})) + \Lambda*(1/(n*\varsigma_1) + 1/((n-1)*\varsigma_2) + \ldots + 1/(3*\varsigma_{n-2}) + 1/(2*\varsigma_{n-1}))$

$\leq (\varepsilon_1/(2*\varsigma_{n-1}) + \varepsilon_2/(2*\varsigma_{n-1}) + \ldots + \varepsilon_{n-2}/(2*\varsigma_{n-1}) + \varepsilon_{n-1}/(2*\varsigma_{n-1})) + \Lambda*(1/(n*\varsigma_1) + 1/((n-1)*\varsigma_2) + \ldots + 1/(3*\varsigma_{n-2}) + 1/(2*\varsigma_{n-1}))$

$= \Lambda(1/(n*\varsigma_1) + 1/((n-1)*\varsigma_2) + \ldots + 1/(3*\varsigma_{n-2}) + 1/(2*\varsigma_{n-1}))$

because $\sum_{i=1}^{n-1} \varepsilon_i = 0$;

Note that: $E_{comp} \geq (\varepsilon_1/(n*\varsigma_n) + \varepsilon_2/(n*\varsigma_n) + \ldots + \varepsilon_{n-2}/(n*\varsigma_n) + \varepsilon_{n-1}/(n*\varsigma_n)) + \Lambda*(1/(n*\varsigma_1) + 1/((n-1)*\varsigma_2) + \ldots + 1/(3*\varsigma_{n-2}) + 1/(2*\varsigma_{n-1}))$

So we establish: $E_{comp} = \Lambda(1/(n*\ln(n)) + 1/((n-1)*\ln(n-1)) + \ldots + 1/(3*\ln 3) + 1/(2*\ln 2))$

Since $\frac{1}{x\ln(x)}$ is a continuous decreasing function, the sum can be bounded using integration:

$$\int_3^{n+1} \frac{1}{x\ln(x)}dx \leq \sum_{i=3}^{n} \frac{1}{i\ln(i)} \leq \int_2^{n} \frac{1}{x\ln(x)}dx$$

Thus if $\Lambda \geq 0$, $(\ln|\ln(n+1)| - \ln|\ln(3)| + 1/2\ln(2))\Lambda \leq E_{comp} \leq (\ln|\ln(n)| - \ln|\ln(2)| + 1/(2\ln(2)))\Lambda$

If $\Lambda < 0$: $(\ln|\ln(n+1)| - \ln|\ln(3)| + 1/2\ln(2))\Lambda \geq E_{comp} \geq (\ln|\ln(n)| - \ln|\ln(2)| + 1/(2\ln(2)))\Lambda$

So:

$$E_{comp} \leq |(1.088 + \ln|\ln(n)|)\Lambda|, \text{ and}$$

$$\Pi \leq \Delta T_n + (\Delta T_{n-1}/2) + |(1.088 + \ln|\ln(n)|)\Lambda|$$

Therefore, we have proved the upper and lower bound of Π in Theorem 1.