Feature Extraction on 3D TexMesh Using Scale-space Analysis and Perceptual Evaluation

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

***Abstract*— Efficient on-line visualization of 3D textured models is essential for a variety of applications including not only games and e-commerce, but also heritage and medicine. To visualize 3D objects online, it is necessary to quickly adapt both mesh and texture to the available computational or network resources. Earlier research showed that after reaching a minimum required mesh density, high-resolution texture has more impact on human perception than a denser mesh. Given limited bandwidth, an important issue is how to extract features that best represent the original object, and how to allocate resources between mesh and texture data to achieve optimal perceptual quality. In this paper, we propose a textured mesh (TexMesh) model, which applies scale-space analysis (SSF) and perceptual evaluation to extract 3D features for textured mesh simplification and transmission. Texture data is divided into fragments to facilitate quality and bandwidth adaptation. Texture quality assignment is based on feature point distribution. On-line transmission is based on statistics gathered during preprocessing, which are stored in a priority queue and lookup tables. Quality of Service (QoS) requested by a client site can be met by applying an efficient adaptive algorithm to ensure optimal use of the specified time and available bandwidth, and at the same time preserving satisfactory quality. Our TexMesh framework integrates feature extraction, mesh simplification, texture reduction, bandwidth adaptation, and perceptual evaluation into a multi-scale visualization framework.**

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

I. INTRODUCTION

EFFICIENT online visualization of 3D objects is essential in multimedia applications. In the past, because of the constraints on rendering hardware and scanning techniques, only limited number of polygons was used to represent the geometry of 3D objects; resulting in poor visual quality. However, with advancements in rendering techniques and scanning technology, 3D objects in the virtual world can now be presented with denser meshes and high-resolution textures. While the availability of an increased amount of 3D range data brings promise to multimedia applications, it also imposes challenges to researchers especially for on-line transmission. Although network speed has been improved significantly in recent years, it is not always fast enough to transmit data with sufficiently low latency to satisfy the Human Visual System (HVS). Another problem on the Internet is fluctuating bandwidth, and bottleneck during periods of high traffic. To solve these problems and provide reasonable support for on-line applications, such as Tele-health, electronic games, etc., the transmitted data has to be adapted to current bandwidth, without significantly affecting human perception.

A. Previous Approaches

Due to different user constraints, the original mesh data have to be simplified in order to display 3D objects efficiently at client sites. The 3-dimensional terrain model and height fields are discussed in [14], [18]. Different mesh simplification techniques have been proposed for more general 3-dimensional models in the last decade [11], [22], [35], [38]. Simplification techniques applied on parametric surfaces can be categorized as follows:

Regular grid – This method is simple to implement, but may miss critical points degrading visual fidelity.

Hierarchical – Differing from the non-adaptive property of regular grids, this method provides the adaptive counterpart of pyramidal structures. Regions are subdivided recursively forming a tree-like hierarchy [5], [44].

Features – This approach performs triangulation based on a set of features or critical points. Southard uses the Laplacian as a measure of curvature to rank feature points, but his approach applies only to planar surface models and tends to distribute points uniformly, causing redundancy on low curvature surfaces and insufficient geometric data on high curvature surfaces [46]. Some other feature detection techniques found in the literature are designed for surface reconstruction, and not for model simplification; e.g., marching cubes [25] and neighborhood graphs [21].

Refinement – An early refinement technique can be traced back to the Douglas's algorithm on 2D curve simplification [15]. Refinement methods in 3D start with a minimal approximation on a set of selected points and apply multiple passes. A test point is initialized at the center of the triangle and it repeatedly steps up in the neighborhood until a local maximum is reached [17]. However, their approach may fail to find the global maximum within the triangle. Schmitt used a two-stage split-and-merge algorithm [43].

Decimation – In opposition to refinement methods, the idea is to start with all the scan-points and recursively remove vertices from the triangulation [23], [57]. Scarlatos [47] suggested an algorithm involving an initial triangulation and three phases: shrinking triangles with high curvature, merging adjacent coplanar triangles, and swapping edges to improve shape.

Optimal – In general, optimal methods are less common than

heuristic methods because they are slower and more expensive to implement. The best-effort simplification is obtained within a tolerance limit, defined by a specific criterion.

B. Motivation

This paper focuses on mesh mapped with high-resolution texture (TexMesh). Despite excellent past research on simplification techniques, what has not been established for 3D TexMesh is a feature extraction technique associating simplification at multiple scales, taking human perception and network adaptation into account. Existing techniques in the literature are inadequate in one or more of the following:

Capability of drastic simplification – Topology preservation [16] can maintain high fidelity but is not efficient when considering level of detail (LOD). When viewing an object at a far distance, the human visual system (HVS) is insensitive to small genus embedded in the object, and therefore rendering more triangles in order to preserve these holes is not an effective way of simplification. The error minimization techniques are useful to reduce over-sampled data, while preserving visual fidelity. However, error metric [13][49] can prevent drastic simplification, which is required when displaying distant objects.

Supporting both local and global simplification – Simplification techniques often focus on local property on 3D surface; they lack consideration of the global structure, which can result in poor performance on surfaces with high-frequency details [29], [34], [40].

Efficient online performance – Many simplification techniques involve relocation of vertices and thus online transmission cannot be incremental [5], [19], [24], [44], [51]. An image-driven simplification method is used to display textures using images from multiple views [33]. However, rendering the entire model for every edge in every viewpoint for different scales is expensive, even with hardware-accelerated rendering.

Inserting vertices without improving perceptual quality – Previous approaches assume that visual quality increases as the number of vertex increases. However, recent perceptual evaluation experiments show that after reaching a certain threshold, further increase in mesh resolution has no significant impact on perceptual quality [37].

Associating of texture reduction with mesh simplification using high-resolution real texture mapping – Textures mentioned in the literature often refers to synthetic or animated texture [50]. Synthetic textures or per pixel color stored in each vertex [10], [12], [20], [45], [48] can be estimated or interpolated. Since high-resolution real texture data is complex and large compared to mesh data, texture reduction will surely speed up 3D visualization. Photo-realistic texture maps are used in [55], but their effort is on recovering geometry from texture patches retrieved from multiple photographs, and not on generating LOD. A distance-based technique is applied to photo-textured terrain [30]; however, color interpolation between pixels is necessary to avoid blocky appearance of terrain texture.

Simplification based on perceptual evaluation – Number of vertices, faces, or an error metric is often used as a criterion to measure efficiency of simplification techniques in the literature [38]. A perceptually driven simplification technique is discussed in [28], but their method applies to the rendering of a view-dependent image, while our TexMesh model applies to a view-independent 3D object. They use Gouraud-shaded meshes while we use real texture mapping. Furthermore, their simplified models still contain redundant data, because the authors admitted that their models could be reduced two to three times further in polygon count without perceptible effect. Watson compared the naming times, rating and preference technique [53], but they limited their study to a particular view of each object, and not a full 360° interactive comparison. Naming times tend to be affected by an individual's prior knowledge of the stimuli. Also, even if the visual quality of an object is unsatisfactory, a judge may still be able to recognize it and name it.

Adaptation to bandwidth fluctuation – Joint geometry/texture progressive coding applies wavelet transform to encode the mesh and texture data for transmission [36], but the method cannot adapt to fluctuating bandwidth.

To support heterogeneous clients, the main challenges are: (1) How to extract features which can best represent the original object, based on human perception for a given bandwidth? (2) How to adapt the extracted data to bandwidth fluctuation minimizing the adverse effect on visual quality? (3) How to allocate the given bandwidth between mesh and texture data? Varakliotis did a detailed study on how dynamic 3D models can be adaptively transmitted taking visual quality into account [52]. His focus is on efficient transmission of the meshes preserving smooth motion transition. Our focus is on extracting static 3D features that can best represent the original model, allocating bandwidth efficiently between mesh and texture, and applying a texture fragmentation approach to achieve quality and bandwidth adaptation. The goal of our TexMesh model is to integrate related research topics from different directions, and fix the inadequacies of previous approaches. While visual fidelity has traditionally been measured using quantitative metric, the TexMesh model verifies geometric quantities with a perceptual metric.

The rest of the paper is organized as follows: Section 2 reviews Laplacian and Gaussian filtering in scale-space, and discusses the spherical version of the filtering function. Section 3 explains how to apply the spherical filtering function in a TexMesh to extract 3D features, and how variable qualities are assigned to fragments based on feature point distribution at multiple scales. Section 4 analyzes perception of mesh refinement. Section 5 presents a summary of the TexMesh framework integrating feature detection, textured mesh simplification, and quality and bandwidth adaptation. Section 6 gives the conclusion and future work.

II. SCALE-SPACE FILTERING

The Gaussian filter is an efficient smoothing tool in computer vision. Based on the Gaussian's kernel, Witkin [54]

and Koenderink formally defined the scale-space concept in 1983-84. Since then the Gaussian kernel has often been studied in conjunction with the multiple scale approach. However, scale-space filtering has mainly been applied in 2D images [3], [26], [31], [32], and only recently has this technique been used in computer graphics, with limited applications in 3D visualization. An intrinsic filter, a generalization of scale space filtering, is used to eliminate noise from image and scanned data [1]. Relatively fewer researchers have looked into 3D model simplification based on feature point analysis at multiple scales. Southard applied Laplacian to rank uniformly distributed feature points on planar surfaces [46]. Gaussian filter is used to detect features at multiple scales [39], but their analysis is not extended to mesh integrated with real texture and adaptive on-line transmission.

A. Level of Detail (LOD) in Scale-Space

We use scale-space filtering (SSF) to extract 3D features. Traversal between the different scales, or LOD [9] is achieved by varying the standard deviation parameter σ; the higher the value 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 in higher scales. In other words, minor features will be removed before major features in the simplification process. The advantage of using SSF is its ability to smooth locally or globally, moderately or drastically, depending on the filter window size and the value of σ. 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.

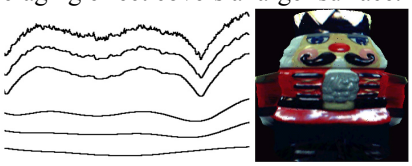


Fig. 1: Increasing scale S_i from top to bottom. S_0 is the original signal generated by 360 scan points extracted near the bottom of the Nutcracker model. Note that the local variation (fine detail) in the original signal is gradually removed and the scaled signal becomes smoother.

Fig. 1 is an example of global smoothing using a window size of 201 pixels on a signal of 360 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.1. Theoretically, 100% global smoothing will end up with a monotonous surface losing all the surface features. The Human Visual System (HSV) is insensitive to details beyond a certain distance. Thus, for a perceivable object, the filter window can be smaller than twice the standard deviation. In our experiments, we applied a window size of 1.4 times the standard deviation, and found that this window size provides sufficient simplification for objects placed at a distance close to infinity.

The zero-crossings at different scales can be computed by applying the second derivative of the Gaussian (called Laplacian-of-Gaussian or *LoG*). Eighteen feature points are identified in the original signal (Fig. 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. 2, left).

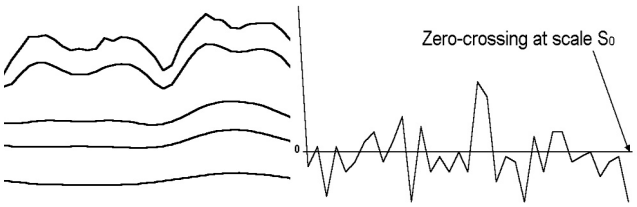


Fig. 2: (Left) The top is the original signal with 18 zero crossings, generated by 36 scan points extracted from the Nutcracker model. The next four smoothed scales have 8, 6, 4, and 2 zero-crossings respectively. (Right) 18 zero crossings detected in the original signal S_0.

SSF in 2D can be summarized by the following equations:

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

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

$$f * S(x,y) = \int_{-t}^{t}\int_{-t}^{t} f(x+u, y+v) w(u,v)\, du\, dv \quad (3)$$

$w_G(x,y)$ represents the weight at pixel (x,y), f represents the original signal (image) and $f*S$ the smoothed image and the weights are assumed to be defined in a square window W of length $2t+1$. In the discrete case, *e.g.,* with a real image, summation is used instead of integrals, and the Gaussian weights are normalized so that the sum of all the weights equals 1.

B. Spherical Approach on Scanned Range Data

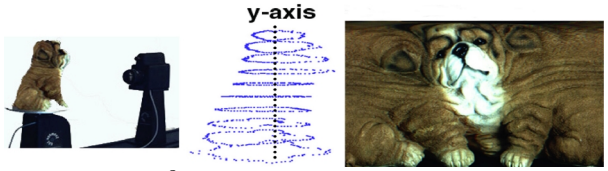


Fig. 3: (left) Zoomage® 3D scanner, (middle) sample of 3D points, and (right) texture image.

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

The SSF of a 3D model is achieved as follows: First note that the data acquired (Fig.3 middle) 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 coordinate, and

R_x denotes the perpendicular distance from the y-axis to the surface of an object for a given (α,y) pair. SSF for a 3D model is thus similar to a 2D image, for the simplified mesh representation considered here, with $f(x,y)$ 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, Equations (1) and (3) need to be modified to (4) and (5):

$$w_G(\alpha,y)=\begin{cases}\dfrac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{\phi\alpha^2}{2\sigma^2}-\frac{\psi y^2}{2\sigma^2}} & (\alpha,y)\in W\\ 0 & elsewhere\end{cases} \quad (4)$$

$$Rx*S(\alpha,y)=\int_{-t}^{t}\int_{-t}^{t}Rx(\alpha+u,y+v)w(u,v)dudv \quad (5)$$

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 along different axes. 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 shows the face features, of a head model, changing towards a smoother spherical surface when going from low to high scales (left to right). The original mesh contains 1,872 vertices and 3,672 faces. The other five meshes have 2,226, 2,108, 1,948, 1,624 and 1,190 faces respectively.

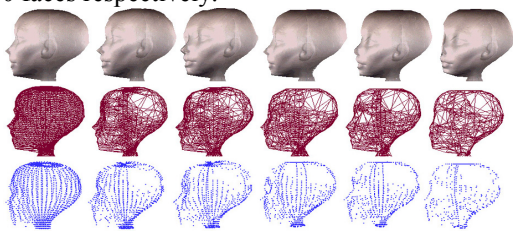


Fig. 4: Increasing scales from left to right (Top) 3D mesh with texture, (Middle) 3D mesh and (Bottom) feature points extracted at each level.

III. THE TEXMESH MODEL

Many studies emphasized the importance of million of triangles to preserve fine details, but ignored the basic fact that high resolution real texture have been shown to have more impact on perceptual quality in 3D visualization [27][37][41]. In view of the size of texture data, compressing high-resolution texture is essential for on-line transmission given limited bandwidth. An integrated approach, based on feature point extraction and distribution, is proposed in this paper.

Feature points in our TexMesh model is defined as a group of vertices, which can best represent the geometry of a 3D model at a given viewing distance. In Fig. 4, 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 crossings are recorded as feature points and assigned the value i. Each feature point is represented by three components: *(i, (tx,ty), (gx,gy,gz))*. The second and third components are the 2D texture and 3D vertex coordinates, respectively. Vertices with stronger persistence will have higher i values (higher priority). During simplification, features of low priority are first removed, leaving more prominent features at higher scales. We apply iterative edge collapse operations to generate each simplified version. During refinement, features of higher priority are inserted first into the coarse model. Since real texture mapping is used in the TexMesh model, texture border has to be taken into account during the simplification process. Vertices associated with border texels are removed only if an operation does not distort the texture pattern. Preprocessing generates a scale map and a fragment map. The scale map records the feature points at each scale, and the fragment map records the feature point distribution in each texture fragment.

A. Scale Map

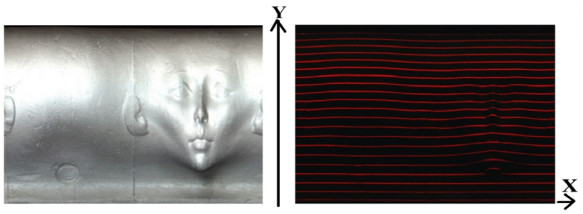


Fig.5: (Left) Texture pattern of a head model associated with the scanned signal, and (Right) scanned laser signal of the head model generated by the Zoomage® 3D scanner.

Mesh vertices are derived from the signals generated by 3D scanners. Fig. 5 (right) shows an example of the scanned signal, which is processed to compute the depth information. Putting aside the depth information (z-coordinate), N vertices can be sorted and assigned unique ids L, *i.e.*, $0 \le L < N$, based on their y then x coordinates. A Scale Map is a 2D structure storing the N vertices in rows and columns corresponding to the y and x values respectively. The default value for each vertex is 0 corresponding to scale 0 (original signal). At each scale S_i only feature points detected at that scale are updated with the value i. During preprocessing, Gaussian filters with increasing sigma values σ_i are used from scale S_0 to S_{max}, *i.e.*, $0 \le i \le$ max, with S_{max} corresponding to scale at infinity. Zero crossings are detected from the filtered space G_i where G_0 represents the set of original unfiltered range data. A priority queue is implemented to store feature points in decreasing i values. Starting from a coarse model, feature points not already included in the mesh can be extracted from the priority queue to refine the model progressively.

Decimation and refinement are performed using edge collapse and vertex split operations. There are two main differences between our edge collapse/vertex split and that used in progressive meshes [24]: (1) There is no vertex relocation between different level of detail in our TexMesh; all vertices at a coarse level is a subset of those at a finer level. Hoppe's vertex split operation generates two new vertex locations in the refined object to replace one vertex in the coarser object. (2) In progressive meshes, the minimum energy cost, recalculated each time a new vertex is introduced

in an edge collapse operation, affects the choice of the next collapsing edge. In the TexMesh model, the order of collapsing edges follows the priority predetermined by applying SSF on the original 3D surface.

Vertex V_R is removed by integrating with its closest neighbor V_C, bringing the edges with it (Fig. 6).

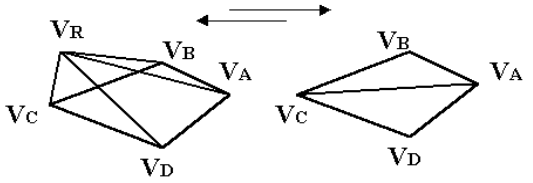


Fig. 6: (Left) A finer version before V_R is removed, and (Right) A coarser version after removing V_R.

Vertex V_R and associated information are transmitted during refinement to reconstruct the fine version.

B. Fragmentation Approach for Texture Transmission

The TexMesh model associates texture reduction with mesh simplification. In a 3D object, with full color real texture, there are two components that can be filtered — the mesh and the texture. We determine the texture quality $q_i(x,y)$ of a fragment *(x,y)* based on the associated feature point distribution $\eta_i(x,y)$, which has a value ∈ [0,1], and is mapped onto a quality range. For example, in the current implementation, we use the JPEG quality scale [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.

The texture image can be transmitted as one block or a collection of sub-blocks. The advantage of dividing into sub-blocks is to make use of distributed networks and apply variable qualities to different texture regions as explained below. The main concern is whether fragmentation will increase the overall volume of transmitted data. 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 supports bandwidth and quality adaptation. After generating the scale map, vertices are distributed onto the corresponding sub-block in a fragment map based on their texel coordinates.

C. Fragment Map and Variable Texture Quality

The image texture 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 macro-blocks, 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, texture with dimension of 4800*1600 pixels can be divided into 7,500 fragments of size 32*32 pixels. Fragments are 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. 7) to compare the fragmented and non-fragmented sizes for different qualities using the Intel JPEG compression library. Each fragment had a dimension of 16*16 pixels. The 24-bit RGB true color texture image was read into memory and partitioned into *N* fragments. These fragments were compressed *independently* at quality Q using the Intel JPEG compression algorithm (Q = 20%, 40%, 50% and 60% were used in the experiments), and saved as individual JPEG files. The total size of these JPEG files was recorded as *sum of fragments*. The texture image, now composed of compressed patches, was output as one JPEG file without further compression. The size of this file was recorded as *non-fragmented*. Experimental results (Fig.7), using five different texture images, show that for high-resolution texture (over 500^2 pixels), the fragmentation approach does not increase the total amount of data transmitted.

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

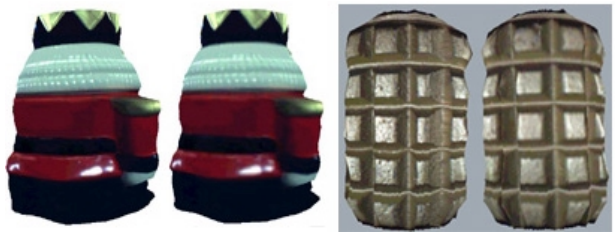
Model texture	Resolution (pixels)	JPEG Quality	Non-fragmented file size (KB)	Sum of fragments (KB)
(a)	1024^2	60%	566	293
	256^2	20%	29	44
(b)	1024^2	60%	305	222
(c)	1024^2	60%	400	253
(d)	1024^2	60%	494	269
(e)	1024^2	40%	383	226
		50%	423	237
		60%	458	249
	512^2	40%	111	62
		50%	122	66
		60%	130	70
	256^2	20%	25	42
		60%	56	47

Fig. 7: Experimental results show that sum of fragments of optimal size is less than the size of a corresponding non-fragmented file for high resolution texture.

The fragmentation approach also supports distributed transmission. To get around network bottlenecks, applications can select different route(s) to transmit, or benefit from multi-servers retrieval [2], by storing texture in multiple repositories. Instead of requesting an image from a single server, fragments of an image can be retrieved from multiple servers in a distributed network [4] and reassembled. Since the texture data in a TexMesh is organized in fragments, the server can select less congested routes to transmit fragments.

Since the HVS is less sensitive to details further away, the texture quality Q_i at each scale S_i needs to increase only when *i* decreases towards a finer model. For the discussion in this paper, we use viewing distance to represent the distance between the 3D object and the viewing platform in the virtual world, which is different from the distance between the viewer and the display device. 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 distribution 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.8 c & d, the grenade has vertical structures on the surface, and therefore the feature point distribution is higher than the back of the nutcracker (Fig.8 a & 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.8 d) shows noticeably lower perceptual quality. While the difference in quality between the grenades is obvious, the degraded shiny patch under the belt of the nutcracker (Fig.8 b) 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 the same quality to every fragment. 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 fluctuates and can adversely affect the expected quality of service (QoS).



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

Fig. 8: A snap shot of the nutcracker 3D model 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).

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

Notation

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

ΔQ – 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.

Q_i – Default texture quality associated with scale S_i, by mapping $[S_0, S_{max}]$ to a quality range, e.g. [100%, 0%] for JPEG quality.

(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 $\eta_i(x,y)$.

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.

D_i – Data size of all fragments at S_i.

A 3D object has the highest quality at S_0 and lowest quality at S_{max}. A quality range, *e.g.* [100%, 0%] for JPEG, can be mapped onto $[S_0, S_{max}]$. A linear mapping is employed in the current implementation for simplicity. A default texture quality Q_i can be obtained based on S_i. At scale i, $f_i(x,y)$ is normalized as:

$$\eta_i(x,y) = \frac{f_i(x,y) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \tag{6}$$

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

$$Q_i + (\eta_i(x,y) - \Gamma)\,\Delta Q \tag{7}$$

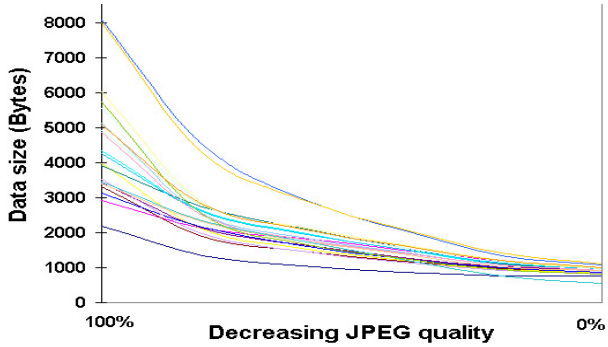


Fig. 9: Different texture patterns were tested to show how decreasing fragment data size relates to reducing quality.

The data size of a texture fragment $d_i(x,y)$ decreases when its quality $q_i(x,y)$ is reduced (Fig. 9). Instead of applying the same quality to every fragment, we apply variable qualities. The threshold Γ is first set to $\bar{\eta}_i \in [0,1]$. Fragments having feature point distribution equal to the threshold is assigned the quality Q_i, above the threshold have a quality higher than Q_i, and below the threshold have a quality lower than Q_i. By increasing the threshold, more fragments will fall below quality Q_i, and thus the total data size of all fragments D_i will decrease. On the other hand, by decreasing the threshold, more fragments will have quality higher than Q_i, and D_i will increase. ΔQ controls the deviation (+/-) from Q_i constrained by an upper and a lower bound. Regions on the 3D surface with higher feature point distribution are displayed with higher quality, and less populated regions are displayed with lower quality texture. A selected range of thresholds is used to estimate the data size of different combination of qualities. Such information is stored in a lookup table (LUT). During online transmission, the target data size is computed based on an estimated bandwidth. The quality set from the LUT best matching the target size is used for bandwidth adaptation.

Our feature extraction and fragmentation approach works with an adaptive transmission strategy; such as the Harmonic Time Compensation Algorithm (HTCA) in [6]. The HTCA defines Π as the deviation from a given time limit for the entire transmission period. It can be proved that Π is bounded

by:

$$\Delta T_n + (\Delta T_{n-1}/2) + (1.088 + \ln|\ln(n)|)\Lambda \quad (8)$$

Given n fragments, Λ is the average difference between the estimated and actual transmission time ΔT_j for the first n-1 fragments, *i.e.,*

$$\Lambda = \frac{\sum_{j=1}^{n-1} \Delta T_j}{n-1}. \quad (9)$$

After testing with three sets of bandwidth values extracted from an Ethernet connection, it was shown that the overall time deviation is within 1% of the defined time limit. Details of the proof and experiments can be seen in [6].

IV. Visual Perception of Mesh Refinement

Since the HVS is insensitive to minute details below a certain visual threshold, a coarser mesh should be used instead of a finer one, to save computation and network resources, if they are visually similar. In this section, we will show that for some scales it is possible to reduce the mesh resolution, without degrading visual quality.

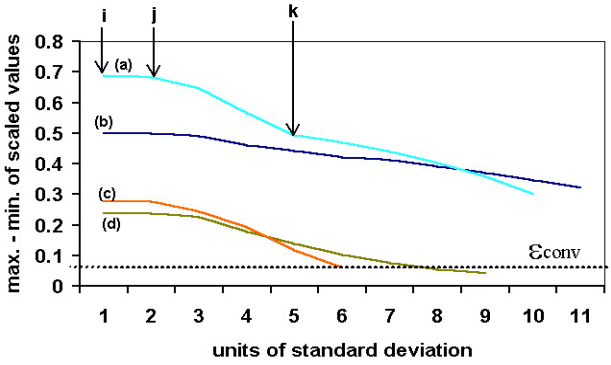


Fig. 10: SSF Convergence Analysis – the difference between the maximum and minimum scaled values diminishes during the smoothing process as the scale increases.

Based on scale-space theory, the number of feature points (structures in the sample space) decreases as scale level increases. Different LOD can be generated, by varying the σ value in Equation (4). However, different 3D models have different surface structures and thus the maximum σ value (σ_{max}) applied to achieve a sufficiently smoothed surface is different. To determine σ_{max}, we recorded the rate of convergence while performing SSF. Fig. 10 is an example of different convergence rates in four sets of sample values, extracted from (a) a dense head, (b) a dense nutcracker, (c) a simplified head and (d) a simplified nutcracker object.

For increasing values of σ on the horizontal axis, the vertical axis represents the difference between the maximum and minimum scaled values. In other words, a difference of 0 means 100% smoothing. Let us define a convergence threshold $\varepsilon_{conv} > 0$. There are two observations:

(1) When the number of sample values increases, σ_{max} also increases. Samples (a) and (b) contain more sample values and require a higher σ_{max} to reach ε_{conv} than samples (c) and (d).

(2) Different models have different surface features and require different σ_{max} to reach ε_{conv}.

A. Perceptual Evaluation Experiments and Analysis of Result based on the Human Visual System

In the perceptual experiments, the visual stimuli were 360° view-independent texture-mapped 3D objects. The illumination and texture resolution were fixed for the visual stimuli, and the judges had to decide whether the simplified object could be viewed at a closer distance without noticeable distortion. Since the stimuli had the same illumination, contrast sensitivity did not affect our result. Visual acuity is significantly higher at the fovea than in the visual periphery [42], and is an important factor when considering gaze-directed perceptual evaluation. However, our rotating 3D objects were view-independent, balancing the visual acuity between the fovea and its periphery after an object had rotated a complete cycle. Until recently [37], perceptual comparisons in the literature have focused on view-dependent display, which shows only a limited number of silhouettes. By contrast, using a 360° view allows all silhouettes to be examined.

Viewing distance is defined from 0 to infinity where an object vanishes. For each object, the scales generated by value $\sigma \in [0, \sigma_{max}]$ are mapped onto corresponding distances in the range [0, ∞]. In the experiments, we used 20 scales in addition to the original signal to cover this range so that S_0 corresponds to $\sigma = 0$ and S_{max} (S_{20}) corresponds to σ_{max}.

In the experiments, a simplified version of the 3D object was generated and displayed at a distance related linearly with its scale, i.e. S_0 is closest and S_{max} is farthest. Judges had to decide whether the coarser version on the right was perceptually similar to the original version on the left. If no, they moved the simplified version away to a satisfactory position. If yes, the judges moved it closer until there was a noticeable distortion. Both objects were rotating, so that the judges could compare a complete 360° view. For each of the five objects (nutcracker, head, dog, grenade and vase), a number of simplified versions were randomly generated. Preliminary results suggest that, in some cases, a coarser version can be used. When plotting the function between distance and scale, it is very close to a step shape (Fig.11). The x-axis is the scale and y-axis is the distance with zero being the closest. The distance increases in the negative direction.

We analyze the result by dividing the graph into alternate red (perceptible) and green (imperceptible) zones, with the red zone corresponding to the slope and the green zone corresponding to the comparatively flat portion. In the red zone, elimination of perceptually important feature points causes a noticeable degradation in visual quality. While in the green zone, feature points eliminated do not have significant perceptual effect. For example, changing the scale from B to A does not improve perceptual quality, while refining from C to B has significant impact. For each object, the step function is unique, containing more or less steps depending on the surface features of the object. The step function suggests that perceptually less important data can be suppressed during refinement, using major scales (all those in the red perceptible

zone) to represent LOD.

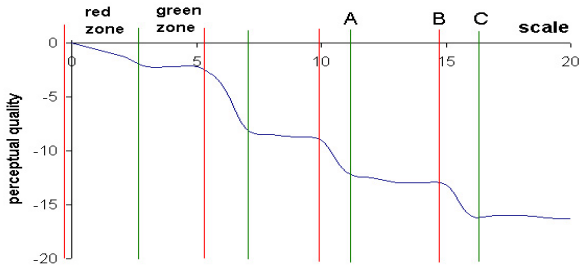


Fig.11: Preliminary results show that at a given distance, different scales can have similar visual quality. Scale relating to distance is close to a step function.

Given a distance d, we locate the corresponding scale S_i. If S_i falls into a green zone, the rightmost major scale is selected, because using other scales (minor scales) in the zone does not improve visual quality. If S_i falls into a red zone, S_i is used because each scale in the zone is a major scale.

The step function is consistent with the convergence property of SSF shown in Fig. 10. We already noted that objects with more scan-points (samples a & b) take longer to converge to a predefined value ε_{conv}, but more importantly, for each sample, the convergence rate is not constant. Since the HVS is insensitive to minute changes below a certain limit (*e.g.* the convergence rate between i and j), refining the scale from 2 to 1 will not have significant impact on the visual quality. However, the convergence rate between j and k indicates a significant effect on visual quality with changing scale.

B. *Mesh Refinement based on Major Scales*

When a 3D object moves closer to the viewer, the mesh scale is upgraded only if the refined mesh improves visual quality. Reddy approximates the contrast sensitivity function (CSF) in dynamic scenes to optimize the amount of detail removed from the scene without the user noticing [60]. By contrast, our method is designed for comparatively static 3D objects. To determine whether mesh refinement should be performed requires measuring perceptual impact. Adding or deleting a structure generates a stimulus to human vision. To compare the perceptual impacts of these stimuli, the dimension of a structure can be used as a visual cue. In each edge collapse operation during preprocessing, when a vertex V_R is removed and integrated with its closest neighbor V_C, the surface change is proportional to the distance ρ_R between V_R and the average plane formed by its neighbors. ρ_R is defined as the *perceptual value* of V_R. If both V_PV_Q and V_QV_C collapse, the perceptual value of the combined stimulus is the cumulated sum of ρ_P and ρ_Q (Fig. 12).

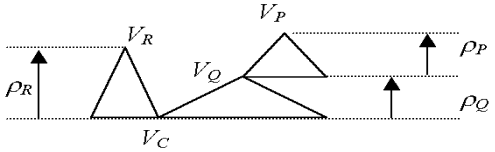


Fig.12 Vertices V_R, V_P and V_Q have perceptual values ρ_R, ρ_P and ρ_Q respectively.

Let $\Delta\wp$ be the maximum perceptual value when refining from S_i to S_{i-1}, and $\wp$ be the length of the cross-section of the 3D object in the direction of $\Delta\wp$. When viewed on the client's display device, the extension $\Delta\wp$ generates an angle $\Delta\theta$ as a stimulus, increasing the original angle θ cast by $\wp$ (Fig. 13). The *Difference Threshold* $K_\wp$, or Just Noticeable Difference (JND), is the minimum change in perceptual value in order to produce a noticeable variation in visual experience. Weber's Law [58] states that the size of the JND is a constant proportional to the original stimulus value. Similar to intensity contrast, we apply Weber's Law on perceptual value, *i.e.*,

$$\frac{\Delta\wp}{\wp} = K \tag{10}$$

If $K > K_\wp$, perceptual impact is significant and S_i is refined to S_{i-1}. Weber's Law can be applied to a variety of stimuli, including brightness, loudness, mass, line length, size, etc.

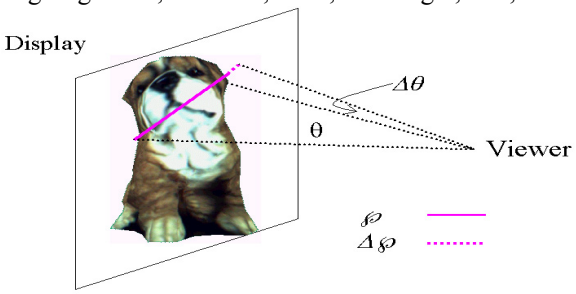


Fig.13: To evaluate the impact of perceptual values $\Delta\wp$, by comparing $\Delta\wp/\wp$ with the difference threshold.

Previous refinement techniques assume that visual quality increases when the number of vertices increases. We show by perceptual experiment that geometrically different meshes can be perceptually similar. Bandwidth should be allocated to texture data if a denser mesh does not improve visual quality.

Another approach is to follow the argument that humans naturally describe an object as consisting of parts and infer 3D shapes of these parts [59], and segment the object into corresponding parts (skeletonization). $\Delta\rho$ is computed by dividing the displacement of V_R by the shortest distance from V_C to the skeleton, when collapsing V_R onto V_C. For spherical – like object, the skeleton can be represented by the center of the object. Instead of computing the fraction $\Delta\rho$ using linear measurements, an alternative is to use the quadric error generated by removing V_R. We performed 361 perceptual evaluations on the nutcracker object (Fig. 1) involving twenty judges, and found that the threshold $K_\wp$ is approximately 0.096 with a correlation coefficient of 95%, when using linear measurement to compute $\Delta\rho$. The result shows that the linear metric predicts visual quality well, closely following human perception.

Based on this preliminary finding, we are extending our effort to increase the number of judges and 3D models to further verify the difference threshold for mesh refinement. We expect that texture resolution and complexity will affect the threshold. This is an issue we will address in future experiments.

V. AN INTEGRATED FRAMEWORK

Online 3D visualization is an expanding area of multimedia

research covering graphics, human perception and network transmission. Quality and bandwidth adaptation is important in 3D textured mesh simplification. Short-term fluctuations are difficult to adapt. In fact, for applications using small images and short transmission time, applying an historic average to estimate the network bandwidth is sufficient [8]. Our method addresses the transmission of high-resolution images where the transmission time is long. An increasingly popular area of research is related to Tele-conferencing, where artists, designers, physicians, surgeons, etc. can exchange ideas remotely. In order for the participants to edit a 3D object collaboratively [7], the entire object needs to be transmitted before editing, so that a complete 360° view is available to every participant.

To handle longer-term fluctuations, an optimal bandwidth monitoring approach was suggested [56] to provide more accurate bandwidth estimation by sacrificing a portion of the time to transmit test data. Our adaptive approach does not need to use test packets. Instead, bandwidth is efficiently utilized by adjusting the quality of the fragments not yet transmitted based on feedback about the traffic. The entire texture map is divided into fragments of optimal size. Given a viewing distance and a historic average bandwidth, the corresponding scale and a default texture quality are selected. The first fragment is transmitted with the default quality. The idea is to re-estimate the current bandwidth based on the transmission time of the previous fragment(s). The server continues to transmit, until a new update is available. The current bandwidth is then used to re-compute the qualities of the remaining fragments. The over or under estimation time from pervious fragment(s) is distributed to the remaining fragments. This adaptive strategy ensures that when the transmission is completed the overall time discrepancy, compared with the predefined time limit, is minimized [6].

In recent work [37], it was found that improving mesh resolution improves perceptual quality following an exponential curve whereas enhancing texture relates to perceptual quality linearly. Based on this finding and the fact that the texture component in a 3D image requires greater storage or bandwidth for communication, 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.

(b) Generate a priority list of feature points from strong to weak persistence. Compute the perceptual value associated with each deleted feature point between scales, and determine the maximum.

(c) Divide the model texture into fragments of optimal size. Generate a fragment map for each scale S_i by distributing the feature points onto the corresponding texture fragment.

(d) For each scale S_i use a lookup table to record the fragment data size, and associated qualities for a selected range of Γ.

Runtime processing

(a) Given a viewing distance, obtain the corresponding scale S_i. Shift S_i to a coarser version if the change does not affect visual quality.

(b) Use the initial bandwidth (historic average) B_0, and T_0 to locate the best matching set of texture fragments in the lookup table. T_0 is the given time limit minus the time required to transmit the mesh data.

(c) Adjust the texture quality of each fragment using an adaptive approach and transmit the texture fragments.

(d) Client site recombines fragments and renders the texture-mapped 3D model.

VI. Conclusion and Future Work

The TexMesh model proposed in this paper integrates mesh simplification and texture reduction based on scale-space analysis. We apply LoG to detect zero-crossings at each scale and generate statistics including a scale map and fragment map during preprocessing. These statistics are used during runtime for efficient extraction and transmission of 3D data. Feature points are transmitted only if they contribute to visual quality. Variable qualities applied to texture fragments, determined by feature point distribution, are readjusted during transmission to adapt to fluctuations in bandwidth. Experimental results show that this adaptive approach utilizes bandwidth more efficiently, and provides better control on QoS.

Packet loss during transmission is a common problem. When no other data is competing for bandwidth, redundant mesh data can ensure that the rendered quality will not be significantly affected by missing vertices. Since our method is designed for transmitting a 3D textured mesh, each 3D vertex is associated with a 2D texel. By transmitting the vertex and texel coordinates in separate packets, the (x,y) coordinates, if lost, can be estimated based on its texel counterpart received in a separate packet. Recall that feature points in the priority queue represent decreasing structure sizes. Therefore the depth component, z, can be estimated by taking the average of the previous and next feature points. However, this packet loss strategy works only when the application can tolerate a certain degree of estimation error. For applications, which require high precision, retransmission is more appropriate.

The step function relating scales with a given distance, which is consistent with the convergence property of scale-space filtering, suggests an efficient way to suppress redundant data during mesh refinement. Simplification techniques in the literature often use number of vertices, triangles, or a quantitative metric to measure efficiency. We show by experiments that perceptual quality is a more reliable measure when the HVS is involved.

View-dependent rendering, when used in a user collaboration situation [7], requires the server to keep track of a large number of different rendered views and transmit different sets of edited data based on each viewpoint. There is a trade-off between rendering the complete object at the outset and on-demand. For example, when the user decides to view

high-resolution medical data, museum exhibits, E-commerce products, etc., reserving time at the beginning to download the complete object enables swift subsequent navigation. The alternative is to wait for incremental update after each change of view. In future experiments, we will incorporate view-dependent perceptual evaluations to improve our results.

REFERENCES

[1] P. Boulanger, O. Jokinen and A. Beraldin, "Intrinsic filtering of range images using a physically based noise model," VI 2002, Calgary.

[2] A. Basu, I. Cheng and Y. Yu, "Multi-Server optimal bandwidth monitoring for QoS based multimedia delivery," IEEE Int'l Symposium on Circuit and Systems, May 2003 Bangkok, Thailand.

[3] D. Bauer and R. Peikert, "Vortex tracking in scale-space," Eurographics-IEEE TCVG Symposium on Visualization, 2002.

[4] A. Basu, M. Pi, I. Cheng and M. Bates, "Distributed retrieval of wavelet images using bandwidth monitoring," Proceeding IAPR/IEEE Int'l Conference on Pattern Recognition, Aug 2002 Quebec City.

[5] D. Brodsky and B. Watson, "Model simplification through refinement", Proc. of Graphics Interface 2000.

[6] I. Cheng and P. Boulanger, "Adaptive online transmission of 3D TexMesh using scale-space analysis," 3DPVT Sept. 2004, Greece.

[7] I. Cheng, M. Bates and A. Basu, "Collaborative online 3D editing," Siggraph 2003 Webgraphics, San Diego CA USA.

[8] I. Cheng, A. Basu, Y. Zhang and S. Tripathi, "QoS specification and adaptive bandwidth monitoring for multimedia delivery," Proc. IEEE EUROCON, Slovakia, 2001.

[9] J. Clark, "Hierarchical geometric models for visible surface algorithms," Comm. ACM, vol. 19, no. 10, 1976, p. 547-554.

[10] D. Cohen-Or, Y. Mann, S. Fleishman, "Deep compression for streaming texture intensive animations", Siggraph 1999.

[11] P. Cignoni, C. Montani, R. Scopigno, " A comparison of mesh simplification algorithms," Computer and Graphics, p. 37-54,1998.

[12] J. Cohen, M. Olano and D. Manocha, "Appearance-preserving simplification", Siggraph 1998.

[13] J. Cohen, A. Varshney, D. Manocha, G. Turk and H. Weber, "Simplification envelopes", Siggraph 1996.

[14] P. Cignoni, E. Puppo, R. Scopigno, "Representation and visualization of terrain surfaces at variable resolution", The Visual Computer Vol.13 (5): p. 199-217, 1997.

[15] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," The Canadian Cartographer, 10(2): p. 112–122, 1973.

[16] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," Computer Graphics (Proc. Siggraph 95), vol. 29, ACM Press, New York, 1995, p. 173-182.

[17] R. Fowler and J. Little, "Automatic extraction of irregular network digital terrain models," Computer Graphics Proceedings, Annual Conference Series, p. 199-207. ACM Siggraph, 1979.

[18] L. Floriani, P. Magillo, E. Puppo, " Variant: A system for terrain modeling at variable resolution", GeoInformatica 4:3, p. 287-315, 2000 Kluwer Academic Publishers.

[19] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics", Siggraph 1997.

[20] M. Garland and P. Heckbert, "Simplifying surfaces with color and texture using quadric error metrics", IEEE Visualization 1998.

[21] S. Gumhold, X. Wang and R. Macleod, "Feature extraction from point clouds", Proc. 10th Int'l Meshing Roundtable 2001.

[22] P. S. Heckbert, M. Garland, "Survey of polygonal surface simplification algorithms", Multiresolution Surface Modeling Course Siggraph 1997.

[23] P. Hinker and C. Hansen, "Geometric optimization," In Proc. Visualization 1993, p. 189–195, San Jose CA, October 1993.

[24] H. Hoppe, "Progressive meshes" Siggraph 1996, L.A. p. 99-108.

[25] L. Kobbelt, M. Botsch, U. Schwanecks and H. Seidel, "Feature sensitive surface extraction from volume data," Siggraph 2001.

[26] A. Kuijper and L. Florack, "Logical filtering in scale space", Institute of Information and Computing Sciences, Utrecht University, TR, 2001.

[27] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni and R. Scopigno, "Protected interactive 3D graphics via remote rendering," Siggraph 2004, Los Angeles, CA USA.

[28] D. Luebke and B. Hallen, "Perceptually driven simplification for interactive rendering", 12th Eurographics Workshop on Rendering Techniques, London, UK 2001.

[29] P. Lindstrom, "Out-of-core simplification of large polygonal models," ACM Computer Graphics (Proc. Siggraph 2000), Vol. 34, 2000, p. 259-262.

[30] P. Lindstrom *et al*., "Level of detail management for real-time rendering of phototextured terrain", TR-95-06, GeorgiaTech.

[31] T. Lindberg, "A scale selection principle for estimating image deformations," ICCV, Cambridge, MA, USA, 1995, p. 134-141.

[32] T. Lindberg, "Feature detection with automatic scale selection," International Journal of Computer Vision, No. 2, 1998.

[33] P. Lindstrom and G. Turk, "Image-driven simplification", ACM Tran. On Graphics, 2000.

[34] K-L. Low and T. S. Tan, "Model simplification using vertex clustering," Proc. 1997 Symp. Interactive 3D Graphics, ACM Press, 1997, p. 75-82.

[35] D. P. Luebke, "A Developer's survey of polygonal simplification algorithms", IEEE Computer Graphics and Applications, May/June '01, 24-35.

[36] M. Okuda and T. Chen, "Joint geometry/texture progressive coding of 3D models", IEEE Int'l Conf. on Image Processing (ICIP), Vancouver 2000.

[37] Y. Pan, I. Cheng and A. Basu, "Quantitative metric for estimating perceptual quality of 3D objects," IEEE Trans. on Multimedia, April 2004.

[38] M. Pauly, M. Gross and L. Kobbelt, "Efficient simplification of point-sampled surfaces", IEEE Visualization 2002, 2002.

[39] M. Pauly, R. Keiser and M. Gross, "Multi-scale feature extraction on point-sampled surfaces," Eurographics 2003, Granada, Spain.

[40] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes," Geometric Modeling in Computer Graphics, Springer-Verlag, Berlin, 1993, p. 455-465.

[41] H. Rushmeier, B. Rogowitz and C. Piatko, "Perceptual issues in substituting texture for geometry," Proceeding of SPIE Vol.3935, p372-383.

[42] J. Rovamo and V. Virsu, "An estimation and application of the human cortical magnification factor", Experimental Brain Research, 37 (1979).

[43] F. Schmitt and X. Chen, "Fast segmentation of range images into planar regions", IEEE Computer Vision and Pattern Recognition p. 710-711, 1991.

[44] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes", IEEE Visualization 2001.

[45] M. Soucy, G. Godin and M. Rioux, "A texture-mapping approach for the compression of colored 3D triangulations", The Visual Computer (1996) 12: p. 503-514.

[46] D. A. Southard, "Piecewise planar surface models from sampled data", in N. M. Patrikalakis, editor, Scientific Visualization of Physical Phenomena, p. 667-680, Tokyo, 1991. Springer-Verlag.

[47] L. L. Scarlatos and T. Pavlidis, "Optimizing triangulations by curvature equalization", IEEE Visualization, p. 333-339, 1992.

[48] P. Sander, J. Snyder, S. Gortler and H. Hoppe, "Texture mapping progressive meshes", Siggraph 2001.

[49] W. Schroeder, J. Zarge and W. Lorensen, "Decimation of triangle meshes", Siggraph 1992, p. 65-70.

[50] G Turk, "Generating texture on arbitrary surfaces using reaction-diffusion", Siggraph 1991.

[51] G Turk, "Re-tiling polygonal surfaces", Siggraph 1992.

[52] S. Varakliotis, "QoS-enabled streaming of animated 3D wireframe models," Ph.D. Thesis Mar 2004, Dept. of CS, Univ. of London.

[53] B. Watson, A. Friedman and A. Mcgaffey, "Measuring and predicting visual fidelity", Siggraph 2001 August, Los Angeles, CA USA.

[54] A. Witkin, "Scale-space filtering," International Joint Conference on AI, 1983, p. 1019-1022.

[55] Y. Yu, A. Ferencz and J. Malik, "Compressing texture maps for large real environments", Siggraph 2000 Sketch.

[56] Y. Yu, I. Cheng and A. Basu, "Optimal adaptive bandwidth monitoring," IEEE Trans. on Multimedia, September 2003.

[57] A.D. Kalvin, C. Cutting, B. Haddad and M. Noz, "Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures," SPIE Vol. *14*, p. 247–259, 1991.

[58] R. Gonzalez and R. Woods, "Digital Image Processing," second edition, 2002 Prentice Hall p39-42.

[59] M. Zerroug and R. Nevatia, "Part-based 3D descriptions of complex objects from a single image," IEEE Trans. on PAMI, Vol 21, no. 9, Sep. 1999.

[60] M. Reddy, "Perceptually optimized 3D graphics," IEEE Applied Perception, September/October 2001.