Parametric Foveation for Progressive Texture and Model Transmission*

Irene Cheng, Anup Basu and Yixin Pan
Department of CS, University of Alberta, Edmonton, Canada T6G 2E8

Abstract
Spatially varying sensing (foveation) has been used in many different areas of Computer Vision, such as image compression and video teleconferencing and in perceptually driven Level of Detail (LOD) representations in graphics. In this work, we show that foveation is advantageous for interactive mesh and texture transmission in online 3D applications. Unlike traditional mesh representations where all 3D vertices need to be transmitted, we only need to transmit a collection of points-of-interest (foveae) and information on only one (rather than three) axis. Thereby, we can achieve a threefold reduction in the amount of data that needs to be transmitted to represent a new 3D model. Our research differs from level of detail (LOD) based approaches using perceptually driven simplification in that (i) the mesh and texture resolutions vary smoothly and continuously in our approach compared to distinct levels of details in adjoining regions in other foveated or multiresolution LOD based methods; and (ii) the approach works for an integrated foveated texture and mesh representation. The current implementation extends our past research in image and video compression [1] and is restricted to regular grid mesh representation produced by 3D scanners.

1. Introduction

3D visualization is an expanding area of multimedia research covering graphics, imaging and network transmission. With the help of advanced scanning technology, a large volume of range data can be generated for model construction. The trend in multimedia applications is to use more polygons in order to produce more realistic 3D scenes. However, a large number of polygons impose another challenge to researchers in terms of storage, processing, rendering and transmission.

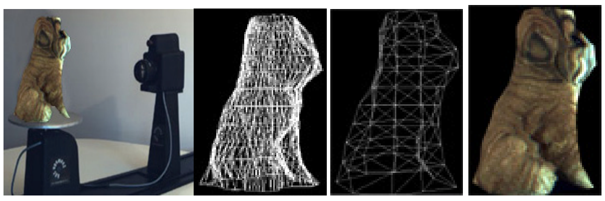


Figure 1: *Range data captured by* the *Zoomage^{TM} 3D scanner: (left) Detailed model with 7200 polygons, (middle,left) mesh and texture mapped object using 180 polygons (right two).*

For most applications, a dense mesh is not necessary; e.g., in Fig.1 when an object is far away a simplified model with 180 polygons (instead of the dense model) is sufficient for rendering without loosing significant details. Many simplification approaches have been introduced in the last decade to reduce unnecessary details [13]. In medical or inspection application, however, it is important to be able to look into the highest detail in scanned data to detect defects or diagnose abnormalities; we address this problem through interactive local updates. Our work is closely related to perceptually driven simplification, progressive meshes, region-of-interest (ROI) processing, selective transmission and user-guided simplification [2,8,11,12,17,20]. Our main contribution is the introduction of a compact parametric method allowing the user to control interactively the variable-resolution (foveated) representation of a photo-realistic texture-mapped mesh. Synthetic texture or color is commonly used in algorithms proposed in other research. Semiautomatic simplifications are discussed in [9,10,14], where simplification is carefully guided by the user assuming no time constraint and not view-dependent. Our simplification process is defined in a mathematical model where time constraint can be specified and that is view-dependent. The user initiates updates by a mouse click, and the appropriate LOD is transmitted. In this paper, we demonstrate an approach based on foveation [1,16,18] that is user driven with low computation and bandwidth requirement specifically for online visualization applications. In these applications, the LOD transmitted is dictated by the bandwidth available and the maximum wait time specified by the viewer.

* The support of an Alberta Science & Research Authority (ASRA), IBM and TelePhotogenics partnership project and a Natural Science and Engineering Research Council (NSERC) Ph.D. scholarship in making this work possible is gratefully acknowledged.

We use foveation in interactive online 3D visualization as follows: Initially a default low-resolution mesh is transmitted from a server to a client. The client maintains a dense mesh on which depth data is interpolated from the initial mesh transmitted. An observer at a client workstation can interactively improve the 3D object displayed by selecting one or more foveae; when a fovea is selected by a mouse click, a foveated mesh with corresponding texture is created at the server and transmitted to the client where the new data is integrated into the existing model. In addition to updates based on user ROI, progressive transmission based on viewpoint during idle periods can also be added to optimize bandwidth utilization. The mesh representation is similar to Normal Meshes [7] and swing wrapper in JAVA.

The advantage of foveation in interactive online 3D visualization include: (i) The ability to provide reasonable quality over a network, such as the Internet, with low and dynamically varying bandwidth; (ii) Fast update of a 3D model based on viewer interaction; (iii) Compact representation of the 3D data transmitted reducing bandwidth utilization; (iv) The ability to adjust the incremental model created and transmitted by a server based on two parameters, thereby ensuring that the time to update a 3D object being displayed does not exceed an acceptable limit. The focus of this paper is on generating an integrated foveated mesh and texture. In past research [1,2,4,12,15] viewing image and video with foveation, foveated meshes and foveated texture mapping was discussed. However, joint texture-mesh foveation and extending the original model in [1] to incorporate cylindrical/spherical foveation was not considered. Note that the word fovea is "loosely" used in this work as exact modeling of the human visual system is not used; we use foveation to mean "variable resolution" or different LODs in a single object. The main application of our work is in interactive viewing of 3D medical images, where the highest detailed scan must be available for a doctor to view if (s)he wishes. Also, the work is meaningful when the mesh has resolution comparable to the texture image; if the storage requirement for a mesh were much smaller than the corresponding texture then it will obviously be simpler to transmit the entire mesh first, and then update the texture progressively.

The remainder of this report is organized as follows: Foveation transform in 360° scanned mesh models is described in Section 2 along with some experimental results. Section 3 describes the data reduction resulting from transmitting foveated meshes. Extending foveation from a 360° cylindrical to a spherical model is discussed in Section 4. Future work and conclusion are outlined in Section 5.

2. Foveated transform for mesh

To minimize bandwidth utilization, a coarse uniform mesh model and a low-resolution texture map is initially transmitted to the client (Figure 2), while the server stores the original detailed geometric model and texture. When a user (client) clicks on a fovea, the fovea coordinates are transmitted back to the server. Based on the fovea and a look-up table, The server generates and compresses the foveated mesh and texture. The foveated mesh and texture image are then clipped using the following two criteria:

(i) Vertices/texels closer to existing foveae than the new fovea are discarded.

(ii) New data are transmitted only if they provide higher resolution of mesh/texture than the coarse model.

Further compression using standard compression algorithms is applied to the foveated data before sending back to the client. The client reconstructs the refined mesh and texture by combining the coarse model and the high-resolution data (around the fovea) received.

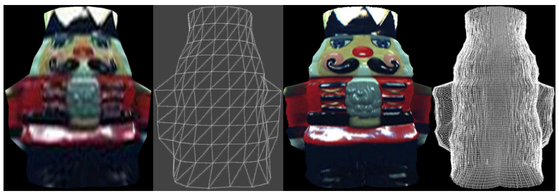


Figure 2: *Front view of coarse & detailed models of Nutcracker toy (Mesh: coarse 20*10, detailed 360*100, and texture: coarse 256*64, detailed 1024*256).*

Implementation details

The foveation transform is defined by two parameters, the scaling factor (s) and distortion parameter (α), that control the distortion at the boundaries of a region of interest in an image or mesh with respect to the fovea. A high α value gives a sharply defined fovea with a poorly defined periphery; a small α value makes the fovea and periphery closer in resolution. Based on this transform, a pixel with polar coordinate (r,θ) is mapped to (v, θ) and vice versa as:

$$v = \ln(r * \alpha + 1) * s \quad (1) \qquad r = \frac{\exp(v/s) - 1}{\alpha} \quad (2)$$

The value s is a scaling factor used to control the overall compression ratio.

$$s = \frac{v_{\max}}{\ln(r_{\max} * \alpha + 1)} \quad (3)$$

The data resulting from the transform above is not rectangular; to simplify several issues we use a rectangular transform, the "Cartesian variable resolution" (CVR) transform [1]. For a given image with the fovea located at (x_0, y_0), for every pixel (x, y) in the original image, one can define the distance in x and y directions as d_x and d_y, respectively, using the equations: $d_x = x - x_0$ (4) $d_y = y - y_0$ (5)

Then, (x, y) is mapped to (x_1, y_1) where:

$$x_1 = x_0 + \ln(d_x * \alpha + 1) * S_x \quad (6)$$
$$y_1 = y_0 + \ln(d_y * \alpha + 1) * S_y \quad (7)$$

In other words, here a pixel is moved from d_x and d_y to d_{vx} and d_{vy} units away from the fovea in the x and y directions:

$$d_{vx} = \ln(d_x * \alpha + 1) * S_x \quad (8) \qquad d_{vy} = \ln(d_y * \alpha + 1) * S_y \quad (9)$$

We implement the foveated compression based on a lookup table using (6) and (7). Note that for manifolds, objects' texture and mesh are cyclic without a vertical boundary. Therefore, fovea in *x* is computed as if it is always located at the center. In other words, (3) and (6) are implemented as (10) and (11) (referred to as *Cylindrical* Foveation).

$$S_x = \frac{width/2}{\ln(r_{\max} * \alpha + 1)} \quad (10)$$

$$x_1 = [x_0 + \ln(d_x * \alpha + 1) * S_x] \bmod width \quad (11)$$

Width is the horizontal resolution. The texture and mesh generated by (10) and (11) are clipped using (12) to reduce unnecessary transmission.

$$|x_i - x_{i-1}| \le R_x \text{ and } |y_i - y_{i-1}| \le R_y \quad (12)$$

R_x, R_y represent resolutions of initial coarse model in x and y direction respectively.

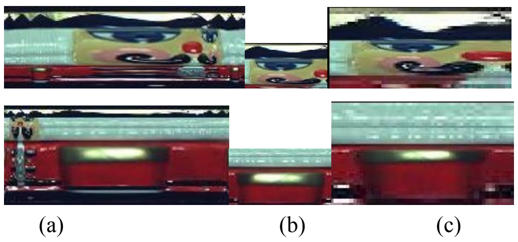
Note that the difference between texture and mesh cylindrical foveation include:

(i) The computation of Z for each mesh vertex.

(ii) Usually, much smaller scaling factors (s_x and s_y)

for mesh compared to texture.

The (X,Y) points computed using the foveated transform may not correspond to any real vertex in the detailed model. Thus depth of vertices (Z) in the refined mesh are computed using bi-linear interpolation of the depths of the 4 nearest neighbor vertices from the detailed model.


(a) (b) (c)

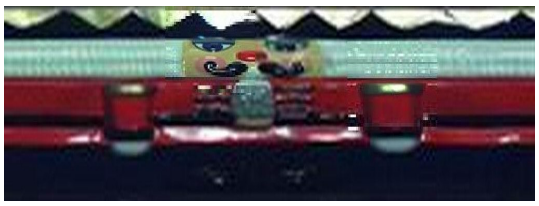


Figure 3: *Refined texture with a fovea on the right eye (top row) and a fovea on the left arm (middle row) (a) Original foveated texture (b) Clipped foveated texture (c) decompression of (b). Bottom: Integrated texture.*

Integration of multiple foveae

Multiple foveae integration is based on accumulating details in a series of transmissions. For each texel/vertex in the map, the ID and distance of the fovea closest to this texel/vertex are stored. When a new fovea is chosen, the distance between this new fovea and each texel/vertex in the image is calculated and compared to the stored distance, if the new distance is shorter, the fovea ID and distance will be updated as shown below:

$$ID_{(x,y)} = i \mid \underset{i}{Min}(|x - x_i| + |y - y_i|) \quad (13)$$

$$Dis_{(x,y)} = \underset{i}{Min}(|x - x_i| + |y - y_i|) \quad (14)$$

where (x_i, y_i) is the ith fovea.

The integration of multiple foveae is shown in Figures 3 and 4, where a second fovea is added to the left arm of the toy.

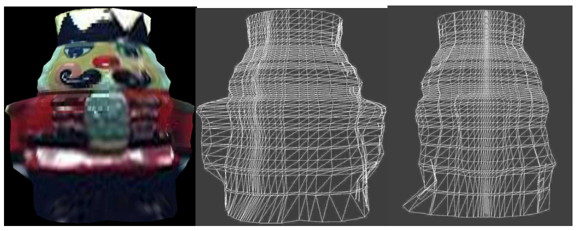


Figure 4: *Refined model with a fovea on the right eye and a fovea on the left arm, scale=95%, alpha=0.5. Left: texture mapped front view; Middle (right): Mesh front (side) view with fovea on right eye (left arm).*

3. Transmission of foveated wireframes

Through a user interface, the viewer controls the distortion parameter (α), which specifies the relative importance of a fovea compared to the periphery. The scaling factor (s) is adjusted to regulate the amount of data transmitted. The assumption is that in online visualization users want updates within a maximum time delay limit T (specified or default). Depending on T and the currently monitored bandwidth (B) [21] the compression ratio and distortion factor are adjusted to update the model in a desirable manner.

Compression of 3D meshes has been discussed extensively in the literature [6]. The advantage of our approach over those proposed in previous research is that complete (X,Y,Z) data representing all the vertices does not need to be transmitted to recreate a mesh at the viewer (client) site during online visualization. We only need to transmit the location and parameters of each fovea specified by a user, and depth (Z) information at estimated vertex locations; since (X,Y) data can be computed at a client from the location and parameters of a fovea. Thus, the amount of mesh related data that needs to be transmitted can be reduced by a factor of 3, compared to a LOD based variable resolution mesh with similar details.

One way of determining the scaling factors for creating the foveated representations is as follows:

Let S denote the scaling factor (possibly two dimensional) for texture and S/N denote the mesh scaling factor, assuming the maximum texture resolution is N times the maximum mesh resolution. Suppose the foveated texture is compressed further using JPEG, and let Q denote the quality parameter in JPEG. Given a fixed distortion factor, let M(S/N) denote the estimated compressed foveated mesh size with scaling factor S/N, and TX(S, Q) denote the estimated compressed foveated texture size given scaling factor S (possibly a 2D vector) and JPEG quality parameter Q. The bandwidth constraint can be expressed as:

$$M(S/N) + TX(S,Q) \leq B \quad (15)$$

In our implementation, we fix M and Q and choose the maximum value of S that satisfies (15). The problem of optimally choosing S, N, and Q to maximize the perceptual quality of the updated view is left to future research. Also, incorporating JPEG2000 features [19] will be considered in the future.

4. Spherical foveated mesh

The discussion so far concentrated on foveated transmission of cylindrical texture and corresponding meshes. For arbitrary 3D objects, say the Nutcracker toy with bottom and top added, the transformation in Section 2 is insufficient to model foveation; e.g., if a viewer clicks on the top of the head all sides of the face should get the same amount of refinement. To extend the model to handle arbitrary convex 3D surfaces, we introduce two angles θ_V & θ_H to represent the mesh vertices in the vertical and horizontal directions respectively. One way of representing a default low resolution mesh for a 3D object is as follows: (i) A center point within the object is chosen as the origin; (ii) θ_V is varied so that we have an even number of equally spaced angles (Figure 5, left), with the top and bottom directions (0 and Π angles) for θ_V included; (iii) θ_H is varied so that we again have an even number of equal angles in a horizontal plane (Figure 5, right), with the purpose of connecting pairs of θ_V on opposite sides except for the top and bottom directions. For example, the points *A* and *B* on Figure 5 (left) along a vertical plane correspond to opposite directions on the horizontal plane Figure 5 (right). The top and bottom directions are used to create two vertices to close the top and bottom of the mesh . Each direction pair (θ_V, θ_H) is extended away from the origin until it intersects the 3D object surface, thereby defining a 3D vertex. Note that the spherical representation makes the update more complicated if a fovea is selected near the poles; an issue we will discuss in detail in future research.

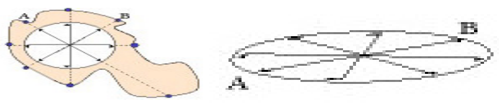


Figure 5: *Creating default mesh for spherical foveation.*

Figure 6 shows two default meshes constructed using the approaches described above. After constructing the default object we need to define a strategy for incrementally updating the model through foveation. Foveated mesh update can be made as follows:

(i) Let ($F\theta_V$, $F\theta_H$) represent the spherical direction corresponding to a fovea clicked on an object by a viewer.

(ii) Create a set of directions of the form: $\{(F\theta_V, F\theta_H), (F\theta_V +/- f(i, s_1, \alpha_1), F\theta_H +/- f(j, s_2, \alpha_2)) | i=1 \ldots m, j = 1, \ldots n\}$, where f() is similar to the function in (2) with $f(m,s_1, \alpha_1) \leq \Pi$ radians & $f(n,s_2, \alpha_2) \leq \Pi$ radians.

(iii) Create an incremental set of foveated vertices by computing (X,Y,Z) on object surface intersecting rays away from origin (object center) in the directions computed in (ii).

(iv) For transmitting a set of incremental vertices, only the Z components in (iii) along with s_1, s_2, α_1, α_2, m, n, and ($F\theta_V$, $F\theta_H$) need to be sent to the client site.

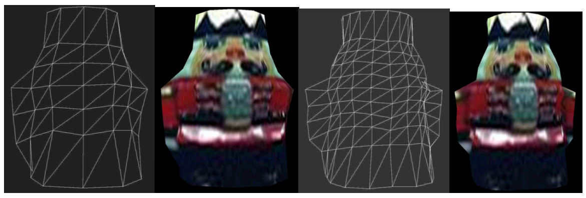


Figure 6: *Default meshes for spherical foveation. Left half: (θ_V, θ_H) = (L*Π/8,K*Π/6); right half: (θ_V, θ_H) = (L*Π/12,K*Π/8); L, K = 0, 1, 2, … .*

5. Conclusion and future work

In this paper we discussed and demonstrated some methods for generating foveated mesh and texture based on interactive specifications of points of interest by a user, through online 3D browsing software. The main advantages of our approach is a fast and simple parametric representation and joint texture-mesh compression based on user-defined ROI. In the near future we plan to demonstrate the software for online viewing of 3D dermatological scans. A high-resolution 3D skin scanner will be used in this application.

Further investigation can be performed in several areas, including: (i) Qualitative assessment [3, 12] of updated models for varying distortion factors; (ii) A strategy for model update during idle times, when the currently available bandwidth is not being utilized to service user requests; (iii) Determining the optimal relative weighting of mesh vs. texture; and (iv) Relating the system parameters to simplification envelopes [5] to measure quality of the viewed model.

6. References

[1] A. Basu, A. Sullivan, and K.J. Wiebe, Variable resolution teleconferencing. *IEEE SMC Conference*, 170-175, 1993 [Extensions in ICPR 1994, and IEEE Transactions on SMC, 1998.]

[2] F.W. Blanke et al. Foveated Rendering of Large Datastreams using a Multi-display, Multi-resolution Parallel Image Compositing System, Tech Report, UT at Austin, 2002.

[3] M. R. Bolin and G. W. Meyer. A Perceptually Based Adaptive Sampling Algorithm," *SIGGRAPH*, 299-309, 1998.

[4] I. Cheng and A. Basu. Efficient visualization of super high resolution 3D images. IEEE 3D PVT Conference Proceedings, Italy, 2002.

[5] J. Cohen, A. Varshney, et al. "Simplification envelopes," ACM *SIGGRAPH,* 119–128, 1996.

[6] P. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. SIGGRAPH 2002, 372-379.

[7] I. Guskov et al., Normal Meshes, SIGGRAPH 2000.

[8] Hoppe, H., View-Dependent Refinement of Progressive Meshes. SIGGRAPH , 1997.

[9] Y. Kho and Michael Garland, User-guided simplification, Proc. ACM I3D, 2003.

[10] G.Li and B. Watson (2001). Semiautomatic simplification Proceedings ACM I3D, 2001.

[11] D. Leubke and C. Erikson, View-Dependent Simplification Of Arbitrary Polygonal Environments, SIGGRAPH'97.

[12] D. Luebke and B. Hallen. Perceptually Driven Simplification for Interactive Rendering. In *Rendering Techniques,* Springer-Verlag, 2001.

[13] D. Luebke et al., "Level of Detail for 3D Graphics", Morgan Kaufmann Publishers, 1st ed., 2002.

[14] E. Pojar and D. Schmalstieg, User-Controlled Creation of Multiresolution Meshes, Proc. ACM I3D, 2003.

[15] T.H. Reeves and J.A. Robinson. Adaptive Foveation of MPEG video. *ACM Multimedia Conference*, 231-241, 1996.

[16] G.Sandini and M. Tistarelli. Vision and space-variant sensing. In *ECCV-94 Workshop,* 398-425, 1994.

[17] D. Schmalstieg and M. Gervautz. Demand-Driven Geometry Transmission for Distributed Virtual Environments. In Proceedings of Eurographics , 421-432, 1996.

[18] E.L. Schwartz. Computational anatomy and functional architecture of striate cortex. Vision Research, 20:645-669, 1980.

[19] D. S. Taubman and M. W. Marcellin. JPEG2000: Image compression fundamentals, standards, and practice. *Kluwer, 2001.*

[20] S. P. To, W. H. Lau and M. Green, A method for progressive and selective transmission of multi-resolution models VRST'99.

[21] Y. Yu, I. Cheng and A. Basu. Optimal adaptive bandwidth monitoring. IEEE Transactions on Multimedia, Sep., 2003.