Fast Rendering of Foveated Volume in the Wavelet Domain

Hang Yu Ee-Chien Chang Zhiyong Huang Zhijian Zheng
School of Computing, National University of Singapore
{yuhang,changec,huangzy,zhengzhi}@comp.nus.edu.sg

1 Introduction

A design issue in remote visualization is on the mapping of the viewer's intention to the relevant data to be sent. One possible approach is to let the viewer interactively indicate the region of interests (ROI), and objects in the ROI will have higher priority during transmission.

For media objects like images and video, *foveation* has been used to formulate the priority of spatial information. Foveation can be treated as a way to distribute information across the space, in order to have wide coverage and yet high concentration in the interesting location. There are a number of works on image transmission and video conferencing exploiting the compression rate provided by foveation [Basu et al. 1993; Basu and Wiebe 1998; Chang et al. 2000]. From another perspective, foveation can be viewed as a way to distribute computing resources across space. For example, Levoy et al [Levoy and Whitaker 1990] propose to use foveation to speed up volume rendering.

In this paper, we presented a novel algorithm for fast rendering volumes with special structure: foveated volumes. Our major contribution is to exploit the implicit structure of the foveated volume, different from other types of data, to accelerate the volume rendering. We render a foveated volume directly in the Wavelet domain, and its running time depends mainly on the number of relevant coefficients. Potentially, foveation can be exploited in remote volume visualization to give a fast rendering that does not depend on the size of the full resolution volume.

2 Notations

In [Chang et al. 2000], a formulation of the "ideal" foveated image is given. Such notion can be easily extended to 3-dimension. Foveation is a process of obtaining a foveated volume from a full resolution volume. It depends on two parameters, the *fovea* $\mathbf{x}$ which is a point in the 3-d space, and *rate* r, the width of the fovea area. One approach to accurately approximate a foveated volume while keeping data size small, is by retaining some coefficients in the Wavelet domain. Let us describe the approximation for images. Similar idea can be applied to volume. Figure 1(a) shows the retained coefficients for the image in Figure 1(b). Coefficients in the shaded squares are retained.

We assume the volume data V is stored in a n^3 array. We denote the foveated volume as $V_{\mathrm{f}}(\mathbf{x}, r)$ and $\mathcal{C}(\mathbf{x}, r)$ is the Wavelet transform of $V_{\mathrm{f}}(\mathbf{x}, r)$.

3 Proposed method

Given the parameter rate r, location of fovea center $\mathbf{x}$, the Wavelet coefficients $\mathcal{C}(\mathbf{x}, r)$ and a view angle θ, we want to compute the rendered image of $V_{\mathrm{f}}(\mathbf{x}, r)$.

One solution to get the rendered image is to first reconstruct V_{f} from $\mathcal{C}(\mathbf{x}, r)$ and next apply direct rendering on $V_{\mathrm{f}}(\mathbf{x}, r)$. This method is costly when n is large. We give an algorithm that avoids reconstructing $V_{\mathrm{f}}(\mathbf{x}, r)$.

3.1 Main idea

Let us describe the main idea using a 2-d example. We want to trace rays in a foveated image along the x-axis as shown in Figure 1(b), giving a 1-d signal as output. In Figure 1(b), a lower resolution sample is depicted as a bigger square, which we call it a *super-pixel* (for volume, we call it a *super-voxel*). Consider a set of rays tracing through a big super-pixel. If the intensity of the rays are the same before hitting the square, then they are also the same upon leaving the square. Thus, from computational aspect, all these rays can be emulated altogether in one step. Since they are the same, we viewed these rays as a *thick* ray, where the thickness is the size of the region it covers.

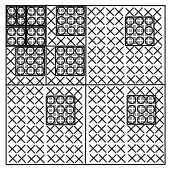
A key observation is that we can always *split* a thick ray, but not merge rays. Considering the situation when a thick ray leaves a square and enters two smaller squares, the ray has to be split into two thinner rays. On the other hand, when two adjacent thin rays leave their respective square and enter a common bigger square, they may not be the same when entering the bigger square as the two rays may be different in intensity. Hence, no computation can be shared.

Figure 1(c) shows how the rays trace half-way through a foveated image. Due to the structure of foveation, we only need to split the rays. Problem arises in the second half of the foveated image if the rays continue to trace toward the right. Since rays can not be merged, in the second half, they have to remain thin. This is not optimal since, intuitively, some computation could be shared in the second half. To overcome that, we trace the rays along x-axis in both directions, as shown in Figure 1(c). The final rendered 1-d signal is the sum of these two sets of rays. We do not set the line where the two set of rays meet as a straight line, because it may cut across a whole square.

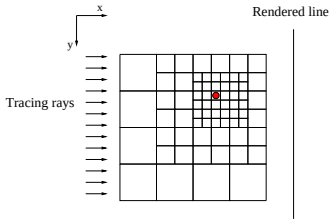
Running time. The computation required is directly proportional to the number of rays, which is the number of super-pixels. Hence, the running time for the 2-d example will be $O(m + n)$ where m is the number of super-pixels and n is the size of the rendered output signal. For volume, the running time will be $O(m + n^2)$ where m is the number of super-pixels, which is approximately the total number of Wavelet coefficients required, and n^2 is the size of the rendered image.

Data Structure. We represent the volume in the Wavelet domain. The super-voxel is not explicitly stored in a spatial structure. We use a data-structure for super-voxels that supports fast reconstruction (i.e. inverse Wavelet transformation) and the rendering process. As a result, only $O(n^2)$ storage space is required. Another issue is on the data structure for thick rays.

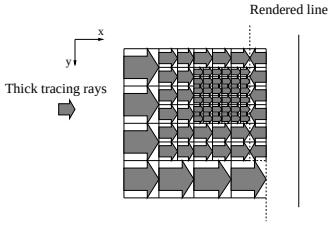
Using non-Haar Wavelet. Our current implementation uses Haar Wavelet. Other types of Wavelets can also be used and they could lead to better approximation of the foveated volume data.



(a) $\mathbf{x} = (11, 5), r = 6$ in Wavelet domain.



(b) $\mathbf{x} = (11, 5), r = 6$ in spatial domain. The red circle shows the location of fovea.



(c) Thick rays rendering.

Figure 1: Wavelet foveation and thick rays rendering.

3.2 Rendering with an arbitrary viewing direction

For simplicity, we only consider rotation of θ degree about the x-axis. We will only give a brief description of our approach. One possible approach to handle rotation is to first obtain the Wavelet transform of the rotated foveated volume [Yu et al. 2004], and apply the orthogonal volume rendering. However, it is not easy to perform a fast and accurate rotation directly in the Wavelet transform. Although the running time of the approximation given in [Chang et al. 2000] is $O(m)$ where m is the number of retained Wavelet coefficients, it still incurs significant overhead. In our algorithm, we incorporate ideas in [Yu et al. 2004] and shear factorization approach [Lacroute and Levoy 1994] into our rendering.

4 Experimental results

We implemented experiments using a data set (a CT scan of a human head with 256x256x225 voxel) on a 2.6GHz Pentium IV PC with 1GB DDR RAM.

Figure 2 shows the rendering result with $\mathbf{x} = (128, 128, 50), r = 50$ and a view angle θ=30. The opacity is set according to the location of the fovea with the highest at the fovea center and decreases to the peripheral. There are only 334643 coefficients required compared to 14745600 coefficients required for the full resolution volume. The rendering rate is 17 fps. Figure 3 shows the rendering time increases as the fovea rate r increases since more Wavelet coefficients are considered as relevant coefficients. The time for rendering with a viewing direction is larger since there are more data to be processed in the shear-warp operation. Figure 4 shows that the rendering time increases as the view angle increases since more data are involved in the shear-warp operation. Figure 5 shows that the rendering time increases as the data width increases. When n is large, the time is proportional to n^2 since m is small and can be omitted. Note that when the width increases from n=1000 to 4000, although the data size increases by a factor of $4^3 = 64$, the rendering time is only increased by a factor of 7.86.

5 Conclusions

We give an algorithm that renders a foveated volume directly in the Wavelet domain. The running time required is $O(n^2 + m)$ where n is the width of the volume (hence, n^2 is the size of the output image), and m is the number of retained wavelet coefficients. Furthermore, our algorithm only requires $O(n^2)$ storage space. We believe that the implementation can still be further improved to achieve more speedup.

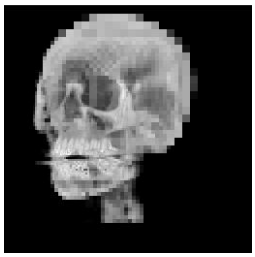


Figure 2: Foveated volume with varying opacity.

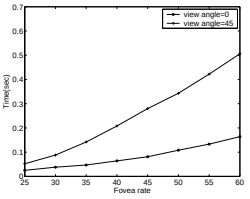


Figure 3: Rendering time versus fovea rate r.

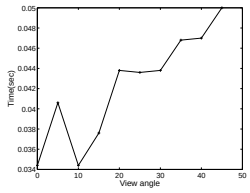


Figure 4: Rendering time versus view angle.

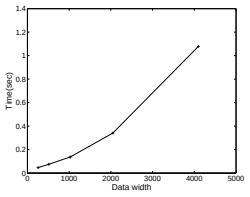


Figure 5: Rendering time versus data width.

References

Basu, A., and Wiebe, K. 1998. Videoconferencing using spatially varying sensing with multiple and moving fovea. *IEEE Trans. on Systems, Man and Cybernetics.*

Basu, A., Sullivan, A., and Wiebe, K. 1993. Variable-resolution teleconferencing. *IEEE System, Man, and Cybernetics.*

Chang, E. C., Mallat, S., and Yap, C. 2000. Wavelet foveation. *Journal of Applied and Computational Harmonic Analysis.*

Lacroute, P., and Levoy, M. 1994. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics.*

Levoy, M., and Whitaker, R. 1990. Gaze-directed volume rendering. In *Computer Graphics.*

Yu, H., Nguyen, V. T., and Chang, E. C. 2004. Rotation of foveated image in the Wavelet domain. *IEEE International Conference on Image Processing.*