Realtime Visualization of Large Images over a Thinwire*†

E.C. Chang and C.K. Yap and T.-J. Yen‡
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012
U.S.A.

Abstract

We investigate realtime techniques for visualization of very large images over a thinwire. We exploit multi-foveated images which have extremely favorable data density, as compared to uniform resolution images. To compensate for the loss of uniform resolution, we now allow the user new dynamic controls over the visualization process. Current servers of large images on the net (e.g., map servers) suffers from a distinctly non-realtime response, its small viewing windows, and visual discontinues in panning/zooming. We have constructed an image server that can overcome these limitations. Our foveation techniques are based on a novel application of wavelets.

1 Introduction

We address the realtime visualization of large scale images over a **thinwire model** of computation. That is, the model comprises an image server and a client viewer connected by a low-bandwidth line. The server may be assumed to have powerful computational resources if necessary; the client has possibly modest resources. Typical examples include image archive servers, terrain viewers and map servers on the world-wide web. Usually, the visualization of large scale images on a thinwire requires the server to precompute many smaller images (say, of size 640x480 pixels) which may show either lower resolution (zoomed-out) views and/or portions of the original image. Clearly this is a suboptimal approach since no preselected set of views can anticipate the needs of all users.

Map servers (e.g., [7], URL `http://www.mapquest.com/`, etc) use an improved approach where the user may zoom and pan over the large image. But current implementations suffer from three problems:

- Visual discontinuities in zooming and panning. One reason is that a brand new image is served up for each zoom/pan request. Another is the discrete nature of the zoom/pan controls.
- Smallness of the fixed-size viewing window (typically about 3"x4.5"). This does not allow much of a perspective.
- Significantly less than realtime response.

Attempting to remove any of these problems might appear difficult under current bandwidth limitations on the web. But our project is able to remove these problems, effectively achieving a realtime and visually-continuous viewing of very large (color) images. The key is to introduce new parameters of control for the user, using the following kind of images:

- Variable resolution over time: this suggests the idea of **progressive transmission**, but ours is a significant generalization of the conventional use of this term (in particular, the user can control the process).
- Variable resolution over space: this refers to images with resolution that varies over its entire extent. We call these **multi-foveated images** because such images need not have only a single foveal region with high resolution.

2 Previous Work

Some forms of the above ideas have been exploited before. Hill et al [2] is one of the early papers that describe many of the features we desire. It has progressive transmission and a form of foveation for a browser of images in an archive. But the realtime requirement does not appear to be a concern. Percival and White [10] describe a progressive transmission technique based on bit-planes that is extremely effective for astronomical data. Reeves and Robinson [8] gives a method to foveate MPEG-standard video in a thin-wire environment. MPEG-standard could provide a few levels of resolution but they consider only a 2-level foveation. The client/viewer can interactively specify the region of interest to the server/sender. Levoy and Whitaker [6] exploits foveation in volumetric visualization.

The production of foveated images using superpixels with fixed geometries have been described by Wallace et al [9] and by Kortum and Geisler [5]. Our wavelet approach is more flexible. Gross et al [4] use wavelets in a way that is remarkably similar to ours, but not for foveation purposes. Kortum and Geisler also conducted psychological experiments in which subjects reported little perceptual difference between foveated and uniform images.

Our system is "interactive" as well as "realtime". What we mean by "realtime" in this paper is that the displayed view are refreshed at video rates in response to a user's panning/zooming commands (albeit, the resolution of some parts may be lower than desired). Again, this idea is not new: it is a key technique in "walkthrough applications", where video rates are achieved by omission of details

*This work is partially supported by an National Science Foundation Grant #CCR-9619846.

†To be presented at IEEE Visualization'97, Late Breaking Hot Topics, accompanied by a video presentation.

‡251 Mercer Street, Courant Institute of Mathematical Science, New York University, New York, NY 10012, E-mail: {eechien, yap, yentj}@cs.nyu.edu

(e.g., [3]). What is new is that our user can now request higher resolution at any part of the visual field just by positioning the mouse cursor there.

3 Contributions of this Paper

Some features of our work are:

- its unique combination of realtime, thinwire and large-scale image in one setting.
- new degrees of user-control provided for realtime visualization of images.
- demonstrating a new standard of performance that can be achieved by large-scale image servers on the world-wide web.

One of our current demos serves a $4096 \times 4096 \times 24$ color image (of a map) which is just over 50 MB after preprocessed for our server. Our basic objectives would not be impacted even if the image size grows by some orders of magnitude. Our preprocessing of this image lead to a slight compression of the original image because of the use of a wavelet transform. But it is important to realize that image compression is a secondary issue for us. We can run our server and clients on a non-dedicated standard workstation or a Pentium class machine. One of our experimental setups has the server running in SUNY Stony Brook and our client program at NYU. We have tested clients on standard Solaris and SGI workstations (with plans to port this to a notebook setting).

The client has a resizable **viewing window**. There is a small, auxiliary **navigation window** whose main purpose is to let the viewer know the size and position of the viewing window relative to the whole image. But clicking in this window will also make the viewing window jump to the clicked location. See appendix for screen shots of these two windows.

The client has the following capabilities: panning, zooming, foveating, varying the foveal resolution. We plan to allow the modification of the shape and size of the foveal region. The user can "foveate" as follows: with the mouse buttons released and the cursor within the viewing window, the image will gain increasing resolution at the positions of the moving cursor. But when the left button is held down, the view will pan continuously, following the mouse. Zooming is not (yet) continuous but in scales of 2. One might think these capabilities as problematic in thin-wire scenarios. But in fact, it is precisely these capabilities that make realtime visualization in thinwires feasible: they allow the user to pin-point areas of interest within (a view of) the entire image. The variable resolution can allow the server to *dynamically* adjust the amount of transmitted data to match the effective bandwidth of the network.

The relation of this paper to [1] is as following. In [1], we introduced our based foveation techniques and investigated basic properties of foveation. The present paper bring these techniques to bear in thinwire setting.

4 Architecture

We have a server holding a large image and, at the other end of a thinwire, we have a client that wants to view this image. We assume that this image does not fit into main memory, and hence we have to organize its representation in secondary memory.

Data Representations. The storage of the image on the server side involves an initial preprocessing step. To use wavelet-based multi-resolution techniques, we begin by constructing and storing a **wavelet transform** of the image. See [1] for details. We currently use the Haar wavelet – some of our techniques exploit properties of this choice although this is by no means essential. Entries in the wavelet transform matrices are called **coefficients**, and these are what is communicated over the thinwire. The transformed image is partitioned into blocks which are stored in individual files in secondary memory.

Next, we address the storage of the same image on the client side. This storage structure is necessarily sparse: initially, nothing is known about the image, but as data is requested and received over the network, the representation fills up. The client side stores the image in each of its levels of resolution (this is clearly redundant). For reference, we call this the **pyramid representation** of the image.

To co-ordinate between the client and server, the server and client each maintains a common **mask** indicating which coefficients have been sent or received. The mask on the client side is used for deciding what coefficients to request. On the server side, the mask is useful for deciding which block of the image can be released from main memory. Why do we have different representations of the images on the server and client side? A partial answer is that the wavelet transform representation, when stored in files in the tiff format, generally gives a better compression. It also gives a better compression ratio for the transmission across the net. But the pyramid representation allows faster display on the client side, because the client doesn't have to reverse-transform an image every time the image displayed.

The Server. The server currently serves each data request at "full level of service" (it serves the exact number of coefficients requested). In general, the server may decide if some request should be satisfied a lower level of service. We are investigating "online competition algorithms" to do this. Also, we have two modes for sending data: either to use (lossless) channel compression or not to use it. In our current experiments, we found no "visible" difference in performance between the two modes. It is probably because the additional time in online compression/decompression masks any gain in transmission speed. But if we were to perform this experiment over a very slow link, eg. a modem, then compression would become useful.

The Client. The client is constructed with three threads:

network, display, manager.

The network thread is responsible for sending and receiving requests from the server. It also currently perform the function of converting any received coefficients into the pyramid representation. The display thread is responsible for both input and output: converting viewer requests to requests to the network thread, and for updating the display windows. There are two modes for display: to always fill the pixels to the highest resolution that is currently available or to fill them up to some user specified level.

The manager thread is the brain of the client program. One of its functions is to convert viewer requests from the display thread into data requests to the network thread. For instance: if the viewer is foveating at a location (x, y), then this converts into requesting data at several levels of resolution (see wavelet techniques below). But some of this data is already available locally, and under the thinwire assumption, it is critical not to ask for the same data again.

To achieve real-time responses when the user makes any "zooming" or "panning" action, after translating the actions into data requests to the server and waiting for the server response. Instead, the display thread would first use the data existing locally, and when new data arrive from the server, the manager thread would then notify the display thread to update the image.

5 Sparse semi-dynamic representation of large matrices

In general, the matrices in the client data structures are sparsely populated. But they are also dynamically changing in the sense that additional entries are continually filled in. Currently, we do not discard data from our matrices and so what we need might be called a "semi-dynamic sparse data structure". This assumes that the client has enough secondary memory to store the entire image. Before we address the sparseness and incremental data issues, we consider just the problem of large size. This solution is in fact used on the server side, where the entire image needs to be stored.

We assume that original image is $N_X \times N_Y$. E.g., $N_X = N_Y = 2^{12}$. We define a size parameter, denoted `BLOCK`. We partition our matrices into chunks of `BLOCK` bytes. Each chunk is stored in its own file. In fact, all the three color components of a chunk are together stored in a single file in tiff format. On the server side, these chunks are wavelet transforms while on the client side these chunks are parts of the pyramid representation. The automatic data compression of tiff is therefore more effective on the server side. Since these chunks store matrix elements, we further have two parameters `BLOCK_X` and `BLOCK_Y` where `BLOCK_X`$\times$`BLOCK_Y=BLOCK`. Typically, we have `BLOCK_X=BLOCK_Y=128`. We assume that $N_X/\mathtt{BLOCK\_X} = N_Y/\mathtt{BLOCK\_Y} = 2^k$ for some integer $k \geq 1$. For sparse matrices, we use an additional **chunk map** which tells us for each block whether it is empty or not. This chunk map is implemented by Boolean matrix.

6 Conclusion

Visualization is ultimately a cognitive process in which it is not useful to send to the eye more information than what the brain can process. We were able to achieve "realtime thinwire large-scale visualization" by selecting the information to be transmitted more deliberately. In visualization applications, we can rely on the user to tell us where she or he wants to see greater details (and thus dispense with eye-tracking). In effect, *we have greatly reduced the bandwidth from server to client, in exchange for a very modest increase of bandwidth from the client to the server.*

- Currently, our server would transmit about 500 to 650 KB per minute on a network which has raw transmission (measured with ftp) about 4 MB per minute. This is because on a fast network, the speed of data transmission is much fast than the speed of the input of user requests. Since our system is driven by user input, the server is actually spending most of its time waiting for requests from the client. The next generation of our system will allow the user to set parameters controlling the behavior of the server and client, to be optimized for a variety of settings and platforms. In particular, the manager thread needs to be smart about changing network conditions, and local display capabilities.

- How "scalable" are our techniques with respect to the three properties of our system (thinwire, realtime, large-scale)? Thinwire restriction: we are currently constructing a version of our system for a modem-based client. Some performance hit is expected. Realtime constraint: there are some research issues for improving the quality/resolution of images under this constraint. Large-scale requirement: we believe that this aspect is the most scalable in our current system. E.g., serving an 30"x50" image of the USA map is estimated to require over 2 GB of storage on the server side but should have little impact on the performance of our current system.

References

[1] Ee-Chien Chang and Chee K. Yap. A wavelet approach to foveating images. *ACM Symp. on Computational Geometry*, 13:397–399, 1997.

[2] Jr. F.S.Hill, Jr. Sheldon Walker, and Fuwen Gao. Interactive image query system using progressive transmission. *Computer Graphics*, 17(3), 1983.

[3] James Helman. Graphics techniques for walkthrough applications. In *Interactive Walkthrough of Large Geometric Databases, Course Notes 32, SIGGRAPH '95*, pages B1–B25, 1995.

[4] Markus H.Gross, Oliver G. Staadt, and Roger Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2), 1996.

[5] Philip Kortum and Wilson S. Geisler. Implementation of a foveated image coding system for image bandwidth reduction. In *Human Vision and Electronic Imaging, SPIE Proceedings Vol. 2657*, pages 350–360, 1996.

[6] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *Computer Graphics*, 24(2), 1990.

[7] Steve Putz. Interactive information services using world-wide web hypertext. *Computer Networks and ISDN Systems*, 27(2):273–280, 1994. Originally in: 1st Int'l Conf. on the World-Wide Web, Geneva, May 25-27, 1994. See `http://www.parc.xerox.com/istl/projects/www94/iisuwwwh.html`.

[8] T.H.Reeves and J.A.Robinson. Adaptive foveation of mpeg video. *Proceedings, 4th ACM International Multimedia Conference*, 1996.

[9] R.S. Wallace, P.W. Ong, B.B. Bederson, and E.L. Schwartz. Space-variant image processing. *Intl. J. of Computer Vision*, 13(1):71–90, 1994.

[10] R.L. White and J.W. Percival. Compression and progressive transmission of astronomical images. In *SPIE Technical Conference 2199*, 1994.

Appendix: Images

Here we provide some screen snapshots of the viewing window and the navigation window. The full image represents a map of the subway system of New York City. The full size of the image is 3151×3793 with 24-bit colors.

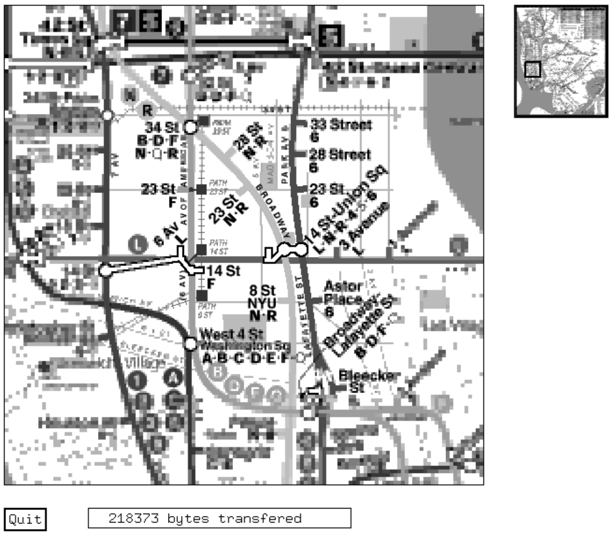

Figure 1: Screen snapshot of a foveated image in the viewing window

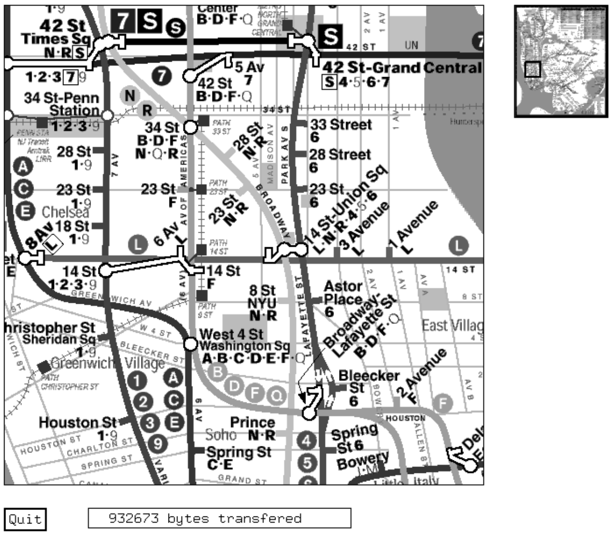

Figure 2: Same area as in Figure 1 but with full resolution in the view window

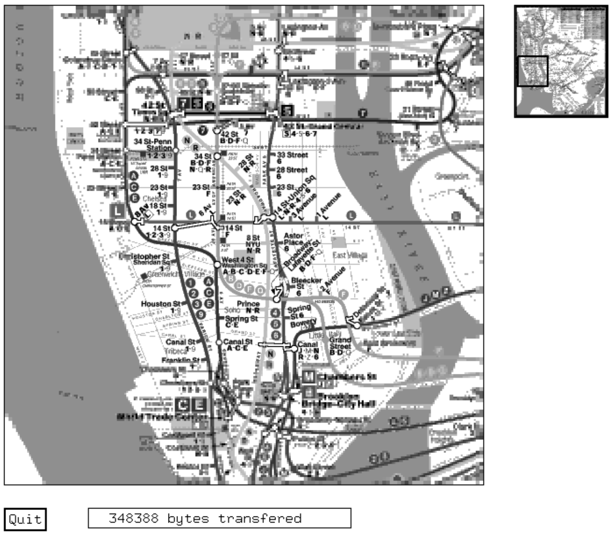

Figure 3: Lower resolution level