Digital Dynamic Telepathology – the Virtual Microscope *

Asmara Afework†, Michael D. Beynon†, Fabian Bustamante†, Angelo Demarzo, M.D.‡, Renato Ferreira†, Robert Miller, M.D.‡, Mark Silberman, M.D.‡, Joel Saltz, M.D., Ph.D.† ‡, Alan Sussman, Ph.D.†, Hubert Tsang

†UMIACS and Dept. of Computer Science
University of Maryland
College Park, MD 20742

‡Department of Pathology
Johns Hopkins Medical Institutions
Baltimore, MD 21287

The Virtual Microscope is being designed as an integrated computer hardware and software system that generates a highly realistic digital simulation of analog, mechanical light microscopy. We present our work over the past year in meeting the challenges in building such a system. The enhancements we made are discussed, as well as the planned future improvements. Performance results are provided that show that the system scales well, so that many clients can be adequately serviced by an appropriately configured data server.

1 Introduction

The Virtual Microscope system provides the ability to access high power and high resolution digital images of entire histopathology slides, assemble individual image panes into a seamless composite, store the image data into a specially tailored database, and sample the dataset for display onto a local or remote client. The display systems are designed to emulate, as much as possible, the actual movement of a slide across a stage, including short latencies between movement and focus or between objective change and focus. In this paper, we concentrate on how the system manipulates and displays high power, high resolution histopathology datasets.

The Virtual Microscope employs a client/server architecture. The client software runs on an end user's PC or workstation, while the database software for storing, retrieving and processing the microscope image data runs on a variety of platforms. The database software can run on a PC at the end user's site, or on a potentially remote high performance parallel or distributed computer. The database software is further decomposed into two parts – a *frontend* process that accepts queries from clients and one or more *backend* processes that store and retrieve the slide image data required to satisfy a query, and return the data to the client. In an earlier paper [1], we discussed in detail the original design of the system, and described several of the challenges in meeting the system requirements. In this paper, we discuss the redesign of the various components to enable concurrent access by multiple users to a slide database with reasonable response time. We also describe additional functionality that we plan to incorporate into the system, including both enhancements for ease of use and for modes of behavior that are not possible with a conventional physical microscope.

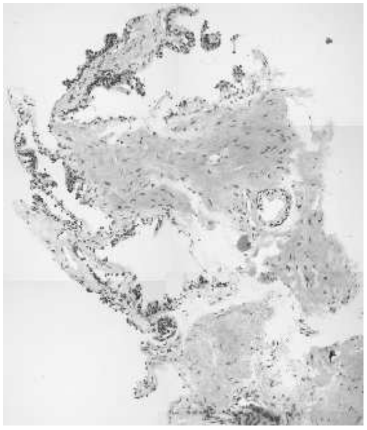

Figure 1: 2x3 slide scan showing prostate tissue at 200x

The main difficulty in providing Virtual Microscope functionality is efficiently dealing with the extremely large quantities of data required to represent a large collection of slides. For example, using a digitizing microscope available at Johns Hopkins Hospital, the scan of a single spot produces a grid of 1000x1000 pixels, and many scans are required to cover the entire slide. In practice, digitizing the specimen shown in Figure 1 at 200x magnification (real cases will require much higher power) requires approximately 70x50 scans, which amounts to several GBytes of data for a single focal plane, and many specimens require multiple planes.

*This research was supported by the National Science Foundation under Grants #ASC 9318183 and #ACI-9619020 (UC Subcontract #10152408), and DARPA under Grants #N66001-97-C-8534 and #DABT 63-94-C-0049 (Caltech Subcontract #9503). The Maryland IBM SP2 hosting the system was provided by NSF CISE Institutional Infrastructure Award #CDA9401151 and a grant from IBM.

Telepathology imaging is the process of examining specimens remotely, of which there are two forms: *static* and *dynamic* [2]. In dynamic mode, the remote pathology consultant can control the stage and actively select the image to be viewed. In static mode, the referring pathologist captures a small set of images to be transmitted to the consultant. A variety of static, dynamic and hybrid forms of telepathology have been demonstrated [3, 4]. In these terms, the Virtual Microscope is best described as a form of completely digital dynamic telepathology.

The Virtual Microscope represents a larger class of applications that involve browsing large multi-dimensional datasets, for example analyzing remote sensing data or visualizing the output of a scientific simulation. A common characteristic of these applications is compact requests for data, with resulting data replies that can be several orders of magnitude larger. Moreover, the amount of data processed by the server in order to produce the reply can be much larger than the reply itself. We are building a set of database services for applications with these characteristics, within our Active Data Repository project [5]. The Virtual Microscope system will then effectively become one customized instance of a database system implemented with the Active Data Repository services.

2 Modifications and enhancements

Communication Protocols As the set of Virtual Microscope system components grew and became more complex, we recognized the need for well-defined communication protocols between the different parts of the system. There are two distinct protocols - one between the client and the server frontend and one between the server frontend and backend processes. The client/server protocol describes how clients interact with the frontend and backend processes of the server. The second protocol is the internal interface used by the server to communicate between the frontend and backend processes.

From the point of view of a client, the system works as shown in Figure 2. The client sends a Client Query (CQ) packet to the server frontend process. The frontend process sends a Client Query Acknowledge (CQA) to the client indicating the number of backend processes that will be sending replies for the query. A Reply (R), containing the image data requested, is sent by each participating backend process to the client. The internal server operation, also shown in Figure 2 starts with a set of incoming CQ's. These queries are transformed into Front Queries (FQ) and sent to the participating backend processes. The backend processes receive the FQ's, read the data from disk, and send R's to the client indicated in the FQ packet.

This detailed protocol specification has allowed us to independently develop new implementations of parts of the system. For example, we have been able to create

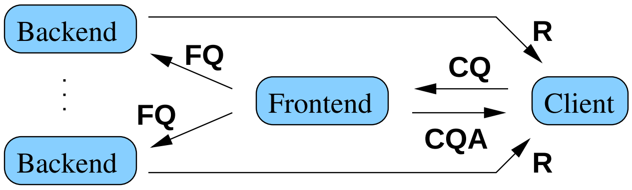

Figure 2: Client Query Control Flow

new client implementations and additional middle-ware implementations (located in the control path between a client and a server), while assuring interoperability with the other components in the Virtual Microscope system.

Client The Virtual Microscope clients are image browsers, with which users can select a slide and navigate through it. After selecting a slide, a user receives a *thumbnail image*, which is a low-resolution overview. A slide can be viewed at any of several available magnifications. A user has two ways of moving through the slide. She can either drag the view box, as shown in Figure 3, across the overview image, or use the fine control buttons to move the view a small amount in any of the four possible directions. When the control buttons are used, the client will only request the data that is not already being viewed, resulting in smaller requests than for dragging actions. In response to the user's actions, queries are generated by the client and sent to the server.

The prototype client implemented in Java [6], as described in [1], now provides two separate windows, one for the control panel and one for the display panel. In this way the user can easily resize the display area. Figure 3 shows a screenshot of the new Java client. The functionality as perceived by the user is the same as in the prototype, except that the display window is continuously updated when the user is panning through the image. To accomplish fast panning, we employ a thumbnail image that is larger than the one displayed in the control window. The client scales the thumbnail on the fly to update the display area, while the user is scanning the slide. Once the user stops dragging and releases the mouse button, a query is generated to the server, and the display area is updated with the full image data from the server. We used the standardized protocol described earlier in the new client.

We have also written a Microsoft Windows client using Visual C++ that presents the same user interface as the Java client. The purpose of building this second client is twofold. First, we wanted to test the correctness and robustness of the standardized protocol by implementing a completely different client using the same protocol, and the C++ client exposed flaws in the design and implementation of the protocols. Second, we wanted to see the performance benefits that might be gained from a compiled client as opposed to the interpreted Java client. We have

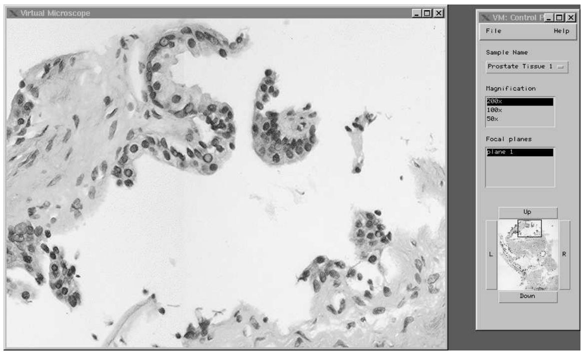

Figure 3: Virtual Microscope client user interface

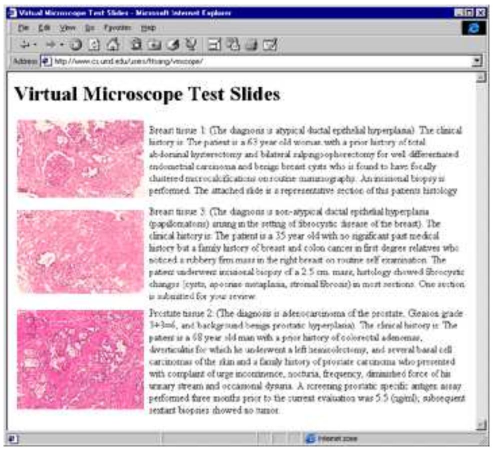

Figure 4: Sample Virtual Microscope slide set web page

not yet been able to quantify the performance benefits, but will do so in the future.

Server There have been many changes to the Virtual Microscope server in the past year. We briefly mention two of them. First, the prototype Java network server was rewritten in C for performance reasons, and is now called the server frontend process. Second, the frontend process no longer stitches together the multiple image blocks that are returned from the server backend processes into the image requested by the client. Instead, the backend processes communicate directly with the clients, and the client does the stitching, eliminating a major bottleneck on the critical data path for returning data to the clients.

In the server backend processes, we intended to use wavelet compression techniques [7] to efficiently store and retrieve data from disks. However, there are several problems associated with a naive use of wavelet compression techniques, the most important being the artifacts that can be generated at the boundaries of each chunk of the image that is stored independently (the image data is blocked to utilize the bandwidth available from multiple disks in the system). In order to effectively use wavelet compression to store the data, much care and effort will have to be put into the implementation of the compression technique, so we have deferred compressing the data.

Web interface The Virtual Microscope client is intended to be used to access slide data remotely over the World Wide Web (WWW). Our initial implementation of this functionality allows the client to act as a *helper application* for a web browser. Therefore, a user viewing a web page containing slide thumbnail images can start the Virtual Microscope client by simply clicking on a hyperlink, assuming that the client is installed on the user's machine. Both versions of the client can be used as helper applications: the Java client runs on any platform that has a Java environment installed and the C++ Windows application runs on any PC running Windows95 or WindowsNT. An example of a web page with several sample slides is shown in Figure 4. While our current implementation requires the client to be installed on the user's machine before accessing a web page, we are creating an applet version of the Java client. This version will then be able to be downloaded and executed as a direct result of activating the hyperlink for a slide. This model of execution avoids many of the version control problems inherent in having the client installed on the user machine, by allowing each web site to also provide the current version of the client Java code along with access to data on that server.

3 Server performance

Performance evaluation of the Virtual Microscope focuses primarily on two metrics. Since it is a highly interactive system, the first order concern is how quickly the system responds to a request from any client. We call this measure *client response time*. A second order concern is how well the system utilizes its available resources. This indicates how well the system will perform as the number of clients is increased. Therefore our second metric, of more concern to a system designer and implementer, is *server utilization*, which measures how effectively the system is utilizing the available computational, storage, and network resources.

The client driver The first step towards running experiments with the Virtual Microscope server is building a driver program that is capable of emulating the behavior of multiple users running the Virtual Microscope client. We first collected event traces from an experienced user working with the system. Based on those traces we produced a statistical high level model of user behavior.

In our model, the user begins at the top left portion of the slide, at low power, and follows a regular scanning pattern over the slide, which can be either top to bottom or left to right. No queries are generated during this scan, since in this scan the user is relying on the expanded thumbnail

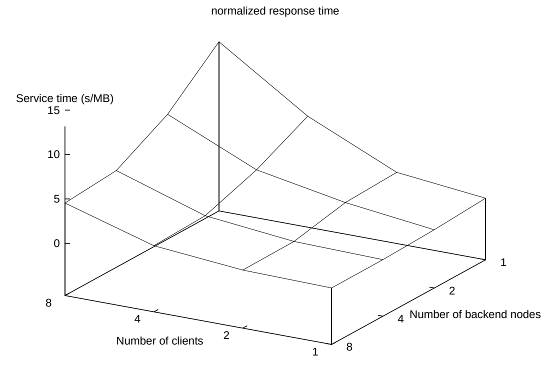

Figure 5: Average normalized response time at each client

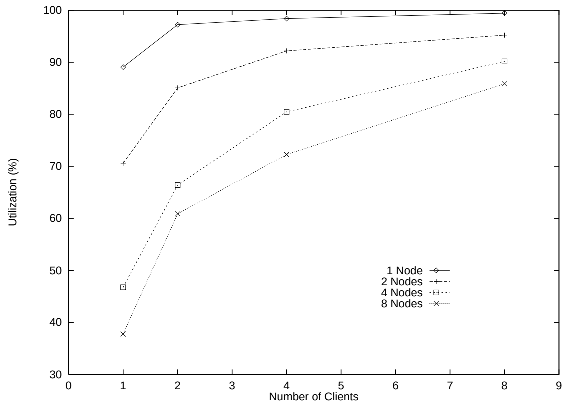

Figure 6: Average utilization of the backend processes, with curves for different numbers of backend processes

image generated on the client from local memory. When the user reaches an *interesting* region of the slide, she stops scanning, and a query at low magnification is generated for that region. The user will then zoom in (increase magnification) and pan around that region for some time, before returning to the standard scanning pattern.

The speed at which the user scans through the slide, how long the user stays in a given region of the slide (which models users *think time*), and what magnifications users use are all parameterized in our model. Therefore, we can easily simulate different types of users, essentially performing the same task.

Results For the experiments, the server frontend process and all backend processes were run on the Maryland IBM SP-2 parallel computer. Each backend process ran on a separate processor and only accessed data residing on the disks attached to that processor. Each client driver ran on a separate Digital Alpha workstation, and client/server communication is over a 10Mb/sec Ethernet.

We collected the response time for each request from the client driver program. Figure 5 summarizes the results. The response time, give in seconds per megabyte (s/MB), is normalized by the size of the query (query size is a measure of how much data has to be processed to answer to the query) and then averaged over all queries for that configuration of client and server processes. The average response time is shown on the z-axis for each combination of number of clients and number of backend processes. As the number of clients generating queries to the server increases, the number of server processes (and the number of available disks) has to increase or client response time suffers. The figure shows that the worst response time corresponds to many clients serviced by one backend process and the best response time corresponds to one client being served by many backend processes.

In addition to client response time, we were also interested in the utilization of the resources available in the server machine. We measured the server utilization factor, which is the ratio between the time the server spends doing useful work and the total run time of a set of queries. Figure 6 shows the average utilization for all backend server processes as the number of clients is varied. A server with only a single backend process is almost fully utilized with a small number of clients. As we increase the number of backend processes (and correspondingly the amount of available resources), we see that the overall utilization drops, implying that more clients can be supported with reasonable response time. This result justifies our effort in parallelizing the server.

4 Future plans

In this section we consider what can be done to improve the system in various ways. Some improvements are to make the Virtual Microscope operate more closely like a physical microscope, while others are to enable operations that are not possible with a traditional physical microscope.

4.1 Usability enhancements

Given our model of client usage, we analyze how the speed at which the sweeping progresses varies due to the content of the slide at the current location. The amount of image data visually processed by the user is lower when sweeping quickly than for sweeping slowly. The idea is that interesting areas can be easily spotted while sweeping at a higher speed. Once an interesting area is identified, sweeping slows down or even stops to increase the amount of perceivable image data. Once the closer examination at higher magnifications is finished, the magnification is reduced and sweeping continues.

A common complaint for the current implementation of the Virtual Microscope is that it takes too long for the full resolution image to be displayed, and that the image appears blocky while dragging. We examine two approaches to alleviate these problems, while at the same time addressing the physical microscope usage pattern. Both are a form of speculative prefetching, where the system tries to *guess* future queries to ensure that image data is already present at the client when needed.

Higher resolution thumbnail The main problem with a blocky thumbnail image is the resolution is too low. Increasing the thumbnail resolution is not an attractive solution because it requires the client to hold the larger thumbnail in memory. A better scheme would hold only the currently relevant portions of the thumbnail in client memory. A method for doing that would be to request compressed blocks from a higher resolution thumbnail store on either the server or the local client disk.

Streaming Streaming involves continuously querying the server for the portion of the slide that is just beyond view in the current sweep direction. Based on the position and directed velocity of a client, the server must decide what image data to send, and at what resolution. Spatially, it is clear that the portion of the image ahead of the current position along the path of the directed velocity is a good choice. In addition, the magnitude of the velocity vector can be used to determine the resolution of the image data that should be returned. With fast sweeping the amount of data that can be visually processed by the user is smaller, so low resolution image data is adequate. Conversely, for slow sweeps high resolution image data would be returned.

4.2 Functionality enhancements

Image Analysis Histologic image analysis [8] comprises morphometry, densitometry and neural networks. Programs designed to carry out the relevant analyses can be run on the Virtual Microscope server. All these analyses embrace quantitative mathematical descriptions of histologic features of tissues with differing methods or rules for interpretation. The morphometric and densitometric descriptions can be subjected to a rule-based expert system analysis using a neural network. After creating a set of rules, the network is trained on case material and errors in predicted outcome are weighted and used to retrain the system. In anatomic pathology this technique has been applied in cytopathology, specifically in exfoliative cervical cytology where it is approved for re-screening.

The marriage of high end image processing and high end computational capacity constitutes a very powerful research tool. A system of this type can be used to carry out multiple-antibody, simultaneous, quantitative tissue immunohistochemistry. The current state of the art largely consists of qualitative evaluations and only allows for single or at most doubly labeled histologic sections. Additionally, three-dimensional reconstruction of tissue sections characterized by a series of confocal planar micrographs taken through a three dimensional tissue specimen will be possible.

Exposing client activity We plan to instrument the client so that it can inform the the server about exactly what section of the slide is being viewed and where the user has pointed the mouse. Certain regions of the slides will also be marked and categorized as tumors or other growths. The combination of these two additional features will allow the server to give feedback about the sections of the slides that users are viewing. This will work in a similar fashion to image maps on web pages. These features will be used in an educational context, where users could be asked to find a tumor or certain growths on a slide, and the server can provide feedback to users about their performance on a particular task, including giving hints about how to go about performing the task.

5 Conclusions

We have presented the features of the Virtual Microscope system that been changed and enhanced since we completed our original prototype. We have provided evidence that shows the redesigned system scales well, in that adding more processing power (and corresponding disk storage) to the server allows the system to serve more clients with reasonable response time.

Acknowledgments We would like to thank Seungho Cho for building the C++ version of the client.

References

[1] R. Ferreira, B. Moon, J. Humphries, A. Sussman, J. Saltz, R. Miller, and A. Demarzo. The Virtual Microscope. In *Proceedings of the 1997 AMIA Annual Fall Symposium*, pages 449–453. American Medical Informatics Association, Hanley and Belfus, Inc., October 1997.

[2] Ronald S. Weinstein, A.K. Bhattacharyya, Anna. R. Graham, and John R. Davis. Telepathology: A ten-year progress report. *Human Pathology*, 28(1):1–7, January 1997.

[3] Michael Weinstein and Jonathan I. Epstein. Telepathology diagnosis of prostate needle biopsies. *Human Pathology*, 28(1):22–29, January 1997.

[4] S. Olsson and C. Busch. A national telepathology trial in Sweden: Feasibility and assessment. *Arch Anat Cytol Pathol*, 43:234–241, 1995.

[5] C. Chang, A. Acharya, A. Sussman, and J. Saltz. T2: A customizable parallel database for multi-dimensional data. *ACM SIGMOD Record*, 27(1):58–66, March 1998.

[6] Ken Arnold and James Gosling. *The Java Programming Language*. The Java Series. Addison-Wesley, 1996.

[7] John J. Benedetto and Michael W. Frazier. *Wavelets: Mathematics and Applications*. Studies in Advanced Mathematics. CRC Press, 1994.

[8] V. Russack. Image cytometry: current applications and future trends. *Crit Rev Clin Lab Sci.*, 31:1–34, 1994.