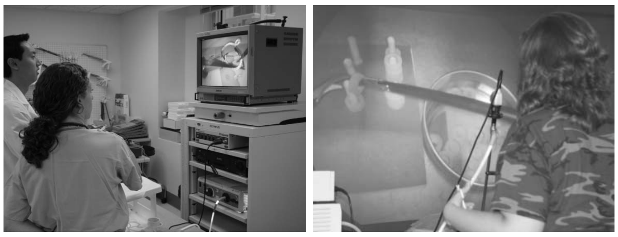
Implementation and Analysis of Scalable Display Architecture for Laparoscopy

C. Andy Martin, Qiong Han, Duncan Clarke, C. Melody Carswell, Adrian Park and W. Brent Seales [*]

Abstract

*We report on a display system to support laparoscopy by loosely coupling input sensors (*e.g.*, cameras) with outputs (*e.g.*, image displays). A low-latency and general-purpose computing platform is inserted between the traditionally tightly-coupled inputs and outputs of a laparoscopic surgical environment. The architecture of the proposed system provides flexibility in the fusion and augmentation of multiple input data sources and in the display of the data on a scalable large display. The performance of the described system is measured. The result indicates that our system has great potential to improve the performance of laparoscopic surgeons.*



(a) Current Surgical Trainer (b) Scalable Display Trainer

Figure 1. Views of (a) the existing laparoscopic training environment and (b) a prototype of a training environment based on a large-scale flexible display.

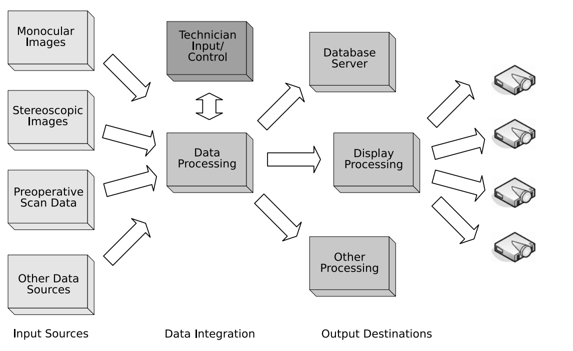
1 Introduction

The current laparoscopic surgical environment consists of disparate data sources that are manually integrated by a surgical team. A laparoscopic surgeon replies on a tightly-coupled camera and monitor system for visual feedback as he manipulates surgical tools [1]. The current camera and monitor system in a laparoscopic surgery creates an information bottleneck and offers a limited field of view (FOV) to the surgeon than a traditional open surgery. Additional monitors are used to display real-time sensory data, such as pulse and heartbeat, and other sources of visual data, such as preoperative 2D and 3D imagery (*e.g.*, CT, MRI, and X-ray images) [4]. The surgeon must integrate all this information in real-time as the surgery moves forward. As a result, it is usually more challenging to acquire the necessary skills for laparoscopic surgery than those for traditional open surgery [3].

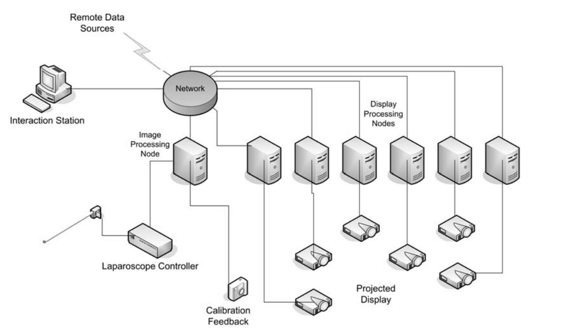
Research has been conducted to overcome the challenge brought upon surgeons by laparoscopic surgery. The areas of research include plug-and-play smart instruments, image-guided techniques augmented with preoperative image data, and intelligent display and storage of data. Some of the work contributes to what has been termed ”the operating room of the future” [8, 17, 18]. (Figure 1) shows a comparison between the current surgical training environment and a scalable display environment. The latter provides more and configurable display area. The benefit of a large working display area for performing virtual-reality and spatial tasks has been well documented [2, 21, 22]. There is also a large body of research documenting the technical aspects of such casually aligned projector systems [5, 6, 12, 15, 16].

Developments in the architecture and organization of high-performance general-purpose computer systems have been largely ignored by the technology infrastructure of the modern laparoscopic surgical suite. [19] proposed to combine large displays, visualization tools, image enhancement, and ergonomic analysis in an integrated system for the laparoscopic environment. The key for this system to function well in a surgical setting is a display architecture with enough bandwidth and little latency to include future high-density data sources such as high-definition video and large 3D data sets, while integrating disparate data sources and displaying them on a scalable display infrastructure. We followed the architecture described in [19] to build a scal-

*This work was supported by project REVEAL, funded by DoD TATRC grant DAMD17-03-2-0015. C. A. Martin, Q. Han, D. Clarke, C. M. Carswell, and W. B. Seales are with the Center for Visualization and Virtual Environments, University of Kentucky. Emails: cam@myfastmail.com, qhan@engr.uky.edu, dclarke@freemontassociates.com, cmcars00@uky.edu, seales@uky.edu. A. Park is director of the Maryland Advanced Simulation, Training, Research and Innovation (MASTRI) Center. Email: apark@smail.umaryland.edu



(a) Conceptual Architecture



(b) System Architecture of Prototype

Figure 2. The system's (a) conceptual architecture based on information flows and (b) prototype architecture to realize this conceptual architecture.

able and information-rich display system. Our main goal is to remove the information bottleneck present in current laparoscopic surgery display system (specifically the tightly coupled camera and monitor system) while providing automatic data integration by a consolidated and unified scalable platform, which would be invaluable to the surgical team.

However, there is little published documentation of the actual implementation and performance of such systems in their intended application environment (such as, in our case, the OR) and analysis of the fitness of these systems for use in their intended domain. In this paper, we describe the implementation of a scalable and information-rich display system in detail, measure the performance characteristics of the system, and analyze the implication of the results on the suitability of the system for use in laparoscopic surgery.

2 System Architecture

The conceptual architecture [19] of our system is demonstrated in Figure 2(a). The system consists of a centralized data processing node to process input data and to output processed data to various output destinations, including a local storage device for archiving, a display processing node controlling a projector array, and other processing devices. The data processing node is controlled by a technician through a software interface. When outputting aggregate visual data, a seamless display is created from scalable and loosely-aligned projectors based on past research efforts [12, 15, 16]. The architecture of the system is summarized next, including its hardware infrastructure and software system.

2.1 Hardware Infrastructure

As shown in Figure 2(a), the heart of our system is the data processing node. This node must have the hardware capabilities of processing data from multiple sources and sending the processed data to various output processing devices. The input data include pre-operative image data, monocular or stereoscopic video images captured in real-time from laparoscopes, and other data sources, such as a patient's vital information. The output data is then integrated for display on a projector array or archived for postoperative analysis. The direct output from the data processing node needs to be further processed by the display processing node to provide a seamless and integrated display to the surgical team via an array of casually-aligned projectors.

The system architecture of our prototype system, realizing the conceptual architecture, is shown in Figure 2(b). In our current implementation, the data processing node is an Intel x86-based computer system with a high-bandwidth low-latency PCIe system bus.

The data processing node is connected via a gigabit-switched IP network to a cluster of display processing nodes. These display processing nodes are housed in a server rack outside of the OR to ease access restrictions and address ambient noise limitations of the OR environment. Each display processing node has a gigabit network interface, a $3.2GHz$ Intel Xeon processor, and a high-end NVidia graphics card. The graphics card is attached to a Dell DLP multimedia projector with a native resolution of 1024×768. The projectors are mounted on a truss structure in the OR and pointed at a rear-projected screen with a usable display area of approximately 20 square feet. This represents an order of magnitude improvement over the typical $2 - 3$ square feet available on flat-panel LCD displays currently used in the clinical setting.

At this moment, our prototype implementation of the display processing cluster of separate nodes has been merged and replaced by a single workstation due to recent advances in motherboard technology and multiple output graphics cards. It is now possible to perform all rendering on a single multi-core workstation, greatly simplifying hardware design, power, and networking requirements.

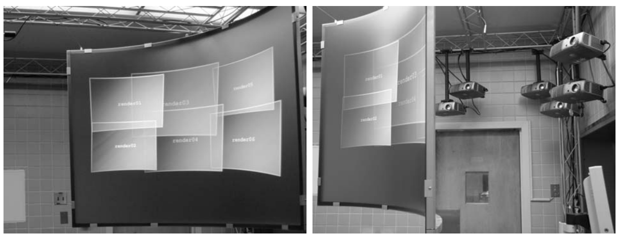

(a) Pre-calibrated Display (b) Projector Array

Figure 3. Pre calibrated projector array output. This shows the raw output of each of the display processing nodes. The initial display of each render node is set to a pre-rendered background image containing the node's host name.

The laparoscope used in this installation is a Stryker 888, a 3-chip CCD camera with analog NTSC video outputs. An NTSC camera with full manual control is used to calibrate the projector array.

Two views of the prototype screen and projector assembly of the University of Maryland Medical Center prototype system are shown in Figure 3.

2.2 Software System

The hardware infrastructure of our prototype system Figure 2(b) supports a software system that distributes digital imagery, video, and other data sources to various output devices.

To provide image and video distribution functionality, the *Chromium* project [10] is used to network and distribute image data across display processing nodes. Chromium is an OpenGL wrapper library which allows OpenGL applications to be distributed across a set of computing nodes by modifying the OpenGL stream with stream processing units (SPU's) (OpenGL is a standard API for describing 2D and 3D rendering of visual data). The Chromium distribution includes standard SPUs, and the SPU API is well documented so custom SPUs may be implemented.

Our Chromium configuration uses the standard *tilesort* SPU, which divides up OpenGL calls from the data processing node by geometric tiles and sends the separate streams to their corresponding display processing nodes. The display processing nodes then use the standard *render* SPU combined with a custom *warp* SPU (detailed in Section 2.2.2), creating a seamless display via the projector array.

We use the software system proposed in [11] (an extension of an earlier system described in [6]) that runs on top of the Chromium installation. This software system consists of four parts: (1) an OpenGL calibration program that calculates the relative projector geometries; (2) the Chromium *warp* SPU that provides warping and blending based upon the calibration result; (3) an OpenGL client/server image processing framework called SmartImage; (4) a Java based GUI to enable simple configuration and interaction with the whole system from an interaction station. All the 4 parts are detailed as follows.

2.2.1 Calibration Program

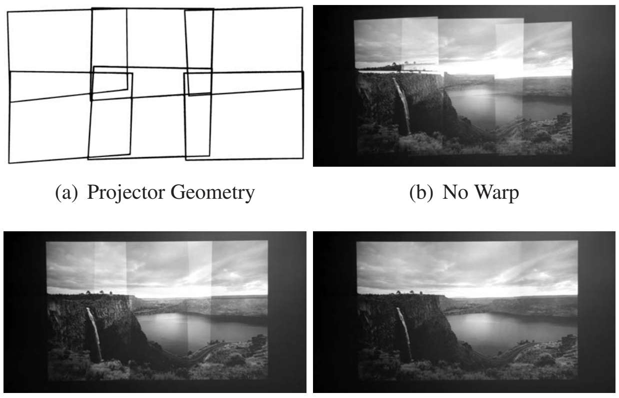
The calibration program is based on a structured-light algorithm that uses circular fiducials, described in [6], which is fast and allows for projecting on non-planar surfaces such as curved or imperfect screens. Projectors parameters are calibrated, and blending alpha masks are generated for each projector from the observed mesh of fiducials based on the technique in [15]. The observed fiducials, the calibrated projector parameters, and the computed alpha masks are saved for later use.

The calibration camera must have an unobstructed view of the entire scene, and the aperture, shutter speed, and focus must be manually adjusted for ambient lighting conditions and brightness levels of the projectors. The aperture is usually fixed to full-open, and the shutter speed is fixed to $1/60 second$ to prevent color separation from the color-wheel inside the DLP projector, which runs at $120Hz$. The video images must be consistent in aperture and shutter speed as an initial background image is sampled and averaged from the first few frames of video and is then subtracted from each following image to prevent the calibration algorithm from being confused by static objects in the camera view.

2.2.2 Warping and Blending

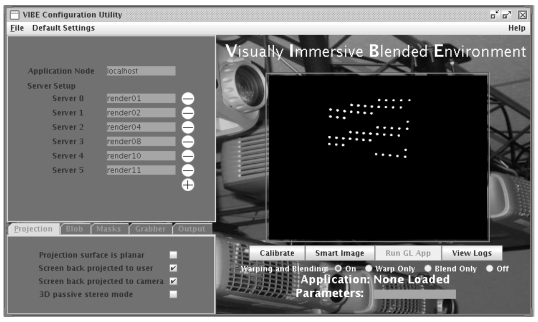
The *tilesort* SPU is configured in logical tiles, which contain a bounding box of the scene data that is applicable to each display processing node. The scene data in the tiles is a clipped OpenGL stream for that particular bounded tile. The *warp* SPU injects the OpenGL commands necessary to warp the final rendered scene onto the calibrated projected mesh by reading the previously saved calibration information [6]. Finally, the *warp* SPU adds OpenGL commands to apply the alpha mask generated for the particular projector in the calibration. This process yields a real-time, seamless image from the perspective of the calibration camera.

Figure 4 shows a projected geometry with no warping or blending correction applied, with only the warp correction applied, and with both the warping and blending applied to provide a step-wise illustration of the correction process.


(a) Projector Geometry (b) No Warp

(c) Warp Only (d) Warp and Blend

Figure 4. Blending and Warping: (a) raw projector relative geometry, (b) image being displayed on the projectors with both the warp and the blend disabled, (c) the same image being displayed with just the warp applied, and (d) both warping and blending being applied to generate a seamless, large-area display.


VIBE Configuration Utility
File Default Settings Help

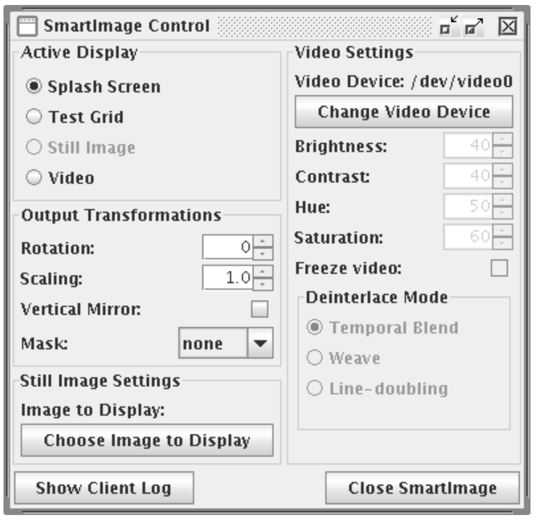
Visually Immersive Blended Environment

Application Node: localhost
Server Setup
Server 0: render01
Server 1: render02
Server 2: render04
Server 3: render08
Server 4: render10
Server 5: render11

Projection | Blob | Masks | Grabber | Output

Projection surface is planar ☐
Screen back projected to user ☑
Screen back projected to camera ☑
3D passive stereo mode ☐

Warping and Blending: On / Warp Only / Blend Only / Off
Application: None Loaded
Parameters:


SmartImage Control

Active Display
[✓ Splash Screen] / Test Grid / Still Image / Video

Output Transformations
Rotation: 0
Scaling: 1.0
Vertical Mirror: ☐
Mask: none

Still Image Settings
Image to Display:

Video Settings
Video Device: /dev/video0

Brightness: 40
Contrast: 40
Hue: 50
Saturation: 60
Freeze video: ☐

Deinterlace Mode
[✓ Temporal Blend] / Weave / Line-doubling

Figure 5. Software interface: a user at the interaction station can control all aspects of the system. A calibration is running in this image, and the observed structured light pattern is visible.

2.2.3 Image Framework

The image framework uses a client/server model. The two communicate via both standard TCP/IP network and shared memory model for efficient image transfer when the client and server reside on the same machine. The framework is designed to be flexible and general enough to allow integration of various image-data sources. It is also possible to keep the client/server model more open by using the ”web and grid” service technology. However, the latency needs to be carefully evaluated under such circumstance, which is beyond the scope of this paper.

Several clients have been implemented: (1) a test grid pattern viewer; (2) a still image viewer; (3) a video file viewer; (4) a real-time video-capture client. The test pattern is used for testing and demonstration. The image and video file viewers are used to visualize high-resolution preoperative data. The real-time video capture viewer is used to display video sources, such as a laparoscopic camera, over the projector array.

The video-capture client includes an efficient software deinterlacer that uses a linear-blending mechanism to generate each deinterlaced frame with low latency at NTSC framerate of $29.97 frames per second (fps)$.

The server receives each set of image data and splits the image into 64×64 pixel chunks for efficient tile-sorting of the image data. The Chromium *tilesort* SPU can only reject sending a texture to a particular node by examining its bounding box, and if the entire image were sent in one large chunk the tile-sorting mechanism would send all of the image data to all of the nodes.

2.2.4 Software Interface

A screen shot of the graphical user interface (GUI) is shown in Figure 5. The GUI is a Java application that runs on an interaction station and communicates with the main data processing node via a network connection. The GUI allows full control of the entire system including running a calibration, launching generic OpenGL applications to run over the projector array, launching the image processing server and clients, controlling the calibration parameters, and controlling the image processing server and clients. It can be used to disable or enable warping and/or blending, to scale, rotate, and mirror an output imagery, to select which real-time video stream to display, and to alter video capture parameters such as brightness or contrast. It can also be used to display the structured light feedback read from the calibration camera during calibration.

3 System Performance

The performance of our system was measured in terms of data throughput and latencies to assess its usefulness in the surgical setting. We analyzed the throughput of the system, the overall latency of the system, and the latencies of individual components.

3.1 Image Throughput

End-to-end image throughput can be measured by the maximal sustainable frame-rate. Our system maintains a constant $29.97fps$ throughput (limited only by the frame rate of the input NTSC video). The frame rate is measured by the data processing node, which can monitor how many frames it renders per second through the Chromium pipeline. Based on the frame rate, the total raw image bandwidth available to the system is at least $221Mbps$ (640×480 images at 24 bits per pixel at $29.97fps$).

It seems feasible to increase the frame rate because the theoretical bandwidth of the network is $1000Mbps$. However, a few existing factors determine that it is not possible to achieve a frame rate much higher than $29.97fps$. No higher frame rates were attempted in our current implementation due to the following factors.

Firstly, $29.97Hz$ is the maximal available capture rate for NTSC input streams. Secondly, the bordering scheme, which is applied to each image chunk to make chunk borders invisible when anti-aliased, adds a 10% overhead of one perimeter to each chunk, increasing the required bandwidth to $260Mbps$.

Thirdly, there is an overhead due to overlap in display tiles used by the *tilesort* SPU, which is necessary for seamlessly ”stitching” multiple projected images. Given a fixed image size, the number of duplicate chunks sent to the projectors is proportional to the amount of overlap between each pair of adjacent projectors. In the current implementation, tile-sorting and duplicate data chunks account for about a 100% increase in distributed data for typical projector overlap (like that shown in Figure 4). The overheads can go above 150% for more significant overlap. Therefore, the $260Mbps$ bandwidth is increased to $520Mbps$ for an average amount of overlap and $650Mbps$ for more significant overlap.

Lastly, due to synchronization, OpenGL constraints, and network design, Chromium is not able to fully utilize the theoretical bandwidth limit $1000Mbps$. The practical limit is roughly $600Mbps$. Therefore, achieving significantly more than $29.97fps$ would not be feasible with the current implementation.

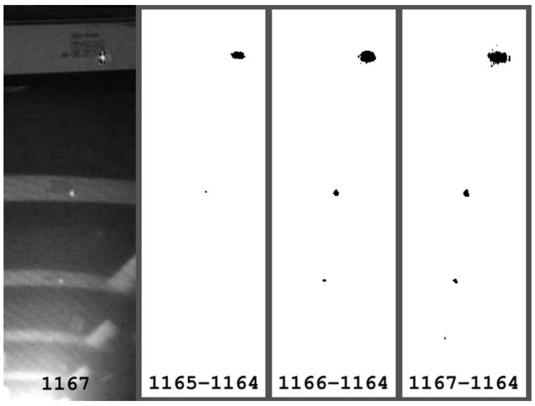


Figure 6. Images of a laser dot captured for the measurement of latency in the Stryker 888 camera and controller.

3.2 Overall Latency

The overall latency was measured by viewing a digital stop-watch through the laparoscopic camera and projecting the video feed of the same stop-watch through the system. This measurement can corroborate our individual component latencies since the overall latency should equal the sum of the component latencies.

A high-shutter-speed still camera was used to take 10 pictures of the original stopwatch and the projected stopwatch image fed through the system. The difference between the two stopwatch time is the overall system latency. The standard error, SE, is defined as $SE = \sqrt{s^2/n}$, where s is the standard deviation and n is the number of observations. In the form of $mean \pm standarderror$, the measured overall latency is $120 \pm 13.3ms$.

3.3 Component Latencies

Each component of the system contributes to the overall system latency. The five main components are the laparoscopic camera and controller, the capture card and capture software, the client/server image processing software, the Chromium distributed rendering framework, and the projectors themselves. The latency of each component is measured and reports as follows.

Camera latency: the latency of a Stryker 888 laparoscope was measured by directly attaching the S-Video output of the camera, which is the input source used in our system, to a high-end medical-grade CRT monitor (Panasonic MT1980) with negligible signal conversion latency. The camera was aimed at the monitor to create several feedback loops and then clamped in place. The entire scene is recorded, and a red laser was shone on the top rim of

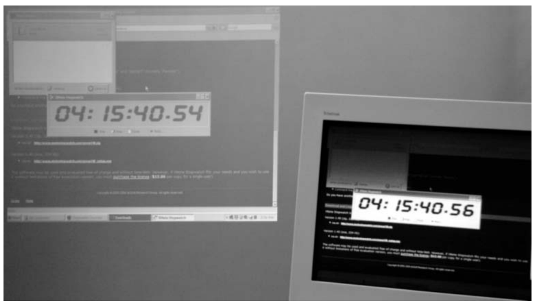


Figure 7. Measurement of latency inherent in a Dell 2300MP projector (left) by comparison with a CRT monitor (right).

the monitor such that it is visible in at least three feedback loops of the system. Figure 6 demonstrates one sample from video frames $1165 - 1167$. The image on the left is of the final frame 1167; the second image is a difference $1165 - 1164$ between the first frame the laser appeared 1165 and the immediately previous frame 1164; the third image shows the next difference $1166 - 1164$; the last image is the difference $1167 - 1164$, showing that the laser dot is visible in all three feedback loops. The laser was switched on and off so the transitions could be observed. The average latency of the camera is analyzed from 15 samples as $16.3 \pm 1.48ms$ (average of 1.46 frames over 3 feedback loops).

Capture latency: this was measured in the frame grabber driver in the Linux kernel. It was defined as the amount of delay between the detection of a $VSYNC$ signal on the analog input indicating the end of a field and the instant when the completed frame was actually returned to the user process. The analysis did not begin until the frame grabber reached a steady state. The latency was measured over 30 steady frames as $66.0 \pm 0.37ms$.

Application latency: this was measured as the delay between image capture in the frame buffer card and the instant when the image frame was handed off to the Chromium layer for processing via a function call to *glutSwapBuffers()*. This latency was measured over 30 stable frames as $6.652 \pm 0.033ms$.

Chromium latency: the latency in the Chromium tile-sorting, distributed rendering, and warping steps was measured by finding the amount of time spent in the OpenGL *glutSwapBuffers()* function call. This routine returns only after every display processing node has finished rendering the entire scene. Again, 30 frames of stable data were collected, and the latency was measured to be $18.53 \pm 0.011ms$.

Projector latency: two identical outputs of a digital clock image from a laptop were simultaneously distributed to a high-end computer CRT monitor with a negligible latency and a target Dell 2300MP projector. A digital camera with the shutter speed set to $1/125$ was used to capture 15 random image pairs of the CRT monitor and the projected images, one of the image pair is shown in Figure 7. The shutter speed is chosen to be slightly more than half of the $60Hz$ refresh rate used by the VGA signal, the same vertical refresh rate used in the system as a whole. The latency of the projector was measured to be $15.3 \pm 2.15ms$.

3.4 Summary

Figure 8 shows the latencies of all components and the entire system. The sum of all the component latencies yields a calculated total latency of $121.6 \pm 3.85ms$. Recall that in Section 3.2, the overall latency was reported to be $120 \pm 13.3ms$. Thus, the component latency measurements are validated by the overall latency as their sum falls neatly within the error range of the overall latency measurement.

The investigation into the component latencies also shows that the overall system latency can be improved by eliminating the digital-to-analog conversion in the laparoscopic camera. By keeping the signal digital, both of capture latency and projector latency can be greatly reduced.

4 Conclusion and Future Work

According to published work, for the display component of a video display system to be usable in real-time, it should have a latency of no more than $200ms$, and a frame rate of at least 10 frames per second [7,9,13,14,20,23,24]. Pausch argues that low latencies are more important than either high resolution or stereoscopic vision [14]. Research on the effects of system delays on users' visual-motor performance and perception find clear negative effects at $250ms$ and above ([7, 24]). Furthermore, for tasks, in which a high degree of precision is required, even smaller latencies might have detrimental effects. MacKenzie and Ware [13] found the effects of display lag to be multiplicative when varied with the Fitts' Law index of difficulty. In their studies, a $75ms$ delay was associated with a 16% increase in movement time and a 36% increase in errors. A $225ms$ delay was associated with a 65% and 214% increase in the same two measures.

Our system maintains the full NTSC frame rate of $29.97fps$ and has a moderate latency of about $120ms$, well below the $200ms$ threshold [14]. Also, it performs real-time deinterlacing of the NTSC video stream thereby improving image quality by removing interlacing artifacts that are readily visible on large and high-resolution video screens.

The large and scalable display size of the system can be used to provide the surgeon a more scalable and flexible environment in which to perform laparoscopic surgery

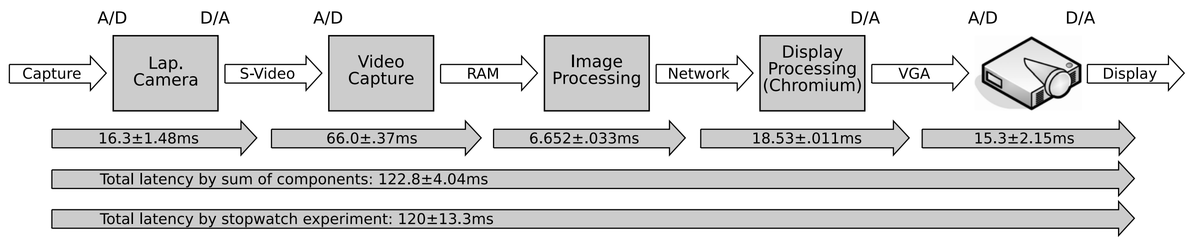


Figure 8. Comparison of measured component latencies and overall latency.

or surgical training. The system can be used to provide new functionality to the laparoscopic environment by decoupling the laparoscopic camera and monitor and by inserting a flexible architecture in-between, which can integrates disparate data sources and output mechanisms. The performance of our system has the potential to fuse and display real-time video from the laparoscopic camera and other high-resolution static imagery such as preoperative scans.

The current system implementation can be improved in several ways. After this manuscript was originally written, the display processing cluster of separate nodes has been replaced by one single server with a newer motherboard, which can have up to 4 video cards installed, with a maximum of 4 video outputs per card (supporting 16 projectors in total). Combined with a quad-core processor, the whole system can be driven by one server, which further eliminates another $10-20ms$ of latency by removing the network bottleneck.

Furthermore, more data sources and more output mechanisms can be added to the architecture to increase spatial and multi-modal integration; higher resolution video imagery can be leveraged to increase the image quality, making the images more vivid and realistic; new hardware technologies can be incorporated to our system such as faster internet topologies and digital video interfaces that avoid analog-to-digital and digital-to-analog conversions to further reduce the system latency.

Ongoing research is also being conducted to evaluate the impact of our system on surgeons, especially those in training, performing various simulated tasks. Our preliminary results have shown that the proposed system has great potential to eventually improve surgeons' performance.

References

[1] Surgical video systems used in laparoscopy. *Health Devices*, 24(1):28–33, January 1995.

[2] P. Baudisch, N. Good, V. Bellotti, and P. Schraedley. Keeping things in context: A comparative evaluation of focus plus context screens, overviews, and zooming. *CHI Letters*, 4(1), April 2002.

[3] R. Berguer, D. L. Forkey, and W. D. Smith. Ergonomic problems associated with laparoscopic surgery. *Surgical Endoscopy*, 13:466–468, 1999.

[4] N. Bitterman. Technologies and solutions for data display in the operating room. *Journal of Clinical Monitoring and Computing*, 20:165–173, 2006.

[5] M. Brown, W. Seales, S. Webb, and C. Jaynes. Building large format displays for digital libraries. *Communications of the ACM*, 44(5), 2001.

[6] M. S. Brown and W. B. Seales. Incorporating geometric registration with PC-cluster rendering for flexible tiled displays. *International Journal of Image & Graphics*, 4(4):653–681),, October 2004.

[7] C. Eberst, N. Stoffler, M. Barth, and G. Farber. Compensation of time delays in telepresence applications by phot-realistic scene prediction of partially unknown environments. In *Proceedings of the International Conference on Robotics and Applications*.

[8] D. M. Herron, M. Gagner, T. L. Kenyon, and L. L. Swanström. The minimally invasive surgical suite enters the 21st century. *Surgical Endoscopy*, 15:415–422, 2001.

[9] E. Hoffman. Fitts' law with transmission delay. *Ergonomics*, 35(1):3–48.

[10] http://chromium.sourceforge.net/. Chromium homepage.

[11] https://halsted.vis.uky.edu/downloads/download.php. VIBE: Visually Immersive Blended Environment.

[12] C. Jaynes, W. B. Seales, K. Calvert, Z. Fei, and J. Griffioen. The Metaverse–a collection of inexpensive, self-configuring, immersive environments. In *7th International Workshop on Immersive Projection Technology/Eurographics Workshop on Virtual Environments, Zurich, Switzerland*, May 2003.

[13] I. MacKenzie and C. Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 488–493, 1993.

[14] R. Pausch. Virtual reality on five dollars a day. In *Proc. of the CHI '91 Conference ono Human Factors in Computing Systems*.

[15] R. Raskar, M. S. Brown, R. Yang, W.-C. Chen, G. Welch, H. Towles, W. B. Seales, and H. Fuchs. Multi-projector displays using camera-based registration. In *IEEE Visualization*, pages 161–168, 1999.

[16] R. Raskar, J. van Baar, and J. X. Chai. A low-cost projector mosaic with fast registration. In *Asian Conference on Computer Vision (ACCV)*, January 2002.

[17] D. W. Rattner and A. Park. Advanced devices for the operating room of the future. *Seminars in Laparoscopic Surgery*, 10(2):85–89, June 2003.

[18] W. B. Seales and J. Caban. Visualization trends: applications in the operating room. *Seminars in Laparoscopic Surgery*, 10(3):107–114, September 2003.

[19] W. B. Seales and D. Clarke. Computing support for information-rich laparoscopy. *Surgical Innovation*, 12(4):357–363, December 2005.

[20] C. Smith, T. Farrell, S. McNatt, and R. Metreveli. Assessing laparoscopic manipulative skills. *American journal of surgery*, 181:547–550, 2001.

[21] D. S. Tan, D. Gergle, P. Scupelli, and R. Pausch. Physically large displays improve performance on spatial tasks. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(1):71–99, March 2006.

[22] F. Tyndiuk, V. Lespinet-Najib, G. Thomas, and C. Schlick. Impact of large displays on virtual reality task performance. In *3rd International conference on computer graphics, virutal reality, visualization and interaction*, pages 61–65, 2004.

[23] C. Ware and R. Balakrishnan. Reaching for objects in VR displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction*, 1(4):331–356, 1994.

[24] B. Watson, N. Walker, W. Ribarsky, and V. Spaulding. Effects of variation in system responsiveness on user performance in virtual environments. *Human Factors*, 40:403–414, 1998.