Understanding Performance in Coliseum, An Immersive Videoconferencing System

H. HARLYN BAKER, NINA BHATTI, DONALD TANGUAY, IRWIN SOBEL, DAN GELB, MICHAEL E. GOSS, W. BRUCE CULBERTSON, and THOMAS MALZBENDER
Hewlett-Packard Laboratories

Coliseum is a multiuser immersive remote teleconferencing system designed to provide collaborative workers the experience of face-to-face meetings from their desktops. Five cameras are attached to each PC display and directed at the participant. From these video streams, view synthesis methods produce arbitrary-perspective renderings of the participant and transmit them to others at interactive rates, currently about 15 frames per second. Combining these renderings in a shared synthetic environment gives the appearance of having all participants interacting in a common space. In this way, Coliseum enables users to share a virtual world, with acquired-image renderings of their appearance replacing the synthetic representations provided by more conventional avatar-populated virtual worlds. The system supports virtual mobility—participants may move around the shared space—and reciprocal gaze, and has been demonstrated in collaborative sessions of up to ten Coliseum workstations, and sessions spanning two continents.

Coliseum is a complex software system which pushes commodity computing resources to the limit. We set out to measure the different aspects of resource, network, CPU, memory, and disk usage to uncover the bottlenecks and guide enhancement and control of system performance. Latency is a key component of Quality of Experience for video conferencing. We present how each aspect of the system—cameras, image processing, networking, and display—contributes to total latency. Performance measurement is as complex as the system to which it is applied. We describe several techniques to estimate performance through direct light-weight instrumentation as well as use of realistic end-to-end measures that mimic actual user experience. We describe the various techniques and how they can be used to improve system performance for Coliseum and other network applications. This article summarizes the Coliseum technology and reports on issues related to its performance—its measurement, enhancement, and control.

Categories and Subject Descriptors: H.4.3 [**Information Systems Applications**]: Communications Applications—*Computer conferencing, teleconferencing, and videoconferencing*

General Terms: Algorithms, Design, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Telepresence, videoconferencing, view synthesis, 3D virtual environments, performance measurement, streaming media, network applications

1. INTRODUCTION

For decades, videoconferencing has been sought as a replacement for travel. Bandwidth limitations and the accompanying issue of quality of the enabled experience have been central to its delayed arrival. Resolution and latency lead the way in objectionable factors but, were these resolved, close behind would come the issues that separate mediated from direct communication: the sense of co-presence, access to

Authors' addresses: Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304-1120; email: {harlyn.baker, nina.bhatti,donald.tanguay,irwin.sobel,dan.gelb,mike.goss,bruce.culbertson,tom.malzbender}@hp.com.

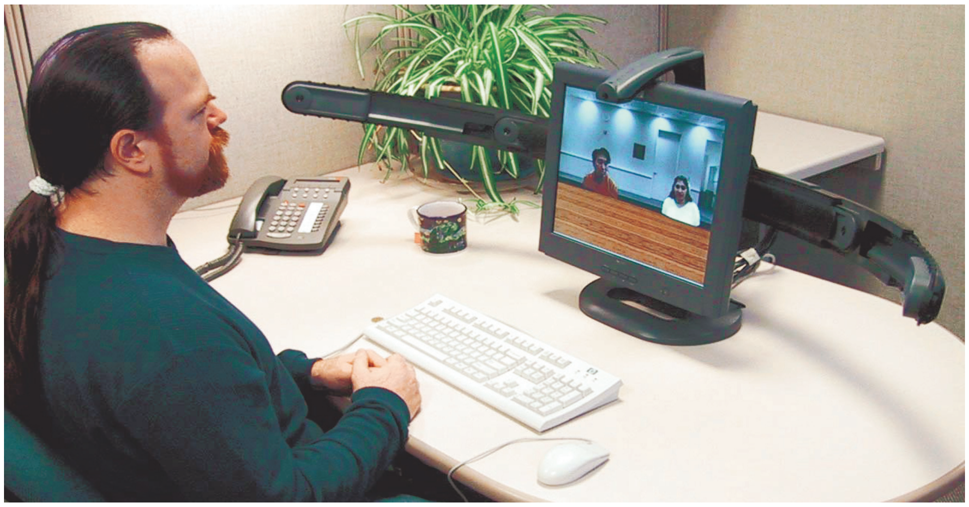

Fig. 1. The Coliseum immersive videoconferencing system.

shared artifacts, the feeling of communication that comes from the passing of subtle through glaring signals that characterize face-to-face meetings. In the Coliseum project, we are working toward establishing a facility to meet these communication needs through a thorough analysis of the computational, performance, and interaction characteristics demanded for universally acceptable remote collaboration and conferencing. Our goal has been to demonstrate, on a single desktop personal computer, a cost-effective shared environment that meets the collaboration needs of its users. The solution must provide for multiple participants—from two to tens—and support them with the required elements of person-to-person interaction. These elements include:

—Acceptable video and audio quality, including resolution, latency, jitter, and synchronization;

—Perceptual cueing such as motion parallax and consistent reciprocal gaze;

—Communicating with words, gestures and expressions over ideas, documents and objects;

—Joining and departing as easy as entering a room.

Traditional telephony and videoconferencing provide some of these elements, including ease of use and audio quality, yet fail on most others. Our Coliseum effort aims to advance the state of videoconferencing by applying recent advances in image-based modeling and computer vision to bring these other elements of face-to-face realism to remote collaboration.

Scene reconstruction, the task of building 3D descriptions using the information contained in multiple views of a scene, is an established challenge in computer vision [Longuet-Higgins 1981]. It has seen remarkable progress over the last few years due to faster computers and improved algorithms (such as Seitz and Dyer [1997], Narayanan et al. [1998], and Pollefeys [1999]). The Coliseum system is built upon the Image-Based Visual Hulls (IBVH) scene rendering technology of MIT [Matusik et al. 2000]. Our Coliseum efforts have shown that the IBVH method can operate at video rates from multiple camera streams hosted by a single personal computer [Baker et al. 2002].

Each Coliseum participant works on a standard PC with LCD monitor and a rig housing five video cameras spaced at roughly 30 degree increments (shown in Figure 1). During a teleconferencing session, Coliseum builds 3D representations of each participant at video rates. The appropriate views of each participant are rendered for all others and placed in their virtual environments, one view of which is

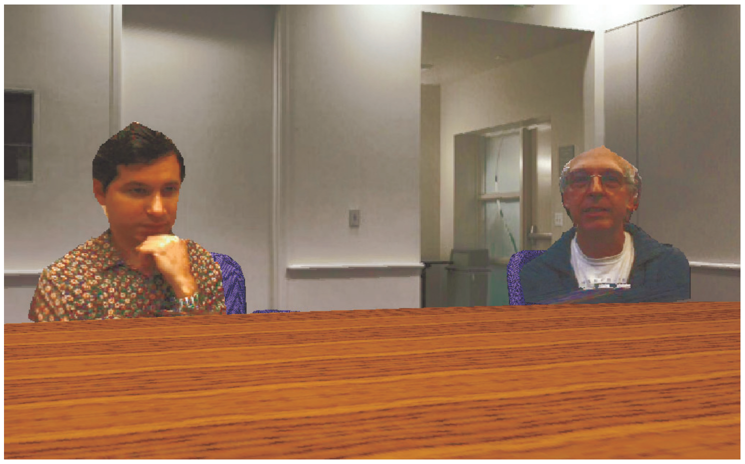

Fig. 2. Two Coliseum users in a shared virtual environment, as seen by a third.

shown in Figure 2. The impression of a shared space results, with participants free to move about and express themselves in natural ways, such as through gesture and gaze.

Handling five video streams and preparing 3D reprojection views for each of numerous coparticipating workstations at video rates has been a formidable task on current computers. Tight control must be exercised on computation, process organization, and inter-machine communication. At project inception, we determined that we needed an effective speedup of about one hundred times over the MIT IBVH processing on a single PC to reach utility. Our purpose in this article is to detail some of the major issues in attaining this performance.

2. RESEARCH CONTEXT

The pursuit of videoconferencing has been long and accomplished [Wilcox 2000]. While available commercially for some time, such systems have, in large part, been met with less than total enthusiasm. Systems rarely support more than two participating sites, and specially equipped rooms are often required. Frame rates and image quality lag expectations, and the resulting experience is of blurry television watching rather than personal interchange. Our intention in Coliseum has been to push the envelope in all dimensions of this technology—display frame rate and resolution, response latency, communication sensitivity, supported modalities, etc.—to establish a platform from which, in partnership with human factors and remote collaboration experts, we may better understand and deliver on the requirements of this domain.

Two efforts similar to ours in their aim for participant realism are Virtue [Schreer et al. 2001] and the National Tele-Immersion Initiative [Lanier 2001]. Both use stereo reconstruction methods for user modeling, and embed their participants in a synthetic environment. As in traditional videoconferencing, these systems are designed to handle two or three participating sites. Neither supports participant mobility. Prince et al. [2002] use Image-Based Visual Hulls for reconstruction and transmission of a dynamic scene to a remote location, although not applying it to multiway communication. Chen [2001] and Gharai et al. [2002] present videoconferencing systems supporting large numbers of users situated individually and reorganized into classroom lecture settings. While both demonstrate some elements we seek—the first examining perceptual issues such as gaze and voice localization and the second including image segmentation to place participants against a virtual environmental backdrop—neither reaches for perceptual realism and nuanced communication in the participant depictions they present.

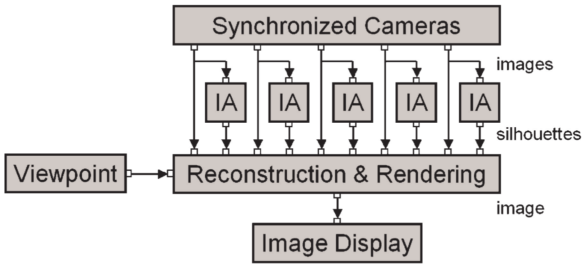

Fig. 3. The simplified Coliseum processing pipeline: image acquisition from synchronized cameras, 2D image analysis, reconstruction, and display of the rendering for a particular viewpoint.

3. THE COLISEUM SYSTEM

Coliseum is designed for desktop use serving an individual conference participant. Five VGA-resolution cameras on a single IEEE 1394 FireWire bus provide video, and a microphone and speaker (or ear bud) provide audio. Coliseum participants are connected over either an Ethernet Local Area Network or an Internet Wide Area Network.

Being a streaming media application, Coliseum has media flowing through a staged dataflow structure as it is processed. This computational pipeline is expressed within the Nizza architectural and programming framework developed at Hewlett-Packard Laboratories [Tanguay et al. 2004]. Figure 3 depicts the simplified processing pipeline for Coliseum, showing the four stages of image acquisition, 2D image analysis, reconstruction and rendering, and display. First, the cameras each simultaneously acquire an image. Second, 2D image analysis (IA) identifies the foreground of the scene and produces silhouette contours (Section 3.1). Third, IBVH constructs a shape representation from the contours and renders a new viewpoint using the acquired video and current visibility constraints (Section 3.2). Finally, the image is rendered and sent for display at the remote site.

Coliseum's viewer renders conference participants within a VRML virtual environment and provides a graphical user interface to the virtual world for use during a session. This allows participants to look around and move through the shared space, with others able to observe those movements.

The Coliseum viewer has features intended to enhance the immersive experience. Consistent display of participants is achieved through their relative placement in the virtual world. An experimental facility for head tracking allows alignment of gaze with the placement of those addressed. In this way, as in the real world, a user can make eye contact with at most one other participant at a time. Head tracking permits the use of motion parallax (Section 3.3), which can further reinforce the immersive experience by making an individual's view responsive to his movements.

Critical to metric analysis of video imagery is acquiring information about the optical and geometric characteristics of the imaging devices. Section 3.4 describes our methods for attaining this through camera calibration. This method is meant to be fast, easy to use, and robust. Sections 3.5 and 3.6 describe the session management and system development aspects of Coliseum.

3.1 Image Processing

The image processing task in Coliseum is to distinguish the pixels of the participant from those of the background and present these to a rendering process that projects them back into the image—deciding which pixels constitute the user and should be displayed. Foreground pixels are distinguished from background pixels through a procedure that begins with establishing a background model, acquired with no one in the scene. Color means and variances computed at each pixel permit a decision on whether a pixel has changed sufficiently to be considered part of the foreground. The foreground is

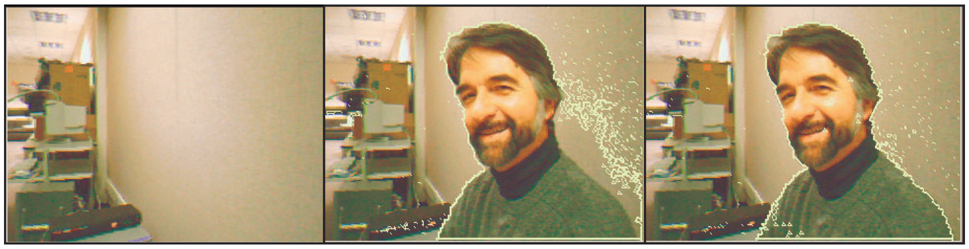
Fig. 4. Left: background image; Center foreground contours; Right: foreground after shadow suppression.

represented as a set of regions, delineated by their bounding elements and characterized by properties such as area, perimeter, and variation from the background they cover.

Ideally, these foreground computations would be occurring at 30 frames per second on all five cameras of our Coliseum system. Sustaining an acceptable frame rate on VGA imagery with this amount of data calls for careful algorithmic and process structuring. In aiming for this, a few principles have guided our low-level image processing:

—Focus on efficiency (i.e., touch a pixel as few times as necessary—once, if possible—and avoid data copying), using performance measuring tools to aim effort;

—Use lazy evaluation [Henderson and Morris 1976] to eliminate unnecessary computation;

—Provide handles for trading quality for speed, so host capability can determine display/interaction characteristics.

Following these guidelines, we have made several design choices to attain high performance:

(1) *Acquire the Raw Bayer Mosaic.* Avoiding explicit color transmission from the cameras enables us to run five full VGA streams simultaneously at high frame rate on a single IEEE 1394 bus. Imagers generally acquire color information with even scan lines of red and green pixels followed by odd scan lines of green and blue pixels (the Bayer mosaic) which are converted to color pixels, typically in YUV422 format. This conversion doubles the bandwidth and halves the number of cameras or the frame rate on a IEEE 1394 bus.

(2) *Employ a Tailored Foreground Contour Extractor.* In one pass over the image, our method determines the major foreground objects, parameterizes them by shape and extent, ranks them by integrated variation from the background and, accommodating to luminance changes—both shadows and gradual light level fluxuations—delivers candidate silhouettes for hull construction. With adjustable sampling of the image, it finds the subject rapidly while retaining access to the high-quality texture of the underlying imagery. Detecting image foreground contours at reduced resolution by increasing the sampling step allows greater image throughput without the loss of image information that accompanies use of a reduced-resolution data source—throughput increases with the square of the sampling. Contour localization doesn't suffer as much as it might with decimated sampling since our method relocalizes using the full image resolution in the vicinity of each detected foreground contour element. Figure 4 demonstrates the illumination adaptation, and Figure 5 shows sampling variations.

(3) *Reduce Foreground Contour Complexity through Piecewise Linear Approximation.* The cost of constructing the visual hull increases with the square of the number of contour elements, so fewer is better. Figure 6 shows this processing.

(4) *Correct Lens Distortion on Foreground Contours Rather than on the Acquired Camera Imagery.* This means we transform tens of vertices rather than 1.5 million pixels on each imaging cycle.

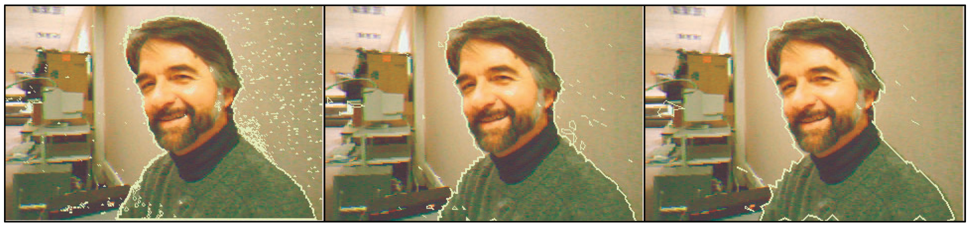

Fig. 5. Various image contour samplings: 1: 4: 8 = 100%, 6%, 1.5% of the image.

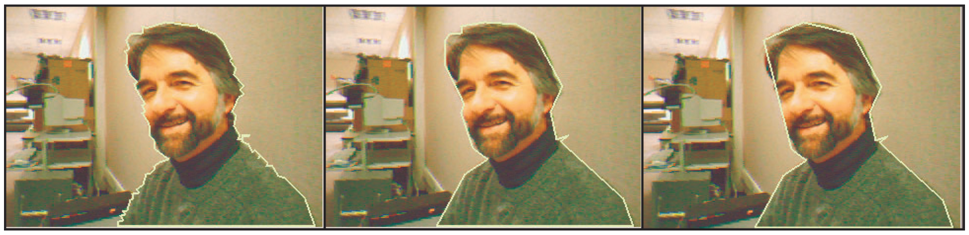

Fig. 6. 525 segment contour, linear approximations (max pixel error, segments) = (4,66): (8,22), (16,14).

(5) *Resample Color Texture for Viewpoint-Specific Rendering only as Needed (on Demand).* With color not explicit (as (1), above), and lens correction postponed (as (4), above), image data for display must be resampled. The on-demand means that only those pixels contributing to other participants' view images will be resampled.

(6) *Parameterize Expensive Operations to Trade Quality for Speed.* For example, rendering a typical 300 by 300 IBVH resultant image for each participant would require 90000 complex ray-space intersections at each time step across all cameras. For efficiency, we parameterize this computation through variable sampling and interpolation of the interior and boundary intersecting hull rays. This and other dialable optimizations can be used to balance processing load and visual quality to meet performance requirements.

3.2 Reconstruction

We use IBVH to render each participant from viewpoints appropriate for each other participant. IBVH back projects the contour silhouettes into three space and computes the intersection of the resulting frusta. The intersection, the visual hull, approximates the geometry of the user. Rendering this geometry with view-dependent texture mapping creates convincing new views. While we could send 3D models of users across the network and render them with the environment model in the Coliseum viewer, less bandwidth is required if we render all the needed viewpoints of a user locally and then send only 2D video and alpha maps. We use MPEG4 to compress the video. Since the majority of displayed pixels comes from the environment model and is produced locally, the video bandwidth requirements are low (about 1.2 Mb/sec). Figure 7 shows the results of foreground contouring, displayed with the visual hull they produce, in the space of the five Coliseum cameras.

While the IBVH algorithm is fast when compared with other reconstruction methods, it has shortcomings. The quality of scene geometry represented depends on the number of acquiring cameras, and surface concavities are not modeled. This geometric inaccuracy can cause artifacts when new views are synthesized. To address this issue, we employed the extension to IBVH of Slabaugh et al. [2002] called Image-Based Photo Hulls (IBPH), which refines the visual hull geometry by matching

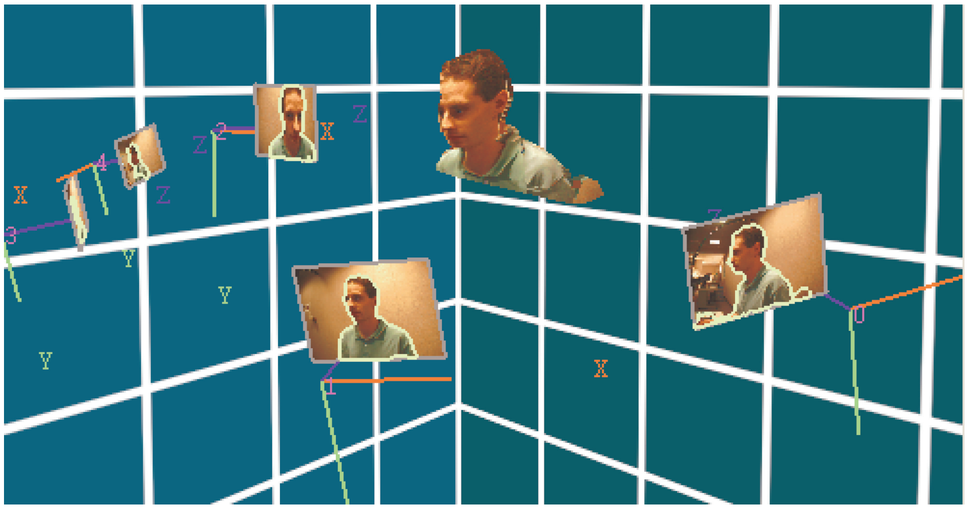

Fig. 7. View of user in Coliseum space: Five cameras surround the rendered user. Each camera shows its coordinate system (in RGB), video frame, and foreground contour.

colors across the images. While resulting in a tighter fit to the true scene geometry and therefore better face renderings, the computational cost was significant, and we did not run the system in real-time sessions with this enhancement. Details may be found in our earlier paper [Baker et al. 2003].

3.3 Motion Parallax

A successful tele-immersion system will make its users feel part of a shared virtual environment. Since our world is three dimensional and presents differing percepts as we move, head movement before the display should induce a corresponding change in view. To achieve this, we developed the capability to track user head position and update the display as appropriate. Unfortunately, the cost of this computation prevented us from employing it in real-time sessions. Baker et al. [2003] provides details of this head-tracking capability.

3.4 Camera Calibration

Our scene reconstruction requires knowledge of the imaging characteristics and pose of each camera. These parameters include:

—*Lens Distortion,* to remove image artifacts produced by each camera's lens (our use of wide-angle lenses exacerbates this).

—*Intrinsics,* that describe how an image is formed at each camera (focal length, aspect ratio, and center of projection).

—*Extrinsics,* relating the pose (3D position and orientation) of each camera to some global frame of reference.

—*Color Transforms,* to enable color-consistent combination of data from multiple cameras in producing a single display image.

All of these parameters must be computed before system use and, in a deployable system such as ours, any of them may need to be recomputed when conditions change.

Figure 8 shows the target we use for parameter estimation—a 10-inch cube with four colored squares on each face (totaling 24 colors plus black and white). A differential operator detects contour edges

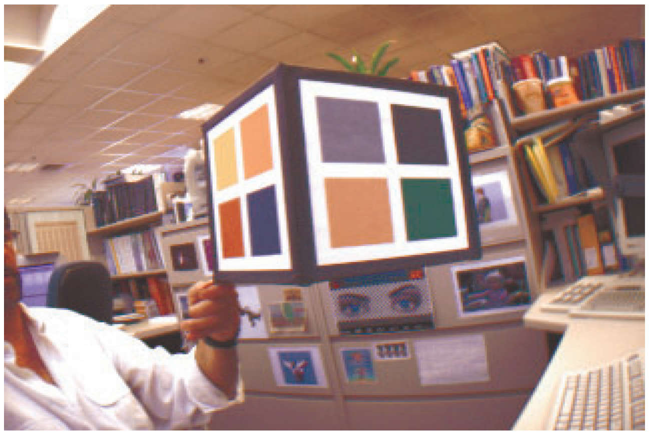

Fig. 8. Calibration target.

in luminance versions of the images, and then a classifier verifies that a detected face contains four squares. The large size and color of the squares make them easier to detect and match, while the multiple faces provides both enough color for good colorimetric modeling, and opportunity for all of the cameras to be acquiring geometric calibration data at the same time.

The face components supply the elements for determining the calibration parameters. Lens distortion correction is computed by determining the radial polynomial that straightens the target faces' black boundaries (Devernay and Faugeras 1995). Intrinsic parameters are derived from the homographies that rectify the colored squares from their projected shapes (Zhang [2000]). Camera extrinsics are estimated in a two-stage process that starts with initial adjacent-pair pose estimates using a nonlinear variant of a stereo solver from Longuet-Higgins [1981] applied to matched square vertices. These poses are chained together and iteratively solved in pairs to minimize error. A bundle adjustment minimizes the total calibration error. The correspondences are implied when observed faces are matched to the target faces, with this matching made more robust by the simultaneous visibility of several faces to a single camera. The color of each square is known—they resemble those of a Macbeth Color Chart—so the colors observed can be used to determine each camera's color transform.

3.5 Session Management

Session management is performed through the Hub subsystem, built using the Microsoft DirectPlay API. A Hub host process for each session runs on a central server and processes connect and disconnect events, notifying session members when other users join or leave. A new user may connect to any existing session, or initiate a new session by starting a new host process. Communications among users during a session are peer to peer. When a new user connects to a session, the local portion of the Hub subsystem determines compatible media types between itself and other users, and notifies the local and remote media transmission and reception modules. These media modules communicate directly using datagram protocols. A multistream UDP protocol allows coordination of different media-type network transmissions. Figure 9 illustrates the dynamic structure of a Coliseum application with session management.

3.6 Software Framework

Streaming media applications are difficult to develop:

—Digital media processings are complex, requiring orchestration among multiple developers.

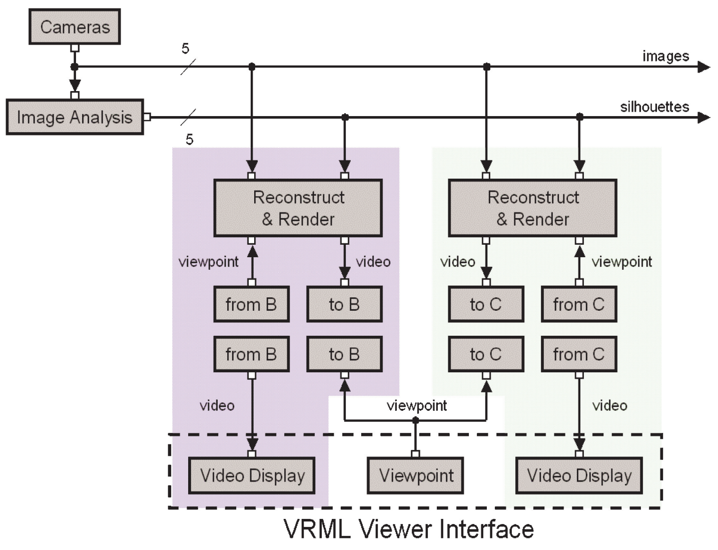

Fig. 9. Coliseum scalable processing pipeline: On a single participant's Coliseum station, the shaded sub-pipelines are added and subtracted from the application as remote participants enter and leave the conferencing session.

—Simultaneous processing of multiple streams uses multithreading and synchronization.

—Real-time user experience requires optimal performance on limited computational resources, requiring flow control and buffer management.

In implementing Coliseum, we have created Nizza [Tanguay et al. 2004], a flexible, multiplatform, software framework that simplifies the development of such streaming media applications. This framework allows an application's processing to be decomposed into a task dependency network and automates exploitation of both task and data parallelism, partitioning operations across as many symmetric multiprocessors as are available.

A dataflow architecture is designed around the travel of data. By observing the data lifecycle throughout the application, one may define a pipeline of distinct processing stages that can be clearly expressed as a directed graph. Our framework addresses all three of the difficulties to developing streaming media applications:

—The application is decomposed into well-defined, connected task modules.

—A scheduler analyzes the task decomposition and then schedules execution.

—The task scheduler achieves real-time performance via automated flow control.

This design simplifies development at all stages, from prototyping to maintenance. A dataflow API hides details of multithreading and synchronization and improves modularity, extensibility, and reusability. We have implemented the framework in C++ on Windows, Windows Mobile, and Linux platforms. Using this framework, we have developed a library of reusable components for the Windows platform (e.g., audio recording and playback, video playback, network connectivity). The streaming media aspects of Coliseum were built using Nizza and the reusable components.

The framework has three main abstractions: *Media* (data unit), *Task* (computation unit), and *Graph* (application unit):

—*Media* objects represent time-stamped samples of a digital signal, such as audio or video. A memory manager reuses memory whenever possible. The new Media abstraction inherits an automatic serialization mechanism for writing into pipes, such as a file or network buffer.

—*Task* objects represent an operation on Media. The abstraction is a black box of processing with input and output pins, each specified with supported types of Media. The Task completely encapsulates the processing details so that the user knows only the functional mapping of input pins to output pins.

—*Graph* objects are implicitly defined by the connectivity of multiple Tasks. Several commands can be issued to a graph including those to start and stop the flow of Media among its Tasks. Each Graph has its own scheduler, which orders the parallel execution of Tasks.

This infrastructure provides three distinct benefits:

—It supports an incremental development strategy for building complex applications—a Graph with a multistage pipeline structure can undergo testing of functional subsets hooked to real or synthesized data sources and sinks.

—The framework allows for dynamic graph construction. When stopped, an application can add and remove Tasks, then start again with a new graph while keeping the unchanged portions intact. We use this technique in the large, dynamic graph of Coliseum, adding and removing portions of the graph as participants enter and depart sessions.

—The graph structure supports instrumentation. Individual nodes in a Nizza graph can be instructed to append timing and related information to data packets, facilitating data-progression assessment. Keeping a functioning Graph intact, a new Task can connect to any of its output pins to monitor activity. This ability to listen and report is useful for gathering statistics in a performance monitor or to effect feedback control in modifying system parameters on the fly.

A designer of a real-time rich media application can choose from several componentized, dataflow-style architectures for media processing. Microsoft DirectShow, for example, may be a good approach for simple applications that use only prepackaged modules (e.g., compression, video capture). It also has a graphical interface for constructing and configuring a dataflow application. However, constructing new modules is difficult, performance metrics are not automatic, it is complex to learn and use, is dependent on other layers beyond our control (such as COM), and is not supported on the Linux platform. In addition, its use of processes rather than threads makes it poor for debugging multi-stream video applications, and its lack of a media scheduler means it discards untimely work rather than suppressing it, wasting capacity in resource-critical applications. The Network-Integrated Multimedia Middleware (NMM) project [Lohse et al. 2003] is an open-source C++ framework designed for distributed computing. NMM makes the network transparent from the application graph, but does not have performance features like Nizza, and is not available for Windows platforms. The Java Media Framework [Gordon and Talley 2003] is multiplatform and has integrated networking via Remote Transport Protocol, but its performance on heavy media (e.g., video) is not competitive with Nizza's. Signal processing software environments such as Ptolemy [Buck et al. 2001], while operating efficiently on 1-D signals such as audio, are not appropriate for our "heavy" media, and often don't support cyclic application graphs, which we have found useful for incorporating user feedback into a processing pipeline. VisiQuest (formerly known as Khoros and now available from AccuSoft[1]) is a commercial visual programming environment for image processing and visualization. While an impressive visual environment, it lacks performance enhancements and metrics.

[1]http://www.accusoft.com

4. PERFORMANCE

Coliseum is a multi-way, immersive remote collaboration system that runs on modest through advanced commodity hardware. We have run sessions with up to ten users (all the Coliseum systems we have available), and between North America and Europe. Success depends on our ability to provide video-conference functions with sufficient responsiveness, audio and video quality, and perceptual realism to take and hold an audience. We evaluate Coliseum's performance in terms of its computational and networking characteristics. Since we aim to support large numbers of participants in simultaneous collaboration, we are reviewing the implications of these measures on the system's scalability.

All the measurements presented are collected on the following equipment and conditions. The application runs on dual Xeon-based PCs with speeds of 2.0, 2.4, 2.8 or 3.06 GHz, each with 1, 2, or 4 GB of memory, and running Windows 2000 or XP. Machines shared a single 1000SX gigabit Ethernet connected by Cisco Catalyst 4003 10/100/1000 switches and were, typically, two to three hops apart for local area tests. The local network was in use at the time for other HP Lab activities. We also conducted wide area network measurements on Internet.[2] Imagery was acquired through Point Grey Dragonfly VGA IEEE 1394 (FireWire) cameras operating at 15 Hz. The PC used for data collection had dual 2.4 GHz Xeon CPUs and 1 GB of memory.

4.1 What to Measure and How to Measure

When approaching a large scale system like Coliseum it is important to frame the questions of what will be measured, why, and how. There are a number of techniques that can be used and the selection of their measures should be based on the intended use.

Our first approach was the common one for measuring performance—using profiling tools such as *prof*, *gprof*, *Vtune*™,[3] etc. which collect run-time, fine grained performance data. These tools collect resource usage data on a function and module basis through sampling, resource counter monitoring, or call graph instrumentation. The sampling method periodically probes the current instruction addresses and provides an accounting of resource use by the function sample bin. Some processors support continuous counter monitoring which tracks hardware and software resources over a specified time interval. However, neither tracks the call graph. The relationships among functions are not preserved so, while a routine's total CPU computation is identified, the call sequence is not. Call graph instrumentation is available but this is accomplished at a cost—instrumentation of the code forcing each function to execute an accounting preamble. This changes the execution profile and, for an application such as Coliseum that is already running near system capacity, it is not practical as full frame rates cannot be maintained.

We experimented with these traditional measurement methods but found them to be unsatisfactory. The profiling tools do not show how much time was spent in synchronization wait states and in the run state for a particular frame set. They only give this information for a call graph as a whole. For example, we identified that the silhouette functions consumed the most CPU time, but this did not provide a picture of the data flow latency.

We determined that we needed a top-down flow-based view of application performance. This called for two further types of measures (see Figure 10):

—Application measures that can be used to tune performance and understand each component's contribution to overall latency, frame rate and resource consumption.

—End-to-end measures that can confirm the application instrumentation and capture the user experience that is beyond the application control points.

[2]http://www.internet2.edu/
[3]http://www.intel.com/software/products/vtune/

The application measures are accomplished through instrumentation of our system to collect timing data at flow control points. Nizza's modular architecture allowed these points to be easily accessed (subsequently, provision was made for this information to be provided automatically at computational nodes in the data chain). Each collection of "frames" is marked with a unique timestamp. As these frames move through the system, timestamps are appended when the frame passes a control point. For example, once contours are extracted for each 5-camera set, their data are available on the five input "data-pins" of the next processing stage. Timing information can be appended here. Similarly, "camera data available" is a synchronization point whose times are tracked and propagated with the data. Clearly, these are the timing values we want, since their integration tells us everything about the system's behavior as its data moves through. The observations include those of latency, and processing and synchronization wait times.

This application instrumentation, while powerful, does not give a complete picture of the user-experienced end-to-end latency. What about the components outside of the application? The latency in the camera hardware and its drivers, and the latency from image composition until it appears on the display are characteristics not accessible from inside the application. What are the effects of these components on performance?

4.2 Use of Performance Data

The previous section described three ways that we evaluated performance: profiling, application instrumentation, and end-to-end measurements; in this section, we describe how these measurements can be used to guide system understanding and improvement. One essential component of performance analysis is repeatability. We repeated experiments to confirm that measurements were consistent. If measurements change from run to run, then the system metrics cannot provide conclusive evidence. Once the system is stable, performance data can be used as a system diagnostic to:

—*Evaluate Different Sections of the Pipeline*. Components can be replaced in our modular architecture and we can evaluate them before and after to assess the "cost" of each component. Cost can be assessed by network traffic, CPU load, maximum frame rate, latency, memory usage, etc. For example, we removed MPEG (encode and decode), and compared pre- and post-numbers to let us calculate the cost of the MPEG modules in the pipeline.

—*Evaluate a "Fix" or "Performance Improvement" Added to the System*. Improving visual computing algorithms can be complex, with both subjective and objective components. An absolute performance timing measure assures that this part of the processing has been improved.

—*Identify Large Resource Users as Targets for Improvement*. Originally, we believed that the network was a bottleneck in the system and were prepared to expend effort to reduce data transmission. After measurement, we realized that this would not have resulted in significant latency reduction.

—As drivers, hardware, or other non-application components change, we use the measures to quantify the effects. Through profiling, we determined that Coliseum was compute bound. To improve computational efficiency, we systematically evaluated compilers and compile options under identical test conditions and tracked the frame rate metric. This brought a 30% increase in frame rate.

4.3 End-to-End Measurements

Latency has a major impact on the usability of a communication system. There are numerous contributors to overall system latency, and we have measured various stages to assemble a picture of the delays

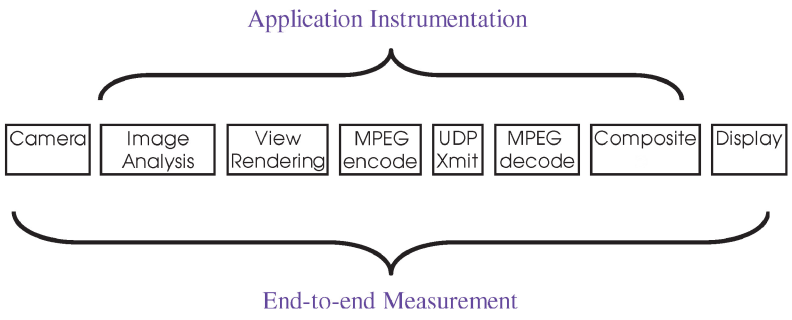

Fig. 10. Application and end-to-end performance measurement.

between actions at one site and observations at the other. Coliseum's latency is composed of the time for:

—*Camera Latency*. An event to be imaged and delivered by the camera driver.

—*Processing Latency*. Receiving the image data from the camera system to process, create the visual hull, render a requested view, and encode this in MPEG4.

—*Network Latency*. Processing by UDP protocol stack, and network transmission.

—*Display Latency*. Dataset reassembly, MPEG decoding, image composition, and return from providing the data to the display drivers.

Measuring camera system latency requires either fancy synchronization gear or an externally observing capture device. Choosing the latter, we used a field-interlaced digital video camera to image both an event and the after-processing display of that event simultaneously. If an event is instantaneous, it is visible at the origination and the display. If it is not instantaneous, then we count the number of frames before it is visible in the output display. We tried several events but needed one that would be fairly atomic—coming "on" in one frame. An incandescent light proved to be inappropriate since it took several frames to reach full illumination. Our event was the illumination from a laser pointer, directed at a Coliseum camera. The laser and the camera's display were simultaneously visible to the observing video camera (see Figure 11). Manual frame-by-frame analysis of the acquired video provided the numbers we sought (Figure 12). We captured several such events, and our tables in Figures 13 and 14 indicate average values.

We measured end-to-end latency in four situations:

—Simple camera driver demonstration program (*TimeSliceDemo)*. [camera and driver latencies]

—Stand-alone version of Coliseum with no network or VRML activity (*Carver*). [camera, drivers, image processing, and simple display latency]

—Coliseum test of two users with live networking, with and without MPEG encoding (*Coliseum*). [complete system test and measure of MPEG impact]

—Coliseum test where the subject is both the sender and receiver of the view (*Glview Loopback*—single person loop-back conference). [end-to-end latency minus negligible network transmission]

Both the *Coliseum* and *Carver* measurements reflect round-trip frame counts, so the one-way latency is half the observed figure. The third test was done to see the effect of MPEG processing on latency.

Figure 13 gives the average video frame count (at 33 ms per frame) for each test. The observing video camera captured 30 frames per second, permitting us to calculate latencies and standard deviations. Some of the time intervals we measured were about a dozen frames, while others were low single digits. Since images are acquired with units of 33ms delay, estimation precision is better for the former than

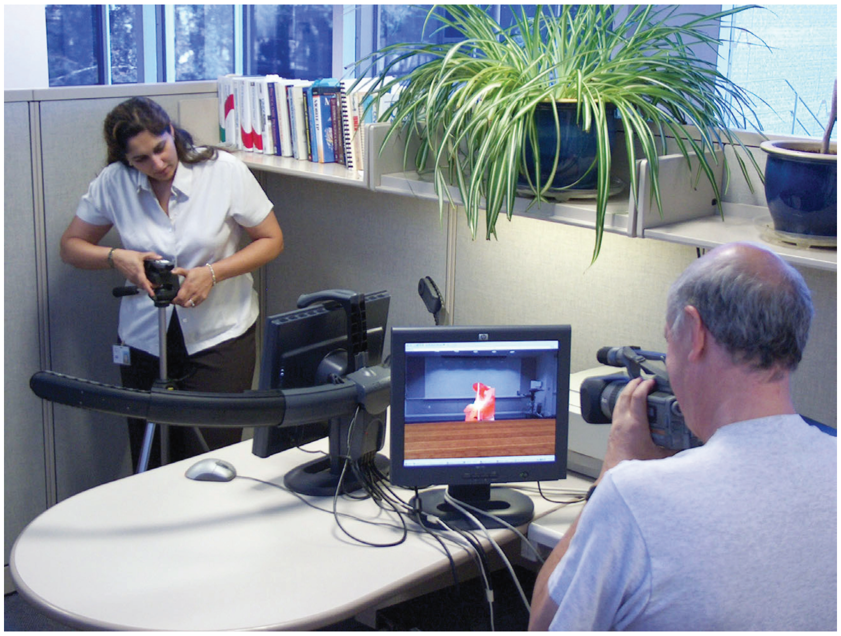

Fig. 11. Capture of laser light onset and propagation to display.

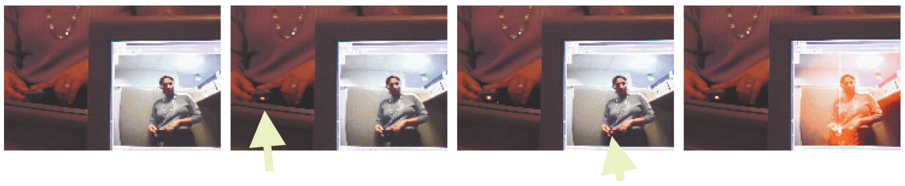

Fig. 12. Latency measurement: 1) no light, 2) light on, 3) propagates to display, 4) saturates.

System	*Frames*	*Mean Latency (ms)*	*stdev*
TimeSliceDemo	4.25	142	16.67
Carver, MPEG	11.63	194	28.05
Coliseum, no MPEG	14.30	238	32.12
Coliseum, MPEG	16.30	271	27.28

Fig. 13. Absolute user-perceived latency tests.

the latter, but our interest has been first in ballpark numbers. Refinement could be obtained, where needed, by measuring on the fields rather than the frames of the interlaced video, and by performing linear interpolation on the observed illuminant brightness, but this we did not do.

TimeSliceDemo gives us an estimate on the latency that lies beyond our control—it is the time it takes the camera to acquire the frame and store it in the computer. Of course, this includes time for the camera to integrate the frame (on average, one half of a frame, or 16 ms, for the event), to charge

Glview Loopback, subject	*Latency*	*Difference*
User	233	38
Bottle	229	42

Fig. 14. Instrument measure of latency (ms).

transfer, digitize, and ship the frame to the PC (one frame), to buffer and DMA the data to memory, and the time for the PC to display the frame after it has arrived (observed as perhaps one frame cycle of the observing camera). The latter period should be discounted. Figure 14 indicates measures of the instrumented latency of this same system version and, comparing the *Coliseum* with MPEG user-perceived tests to the instrument latency figures, we find differences of 38 and 42 milliseconds. This difference represents the latency that should be added to instrumented error to derive an estimate of end-user experienced latency.

We observe that the absolute user-experienced latency in *Coliseum* ranges from 244 to 298 milliseconds. Enabling MPEG encoding and decoding increases latency by 33 milliseconds. MPEG encoding reduces the amount of data each participant sends, but does this at the cost of additional processing. This indicates a tradeoff we must consider in our control considerations in system balancing.

There is a 77-millisecond difference between *Coliseum* and our standalone *Carver* application. This is attributable to the VRML viewer and network activity. We will see that network activity load is minimal and that the addition is due to the VRML control, which currently (and inappropriately) uses a busy-wait loop for its user interface.

4.4 Application Instrumentation

We instrumented the code both for the *Carver* application (non-networked) and for *Coliseum* itself (see Figure 15). The instrumentation collects continuous timing data. These data are sent with each frame set to the corresponding host. Timing data were collected using a lightweight system call (Windows XP's `QueryPerformanceCounter`) that sampled the processor clock. Each resulting data set contained time stamps indicating when the camera frame set was first available (t_0). After the image analysis is complete, another timestamp is taken (t_1). The receiving host records the time it received the frame (t'_2), and the timestamp after decoding, compositing, and displaying the resultant image (t'_3). $\Delta t'$ is the time the system waits to piggyback timing information onto the outgoing frameset of the receiving host, indicated by "Wait for Camera Data." This piggybacking avoids the introduction of additional network traffic and brings only nominal overhead. The bottom of the figure shows the return part of the journey.

Processor clocks are not synchronized, so we cannot directly compare timestamp values across machines. Machine clock rates are stable, however, and elapsed time values can be used across host boundaries. We make extensive use of this fact to derive the timing contributions of each component of the timeline. The round time (*RT*)—camera to display then next camera data back to originating display—is calculated as the time from the camera data to when a corresponding frame was received from the other host minus the time waiting to piggyback the data:

$$RT = (t_3 - t_0) - \Delta t$$

The total network time (*NT*)—protocol stack processing time and transmission—can be determined from the roundtrip less the time in each host for processing, MPEG, composition, and display:

$$NT = RT - (t_1 - t_0) - (t_3 - t_2) - (t'_3 - t'_2) - (t'_1 - t'_0)$$

One-way network time is half *NT* since *NT* was a round trip measure.

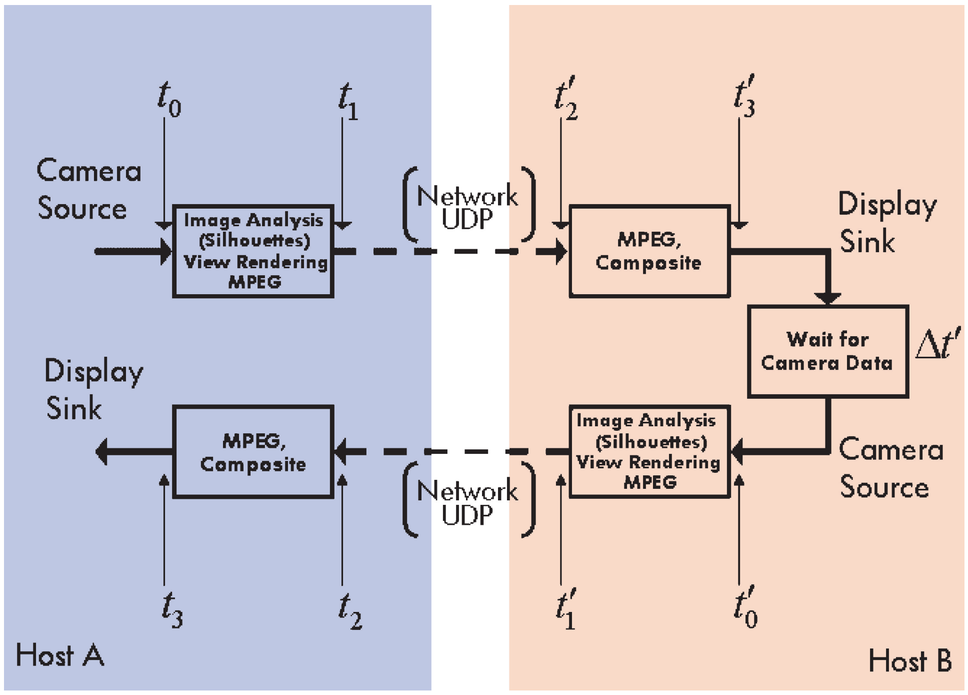

Fig. 15. Performance measurement: data routing and measurement taps.

System	*Subject*	*1-way Latency*	*Image Generation*	*Network*	*Display*	*Frames /sec*	*CPU Utilization*
Coliseum	User	112	71	3	35	15.00	80%
	Prerecorded User	151	121	4	40	15.36	83%
	Bottle	130	95	4	45	15.00	80%
	Prerecorded Bottle	134	102	3	39	17.68	78%
Carver	User	78	68	NA	10	15.00	75%
	Prerecorded User	93	77	NA	14	20.99	100%
	Bottle	65	57	NA	7	15.00	61%
	Prerecorded Bottle	95	71	NA	21	22.42	100%

Fig. 16. Analysis of latency (ms).

Figure 16 presents data for the *Coliseum* and *Carver* tests. We tested each application with four different subjects—a user, a prerecorded dataset of a user, a phantom 5-gallon water bottle, and a prerecorded dataset of the bottle (see Figure 17). The bottle subject is placed at approximately head height but is stationary and several times larger than a user's head. The prerecorded data allow us to exercise the systems without the camera frame rate limitation, although memory and disk accesses can similarly affect performance. For each subject, we give the one-way latency, time to generate the image, network delay, and display time. The table also shows the achieved average frame rate and CPU

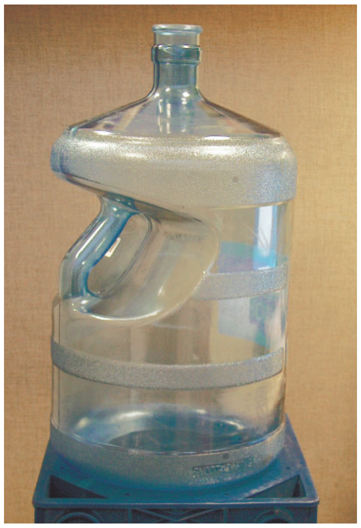

Fig. 17. Bottle used as phantom for performance measurements.

utilization. Note that these data were compiled from a later release of the system and therefore should not be directly compared to the data for the user-perceived latency results.

4.5 Networking Requirements

Though video usually requires considerable network bandwidth, *Coliseum's* bandwidth needs are quite modest. This is because the virtual environment usually occupies the overwhelming majority of the area on a *Coliseum* screen and, being maintained locally, is not part of the video stream. MPEG4 further reduces the bandwidth requirement. At 15 frames/sec., we measured a typical *Coliseum* video stream to be 616 Kb/sec. Although *Coliseum* can use TCP or UDP as a transport, all tests were conducted with UDP. Using UDP, there could be hundreds of *Coliseum* video streams before overloading a gigabit network.

We measured latency in our local area network where the two participant hosts were two network-switched hops apart. The average latency was 3 ms, so the network contributes about 2% to overall latency. To characterize the wide area performance of the system, we measured latency on Internet 2 from HP Labs Palo Alto to a site at the University of Colorado in Boulder. Our tests showed an average network latency of 25 ms.

4.6 Other Measures of Performance

While much of the above discussion centers around latency, we include other measures of performance as well. We focused a lot of work on latency because of the challenges in its measurement. The *Carver* and *Coliseum* applications provided continuous monitoring of frame rates and network traffic. In the profiling and application instrumentation tests, we measure CPU and memory usage. We did not vary image quality or size as a result of system load so we did not measure these metrics. Our system is designed to provide the maximum frame rate possible. In the tests of large numbers of users, the frame rate degrades as CPU load rises in accommodating the increase in simultaneous renderings.

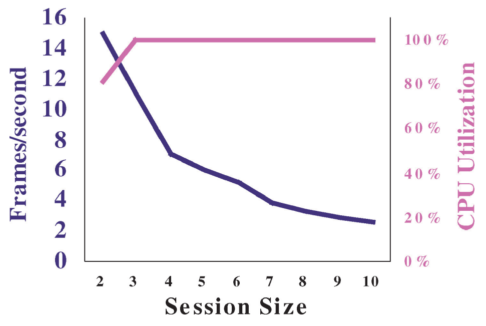

Fig. 18. Change in fps and cpu utilization for increasing Coliseum session sizes.

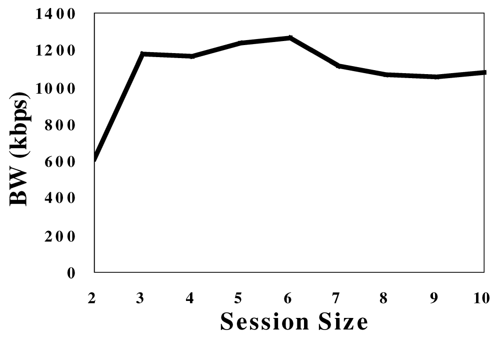

Fig. 19. Aggregate bandwidth for different size Coliseum sessions.

4.7 Scalability

Since a major goal of Coliseum is to support video conferencing among large groups of people, scalability is an important system characteristic. We measured the system's scalability by conducting sessions of increasing population, from 2 to 10 participants (10 being the number of Coliseum systems on site). Figure 18 shows that, as session size increased, system performance (frames/sec) dipped due to the increased workload of creating images and MPEG streams for the expanding number of view renderings required. While frame rate degraded, the total aggregate bandwidth sent by one user remained fairly constant, which means that the system adapts to more users in a work-conserving manner. Figure 19 shows that bandwidth climbed from 616 Kb/sec to 1178 Kb/sec as the CPU utilization saturated and then leveled off after the 6-user session. All in all, the bandwidth varied 16% over the course of these session sizes. At least this much variation is expected over any collection of runs, as the bandwidth is sensitive to user movement and image size.

4.8 Performance Summary

In two-way sessions, we have achieved a rate of 15 frames per second, the maximum the FireWire bus can support (five cameras at higher rates on FireWire is only possible with image size reduction). Our throughput to date indicates that we have achieved about a thirty-five-times speedup from algorithmic and architectural innovations and a three-times speedup through processor evolution, meeting our beginning requirement of a hundred-fold speedup. From tests on larger numbers of users, we find that the computational complexity of the system dominates performance. There are a number of parameters that can be used to reduce computation at the expense of visual quality, and adjustment of these would allow support of more users while maintaining interactive frame rates. The current system reduces frame rate but maintains image quality. As the numbers of users grows, performance stabilizes, with bandwidth served remaining relatively constant.

Our extensive measurements of *Coliseum* provides a clear breakdown of latency

—Camera latency: 20%

—Processing Latency*:* 50%

—Network Latency: 2% to 10% (local or wide area)

—Display Latency: 25%

These measures can direct strategies for controlling delay and improving system performance for large numbers of users. Since Coliseum is a highly compute-intensive application, we have the potential to control the end node behavior and therefore overall system performance. With facility for graph-level performance monitoring (Section 3.6) and control parameters for adjusting the quality—and therefore speed—of image processing and display computations (Section 3.1), we have the tools we need for balancing throughput with user needs. While statically configured for the evaluations, we report here (with an image sampling step of 2 pixels, a four-pixel maximum deviation for linear approximations, and a hull ray sampling step of 4 pixels), these parameters may be adjusted over time and across cameras to meet bandwidth and throughput demands.

5. CONCLUSIONS

Coliseum creates an immersive experience by building dynamic, 3D user models, embedding them in a shared, virtual space through which users are free to move, and generating unique views of the space for each user. The views convey reciprocal gaze and can be made responsive to head movements. Interactive performance is achieved through streamlined image processing and a software framework that is tuned for streaming media applications. This represents the first implementation of an immersive collaboration system supporting an arbitrary number of users and aimed at three-dimensional realism. While the possibility of such systems has often been discussed, actual implementations have been incomplete, operating only one-way, using cartoon avatars, or requiring substantial special purpose hardware. Employing commodity PCs and simple video cameras, we have run fully symmetric Coliseum sessions with as many as ten users.

Instrumenting the system with timing recorders enabled precise post hoc as well as on-the-fly performance measurement. This tooling permits review of system performance, as described, for computational assessment and restructuring and, as proposed here, dynamic system adjustment to attain required levels of service.

6. FUTURE WORK

Coliseum continues to evolve to meet a new set of goals. As it stands, it presents an effective mechanism for the proverbial “talking heads” videoconferencing—with the twist that the participants are 3D renderings of themselves and the environment is synthetic. Realistically, there are numerous developments that remain before this could be considered a viable alternative to travel for collaborative remote conferencing. Obvious improvements include increasing the frame rate, reducing latency, raising the quality at which people are displayed, and reconfiguring computation to enable more advanced features (such as head tracking). We are addressing these advancements in several ways:

—*Frame Rate.* Development of a multicamera VGA capture system that streams synchronized video from two dozen or more cameras at 30Hz [Baker and Tanguay 2004].

—*Latency.* A more effective camera interface (for the above) that simplifies frame-set organization and reduces camera latency to its minimum.

—*Display Quality.* Higher resolution capture through multicamera integration [Baker and Tanguay 2004]. In addition, we may raise the camera imaging resolution from VGA to XGA in the near future.

—*Computation.* Migration of compute-intensive image-display operations to PC graphics processors where possible. This will free up resources to let us do more of the operations that will increase the quality of the user's experience (e.g., obtaining better geometry, tracking heads, or correcting gaze direction).

Objects, documents, and elements of the personal environment (such as white boards) also play an important role in collaborative interactions. A recent extensive survey of videoconferencing [Hirsh et al. 2004] indicates the importance of providing a high quality experience, one that will lift acceptance above that of audio conferencing for remote-participation meetings. The survey emphasizes the needs: ease of use, system reliability, high video quality, and provision of a general environmental context including the ability to share work-related objects. Although these have all been goals of our Coliseum work, it is the latter point, most particularly, that has been influencing our current direction. While always aiming to integrate imaged artifacts into the shared virtual space of our heads-and-shoulders depiction (i.e., documents, prototype boards, etc.), we are now moving toward providing full body and workspace coverage as well. At the same time, we choose to support inclusion of multiple participants at a site. These changes have major implications for computational performance and the imaging and display sides of our conferencing: We need larger images—higher resolution in capture and display—and will move to include projection to accommodate the increased scale of presentation needed. This direction will continue to put heavy demand on computational facilities, and increases the demand for thorough understanding of performance issues and careful allocation of overworked resources.

Advancing the technology base of our collaborative videoconferencing effort must proceed with a foundation including both innovative design of algorithms and devices, and transparent mechanisms for instrumenting, monitoring, and adapting the system, while maintaining constant attention on the needs and preferences of the user. We have found that augmenting a networked interactive application like Coliseum with monitoring instrumentation is critical to understanding its behavior and dynamic structure.

ACKNOWLEDGMENTS

Mike Harville, John MacCormack, the late David Marimont, Greg Slabaugh, Kei Yuasa, and Mat Hans made important contributions to this project. Sandra Hirsh and Abigail Sellen directed our user studies enquiries. Wojciech Matusik, Chris Buehler, and Leonard McMillan provided guidance on the original IBVH system from MIT.

REFERENCES

Baker, H. H., Tanguay, D., Sobel, I., Gelb, D., Goss, M. E., Culbertson, W. B., and Malzbender, T. 2002. The Coliseum Immersive Teleconferencing System. In *Proceedings of the International Workshop on Immersive Telepresence* (Juan Les Pins, France, Dec.). ACM, New York.

Baker, H. H., Bhatti, N., Tanguay, D., Sobel, I., Gelb, D., Goss, M. E., MacCormick, J., Yuasa, K., Culbertson, W. B., and Malzbender, T. 2003. Computation and performance issues in Coliseum, an immersive teleconferencing system. In *Proceedings of the 11th ACM International Conference on Multimedia* (Berkeley, Calif., Nov.). ACM, New York. 470–479.

Baker, H. H. and Tanguay, D. 2004. Graphics-accelerated panoramic mosaicking from a video camera array. In *Proceedings of Vision, Modeling, and Visualization Workshop* (Stanford, Calif., Nov.). IOS Press. 133–140.

Buck, J., Ha, S., Lee, E. A., and Messerschmitt, D. G. 2001. Ptolemy: A framework for simulating and prototyping heterogeneous systems. In *The Morgan-Kaufmann Systems on Silicon Series, Readings in Hardware/Software Co-Design*, Morgan-Kaufmann, San Francisco, Calif., 527–543.

CHEN, M. 2001. Design of a virtual auditorium. In *Proceedings of the ACM International Conference on Multimedia* (Ottawa, Ont., Canada, Sept.). ACM, New York, 19–28.

DEVERNAY, F. AND FAUGERAS, O. 1995. Automatic calibration and removal of distortion from scenes of structured environments. SPIE, 2567 (San Diego, Calif., July). 62–72.

GHARAI, L., PERKINS, C., RILEY, R., AND MANKIN, A. 2002. Large scale video conferencing: A digital amphitheater. In *Proceedings of the 8th International Conference on Distributed Multimedia Systems* (San Francisco, Calif., Sept.).

GORDON, R. AND TALLEY, S. 1999. *Essential JMF: Java Media Framework.* Prentice-Hall, Englewood Cliffs, N.J.

HENDERSON, P. AND MORRIS, J. H. JR. 1976. A lazy evaluator. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN Symposium on Principles on Programming Languages*. ACM, New York, 95–103.

HIRSH, S., SELLEN, A., AND BROKOPP, N. 2004. Why HP people do and don't use videoconferencing systems. Hewlett-Packard Laboratories, External Tech. Rep. HPL-2004-140R1.

LANIER, J. 2001. Virtually There. *Scientific American*, Apr., 66–75.

LOHSE, M., REPPLINGER, M., AND SLUSALLEK, P. 2003. An open middleware architecture for network-integrated multimedia. Lecture Notes in Computer Science, vol. 2515/2002, Springer-Verlag, Heidelberg, Germany, 327–338.

LONGUET-HIGGINS, H. C. 1981. A computer algorithm for reconstructing a scene from two projections. *Nature 293*, 133–135.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S., AND MCMILLAN, L. 2000. Image-based visual hulls. In *Proceedings of the SIGGRAPH 2000*. ACM, New York, 369–374.

NARAYANAN, R., RANDER, P., AND KANADE, T. 1998. Constructing virtual worlds using dense stereo. In *Proceedings of the International Conference on Computer Vision*. (Bombay, India). 3–10.

PESCE, M. 2002. *Programming Microsoft DirectShow for Digital Video, Television, and DVD.* Microsoft Press.

POLLEFEYS, M. 1999. Self-calibration and metric 3D reconstruction from uncalibrated image sequences. Ph.D. dissertation. ESAT-PSI, K.U. Leuven.

PRINCE, S., CHEOK, A., FARBIZ, F., WILLIAMSON, T., JOHNSON, N., BILLINGHURST M., AND KATO, H. 2002. Real-time 3D interaction for augmented and virtual reality. In *Proceedings of SIGGRAPH 2002*. ACM, New York, 238.

SCHREER, O., BRANDENBURG, N., ASKAR, S., AND TRUCCO, E. 2001. A virtual 3D video-conferencing system providing semi-immersive telepresence: A real-time solution in hardware and software. In *Proceedings of the International Conference on eWork and eBusiness*. (Venice., Italy). 184–190.

SEITZ, S. AND DYER, C. 1997. Photorealistic scene reconstruction by voxel coloring. In *Proceedings of Computer Vision and Pattern Recognition Conference*. (Puerto Rico). 1067–1073.

SLABAUGH, G., SCHAFER, R., AND HANS, M. 2002. Image-based photo hulls. In *Proceedings of the 1st International Symposium on 3D Processing, Visualization, and Transmission*. (Padua, Italy). 704–708.

TANGUAY, D., GELB, D., AND BAKER, H. H. 2004. Nizza: A framework for developing real-time streaming multimedia applications. Hewlett-Packard Laboratories, Technical Report, HPL-2004-132.

WILCOX, J. 2000. *Videoconferencing, The Whole Picture*, Telecom Books, N.Y., ISBN 1-57820-054-7.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *IEEE Trans. Patt. Anal. Mach. Intell. 22*, 11, 1330–1334.

Received January 2005; accepted January 2005