Computation and Performance Issues In Coliseum, An Immersive Videoconferencing System

H. Harlyn Baker, Nina Bhatti, Donald Tanguay, Irwin Sobel, Dan Gelb,
Michael E. Goss, John MacCormick, Kei Yuasa, W. Bruce Culbertson,
Thomas Malzbender
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-129
July 15th , 2003*

telepresence, videoconferencing, view synthesis

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 monitor 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. This paper summarizes the technology, and reports on issues related to its performance.

* Internal Accession Date Only
Approved for External Publication

Computation and Performance Issues In Coliseum, An Immersive Videoconferencing System

H. Harlyn Baker, Nina Bhatti, Donald Tanguay, Irwin Sobel, Dan Gelb, Michael E. Goss, John MacCormick, Kei Yuasa, W. Bruce Culbertson and Thomas Malzbender

Hewlett-Packard Laboratories
Palo Alto, CA

Abstract

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 monitor 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. This paper summarizes the technology, and reports on issues related to its performance.

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

Keywords Telepresence, Videoconferencing, View Synthesis.

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 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 or more—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 the walking into a room

Figure 1. The Coliseum immersive videoconferencing system

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 81]. It has seen remarkable progress over the last few years due to improved algorithms [Seitz 97, Narayanan 98, Pollefeys 99] and faster computers. The Coliseum system is based on the Image-Based Visual Hulls (IBVH) image-based rendering scene reconstruction technology of MIT [Matusik 00]. Our recent Coliseum efforts have shown that the IBVH method can operate at video rates from multiple camera streams hosted by a single personal computer [Baker 02].

Each Coliseum participant works on a standard PC with LCD monitor and a rig housing five video cameras spaced at roughly 30 degree increments, as 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 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 voice, gesture, and gaze.

Handling five video streams and preparing 3D reprojection views for each of numerous coparticipating workstations at video rates is a formidable task. Tight control must be exercised on computation, process organization, and inter-desktop communication. At project inception, we determined 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 paper is to detail some of the major issues in attaining this performance.

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

2 Related Work

The pursuit of videoconferencing has been long and accomplished [Wilcox 00]. While available commercially for some time, such systems have been met, in large part, 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 01] and the National Tele-Immersion Initiative [Lanier 01]. Both use stereo reconstruction methods for user modeling, and embed their participants in a synthetic environment. As in traditional videoconferencing, these systems are designed for a small fixed number of participating sites. Neither supports participant mobility. [Prince 02] uses Image-Based Visual Hulls for reconstruction and transmission of a dynamic scene to a remote location, although does not apply it to multi-way communication. [Gharai 02] and [Chen 01] 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 including image segmentation to place participants against a virtual environmental backdrop and the second examining perceptual issues such as gaze and voice localization – neither reaches for perceptual realism and nuanced communication in the participant depictions they present.

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 an ethernet or internet.

Coliseum is a streaming media application and, as is typical for such applications, has media flowing through a staged dataflow structure as it is processed. 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.

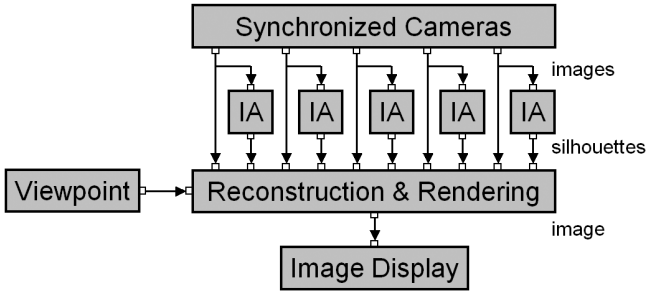

Figure 3. The simplified Coliseum processing pipeline: image acquisition from synchronized cameras, 2D image analysis, reconstruction, and display of the rendering for a particular view point.

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. Head tracking will allow alignment of gaze with the placement of those addressed. In this way, as in the real world, a user will make eye contact with at most one other participant at a time. Use of motion parallax (section 3.3) further reinforces 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 for other participants. 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 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 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. This enables us to run five full VGA cameras simultaneously at high frame rate on a single IEEE 1394 bus. Imagers generally acquire color information with even scan lines of alternating red and green pixels followed by odd scan lines of alternating green and blue pixels (termed the Bayer mosaic); the camera converts these to color pixels, typically in YUV422 format. This conversion doubles the bandwidth, and would necessitate halving the number of cameras or the frame rate on the limited IEEE 1394 bus.
2. Employ a fast and specialized foreground contour extractor. In one pass over the image, ours determines the major foreground objects, ranks them by variation from the background, and accommodates to luminance changes – both shadows and gradual light level fluxuations. 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 adjusting 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.

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

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

3. Reduce foreground contour complexity by using piecewise linear approximations. 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.

Figure 6. Contour of 525 segments

Linear approximations with (max pixel error, segments) = (4,66) : (8,22) , (16, 14).

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.
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.

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 (see

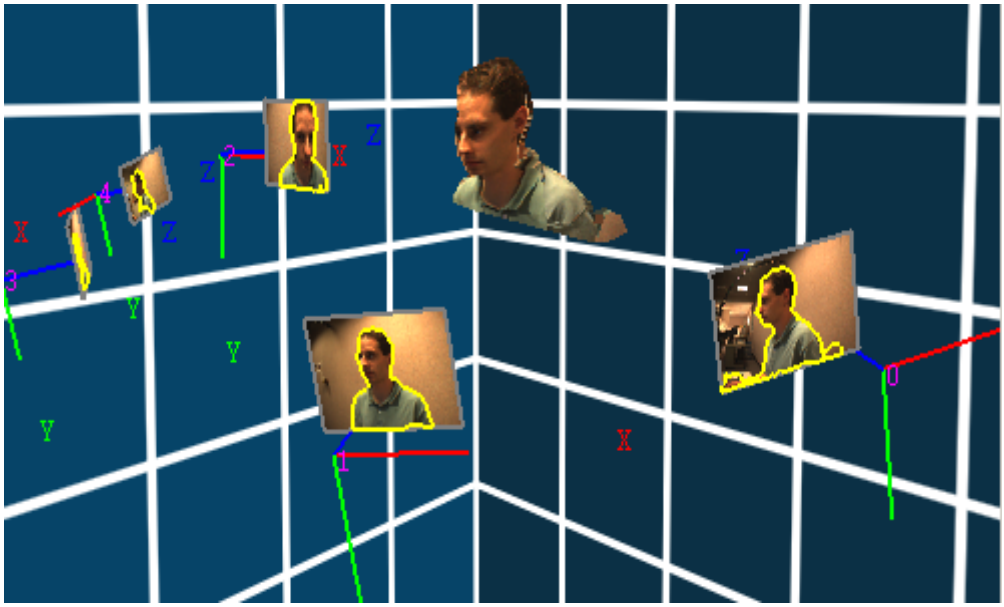

Figure 7. View of user in Coliseum space: Five cameras surround the rendered user. Each camera shows its coordinate system (in RGB), video frame, and

Figure 8). 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. 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 an extension to IBVH called Image-Based Photo Hulls (IBPH) [Slabaugh 02] which refines the visual hull geometry by matching colors across the images. This results in a tighter fit to the true scene geometry.

Using an assumption of Lambertian reflectance, the voxel coloring method of [Seitz 97] can be used to test whether a 3D point lies on a surface—points that project to similar colors in all the images that see them are termed *photo consistent*. IBPH uses this test to refine the coarse visual hull produced by IBVH. IBPH iteratively evaluates the photo consistency of IBVH surface points by moving them a small step in or out with respect to the synthesized viewpoint in a process similar to space carving [Culbertson 99]. Refinement is continued until the geometry projects to photo-consistent colors in the images.

Figure 8 shows that the improved geometric accuracy of the photo hull can make significant improvement to the quality of the synthesized view. In the original IBVH-produced image, adjacent textures from different cameras appear to be misregistered because of the inaccurate geometry; this error is reduced in the IBPH image. The depth maps reveal the increased geometric detail of the IBPH model that leads to this improvement.

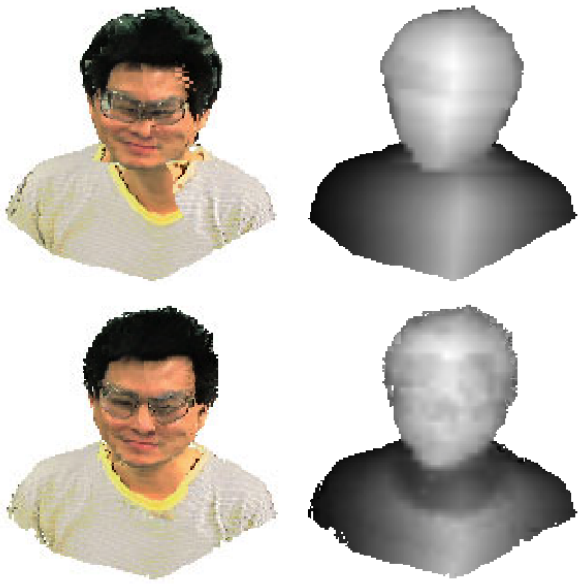

Figure 8. IBVH (top) versus IBPH (bottom). Texture-mapped model (left) and depth map (right).

3.3 Motion parallax

A successful tele-immersion system will make its users feel part of a shared virtual environment. The use of low-latency motion parallax (quantified in [Regan 99]) will provide this. Considering the screen as a window onto a virtual world, head movement should induce a corresponding view change. To achieve this, we track user head position and update the display as appropriate.

Head position is estimated using an image-based tracker that operates by fitting a 3D ellipsoid to the silhouette contour of the head in each of the five Coliseum camera views. Since these silhouettes are already being used for reconstruction, the additional cost of the head tracker is fairly small.

The tracker operates in linear and nonlinear modes. In the linear mode, a 2D ellipse is fit to the head contour in each image independently using robust reweighted least squares [Fitzgibbon 99]. These head contours are obtained from that part of the foreground silhouette above an estimated shoulder height. For stability, this is heavily filtered with a time constant of approximately one second. The 3D head position is estimated as the point of closest intersection of the rays obtained by back-projecting the ellipse centers. In the nonlinear mode, we model the head as a 3D ellipsoid of fixed size, and use Levenberg-Marquardt optimization to minimize an error based on the distance of head contour points from the ellipsoid's projection.

The linear mode generates an independent solution for each frame, but the standard deviation of its estimates (8 mm) is over twice that of the nonlinear approach (3 mm). The nonlinear mode, on the other hand, requires a good initial estimate from the previous frame. Therefore, we use linear mode to initialize when a user first appears or immediately after an unexpected failure, and then switch to nonlinear mode for better incremental performance. A simple adaptive filter is used to eliminate jitter when the user's head is motionless.

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 9 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 in intensity 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 95]. Intrinsic parameters are derived from the homographies that rectify the colored squares from their projected shapes [Zhang 00]. 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 [Longuet-Higgins 81] 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.

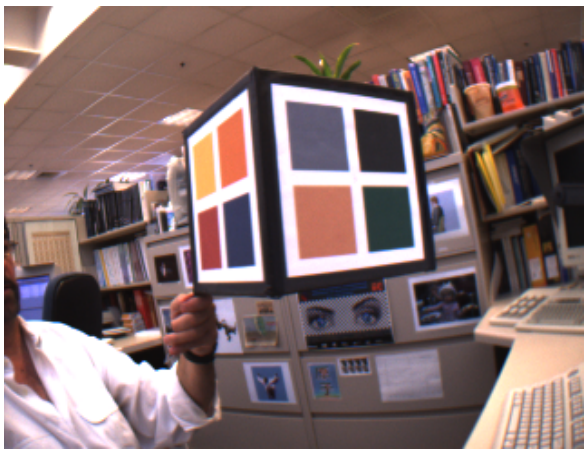

Figure 9. Calibration Target

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 multi-stream UDP protocol allows coordination of different media-type network transmissions. Figure 10 illustrates the dynamic structure of a Coliseum application with session management.

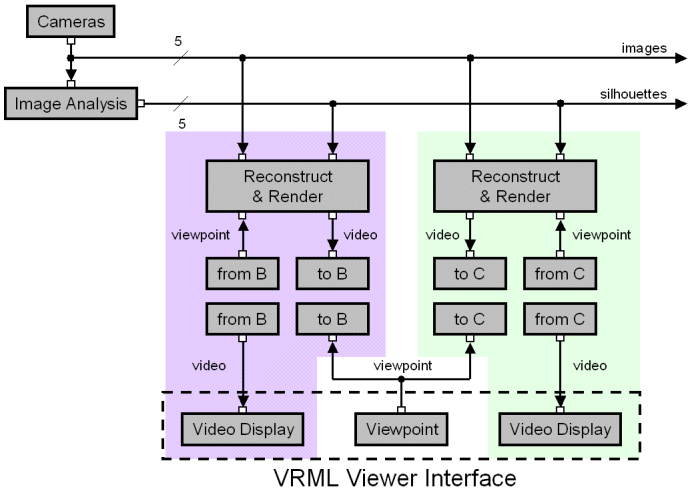

Figure 10. Coliseum scalable processing pipeline: On a single participant's Coliseum station, the shaded sub-pipelines are added and subtracted from the application as remote participants enter and leave the conferencing session.

3.6 Software framework

Streaming media applications are difficult to develop:

1. Digital media processing and services are inherently complex and often require orchestration among multiple developers.
2. These applications require simultaneous processing of multiple content streams which implies use of advanced programming techniques such as multithreading and synchronization objects.
3. To deliver a real-time user experience, optimal performance is required on limited computational resources, and this necessitates flow control and buffer management.

In implementing Coliseum, we have created a flexible, multi-platform, 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 the parallelism among those tasks.

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:

1. A designer decomposes the application into well-defined, connected task modules to isolate the complexity of each processing stage.
2. An automatic scheduler analyzes the task decomposition and then automates parallelism during execution.
3. 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 both Windows 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 the framework and the reusable components.

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

1. ***Media*** objects represent time-stamped samples of a digital signal, such as audio or video. A memory manager tracks the use of Media objects through reference counting and, for efficiency, reuses memory whenever possible. The new Media abstraction inherits an automatic serialization mechanism for writing into pipes, such as a file or network buffer.
2. ***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.
3. ***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:

1. 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.
2. 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.
3. The graph structure supports instrumentation. Keeping a functioning Graph intact, a new Task can connect to any of its output pins to monitor activity. This ability to "listen" is useful for gathering statistics in a performance monitor or to effect feedback control in modifying system parameters on the fly.

The alternative to developing our own infrastructure was to use an existing system such as Microsoft's DirectShow. This was not acceptable. Aside from being complex to learn and use, and dependent on other layers beyond our control (such as COM), DirectShow's use of processes rather than threads makes it poor for debugging multi-stream video applications. In addition, its lack of a media scheduler means it employs a philosophy of discarding unused work rather than preventing it from being done in the first place. This wastes capability in resource-critical applications.

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 videoconference functions with sufficient responsiveness, audio and video quality, and perceptual realism to take and hold an audience. Our objective of developing this

facility to run on single personal computers means we have to reach high levels in our analysis algorithms as well as in their performance. Algorithm success will be judged through observation of users at work with the system. Before we reach this point, we must ensure that Coliseum is structured to support the frame rates and latencies needed for an acceptable user experience.

We have begun evaluating 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.

The following tests were conducted using 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. The network was in use at the time for other HP Lab activities. 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 Latency

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

- ***Camera latency***: the time between an event occurring and the camera making its observations available to a local application.
- ***Processing Latency:*** the time it takes after receiving the image data from the camera system to process it, create the visual hull, render a requested view for a receiver, encode this in MPEG4, and then package and ship it to the intended recipient via UDP.
- ***Network Latency***: the time from transmittance of the packets from the sender to their receipt by the receiving host, typically two switched gigabit network segment hops.
- ***Display Latency:*** the time between packet reception, dataset reassembly, MPEG decoding, and return from providing the data to the display drivers.

Measuring camera system latency requires an externally observing capture device. We used a field-interlaced digital video camera to simultaneously image both an event and the display of that event. 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. Manual frame-by-frame analysis of the acquired video provided the numbers we sought. We captured several such events, and the table below indicates average values.

We measured camera latency in four situations:

1. A simple camera driver demonstration program (*TimeSliceDemo).*
2. A standalone version of Coliseum with no network or VRML activity (*Carver*).
3. A Coliseum test of two users with live networking, with and without MPEG encoding.
4. A Coliseum test where the subject is both the sender and receiver of the view (a single person loop-back conference).

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 11 gives the average video frame count (33 ms each) for each test. The observing video camera captured 30 frames per second, permitting us to calculate latencies and standard deviations. *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 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 12 indicates measures of the instrumented latency of this same system version and, comparing the Coliseum with MPEG user 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 32 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. The difference between these 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 uses a busy-wait loop for its user interface (this will soon be changed).

To determine the contributions to our latency, we manually instrumented the code both for the Carver application (non-networked) and for Coliseum. Timing data were collected at various points in the code and recorded with the data set. Each data set contained time stamps indicating when the camera frame set was first available, image analysis and processing time. The receiving host recorded the time to decode and display the image. We determined the roundtrip time by adding all the timings from both the sender and receiver and subtracting the wait time for piggybacking the timing data on outgoing datasets. One-way latency is half this value.

Figure 13 presents data for the Coliseum and Carver tests. Carver is the non-networked version of Coliseum with no VRML viewer. We tested each application with four different subjects, a user, a prerecorded dataset of a user, a 5-gallon water bottle, and a prerecorded dataset of the bottle. 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 system without the

System	*Mean Latency* *Frames*	*(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

Figure 11. Absolute user-perceived latency tests.

Glview Loop-back Subject	*Latency*	*Difference*
User	233	38
Bottle	229	42

Figure 12. Instrument measure of latency (ms).

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%

Figure 13. Analysis of latency (ms)

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 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.

4.2 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 fps, we measured a typical Coliseum video stream to be 616 Kbps. While 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 network latency in our local area network where the two participant hosts were two network-switched hops apart. The average network latency was 3 milliseconds, 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 about 25 milliseconds.

4.3 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 14 shows that, as session size increased, system performance (fps) decayed 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. This indicates that the system adapts to more users in a work conserving manner. Figure 15 shows that bandwidth climbed from 616 Kbps to 1178 Kbps as the CPU utilization saturated and then leveled off until the session of 6 users. All in all the bandwidth varies 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.

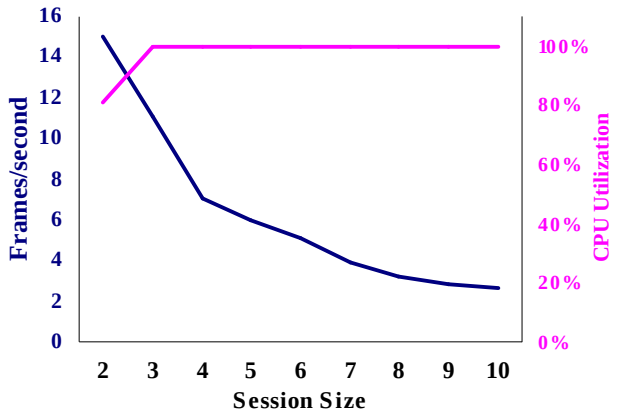

Figure 14. Change in frame rate (fps) and cpu utilization for increasing Coliseum session sizes.

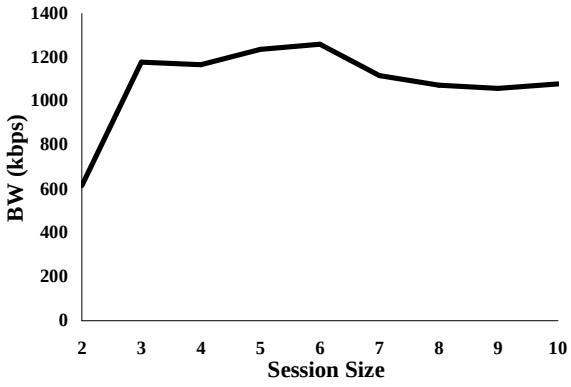

Figure 15. Aggregate bandwidth for size Coliseum sessions.

4.4 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 perceptual quality, and adjustment of these would allow support of more users while maintaining interactive rates. The current system reduces frame rate but maintains image quality. As the number 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 will direct our 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.

5 Future Work

There are numerous developments that remain in making Coliseum a viable alternative to travel for collaborative remote conferencing. Most immediately, we are working at increasing frame rate and reducing latency. We are also working on algorithms to improve the quality of the video—both the textures rendered and the geometry on which they rest. While focusing Coliseum on realistic depiction of people, we know that objects and documents also play an important role in collaborative interactions, and will be integrating imaged artifacts into the

shared virtual space. In addition, we are working at driving down the system's cost and getting it into the hands of study subjects to aid in our evaluation of user effects.

With the level of mediation we seek in Coliseum, we introduce issues not present in traditional conferencing, such as establishing participant layout and maintaining a mutually consistent environment. Since we control these issues, we additionally acquire the capability of moving beyond the physically realizable, and may consider the advantages of unorthodox participant configurations. This gives us the opportunity for super realism, such as coercing gaze to match intent when the acquired imagery doesn't provide this, highlighting the image of a speaker, incorporating documents and other artefacts into the shared world, placing participants around artefacts, interpreting gestures, et cetera.

6 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 depth. 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 in the past, actual implementations have been incomplete, operating only one-way, using cartoon avatars, or requiring substantial special purpose hardware. Using commodity PCs and simple video cameras, we have run fully symmetric Coliseum sessions with as many as ten users. Our focus now is on structuring our processing and communications for best performance, and then getting the system into user's hands for functionality studies.

Acknowledgements

Mike Harville, David Marimont, Greg Slabaugh, and Mat Hans have made important contributions to this project. Sandra Hirsh and Abigail Sellen are leading our user studies enquiries. Wojciech Matusik, Chris Buehler, and Leonard McMillan provided guidance on the original IBVH system.

References

[Baker 02] Baker, H. H., D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson and T. Malzbender, "The Coliseum Immersive Teleconferencing System," *Proc. International Workshop on Immersive Telepresence*, Juan Les Pins, France, December 2002, ACM Press.

[Chen 01] Chen, M. "Design of a Virtual Auditorium," *Proc. ACM Multimedia 2001,* Ottawa, September 2001, 19-28.

[Culbertson 99] Culbertson, W., Malzbender, T., Slabaugh, G., "Generalized Voxel Coloring," International Workshop on Vision Algorithms, 1999, Springer-Verlag LNCS 18883, pp. 100-115.

[Devernay 95] Devernay, F., Faugeras, O., "Automatic Calibration and Removal of Distortion from Scenes of Structured Environments," SPIE, volume 2567, San Diego, CA, July 1995.

[Fitzgibbon 99] Fitzgibbon, A., Pilu, M., Fisher, R., "Direct least-squares fitting of ellipses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), pp. 476-480, May 1999.

[Gharai 02] Gharai, L., C. Perkins, R. Riley, A. Mankin, "Large Scale Video Conferencing: A Digital Amphitheater," *Proc. 8th International Conference on Distributed Multimedia Systems*, San Francisco, CA, September 2002.

[Lanier 01] Lanier, J., "Virtually There," *Scientific American*, April 2001, pp. 66-75.

[Longuet-Higgins 81] Longuet-Higgins, H. "A Computer Algorithm for Reconstructing a Scene From Two Projections," *Nature* 293:133--135, 1981.

[Matusik 00] W. Matusik, C. Buehler, R. Raskar, S. Gortler, L. McMillan, "Image-based Visual Hulls," *SIGGRAPH 2000*, pp. 369-374.

[Narayanan 98] Narayanan, R., Rander, P., Kanade, T., "Constructing Virtual Worlds Using Dense Stereo," *Proceedings of International Conference on Computer Vision (ICCV)*, pp. 3-10, 1998.

[Pollefeys 99] Pollefeys, M., "Self-calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences," *Ph.D. Thesis*, ESAT-PSI, K.U. Leuven, 1999.

[Prince 02] Prince, S., Cheok, A., Farbiz, F., Williamson, T., Johnson, N., Billinghurst M., Kato H., "Real-Time 3D Interaction for Augmented and Virtual Reality", *SIGGRAPH 2002 Technical Sketch*, p.238.

[Regan 99] Regan, M, G. S. P. Miller, S. M. Rubin, C. Kogelnik, "A low-latency, real time hardware light field renderer," *SIGGRAPH '99*, 287-290.

[Seitz 97] Seitz, S., Dyer, C., "Photorealistic Scene Reconstruction by Voxel Coloring,", *Proceedings of Computer Vision and Pattern Recognition Conference*, 1997, pp. 1067-1073.

[Schreer 01] Schreer, O., Brandenburg, N., Askar, S., Trucco, E., "A Virtual 3D Video-Conferencing System Providing Semi-Immersive Telepresence: A Real-Time Solution in Hardware and Software," *Proc. Int.; Conf. on eWork and eBusiness*, Venice, Oct. 2001.

[Slabaugh 02] Slabaugh G., Schafer, R., and Hans, M., "Image-Based Photo Hulls," *1st International Symposium on 3D Processing, Visualization, and Transmission*, 2002, pp. 704-708.

[Wilcox 00] Wilcox J., *Videoconferencing, The Whole Picture*, Telecom Books, N.Y., ISBN 1-57820-054-7, April 2000.

[Zhang 00] Zhang, Z., "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11), pp. 1330-1334, November 2000.