Real-time Terascale Implementation of Tele-immersion

Nikhil Kelshikar[1⋆], Xenophon Zabulis[1], Jane Mulligan[4], Kostas Daniilidis[1], Vivek Sawant[2], Sudipta Sinha[2], Travis Sparks[2], Scott Larsen[2], Herman Towles[2], Ketan Mayer-Patel[2], Henry Fuchs[2], John Urbanic[3], Kathy Benninger[3], Raghurama Reddy[3], and Gwendolyn Huntoon[3]

[1] University of Pennsylvania
[2] University of North Carolina at Chapel Hill
[3] Pittsburgh Supercomputing Center
[4] University of Colorado at Boulder

Abstract. Tele-immersion is a new medium that enables a user to share a virtual space with remote participants, by creating the illusion that users at geographically dispersed locations reside at the same physical space. A person is immersed in a remote world, whose 3D representation is acquired remotely, then transmitted and displayed in the viewer's environment. Tele-immersion is effective only when the three components, computation, transmission, and rendering - all operate in real time . In this paper, we describe the real-time implementation of scene reconstruction on the Terascale Computing System at the Pittsburgh Supercomputing Center.

1 Introduction

Tele-immersion enables users at geographically distributed locations to collaborate in a shared space, which integrates the environments at these locations. In an archetypical tele-immersion environment as proposed at the origin of this project [8, 4], a user wears polarized glasses and a tracker capturing the head's pose. On a stereoscopic display, a remote scene is rendered so that it can be viewed from all potential viewpoints in the space of the viewer. To achieve this, an architecture that enables real-time view-independent 3D scene acquisition, transmission, and rendering in a real-time fashion is proposed (see Fig. 1). Most of the computational challenges are posed in the 3D scene acquisition. This stage deals with the association of pixels with the 3D coordinates of the world points they depict, in a view independent coordinate system. This association can be based on finding pixel correspondences between a pair of images. The derived correspondences constitute the basis of the computation of a disparity map, from which the depth of depicted world points and, in turn, their coordinates can be estimated.
The solution of the correspondence problem is associated with many challenging open topics, such as establishing correspondences for pixels that reside in textureless image regions, detecting occlusions, and coping with specular illumination effects.This involves a trade-off between being conservative and producing many holes in the depth or being lenient, covering everything with the cost of having outliers.

⋆ Contact person: Nikhil Kelshikar, nikhil@grasp.cis.upenn.edu, University of Pennsylvania, GRASP Laboratory, 3401 Walnut St., Philadelphia, PA 19104-6228. All three sites acknowledge financial support by the NSF grant IIS-0121293.

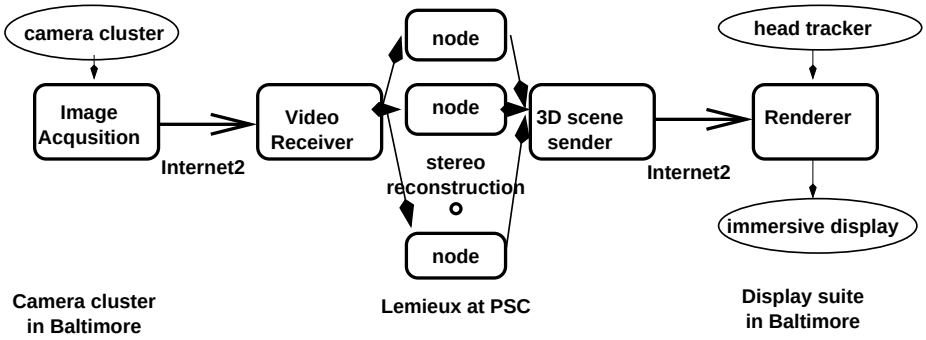

Fig. 1. System architecture. Images are acquired with a cluster of cameras, processed by a parallel computational engine to provide the 3D-description, transmitted, and displayed immersively.

As of summer 2001, we had achieved an 8Hz acquisition rate with limited depth quality, limited resolution and a very small operation space. Eventually, the original collaborative project [3] to produce a perceptually realistic tele-immersive environment, reached a level of maturity such that the remaining performance bottlenecks were well understood. We then established a goal to produce a real-time version with a dramatic increase in the volume of the scanned area, resolution, and depth accuracy. As opposed to other systems which capture one person inside a small area, we employed an array of stereo systems distributed so as they capture all actions from the span of all possible viewpoints of the remote viewer (wide-area scanning). Each stereo unit provides a $2 + \frac{1}{2}$D view as well as the correctly registered texture of this view.

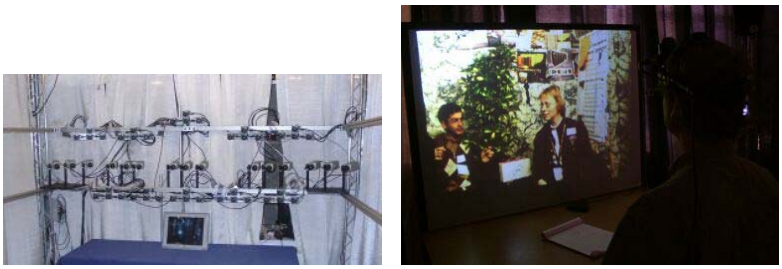

Fig. 2. 30 (of the 55 pictured) cameras were used in the November 2002 Supercomputing Conference demonstration (left). The acquired scene computed at PSC was displayed immersively (right).

To achieve this dramatic improvement in space and resolution, while maintain real-time performance, a significant increment of computational power is required, boosting the computational requirements to the supercomputing level. Such a resource was available at a remote location and, thus, we established one of the first applications where sensing, computation, and display are at different sites and coupled in real time. To overcome the transmission constraints, an initial implementation contains a video server transmitting video streams using TCP/IP and a reliable UDP transmission of the depth maps from the computation to the display site.

1.1 Computational and Networking Requirements

A single camera unit requires a triad of cameras to be grabbing pictures at 30 frames/second at VGA resolution. We use two monochromatic cameras (8 bits per pixel) and one color camera (8 bits per pixel; the color conversion is done in software). The images are captured at 640×480 pixel resolution. This produces data of 7.03 Mbits/frame. At the data capture rate of 30fps, we would produce data at 205.7 Mbits/sec. For each acquisition site we proposed to use a cluster of 10 or more camera units to adequately reconstruct the entire acquisition room, thus increasing data rates to 2.02 Gbits/sec per site. This is the amount of raw data that must make its way to the computing engine. The computing engine must execute $640 \times 480 \times 100$ depths $\times\ 31 \times 31$ kernel size $\approx 29.5G$ multiplications and additions per camera unit. The non-parallel version needed approximately $320 \times 240 \times 64$ depths $\times\ 5 \times 5$ kernel size $\approx 122M$ multiplications and additions per camera unit (actually twice as much because it used trinocular stereo). The produced 3D data stream is 11.72 Mbits/frame and consists of 16 bit inverse depth and a 24 bit color texture. This also scales as the number of camera units and users.

1.2 Related Work

Our real-time wide-area stereo scanning is not directly comparable to any other system in the literature. No other existing system combines viewpoint independent wide-area acquisition with spatially augmented displays and interaction. Though the explosion of Internet has produced many systems claiming tele-presence, none of them is working towards the scale necessary for tele-immersion. The closest to the proposed work is CMU's Virtualized Reality [7] and its early dedicated real-time parallel architecture for stereo. Other multi-camera systems include the view-dependent visual hull approach [5], the Keck laboratory at the University of Maryland [1] and the Argus system at Duke University [2].

2 Acquisition Algorithm

We now elaborate the main steps of the reconstruction algorithm [6] emphasizing the factors that affect the quality of reconstruction and the processing time. The initial implementation is based on two images but it is easily extensible to a polynocular configuration. We rely on the well-known stereo processing steps of matching and triangulation, given that the cameras are calibrated.

Rectification When a 3D-point is projected onto the left and the right image plane of a fixating stereo-rig the difference in the image positions is both in horizontal and vertical directions. Given a point in the first image we can reduce the 2D search to a 1D search if we know the so called *epipolar geometry* of the camera which is given from calibration. Because the subsequent step of correlation is area based, and for reduction of time complexity, we first perform a warping of the image that makes every epipolar line horizontal. This image transformation is called *rectification* and results in corresponding points having coordinates (u, v) and $(u - d, v)$, in left and right rectified images, respectively, where d is the horizontal disparity.

Matching: The degree of correspondence is measured by a modified normalized cross-correlation measure $c(I_L, I_R) = \frac{2\ cov(I_L, I_R)}{var(I_L) + var(I_R)}$., where I_L and I_R are the left and right rectified images over the selected correlation windows. For each pixel (u, v) in the left image, the matching produces a correlation profile $c(u, v, d)$ where d ranges over a disparity range. The definition domain is the so called *disparity range* and depends on the depth of *working volume*, i.e. the range of possible depths we want to reconstruct. The time complexity of matching is linearly proportional to the size of the correlation window as well as to the disparity range.
We consider *all* peaks of the correlation profile as possible disparity hypotheses. This is different from other matching approaches which early decide on the maximum of the matching criterion. We call the resulting list of hypotheses for all positions a *disparity volume*. The hypotheses in the disparity volume are pruned out by a *selection procedure* based on the constraints imposed by the following:

- Visibility: If a spatial point is visible then there can not be any other point in the viewing rays through this point and the left or right camera.
- Ordering: Depth ordering constrains the image positions in the rectified images. Both constraints can be formulated in terms of disparities without reconstructing the considered 3D-point.

The output of this procedure is an integer *disparity map*. To refine the 3-D position estimates, a *sub-pixel correction* of the integer disparity map is computed which results in a sub-pixel disparity map. To achieve fast sub-pixel estimation we fit a quadratic polynomial on the five-neighborhood of the integer disparity at the correlation maximum.

Reconstruction Each of the stereo rigs is calibrated before the experiment using a modification of Bouguet's camera calibration toolbox. Given estimates of the two, 3×4, projection matrices for the left and the right camera and the disparity at each point the coordinates of a 3D-point can be computed.

Color Image Warping The stereo cameras used to compute the 3D points are monochromatic cameras. A third color camera is used to color the 3D points. The calibration technique also estimates the projection matrix for the color camera. The projection matrix is used to compute a lookup table of where the 3D point lies in the color image. This lookup table is to map color to the 3D point set.

Depth Stream Next, the 3D depth and the color image must be sent to the remote viewer. Depth is encoded into a 3D stream which consists of a 16bit inverse depth image and a 24 bit RGB color image. This stream is then encoded in a raw byte format and transmitted over the network. The renderer also receives (once, during initialization) the inverse projection matrix for mapping the viewer coordinate system to the world coordinate system.
The error in the reconstruction depends on the error in the disparity and the error in the calibration matrices. Since the action to be reconstructed is close to the origin of the world coordinate system the depth error due to calibration is negligible compared to the error in the disparities. The principal concern is the number of outliers in the depth estimates which result in large peaks usually appearing near occlusion or texture-less areas.

3 Rendering

It is the task of the rendering system to take the multiple independent streams of 3D depth maps and re-create a life-size, view-dependent, stereo display of the acquired scene. Received depth maps are converted into 3D points and rendered as point clouds from a user tracked viewpoint. Multiple depth map streams are time synchronized and simply Z-buffered to create a composite display frame. Display is accomplished using a two-projector passive stereo configuration, and the user's eye positions are estimated with a HiBall wide-area tracker as described in [3].

While it is relatively easy to scale the video capture and reconstruction front-end, it is a most difficult task to architect a rendering system that is not the system bottleneck. At 640×480 resolution, each depth map stream arriving at the rendering system includes 11.73 Mbits per frame. Without data compression, 10 streams operating at 10 fps can generate a 1.3 Gbps data rate. At 80 per cent reconstruction efficiency (or approximately 250K points per 640×480 resolution stream), ten streams tend to produce 2.5M points to render requiring a point rendering performance of greater than 75M points/sec for truly smooth 30Hz, view-dependent rendering.

Our current system is architectured around a three-PC Linux cluster interconnected with a gigabit network. One PC serves as the network aggregation node for the multiple depth map streams arriving from the Terascale Computing System (TCS) platform. The other two PCs render the left and right-eye views. For performance reasons, data arriving into the aggregation node are multicast using the UDP protocol to the two rendering nodes. These PCs are 2.4GHz dual-processor Dell workstations and the rendering is facilitated by Nvidia GeForce 4 ti4600 cards. Using `Vertex_Array_Range` extensions rather that OpenGL displays, we are able to render 3D point clouds with up to 2M points at 30Hz. The 30Hz view-dependent display loop runs asynchronous to depth map stream update rate, which was limited to approximately 1Hz during the SC02 tests to avoid increasing latency and network buffering. Frames in each stream are time stamped so the render system can re-synchronize these independent streams. Frame swap synchronization between left and right-eye PCs is achieved with simple out-of-band protocols.

4 Terascale System/Parallel Porting Plan

The original development system consisted of a triple of cameras connected to a dual processor machine running at 2.4 GHz. There were five such camera triples connected to five different servers. Each server was used to grab three 320×240 pixel images. The acquired data was processed locally. To reduce the processing time, only the foreground was processed for stereo matching and reconstruction. The serial system used two processors per stream. Each of the processors processed half the image. The algorithm used small correlation kernels of 5×5 size. This system runs at 8 frames per second. The main bottleneck here is the processing hardware.

The quality of the reconstruction was not satisfactory and the real-time requirement precluded the use of any sophisticated algorithms. The images used were low resolution. The use of background subtraction in the images eliminated 66% of the data, hence the viewer could only see the remote participant in an artificial surrounding.

Complete reconstruction of the scene from more cameras using higher resolution images and more sophisticated algorithms, requires much more processing time. The real-

time constraint of this system required us to harness much more processing power. It became obvious that this serial based system would have to be migrated to a parallel platform. The platform chosen was the Terascale Computing System at the Pittsburgh Supercomputing Center. It comprises 3000 1GHz Alpha processors and is called Lemieux. The key parallelization insights were:

- The problem decomposes naturally by each camera stream.
- Serial image analysis code can remain fundamentally the same in the parallel implementation.
- Each processor would process a fraction of the image.

It was decided that a parallel framework, properly constructed, would allow the retention of the serial image analysis code and approach without sacrificing excellent scalability to thousands of processors. In addition, past parallel coding experience led to the incorporation of several other design goals:

- Define explicit parallelization interface to existing serial code.
- Anticipate the need for run-time debugging.
- Demand full scalability - allow no serial code to remain, introduce none.
- Permit partial stream asynchronicity during development, but demand fully asynchronous code when done.
- Design with physical I/O constraints in mind.

In addition, previous experience in development of real-time (defined as maximum allowable latency) codes on other Unix-like parallel platforms led us to anticipate that system services, and particularly ”heart beats” would be problematic. Code with maximum tolerance would be necessary.

5 Data Flow

The parallel framework that was developed was based on servicing each stream independently of others, of scaling within each stream and assuring that physical I/O points on the system were not allowed to become bottlenecks. The resulting schematic for a single stream looks as shown in Fig. 3, left. Data is received over the Internet into a designated Lemieux input node equipped with a 100Mb ethernet interface. It is then divided up into congruent horizontal bands for each of the three cameras and distributed to a variable number of computational nodes for image analysis. These analyzed images are then gathered to a single node, which combines them into a processed frame, and broadcast over the Internet to a remote display station. Note that the streams retain full independence in this schematic. This was maintained in software, even while retaining a coherent single executable, with use of MPI communicator mechanisms.

6 Performance

The switch to a terascale system allowed us to accommodate for more computationally intensive algorithms with higher quality images. We incorporated several changes notably changing the search area and the correlation kernel size. However, the core stereo matching algorithm was not changed, only the parameters used to operate it were tuned to increase match reliability.

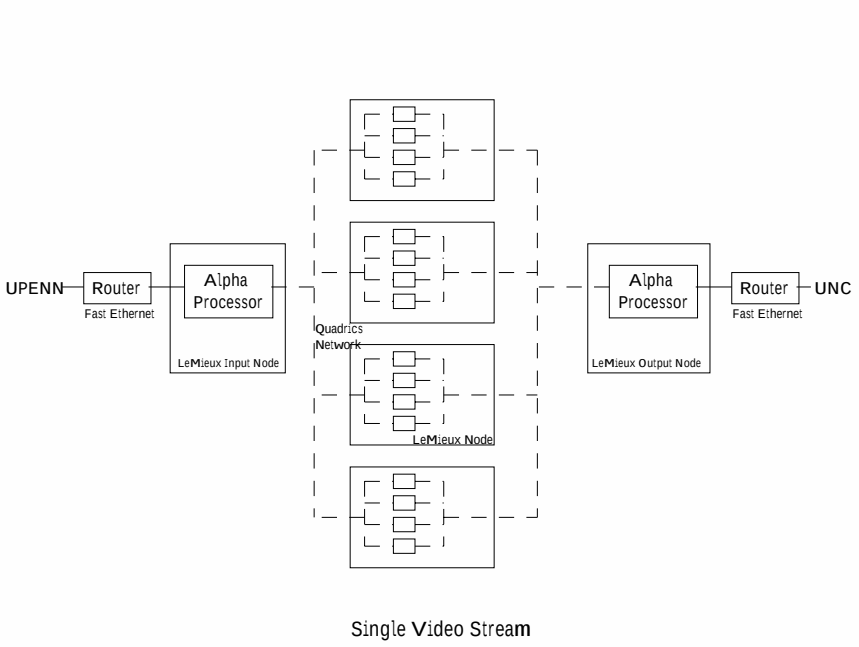

Fig. 3. Single stream data flow.

Computation The parallel system operates on images four times the size of those used in the serial algorithm. The correlation window size is 31×31 rather than 5×5 increasing computation approximately 36 times. However we use binocular instead of trinocular stereo due to which we have to perform matching once rather than performing pairwise matching. Thus, the new system requires at least 72 times more computation. Since we do not perform background subtraction, an additional order of magnitude of complexity is required.

The correlation window size is the main parameter affecting performance. The complexity of the stereo algorithm is $\mathcal{O}(m^2n^2)$ where m^2 is the size of the correlation kernel and n^2 is the size of the image. We ran a series of tests to verify the performance and the scalability of the system. The performance of the real-time system, with networked input of video and network output of 3D streams, is constrained by many external factors which could cause bottlenecks. Hence for performance analysis of the parallel algorithm we switched to file based I/O. The image streams are read from disk and we measure the time for image distribution on the cluster, image analysis and 3D data gathering from the various cluster nodes which contribute to total processing time.

The reconstruction algorithm broadcasts the image to be processed on a particular node in its entirety. Hence as the number of PE's used for the particular stream increases, so does the broadcast time, as shown in Fig. 4, left. Each processor would then perform stereo matching on a small strip of the entire image. This is the lowest level of parallelization. As the number of processors increases, each processor processes fewer pixels. Fig. 4 (right) shows the speedup in the process frame routine which performs

image rectification, stereo matching and the reconstruction of the 3D points. We show the processing time for seven different correlation window sizes. The reconstructed 3D

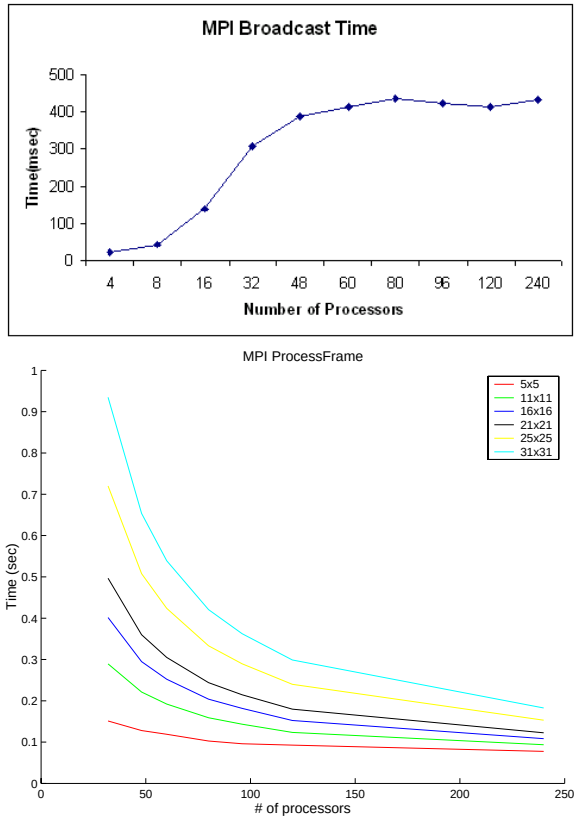

Fig. 4. Left: The time required to broadcast images to each node increases as the number of processors increases. Right: Total processing time in msec vs number of processors. Each plot corresponds to a different kernel size.

points have to be re-assembled as different parts of the images are reconstructed on different nodes. This gather operation speeds up with number of processors because less data must be gathered from each node.

Based on the above studies we have observed that the algorithm scales very efficiently with the number of processors per stream. The program parallelizes so that each stream is synchronized when reading data and runs independently of each other and hence does not affect individual performance. Each stream of images has similar parameters and hence execution time is almost the same.

Networking We conducted several networking and performance tests during the Supercomputing 2002 (SC2002) Conference in Baltimore, MD. The video servers and the rendering system on SC2002 exhibition floor communicated with the PSC Lemieux system over Abilene, the high-speed Internet2 backbone. The video servers and the

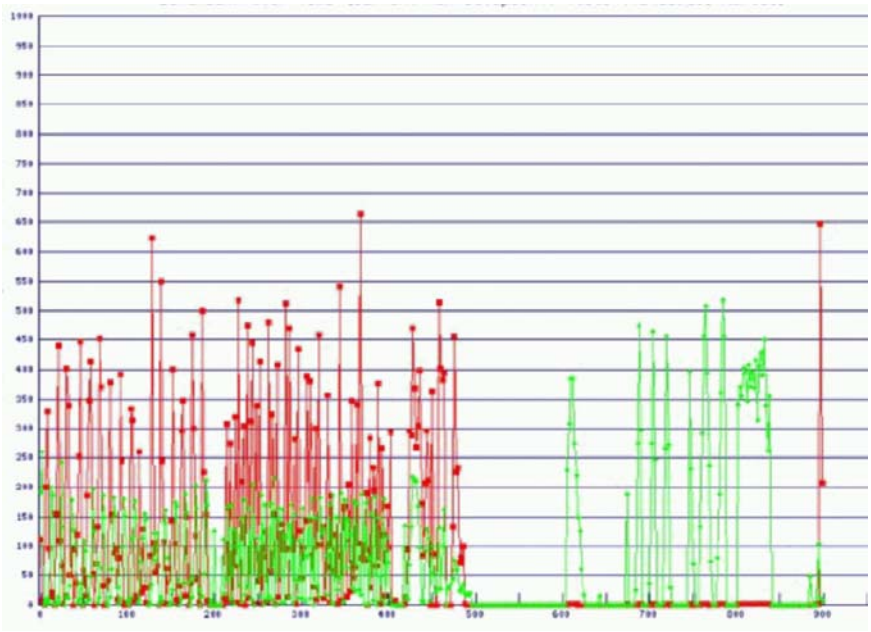

Fig. 5. Bandwidth usage plot from Nov 19 2002 SuperComputing 2002 demonstration, showing Mbps over time. The darker line (red) shows the data traffic from acquisition to computation site, while the lighter line)green) shows the traffic from computation to display site.

rendering system were each connected to Abilene by separate Gigabit links. Tests were performed with nine streams of images (from nine video servers.) The image data originating at video servers was transmitted to Lemieux using TCP. The data was processed on Lemieux and the depth/texture data sets were (optionally) sent back to the rendering system, using Reliable UDP (RUDP), a protocol specifically designed by us for this application. RUDP provides reliable data transmission required by the application without any congestion control, thereby providing better throughput than TCP.

In Fig. 5, shown is the bandwidth usage for the run during the “Bandwidth Challenge” at SC2002. The run consists of two phases. The first phase involved the entire system from image acquisition to rendering. An average frame rate of 1 frame/sec was achieved during this phase. The average data rate from video servers to Lemieux over TCP was 63 Mbps with peak rates over 200 Mbps. From Lemieux to the rendering system, RUDP was able to achieve data rates of over 105Mbps with peaks over 700Mbps (see the caption on the plot.) The frame rate achievable in this end-to-end operation was limited by the capacity of the rendering system to consume the depth/texture data. In the second phase, the rendering system was left out. The image data was processed at Lemieux as before, but it was not transmitted back to the rendering system. In this phase, the frame rate up to 8 frames/sec was achieved and the data rate over 500 Mbits/s was observed for the image transmission over TCP. The burstiness of the traffic is due to synchronous or lock-step operation of the system. During the entire run, 1080 processors (120 per

stream) were employed for the stereo reconstruction on Lemieux. The output of each processors are depth estimates for 4 rows of 640 pixels width.

7 Conclusions - Outlook

We have ported a real-time application from a dedicated serial environment to one that crosses a wide area network and utilizes a centralized parallel computing resource. The parallel code demonstrates excellent scalability and continues to exploit a friendly segmentation between the image analysis and parallel framework which allows concurrent work by all development groups. This nearly perfect scalability, both by PE per stream and by stream, makes us optimistic that we will be able to continue our overall performance gains by the three routes of:

- Better per processor performance through optimization
- Larger machine size runs on a routine basis
- Larger platforms that become available.

We are easily capable of saturating our current, routinely available networking connection. Imminent network enhancements will permit progress on that front. To improve reconstruction quality, we will employ an adaptive window size in order to preserve discontinuities as well as inter-scanline processing to alleviate rectification inaccuracies. In the quest for further performance improvements, we are actively investigating advanced rendering techniques including moving the conversion from depth maps (integer) to 3D points (floats) from the CPU into the graphics hardware. In the future, we would like to apply image compression techniques to reduce the data bandwidth requirements. We are also exploring the issue of latency and susceptibility to network congestion to develop a simple protocol that will minimize both, improving multi-stream throughput. The task of developing a rendering architecture that will scale as easily and linearly as independent camera-capture/reconstruction streams remains a significant research challenge.

References

1. P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *IEEE Workshop on Omnidirectional Vision*, Hilton Head Island, SC, June 12, 2000.
2. D.J. Brady, R.A. Stack, S. Feller, L. Fernandez E. Cull, D. Kammeyer, and R. Brady. Information flow in streaming 3d video. In *Three-Dimensional Video and Display Devices and Systems, SPIE PRESS Vol. CR76*, 2000.
3. H. Towles et al. 3d tele-collaboration over internet2. In *International Workshop on Immersive Telepresence, Juan-les-Pins, France, 06 Dec*, 2003.
4. J. Lanier. Virtually there. *Scientific American*, pages 66–75, April 2001.
5. W. Matusik, C. Buheler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of ACM SIGGRAPH*, 2000. to appear.
6. J. Mulligan, V. Isler, and K. Daniilidis. Trinocular stereo: A new algorithm and its evaluation. *International Journal of Computer Vision*, 47:51–61, 2002.
7. P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. Int. Conf. on Computer Vision*, pages 3–10, 1998.
8. R. Raskar, G.Welch, M.Cutts, A.Lake, L.Stesin, and H.Fuchs. The office of the future: A unified approach to image- based modeling and spatially immersive displays. In *ACM SIGGRAPH*, pages 179–188, 1998.