High-Quality Visualization for Geographically Distributed 3-D Teleimmersive Applications

Ramanarayan Vasudevan, *Student Member, IEEE*, Gregorij Kurillo, Edgar Lobaton, *Member, IEEE*, Tony Bernardin, Oliver Kreylos, Ruzena Bajcsy, *Life Fellow, IEEE*, and Klara Nahrstedt, *Fellow, IEEE*

***Abstract*—The growing popularity of 3-D movies has led to the rapid development of numerous affordable consumer 3-D displays. In contrast, the development of technology to generate 3-D content has lagged behind considerably. In spite of significant improvements to the quality of imaging devices, the accuracy of the algorithms that generate 3-D data, and the hardware available to render such data, the algorithms available to calibrate, reconstruct, and then visualize such data remain difficult to use, extremely noise sensitive, and unreasonably slow. In this paper, we present a multi-camera system that creates a highly accurate (on the order of a centimeter), 3-D reconstruction of an environment in real-time (under 30 ms) that allows for remote interaction between users. This paper focuses on addressing the aforementioned deficiencies by describing algorithms to calibrate, reconstruct, and render objects in the system. We demonstrate the accuracy and speed of our results on a variety of benchmarks and data collected from our own system.**

***Index Terms*—Human-computer interaction, real-time, stereo reconstruction, virtual reality, visualization, 3-D teleimmersion, 3-D video.**

I. Introduction

ROBUST accurate real-time generation of 3-D data from real-life scenes has proved extremely difficult. In fact, most of the content we now enjoy on 3-D displays is either generated entirely offline or is synthetically generated. In this paper, we focus on a particularly important application of 3-D content generating technology: video conferencing systems.

Most existing video conferencing systems make some attempt to humanize remote interaction, but few are able to provide the desired immersive component of actual face-to-face communication. These systems, which rely on two-dimensional video streams between remote users, fall short of providing the desired immersive component for a number of reasons including not allowing users to establish eye contact, not placing all users inside the same environment, or not allowing users to jointly interact with synthetic objects. Limited attempts have been made to create a more immersive experience using large displays, gaze preservation through multi-camera capturing systems [1], and matching environments (e.g., tables, chairs) between the remote locations that create the illusion of continuity of the physical space into the screen. In contrast to such systems, an immersive experience, as illustrated in Fig. 1, is one that generates a full body real-time 3-D reconstruction that realistically represents a user's appearance and completely models the dynamics of movement such as facial expressions, chest deformation during breathing, and movement of hair or clothing.

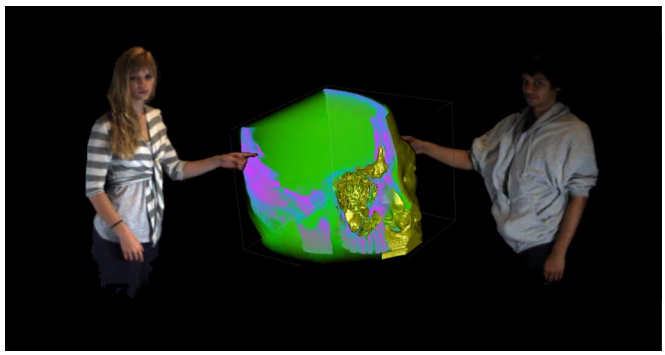

Fig. 1. Snapshot of a 3-D reconstruction of two users in separate Teleimmersion systems discussing a synthetic dataset in a virtual environment.

Teleimmersion or TI is an emerging technology that allows users to collaborate remotely by generating a realistic 3-D representation of users in real-time and placing them inside a shared virtual space [2]. Such virtual meeting spaces could allow for the possibility of collaborative work on 3-D data such as medical data, scientific data, and design models or the remote teaching of physical activities. Unfortunately, the accurate construction of 3-D data at high frame rates has always been the common shortcoming of TI systems. In this paper, we describe a multi-camera system that creates a highly accurate (on the order of a cm), real-time (under 30 ms), 360° 3-D reconstruction of users. The main contributions of this work are three-fold. First, in Section IV, we develop a robust calibration of the multi-camera system and the physical space to ensure the preservation of spatial correspondences between various TI systems. Second, in

Manuscript received November 01, 2010; accepted January 24, 2011. Date of publication March 10, 2011; date of current version May 18, 2011. This work was supported in part by NSF (grants: 0703787, 0724681, 0937060), HP Labs, EADS, and CITRIS at University of California, Berkeley. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhengyou Zhang.

R. Vasudevan, G. Kurillo, and R. Bajcsy are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: ramv@eecs.berkeley.edu; gregorij@eecs.berkeley.edu; bajcsy@eecs.berkeley.edu).

E. Lobaton is with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599 USA (e-mail: lobaton@cs.unc.edu).

T. Bernardin and O. Kreylos are with the Institute for Data Analysis and Visualization, University of California Davis, Davis, CA 95616 USA (e-mail: tbernardin@cs.ucdavis.edu; kreylos@cs.ucdavis.edu).

K. Nahrstedt is with the Department of Computer Science, University of Illinois at Urbana Champaign, Urbana, IL 61801 USA (e-mail: klara@cs.uiuc.edu).

This paper has supplementary downloadable material available at http://ieeexplore. ieee.org. The total size is 63.5 MB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2011.2123871

Section V, we introduce a novel multi-scale representation that allows for a highly accurate reconstruction of a scene while allowing for high compressibility of the produced 3-D data. Third, in Section VI, we describe how the data produced from the multiple cameras can be integrated together to improve the overall visual quality of the reconstruction. The rest of the discussion includes a brief overview of related work in Section II, an overall description of the system in Section III, and a description of various applications that illustrate the robustness and utility of the proposed system in Section VII.

II. Related Work

In the past decade, several have attempted to develop real-time 3-D reconstruction systems for marker-less capture of the human body for telepresence in a virtual environment. Most real-time approaches fall into one of three categories depending upon their computational approach: 1) image-based reconstruction using dense stereo, 2) voxel-based methods with spatial sampling, and 3) silhouette-based reconstruction.

We begin by describing several systems that perform 3-D reconstruction using image-based reconstruction. In 1999, Kanade *et al.* presented one of the first full-body capturing systems using image-based reconstruction with a large number of distributed cameras to capture human movement at close to real-time (less than a single frame per second) [3]. The first TI system was developed by researchers at the University of Pennsylvania who used several stereo camera triplets for image-based reconstruction of the upper body [4]. Sang *et al.* presented a faster TI system based on dense-stereo depth maps obtained via trinocular 3-D reconstruction [5]. The presented approach, however, had two major shortcomings: its slow speed and its unreliability in texture-less regions of the scene which resulted in missing data.

Several systems have employed voxel-based methods to perform 3-D reconstruction. Hasenfratz *et al.* proposed a voxel-based interactive virtual reality system that featured real-time (25 frames per second) 3-D reconstruction of the human body [6]. However, the approach had limited ability to acquire details such as clothing and facial features due to the inherent limitations of space carving methods. Schreer *et al.* attempted to overcome this limitation by incorporating a depth estimate from an image-based reconstruction [7]. The system, however, required a large number of cameras to generate a 3-D model of the user.

Several systems perform 3-D reconstruction using silhouette-based methods. Baker *et al.* proposed a desktop 3-D reconstruction systems that used five different views to obtain a 3-D model of the user via a visual-hull approach [8]. The system was run on a single PC which performed 3-D reconstruction and rendering of the users in a simple virtual meeting room. A full-body TI system was introduced by Gross *et al.* who applied silhouette-based 3-D reconstruction from several views to capture the user inside a CAVE-like environment [9]. Two such systems were built to offer a collaborative platform connecting geographically distributed users. Unfortunately, silhouette-based methods fail to reconstruct concave regions in the scene and tend to be inaccurate unless numerous cameras are employed.

In recent years, several benchmarks to compare the accuracy of various types of stereo reconstructions have become available. Scharstein *et al.* present a thorough overview of a variety of standard algorithms to perform 3-D reconstruction (most do not work in real-time) and provide a powerful benchmark that has become an industry standard [10]. A brief review of the results of a variety of algorithms on this benchmark illustrates rather clearly that image-based reconstructions, in general, provide far more accurate results than the other two approaches. Image-based approaches make fewer explicit assumptions about the object to be imaged and therefore tend to be more accurate at the expense of speed.

The use of multiple cameras also provides a distinct advantage over direct-ranging sensors (e.g., Kinect) which cannot satisfy the multi-viewer multi-viewpoint experience since critical areas of the scene (i.e., object boundaries, transparencies, etc.) are poorly reconstructed, poorly resolved, or missed altogether due to the nature of the underlying technology (e.g., multi-sampling within a measurement pixel). Redundancy of such sensors could potentially improve the experience; however, this approach would require complex hardware-based triggering to efficiently deal with interference between multiple range sensors.

TI systems, in general, have focused almost entirely on the speed of the system and have not focused on the accuracy of the reconstruction. In this paper, we describe a TI framework that employs image-based reconstruction to get accurate results in real-time. We compare our approach on the aforementioned benchmark to illustrate the strength of our method. The approach can be applied to various camera configurations, from a single stereo view to multi-view 360-degree reconstruction, to achieve accurate 3-D reconstruction.

III. System Overview

The goal of the TI system presented in this paper is to allow for geographically distributed collaboration in a shared virtual environment. To this end, each user interacts in the virtual environment via their own local TI station. Each station maintains a local copy of the entire virtual space in order to allow model manipulation and postprocessing of data locally. With these requirements in mind, each station must perform the following three tasks: 1) computation of a 3-D reconstruction of local objects, 2) communication of 3-D data to other stations, and 3) visualization of the virtual environment with other remote users.

An appropriate choice of representation for the data should take into consideration the aforementioned goals. Namely, the system should be able to reconstruct any object that is present at each station without making *a priori* assumptions about the object to be reconstructed. The 3-D reconstruction should also be fast and accurate, which suggests that the reconstruction should only occur at informative points and then be filled in using interpolation where appropriate. Since we require real-time streaming of 3-D data, we must choose a representation that allows for fast and efficient compression and decompression. Though each station creates a separate 3-D model of an object based on views from multiple camera clusters, in order to visualize the objects, these views must be integrated. With these requirements in mind, we argue that the ideal representation of an object from a view is a mesh as opposed to a set of scattered points without connectivity information. We describe the different components of the TI system as illustrated in

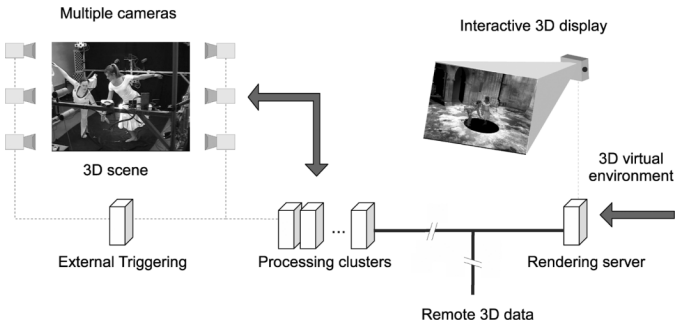

Fig. 2. Components of the TI system. Capturing component is displayed to the left. Data are then processed using a computing cluster. The model is transmitted over the network for display at another TI station.

Fig. 2. Though we describe two particular hardware instantiations of the TI system (a general and portable apparatus), we stress that the algorithms presented in the next few sections are independent of the particular hardware choices.

The general TI apparatus consists of 48 Dragonfly cameras (Point Grey Research Inc., Vancouver, BC, Canada), each with a resolution of 640 × 480 pixels. The cameras are arranged in 12 clusters of three grayscale cameras for stereo reconstruction and a color camera for texture acquisition. Ten of the clusters are equipped with 6-mm lenses while the remaining two clusters have 3.8-mm lenses for wider capture space. The clusters cover 360° of the workspace of about $2.0 \times 2.0 \times 2.5\ \mathrm{m}^3$. The cameras of each cluster are connected through IEEE 1394a (FireWire) interface to a dedicated server which performs image acquisition and stereo reconstruction. The server computers used for the reconstruction have two Dual-Core Intel Xeon 2.33-GHz processors, 2 GB of memory, and 1 Gbps connection to Internet 2. The portable TI apparatus consists of four color Bumblebee cameras (Point Grey Research Inc., Vancouver, BC, Canada), each with a resolution of 640 × 480 pixels. Each cluster is equipped with 3.8-mm lenses for wider capture space. The clusters cover 180° of the workspace of about $2.0 \times 2.0 \times 2.5\ \mathrm{m}^3$. The cameras of each cluster are connected through IEEE 1394a (FireWire) interface to a dedicated server which performs image acquisition and stereo reconstruction. The server computers used for the reconstruction have two Intel Dual-Core 2.33-GHz processors, 2 GB of memory, and 1 Gbps connection to Internet 2.

The display system for both systems consists of a rendering computer with Intel Dual Core CPU, 2.66-GHz processors, 2 GB of memory, and two NVIDIA GeForce 8800 GTS graphics cards. The renderer can receive compressed 3-D data directly from the cluster computers in separate network streams or indirectly through a gateway computer which can also connect to a remote site. The renderer supports different display options, such as single or multiple desktop displays and various passive or active stereo systems. The users can also use different interface devices (e.g., wireless 3-D mouse, Wii remote) to interact with the virtual environment.

IV. Calibration

In this section, we describe a hierarchical approach to the calibration of the TI system. In the lowest level, we calibrate internal camera parameters using Zhang's method [11] through homography obtained from a checkerboard. Once each cluster is internally calibrated, we can proceed with the external calibration of clusters to obtain their position and orientation with respect to a reference camera. The external calibration is performed using a calibration object fashioned out of two LEDs positioned at a fixed distance. Finally, we calibrate the display and the tracking system to determine the spatial relationship of the data with respect to a world coordinate system in the virtual environment. During our discussion, we assume some familiarity with calibration. A formal introduction to this topic can be found in [12].

A. Stereo Cluster Calibration

The cameras for the TI system can be arranged in various patterns, from simple two-camera clusters to multi-camera linear arrays. As long as the cameras within the cluster have large overlap, they can be calibrated simultaneously. Calibration of each stereo cluster is performed using Zhang's method. A planar checkerboard target is placed in different positions and orientations to generate a set of points for homography calculation. We use the standard pinhole camera model:

$$\mathbf{x}_i = \mathbf{K}_f \mathbf{\Pi}_0 \mathbf{G} \mathbf{X}_i. \tag{1}$$

The model in (1) represents the transformation from a homogeneous 3-D point $\mathbf{X}_i \in \mathbb{R}^4$ seen by camera to the corresponding image pixel coordinate $\mathbf{x}_i$ defined on the image plane. $\mathbf{K}_f \in \mathbb{R}^{3\times3}$ represents the camera matrix which contains the focal length and the optical center. $\mathbf{\Pi}_0 \in \mathbb{R}^{3\times4}$ is the standard projection matrix. $\mathbf{G} \in \mathbb{R}^{4\times4}$ contains rotational matrix and position of the camera center from the object coordinate system origin.

The calibration algorithm uses a set of known points on a planar checkerboard and a set of detected corner features on the image as their projection. A system of linear equations is formed and solved via singular value decomposition to obtain the initial value of the focal length, optical center, and distortion. The minimization function is defined as the reprojection error between M image points $\mathbf{x}_i$ obtained in P positions of the calibration board and the points projected through the camera model with the linear parameters as an initial guess. After each camera is calibrated independently, the relative orientation and position of the cameras within the cluster (R_{i0}, t_{i0}) are computed. The relative relationship between an arbitrary camera C_i and selected reference camera C_0 can be expressed as follows:

$$\mathbf{R}_{i0} = \mathbf{R}_i \mathbf{R}_0^{-1}, \quad \mathbf{t}_{i0} = \mathbf{t}_i - \mathbf{R}_i \mathbf{R}_0^{-1} \mathbf{t}_0. \tag{2}$$

Next, we rewrite (1) to only consider the orientation and position of the reference camera and the relative orientation and position between cameras:

$$x_i = \mathbf{K}_{fi} \mathbf{\Pi}_0 \begin{bmatrix} \mathbf{R}_{i0}\mathbf{R}_0 & \mathbf{t}_{i0} + \mathbf{R}_{i0}\mathbf{t}_0 \\ 0 & 1 \end{bmatrix} \mathbf{X}_0. \tag{3}$$

Finally, nonlinear optimization of the external camera parameters within the cluster is performed using Levenberg-Marquardt algorithm. The error function is defined as the total reprojection error, i.e., the sum of reprojection errors of all the grid points M as seen by N cameras in P checkerboard positions.

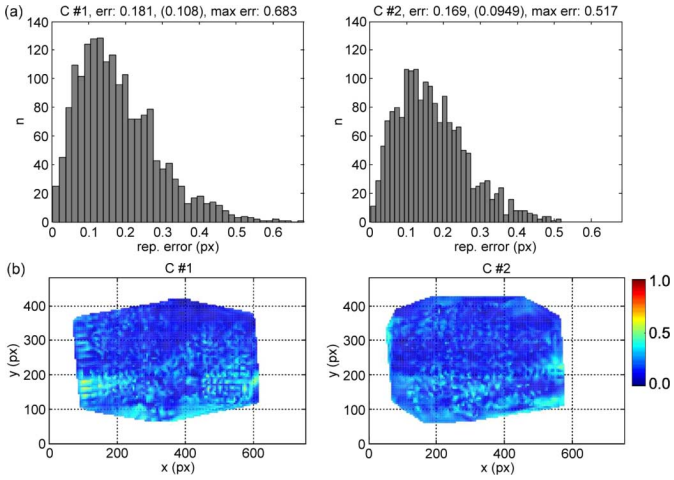

Fig. 3. (a) Distribution of the reprojection error of the checkerboard points for a pair of cameras inside a stereo cluster. (b) Distribution of reprojection error across the image for each camera. The total mean reprojection error was 0.18 (0.11) pixels.

Since the internal parameters of the cameras are independent and have already been optimized, only the external parameters are considered during this optimization. In total, $6 \times (N-1) + 6 \times P$ parameters are optimized.

Usually, one needs to collect about 15–20 images of the checkerboard in various poses to obtain accurate calibration. The typical reprojection error for a pair of cameras is between 0.12 and 0.20 pixels. Fig. 3(a) shows the error distribution for a calibrated stereo pair. Notice that it is approximately a skewed Gaussian distribution. Fig. 3(b) illustrates the reprojection error across each image. Areas with consistently high reprojection error indicate an inadequate model which may suggest deviations in the lens construction or other issues with the camera.

B. External Calibration

Before we externally calibrate the camera clusters, we construct a vision graph to identify the cameras with largest overlap. In contrast to other methods [13]–[15], our approach resolves Euclidean reconstruction (preserving metric information) and introduces novel parameter reduction in the case of two-point bar calibration which increases the robustness of the calibration [16].

Our external calibration algorithm consists of the following steps:

1) image acquisition and sub-pixel marker detection on multiple cameras;
2) composition of adjacency matrix to construct a weighted vision graph that describes interconnections between the cameras (e.g., the number of common points);
3) computation of the fundamental, $\mathbf{F}$, and essential, $\mathbf{E}$, matrices with RANSAC;
4) essential matrix decomposition into rotation and translation parameters defined up to a scale factor, λ;
5) determination of the scale factor, λ, through triangulation and LM optimization;
6) optimal path search using Dijkstra algorithm;

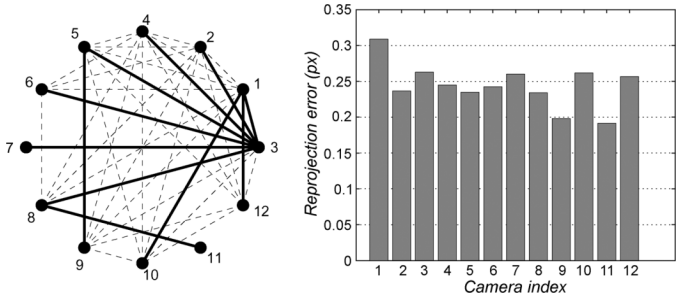

Fig. 4. Vision graph generated for 12 stereo clusters with the cluster #3 selected as the reference cluster (left) and typical reprojection error for the external cluster calibration (right).

7) global optimization of the parameters using sparse bundle adjustment (SBA) [17].

Our external calibration approach assumes that at least any two clusters overlap. Based on captured marker positions, pair wise geometric relationships are established between camera pairs with large numbers of common points. The relationship between the points captured by each camera can be described by the fundamental matrix, $\mathbf{F}$, for the image coordinates, $\mathbf{x}$, and by the essential matrix, $\mathbf{E}$, for the normalized image coordinates, $(\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x})$:

$$\mathbf{x}_{i2}^T \mathbf{F} \mathbf{x}_{i1} = 0 \quad \text{and} \quad \hat{\mathbf{x}}_{i2}^T \mathbf{E} \hat{\mathbf{x}}_{i1} = 0. \tag{4}$$

The fundamental matrix is calculated from the image points using the 8-point algorithm while the outliers are removed by the RANSAC method. From the fundamental matrix, the essential matrix is computed using predetermined internal calibration parameters. The essential matrix can then be decomposed into a rotational matrix $\mathbf{R}$ and a position vector $\mathbf{T}$. Using singular value decomposition, the matrices $\mathbf{T}$ and $\mathbf{R}$ can be obtained [18]. Results obtained from the essential matrix decomposition are further optimized using the LM algorithm. Due to the nature of the essential matrix, the translation between cameras can only be obtained up to a scale factor which can be obtained by knowing the distance between the two LED markers.

To solve for the global calibration, only the transformations for the relevant pairs of cameras are calculated. The pairs are automatically selected from the vision graph which is constructed based on the number of overlapping points between the camera clusters. In the vision graph, weighted connections represent the reciprocal of the number of common points seen by both reference cameras. If there are no common points or the number of points is too small, the connection is removed. Using the shortest path from the reference camera to each camera, we can calculate the absolute position of each camera (Fig. 4). Let i, j, and k be indices of consecutive cameras on the path found in the vision graph. From pairwise calibration, the transformations from i to j and from j to k are denoted as $(\mathbf{R}_{ij}, \mathbf{t}_{ij})$ and $(\mathbf{R}_{jk}, \mathbf{t}_{jk})$. The transformation from i to k can be calculated as follows:

$$\mathbf{t}_{ik} = \mathbf{t}_{ij} + \mathbf{R}_{ij}\mathbf{t}_{jk} \quad \text{and} \quad \mathbf{R}_{ik} = \mathbf{R}_{ij}\mathbf{R}_{jk}. \tag{5}$$

After the initial solution of the relative position and orientation of the cameras are obtained, the results are globally optimized using nonlinear optimization to reduce the reprojection errors.

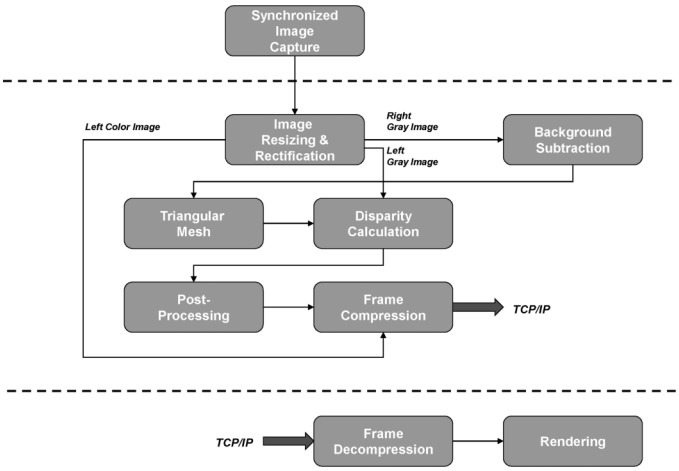

Fig. 5. Block diagram of the stereo algorithm for the 3-D mesh generation.

The sparse nature of the optimization problem (i.e., all cameras cannot see all the points) ensures that the optimal solution can be obtained efficiently using SBA [19].

For external calibration of our 48 camera setup, we use a rigid metal bar with two LED markers attached on each end. We chose Luxeon I LED (Philips Lumileds Lighting Company, San Jose, CA), with a brightness of 30.6 lm and 160° emitting angle. The two LEDs are placed on a metal bar at a distance of 298 mm. The complete external calibration of our general TI apparatus which has 12 stereo clusters with about 3000 3-D points, takes between 10–15 s on an Intel Xeon 3.20-GHz processor with 1 GB of memory. The mean reprojection error between all the cameras is typically between 0.25 and 0.40 pixels with the standard deviation between 0.04 and 0.12 pixels as illustrated in Fig. 4. In our setup, these errors result in the cluster position errors of about 0.5% and orientation errors of about 0.1°.

C. Calibration of Physical Space

Once the calibration of the cameras is performed, the reference camera cluster needs to be aligned with the display to achieve correct scale for each user. We propose a simple method for calibrating the reference cluster (i.e., camera space) to the physical space of each TI station. The calibration of the physical space is performed by acquiring one image of the checkerboard placed in the vertical position in front of the reference camera and aligned with the display. The checkerboard can also be equipped with calibration markers if a tracking system needs to be aligned with the 3-D data (e.g., for tracking input devices or stereo glasses). Since the reference camera calibration is known, it is possible to determine the exact position and orientation from one checkerboard image.

V. 3-D Reconstruction

In this section, we explain how we construct a 3-D model of an object using a calibrated camera cluster. Fig. 5 illustrates the algorithmic pipeline for each stereo cluster. Our focus, in this section, are the algorithms drawn between the dotted lines. We assume some familiarity with 3-D reconstruction. A more formal introduction to this topic can be found in [12] and [20].

Determining the 3-D coordinates of a point using a calibrated stereo pair is in fact equivalent to matching the projection of the point in each of the image planes. To simplify the matching procedure, the image is first transformed via a process known as **rectification**, which reduces the problem into a linear search problem along vertically aligned images. More concretely, given rectified left and right gray scale images, $I_L, I_R : \mathbb{R}^2 \to \mathbb{R}$, our objective is to find the **disparity map**, $d : \mathbb{R}^2 \to \mathbb{R}$, such that

$$I_R(x, y) = I_L\left(x + d(x, y), y\right). \tag{6}$$

For simplicity, we model the domain and range of each image as the continuum, but the methods presented in this section generalize to a discrete space in a straightforward manner. The disparity function determines the 3-D position of every point that is simultaneously observable by both cameras in a stereo pair.

Traditional methods to determine this match either employ a local approach like normalized cross correlation or global optimization techniques [21]. The normalized cross correlation techniques rely on a fixed window to perform matching, which assumes unreasonably that all depths within a window are identical. This assumption ensures that the algorithm is able to work quickly, but unfortunately produces extremely inaccurate results. Global optimization techniques begin by associating a cost to each pixel's disparity that depends not only on how well it matches to a pixel in the other image, but also how well this disparity matches to neighboring pixels' disparity. This cost function is then minimized in order to determine the disparity of each pixel. Though these optimization techniques produce more accurate results than the entirely local approaches, they are computationally expensive.

In this paper, we employ a hybrid approach to resolve the problem. We construct a triangular mesh over the set of visible points on the domain of the right image, and then construct a disparity map using a version of normalized cross correlation over the nodes of the mesh. Importantly, the size of each triangle dictates an appropriate window size over which to perform matching while simultaneously reducing the number of points to perform matching over. Once the disparity is determined at these points, interpolation can be employed to determine depth in between nodes. Several post-processing techniques exist that act similarly to global optimization by enforcing some regularity on the disparity map (e.g., smoothness). In our case, we employ anisotropic diffusion which when defined on our mesh converges quickly [22].

We continue our discussion by finding a graphical model of an image which yields an appropriate triangular mesh upon which we can define neighborhood-based operators. Then, we specify how a regularized disparity is computed using this model. Finally, we analyze the accuracy of our results and the efficiency of our meshing approach in compressing data.

A. Image Model

We begin by defining the amount of variation on the gray scale values around a point on the domain. We want the size of a triangle in the mesh to dictate the scale at which to perform

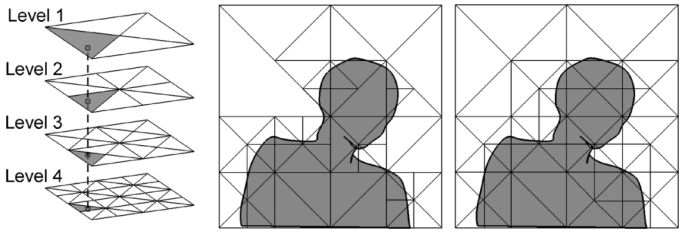

Fig. 6. Triangular meshing of an image domain: A hierarchy of meshes is used to define the neighborhoods $\{S_{x,n}\}$ around any point x (left). A collection of sets is highlighted for a point in the domain. A non-conforming meshing of the domain obtained by selecting the neighborhoods specified by the index of support $\eta(x,\tau)$ for a given threshold, τ, which corresponds to the desired amount of variation (middle). A conforming meshing obtained after refining non-conforming triangles (right).

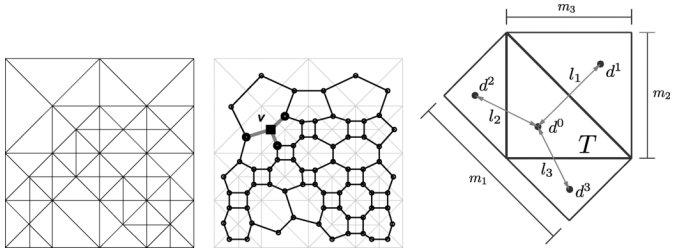

Fig. 7. Graphical representation of the mesh: A conforming triangular mesh (left). The graph constructed by representing each triangle by a point and drawing edges between triangles that share an edge (middle). Discretization of the diffusion operator on the triangulated domain (right).

matching (i.e., every point in a triangular region should be at approximately the same depth); therefore, an ideal mesh would give more detail to regions with large values of variation since these areas generally correspond to places with depth discontinuities.

For every point x in $\mathbb{R}^2$, let $\{S_{x,n}\}_{n=1}^{N}$ denote a decreasing collection of sets, where $N \in \mathbb{N}$ is some fixed number and $S_{x,n_1} \subset S_{x,n_2}$ if $n_1 > n_2$. The variation around a point x at level n is then defined as

$$V(x,n) = \frac{1}{|S_{x,n}|} \int_{S_{x,n}} \left| I(y) - \bar{I}_{S_{x,n}} \right|^2 dy \tag{7}$$

where $\bar{I}_{S_{x,n}}$ is the mean intensity value over $S_{x,n}$. Let the **index of support** be the function $\eta : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{N}$ defined as

$$\eta(x,\tau) = \min \{n | V(x,n) < \tau\} \tag{8}$$

where τ is a user-specified threshold which identifies the amount of variation allowed in a fixed region within an image. The index of support η defines the neighborhood around x at which we first find the desired amount of variation.

Before generating the triangular mesh of the image domain, a coarse mesh of right isosceles triangles is generated at level $n = 1$ as illustrated in Fig. 6. This coarse mesh is then refined by bisecting each triangle. The refinement procedure in a particular region is halted when the variation in each triangle is less than a user-specified threshold τ. The index of support, $\eta(x,n)$, is then equivalent to choosing the coarsest level triangular mesh that covers the domain while satisfying the user-defined threshold.

As we discussed earlier, our goal is to mimic the effects of a global optimization procedure by refining our initial local disparity estimates via anisotropic diffusion. To define such a post-processing step, our mesh must be able to share information between neighboring nodes which requires that our mesh have no nodes that are on the middle of another triangle's edge. A mesh satisfying such a property is referred to as a **conforming** mesh [23]. A conforming triangular mesh is constructed by following Algorithm 1. Importantly, the steps required to construct the mesh using the Bisection Algorithm can be stored as a list corresponding to the triangles at which a bisection took place, which as we discuss in Section V-E gives an efficient way to compress the mesh. Fig. 6 illustrates the outcome of the algorithm.

A graphical representation, $\mathcal{G}$, of the image is then constructed by letting each triangle in the mesh correspond to a vertex and letting edges in the graph correspond to triangles that share an edge. Fig. 7 illustrates the construction of such a graph. Notice that by requiring the mesh to be conforming, each node in the graph (except those sitting at the edge of the image) has exactly three edges connecting to it.

Algorithm 1: Bisection Algorithm

1: Initialize the procedure with a coarse triangulation of the image domain.

2: **for** each triangle i in the mesh **do**

3: Let n correspond to the level to which i belongs.

4: **if** the variation in i is greater than τ **then**

5: Bisect i.

6: Let j be the neighboring triangle to i opposite to its right angle and suppose j belongs to level n'.

7: **if** $n' \geq n$ **then**

8: Repeat steps 5 through 9 using j.

9: **end if**

10: **end if**

11: **end for**

B. Disparity Computation

Since each vertex of the graph, v, uniquely corresponds to an image coordinate, (x,y), in the right image, I_R, in order to determine a depth, it is sufficient to construct a disparity map, $d : \mathcal{G} \to \mathbb{R}$. We begin by defining a matching score, $\mathcal{C} : \mathcal{G} \times \mathbb{R} \to \mathbb{R}$. At a point $(v,r) \in \mathcal{G} \times \mathbb{R}$, with v corresponding to a triangle, i, the matching score is defined as the average normalized cross correlation between the reference window centered at each of the corners of i in the right image and a window

centered at the same coordinate as each of the corners of i in the left image after a horizontal translation by r. Importantly the size of the neighborhoods used during the normalized cross correlation step are dictated by the level of the triangle i (i.e., a small triangle employs a small correlation window and a large triangle uses a large correlation window).

We then define an approximation to the disparity, d_0, by selecting for each vertex $v \in \mathcal{G}$ the displacement value that gives the greatest correlation score. That is

$$d_0(v) = \arg\sup_{r \in \mathbb{R}} \mathcal{C}(v, r). \tag{9}$$

Let the set of vertices $V_c = \{v | d_0(v) < \hat{\tau}\}$ be the set of points for which we do not have a confident match, i.e., its correlation score is smaller than some user-specified threshold $\hat{\tau}$. These are the vertices that are processed during the anisotropic diffusion step described in the next section.

C. Anistropic Diffusion

At this point, we have an initial estimate of the disparity, d_0, but in order to approximate a global method, we enforce a regularity condition on the disparity values of V_c. In practice, regularity conditions generally take the form

$$E(d) = \frac{1}{2} \int_{\mathbb{R}^2} \alpha(x, y) |\nabla d|^2 dx dy \tag{10}$$

where d is the disparity we are attempting to compute, $\alpha(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is a weighting function known as the **diffusivity function**, and ∇ denotes the gradient operation. This diffusivity function penalizes large variations of d. We can impose additional constraints on this function in order to ensure that it behaves appropriately given our task such as large penalties at homogeneous regions in the image or small penalties at inhomogeneous regions in the image. The first requirement describes our expectation that homogeneous regions of the image should have approximately constant disparity, whereas the second requirement allows for high variability in the disparity at or near edges. The diffusivity function therefore depends on the image itself.

In order to compute the disparity, (10) must be minimized, but this process is generally cumbersome. A typical approach to minimizing the energy functional in (10), using the Euler-Lagrange equations, leads to a diffusion process of the form

$$d_t = \nabla \cdot (\alpha(x, y) \nabla d) \tag{11}$$

where $\nabla \cdot$ denotes the divergence operation and t is the evolution parameter [24]. To compute the disparity, this diffusion process is initialized with some estimate and then allowed to evolve according to (11) until it converges, which in practice can take quite some time. Fortunately, using our triangular mesh, this diffusion process takes a more computable form that quickly converges in practice. To arrive at this computable form, we begin by integrating the diffusion equation in a triangular region, T, and then apply Gauss's Theorem:

$$\begin{aligned} \int_T d_t(x, y, t) dx dy &= \int_T \nabla \cdot (\alpha(x, y) \nabla d(x, y, t)) \, dx dy \\ &= \int_{\partial T} \alpha(x, y) \nabla d(x, y, t) \cdot \hat{n} ds \\ &= \sum_{i=1}^{3} \int_{S_i} \alpha(x, y) \nabla d(x, y, t) \cdot \hat{n} ds \end{aligned}$$

where ∂T is the boundary of the triangular region considered, S_i are the sides of the triangle, and $\hat{n}$ is the normal to each of these sides.

Using the notation introduced in Fig. 7, we discretize the diffusion equation. As noted above, every triangle has exactly three neighbors. Let d^0 be the disparity value to be updated and d^i's the disparity values of the neighboring vertices in $\mathcal{G}$, then

$$\frac{(d_{n+1}^0 - d_n^0)}{\Delta t} A_T = \sum_{i=1}^{3} \alpha_i m_i \frac{(d_n^i - d_n^0)}{l_i} \tag{12}$$

where A_T is the area of the triangle T, Δt is the time step, α_i is $\alpha(x, y)$ evaluated at the side S_i, m_i is the length of the hypotenuse of neighboring triangle i, and l_i is the distance between the d^i and d^0. In order to simplify this equation, we first assume that m_i/d_i is constant. This is a good approximation in practice. We then combine $\Delta t(m_i/l_i)$ into a single constant, c, and our discrete update equation that performs anisotropic diffusion becomes

$$d_{n+1}^0 = \left(1 - c \sum_{i=1}^{3} \frac{\alpha_i}{A_T}\right) d_n^0 + c \sum_{i=1}^{3} \frac{\alpha_i}{A_T} d_n^i. \tag{13}$$

At this point, we substitute for α_i given our requirements. Let I^0 denote the intensity of the base vertex and I^i denote the intensity of the neighboring vertices in $\mathcal{G}$. We then set $\alpha_i = A_T g(|I^0 - I^i|)$, where $g : \mathbb{R} \rightarrow \mathbb{R}$ is a monotone decreasing function. Observe that this choice satisfies our required properties. Namely, due to our choice of representation, large size triangular regions correspond to homogeneous regions. α_i is evaluated at the edge between triangles wherein there could be a dramatic change in the intensity (i.e., an inhomogeneous region). Therefore, we measure the homogeneity of this edge by comparing the intensity of neighboring vertices. Combining these two terms measuring the homogeneity, we arrive at our diffusivity function. With this substitution, the discrete update equation that performs anistropic diffusion becomes

$$d_{n+1}^0 = \left(1 - c \sum_{i=1}^{3} g\left(|I^0 - I^i|\right)\right) d_n^0 + c \sum_{i=1}^{3} g\left(|I^0 - I^i|\right) d_n^i. \tag{14}$$

We initialize this discrete update equation using our initial estimate for the disparity, d_0. We obtain d from d_1 by applying N_d

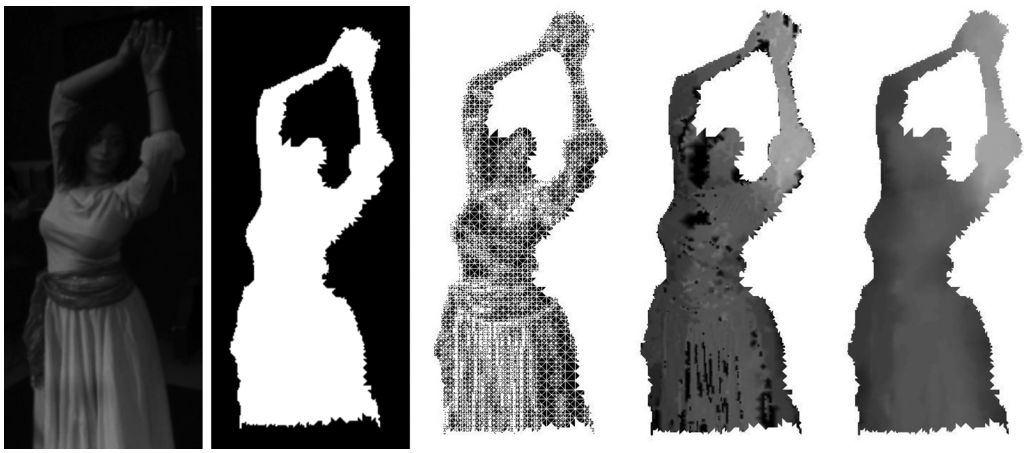

Fig. 8. A 320 × 240 image taken from a single camera in a stereo cluster (left), a background subtracted image generated for that image (second from the left), the mesh generated for this image (middle), the pre-processed disparity image (second from the right), and the post-processed disparity image (right). Note that lighter gray values indicate that the object in the scene is closer, darker gray indicate that an object in the scene is further away, and black indicates areas of uncertainty.

TABLE I
AVERAGE FRAME RATE FOR A TYPICAL IMAGE SEQUENCE IN THE TI SYSTEM ON TWO DUAL CORE 2.33-GHZ MACHINES OBTAINED USING TI STEREO PAIRS EACH WITH SIZE 320 × 240 AND 640 × 480 WITH APPROXIMATELY 10 000 TRIANGLES PER FRAME

	320 × 240	640 × 480
Triangulation	3.83 ms	15.59 ms
Disparity	15 ms	22.84 ms
Post-Processing	1.78 ms	3.61 ms
Decompression	0.78 ms	4.98 ms
Total	21.39 ms	47.02 ms

steps of anisotropic diffusion. In practice, after ten steps, this procedure converges. We can then obtain a disparity map over the entire image domain by interpolating the values obtained for each of the triangular regions.

Fig. 8 illustrates the output of our algorithm at various steps in the reconstruction pipeline for a typical 320 × 240 image from our portable TI apparatus. Table I describes the average time required to perform each step of our algorithm using an image sequence similar to the one presented in Fig. 8.[1]

D. Analysis of Disparity

At this point, we can compare the effectiveness of our algorithm in calculating disparities on the traditional aforementioned benchmarks. The benchmarks consist of dozens of pictures. The two images that the benchmark has identified as the most difficult are found in the left column of Fig. 9. The accuracy of the measurements is calculated against a ground truth image, which can be found in the center column of Fig. 9, generated by a laser range finder. Error, in this domain, is generally calculated by the percentage of pixels that differ in their returned disparity from the ground truth by more than one. Put more clearly, this is approximately the number of pixels that differ in their returned value by more than an order of magnitude greater than a single centimeter. The output of our stereo algorithm on the images found in the left column of Fig. 9 calculated on two dual core 2.33-GHz machines can be found in the right column of the same figure. A quantitative comparison of our algorithm can be found in Table II. We include the most accurate performers on this benchmark in the same table. Wang *et al.* employed a dual core 1.6-GHz machine [26], Bleyer *et al.* employed a 2-GHz Pentium 4 machine [27], and Klaus *et al.* employed a dual core 2.21-GHz machine [28].

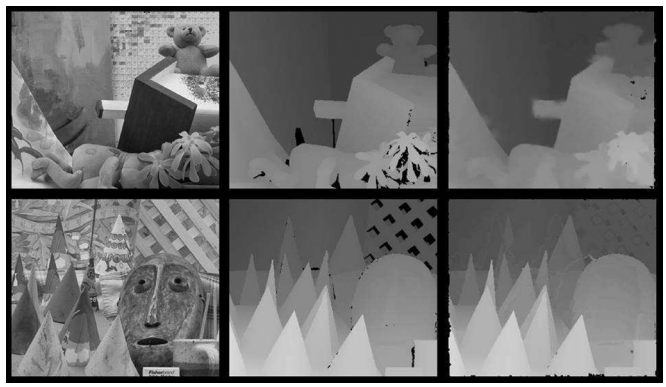

Fig. 9. Two images, each of size 450 × 375, from the benchmark developed by Scharstein *et al.* [25] (left column), the ground truth for these two images produced using a laser range finder (center column), and the output of our stereo algorithm (right column). Note that lighter gray values indicate that the object in the scene is closer, darker gray indicate that an object in the scene is further away, and black indicates areas of uncertainty.

E. Communication of Reconstruction

Sharing the models between different stations requires transmitting the triangulation models in real-time. A standard format for transmitting triangulation information consists of transmitting nodal values, and then specifying the triangles based on the node indices. In particular, if we consider RGB values (3 bytes) and depth values (2 bytes) per pixel, we have 5 bytes per node. Each triangle must specify 3 vertices and each vertex requires at least 3 bytes (since there are more than 2^{16} possible nodes in our representation). Hence, there are at least 9 bytes per triangle.

Given our choice of representation, we can do better. Since our triangulation results from a bisection scheme, it is possible

[1]The video icme_project.avi included in the supplementary material is an illustration of the local reconstruction of a user from a single stereo camera.

TABLE II
QUANTITATIVE COMPARISON OF OUR ALGORITHM AGAINST THE TOP PERFORMERS ON THE BENCHMARK DEVELOPED BY SCHARSTEIN ET AL. [25]. THE TEDDY AND CONE IMAGE CORRESPOND TO THE TOP AND BOTTOM ROWS OF FIG. 9, RESPECTIVELY. OUR OUTPUT WAS PRODUCED WITH APPROXIMATELY 40 000 TRIANGLES IN BOTH INSTANCES

Process	Our's	Wang	Bleyer	Klaus
Teddy 1-Pixel Error	8.15%	8.31%	6.54%	7.06%
Teddy Speed	42.1ms	20s	100s	14s
Cone 1-Pixel Error	8.56%	7.18%	8.62%	7.92%
Cone Speed	53.8ms	20s	100s	25s

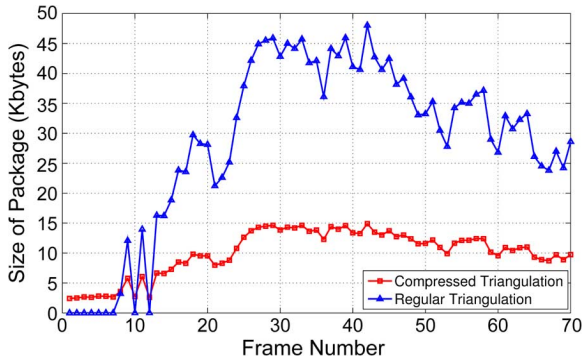

Fig. 10. Comparison of package size between the standard and our encoded format.

to specify how the representation was generated instead of specifying each node in the triangle. That is, we can specify which triangles are bisected. This encoding scheme for the triangulation yields large gains given enough triangles in the representation. No additional time is required to encode the representation since the information for encoding the representation is generated at the same time the representation is initially computed.

In Fig. 10, we compare the size of the data package generated over a typical image sequence after background subtraction from the TI system (each image has size 320×240). Typical images in the sequence look similar to the left images in Fig. 8. In the sequence used for Fig. 10, no object is present in the field of view of the camera for the first seven frames during which time the compressed package is larger than the standard package, but from then on, the compressed package is close to three times smaller than the standard package. Notice, each frame is less than 15 Kbytes.

VI. 3-D VISUALIZATION

The TI renderer is responsible for providing a collaborative, immersive, virtual environment by integrating the 3-D video streams coming from the local and remote TI stations. The renderer allows users to freely reposition and reorient reconstructed 3-D objects within the environment and record and later play back with full control incoming streams. The immersive environment and the 3-D video components each present a set of challenges to the real-time performance expected of the renderer. In this section, we highlight these challenges and how we address them.

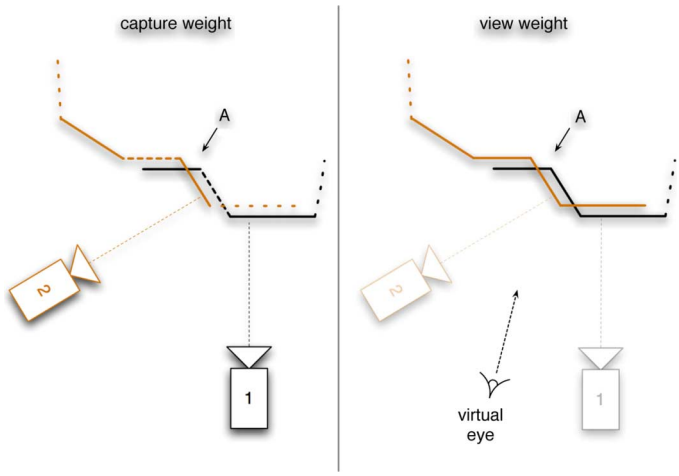

Fig. 11. Contribution of each triangle is determined by: (Left) how well the corresponding capturing camera is able to resolve that triangle; (Right) how much that triangle is likely to contribute to the final image given the current view parameters. Filled lines represent important triangles versus unimportant ones which are depicted as dashed lines.

Fig. 12. Illustration of the difference between naïvely projecting facades (left) and after compositing different facades using the weight factors (right).

Immersive environments must satisfy a strict set of requirements to ensure usability [29]. For example, a steady refresh rate of 60 frames per second for virtual reality displays is required to prevent discomfort or motion sickness. Given this fact, the TI renderer must satisfy the following requirements: first, it should optimize the necessary processing for speed, fully exploiting all available computational resources; and second, its design must separate the rendering from 3-D video processing to ensure that the visual refresh is not stalled.

We optimize the stream management by exploiting two levels of parallelism: task parallelism and data parallelism. At the task level, we distinguish between the rendering thread and the 3-D video processing thread. The two task sets communicate via triple-buffered storage which receives reconstructed video data produced by the processing thread and then serves it to the rendering thread. The buffered storage guarantees that complete data is always available for the display independent of erratic video processing behavior. Conversely, the video processing is able to produce representations unhampered by the rendering thread's lock on data. The 3-D video stream bundles the data,

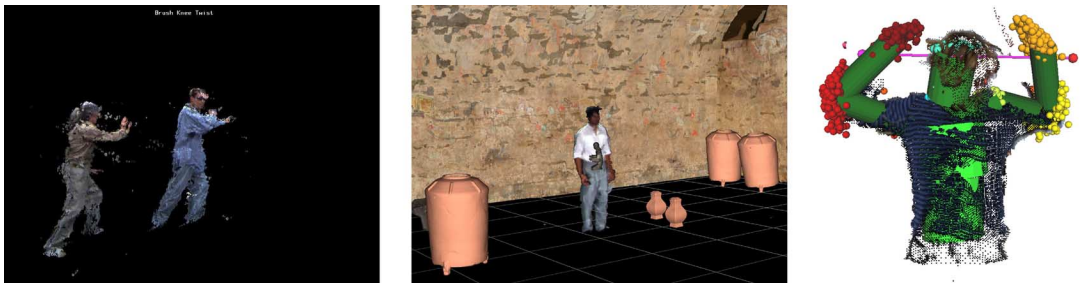

Fig. 13. Illustration of the various applications of the TI system. An example of remote learning of physical actions wherein Tai Chi is being taught in 3-D (left). An example of cyber-archeology (middle). An illustration of the tracking (drawn in green) that can be constructed using particle filtering (the yellow and red dots) in tandem with our representation (right).

called *facades*, coming from each of the individual clusters of a TI system. The video processing task is split further into multiple concurrent tasks for each facade. The CPU cores are used to uncompress the data, but are not used to produce the final virtual space 3-D projection of the facades. Instead, the transformation from the 3-D reconstruction done at each capturing cluster to the calibrated cross-cluster 3-D space representation is most suited for the data parallel processing step.

If we want a high-quality rendition that is visually consistent over the duration of playback, relying on a naïve approach (i.e., projecting the different facades into the cross-cluster 3-D space without enforcing some type of consistently between different facades) would not produce a well-defined surface due to the noise in the reconstruction of each facade and the color mismatch between different facades since the capturing clusters are not photometrically calibrated. Our approach to the visual representation addresses these issues by 1) leveraging the built-in functionality of the graphics hardware to rapidly project the triangles and compute screen coverage in real-time and 2) compositing the individual contributions of all reconstructions at each screen pixel to obtain better quality visualization. In the first phase of the rendering process, the individual contributions for each reconstruction are contributed relative to the current view. For this purpose, each reconstruction is transformed and rasterized based on the current view into an off-screen buffer using a custom fragment program. For each covered fragment, we generate the usual color and depth contribution and compute and store weight factors used during the compositing of multiple views. As illustrated in Fig. 11, we determine the weight factor using two pieces: the capture and view weight.

To determine the view weight, we consider the orientation of each triangle in a reconstruction with respect to the capturing direction of the corresponding camera cluster. Triangles directly facing the capturing cluster receive a high weight, and as the triangles face away from the camera, the weight drops towards zero at the orthogonal orientation. This first weight relies on a TI system using many clusters to cover the environment which produces many overlapping regions, where better resolved triangles in the overlap should naturally be favored. Similar to the capturing weight, to determine the view weight we consider the current viewing parameters to determine interesting parts of the reconstruction with respect to the current view. Again the dot product is taken, but here it is calculated between the normal of each triangle and the viewing direction. This second weight, thus, favors the facades from camera clusters that are aligned with the current view.

The two weights are multiplied together to form the final weight of the contribution. In the second phase, the contributions at each pixel of the screen are composited independently. Care has to be taken not to blend all the stored contributions to retain proper occlusion, e.g., a hand in front of a torso. Our current implementation determines the closest contribution and blends it with those within a user-specified depth range. Fig. 12 illustrates the difference between using a naïve approach wherein everything is projected without blending and our approach.[2]

VII. Applications

In the past several years, we have demonstrated the utility of the TI system via several interdisciplinary applications as illustrated in Fig. 13. In this section, we briefly review several of these applications.

One of the first applications of the TI framework presented in this paper was during a distributed dance performance across the continent (between Berkeley, CA and Urbana-Champaign, IL) in a shared virtual environment. The TI system provided a bi-directional connection between two locations, allowing the dancers to see each other side-by-side in the same virtual space. The dancers had to accommodate to the networking limitations of the system (delays ranged between 500 to 1500 ms).[3] The TI system also introduced the novel concept of "virtual touch" wherein tactile and haptic feedback rely on visual information from the rendering [30].

In collaboration with Stanford University, we have examined the utility of immersive environments when compared with 2-D video for learning [31]. To do this, we captured a Tai Chi teacher in our TI system performing several moves and then reprojected this information into a virtual space alongside the real-time 3-D data of a student. The student was then able to see the teacher's and his or her own data in real-time side by side as in a virtual mirror. Two sets of experiments were then performed with 40 participants comparing the immersive and video learning. The result of the study conclusively demonstrated according to a self-report and an objective performance measure that users learned much better in the TI system rather than via video.

[2]The video J360.mov included in the supplementary material is an illustration of the local reconstruction of a user for a single frame from multiple views using our blending scheme.

[3]The video DanceLab.mov included in the supplementary material is an illustration of two dancers collaborating together over the network using the portable TI system.

Similar collaborative ideas have been applied to the application of cyber-archaeology for remote real-time interaction with 3-D archaeological models in a shared virtual environment. The framework addresses key issues in modern archeology, the reversibility of the excavation process, and the accessibility of data during the interpretation process [32]. In collaboration with UC Merced, we are developing a virtual participatory platform where 3-D models, photos, movies, maps, and other spatial data are presented to geographically distributed users in a virtual environment. Each user navigates and interacts from a first person perspective while users from remote locations also participate using their own TI station.

The real-time TI system can also be used to perform markerless motion capture to provide 3-D data for full-body tracking using a particle filter [33]. Importantly we show that by applying this method to our representation of 3-D space rather than a point wise representation of the 3-D space, the resulting full-body tracking is far more accurate and faster. Though this has obvious applications to user interface design, we briefly describe how we have applied a simplified real-time data analysis technique to this tracking data to perform tele-rehabilitation between a "therapist" and a "patient" in a virtual environment. The patient was required to track the movement of the therapist who was performing a stepping-in-place task. Hip angles were extracted in real-time from the data and projected on the screen to augment the visual feedback for the users [34].

VIII. Conclusion

In this paper, we described a multi-camera system that creates a highly accurate, 3-D reconstruction of an environment in real-time. We began by providing an overall systems perspective and then described in detail a calibration, representation, reconstruction, and then visualization methodology that together provide a state-of-the-art TI system. The hierarchical calibration approach allows for robust and flexible calibration of multiple cameras with various pairwise overlap. The representation and reconstruction of 3-D data is simultaneously flexible and accurate which allows for high levels of compressibility and easy visualization. The reconstruction is amongst the top performers on an industry wide benchmark for accuracy and it is easily one of the fastest reconstructions available. The visualization technique employs the nature of the representation and the viewing direction to build high-quality depiction that is visually consistent. The technique works rapidly by taking advantage of standard graphics hardware.

Future work will focus on employing the reconstruction to build a single unified model of the 3-D environment from the various views as opposed to the current strategy of simply relying on the visualization technique to correct for possible inconsistencies amongst the various views. This would have the added benefit of reducing the final size for the entire 3-D environment, which would have the benefit of reducing the overall network bandwidth required for a particular TI station. Full-body 3-D reconstruction of users in real-time offers new possibilities for immersive and TI applications. The users can be immersed inside computer generated existing or non-existing environments, such as ancient buildings and future architectural designs to allow interactive exploration. The 3-D capturing framework presented can also provide data for human motion analysis and modeling. Extracted kinematic parameters could be applied as online feedback to a user for training of physical movements (e.g., dancing, physical therapy, and exercise).

References

[1] D. Nguyen and J. Canny, "Multiview: Spatially faithful group video conferencing," in *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems*, New York, 2005, pp. 799–808.

[2] T. DeFanti, D. Sandin, M. Brown, D. Pape, J. Anstey, M. Bogucki, G. Dawe, A. Johnson, and T. S. Huang, "Technologies for virtual reality/tele-immersion applications: Issues of research in image display and global networking," in *Proc. EC/NSF Workshop Research Frontiers in Virtual Environments and Human-Centered Computing*, Jun. 1–4, 1999.

[3] T. Kanade, P. Rander, S. Vedula, and H. Saito, "Virtualized reality: Digitizing a 3D time varying event as is and in real time," in *Mixed Reality, Merging Real and Virtual Worlds*. New York: Springer-Verlag, 1999, pp. 41–57.

[4] J. Mulligan and K. Daniilidis, "Real time trinocular stereo for tele-immersion," in *Proc. 2001 Int. Conf. Image Processing*, Thessaloniki, Greece, 2001, pp. 959–962.

[5] S. Jung and R. Bajcsy, "A framework for constructing real-time immersive environments for training physical activities," *J. Multimedia*, vol. 1, no. 7, pp. 9–17, 2006.

[6] J. Hasenfratz, M. Lapierre, and F. Sillion, "A real-time system for full-body interaction with virtual worlds," in *Proc. Eurographics Symp. Virtual Environments*, The Eurographics Association, 2004, pp. 147–156.

[7] O. Schreer, I. Feldmann, N. Atzpadin, P. Eisert, P. Kauff, and H. Belt, "3dpresence—A system concept for multi-user and multi-party immersive 3d videoconferencing," in *Proc. 5th Eur. Conf. Visual Media Production (CVMP 2008)*, Nov. 2008, pp. 1–8.

[8] H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. Gross, W. Culbertson, and T. Malzenbender, "The coliseum immersive teleconferencing system," in *Proc. Int. Workshop Immersive Telepresence*, Juan-les-Pins, France, 2002.

[9] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. V. Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt, "blue-c: A spatially immersive display and 3d video portal for telepresence," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 819–827, 2003.

[10] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1, pp. 7–42, 2002.

[11] D. Zhang, Y. Nomura, and S. Fujii, "Error analysis and optimization of camera calibration," in *Proc. IEEE/RSJ Int. Workshop Intelligent Robots and Systems (IROS 91)*, Osaka, Japan, 1991, pp. 292–296.

[12] Y. Ma, S. Soatto, J. Kosecka, Y. Ma, S. Soatta, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision*. New York: Springer, 2004.

[13] X. Cheng, J. Davis, and P. Slusallek, "Wide area camera calibration using virtual calibration objects," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2000)*, 2000.

[14] I. Ihrke, L. Ahrenberg, and M. M. Magnor, "External camera calibration for synchronized multi-video systems," in *Proc. 12th Int. Conf. Computer Graphics, Visualization and Computer Vision 2004*, Plzen, Czech Republic, Feb. 2004, vol. 12, pp. 537–544.

[15] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multicamera self-calibration for virtual environments," *Presence*, vol. 14, no. 4, pp. 407–422, 2005.

[16] G. Kurillo, Z. Li, and R. Bajcsy, "Wide-area external multi-camera calibration using vision graphs and virtual calibration object," in *Proc. 2nd ACM/IEEE Int. Conf. Distributed Smart Cameras (ICDSC 08)*, Stanford, CA, Sep. 7–11, 2008, IEEE.

[17] M. Lourakis, Levmar: Levenberg-Marquardt Nonlinear Least Squares Algorithms in C/C++, Jul. 2004. [Online]. Available: http://www.ics.forth.gr/~lourakis/levmar.

[18] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York: Cambridge Univ. Press, 2004.

[19] M. Lourakis and A. Argyros, The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm, Institute of Computer Science—FORTH, Heraklion, Crete, Greece, Tech. Rep. 340, Aug. 2004. [Online]. Available: http://www.ics.forth.gr/~lourakis/sba.

[20] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Englewood Cliffs, NJ: Prentice-Hall Professional Technical Reference, 2002.

[21] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2006, pp. 519–528.

[22] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1990.

[23] J. Maubach, "Local bisection refinement for N-simplicial grids generated by reflection," *SIAM J. Sci. Comput.*, vol. 16, p. 210, 1995.
[24] T. Chan and J. Shen, *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*. Philadelphia, PA: SIAM, 2005.
[25] D. Scharstein, R. Szeliski, and M. Coll, "High-accuracy stereo depth maps using structured light," in *Proc. 2003 IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, 2003, vol. 1.
[26] Z. Wang and Z. Zheng, "A region based stereo matching algorithm using cooperative optimization," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition 2008*, Anchorage, AK.
[27] M. Bleyer and M. Gelautz, "A layered stereo matching algorithm using image segmentation and global visibility constraints," *ISPRS J. Photogram. Remote Sens.*, vol. 59, no. 3, pp. 128–150, 2005.
[28] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proc. Int. Conf. Pattern Recognition*, 2006, vol. 2.
[29] G. Kurillo, R. Bajcsy, K. Nahrstedt, and O. Kreylos, "Immersive 3d environment for remote collaboration and training of physical activities," in *Proc. IEEE Virtual Reality Conf. (VR 2008)*, Reno, NV, Mar. 8–12, 2008, pp. 269–270.
[30] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy, "Enabling multi-party 3d tele-immersive environments with viewcast," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMCCAP)*, accepted for publication.
[31] J. N. Bailenson, K. Patel, A. Nielsen, R. Bajcsy, S. Jung, and G. Kurillo, "The effect of interactivity on learning physical actions in virtual reality," *Media Psychol.*, to be published.
[32] M. Forte and G. Kurillo, "Cyberarchaeology—Experimenting with teleimmersive archaeology," in *Proc. 16th Int. Conf. Virtual Systems and Multimedia (VSMM 2010)*, Seoul, Korea, Oct. 20–23, 2010.
[33] S. Hauberg, S. Sommer, and K. S. Pedersen, "Gaussian-like spatial priors for articulated tracking," in *Proc. ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., 2010, pp. 425–437, Springer-Verlag, Berlin, Germany.
[34] G. Kurillo, T. Koritnik, T. Bajd, and R. Bajcsy, "Real-time 3d avatars for tele-rehabilitation in virtual reality," in *Proc. 18th Medicine Meets Virtual Reality (MMVR) Conf.*, Newport Beach, CA, Feb. 2011, pp. 290–296.

Ramanarayan Vasudevan (S'10) received the B.S. degree in electrical engineering and computer sciences and an honors degree in physics from the University of California, Berkeley, in 2006 and the M.S. degree in electrical engineering from the University of California, Berkeley, in 2009.

His research interests include sensor networks, computer vision, hybrid systems, and optimal control. He is the recipient of the 2002 Regent and Chancellor's Scholarship.

Gregorij Kurillo received the B.Sc. and Ph.D. degrees from School of Electrical Engineering, University of Ljubljana, Ljubljana, Slovenia, in 2001 and 2006, respectively.

He received the highest national award for his undergraduate thesis work. He was a Research Assistant with the Laboratory of Robotics and Biomedical Engineering at the same institution from 2002 to 2006. He was a Postdoctoral Researcher at University of California (UC), Berkeley, from 2006–2009. Since 2009, he has been assigned to the Research Engineer position to manage work on the Teleimmersion project at UC Berkeley. His research interests include camera calibration, stereo vision, image processing, robotics, and collaborative virtual reality.

Edgar Lobaton (M'09) received the B.S. degrees in mathematics and electrical engineering from Seattle University, Seattle, WA, in 2004 and the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 2009.

He is currently a Post-Doctoral Researcher at the Department of Computer Science at the University of North Carolina at Chapel Hill. He was previously engaged in research at Alcatel-Lucent Bell Labs in 2005 and 2009. His research interests include sensor networks, computer vision, tele-immersion, and motion planning. He is the recipient of the 2009 Computer Innovation Fellows post-doctoral fellowship award, the 2004 Bell Labs Graduate Research Fellowship, and the 2003 Barry M. Goldwater Scholarship.

Tony Bernardin received the B.S. degree in computer science from the Universität Karlsruhe (TH), Karlsruhe, Germany, and the Ph.D. degree in computer science from the University of California, Davis, in 2009 under the supervision of B. Hamann.

His primary research interests are visualization, computer graphics, and virtual reality, with a focus on immersive visualization applications in the computational and earth sciences.

Oliver Kreylos received the B.S. and M.S. degrees in computer science from the Universität Karlsruhe (TH), Karlsruhe, Germany, and the M.S. degree in computer science from the University of California (UC), Davis. He received the Ph.D. degree in computer science from the UC Davis, in 2003 under the supervision of B. Hamann.

He is an assistant research scientist with the UC Davis W.M. Keck Center for Active Visualization in the Earth Sciences (KeckCAVES), and the UC Davis Institute for Data Analysis and Visualization (IDAV). His primary research interests are visualization, computer graphics, and virtual reality, with a focus on immersive visualization applications in the computational and earth sciences.

Ruzena Bajcsy (LF'08) received the Master's and Ph.D. degrees in electrical engineering from Slovak Technical University, Bratislava, Slovak Republic, in 1957 and 1967, respectively, and the Ph.D. in computer science from Stanford University, Stanford, CA, in 1972.

She is a Professor of Electrical Engineering and Computer Sciences at the University of California, Berkeley, and Director Emeritus of the Center for Information Technology Research in the Interest of Science (CITRIS). Prior to joining Berkeley, she headed the Computer and Information Science and Engineering Directorate at the National Science Foundation.

Dr. Bajcsy is a member of the National Academy of Engineering and the National Academy of Science Institute of Medicine as well as a Fellow of the Association for Computing Machinery (ACM) and the American Association for Artificial Intelligence. In 2001, she received the ACM/Association for the Advancement of Artificial Intelligence Allen Newell Award, and was named as one of the 50 most important women in science in the November 2002 issue of *Discover Magazine*. In 2008, she was the recipient of the Benjamin Franklin Medal for Computer and Cognitive Sciences.

Klara Nahrstedt (F'08) received the Diploma in mathematics and numerical analysis from Humboldt University, Berlin, Germany, in 1985 and the Ph.D. degree from the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, in 1995.

She is Ralph and Catherine Fisher Professor at the Computer Science Department, University of Illinois at Urbana-Champaign. She was a research scientist in the Institute for Informatik in Berlin until 1990. Her research interests are directed toward multimedia distributed systems and networking, and tele-immersive systems.

Prof. Nahrstedt is the recipient of the University Scholar Award and the Humboldt Research Award. She is the associate editor of *ACM TOMCCAP*. She is the chair of the ACM SIG Multimedia. She is the member of ACM.