

Multi-camera tele-immersion system with real-time model driven data compression

A new model-based compression method for massive dynamic point data

Jyh-Ming Lien · Gregorij Kurillo · Ruzena Bajcsy

Published online: 19 May 2009
© Springer-Verlag 2009

Abstract Vision-based full-body 3D reconstruction for tele-immersive applications generates large amount of data points, which have to be sent through the network in real time. In this paper, we introduce a skeleton-based compression method using motion estimation where kinematic parameters of the human body are extracted from the point cloud data in each frame. First we address the issues regarding the data capturing and transfer to a remote site for the tele-immersive collaboration. We compare the results of the existing compression methods and the proposed skeleton-based compression technique. We examine the robustness and efficiency of the algorithm through experimental results with our multi-camera tele-immersion system. The proposed skeleton-based method provides high and flexible compression ratios from 50:1 to 5000:1 with reasonable reconstruction quality (peak signal-to-noise ratio from 28 to 31 dB) while preserving real-time (10+ fps) processing.

Keywords 3D tele-immersion · Multi-camera tele-immersion system · Point data compression · Model-based compression

J.-M. Lien (✉)
George Mason University, Fairfax, VA, USA
e-mail: jmlien@gmu.edu

G. Kurillo · R. Bajcsy
University of California, Berkeley, CA, USA

G. Kurillo
e-mail: gregorij@eecs.berkeley.edu

R. Bajcsy
e-mail: bajcsy@eecs.berkeley.edu

1 Introduction

Tele-immersion (TI) is aimed to enable users in geographically distributed sites to collaborate and interact in real time inside a shared simulated environment as if they were in the same physical space [4, 14, 26, 46]. The TI technology combines virtual reality for rendering, computer vision for image acquisition and 3D reconstruction, and various networking techniques for transmitting the data between the remote sites. All these steps need to be done in real time with the smallest delays possible. By combining these technologies, physically dislocated users are rendered inside the same virtual environment. TI is aimed at different network applications, such as collaborative work in industry and research, remote evaluation of products for ergonomics, remote learning and training, coordination of physical activities (e.g., dancing [48], rehabilitation [15]), and entertainment (e.g., games, interactive music videos, and ‘edutainment’ [32]). Figure 1 shows an example of a student learning tai-chi by following the motion of a tai-chi master captured using our system. In addition, 3D data captured locally could be used for kinematic analysis of body movement (e.g., in medicine [40] and rehabilitation [21, 22, 33]) or in computer animation [19]. The virtual environment can also include different synthetic objects (e.g., three-dimensional models of buildings) or digitized objects (e.g., magnetic resonance image of a brain), which can be explored through 3D interactions.

Figure 2 shows our TI apparatus and an example of its 360-degree stereo reconstruction of a tai-chi master. The stereo reconstruction is represented in the form of point cloud. We will discuss more details regarding our TI system in Sect. 6.1.

Major bottlenecks of existing TI systems A critical aspect of a TI system is the interactiveness. In order to provide



Fig. 1 A student (right, in blue shirt) is learning the tai-chi movements by following pre-recorded 3D TI data from a tai-chi teacher. These images (both the student and the tai-chi teacher) are rendered from the reconstructed 3D points using our tele-immersion system

interactiveness, we must satisfy the *real-time* requirement. The term ‘real time’ in this paper refers to real-time capturing and/or rendering, which should run with at 10 frames per second (fps) at least. One of the major bottlenecks preventing the current tele-immersion systems from fulfilling the real-time requirement is the transfer of large amount of data points generated by the stereo reconstruction from multiple images to the remote site. For example, data from images of 640×480 pixel depth and color maps from 10 camera clusters with a frame rate of 15 fps would require 220 MB/s network bandwidth. When two TI systems communicate, 440 MB/s bandwidth is required. Current image/video-based compression method [48] in our TI system only allows transmission of data with the rate of 5 or 6 frames per seconds.

Note that although network bandwidth may increase, an efficient and effective compression method will still be an important step in our TI system as the size of the data generated is becoming even larger. In the near future we will improve the resolution of the captured images and the frame rate by optimizing the depth calculation using finite elements and triangular wavelets [30]. These improvements will inevitably increase the bandwidth requirements for transfer of data through the network in real time. For this purpose, different compression schemes for real-time 3D video should be investigated.

In this paper, we focus on the data compression using motion estimation for TI applications. The proposed compression method provides high and flexible compression ratios with reasonable reconstruction quality. We will first provide an overview of our method and a review on related work in Sects. 2 and 3. In Sect. 4, we briefly discuss the idea of “iterative closest points” (ICP) method. Then, we will describe a novel technique of skeleton-based compression of 3D point data captured by our system. We discuss how to compress the data using the estimated articulated motions and how to encode non-rigid motions using regular grids (residual maps) in Sect. 5.1. Finally, we propose several efficient ways to detect temporal coherence in our residual maps including accumulating residuals from a few consecutive frames and

compress them using established video compression techniques in Sect. 5.2. Robustness and efficiency of the algorithm is examined both theoretically and experimentally in Sect. 6.

Main contribution To our knowledge we are the first to propose skeleton-based compression method for TI 3D point data. This paper presents an extensive study of our preliminary results [29] of the compression method. We focus on providing both theoretical and experimental evidences to demonstrate the advantages of the new compression method over the existing methods. We also provide a better motion estimation method that considers both local and global fitting errors.

Our compression method provides tunable and high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality. The “peak signal-to-noise ratio” of the proposed compression method is between 28 dB and 30.8 dB. Most importantly, our method can estimate motions from the data captured by our TI system in real time (10+ fps).

2 An overview of model driven compression

Few compression methods [46, 48] for TI data have been proposed. Most of these methods are image-based compressions. In this work, we focus on compressing 3D point data instead.

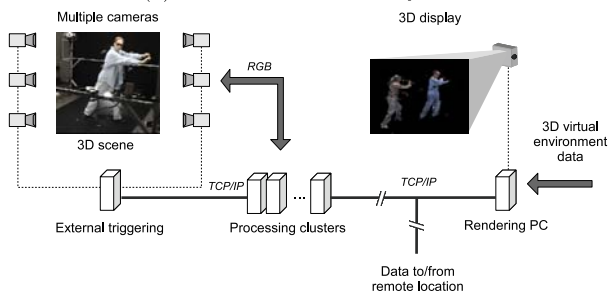
Several methods have been proposed to compress static or dynamic point data [3, 13, 20, 31, 36, 42]. However, there are several issues that have not yet been addressed in the literature, thus making our compression problem much more challenging. The main challenges include:

1. real-time requirement
2. only a short period of time for data accumulation, and
3. multiple or no data correspondences.

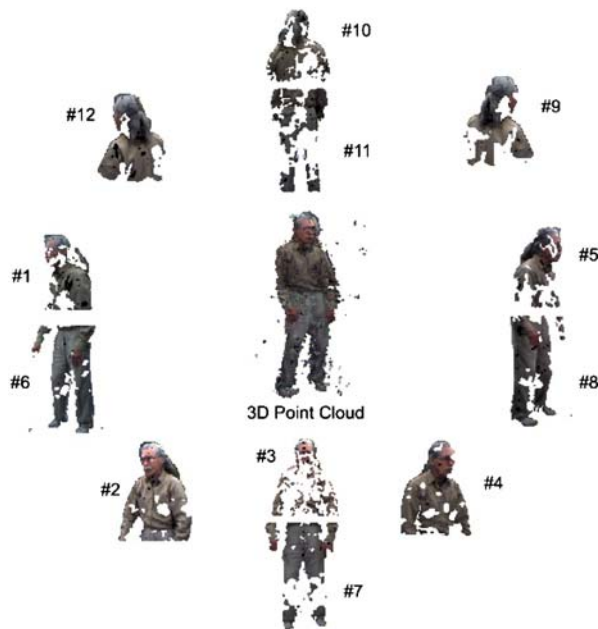
The real-time requirement prevents direct implementation of the majority of the existing compression methods for point data. For example, many methods [13, 36] are designed for off-line compression. These methods usually take



(a) Multi-camera stereo system



(b) System overview



(c) Full-body 3D reconstruction

Fig. 2 (a) Multi-camera stereo system for tele-immersion consists of 48 cameras arranged in 12 stereo clusters. The captured images are processed by 12 multi-core computers to provide 360-degree 3D full-body reconstruction in real time. (b) A schematic diagram of the connection and interfaces of our system. The processing cluster performs stereo reconstructions in parallel. (c) Partial depth image data captured by each of the 12 stereo camera clusters, which are accurately calibrated to a reference coordinate system, are combined inside the renderer into a full-body 3D reconstruction

tens of seconds to several minutes to compress a few thousands of points; our system creates about 50–100 thousand points in each frame. The real-time constraint also requires us to accumulate limited amount of data, i.e., the captured data should be transmitted as soon as they become available. However, the methods that exploit temporal coherence usually require data accumulation for a long period of time [3, 20, 31, 42].

Finding correspondences between two consecutive point data is another challenging problem. In many existing motion compression methods [16, 27, 49], the correspondence information is given. This assumption becomes unrealistic for our TI data. To make the problem of correspondence even more difficult, a point in a point set may correspond to zero or multiple points in the next point set. A point may not have any correspondence because the point may become occluded in the next frame from all cameras, and, similarly, a point can even correspond to multiple points because the point is in the visible regions of multiple cameras and is reconstructed multiple times.

Our approach This paper aims to address the challenges mentioned above in TI data compression. To cope with these challenges, we use a model-based approach. The main idea of this work is to take the advantage of prior knowledge of the objects in the TI environments, e.g. human figures, and to represent their motions using just a few parameters, e.g., joint positions and angles.

More specifically, our proposed method compresses points that represent human motion using motion estimation. Therefore, instead of transmitting the point clouds, we can simply transmit the motion parameters (e.g., joint positions and angles). This approach is based on the assumption that most points will move under rigid-body transform along with the associated skeleton.

In reality, point movements may deviate from this assumption, such as muscle movements and hair or cloth deformations. Our experimental results also show that when these “secondary” motions are discarded, the entire human movement becomes very unnatural. The same issue has also been observed by others, e.g., Arikan [3], in compressing optical-based motion capture data. Therefore, we further compress the deviations from the rigid movements. As we will see later (in Sect. 5.2), the deviations (we call “prediction residuals”) in most cases are small and can be dramatically compressed without reducing the quality significantly. An overview of this skeleton-based approach is shown in Fig. 3.

3 Related work

Many applications of distributed virtual reality [19, 28, 47] feature avatars to represent the human user inside the

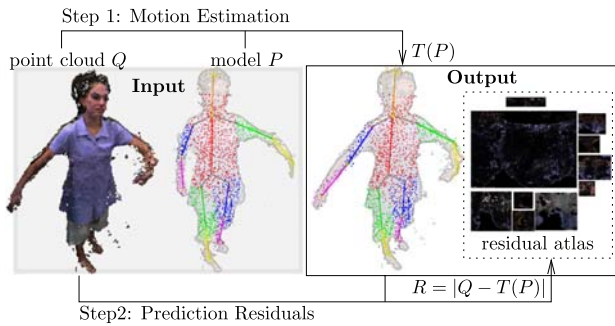


Fig. 3 Skeleton-based compression of the point data Q , where P is the skeleton, and T is the motion parameter (e.g., the skeleton configuration) that transforms P to Q . The prediction residual R is the difference between $T(P)$ and Q

computer-generated environment. An avatar is often designed as a simplified version of the user's physical features, while its movements are controlled via tracking of the user in the real world. Due to the limitations of the model appearance and movement ability, the interactive experience with other users is much more restricted. The level of immersion and interaction with the virtual environment can be increased by using realistic real-time models of the users. The vision-based tele-immersion systems are designed to address these issues.

One of the first TI systems was presented by the researchers at University of Pennsylvania [35]. Their system consisted of several stereo camera triplets used for the image-based reconstruction of the upper body, which allowed a local user to communicate to remote users while sitting behind a desk. A simplified version of the desktop tele-immersive system based on reconstruction from silhouettes was proposed by Baker et al. [4].

Blue-c tele-immersion system presented by Gross et al. [12, 46] allowed full-body reconstruction based on silhouettes obtained by several cameras arranged around the user. Reconstruction from the silhouettes provides faster stereo reconstruction as compared to the image-based methods; however, this approach is limited in accuracy, ability to reconstruct concave surfaces and discrimination of occlusions by objects or other persons inside the viewing area. Voxel-based interactive virtual reality system presented by Hasenfratz et al. [14] featured fast (25 fps) real-time reconstruction of the human body but was limited in acquiring details, such as clothing and facial features.

Our TI system has 360-degree full-body 3D reconstruction from images using twelve stereo triplets [18]. The captured 3D data are transferred using TCP/IP protocol to the rendering computer which reconstructs and displays point clouds inside a virtual environment at the local or remote sites in real time. The presented system has been successfully used in remote dancing applications [48] and learning of tai-chi movements [37] (also see Fig. 1). Videos and images are available at: <http://tele-immersion.citris-uc.org>.

TI data compression Only a few methods have been proposed to compress TI data. Kum and Mayer-Patel [24] proposed an algorithm to compress TI data in real time. Their method aimed to provide scalability to camera size, network bandwidth, and rendering power. Their method has 5:1 compression ratio. Recently, Yang et al. [48] proposed a real-time compression method to compress depth and color maps using lossless and lossy compression, respectively. Their method has 15:1 compression ratio. Unfortunately, the volume produced by these real-time compression methods is still too large to be efficiently transmitted in real time.

Lamboray et al. [25] proposed and studied several encoding techniques for streaming point set data based on the ideas of pixel-based differential prediction and redundancy. Their method is also based on image/video compression, i.e., MPEG video compression. Gross et al. [25, 46] also proposed an idea based on a concept similar to the back-face culling by transmitting only visible data. While we can certainly apply this strategy to points, it limits us from supporting some important functionalities, including interaction with virtual objects and motion analysis or understanding. In addition, transmitting only visible data will not provide any benefits when multiple views are needed, such as the tai-chi learning example shown in Fig. 1.

4 Preliminaries: motion estimation and ICP

Our method is heavily based on the idea of motion estimation. Extensive work has been done to track human motion in images (see surveys [2, 11]). Much of the existing work cannot be used to solve our motion estimation problem directly. For example, one of the most promising works that produce robust tracking is particle filter based tracking methods [17]. However, these methods are slow, thus cannot satisfy our real-time requirement. In this paper, we focus on the motion estimation from 3D points.

A popular method called "Iterative Closest Points" (ICP) [6, 41] has shown to be an efficient method for estimating rigid-body motion [43] in real time. For two given point sets P and Q , ICP first computes corresponding pairs $\{(p_i \in P, q_i \in Q)\}$. Using these corresponding pairs, ICP computes a rigid-body transform T such that the "matching error" defined in (1) between P and Q is minimized.

$$\text{error} = \arg \min_T \sum_i |(T(p_i), q_i)|^2, \quad (1)$$

where $|x, y|$ is the Euclidean distance between x and y .

The main step for minimize the matching error (see details in [6]) is to compute the cross-covariance matrix Σ_{PQ} of the corresponding pairs $\{p_i, q_i\}$,

$$\Sigma_{PQ} = \frac{1}{n} \sum_{i=1}^n [(p_i - \mu_p)(q_i - \mu_q)^t], \quad (2)$$

where μ_p and μ_q are the centers of $\{p_i\}$ and $\{q_i\}$, respectively, and n is the size of $\{p_i, q_i\}$. As outlined in Algorithm 4.1, ICP iterates these steps until the error is small enough. An important property of ICP is that the error always decreases monotonically to a local minimum when Euclidean distance is used for determining corresponding points.

Algorithm 4.1 ICP(P, Q, τ)

repeat

$$\left\{ \begin{array}{l} \text{find corresponding points } \{(p_i \in P, q_i \in Q)\} \\ \text{compute error and } T \text{ in (1)} \\ P = T(P) \end{array} \right.$$

until error $< \tau$

ICP has also been applied to estimate motions of articulated objects, e.g., hand [10], head [34], upper [9] and full bodies [23]. In these methods, ICP is applied to minimize the distance between each body part and the point cloud Q , i.e., $\text{error}(P, Q) = \sum_l \text{error}(P_l, Q)$, where P_l represents the points of the body part l . However, this approach lacks a global mechanism to oversee the overall fitting quality. For example, it is common that two body parts can overlap and a large portion of Q is not “covered” by P . Therefore, considering the global structure as a whole is needed. To the best of our knowledge, joint constraint [9, 23] is the only global measure studied.

5 Skeleton-based compression

One of the major bottlenecks of our current system for tele-immersion is the transfer of large amount of data points. Few methods [46, 48] have been proposed to address this problem using image and video compression techniques, but the data volume remains to be prohibitively large. In this section we will discuss our solutions to address the problems in TI data compression from a model driven approach. Figure 3 shows an overview of this model driven approach, which consists of two main steps: Motion estimation (Sect. 5.1) and Residual prediction (Sect. 5.2).

Our compression method provides (flexible) high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality. Our method can estimate motions from the data captured by our TI system in real time (10+ fps).

5.1 Motion estimation

We extend ICP to estimate motion of dynamic point data in real time. Our approach uses an initialization (may not be real-time) to generate a skeleton of the subject from the first point cloud and then a real-time tracking method will

fit the skeleton to the point clouds captured from the rest of the movements. Currently, the initialization step is done manually. Several methods, e.g., [7], exist and can provide us an initial skeleton to start the process. In this paper, we will focus on the motion tracking aspect.

5.1.1 Model and segmentation

Our method uses the segmentation of the initial frame to estimate the motion. The intuition behind this is to take the advantage that the *appearances* of the initial frame and the remaining frames are usually similar. From now on, we denote the point set in the first frame as P . We compute a segmentation of P using a given skeleton S . A skeleton S is organized as a tree structure, which has a root link, l_{root} , representing the torso of the body. Each link has the shape of a cylinder. An example of a skeleton is shown in Fig. 5. After segmentation, each point of P is associated with a link. We denote the points associated with a link l and a skeleton S as P_l and P_S , respectively. The associated points will move rigidly with the link as we estimate the motion. Note that $P_l \subset P_S$ and, initially, $P_S = P$.

5.1.2 ICP of articulated body

Now, we can start to fit the skeleton S (and its associated points P_S) to the next point cloud Q . Our goal here is to find a rigid-body transform for each link so the (global) distance between P_S and Q is minimized. Our method ARTICP is outlined in Algorithm 5.1.

Algorithm 5.1 ARTICP(S, Q, τ)

cluster Q

$q.\text{push}(l_{\text{root}})$

while $q \neq \emptyset$

$$\mathbf{do} \left\{ \begin{array}{l} l \leftarrow q.\text{pop}() \\ T \leftarrow \text{ICP}(P_l, Q) \\ \mathbf{for} \text{ each child } c \text{ of } l \\ \mathbf{do} \left\{ \begin{array}{l} \text{apply } T \text{ to } c \\ q.\text{push}(c) \end{array} \right. \end{array} \right.$$

global fitting

Clustering ARTICP first clusters the current point cloud Q . Each cluster has a set of points with similar *outward normals* and *colors*. These clusters will be used in ICP for finding better correspondences more efficiently. (See details in Sect. 5.1.3.)

Hierarchical ICP Next, we evoke ICP to compute a rigid-body transform for each link. Note that we do this in a fashion that the torso (root) of the skeleton is fitted first and limbs

are fitted last. By considering links in this order, ARTICP can increase the convergence rate by applying the transform not only to the link under consideration but also to the child-links. The rationale behind this is that a child-link, e.g., limbs, generally moves with its parent, e.g., torso. If the child-link does not follow the parent's movement, the movement of the child-link is generally constrained and is usually easier to track.

Articulation constraint We consider the articulation constraint, i.e., the transformed links should remain jointed. This is simply done by replacing both of the centers μ_p and μ_q in (2) by the joint position.

Global fitting Now, after we apply ICP to the individual links, the skeleton S is roughly aligned with the current point cloud Q but may still be fitted incorrectly without considering the entire skeleton as a whole. We propose to estimate the global fitting error as:

$$\text{global_error}(P, Q) = |F_e(P) - F_e(Q)|, \quad (3)$$

where the function F_e transforms a point set to a space where the difference is easier to compute. A more detailed discussion on F_e can be found in Sect. 5.1.4. Note that (3) also computes the prediction residuals.

Features Before discussing any details, we summarize the main features of our ARTICP below:

- hierarchical fitting for faster convergence
- articulation constraint
- monotonic convergence to local minimum guaranteed
- global error minimization.

5.1.3 Robustness, efficiency and convergence

Real time is one of the most important requirements of our system. Computing correspondences is the major bottleneck of ICP. Preprocessing the points using some spatial partitioning data structures, e.g., k-d tree [5], can alleviate this problem. Considering additional point attributes, e.g., color and normal information, can also increase both ICP's efficiency and robustness. Therefore ideally we would like to use all these tools to help us achieve the real-time requirement.

However, combining normals and colors with k-d tree is not trivial. For instance, the performance of k-d tree degrades quickly (in fact, exponentially) when the dimensionality of the space becomes high. Moreover, considering additional attributes usually does not guarantee monotonically convergence, which is an important property provided by ICP.

To gain efficiency, robustness, and convergence using point colors and normals and k-d tree, we construct a data

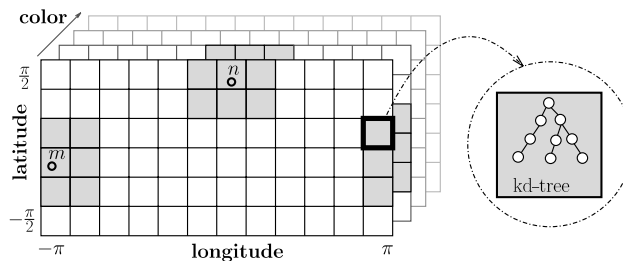


Fig. 4 A (n, c) -clustered k-d tree. Points n and m only look for their corresponding points in the gray regions around n and m

structure called (n, c) -clustered k-d tree (shown in Fig. 4), where n and c indicate normal directions and color, respectively, associated with the points. In this data structure, we cluster normals by converting them into a geographic coordinate system and cluster colors using hues and saturations. Then, we construct a k-d tree from the 3D point positions in each cluster.

When a point m looks for its corresponding point, ICP first determines which cluster m falls in and then examines the points in m 's neighboring clusters using their k-d trees. By doing so we can find better correspondences efficiently guaranteeing that ICP can always converge to a local minimum. Theorem 5.1 proves this property.

Theorem 5.1 Using an (n, c) -clustered k-d tree, ICP can always converge monotonically to a local minimum.

Proof Besl and McKay [6] have shown that ICP can always converge monotonically. An important fact that makes this true is that the corresponding points of a point p are always extracted from the same point set Q . The reason why considering point normals may not have monotone convergence is that in each iteration p 's corresponding point can be extracted from a different subset of Q . On the contrary, ICP using an (n, c) -clustered k-d tree will always search for p 's corresponding point from the same set of clusters and therefore is guaranteed to converge monotonically. \square

5.1.4 Global fitting

So far, we only consider fitting each link to the point cloud independently. However, as we mentioned earlier, considering links independently sometimes produces incorrect estimation even when the fitting error (1) is small. Therefore, we realize that, in order to get more robust results, it is necessary to consider the entire skeleton as a whole.

Global error Global fitting error (3) is measured as the difference between the skeleton associated point set P_S and the current point set Q . Due to the size difference and ambiguous correspondences of P_S and Q , it is not trivial to compute the difference directly. We need a function F_e to transform P_S and Q so that the difference is easier to estimate.

In fact, our selection of F_e has been mentioned before (in Sect. 5.1.1: Segmentation). It is important to note that the segmentation process is a global method, i.e., each link competes with other links to be associated with a point. Because $F_e(P_S)$ is already known at the beginning, we only need to compute $F_e(Q)$ at each time step. After segmenting Q , each link l will have two associated point sets, P_l and Q_l . To measure the difference of P_l and Q_l , we use a compact shape descriptor: the second moment, μ , of P_l and Q_l , and compute the difference of P_l and Q_l as

$$|\mu(P_l) - \mu(Q_l)|. \tag{4}$$

The idea behind this equation is that the second moment provides an idea of how points distribute around the center of the link. Therefore, an incorrect estimation will produce a large error.

When the error of any link is above a user-defined threshold, we start to minimize the global error. The minimization process is iterative and multi-resolution. That is, our global minimization method focuses on the difficult areas (i.e., areas that have larger error) by fitting P_l to Q_l and ignores the links that have already been fitted correctly. More specifically, to minimize the global error, we separate the links in the skeleton S and the points in the current point set Q into two subgroups, i.e., separate S into S^+ and S^- , and Q into Q^+ and Q^- . A link of S is in S^+ if its global error is lower than the threshold, otherwise the link is in S^- . A point of Q is in Q^+ if the point is a link in S^+ , otherwise the point is in Q^- . By separating the links and points into subgroups, we can ignore the groups that are considered as correctly estimated and repeat ICP (Algorithm 5.1) using only links in S^- and the point set Q^- , i.e., we call $\text{ARTICP}(S^-, Q^-, \tau)$. This process is iterated until all links

have acceptable global errors or until no improvement can be made.

5.2 Prediction residuals

Motion estimation brings the skeleton S (and its associated point set P_S) close to the current point cloud Q . Nevertheless, due to several reasons, e.g., non-rigid movements and estimation errors, our model P_S may not match Q exactly. We call the difference between P_S and Q “prediction residuals” (or simply residuals). Because after motion estimation P_S and Q are aligned to each other, we expect the residuals to be small. In this section, we present a method to compute the prediction residuals.

Similarly to (4) for estimating global errors, we compute the prediction residuals for each link by measuring the difference between P_l and Q_l . However, this time the difference is measured by regularly re-sampling the points on a grid. More precisely, we project both P_l and Q_l to a regular 2-D grid embedded in a cylindrical coordinate system defined by the link l . Because P_l and Q_l are now encoded in regular grids, we can easily compute the difference, which can be compressed using image compression techniques. Figures 5(a)–(c) illustrate the prediction residual computed from the torso link. Note that because this projection is invariant from a rigid-body transform, we only need to re-sample Q_l at each time step.

We determine the size of a 2-D grid from the shape of a link l and the size of l 's associated points $|P_l|$. We make sure that our grid size is at least $2|P_l|$ using the following formulation, i.e., the width and the height of the grid are $2\pi R_l S$ and $L_l S$, respectively, where R_l and L_l are the radius and the length of the link l and $S = \sqrt{\frac{|P_l|}{\pi R_l L_l}}$. We call that a grid encodes $x\%$ prediction residuals if the grid has size $2|P_l| \cdot \frac{x}{100}$. As we will see later, we can tune the grid size to produce various compression ratios and qualities.

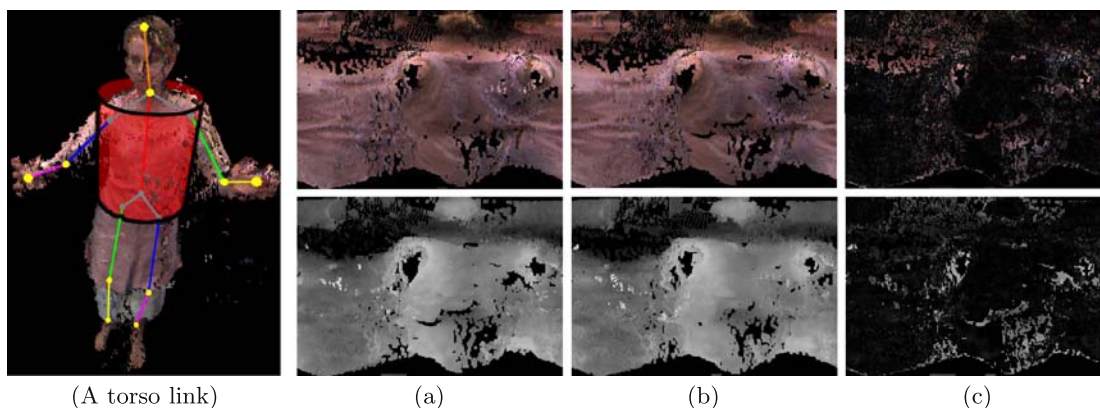


Fig. 5 A skeleton and the torso link shown as a cylinder. (a) Color (top) and depth (bottom) maps at time $t - 1$ of the torso link. The Y-axis of the maps is parallel to the link. (b) Color (top) and depth

(bottom) maps at time t of the torso link. (c) The differences between the maps at times $t - 1$ and t

6 System overview and experimental results

In this section, we first describe briefly about our TI system (Sect. 6.1), and we present our experimental results to evaluate our method using both synthetic (Sect. 6.2) and TI data (Sect. 6.3). Synthetic data is generated by sampling points from a polygonal mesh animated using the mocap data obtained from the online motion capture (mocap) library at Carnegie Mellon University [8]. We study synthetic data because it allows us to compare estimated motion with mocap, thus providing a ‘ground truth’ for us to evaluate our method. To evaluate the quality of our skeleton-based compression on the TI data, we project the decompressed data with various levels of prediction residuals back to each of the 6 virtual camera coordinates. We also compare the compression ratio and quality of our skeleton-based compression to H.264 video compression [1] and Yang et al.’s method [48]. All the experimental results in this section are obtained using a Pentium 4 3.2 GHz CPU with 512 MB of RAM. The best way to visualize our results is via the videos which can be found from our project web page: <http://tele-immersion.citris-uc.org>.

6.1 System overview

Our tele-immersion apparatus consists of 48 Dragonfly cameras [39], which are arranged in 12 clusters covering 360 degree view of the user(s). The cameras, equipped with 6 and 3.8 mm lenses, are mounted on an aluminum frame with dimensions of about $4.0 \times 4.0 \times 2.5 \text{ m}^3$ (Fig. 2(a)). Each cluster consists of three black and white cameras intended for stereo reconstruction and a color camera used for texture acquisition. The cluster PCs are dual or quad CPU (Intel Xeon, 3.06 GHz) machines with 1 GB of memory.

Synchronization Image-based stereo reconstruction requires precise synchronization between the cameras to calculate per-pixel correspondence. To synchronize image acquisition on all 48 cameras, we use external triggering on the cameras. The camera triggering pins are connected to the parallel port of the triggering server. In each data acquisition loop, the trigger computer sends a signal to initiate image capturing while TCP/IP messaging is used by the cluster computers to notify the server once the reconstruction has been completed.

Calibration The accuracy of the stereo reconstruction is greatly affected by the quality of the camera calibration. The errors can occur due to lens distortion, misalignment between image and lens plane, and deviations of position and rotation between the stereo cameras [50, 51]. The calibration of our multi-camera TI system is performed in two steps. The intrinsic calibration of camera parameters (i.e. distortion, optical center) is performed by the well-known Tsai

algorithm [44] using a checkerboard. Extrinsic parameters of each camera (i.e. position, orientation) are obtained in the second step by capturing position of a LED marker in about 10,000 points located inside the overlapping volume of all cameras.

Reconstruction algorithm The stereo reconstruction algorithm, which runs independently on each cluster computer, is composed of several steps [18]. First, the image is rectified and the distortion of the lens is corrected. Next, the background is subtracted from the image using a modified Gaussian average algorithm [38]. Finally, the depth map is computed pairwise from the three gray scale images using triangulation method.

Rendering Data streams from the twelve calibrated clusters are composed into a 3D image by a point-based rendering application developed using OpenGL. Based on camera calibration parameters the renderer combines the points received from each cluster into the full 3D reconstruction of the TI user (Fig. 2(c)). The complexity of the stereo reconstruction is currently limiting the frame rate to about 5 to 6 frames per second (fps). The required network bandwidth for this acquisition rate is below 5 Mbps. With our new stereo reconstruction algorithm that is currently under development [30], we anticipate a significant frame rate and resolution increase. With this improvement on the captured data, the bandwidth requirements can easily exceed 1 Gbps, therefore a better compression technique of the 3D data, such as the one proposed in this paper, becomes even more critical.

6.2 Motion estimation of synthetic data

We study the robustness of our motion estimation using four synthetic data sets: dance, crouch & flip, turn and tai-chi. Each frame in the synthetic data contains about 10,000 points without color, and 75% of the points are removed randomly to eliminate easy correspondences. The table below shows a summary of the mean motion estimation frame rates. Clearly, all motions can be estimated in real time (i.e., more than 24 frames per second) by our method.

Motion estimated	Dance (Fig. 6)	Crouch & flip (Fig. 7)	Turn (Fig. 8)	Tai-chi (not shown)
Avg. fps	39.1 fps	29.6 fps	48.4 fps	61.3 fps

The motion capture data from Carnegie Mellon University [8] is composed of a skeleton and a sequence of joint angles for each joint. In order to measure the quality of the estimated motion, we convert the joint angles to joint posi-

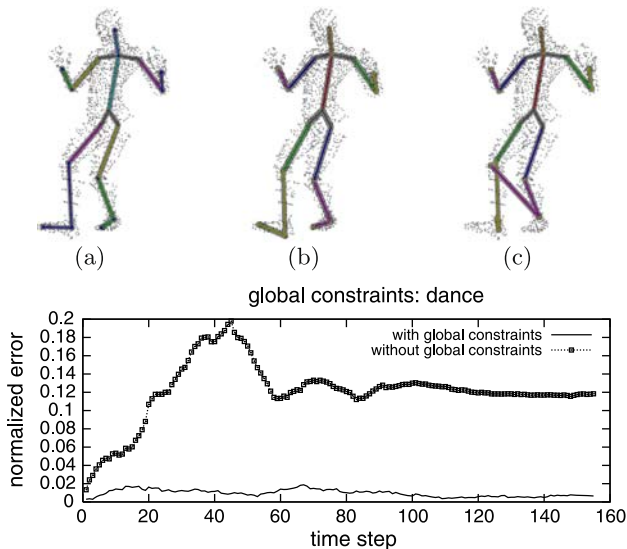


Fig. 6 *Top*: The postures from (a) Rond de Jambe motion capture data, and (b) estimation with global constraints and (c) without global constraints. *Bottom*: Tracking errors with and without global constraints. The *mean* errors with and without global constraints are 0.01 and 0.12, respectively, and the *maximum* errors with and without global constraints are 0.02 and 0.20, respectively

tions. Then, we measure the quality of estimated motion as the normalized distance offset e_t between all joint pairs, i.e.,

$$e_t = \frac{1}{n \cdot R} \sum_{i=1}^n |j_i^{\text{est}} - j_i^{\text{mocap}}|,$$

where n is the number of joints, j_i^{est} and j_i^{mocap} are the i th estimated and mocap joint positions, respectively, and R is the radius of the minimum bounding ball of the point cloud, i.e., we scale the skeleton so that the entire skeleton is inside a unit sphere. In the following paragraphs, we will compare the qualities of our method in several scenarios.

With and without global constraints We compare the difference between the tracking algorithm with and without articulation and global fitting constraints. Figure 6 shows that global constraints *do* have a significant influence on motion estimation quality.

Downsampling factor In this experiment, we study how point size (% of downsampling) affects the motion estimation quality. A 10% downsampling from a point set with 10,000 points means that only 1000 points are used for estimating motion. Figure 7 shows our experimental results. As we can see from the results, the downsampling rate does not have a significant influence on the quality until down to 1% for this crouch & flip motion. This is important because this experiment shows that we indeed can downsample significantly without introducing visible estimation errors.

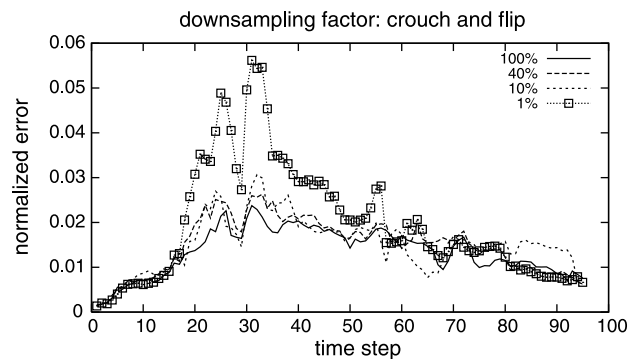


Fig. 7 Tracking errors with different downsampling factors

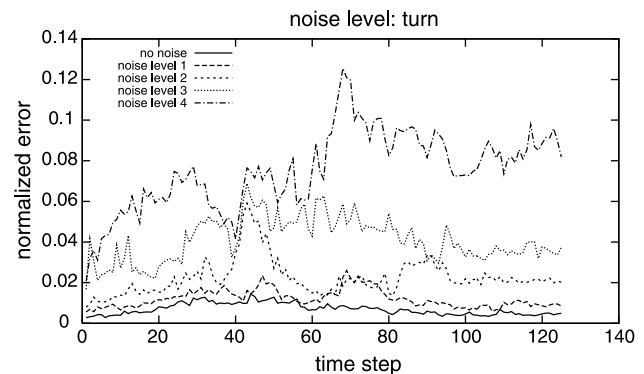
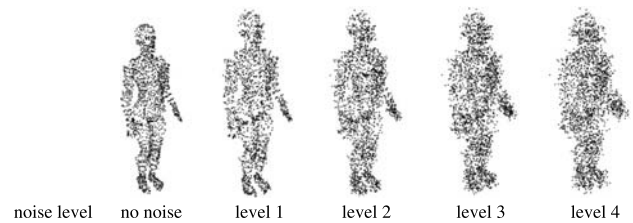


Fig. 8 Tracking errors with different noise intensities

Noise level In this experiment, we study how noise affects the quality. The noise is introduced by perturbing each point in a random direction with a distance x , where x is a random variable with a Gaussian distribution. From Fig. 8, it is clear that as we increase the noisiness of the points, the difference between the estimated motion and the “ground truth” increases. However, the overall difference remains small even for very noisy data.

6.3 Compressing TI data

We evaluate the results of our compression method using four motion sequences captured by our TI system. These

motions are performed by two professional dancers, one a student and one a tai-chi master. The data captured by our TI system have about 7 to 9 thousand points in each frame. Because the TI data is much noisier than the synthetic data and can have significant missing (due to occlusions) or overlapping data (due to overlapping camera views), estimating motion from the TI data is more difficult, thus we use 50% of the initial point set as our model. This value is selected experimentally. The table below shows that we can still maintain at least 10 fps interactive rate in all studied cases.

Motion estimated	Dancer 1 (Fig. 3)	Dancer 2 (Fig. 5)	Student (Fig. 1)	Tai-chi master (Fig. 10)
Avg. fps	11.9 fps	11.5 fps	12.5 fps	12.6 fps

Quality Unlike synthetic data, we do not have a ground truth to directly evaluate the quality of our method using TI data. Our approach is to compute the differences between the point data without compression (called *input data*) and the data *compressed and decompressed* using the proposed method (called *compressed data*). In order to measure the difference between these two point sets, we render images of each point set from six camera views in each time frame. *These cameras are 60 degrees separated and are located at radius and y coordinate so that the whole body can be captured.* Then, we compute the “peak signal-to-noise ratio” (PSNR) for each pair of images (I, J) rendered from the input data and compressed data by the same camera in the same frame. PSNR is defined as: $20 \log_{10} \left(\frac{255}{\text{rmse}} \right)$ [45], where $\text{rmse} = \sqrt{\sum_i |I_i - J_i|^2}$ is the root mean squared error of images I and J . A larger PSNR indicates a better quality. Typical PSNR values in lossy image compression are between 20 and 40 dB [45]. The results of our study are summarized in Table 1.

In Table 1, We also measure the quality by varying the levels of residual considered. We considered three compression levels, i.e., compression without residuals and with 50% and 100% residuals. A compression with $x\%$ residuals means its residual maps are $x\%$ smaller than the full residual maps. We see that encoding residuals indeed generates

Table 1 Compression quality. The “peak signal-to-noise ratio” (PSNR) is computed between rendered images of the input and compressed point data. PSNR is defined as: $20 \log_{10} \left(\frac{255}{\text{rmse}} \right)$, where $\text{rmse} = \sqrt{\sum_i |I_i - J_i|^2}$ is the root mean squared error of images I and J . We

Motion		Dancer 1	Dancer 2	Student	Tai-chi master
Avg. PSNR	no residuals	23.87 dB	24.10 dB	25.82 dB	26.27 dB
	25% residuals	25.77 dB	26.18 dB	27.96 dB	28.75 dB
	100% residuals	27.95 dB	28.27 dB	29.91 dB	30.83 dB

better reconstruction than that without residuals by 4 dB. Figure 9 provides an even stronger evidence that considering residuals always produces better reconstructions for all frames. Another important observation is that the compression quality remains to be the same (around 30) for the entire motion. Figure 10 shows that the difference is more visible when the prediction residuals are not considered.

We would like to note that using PSNRs may not fully reflect the compression quality. For example, the compressed data generated using no residuals looks much more robotic and unnatural than that generated using only 25% residuals. This difference can be easily observed from the rendered animation but not in Table 1. Another property that does not reflect by measuring PSNRs is that some noisy data or outliers are removed after compression.

Compression ratio One of the motivations of this work is that no existing TI compression methods can provide compression ratios high enough to provide real-time transmission. In this experiment, we show that our skeleton-based compression method can achieve 50:1 (with 100% residuals) to 5000:1 (without residuals) compression ratios. As we have shown earlier, our compression method can provide different compression ratio by varying the level of residuals considered during encoding. To increase our compression

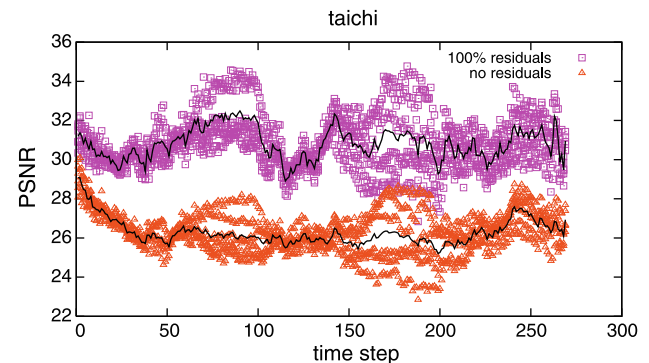


Fig. 9 PSNR values from the tai-chi motion. Each point in the plot indicates a PSNR value from a rendered image. For each time step, there are 12 dots plotted, which indicate 12 images rendered from two point sets with 100% and 0% residuals. The bold lines are the mean PSNR values for 100% (top) and 0% (bottom) residuals

also measure the quality by varying the levels of residual considered. A compression with $x\%$ residuals means its residual maps are $x\%$ smaller than the full residual maps

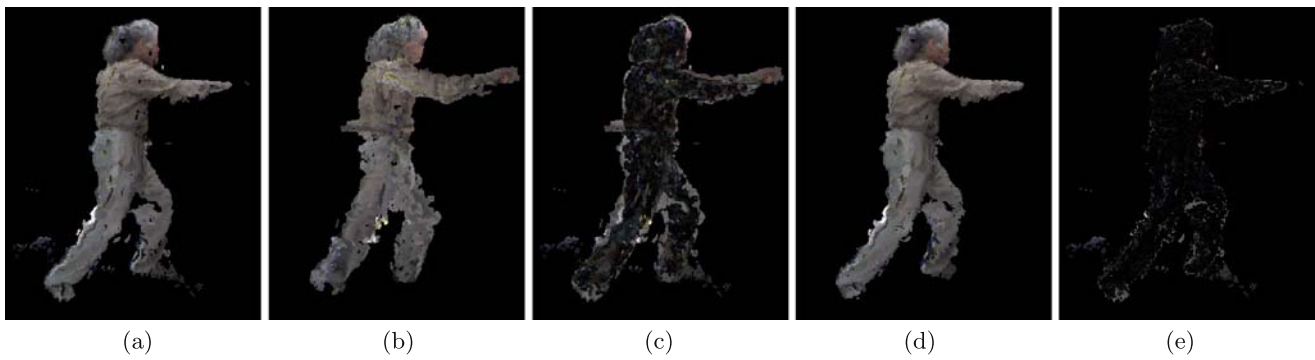


Fig. 10 Reconstructions from the compressed data and their differences with the uncompressed data. **(a)** Uncompressed data. **(b)** Compressed without residuals. **(c)** Difference between **(a)** and **(b)**. **(d)** Compressed with 25% residuals. **(e)** Difference between **(a)** and **(d)**

Table 2 Compression ratio. Both Yang et al.'s [48] and H.264 (we use an implementation from [1]) compression methods took the color and depths images as their input and output. The compression ratio of

H.264 reported in this table is obtained using 75% of its best quality. We use jpeg and png libraries to compress color and depth residuals, respectively

Motion		Dancer 1	Dancer 2	Student	Tai-chi master
Size before compression		142.82 MB	476.07 MB	1.14 GB	988.77 MB
Compression ratio	Yang et al.'s [48]	11.36	11.73	10.23	14.41
	H.264 [1]	64.04	53.78	32.04	49.58
	no residuals	1490.31	3581.52	5839.59	5664.55
	25% residuals	195.62	173.52	183.80	183.82
	100% residuals	66.54	55.33	60.29	61.43

furthermore, we use jpeg and png libraries to compress color and depth residuals, respectively. In addition, we compare our compression method to other compression methods, including the method proposed by Yang et al. [48] and H.264 (for H.264, we compare to an implementation in Apple's QuickTime [1]). We summarize our experimental results in Table 2.

We would like to note that we can achieve very high compression ratio while maintaining reasonable reconstruction quality (as shown earlier) because our method is fundamentally different from the other two methods. Both Yang et al.'s and H.264 are image (or video) based compressions, which take color and depth images as their input and output. On the contrary, our skeleton-based compression converts the color and depth images to motion parameters and prediction residuals. Moreover, even though H.264 provides high quality and high ratio compression, H.264 is not a real-time compression for the amount of data that we considered in this work.

7 Conclusion

In this paper we have presented a skeleton-based data compression aimed for the use in tele-immersive environments.

Data produced by the full-body 3D stereo reconstruction requires high network bandwidth for real-time transmission. Current implementation of the image/video-based compression method [48] in our tele-immersion system only allows transmission of data with the rate of 5 or 6 fps. Using model-based compression techniques can significantly reduce the amount of data transfer between the remote TI sites.

Using the real-time (10+ fps) motion estimation technique described in this paper, we can compress data by converting point cloud data to a limited number of motion parameters while the non-rigid movements are encoded in a small set of regular grid maps. Prediction residuals are computed by projecting the points associated with each link to a regular 2D grid embedded in a cylindrical coordinate system defined by the skeleton link.

Our experiments showed that our compression method provides adjustable high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality with peak signal-to-noise ratio (PSNR) from 28 to 31 dB.

Limitations and future work The proposed method, however, has several drawbacks. In the case of occlusions, the motion estimation fails since the skeleton cannot be fitted uniquely into the point cloud data. The algorithm is also not

able to cope with non-rigid motions of the body (e.g. long hair, user wears a skirt, etc.). Compared to the current video compression standards, the PSNR values of described algorithm are still low. The described algorithm can be extended to motion-based compression of multiple targets (i.e. two or more users inside the TI system).

Currently, we are upgrading our TI system with an improved and more robust stereo reconstruction algorithm which will produce less noisy point clouds with higher frame rate (15 to 20 fps) [30]. We have shown that the integration of the described skeleton-based compression algorithm can significantly reduce our network bandwidth requirements. Extracted skeleton information can be used to interact with the objects in the virtual environment or to analyze subject's movement. The kinematic data can also be applied as a feedback to the user during learning of motions. For example, in the case of tai-chi learning via tele-immersion [37], the student could receive real-time feedback on how close his/her movement is to the teacher's movement. In addition, the skeletonization method can be applied off-line to analyze motion in connection with other motion sensor technologies (e.g. accelerometers).

References

1. Adobe. Quicktime 7.0 h.264 implementation (2006)
2. Aggarwal, J.K., Cai, Q.: Human motion analysis: a review. *Comput. Vis. Image Underst.* **73**(3), 428–440 (1999)
3. Arikan, O.: Compression of motion capture databases. *ACM Trans. Graph.* **25**(3), 890–897 (2006)
4. Baker, H., Tanguay, D., Sobel, I., Gelb, D., Gross, M., Culbertson, W., Malzenbender, T.: The coliseum immersive teleconferencing system. In: *Proceedings of International Workshop on Immersive Telepresence, Juan-les-Pins, France* (2002)
5. Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Commun. ACM* **18**(9), 509–517 (1975)
6. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(2), 239–256 (1992)
7. Cheung, K.M., Baker, S., Kanade, T.: Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2003
8. CMU. Graphics Lab Motion Capture Database. Carnegie Mellon University. <http://mocap.cs.cmu.edu/>
9. Demirdjian, D., Darrell, T.: 3-D articulated pose tracking for untethered deictic reference. In: *ICMI'02: Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, Washington, DC, p. 267. IEEE Comput. Soc., Los Alamitos (2002)
10. Dewaele, G., Devermay, F., Horaud, R.: Hand motion from 3D point trajectories and a smooth surface model. In: *ECCV* (1), pp. 495–507 (2004)
11. Gavrilu, D.M.: The visual analysis of human movement: a survey. *Comput. Vis. Image Underst.* **73**(1), 82–98 (1999)
12. Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Gool, L.V., Lang, S., Strehlke, K., Moere, A.V., Staadt, O.: Blue-c: a spatially immersive display and 3D video portal for telepresence. *ACM Trans. Graph.* **22**(3), 819–827 (2003)
13. Gumhold, S., Karni, Z., Isenburg, M., Seidel, H.-P.: Predictive point-cloud compression. In: *Siggraph 2005 Sketches* (2005)
14. Hasenfratz, J., Lapierre, M., Sillion, F.: A real-time system for full-body interaction with virtual worlds. In: *Proceedings of Eurographics Symposium on Virtual Environments*, pp. 147–156. Eurographics Association, Aire-la-Ville (2004)
15. Holden, M.K.: Virtual environments for motor rehabilitation: review. *Cyberpsychol. Behav.* **8**(3), 187–211 (2005)
16. Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. In: *SCA'03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Aire-la-Ville, Switzerland, pp. 126–135. Eurographics Association, Aire-la-Ville (2003)
17. Isard, M., Blake, A.: Condensation—conditional density propagation for visual tracking. *Int. J. Comput. Vis.* **29**(1), 5–28 (1998)
18. Jung, S., Bajcsy, R.: A framework for constructing real-time immersive environments for training physical activities. *J. Multimed.* **1**(7), 9–17 (2006)
19. Kalra, P., Magnenat-Thalman, N., Mocozet, L., Sannier, G., Aubel, A., Thalman, D.: Real-time animation of realistic virtual humans. *IEEE Comput. Graph. Appl.* **18**(25), 42–56 (1998)
20. Karni, Z., Gotsman, C.: Compression of soft-body animation sequences. *Comput. Graph.* **28**(1), 25–34 (2004)
21. Keshner, E.A.: Virtual reality and physical rehabilitation: a new toy or a new research and rehabilitation tool? *J. Neuroengineering Rehabil.* **1**(8) (2004)
22. Keshner, E., Kenyon, R.: Using immersive technology for postural research and rehabilitation. *Assist. Technol.* **16**, 54–62 (2004)
23. Knoop, R.D.S., Vacek, S.: Sensor fusion for 3D human body tracking with an articulated 3D body model. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Walt Disney Resort, Orlando, Florida, 15 May 2006
24. Kum, S.-U., Mayer-Patel, K.: Real-time multidepth stream compression. *ACM Trans. Multimedia Comput. Commun. Appl.* **1**(2), 128–150 (2005)
25. Lamboray, E., Würmlin, S., Gross, M.: Real-time streaming of point-based 3D video. In: *VR'04: Proceedings of the IEEE Virtual Reality 2004*, Washington, DC, p. 91. IEEE Comput. Soc., Los Alamitos (2004)
26. Lanier, J.: Virtually there. *Sci. Am.* **4**, 52–61 (2001)
27. Lengyel, J.E.: Compression of time-dependent geometry. In: *SI3D'99: Proceedings of the 1999 Symposium on Interactive 3D Graphics*, New York, NY, pp. 89–95. ACM, New York (1999)
28. Li, L., Zhang, M., Xu, F., Liu, S.: Ert-vr: an immersive virtual reality system for emergency rescue training. *Virtual Real.* **8**(3), 194–197 (2005)
29. Lien, J.-M., Bajcsy, G.K.R.: Skeleton-based data compression for multi-camera tele-immersion system. In: *Proceedings of the International Symposium on Visual Computing (ISVC)*, pp. 347–354 (2007)
30. Lopez, E.J.L.: Finite element surface-based stereo 3D reconstruction, April 2006. Poster presentation given at the Trust NSF Site Visit
31. Marc Alexa, W.M.: Representing animations by principal components. *Comput. Graph. Forum* **19**(3), 411–418 (2000)
32. Mason, H., Moutahir, M.: Multidisciplinary experiential education in second life: a global approach. In: *Second Life Education Workshop*, San Francisco, California, pp. 30–34 (2006)
33. McComas, J., Pivik, J., Laflamme, M.: Current uses of virtual reality for children with disabilities. *Virtual Environments in Clinical Psychology and Neuroscience* (1998)
34. Morency, L.-P., Darrell, T.: Stereo tracking using ICP and normal flow constraint. In: *Proceedings of International Conference on Pattern Recognition* (2002)

35. Mulligan, J., Daniilidis, K.: Real-time trinocular stereo for tele-immersion. In: Proceedings of 2001 International Conference on Image Processing, Thessaloniki, Greece, pp. 959–962 (2001)
36. Ochotta, T., Saupe, D.: Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In: Symposium on Point-Based Graphics, pp. 103–112 (2004)
37. Patel, K., Bailenson, J.N., Hack-Jung, S., Diankov, R., Bajcsy, R.: The effects of fully immersive virtual reality on the learning of physical tasks. In: Proceedings of the 9th Annual International Workshop on Presence, Ohio, USA, pp. 87–94 (2006)
38. Piccardi, M.: Background subtraction techniques: a review. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Hague, Netherlands, pp. 3099–3104 (2004)
39. Point Grey Research Inc, Vancouver, Canada
40. Robb, R.: Virtual reality in medicine: A personal perspective. *J. Vis.* **5**(4), 317–326 (2002)
41. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling (3DIM), pp. 145–152 (2001)
42. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: SCA'05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, New York, NY, pp. 209–217. ACM, New York (2005)
43. Simon, D., Hebert, M., Kanade, T.: Real-time 3-D pose estimation using a high-speed range sensor. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA'94), vol. 3, pp. 2235–2241 (1994)
44. Tsai, R.: A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE J. Robot. Autom.* **RA3**(4), 323–344 (1987)
45. William Pennebaker, J.M.: *JPEG: Still Image Data Compression Standard*. Springer, Berlin (1992)
46. Würmlin, S., Lamboray, E., Gross, M.: 3D video fragments: dynamic point samples for real-time free-viewpoint video. *Comput. Graph.* **28**, 3–14 (2004)
47. Yang, Y., Wang, X., Chen, J.X.: Rendering avatars in virtual reality: integrating a 3D model with 2D images. *Comput. Sci. Eng.* **4**(1), 86–91 (2002)
48. Yang, Z., Cui, Y., Anwar, Z., Bocchino, R., Kiyancilar, N., Nahrstedt, K., Campbell, R.H., Yurcik, W.: Real-time 3D video compression for tele-immersive environments. In: Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'06), San Jose, CA (2006)
49. Zhang, J., Owen, C.B.: Octree-based animated geometry compression. In: DCC'04: Proceedings of the Conference on Data Compression, Washington, DC, p. 508. IEEE Comput. Soc., Los Alamitos (2004)
50. Zhang, D., Nomura, Y., Fujii, S.: Error analysis and optimization of camera calibration. In: Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS'91), Osaka, Japan, pp. 292–296 (1991)
51. Zhao, W., Nandhakumar, N.: Effects of camera alignment errors on stereoscopic depth estimates. *Pattern Recogn.* **29**(12), 2115–2126 (1996)



Jyh-Ming Lien is currently (2007–) Assistant Professor of the Computer Science Department at the George Mason University. He received B.Sc. (1999) and Ph.D. (2006) degrees in Computer Science from the National Chengchi University, Taiwan and from the Texas A&M University, respectively. He was a Postdoctoral Researcher at the University of California, Berkeley.



Gregorij Kurillo received B.Sc. (2001) and D.Sc. (2006) degrees in Electrical Engineering from the University of Ljubljana, Slovenia. There he was Research Assistant with the Laboratory of Robotics and Biomedical Engineering from 2002 to 2006. He has participated in two European Union projects aimed at Assistive Technology. Dr. Kurillo has been Postdoctoral Researcher with the University of California, Berkeley, since 2006, where he is working on the tele-immersion project. His research interests include multi-camera calibration, stereo vision, virtual reality, and robotics.



Ruzena Bajcsy was Director of CITRIS and Professor of EECS Department at the University of California, Berkeley. Prior to coming to Berkeley, she was Assistant Director of the Computer Information Science and Engineering Directorate (CISE) between December 1, 1998 and September 1, 2001. She came to the NSF from the University of Pennsylvania where she was Professor of Computer Science and Engineering. She was also Director of the University of Pennsylvania's General Robotics and Active Sensory Perception Laboratory, which she founded in 1978. In 2001 she became a recipient of the ACM A. Newell award.

Dr. Bajcsy received her Master's and Ph.D. degrees in Electrical Engineering from Slovak Technical University in 1957 and 1967, respectively. She received a Ph.D. in Computer Science in 1972 from Stanford University, and since that time has been teaching and doing research at Penn's Department of Computer and Information Science. She began as Assistant Professor and within 13 years became Chair of the department.