SKELETON-BASED COMPRESSION OF 3-D TELE-IMMERSION DATA

Jyh-Ming Lien[*]

George Mason University
Fairfax, VA, 20120

Ruzena Bajcsy[†]

University of California
Berkeley, CA, USA, 94720

ABSTRACT

Due to technology advances, reconstructing three-dimensional representations of physical environments in **real time** using cheap and non-professional cameras and PCs becomes possible. These advances will make 3-D tele-immersive environments available for everyone in the near future. However, the amount of data captured by a Tele-Immersion (TI) system can be very large. Compression is needed to ensure real-time transmission of the data. Due to this real-time requirement, compressing TI data can be a challenging task and no current techniques can effectively address this issue. In this paper, we propose a skeleton-based compression. The main idea of this approach is to take advantage of prior knowledge of objects, e.g., human figures, in the physical environments and to represent their motions using just a few parameters. The proposed compression method provides tunable and high compression ratios (from 50:1 to 5000:1) with reasonable reconstruction quality. Moreover, the proposed method can estimate motions from the noisy data captured by our TI system in real time.

1. INTRODUCTION

Three-dimensional Tele-Immersion (TI) systems aim to provide a rich communication medium that captures 3-D data from physically separated environments and projects the captured data into a shared virtual space in *real time* [1, 2]. Due to recent technology advances, a 3-D TI system can be constructed cheaply using off-the-shelf products. Figure 1 shows a 360-degree full-body reconstruction from our TI system.

One of the major bottlenecks of the current 3-D TI systems is the difficulty of transmitting the huge amount of data in real time. For example, our system, which generates 640×480 pixel depth and color maps from 10 camera clusters in a rate of 15 frames per second (fps), requires a 220 MB/s bandwidth. When two TI systems communicate, the required bandwidth becomes 440 MB/s to guarantee real-time interaction. Several methods [3, 4] have been proposed to address this problem using image and video compression techniques, but the data volume remains to be prohibitively large. Currently, our system, using the compression method proposed by Yang et al. [4], can only reach 4$\sim$5 fps when connected with a single remote TI.

TI data. Our TI data is composed of a stream of points as shown in Figure 1. These points are captured from our TI system with 10 calibrated camera clusters. Each camera cluster has three b/w and one color cameras and generates both depth and color maps. Points generated by the clusters are registered to a global coordinate system. Because of the nature of the stereo reconstruction, our data can be very noisy.

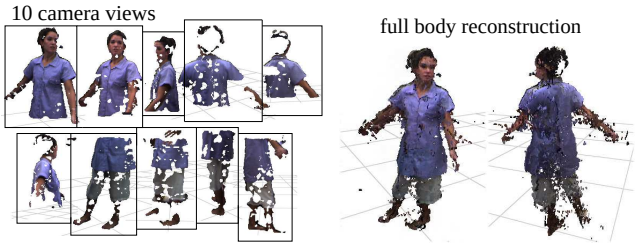


Fig. 1. Point data captured from our TI system with 10 calibrated camera clusters. All clusters are registered to a global coordinate system. The full 3-D reconstruction is simply a union of the individual reconstructions without any post processing, e.g., hole filling or noisy point removal, due to the real-time constraint.

Challenges of compressing TI data. Methods have been proposed to compress static or dynamic point data [5, 6, 7, 8, 9, 10]. However, there are several issues that have not yet been addressed in the literature, thus making our compression problem more challenging. The main challenges include:

1. real-time constraint,
2. very little time for data accumulation, and
3. multiple or no data correspondences.

The real-time constraint forbids us to directly use most of the existing compression methods for point data, e.g., the methods [5, 6] designed for off-line compression (which usually takes tens of seconds to several minutes to compress a few thousands of points, and our system creates about 7$\sim$9 thousand points in each frame). The real-time constraint also requires us to accumulate only very little data, i.e., the captured data should be transmitted as soon as they become available. However, data accumulation is usually necessary for the methods that exploit temporal coherence [7, 8, 9, 10].

Finding correspondences between two consecutive point data is another challenging problem. In many existing motion compression methods [11, 12, 13], the correspondence information is given. This is not the case in our TI data. To make the problem 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 so that no cameras can see it, and, similarly, a point can correspond to multiple

[*]jmlien@gmu.edu
[†]bajcsy@eecs.berkeley.edu

1-4244-1354-0/07/$25.00

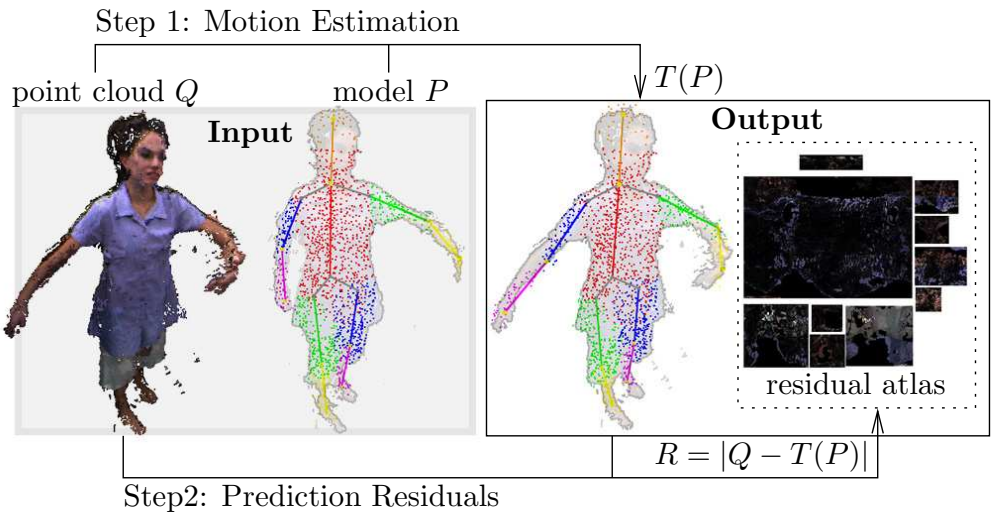


Fig. 2. Skeleton-based compression of the point data Q, where P is our model, and T is the motion parameter (e.g., the skeleton configuration) that transforms P to Q. The prediction residuals R is the difference between $T(P)$ and Q.

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. We use a model based approach. The main idea of this work is to take advantage of prior knowledge of objects, e.g. human figures, in the TI environments 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. This approach is based on the assumption that most points will move under rigid-body transform along with the 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 [8], in compression motion capture data. Therefore, we further compress the deviations from the rigid movements. As we will see later (in Section 4), the deviations (we call prediction residuals) in most cases are small. An overview of this skeleton-based approach is shown in Figure 2.

Main contribution. As far as we know, we proposed the first skeleton-based compression method for TI data. Our compressor provides (tunable) 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. RELATED WORK

TI data compression. Only a few methods have been proposed to compress TI data. Kum et al. [3] 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. [4] 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 transmitted in real time.

Motion estimation. Extensive work has been done to track human motion in images (see surveys [14, 15]). In this paper, we focus on motion estimation from 3-D points.

A popular method called "Iterative Closest Points" (ICP) [16, 17] has shown to be an efficient method for estimating rigid-body motion [18] 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 Eq. 1 betweem P and Q is minimized.

$$error = \underset{T}{\operatorname{argmin}} \sum_i |(T(p_i), q_i)|^2 \,. \tag{1}$$

The main step for minimize the matching error (see details in [16]) is to compute the cross-covariance matrix Σ_{PQ} of the corresponding pairs $\{p_i, q_i\}$,

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

where μ_p and μ_q are the centers of $\{p_i\}$ and $\{q_i\}$, resp., and n is the size of $\{p_i, q_i\}$. As outlined in Algorithm 2.1, 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 finding corresponding points.

Algorithm 2.1: ICP(P, Q, τ)

repeat
- find corresponding points $\{(p_i \in P, q_i \in Q)\}$
- compute $error$ and T in Eq. 1
- $P = T(P)$

until $error < \tau$

ICP has also been applied to estimate motions of articulated objects, e.g., hand [19], head [20], upper [21] and full bodies [22]. In these methods, ICP is applied to minimize the distance between each body part and the point cloud Q, i.e., $error(P, Q) = \sum_l error(P_l, Q)$, 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 [21, 22] is the only global measure studied.

3. 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. Several methods, e.g., [23], exist and can provide us an initial skeleton to start the process. In this paper, we will focus on the motion tracking aspect.

3.1. Model and Segmentation

Our method uses the segmentation of the initial frame to estimate the motion. This intuition behind this is the advantage of 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 Figure 4. 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, $P_S = P$.

3.2. 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. ARTICP is outlined in Algorithm 3.1.

Algorithm 3.1: ARTICP(S, Q, τ)

cluster Q
q.push(l_{root})
while $q \neq \emptyset$
do
- $l \leftarrow q$.pop()
- $T \leftarrow$ICP(P_l, Q)
- **for** each child c of l
  - **do**
    - apply T to c
    - q.push(c)

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 Section 3.3.)

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 applies the transform to the children of the link. 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 easy 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 Eq. 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:

$$\text{global\_error}(P, Q) = |F_e(P) - F_e(Q)|, \qquad (3)$$

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 Section 3.4. Note that Eq. 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.

3.3. 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 [24], can alleviate this problem. Considering additional point attributes, e.g., color and

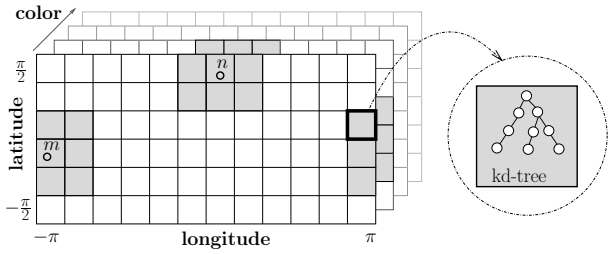


Fig. 3. A (n, c)-clustered k-d tree. Points n and m only look for their corresponding points in the gray regions around n and m.

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 properly provided by ICP.

To gain efficiency, robustness, and convergence using point colors and normals and k-d tree, we construct a data structure called (n, c)-clustered k-d tree (shown in Figure 3), 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 3-D point positions in each cluster.

When a point m looks for its corresponding point, ICP first finds 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 3.1 proves this property.

Theorem 3.1. *Using a (n, c)-clustered k-d tree,* ICP *can always converge monotonically to a local minimum.*

Proof. Besl and Mckay [16] 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 uses a (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. □

3.4. 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 (Eq. 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 (Eq. 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 Section 3.1): 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

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

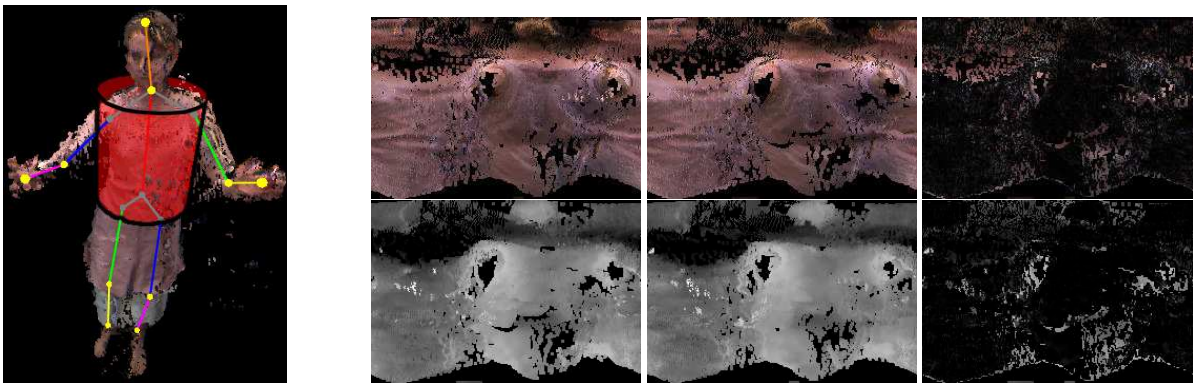
The idea behind this equation is that the second moment provides an idea of how points distribute around the center of the link. Therefore, an incorrect estimation will produce a large error.

When the error of any link is above a user-defined threshold, we start to minimize the global error. To minimize the global error, we separate the skeleton S and the current point set Q into groups: S^+, S^-, 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 link 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 3.1) using only links in S^- and the point set Q^-, i.e, we call ARTICP(S^-, Q^-, τ). This process is iterated until all links have low global errors or until no improvement can be obtained.

4. PREDICTION RESIDUALS

Motion estimation brings the skeleton S (and its associated point set P_S) close to the current point cloud Q. However, 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.

Similar to Eq. 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 4(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.



(A torso link) (a) (b) (c)

Fig. 4. A skeleton and the torso link shown as a cylinder. (a) Color and depth maps at time $t-1$ of the torso link. The Y-axis of the maps is parallel to the link. (b) Color and depth maps at time t of the torso link. (c) The differences between the maps at times $t-1$ and t.

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 $L_l S$, resp., 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.

5. EXPERIMENTAL RESULTS

We use both synthetic and TI data to evaluate our method. Synthetic data is generated by sampling points from a polygonal mesh animated using the mocap data from Carnegie Mellon University. Next, we study synthetic data because it provides a 'ground truth' for us to evaluate our method. We study the quality of our skeleton-based compression on the TI data with various levels of prediction residuals considered. We also compare our skeleton-based compression to H.264 video compression [25] and Yang et al.'s method [4]. 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 page at: *http://www.cs.gmu.edu/∼jmlien/research/TI/*

5.1. Motion estimation of synthetic data

We study the robustness of our motion estimation using three synthetic data sets: dance, crouch&flip, turn and tai chi. Each frame in the synthetic data contains about 10,000 points, and 75% of the points are removed randomly to eliminate easy correspondences. The table below shows a summary of the averaged motion estimation frame rate. All motions can be estimated in real time by our method.

motion estimated	dance (Fig. 5)	crouch&flip (Fig. 6)	turn (Fig. 7)	tai-chi (in video)
avg. fps	39.1 fps	29.6 fps	48.4 fps	61.3 fps

The motion capture data from Carnegie Melllon University 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 positions. Then, we measure the quality as the normalized distance offset e_t between joints, i.e.,

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

where n is the number of joints, j_i^{est} and j_i^{mocap} are the i-th estimated and mocap joint positions, resp., 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 5 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. That is 10% downsampling means that only 1000 points from a point set with 10,000 points are used for estimating motion. Figure 6 shows our experimental results. As we can see from the result, the downsampling rate does not have a significant influence on the quality (at least down to 1% for this crouch & flip motion).

Noise level. In this experiment, we study how noise affects the quality. From Figure 7, 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.

5.2. 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 student and one tai-chi master. The data captured by our TI system have

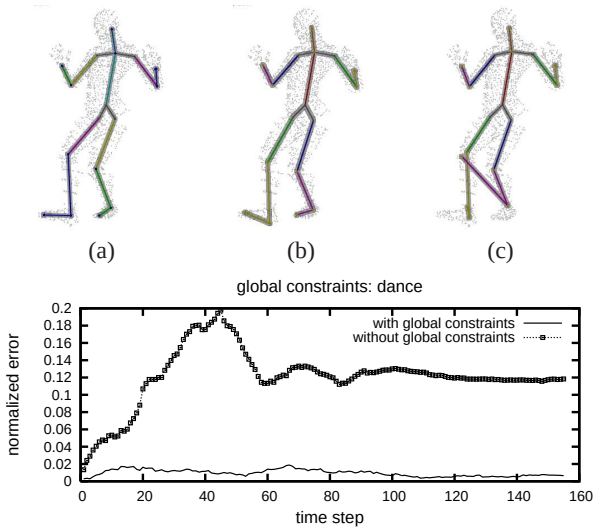


Fig. 5. Top: The postures from (a) motion capture, and (b) estimation with global constraints and (c) without global constraints. Bottom: Tracking errors with and without global constraints.

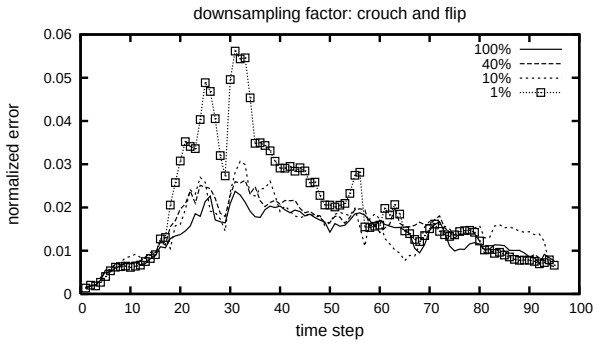


Fig. 6. Tracking errors with different downsampling factors.

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 or overlapping data, estimating motion from the TI data is more difficult, thus we use 50% of the initial point set as our model. The table below shows that we can still maintain at least 10 fps interactive rate in all studied cases.

motion estimated	dancer 1 (Fig 2)	dancer 2 (Fig 4)	student (in video)	tai-chi master (Fig 9)
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 (60 degree separated) camera views in each time frame. Then, we compute the "peak signal-to-noise ratio" (PSNR) for each pair of images (I, J) rendered from

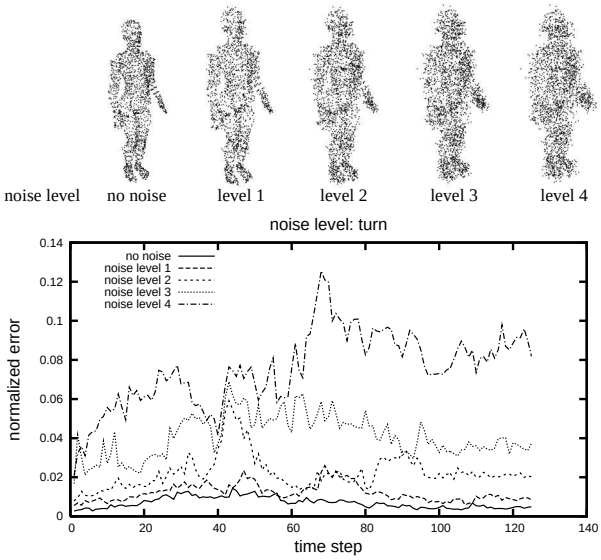


Fig. 7. Tracking errors with different noise intensities.

the input data and compressed data by the same camera in the same frame. PSNR is defined as: $20 \log_{10} (\frac{255}{rmse})$, where $rmse = \sqrt{\sum_i |I_i - J_i|^2}$ is the root mean squared error of images I and J. A larger PSNR indicates a better quality. Typical PSNR values in image compression are between 20 and 40 dB. The results of our study are summarized in Table 1.

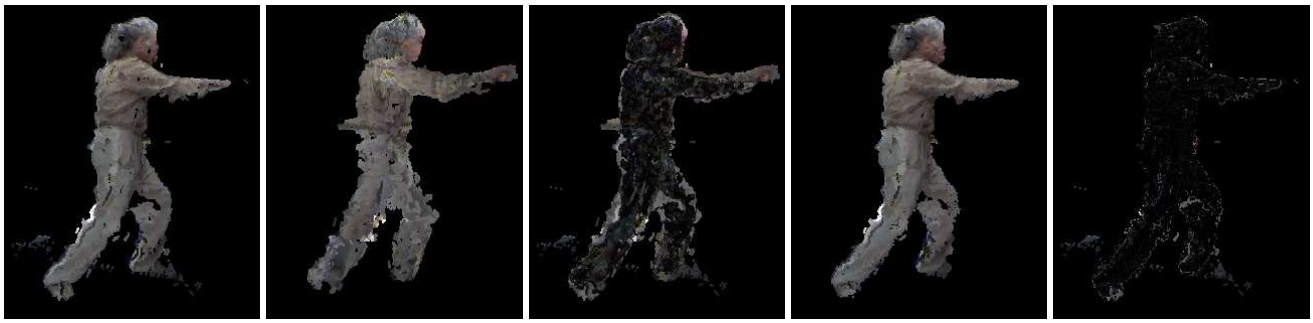
In Table 1, We also measure the qualify by varying the levels of residual considered. We considered three compression levels, i.e., compression without residuals and with 50% and 100% residuals. A compression with $x\%$ residuals means its residual maps are $x\%$ smaller than the full residual maps. We see that encoding residuals indeed generates better reconstruction than that without residuals by 4 dB. Figure 8 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 9 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.

Compression Ratio. One of the motivations of this work is that no existing TI compression methods can provide high compression ratios. In this experiment, we show that our skeleton-based compression method can achieve 50:1 (with 100% residuals) to 5000:1 (without desiduals) 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 furthermore, we use jpeg and png libraries to compress color and depth residuals, respectively. In addition, we

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: $20 \log_{10}(\frac{255}{rmse})$, where $rmse = \sqrt{\sum_i |I_i - J_i|^2}$ is the root mean squared error of images I and J. We 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.

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



(a) (b) (c) (d) (e)

Fig. 9. 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 residuals. (e) Difference between (a) and (d).

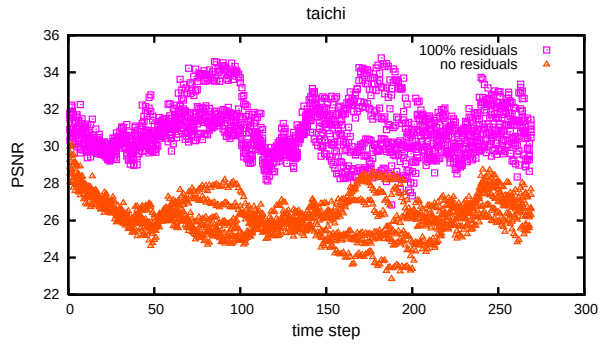


Fig. 8. PSNR values from the tai-chi motion. Each point in the plot indicates a PSNR value from a rendered image. For each time step, there are 12 points, which indicate 12 images rendered from two points with 100% and 0% residuals.

compare our compression method to other compression methods including the method proposed by Yang et al.'s [4] and H.264 (we use an implementation from [25]). We summarize our experimental results in Table 2.

We would like to note that because our method is fundamentally different from the other two methods, we can achieve very high compression ratio while maintaining reasonable reconstruction quality (as shown earlier). 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.

6. DISCUSSION AND FUTURE WORK

We proposed a skeleton-based compression method to convert point set data to a few motion parameters and a set of residual maps, whose size can be tuned adaptively according to available space and time. Our motion estimation method, ARTICP, can efficiently estimate the motions from synthetic and TI data at interactive rates (10∼60 frames per second).

Despite our promising results, our method has limitations. First, we observed that our motion estimation fails when many occlusions occurred. This problem becomes more serious when multiple subjects appear in the scene. Second, we assume that points move under rigid-body transform and non-rigid motions are usually comparatively small. However, this assumption becomes invalid when the subjects wear skirts or long hairs. Third, although the quality of our compression method is reasonable, its PSNR values are still low compared to the current video compression standards. It is not clear at this point how our method can be improved to produce higher quality results.

Currently, we are exploring the possibility of estimating motions from multiple targets. Our compression method can also be extended in several ways. For example, we would like to investigate efficient ways to detect temporal coherence in our residual maps. One possible approach is to accumulate residuals from a few frames and compress them using video compression techniques.

Note that even though we focus only on the application in data compression in this paper, the result of this work can also be used broadly in applications including human activity recognition and analysis [26, 27], marker-less motion capture, and Ergonomics.

Table 2. Compression ratio. Both Yang et al.'s [4] and H.264 (we use an implementation from [25]) 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. [4]	11.36	11.73	10.23	14.41
	H.264 [25]	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

7. REFERENCES

[1] K. Daniilidis, J. Mulligan, R. McKendall, G. Kamberova, D. Schmid, and R. Bajcsy, "Real-time 3d tele-immersion," 2000.

[2] Zhenyu Yang, Klara Nahrstedt, Yi Cui, Bin Yu, Jin Liang, Sang hack Jung, and Ruzena Bajscy, "Teeve: The next generation architecture for tele-immersive environment," in *ISM '05: Proceedings of the Seventh IEEE International Symposium on Multimedia*, Washington, DC, USA, 2005, pp. 112–119, IEEE Computer Society.

[3] Sang-Uok Kum and Ketan Mayer-Patel, "Real-time multidepth stream compression," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 1, no. 2, pp. 128–150, 2005.

[4] Zhenyu Yang, Yi Cui, Zahid Anwar, Robert Bocchino, Nadir Kiyanclar, Klara Nahrstedt, Roy H. Campbell, and William Yurcik, "Real-time 3d video compression for tele-immersive environments," in *Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'06)*, San Jose, CA, 2006.

[5] Stefan Gumhold, Zachi Karni, Martin Isen-burg, and Hans-Peter Seidel, "Predictive point-cloud compression," in *Siggraph 2005 Sketches*, 2005.

[6] Tilo Ochotta and Dietmar Saupe, "Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields," in *Symposium on Point-Based Graphics*, 2004, pp. 103–112.

[7] Wolfgang Müller Marc Alexa, "Representing animations by principal components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[8] Okan Arikan, "Compression of motion capture databases," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 890–897, 2006.

[9] Zachi Karni and Craig Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25–34, 2004.

[10] Mirko Sattler, Ralf Sarlette, and Reinhard Klein, "Simple and efficient compression of animation sequences," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, New York, NY, USA, 2005, pp. 209–217, ACM Press.

[11] Lawrence Ibarria and Jarek Rossignac, "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, Switzerland, 2003, pp. 126–135, Eurographics Association.

[12] Jerome Edward Lengyel, "Compression of time-dependent geometry," in *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, New York, NY, USA, 1999, pp. 89–95, ACM Press.

[13] Jinghua Zhang and Charles B. Owen, "Octree-based animated geometry compression," in *DCC '04: Proceedings of the Conference on Data Compression*, Washington, DC, USA, 2004, p. 508, IEEE Computer Society.

[14] J. K. Aggarwal and Q. Cai, "Human motion analysis: a review," *Comput. Vis. Image Underst.*, vol. 73, no. 3, pp. 428–440, 1999.

[15] D. M. Gavrila, "The visual analysis of human movement: a survey," *Comput. Vis. Image Underst.*, vol. 73, no. 1, pp. 82–98, 1999.

[16] Paul J. Besl and Neil D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[17] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2001, pp. 145–152.

[18] David Simon, Martial Hebert, and Takeo Kanade, "Real-time 3-d pose estimation using a high-speed range sensor," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '94)*, May 1994, vol. 3, pp. 2235–2241.

[19] Guillaume Dewaele, Frederic Devernay, and Radu Horaud, "Hand motion from 3d point trajectories and a smooth surface model.," in *ECCV (1)*, 2004, pp. 495–507.

[20] Louis-Philippe Morency and Trevor Darrell, "Stereo tracking using icp and normal flow constraint," in *Proceedings of International Conference on Pattern Recognition*, 2002.

[21] David Demirdjian and Trevor Darrell, "3-d articulated pose tracking for untethered diectic reference," in *ICMI '02: Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, Washington, DC, USA, 2002, p. 267, IEEE Computer Society.

[22] R. Dillmann S. Knoop, S. Vacek, "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, May 15 2006.

[23] Kong Man Cheung, Simon Baker, and Takeo Kanade, "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.

[24] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sept. 1975.

[25] Adobe, "Qicktime 7.0 h.264 implementation," 2006.

[26] Gutemberg Guerra-Filho, Cornelia Fermüller, and Yiannis Aloimonos, "Discovering a language for human activity," in *Proc. of the AAAI 2005 Fall Symposium on An-ticipatory Cognitive Embodied Systems*, 2005.

[27] Gutemberg Guerra-Filho and Yiannis Aloimonos, "Human activity language: Grounding concepts with a linguistic framework," in *Proc. of the 1st International Con-ference on Semantics and Digital Media Technology (SAMT)*, 2006.