Model Driven Compression of 3-D Tele-Immersion Data

Jyh-Ming Lien
Ruzena Bajcsy

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-170
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-170.html

December 12, 2006

Model Driven Compression of 3-D Tele-Immersion Data

Jyh-Ming Lien Ruzena Bajcsy
{neilien,bajcsy}@eecs.berkeley.edu
University of California Berkeley
Berkeley, CA USA

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 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 model driven 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 three-dimensional data from physically separated environments and projects the captured data into a shared virtual space in *real time* [6, 25]. Due to recent technology advances, a 3-D TI system can be constructed cheaply using off-the-shelf products. Figure 1 shows a 180-degree full-body reconstruction from our TI system.

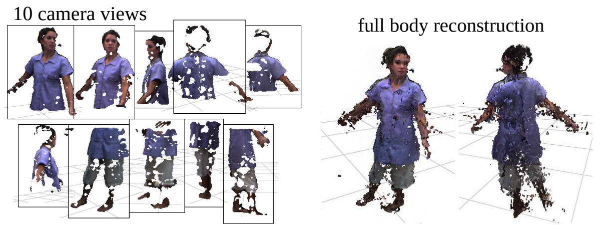


Figure 1: Point data captured from our TI system with 10 calibrated camera clusters. All clusters are registered to a global coordinate system. The full 3-D reconstruction is simply a union of the individual reconstructions without any post processing, e.g., hole filling or noisy point removal, due to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

One of the major bottlenecks of the current 3-D TI systems is in 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. Several methods [15, 24] have been proposed to address this problem using image and video compression techniques, but the data volume remains to be prohibitedly large. Currently, our system, using the compression method by Yang et al. [24], can only reach 4$\sim$5 fps when connected with a single remote TI.

Challenges of compressing TI data. Our TI data is composed of a stream of points as shown in Figure 1. Methods have been proposed to compress static or dynamic point data. However, there are issues that have not yet been addressed in the literature, thus making our compression problem more challenging. The main challenges include:

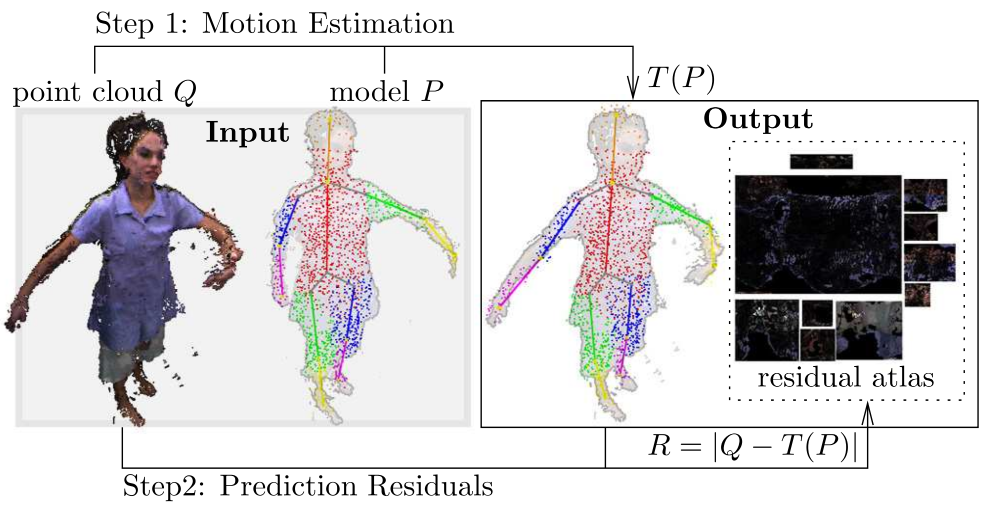


Figure 2: Model driven compression of the point data Q, where P is our model, and T is the motion parameter (e.g., the skeleton configuration) that transforms P to Q. The prediction residuals R is the difference between $T(P)$ and Q.

1. real-time constraint,
2. little or no 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 [12, 19] 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 8$\sim$10 thousand points in each frame). The real-time constraint also requires us to accumulate little or no 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 the temporal coherence [17, 3, 14, 22].

Finding correspondences between two point data is another challenging problem. In many existing motion compression methods [13, 16, 26], the correspondence information is given. This is not the case in our TI data. Moreover, a point in a point set may correspond to zero (e.g., due to occlusion) or multiple (reconstructed in multiple views) points in the next point set.

Our approach. This paper aims to address the problems in TI data compression from a model driven 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. 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 model driven approach is shown in Figure 2.

Key results. Our compressor provides (tunable) 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).

2 Related Work

TI data compression. Only a few methods have been proposed to compress TI data. Kum et al. [15] 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. [24] 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 [2, 9]). In this paper, we focus on motion estimation from 3-D points.

A popular method called "Iterative Closest Points" (ICP) [4, 20] has shown to be an efficient method for estimating rigid-body motion [23] 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 for

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

The main step for solving Eq. 1 (see details in [4]) is to compute the cross-covariance matrix Σ_{PQ} of the corresponding pairs $\{p_i, q_i\}$,

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

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), Besl and Mckay [4]

```
Input. Two point clouds, $P$ and $Q$
1: repeat
2:     find corresponding points $\{(p_i \in P, q_i \in Q)\}$
3:     compute $error$ and $T$ in Eq. 1.
4:     $P = T(P)$
5: until $error < \tau$
```

ICP has also been applied to estimate motions of articulated objects, e.g., hand [8], head [18], upper [7] and full bodies [21]. 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 [7, 21] is the only global measure studied.

On the contrary, our ICP-based motion tracking method, called ARTICP, considers both joint and global fitting constraints. More specifically, 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 the next two sections. Note that Eq. 3 also computes the prediction residuals (shown in Figure 2).

3 Motion Estimation

We extend ICP to estimate motion of dynamic point data in real-time. Our approach uses a computationally more expensive (may not be real time) initialization to generate an accurate skeleton of the subject 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., [5], exist and can provide us an initial skeleton to start the process. In this paper, we will focus on the tracking aspect.

3.1 Model and Segmentation

Unlike most ICP-based work [18, 8, 7, 21], our method uses the segmentation of the initial frame to estimate the motion by taking the advantage of that the appearances of the initial frame, denoted by P, and the remaining frames are usually similar. 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. After segmentation, each point of P is associated with a link. A point p is associated with a link l if l is closest to p. The closeness is computed as the follows:

$$\text{closeness}(p,l) = \begin{cases} 0 & \vec{n}_p \cdot \overrightarrow{l\,p} \leq 0 \\ 1 & p \in l \\ e^{-\text{dist}(p,\partial l)/r} & \text{otherwise} \end{cases}, \tag{4}$$

where $\vec{n}_p$ is the normal of p, $\overrightarrow{l\,p}$ is the vector from (p's closest point on) link l to p, ∂l is the surface of the link and r is the radius of the link. We denote the points associated with a link l and a skeleton S as P_l and P_S, respectively.

3.2 ICP of Articulated Body

Now, we can start to fit the skeleton 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. The main features of our ARTICP include:

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

ARTICP is outlined in Algorithm 3.1. We will discuss each of these important features in detail.

Algorithm 3.1 ARTICP(S, Q)

```
Input. a skeleton $S$, and the current point cloud $Q$.
1: cluster $Q$                              ▷ see Section 3.3
2: $q$.push($l_{root}$)                        ▷ $q$ is a queue
3: while $q \neq \emptyset$ do
4:     $l \leftarrow q$.pop()                       ▷ $l$ is a link
5:     $T \leftarrow$ ICP($P_l$, $Q$)          ▷ $T$ is a rigid-body transform
6:     for each child $c$ of $l$ do
7:         apply $T$ to $c$
8:         $q$.push($c$)
9: global fitting                           ▷ see Section 3.4
```

Clustering. ARTICP first clusters the current point cloud Q. Each cluster has a set of points with similar outward normals and similar colors. These clusters will be used in ICP for finding better correspondences. (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, 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 consider a global fitting step in Section 3.4.

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. Considering additional point attributes, e.g., color and normal information, can increase both ICP's efficiency and robustness. Some spatial partitioning data structures, e.g, k-d tree, can also alleviate this problem, but the performance of these data structures degrades quickly when the dimensionality of the space becomes high. Moreover, considering addition attributes usually does not guarantee monotonically convergence in ICP.

To gain efficiency, robustness, and convergence using point colors and normals, we construct a data structure called (n, c)-clustered kd-tree (shown in Figure 3). In this data structure, we cluster normals using 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 will only exam the points in m's the neighborhood in a (n, c)-clustered kd-tree. By doing so we can find better correspondences efficiently while guarantee that ICP can always converge to a local minimum. Theorem 3.1 proves this property.

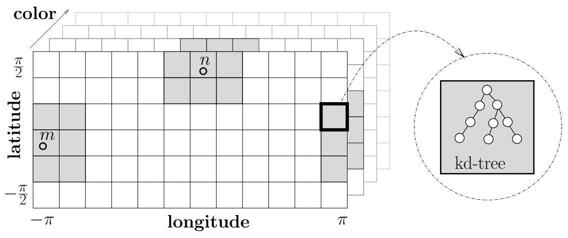


Figure 3: A (n, c)-clustered kd-tree. Points n and m only look for their corresponding points in the gray regions around n and m.

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

Proof. Besl and Mckay [4] 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 monotonic convergence is that in each iteration p' corresponding point can be extracted from a different subset of Q. On the contrary, ICP uses a (n, c)-clustered kd-tree will always search for p's corresponding point from the same set of clusters, 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 causes overlapping 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 whole.

Global error. Global fitting error (Eq. 3) is measured as the difference between the model P_S and the current point cloud 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 the function F_e to transform P_S and Q so that the difference is easier to estimate. Our selection of F_e has indeed been mentioned before (in Section 3.1): Segmentation. It is important to note that the segmentation process is a global method, in which each link competes with other links to be associated with a point (using Eq. 4). Because $F_e(P_S)$ is already known at the beginning, we only need to compute $F_e(Q)$. 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 third moment, μ_3, of P_l and Q_l. To summarize, our global function is approximated as:

$$\text{global\_error}(P_S, Q) = \sum_{l \in S} |\mu_3(P_l) - \mu_3(Q_l)| \,. \quad (5)$$

When the error is above a user-defined threshold, we start to minimize the global error.

Minimizing global error. To minimize the global error, we iterate between the following two steps: matching P_l to Q_l using ICP and segmenting Q (i.e., computing a new Q_l). Another benefit of Eq. 5 is that we can distinguish the quality of each link. Therefore, we can just apply this iteration to the links with intolerably large errors.

4 Prediction Residuals

Motion estimation brings the skeleton 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 P_S and Q are close 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. 5, we compute the prediction residuals for each link by finding the difference between P_l and Q_l. However, this time the difference is measured by re-sampling the points. 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. Figure 5

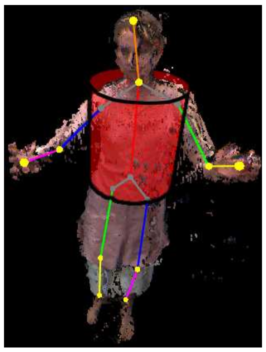


Figure 4: A torso link.

illustrates the prediction residual computed from the torso link in Figure 4. Because this projection is invariant from a rigid-body transform, we only need to re-sample Q_l at each time step.

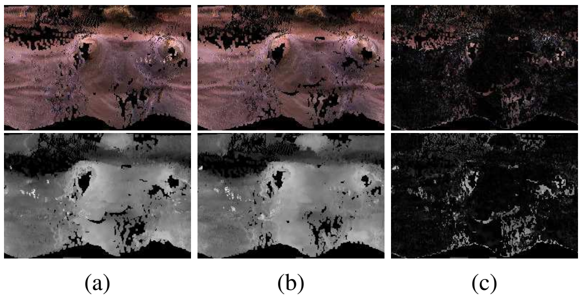


Figure 5: (a) Color and depth maps at time $t-1$ of the torso link in Figure 4. The Y-axis of the maps is parallel to the link. (b) Color and depth maps at time t of the same torso link. (c) The differences between the maps at times $t-1$ and t.

We determine the size of a 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 100% prediction residuals if the grid has size $2|P_l|$. 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 from a polygonal mesh animated using the mocap data from Carnegie Mellon University. We study synthetic data because it provides a 'ground truth' for us to evaluate our method. We study the quality of our model-driven compression on the TI data with various levels of prediction residuals considered. We also compare our model-driven compression to H.264 video compression [1] and Yang et al.'s method [24]. All the experimental results in this section are obtained using a Pentium4 3.2GHz CPU with 512 MB of RAM. The best way to visualize our results is via the submitted videos.

5.1 Motion estimation of synthetic data

We study the robustness of our motion estimation using three synthetic data sets, dancing, crouch&flip, and turning. 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.

motion estimated	dance (Fig. 6)	crouch&flip (Fig. 7)	turn (Fig. 8)	tai-chi (in video)
avg. fps	39.1 fps	29.6 fps	48.4 fps	61.3 fps

The quality of the estimated motion is measured 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, j_i^{est} and j_i^{mocap} are the 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.

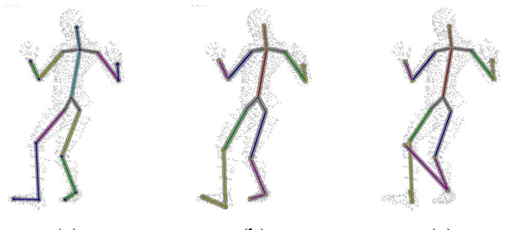
With and without global constraints. We compare the difference between the results from the tracking algorithm with and without articulation and global fitting constraints. Figure 6 shows that global constraints *do* have a significant influence on motion estimation quality.

Downsampling factor. In this experiment, we study how point size (% of downsampling) affects the motion estimation quality. Figure 7 shows that 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 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.

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 dancers, one student and



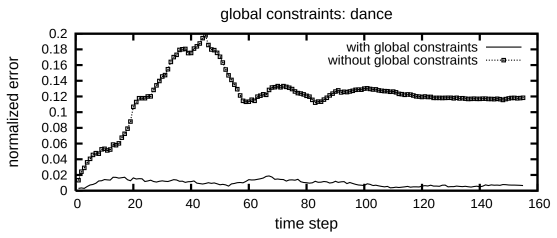


Figure 6: Top: The postures from (a) motion capture, and (b) estimation with global constraints and (c) without global constraints. Bottom: Tracking errors with and without global constraints.

one tai-chi master. The date captured by our TI system have about 8 to 10 thousand points in each frame. Because the TI data is much noisier than the synthetic data and can have significant missing 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 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 evaluate the quality in the TI data. Instead, we directly measured the quality of our method as the difference between the point data before and after our model-driven compression. More specifically, we compute the "peak signal-to-noise ratio" (PSNR) of the images rendered from uncompressed and compressed point data. In our experiments, two sets of images (one for each point set) are rendered from six (60 degree separated) camera views in each frame. The results of our study are summarized in Table 1.

In Table 1, we compare the reconstruction quality by

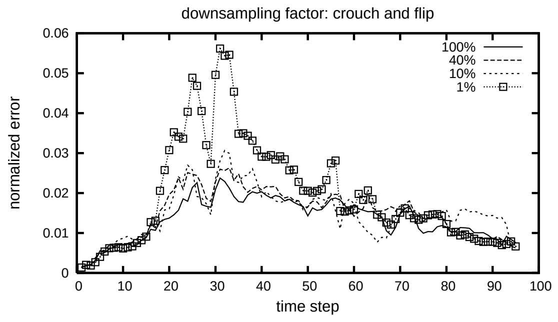


Figure 7: Tracking errors with different downsampling factors.

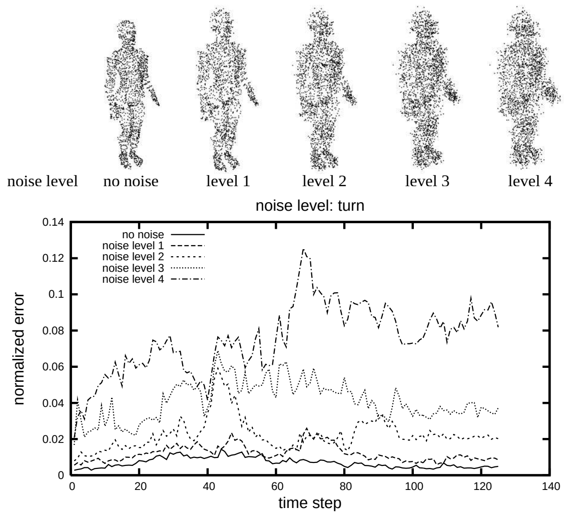


Figure 8: Tracking errors with different noise intensities.

computing PSNRs w.r.t the uncompressed data. Typical PSNR values in image compression are between 20 and 40 dB. We considered three compression levels, i.e., compression without residuals and with 50% and 100% residuals. We see that encoding residuals indeed generate 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.

Table 1: Compression quality. The "peak signal-to-noise ratio" (PSNR) is computed between rendered images of the compressed and uncompressed 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 qualify 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

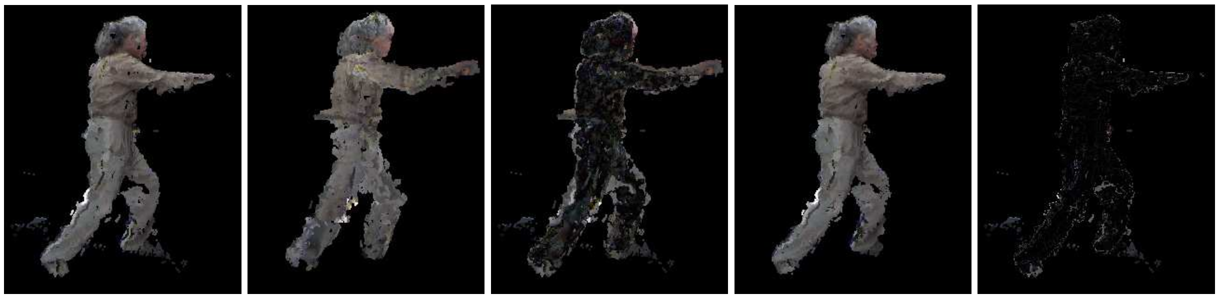


Figure 10: Reconstructions from the compressed data and their differences with the uncompressed data. (a) Uncompressed data. (b) Compressed without residuals. (c) Difference between (a) and (b). (d) Compressed with residuals. (e) Difference between (a) and (d).

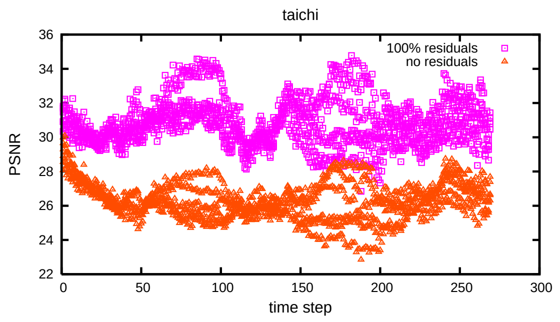


Figure 9: PSNR values from the tai-chi motion. Each point in the plot indicates a PSNR value.

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 model-driven compression method can achieve 50:1 to 5000:1 compression ratios. As we have shown earlier, our compression method can provide different compression ratio by varying the level of residuals considered during encoding. 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 model-driven 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 model driven compression method to convert point set data to a few motion parameters and a set of

Table 2: Compression ratio. Both Yang et al.'s [24] and H.264 (we use and 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. [24]	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

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$\sim$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 comparing 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 [11, 10], markerless motion capture, and Ergonomics.

References

[1] Adobe. Qicktime 7.0 h.264 implementation, 2006.

[2] J. K. Aggarwal and Q. Cai. Human motion analysis: a review. *Comput. Vis. Image Underst.*, 73(3):428–440, 1999.

[3] O. Arikan. Compression of motion capture databases. *ACM Trans. Graph.*, 25(3):890–897, 2006.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[5] K. M. Cheung, S. Baker, and T. 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.

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

[7] D. Demirdjian and T. Darrell. 3-d articulated pose tracking for untethered diectic reference. In *ICMI '02: Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 267, Washington, DC, USA, 2002. IEEE Computer Society.

[8] G. Dewaele, F. Devernay, and R. Horaud. Hand motion from 3d point trajectories and a smooth surface model. In *ECCV (1)*, pages 495–507, 2004.

[9] D. M. Gavrila. The visual analysis of human movement: a survey. *Comput. Vis. Image Underst.*, 73(1):82–98, 1999.

[10] G. Guerra-Filho and Y. Aloimonos. Human activity language: Grounding concepts with a linguistic framework. In *Proc. of the 1st International Conference on Semantics and Digital Media Technology (SAMT)*, 2006.

[11] G. Guerra-Filho, C. Fermüller, and Y. Aloimonos. Discovering a language for human activity. In *Proc. of the AAAI 2005 Fall Symposium on An-ticipatory Cognitive Embodied Systems*, 2005.

[12] S. Gumhold, Z. Karni, M. Isen-burg, and H.-P. Seidel. Predictive point-cloud compression. In *Siggraph 2005 Sketches*, 2005.

[13] L. Ibarria and J. 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*, pages 126–135, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[14] Z. Karni and C. Gotsman. Compression of soft-body animation sequences. *Computers & Graphics*, 28(1):25–34, 2004.

[15] S.-U. Kum and K. Mayer-Patel. Real-time multi-depth stream compression. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(2):128–150, 2005.

[16] J. E. Lengyel. Compression of time-dependent geometry. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 89–95, New York, NY, USA, 1999. ACM Press.

[17] W. M. Marc Alexa. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, 2000.

[18] L.-P. Morency and T. Darrell. Stereo tracking using icp and normal flow constraint. In *Proceedings of International Conference on Pattern Recognition*, 2002.

[19] T. Ochotta and D. Saupe. Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields. In *Symposium on Point-Based Graphics*, pages 103–112, 2004.

[20] 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)*, pages 145–152, 2001.

[21] R. D. 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.

[22] M. Sattler, R. Sarlette, and R. Klein. Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217, New York, NY, USA, 2005. ACM Press.

[23] D. Simon, M. Hebert, and T. 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)*, volume 3, pages 2235–2241, May 1994.

[24] Z. Yang, Y. Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. H. Campbell, and W. 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.

[25] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S. hack Jung, and R. Bajscy. Teeve: The next generation architecture for tele-immersive environment. In *ISM '05: Proceedings of the Seventh IEEE International Symposium on Multimedia*, pages 112–119, Washington, DC, USA, 2005. IEEE Computer Society.

[26] J. Zhang and C. B. Owen. Octree-based animated geometry compression. In *DCC '04: Proceedings of the Conference on Data Compression*, page 508, Washington, DC, USA, 2004. IEEE Computer Society.