Motion Capture with Ellipsoidal Skeleton using Multiple Depth Cameras

Liang Shuai, Chao Li, Xiaohu Guo, Balakrishnan Prabhakaran, and Jinxiang Chai

Abstract—This paper introduces a novel motion capturing framework which works by minimizing the fitting error between an ellipsoid based skeleton and the input point cloud data captured by multiple depth cameras. The novelty of this method comes from that it uses the ellipsoids equipped with the spherical harmonics encoded displacement and normal functions to capture the geometry details of the tracked object. This method is also integrated with a mechanism to avoid collisions of bones during the motion capturing process. The method is implemented parallelly with CUDA on GPU and has a fast running speed without dedicated code optimization. The errors of the proposed method on the data from Berkeley Multimodal Human Action Database (MHAD) are within a reasonable range compared with the ground truth results. Our experiment shows that this method succeeds on many challenging motions which are failed to be reported by Microsoft Kinect SDK and not tested by existing works. In the comparison with the state-of-art marker-less depth camera based motion tracking work our method shows advantages in both robustness and input data modality.

Index Terms—Motion Capture, Skeleton Tracking, Depth Sensing, Ellipsoidal Skeleton

1 Introduction

The motion capturing (or tracking) problem is useful in many applications such as computer animation and tele-immersion system. There are two major types of motion capturing methods, the marker based methods and marker-less methods. The marker based methods are accurate, but the equipments used are usually expensive and require special uniform with optical markers on the tracked object. The marker-less methods only utilize the data from affordable multi-view or depth cameras. Although the accuracy of this kind of methods is limited, it is popular due to its low cost and high usability.

There are two major categories of the marker-less motion tracking methods, the machine learning based and registration based methods. The machine learning based methods usually formulate the pose tracking problem as a per-pixel labeling problem, and solve it by applying some classification frameworks. Several probabilistic models have been applied, including the Gaussian Process (GP) [1], Markov Random Fields (MRFs) [2], Markov Chain Monte Carlo (MCMC) [3], and randomized decision tree [4], etc. The computation time for such kind of methods is usually short and with small variance, but the disadvantage is that without large and comprehensive training data sets these methods may fail.

The registration based methods use templates to represent the tracked objects and attempt to align the templates with the observed data by minimizing certain energy functions. The templates here can be articulated simple objects (such as cylinder or box) [5], [6], [7], or triangular meshes [8], [9], [10]. The articulated simple objects do not capture the geometry details of the tracked objects thus limit the motion tracking accuracy. The triangular meshes capture the geometry details better, but computing the deformation of triangular meshes is time-consuming. There are also methods using cylinders attached with isotropic Gaussian functions to approximate the geometry of tracked object [11], [12]. Our idea is similar to this but we use spherical harmonics functions equipped ellipsoids and experiment shows this method can approximate the geometry more precisely (see section 3.2).

In this paper a novel registration based motion capturing method is developed for multiple depth cameras (Microsoft Kinects and Kinect v2s). To overcome the limitations of both the triangular mesh model and simple articulated model described above, the proposed method uses a skeleton consists of articulated ellipsoids equipped with displacement and normal information to capture the geometry details of the tracked object. In order to deal with the occlusion problem multiple depth cameras are used to capture the full body motion data. This method only uses the depth image for motion tracking, yet still having a high accuracy in most cases. The main contributions of this work can be summarized as follows:

1) A novel ellipsoid-based skeleton is designed such that the geometry detail of the tracked object can be well captured, which makes the proposed motion capturing method accurate and robust.
2) The proposed motion tracking method only uses the depth information of the tracked object and is formulated as a pure energy optimization, without any other modalities of data or training data needed.
3) The experiment shows that the proposed method succeeds on many challenging motions which are not captured by the Kinect SDK and not tested in literatures.

- *Liang Shuai, Chao Li, Xiaohu Guo and Balakrishnan Prabhakaran are with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA.*
- *Jinxiang Chai is with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA.*
- *Corresponding Author: Xiaohu Guo, E-mail: xguo@utdallas.edu.*

2 Related Work

2.1 Motion Tracking

Recently, the field of marker-less motion capturing has made great progress, especially with the wide use of Microsoft Kinect. The motion capturing method provided in Kinect SDK [13] is a typical machine learning based method. It infers the body part to which each depth image pixel belongs based on randomized decision forests trained with a large and highly varied depth image data set. Girshick et al. [14] presented another regression forest-based method for detecting human poses from single depth or silhouette image. These machine learning based methods are highly relied on the training data.

The registration based motion tracking methods are usually based on the Iterative Closest Point (ICP) technique [5], [15], [16]. Due to its sensitivity to initial poses and proneness to local minimum, the ICP method often fails to track human motions as illustrated in [7], [17]. Wei et al. [7] introduces the Maximum A Posteriori (MAP) framework for registration and recovers the tracking failure with a method similar to [13], but their method still fails in some cases with occlusions or fast movements. Zhang et al. [17] uses triangular mesh to model the tracked object and computes the the signed distance filed (SDF) from the mesh for registration, with the help of pressure sensor data on foot to alleviate the effect of data noise on the tracking result.

There exist some optimization frameworks [6], [7] which measure the fitting errors on the projected 2D images. These methods can also take the silhouette information as an important clue for finding correspondences [18]. However, such methods usually work with a single camera thus suffer from occlusions. Ye et al. [19] presents an algorithm to recover human skeleton, body geometry and camera poses simultaneously with multiple handheld Kinects, but no complex motions are shown in the paper and the optimizing time is around 10 seconds per frame due to the computational complexity.

There are also different modalities of the source data used by various motion capturing methods, including the depth image only [20], both the color and depth images [21], and even depth image and motion sensor data [17], [22]. The more data modalities involved the tracking system can be more robust, yet more efforts need to be taken to handle the potential calibration errors between different types of data.

2.2 Skeleton Rigging and Extraction

In skeleton rigging, the goal is to change the size and pose of the given skeleton to fit with the target model [23], [24], [25]. Baran and Popović [23] described an automatic skeleton rigging method, in which the input model is approximated with medial surface and packed with spheres first, then a geometric graph is constructed for skeleton posing. In skeleton extraction, the major task is to deduce a proper topology of the skeleton for the given model, along with proper bone sizes and poses [26], [27], [28]. Lu et al. [26] proposed a method to compute the ellipsoidal bounding volumes for 3D object, thus the optimal segmentation and the skeleton of the object can be computed.

In our motion tracking framework, the initialization phase requires the skeleton to be well aligned with one frame of the input data. Currently this process is done manually, but we believe with the skeleton rigging technique like [23] this can be done automatically. The topology of our skeleton is also pre-defined since we only work with human body tracking in the experiment. But if some other objects are going to be tracked the skeleton extraction methods like [26] can also provide a fast approach to determine the skeleton topologies.

2.3 Ellipsoid and Spherical Harmonics

As introduced earlier our method uses an ellipsoidal bone based skeleton model, rather than cylinder or cube based skeletons as in traditional methods [6], [7]. One advantage of ellipsoid over cube or cylinder is that it can approximate the shape better as illustrated in some existing works [29], [30]. Another reason we choose ellipsoid as the representation of bones is that we can use spherical harmonics to encode the surface displacement and normal information for ellipsoidal bones since the ellipsoid can be easily scaled into a unit sphere, on which the spherical harmonics are defined. Similar usage of the spherical harmonics can also be found in some other research problems [31], [32].

3 Algorithms

3.1 Ellipsoidal Skeleton

In computer animation a skeleton is usually defined as a set of line segments (*bones*) and their connections (*joints*). The problem with line segment is that it cannot represent the geometry information of the corresponding body part. Since we treat the motion tracking problem as a fitting process between the skeleton and point cloud, it becomes necessary for each bone in the skeleton to capture certain geometry information such as displacement and normal. To deal with this problem we define the *ellipsoidal skeleton* $\mathcal{S} = (\mathcal{B}, \mathcal{C})$, where $\mathcal{B}$ is a collection of ellipsoids representing the bones, and $\mathcal{C}$ is a collection of vector pairs representing the joints. Before discussing the detail of how the geometry information is captured through the ellipsoidal skeleton, we firstly introduce some related definitions in the following.

The equation of an arbitrary ellipsoid in Cartesian coordinate system is:

$$\|\mathbf{SR}(\mathbf{x} - \mathbf{p})\|^2 = 1, \tag{1}$$

where $\mathbf{x}$ serves as the variable representing an arbitrary point on the ellipsoid surface, $\mathbf{p}$ is the center of the ellipsoid, $\mathbf{S}$ and $\mathbf{R}$ are 3×3 scaling and rotation matrices correspondingly. The scaling matrix $\mathbf{S}$ is determined by the axis lengths of the ellipsoid, denoted by a, b, and c:

$$\mathbf{S} = \begin{bmatrix} 1/a & & \\ & 1/b & \\ & & 1/c \end{bmatrix}. \tag{2}$$

The rotation matrix $\mathbf{R}$ gives the orientation of the ellipsoid. It represents such a rotation that aligns the ellipsoid axes of length a, b, and c with the coordinate system axes x, y, and z correspondingly.

Through Eq. (1) it is obvious that an ellipsoid can be determined by its center $\mathbf{p}$, scaling matrix $\mathbf{S}$ and rotation matrix $\mathbf{R}$. So the collection of ellipsoidal bones can be represented as:

$$\mathcal{B} = \{(\mathbf{p}_j, \mathbf{R}_j, \mathbf{S}_j) \mid \text{for each bone } j\} \tag{3}$$

The bones in our ellipsoidal skeleton are connected through *constraint vectors*. The constraint vector is a vector defined in the *local coordinate system* of the ellipsoidal bone, which is aligned with the three axes of the ellipsoid. If two bones, centered at $\mathbf{p}_1$ and $\mathbf{p}_2$ as in Fig. 1, are connected at joint $\mathbf{q}$, their constraint vectors $\bar{\mathbf{v}}_1$ and $\bar{\mathbf{v}}_2$ should point from $\mathbf{p}_1$ and $\mathbf{p}_2$ to $\mathbf{q}$ correspondingly. In other words, the two constraint vectors fulfill the following equation in the global coordinate system:

$$\mathbf{p}_1 + \mathbf{R}_1^\top \bar{\mathbf{v}}_1 = \mathbf{p}_2 + \mathbf{R}_2^\top \bar{\mathbf{v}}_2. \tag{4}$$

Here $\mathbf{R}^\top$ means to transform (rotate) the constraint vector from its local coordinate system to the global coordinate system. Defining the constraint vector in its local coordinate system makes it an invariant with respect to the position and orientation of the bone. So the joints of the ellipsoidal skeleton can be defined as a collection of constraint vector pairs:

$$\mathcal{C} = \{(\bar{\mathbf{v}}_{k,l}, \bar{\mathbf{v}}_{k,r}) \mid \text{for each joint } k \text{ connecting bone } l \text{ and } r\}. \tag{5}$$

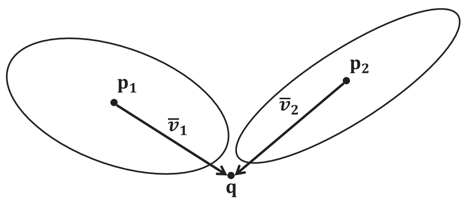


Fig. 1: Ellipsoidal skeleton: bones and constraints.

Choosing Degree of Freedom (DOF). To animate the ellipsoidal skeleton proper DOFs can be chosen for the bone centers $\{\mathbf{p}_j\}$ and rotation matrices $\{\mathbf{R}_j\}$, while keeping the bone sizes $\{\mathbf{S}_j\}$ and constraint vectors $\{(\bar{\mathbf{v}}_{k,l}, \bar{\mathbf{v}}_{k,r})\}$ fixed. The bone center $\mathbf{p}$ is a point in 3D space so it has 3 DOFs, which can be represented by its Cartesian coordinates $\mathbf{p} = (x, y, z)^\top$. A rotation in 3D space also has 3 DOFs, which can be represented by the Tait-Bryan angles $\mathbf{R} = \mathbf{R}(\alpha, \beta, \gamma)$. So the maximal DOFs of the animated ellipsoidal skeleton is the number of bones multiplied by 6.

Note that the DOFs for $\mathbf{p}$ and $\mathbf{R}$ are not independent if the bones satisfy the joint constraints Eq. (4). But in our framework the joint constraints are not strictly enforced, which means the joint positions of two connected bones do not necessarily coincide. They are allowed to be apart within a certain range if a better pose fitting can be obtained. This situation may happen, because in most practical cases the joint positions can not be determined precisely.

The rotation matrix $\mathbf{R}_j$ in the ellipsoidal bone definition Eq. (3) represents the rotation from the global coordinate system to the local coordinate system of bone j. A more reasonable way in human body animation to define the rotations of bones is through a hierarchical structure, in which the bones are organized hierarchically according to the real human body structure. In this case the rotation of each bone can be defined as the one from its parent's local coordinate system to its own local coordinate system, referred as $\mathbf{R}'_j$. If the path in the skeleton hierarchy from the root bone 0 to bone j follows $0, ..., j$, then there exists the chained rotation relationship:

$$\mathbf{R}_j = \mathbf{R}'_j \mathbf{R}'_{j-1} ... \mathbf{R}'_0. \tag{6}$$

Notice $\mathbf{R}'_0$ here means the rotation from the global coordinate system to the local coordinate system of bone 0. With this hierarchial rotation, not all the bones have 3 DOFs in human body animation according to the real human joint structure. For example, the knee only has 1 DOF. So the hierarchial definition of rotation can reduce the DOFs of the skeleton. The trade-off is that more computation is needed for the chained rotation multiplication.

3.2 Geometry Fitting

As we introduced earlier the ellipsoidal skeleton will capture the geometry information of the tracked object (usually point cloud data). The axis lengths of an ellipsoid can be adjusted to fit the shape of the corresponding part of the tracked object, and this can provide good approximations for some body parts, for example the arms. In order to provide a better approximation for the surface shape of the tracked object, we define the displacement function with spherical harmonics on the ellipsoidal bone as follows.

According to the theory of spherical harmonics, the Laplace's spherical harmonics Y_l^m form an orthonormal basis for the square-integrable functions defined on the unit sphere. In other words, any square-integrable function $f(\theta, \varphi)$ defined on the unit sphere can be expanded as a linear combination of the real part of the spherical harmonics:

$$f(\theta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} f_{lm} Y_{lm}(\theta, \varphi), \tag{7}$$

where $Y_{lm}(\theta, \varphi)$ is the real part of the spherical harmonic function of degree l and order m, and f_{lm} is the corresponding spherical harmonic coefficient.

The displacement functions can be defined with spherical harmonics on our ellipsoidal bones to better approximate the shapes of the fitted object. The displacement function is defined in the local coordinate system of the corresponding ellipsoidal bone. For an arbitrary point $\mathbf{x}_i$ on the fitted object surface, firstly it will be assigned to a particular bone j $(\mathbf{p}_j, \mathbf{R}_j, \mathbf{S}_j)$ based on certain criteria (to be introduced shortly). Then $\mathbf{x}_i$ will be transformed into the local coordinate system of bone j through a translation and a rotation:

$$\mathbf{x}'_{i,j} = \mathbf{R}_j(\mathbf{x}_i - \mathbf{p}_j). \tag{8}$$

We define the *footprint* of $\mathbf{x}_i$ on bone j as the projection of $\mathbf{x}'_{i,j}$ on the ellipsoid surface (ellipsoidal normalization):

$$\mathbf{v}_{i,j} = \frac{\mathbf{x}'_{i,j}}{\|\mathbf{S}_j \mathbf{x}'_{i,j}\|} = \frac{\mathbf{R}_j(\mathbf{x}_i - \mathbf{p}_j)}{\|\mathbf{S}_j \mathbf{R}_j(\mathbf{x}_i - \mathbf{p}_j)\|}, \tag{9}$$

and the displacement at $\mathbf{v}_{i,j}$ w.r.t. $\mathbf{x}_i$ can be defined as

$$\mathbf{d}_{i,j} = \mathbf{x}'_{i,j} - \mathbf{v}_{i,j} \tag{10}$$

The footprint $\mathbf{v}_{i,j}$ can be transformed onto the unit sphere through the scaling matrix $\mathbf{S}_j$:

$$\mathbf{u}_{i,j} = \mathbf{S}_j \mathbf{v}_{i,j} = \frac{\mathbf{S}_j \mathbf{R}_j (\mathbf{x}_i - \mathbf{p}_j)}{\|\mathbf{S}_j \mathbf{R}_j (\mathbf{x}_i - \mathbf{p}_j)\|} \tag{11}$$

If there exists a bijective mapping between the unit sphere surface and the fitted object surface, which means the displacement $\mathbf{d}_{i,j}$ is well defined on every point $\mathbf{u}_{i,j}$ on the unit sphere, then it can be expanded through the spherical harmonics

$$\mathbf{d}_{i,j}(\mathbf{u}_{i,j}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \bar{\mathbf{d}}_{lm} Y_{lm}(\mathbf{u}_{i,j}) \tag{12}$$

Note the displacement $\mathbf{d}_{i,j}$ is a 3×1 vector so we need 3 sets of spherical harmonic coefficients for the expansion:

$$\bar{\mathbf{d}}_{lm} = (\bar{d}_{lm,x}, \bar{d}_{lm,y}, \bar{d}_{lm,z})^{\top}. \tag{13}$$

In a similar way, the normal on the object surface can also be represented as a linear combination of the spherical harmonics. Suppose the normal at point $\mathbf{x}_i$ is $\mathbf{n}_i$ and $\mathbf{x}_i$ is assigned to bone j. The normal $\mathbf{n}_i$ will be transformed into the local coordinate system of bone j through rotation matrix $\mathbf{R}_j$, then expanded with spherical harmonics:

$$\mathbf{n}_{i,j}(\mathbf{u}_{i,j}) = \mathbf{R}_j \mathbf{n}_i = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \bar{\mathbf{n}}_{lm} Y_{lm}(\mathbf{u}_{i,j}), \tag{14}$$

where the spherical harmonic coefficient $\bar{\mathbf{n}}_{lm}$ is also a 3×1 vector:

$$\bar{\mathbf{n}}_{lm} = (\bar{n}_{lm,x}, \bar{n}_{lm,y}, \bar{n}_{lm,z})^{\top}. \tag{15}$$

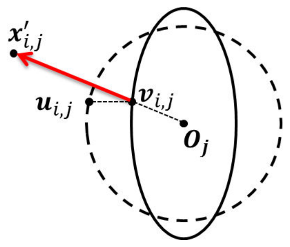


Fig. 2: Displacement at footprint $\mathbf{v}_{i,j}$ on the ellipsoid centered at $\mathbf{O}_j$. $\mathbf{x}'_{i,j}$ is the local coordinate of data point $\mathbf{x}_i$, and $\mathbf{u}_{i,j}$ is the mapping of $\mathbf{v}_{i,j}$ on the unit sphere.

To evaluate the displacement and normal functions, i.e., Eq. (12) and Eq. (14), the spherical harmonic coefficients $\bar{\mathbf{d}}_{lm}$ and $\bar{\mathbf{n}}_{lm}$ need to be determined. With some sampling points on the fitted object surface, $\bar{\mathbf{d}}_{lm}$ and $\bar{\mathbf{n}}_{lm}$ can be obtained by minimizing the following energies for each bone j:

$$E_{d,j}(\bar{\mathbf{d}}_{lm}) = \sum_{i=1}^{N_j} \|\mathbf{x}'_{i,j} - \mathbf{v}_{i,j} - \sum_{l=0}^{L} \sum_{m=-l}^{l} \bar{\mathbf{d}}_{lm} Y_{lm}(\mathbf{u}_{i,j})\|^2 \tag{16}$$

$$E_{n,j}(\bar{\mathbf{n}}_{lm}) = \sum_{i=1}^{N_j} \|\mathbf{R}_j \mathbf{n}_i - \sum_{l=0}^{L} \sum_{m=-l}^{l} \bar{\mathbf{n}}_{lm} Y_{lm}(\mathbf{u}_{i,j})\|^2, \tag{17}$$

where N_j is the number of sampling points assigned to bone j, and L is the maximum degree used for spherical harmonics. The higher the degree of spherical harmonics l used, the better the targeting function is approximated, but the more computation time and storage are needed. With experiments we find $L = 6$ is a good choice for both fitting quality and computation resource. The geometry fitting result of a 3D human model, which is sampled by selecting its mesh vertices, is shown in Fig. 3.

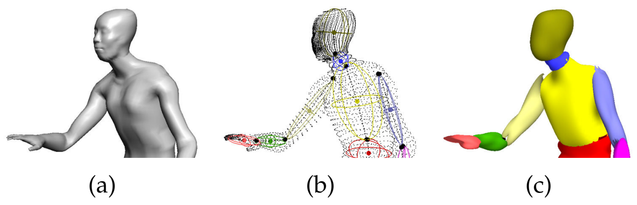


Fig. 3: Use spherical harmonics encoded displacement function to fit the shape of a 3D human model. (a) 3D model. (b) Sampling points and ellipsoidal skeleton. (c) Spherical harmonics based displacement on the ellipsoidal surface of each bone.

3.3 Point Cloud Segmentation

In the motion tracking problem the point cloud segmentation (or labeling) is a challenge problem. But if we have already obtained a well aligned ellipsoidal skeleton along with the spherical harmonics represented displacement and normal functions, the segmentation can be easily achieved through partitioning with nearest distance. Given a point $\mathbf{x}_i$ that needs to be labeled, the corresponding point on the displaced surface of bone j can be found as:

$$\hat{\mathbf{x}}_{i,j} = \mathbf{v}_{i,j} + \mathbf{d}_{i,j}, \tag{18}$$

where the footprint $\mathbf{v}_{i,j}$ and displacement $\mathbf{d}_{i,j}$ are computed through Eq. (9) and Eq. (12). Then the distance from $\mathbf{x}_i$ to bone j can be defined as the Euclidean distance between $\mathbf{x}_i$ and the transformed $\hat{\mathbf{x}}_{i,j}$ in the global coordinate system:

$$d_{seg}(\mathbf{x}_i, j) = \|\mathbf{x}_i - (\mathbf{R}_j^{\top} \hat{\mathbf{x}}_{i,j} + \mathbf{p}_j)\|, \tag{19}$$

and $\mathbf{x}_i$ will be assigned to the nearest bone measured by d_{seg}

$$label(\mathbf{x}_i) := \arg\min_j d_{seg}(\mathbf{x}_i, j). \tag{20}$$

In the motion tracking process the skeleton is not guaranteed to be well aligned with the tracked object, so the nearest bone of a point in the data set may not be always the correct one. In this case we will use the normal information to assist with the segmentation. Specifically, for each point $\mathbf{x}_i$ the normal at its corresponding point on the displaced ellipsoidal bone surface can be computed through the spherical harmonics represented normal function Eq. (14):

$$\hat{\mathbf{n}}_i = \mathbf{R}_j^{\top} \sum_{l=0}^{L} \sum_{m=-l}^{l} \bar{\mathbf{n}}_{lm} Y_{lm}(\mathbf{u}_{i,j}). \tag{21}$$

If the direction of $\hat{\mathbf{n}}_i$ differs too much with $\mathbf{n}_i$, which is the normal at $\mathbf{x}_i$, then $\mathbf{x}_i$ will not be assigned to bone j. The difference can be easily measured with the dot product between $\hat{\mathbf{n}}_i$ and $\mathbf{n}_i$.

Additionally, in the geometry fitting process described in Sec. 3.2, the initial input point cloud also needs to be segmented. The spherical harmonics coefficients for the displacement and normal functions are not available yet at that time. In this case we just set the displacements to zero vectors and do not use normal information for segmentation assistance, which means only the distance from the point to its footprint is used. To obtain a good segmentation we may manually adjust the pose of skeleton to let it align with the point cloud well in the initialization stage.

The point cloud segmentation procedure can be summarized as the following parallel algorithm.

Algorithm 1 Point Cloud Segmentation

```
Require: The skeleton $\mathcal{S} = (\mathcal{B}, \mathcal{C})$ is well aligned with point cloud $\{(\mathbf{x}_i, \mathbf{n}_i)|\forall i\}$.
Ensure: $label(\mathbf{x}_i)$ is a proper labeling for each $\mathbf{x}_i$.
 1: for each point $\mathbf{x}_i$ parallelly do
 2:     $d \leftarrow \infty$;
 3:     $label(\mathbf{x}_i) \leftarrow null$;
 4:     for each bone $j$ do
 5:         $\mathbf{n}_j \leftarrow \hat{\mathbf{n}}_i$ as in Eq. (21);
 6:         $d_j \leftarrow d_{seg}(\mathbf{x}_i, j)$ as in Eq. (19)
 7:         if $\mathbf{n}_i \cdot \mathbf{n}_j >$ certain threshold and $d_j < d$ then
 8:             $d \leftarrow d_j$;
 9:             $label(\mathbf{x}_i) \leftarrow j$;
10:         end if
11:     end for
12: end for
```

3.4 Pose Fitting

After segmenting the point cloud according to an initial skeleton configuration with Algorithm 1, the next step is to update the initial skeleton to let it fit with the pose of the input data. The fitting error can be measured by the summation of distances from the input points to their corresponding points on the displaced ellipsoid surfaces. Following our above notations the following energy term is defined to measure the distance based fitting error:

$$E_{dist}(\mathbf{p}_j, \mathbf{R}_j) = \frac{w_1}{N}\sum_{i=1}^{N}\|\mathbf{S}_j(\mathbf{x}'_{i,j} - \mathbf{v}_{i,j} - \sum_{l=0}^{L}\sum_{m=-l}^{l}\bar{\mathbf{d}}_{lm}Y_{lm})\|^2, \tag{22}$$

where w_1 is the weight for the distance energy term, N is the number of points in input data, and $j = label(\mathbf{x}_i)$ is the assigned bone for point $\mathbf{x}_i$. This energy term is similar to Eq. (16), but the difference is the optimizing variables here are the bone centers $\mathbf{p}_j$ and orientations $\mathbf{R}_j$. Another difference is the distance is scaled by $\mathbf{S}_j$. The experiment in Sec. 5.1 will show that by applying the scaling matrix $\mathbf{S}_j$ on the energy term the small pose changing like twisting along arms can be accurately captured. In our experiment the distance is measured in meters, and we choose $w_1 = 10$ so that the optimized energy is around 1.

The energy term above only considers the fitting error of each bone individually. The bone centers and orientations should also fulfill the joint constraints in Eq. (4). As discussed above, in our skeleton the joints are defined by constraint vector pairs Eq. (5) and the joint constraint is not strictly enforced, which allows the pose to be fitted more accurately. The energy term for the joint constraint fitting error is defined as follows:

$$E_{cons}(\mathbf{p}_j, \mathbf{R}_j) = \frac{w_2}{M'}\sum_{k=1}^{M'}\|\mathbf{p}_{k,l} + \mathbf{R}_{k,l}^{\top}\bar{\mathbf{v}}_{k,l} - \mathbf{p}_{k,r} - \mathbf{R}_{k,r}^{\top}\bar{\mathbf{v}}_{k,r}\|^2, \tag{23}$$

where w_2 is the weight for constraint energy term and M' is the number of joints. Other notations follow the skeleton definition in Eq. (3) and Eq. (5). In the experiment we set $w_2 = 1000$ initially and increase it gradually to 5000 so that it is comparable to the first energy term and does not constrain the bone movement too much, especially in the first iteration.

In practice we find that the fitted skeleton pose falls into local minimal in some cases where some bones collide with each other. To deal with this problem we define another energy term to avoid the bone collisions. Suppose we want to avoid the collision between bones j and k. Firstly the surface of ellipsoidal bone j is sampled into a point set $\{\mathbf{y}_i \mid i = 1, ..., N'_j\}$. If there is a point $\mathbf{y}_i$ whose distance to $\mathbf{p}_k$ is less than the radius of ellipsoid k, it can be concluded that these two bones are collided. To eliminate the anisotropy of the radius of an ellipsoid, we can transform the ellipsoid into unit sphere with the rotation and scaling matrices of the corresponding ellipsoid for distance computing. Because $\{\mathbf{y}_i\}$ is sampled in the local coordinate system of bone j, we also need to transform it into the global coordinate system before computing the distance. So the energy term for avoiding collision can be defined as:

$$E_{col}(\mathbf{p}_j, \mathbf{R}_j) = \frac{w_3}{N'}\sum_{(j,k)}\sum_{i=1}^{N'_j} e^{s(r - \|\mathbf{S}_k\mathbf{R}_k(\mathbf{R}_j^{\top}\mathbf{S}_j^{-1}\mathbf{y}_i + \mathbf{p}_j - \mathbf{p}_k)\|^2)}, \tag{24}$$

where w_3 is the weight for this energy term and N' is the number of all sampling points. The parameter s is the attenuation coefficient of the exponentiation and r is the threshold radius. If the scaled distance from $\mathbf{y}_i$ to $\mathbf{p}_k$ is less than r, then the energy will become very large. In our experiment we set $w_3 = 0.1$ and $s = 5$, and r ranges from 0.5 to 1.5, depending on the bone pairs we are handling.

Suppose there are M bones in total. The optimal pose $\{(\mathbf{p}_j, \mathbf{R}_j)|j = 1...M\}$ with respect to a given segmented point cloud can be solved by optimizing the combined energy of all the three energy terms defined above:

$$E_{pose} = E_{dist} + E_{cons} + E_{col}. \tag{25}$$

This is a non-linear optimization problem so we solve it with the parallel L-BFGS-B algorithm [33], which is a bounded non-linear optimization algorithm. Despite its fast convergence, the primary reason we choose L-BFGS-B algorithm is because with this bounded algorithm we can set boundaries for the rotation angles $(\alpha_j, \beta_j, \gamma_j)$ of each bone according to real human joint limit during optimization, which increases the robustness of our method.

3.5 Motion Tracking

The point cloud segmentation and pose fitting procedures are both relied on each other. So we use the Lloyd's algorithm to do this two procedures iteratively for the motion tracking purpose. Our algorithm starts with an initial skeleton pose configuration, according to which the input point cloud is segmented. Then the pose of the skeleton is updated with respect to the segmentation result, and the updated skeleton will be used for next iteration. Since each step in the iteration will not increase the energy, the convergence of Lloyd's process is guaranteed.

The optimized skeleton pose for the current frame of data will be used as the initial configuration for the next frame. To improve the robustness and accelerate the convergence, we use a heuristic algorithm to adjust the position of the initial skeleton before optimization. Suppose the centroids of current and previous data frames are $\mathbf{c}_t$ and $\mathbf{c}_{t-1}$, then the initial skeleton will be translated by $(\mathbf{c}_t - \mathbf{c}_{t-1})$ so it is closer to the current frame of point cloud data.

The motion tracking process with our ellipsoidal skeleton is described as the following Algorithm 2.

Algorithm 2 Motion Tracking

```
Require: The skeleton $\mathcal{S} = (\mathcal{B}, \mathcal{C})$ is close to the initial frame of data.
Ensure: The skeleton $\mathcal{S}_t$ captures the pose of each frame $t$.
 1: $t \leftarrow 0$;
 2: $\mathcal{S}_0 \leftarrow \mathcal{S}$;
 3: while data of frame $t$ exists do
 4:     $\{(\mathbf{x}_i, \mathbf{n}_i) | i = 1...N\} \leftarrow$ data of frame $t$;
 5:     $\mathbf{c}_t \leftarrow$ centroid of frame $t$;
 6:     if $t > 0$ then
 7:         $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$;
 8:         $\mathbf{c}_t \leftarrow$ centroid of frame $t$;
 9:         translate $\mathcal{S}_t$ by $(\mathbf{c}_t - \mathbf{c}_{t-1})$;
10:     end if
11:     while not converged do
12:         run Algorithm 1 with $\mathcal{S}_t$ and $\{(\mathbf{x}_i, \mathbf{n}_i) | i = 1...N\}$;
13:         $\{label(\mathbf{x}_i) | i = 1...N\} \leftarrow$ segmentation result;
14:         minimize energy $E_{pose}$ with L-BFGS-B;
15:         $\{(\mathbf{p}_j, \mathbf{R}_j) | j = 1...M\} \leftarrow$ optimized pose;
16:         update $\mathcal{S}_t$ with $\{(\mathbf{p}_j, \mathbf{R}_j) | j = 1...M\}$;
17:     end while
18:     $t \leftarrow t + 1$.
19: end while
```

4 Experiment

4.1 Algorithm Implementation

To reduce the running time our algorithms are all implemented parallelly with Nvidia CUDA and running on GPU. The point cloud segmentation algorithm (Algorithm 1) is already a parallel algorithm, so in the implementation we just assign one CUDA thread for each point $\mathbf{x}_i$ to do the computation. Both of the geometry fitting and pose fitting procedures are energy optimization problems. As mentioned before we use the L-BFGS-B algorithm to solve the energy optimization problem. With an existing GPU implementation of the L-BFGS-B algorithm [33], we only need to compute the energy value and gradient of the energy function and feed them to the L-BFGS-B algorithm implementation.

The energy functions we are optimizing (Eqs. (16), (17), (22), (23) and (24)) are all in the form of small energy term summation. In the parallel implementation one CUDA thread is used to compute the energy value or gradient for one small energy term. To add all the energy value elements a *parallel reduction* algorithm is used to utilize the parallel computing resource. Adding the energy gradient elements is a little complicated because only the gradient elements of the same variable should be added, which leads to the *regional reduction* problem: given $\{(x_i, y_i) \mid i = 1, ..., N, y_i \leq M, y_i \in \mathbb{N}, M \leq N\}$, find $\{S_j \mid S_j = \sum_{y_i=j} x_i\}$. In this problem each element x_i (e.g., some value of a point) is associated with the region y_i (e.g., the label of a bone), and all $\{x_i\}$ within the same region need to be summated. The regional reduction problem can be efficiently solved with the help of the *parallel prefix sum* algorithm [34] as follows:

Algorithm 3 Parallel Regional Reduction

```
Require: $\{(x_i, y_i) \mid i = 1, ..., N, y_i \leq M, y_i \in \mathbb{N}, M \leq N\}$.
Ensure: $\{S_j \mid S_j = \sum_{y_i=j} x_i\}$.
 1: allocate array $flags$, $indices$ of size $M \times N$;
 2: allocate array $x'$ of size $N$;
 3: for $k \leftarrow 1, ..., M \times N$ parallelly do
 4:     $i \leftarrow k\%N$;
 5:     $flags[k] \leftarrow 1$ if $y_i = k/N$, else 0;
 6: end for
 7: $indices \leftarrow$ parallel inclusive prefix sum of $flags$;
 8: for $i \leftarrow 1, ..., N$ parallelly do
 9:     $j \leftarrow indices[y_i \times N + i]$;
10:     $x'[j] \leftarrow x_i$;
11: end for
12: for $j \leftarrow 1, ..., M$ parallelly do
13:     $a \leftarrow 1$ if $j = 1$, else $indices[(j-1) \times N] + 1$;
14:     $b \leftarrow indices[j \times N]$;
15:     if $a \leq b$ then
16:         $S_j \leftarrow$ parallel reduction of sub-array $x'[a : b]$;
17:     end if
18: end for
```

Algorithm 3 computes new positions of the input elements regarding to their regions with the prefix sum algorithm first. Then the input elements are shuffled according to the computed positions so that the elements from the same region are continuous in the shuffled array. Finally a parallel reduction is performed for each region of the shuffled array. In Algorithm 3 the inclusive prefix sum is used because the arrays begin from the index of 1. In practice an array usually starts from the index of 0 so the exclusive prefix sum is more convenient to use.

4.2 Input Data

The motion tracking framework we proposed works with the point cloud data of human body, which is usually captured through depth cameras like Microsoft Kinect [35]. To fully evaluate our method both the Kinect and Kinect v2 are used for data capturing in the experiments. The Berkeley Multimodal Human Action Database (MHAD) [36] contains not only the Kinect depth images but also motion tracking marker positions so it can be used as ground truth to evaluate our method. We also use the point cloud sampled from 3D models to test our method against the interference of synthesized scanning noise.

Kinect Data. Microsoft Kinect can stream color and depth images of the scene at about 30 FPS. Our method only uses the depth image, which is equivalent to point cloud data. To segment a person out from the background of the scene we simply use the player index provided by Kinect SDK. A single Kinect only gives partial sampling points of a human body. To capture the complete body data we use multiple Kinects simultaneously. But the problem raised with multiple Kinects is that an extrinsic calibration is needed to align the point cloud data from different Kinects. We use the method described by Auvinet et al. [37] for the extrinsic calibration of multiple Kinect cameras. Another problem with Kinect captured data is that it does not include the normal information. However, the normal at each point can be simply computed as the cross product of vectors from that point to its neighbors on the depth image. In order to get smoothed normal across the surface, the depth image will be smoothed before normal computation using the *bilateral filtering* [38]. The advantage of bilateral filtering is that its edge-preserving feature will keep the silhouette of human body.

Berkeley MHAD. The Berkeley MHAD [36] provides multi-modality data of basic human actions, including optical marker based motion capture data, multi-view video data, color and depth image data (from 2 Kinects), accelerometer data as well as audio data. All the data provided in this database are geometrically calibrated and temporally synchronized, although the errors in calibration and synchronization still exist due to the hardware limitation. We use the depth image data from 2 Kinects as input of our system and the motion capture data for evaluation of our tracking result.

Synthesized Point Cloud from Animated 3D Models. We use some existing methods to generate synthetic point cloud data from animated 3D models. First we get a rigged skeleton for an input human body model using the method described by Baran and Popović [23]. Then we map the rotations and translations in the skeleton animation files found in CMU Motion Capture Database [39] to the rigged skeleton. With the mapped motion sequence of skeleton the 3D model can be animated using the classic skinning animation method. Finally we sample the animated 3D models into point coulds with artificial scanning noises described by Berger et al. [40]. The advantage of this sampling method is that it can simulate the scanning noises in a controlled manner.

4.3 Skeleton Definition for Human Body

The ellipsoidal skeleton used for human motion tracking consists of 17 bones (shown in Fig. 4a). As seen in the figure initially the orientation of the local coordinate system of each ellipsoidal bone is aligned with the global coordinate system, which is the same as the Kinect camera coordinate system (i.e. centered at the camera and having $z+$ direction towards the shooting direction). Notice in Figure 4a the skeleton is facing the camera.

The orientation of each bone in our ellipsoidal skeleton is measured as the rotation of the bone from its initial pose, which is illustrated in Figure 4a, to its current pose. In our framework *Tait-Bryan angles* are adopted to represent

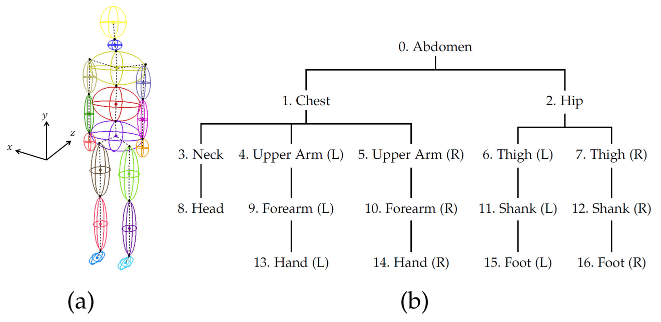


Fig. 4: (a) Ellipsoidal skeleton used in our motion tracking experiments. (b) Rotation hierarchy of human body skeleton.

the rotations in 3D space. Tait-Bryan angles are essentially 3 rotation angles with respect to the 3 axes in Cartesian coordinate system. There are different rotation conventions for Tait-Bryan angles due to the different orders of rotation axes and the choices of either fixed or rotating coordinate system (which is called extrinsic or intrinsic convention). In this paper the Tait-Bryan angles of each bone about x, y, and z axes are defined as α, β, and γ, and the extrinsic x-y-z rotation convention is used. Notice the extrinsic x-y-z convention is equivalent to the intrinsic z-y'-x'' convention, where y' and x'' represent the rotated axes. As discussed earlier the rotations of bones in our ellipsoidal skeleton are defined in a hierarchical structure, such that we can set boundaries for the rotation angles according to real human joint structures during the L-BFGS-B optimization. The rotation hierarchy of human body skeleton used in our experiment is rooted at abdomen and spread to limbs, as illustrated in Figure 4b. With the initial skeleton pose and coordinate system configuration shown in Figure 4a, the boundaries of rotation angles for each bone used in our experiment are listed in Table 1.

TABLE 1: Rotation angle boundaries of human body skeleton.

Bone(s)	α	β	γ
Abdomen	–	–	–
Chest	$[-\pi/2, \pi/4]$	$[-\pi/8, \pi/8]$	$[-\pi/8, \pi/8]$
Hip	$[-\pi/4, \pi/2]$	$[-\pi/8, \pi/8]$	$[-\pi/8, \pi/8]$
Neck	$[-\pi/8, \pi/8]$	$[-\pi/4, \pi/4]$	0
Head	$[-\pi/2, \pi/2]$	$[-\pi/4, \pi/4]$	$[-\pi/4, \pi/4]$
Left Upper Arm	$[-\pi/2, \pi]$	$[-\pi/2, 0]$	$[-\pi, \pi/4]$
Right Upper Arm	$[-\pi/2, \pi]$	$[0, \pi/2]$	$[-\pi/4, \pi]$
Left Forearm	$[0, \pi]$	$[-\pi, 0]$	0
Right Forearm	$[0, \pi]$	$[0, \pi]$	0
Hands	$[-\pi/2, \pi/2]$	0	$[-\pi/4, \pi/4]]$
Left Thigh	$[-\pi/4, \pi]$	$[-\pi/4, \pi/4]$	$[-\pi, \pi/4]$
Right Thigh	$[-\pi/4, \pi]$	$[-\pi/4, \pi/4]$	$[-\pi/4, \pi]$
Shanks	$[-\pi, 0]$	0	0
Feet	$[-\pi/4, \pi/4]$	$[-\pi/8, \pi/8]$	0

4.4 Working Flow

The working flow of our motion tracking experiment consists of two phases: geometry fitting and motion tracking, as described in Sec. 3.2 and Sec. 3.5.

In the geometry fitting phase we capture a T-pose of the subject, then manually adjust the size and orientation of each bone in the skeleton according to the input data. Finally we compute the spherical harmonic coefficients for the surface displacements and normals (if applicable) by optimizing the corresponding energies in Eq. (16) and Eq. (17).

In the motion tracking phase, we capture depth images for different actions of the subject. The number of depth cameras we used is 4 for Kinect, and 3 for Kinect v2. Then the depth images are calibrated, synchronized, and merged into point and normal data as described in Sec. 4.2. Finally we run the motion tracking algorithm (Algorithm 2) with the merged data and use the initial skeleton from previous phase as input.

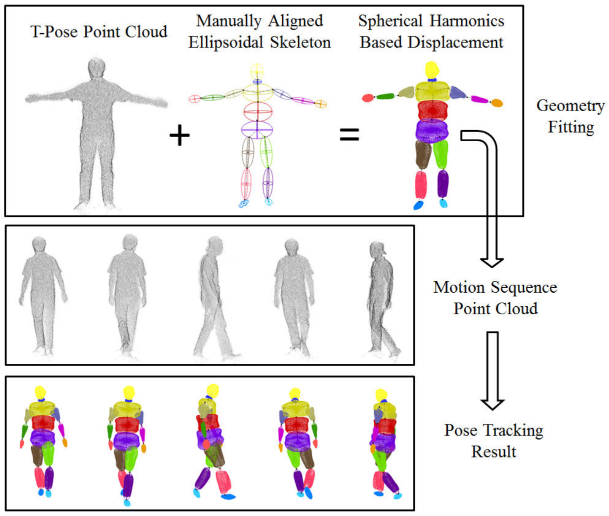

T-Pose Point Cloud
Manually Aligned Ellipsoidal Skeleton
Spherical Harmonics Based Displacement
+
=
Geometry Fitting
Motion Sequence Point Cloud
Pose Tracking Result

Fig. 5: Working flow of our proposed framework.

5 Evaluation

5.1 Different Tracking Schemas

In the proposed pose tracking framework we introduced several strategies to increase the accuracy and robustness of the tracking process, including the spherical harmonics encoded displacement function (Eq. (12)), the rotation angle boundary (Table 1), the scaled distance energy term (Eq. (22)), and the collision avoiding energy term (Eq. (24)). In this section we will illustrate the effectiveness of these strategies with experiment results on 3D model data and Kinect data.

Spherical Harmonics Displacement (SHD). The spherical harmonics encoded displacement functions on the ellipsoidal bones can capture the geometry of the tracked object, thus can produce a more accurate tracking result. Figs. 6b and 6c show the head part of the tracking results on a 3D model (as in Fig. 6a) with and without SHD (which means zero displacement thus tracking with ellipsoids) respectively, while Figs. 6e and 6f show the ellipsoidal bones as boxes for better visualization of the bone orientation. We can see that with SHD the orientation of head is far more accurate than the tracking schema without SHD.

Scaled Distance Energy and Rotation Angle Boundary. While defining the energy term for distance based fitting error (Eq. (22)), a scaling matrix $\mathbf{S}_j$ is multiplied to the distance between the data point and its corresponding point on the displaced ellipsoid surface. The scaling matrix $\mathbf{S}_j$ is exactly the one in the definition of the ellipsoidal bone (Eq. (3)), so the distance error can be normalized based on the size of each ellipsoidal bone. The normalization makes the energy more sensitive to rotations so some small pose-changing like arm-twisting can be captured in our method, as shown in Fig. 7. Notice in Fig. 7 the bones of our ellipsoidal skeleton are rendered as boxes which are aligned with the axis directions of their corresponding bones, such that the orientation of each bone can be seen clearly. Figs. 7b and 7c show the difference of the tracking result without and with applying the scaling matrix on a key frame of the arm-twisting motion. It is obvious in Fig. 7c that with the scaling matrix the rotations of hands are correctly captured. Furthermore, we also propose to use the rotation angle boundary to produce reasonable orientations of bones according to real human body structure (Sec. 4.3). Fig. 7d illustrates the effectiveness of this strategy. According to human joint structure there is no relative twisting between the forearm and hand, so we remove the DOF around y axis of hand by setting the corresponding angle boundary to $[0, 0]$ (Table 1). Comparing with Fig. 7c (which does not set the rotation angle boundary), the rotations of forearms are also correctly captured, as shown in Figure 7d.

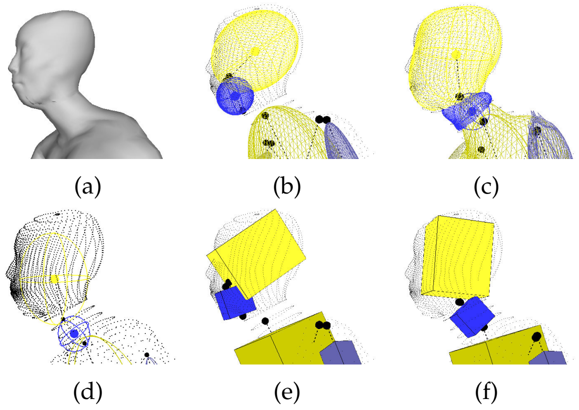


Fig. 6: Tracking with/without spherical harmonic based displacement (SHD): (a) 3D model; (d) initial pose; (b,e) result without SHD; (c,f) result with SHD. In (e,f) the ellipsoidal bones are shown as boxes for better visualization of their orientation.

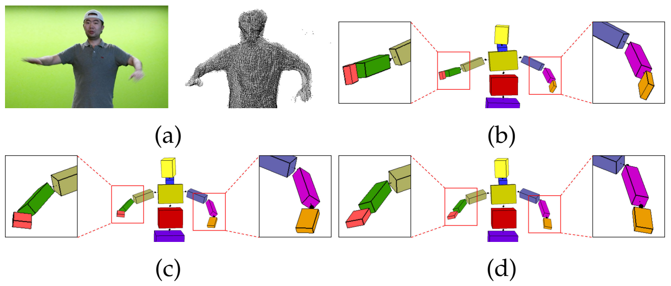


Fig. 7: Tracking results on arm-twisting with different choices of distance error energy term and rotation angle boundary. (a) The color image and input point cloud. Tracking result using the energy term (b) without scaling matrix $\mathbf{S}_j$; (c) with scaling matrix $\mathbf{S}_j$; (d) with scaling matrix $\mathbf{S}_j$ plus rotation angle boundary.

Collision Avoidance Energy. We also proposed a collision avoidance energy term (Eq. (24)) to prevent the optimization from falling into local minimum. In a successful pose tracking result there should be no collisions between bones. On the other hand, in our experiment we also find in a failed tracking result there is usually a collision between the bones. So we propose the collision avoidance energy term to increase the robustness of our pose tracking method. Fig. 8 shows the difference of tracking results without and with the collision avoidance energy term on a frame of the walking motion data captured by Kinect. We can see that with collision avoidance the legs of the person in Fig. 8c does not cross over as in the failure result (Fig. 8b).

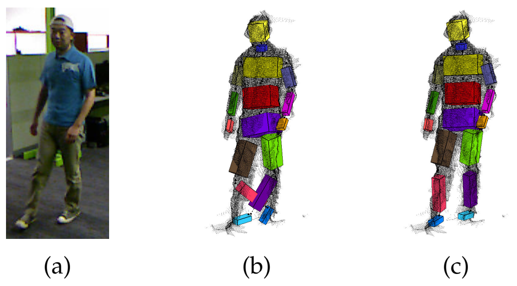


Fig. 8: Tracking result (b) without, and (c) with collision avoidance energy term on a frame of the walking motion data captured by Kinect.

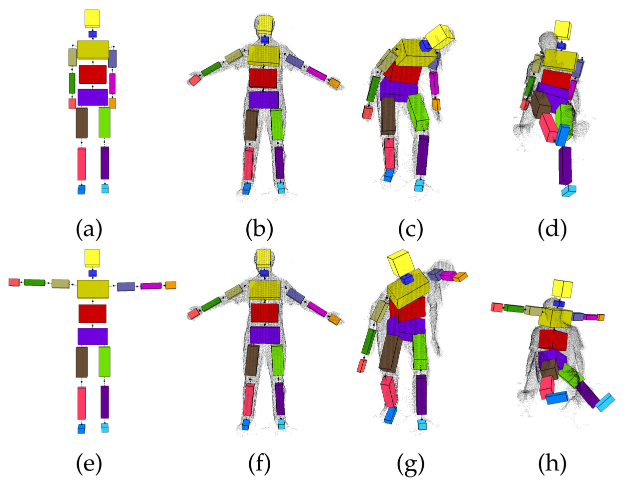


Fig. 9: Tracking with two different initial poses (a) and (e): (b,c,d) are computed from initialization (a); (f,g,h) are computed from initialization (e).

TABLE 2: Running-time for results in Fig. 9.

Initial Pose	Input Frame	Optimized Pose	Lloyd Iter. #	L-BFGS-B Iter. #	Time (ms)
Fig. 9a	#0	Fig. 9b	7	37	305
	#50	Fig. 9c	9	208	1259
	#100	Fig. 9d (Failed)	10	66	672
Fig. 9e	#0	Fig. 9f	2	18	121
	#50	Fig. 9g (Failed)	6	141	805
	#100	Fig. 9h (Failed)	6	68	590

5.2 Different Initializations

Same as other energy optimization based methods our pose tracking will fall into local minimum from a bad initialization. So in the normal tracking process we assume the initial pose of the skeleton does not differ too much with the tracked object. Since our method adopts several strategies to increase the accuracy and robustness, such as the ellipsoidal bone displacement and collision avoiding energy term, it is expected to produce a successful result from some challenging initial configurations. Fig. 9 shows some tracing results from two different initial poses: (1) the initial pose with all rotation angles set to zero (Fig. 9a); and (2) a T-pose initialization (Fig. 9e). The tracking results on a standing pose with arms opened are all successful for both of the initial poses (Fig. 9b and Fig. 9f), but the running time for T-pose initialization is less since it is closer to the targeting pose (as in Table 2). For the bended pose the first initial pose takes a quite long time to converge, and the tracking result is successful (Fig. 9c). But starting with the second T-pose our method fails to converge to a correct result (Fig. 9g). Both of the initial poses fail to converge to a successful result for the sat down pose, as shown in Figs. 9d and 9h, since the initial poses differ too much from the targeting shape. All the running times of the tracking results in Fig. 9 are listed in Table 2.

5.3 Tracking Accuracy

We use the Berkeley MHAD [36] to evaluate the accuracy of our pose tracking method. For each experiment subject (person) the MHAD has recorded the positions of 43 optical markers, which are served as the ground truth for our tracking error measurement. The tracking error is computed as the distance between the MHAD marker position and the computed marker position from our tracked skeleton. To extract the marker positions from the tracked skeleton, we firstly assign the MHAD markers to their corresponding bones in our human body ellipsoidal skeleton (Fig. 4a), as shown in Table 3. Notice no marker is assigned to neck due to the marker configuration in MHAD. Then we compute the coordinates of the MHAD markers in the local coordinate systems of their corresponding ellipsoidal bones with the positions in a standard frame of data (usually the initial frame). Finally the marker positions for our tracked skeleton can be computed by transforming the local coordinates of the markers into the global coordinate system for each frame of data. The basic assumption for the marker position extraction is that each marker is relatively fixed on its assigned bone, but in reality this is not true due to the deformable nature of human body. So the measured error in this way is not absolutely precise.

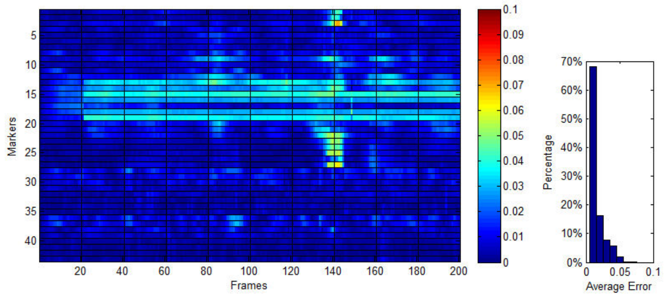
TABLE 3: Correspondence between MHAD markers and the bones of our ellipsoidal skeleton for human body.

Bone	MHAD Marker IDs	Bone	MHAD Marker IDs
Abdomen	4, 7, 8, 11	Left Hand	17, 18, 19
Chest	5, 6, 9, 10, 12, 20	Right Hand	25, 26, 27
Hip	28, 29, 36, 37	Left Thigh	30, 31
Head	1, 2, 3	Right Thigh	38, 39
Left Upper Arm	13, 14	Left Shank	32, 33
Right Upper Arm	21, 22	Right Shank	40, 41
Left Forearm	15, 16	Left Foot	34, 35
Right Forearm	23, 24	Right Foot	42, 43

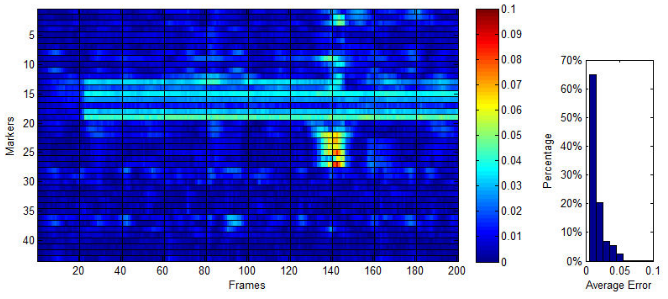
In order to investigate the influence of some external factors on the tracking accuracy, including the calibration error between multiple depth cameras and the quality of the input data, we also use some synthetic data from 3D models for the tracking accuracy measuring experiment before using the data from MHAD. The synthetic data is obtained by sampling an animated sequence of a 3D model into point cloud data. One problem is that there is no marker information on 3D model data, but we can produce some virtual markers by manually selecting some vertices from the model. In the experiment we pick 43 markers with the same positioning configuration as in the MHAD.

5.3.1 Result on Synthesized Point Cloud from 3D Models

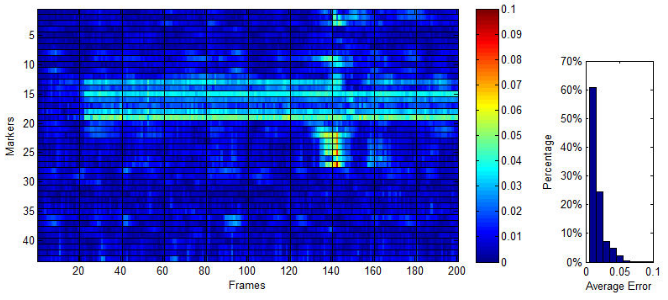
In the synthetic data testing we use the point cloud data sampled from an animated dancing model with a sequence of 200 frames. The model size is normalized according to the height of the model, so the measured distance error is also normalized. Fig. 10a shows that over 80% of the tracking errors are below 0.02 for the prefect dancing model data.



(a) Perfect dancing model data.



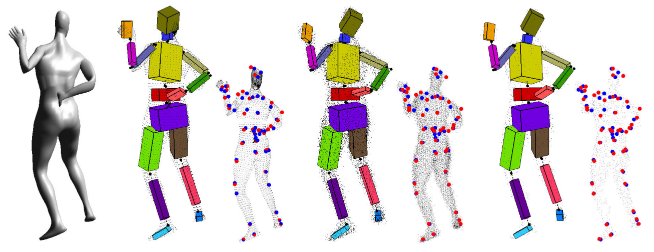
(b) Noisy dancing model data.



(c) Noisy and incomplete dancing model data.

Fig. 10: Normalized fitting errors on dancing model data.

From the error color map in Fig. 10a we notice that larger errors occur around frame 143 at markers on head, right arm, and right hand. By visualizing the input model of frame 143 in Fig. 11a, we can see the head is strongly deformed and the right hand goes inside the body. So the tracking errors at these parts are mainly caused by the geometry changing of the input data. Even though the tracking result in Fig. 11b still looks fine.



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

Fig. 11: Input and tracking results on frame 143 of dancing model data. (a) Input model. The tracked pose, manually-selected markers (blue) and our tracked markers (red) for: (b) perfect, (c) noisy, and (d) noisy and incomplete dancing model data.

We also synthesize some scanned data of the same model sequence using the method described by Berger et al. [40] to investigate the influence of noise and incompleteness of input data on our method. We first add noise into the testing data, and the tracking error in Fig. 10b indicates that the tracking result does not change too much comparing with the prefect data, although the percentage of errors below 0.01 decreases a little. We further add incompleteness into the noisy data, and there is also no significant downgrade of the tracking result, as shown in Fig. 10c. Figs. 11c and 11d show the tracked pose and marker positions at the frame 143 of the two data sets, and the results are still good. The full tracking result on all the frames of dancing model data is shown in our supplementary video.

5.3.2 Result on Berkeley MHAD

The Berkeley MHAD [36] contains 12 subjects (persons) and each subject performs the same 12 actions, which are: 1) *jumping in place*, 2) *jumping jacks*, 3) *bending*, 4) *punching*, 5) *waving two hands*, 6) *waving one hand*, 7) *clapping hands*, 8) *throwing a ball*, 9) *sit down then stand up*, 10) *sit down*, 11) *stand up*, and 12) *T-pose*. Since actions 10 and 11 are decompositions of action 9, we exclude them from our pose tracking experiment. The T-pose action (action 12) is used as the input of our geometry fitting process, which computes the spherical harmonic coefficients for displacements and normals, thus is not used in the pose tracking process either. Due to the page limit we only choose subject 2 as a delegate in this paper.

The color map in Fig. 12 shows the average tracking errors on Berkeley MHAD for each of the 9 actions and 43 markers. The statistic shows that over 80% average tracking errors are below 6 cm. The tracking results on all the actions in MHAD can be checked in our supplementary video for a more intuitive evaluation.

From Fig. 12 we can see the average tracking error on action 4 (punching) is bigger comparing with other actions. So we visualize the tracking errors of all the frames in action 4 for further analysis, as shown in Fig. 13.

In Fig. 13 we notice that the tracking errors on arms and hands (markers 14 to 19, and 22 to 27) are relatively larger, which is reasonable because in the punching action the major movement comes from arms and hands. The color map in Fig. 13 indicates particular frames where the tracking

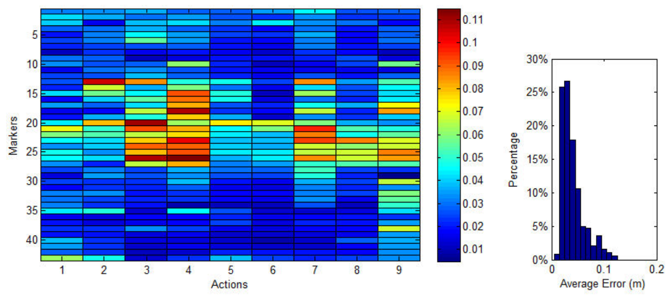


Fig. 12: Average tracking errors on Berkeley MHAD.

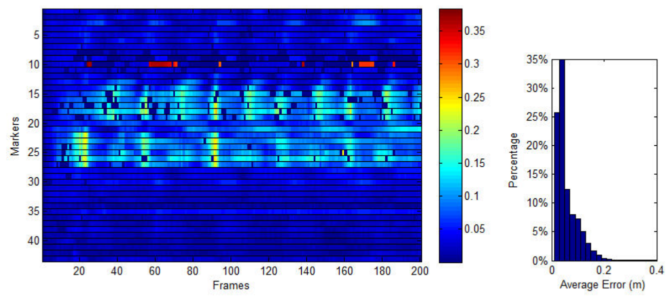


Fig. 13: Tracking errors on action 4 (punching) in MHAD.

errors on arms and hands are relatively larger. We pick frames 23 and 92 among them and visualize the tracked pose, markers, along with the color image in Fig. 14. From Fig. 14 we can see the tracked poses are still acceptable in these worse cases.

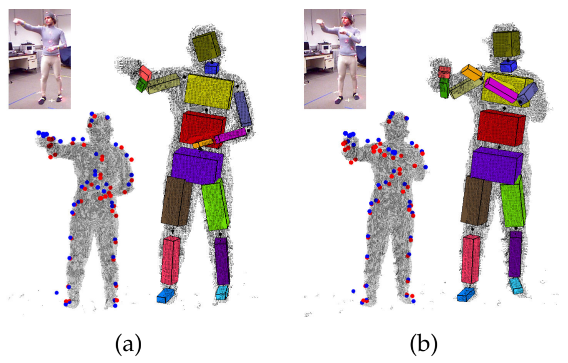


Fig. 14: Tracking results on data frames (a) 23 and (b) 92 in action 4 (punching) where the tracking errors are large. In each frame the color image is on the top-left, the MHAD markers (blue) and our computed markers (red) are on the bottom-left, and our tracked pose is on the right.

The largest tracking error in Fig. 13 comes from marker 10, but the error is not faithful because the optical marker positions provided in MHAD is inaccurate in this case. Marker 10 is attached on the chest of the person, and its positions are not recorded correctly for some frames in MHAD, probably due to occlusion. Fig. 15 shows the positions of marker 10 of data frames 25 and 30. In Fig. 15a the reported position of marker 10 in HMAD (blue) is far away from the chest, and in Fig. 15b the position of MHAD marker 10 is at the origin (in which case we count the tracking error as 0). Fig. 15 also shows the computed marker positions (red) from our pose tracking result, which are all on the chest of the person.

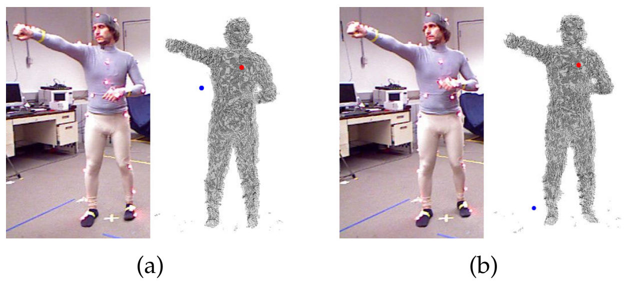


Fig. 15: MHAD marker positions (blue) and our computed marker positions (red) for marker 10 on data frames (a) 25 and (b) 30.

5.4 Comparison

We compare the average joint position error of our method with 2 state-of-art depth camera based marker-less motion tracking methods, Wei et al. [7] and Zhang et al. [17] (we use their results without pressure data for a fair comparison), as well as traditional ICP and Kinect SDK [13], as shown in Fig. 16. The input data and tracking results of other methods are all from [17]. From Fig. 16 we can see our method performs a little better than Wei et al. [7], and much better than ICP and Kinect SDK [13]. Although our method performs a little worse than Zhang et al.'s [17] on the actions presented in their paper, we see a failure case with their method on which our method performs well. Fig. 17 shows 2 consecutive frames in a jumping action and the tracking results of our method and Zhang et al.'s [17]. The missing data in the first frame causes the failure of their method on right arm, while our method still works fine. Notice the input data we obtained from the authors of [17] does not include normal data so our method runs without the help of normal information for segmentation. In addition, the input data is not well calibrated, which also reduces the tracking quality of our method since the geometry of poorly calibrated data is not accurate. So it is reasonable that our method performs a little worse than Zhang et al.'s [17].

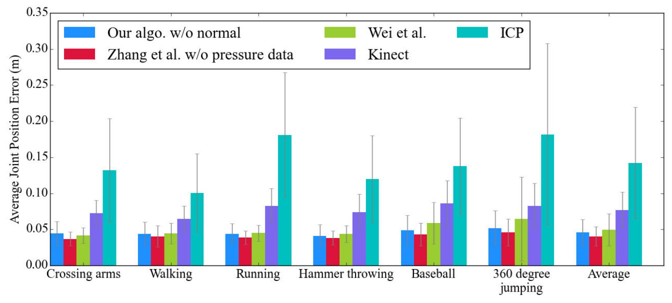


Fig. 16: Average tracking error comparison.

5.5 Variety of Motions

To further test the performance of our pose tracking method we capture various motions with both Microsoft Kinects

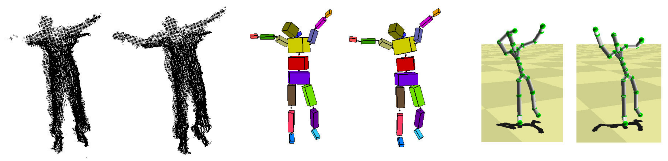


Fig. 17: Point cloud (left) and tracking results of our method (middle) and Zhang et al.'s [17] method (right) for 2 consecutive frames.

and Kinect v2s in the experiment. To alleviate the occlusion problem in data capturing we use multiple depth cameras, specifically 4 Kinects or 3 Kinect v2s, which have shooting directions forming 90 or 120 degree angles with each other. The multiple cameras are calibrated with the method described by Auvinet et al. [37]. There is no optical marker tracking system available in our experiment so the tracking result can not be compared with any ground truth, and we can only visually check the tracking performance along with the depth and color images. However, comparing with the skeletons reported by Kinect SDK, our tracking method shows a higher success rate in the experiment. Fig. 18 shows the tracking results on some motions captured by Kinects or Kinect v2s, along with the color/depth images and Kinect SDK reported skeletons. One major problem with Kinect SDK skeleton is that it performs bad if there is an occlusion. Our method uses the fused data from multiple cameras so does not suffer from this issue. Another problem with the skeleton from Kinect SDK is that the bones are determined by the joint positions, which are not always accurate, so the lengths of bones change from one frame to another, and sometimes become inaccurate. In our framework we use the predefined skeleton model so it can avoid this problem. For the comprehensive pose tracking results on Kinect and Kinect v2 data the readers can check our supplementary video.

5.6 Failure Cases

From the observation of our detailed experiments we summarize two major reasons which cause the failure of our motion tracking method. One reason is the bad initialization of the skeleton, as described in Sec. 5.2. This is a common problem for the energy optimization based motion tracking methods. Although we have several mechanisms to reduce the chance of falling into local minimum, our method still fails for some bad initialization. The other reason is the quality of the input data. Moderate noise and incompleteness of the input data can reduce the performance of our method, as shown in Sec. 5.3. Even worse data quality may cause the failure of our pose tracking method, as shown in Fig. 19. In this figure the captured data frames from different Kinects at the same time are not well synchronized when a fast punching is performed, and our tracked poses of the right arm and hand are incorrect due to the bad data quality.

5.7 Performance

The hardware platform for running all our motion tracking experiments consists of Intel Xeon E5620 2.4GHz CPU, Nvidia GeForce GTX 780 Ti graphics card, 20GB DDR3 RAM and 64-bit Microsoft Windows 7 Enterprise operating system. The typical running time along with the data size and optimization iteration number for different data types in our experiment are reported in Table 4. We can see the running time is affected by the data size and optimization iteration number. The average running time ranges from 200 ms to 400 ms, while the maximum time can exceed 1 second.

TABLE 4: Running-time of different types of data.

Data Set (# of Frames)		Point #	Lloyd Iter. #	L-BFGS-B Iter. #	Time (ms)
Kinect 2: Sitting Down (Fig. 18f) (226)	*avg.*	25620	5	25	193
	max.	36620	9	83	501
Kinect: Bending (Fig. 18a) (175)	*avg.*	82184	7	35	429
	max.	94573	29	121	1190
MHAD: Action 1 (Fig. 12) (135)	*avg.*	47912	6	46	389
	max.	54460	24	125	926
Perfect Dancing Model (Fig. 10a) (200)	*avg.*	7061	9	71	396
	max.	7061	29	186	892

6 Limitation and Future Work

One major limitation of our proposed motion tracking framework is that it requires a manual alignment and adjustment for the skeleton initialization in the geometry fitting process (Sec. 3.2). In the future we would like to explore an automatic method for the geometry fitting, by following the ideas from skeleton extraction and rigging works [23], [26].

Another major limitation of this work is our motion tracking system is not exactly running in real-time. Possible approaches to reduce the running time include reducing the size of input points by down-sampling the depth image, adopting a different optimization framework which converges faster, and optimizing the GPU implementation of our algorithm.

Currently only the depth images are used for our method. In the future it is possible to integrate this method with other modalities of data, such as color images and/or motion sensors, to further improve the tracking accuracy.

References

[1] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[2] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of Markov random fields for segmentation of 3D scan data," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 169–176.

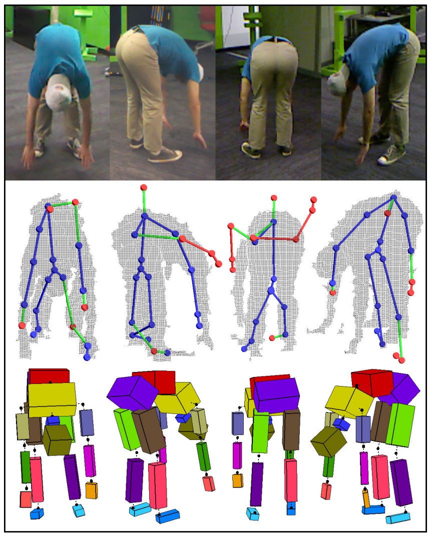
[3] M. Siddiqui and G. Medioni, "Human pose estimation from a single view point, real-time range sensor," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2010, pp. 1–8.

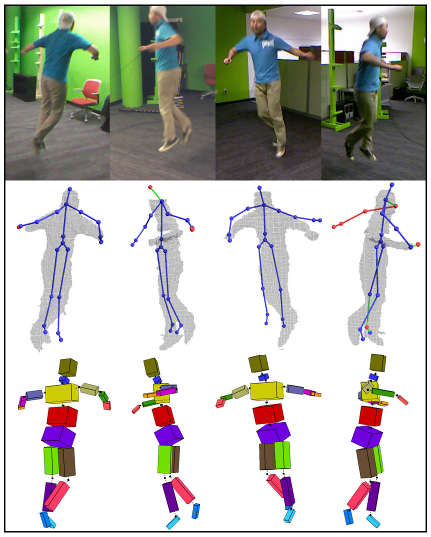
[4] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. H. Torr, "Randomized trees for human pose detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

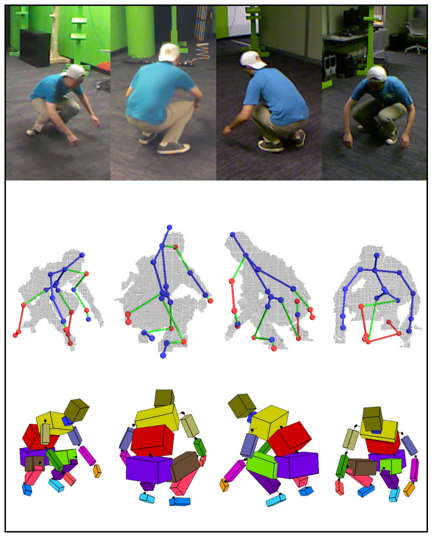
[5] S. Knoop, S. Vacek, and R. Dillmann, "Sensor fusion for 3D human body tracking with an articulated 3D body model," in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1686–1691.

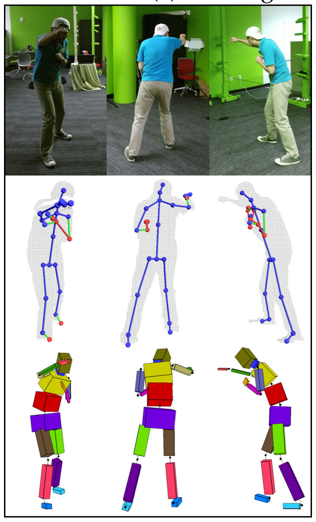
[6] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3D tracking of hand articulations using Kinect," in *Proceedings of the 22nd British Machine Vision Conference*, vol. 1, no. 2, 2011, p. 3.

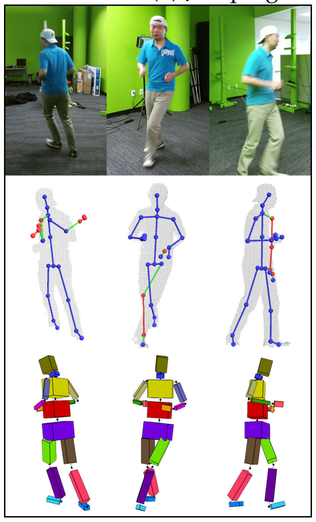
[7] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 188:1–188:12, 2012.

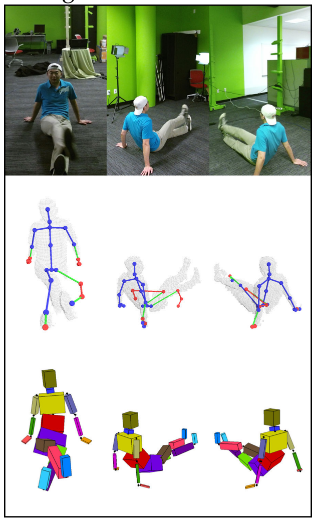

(a) Bending.


(b) Jumping and turning around.


(c) Crouching.


(d) Punching.


(e) Jogging.


(f) Sitting down.

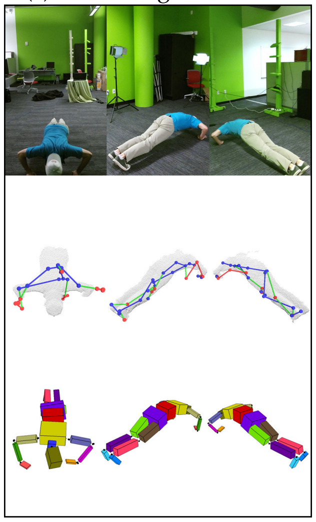

(g) Pushing up.

Fig. 18: Tracking results on various motions captured by 4 Kinects (a, b and c) or 3 Kinect v2s (c, d, e and f). In each sub-figure the color images, depth images, along with Kinect SDK reported skeletons, and our tracked skeleton are shown from top to bottom. The data from different Kinects for the same frame and our tracked skeleton from the same viewpoints are displayed from left to right. When visualizing the skeleton from Kinect SDK, the tracked joints are rendered with blue, and the inferred joints (due to occlusion) are rendered with red. The color of bones are determined by the color of their joints (red/blue if both joints are red/blue, and green if one joint is red and the other is blue).

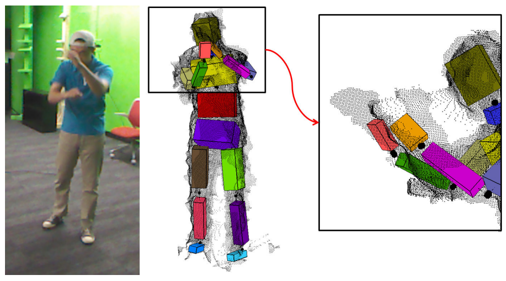


Fig. 19: Failure case of our pose tracking method in the punching action captured by 4 Kinects.

[8] T. K. Dey, B. Fu, H. Wang, and L. Wang, "Automatic posing of a meshed human model using point clouds," *Computers & Graphics*, vol. 46, pp. 14–24, 2015.

[9] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Transactions on Graphics*, vol. 27, no. 3, p. 98, 2008.

[10] C. Wu, C. Stoll, L. Valgaerts, and C. Theobalt, "On-set performance capture of multiple actors with a stereo camera," *ACM Transactions on Graphics*, vol. 32, no. 6, p. 161, 2013.

[11] C. Stoll, N. Hasler, J. Gall, H.-P. Seidel, and C. Theobalt, "Fast articulated motion tracking using a sums of gaussians body model," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 951–958.

[12] A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt, "Efficient ConvNet-based marker-less motion capture in general scenes with a low number of cameras," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 1297–1304.

[14] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, “Efficient regression of general-activity human poses from depth images,” in *IEEE International Conference on Computer Vision*, 2011, pp. 415–422.

[15] D. Grest, J. Woetzel, and R. Koch, “Nonlinear body pose estimation from depth images,” in *Proceedings of the 27th DAGM conference on Pattern Recognition*, 2005, pp. 285–292.

[16] Y. Zhu and K. Fujimura, “Constrained optimization for human pose estimation from depth sequences,” in *Proceedings of the 8th Asian Conference on Computer Vision*, 2007, pp. 408–418.

[17] P. Zhang, K. Siu, J. Zhang, C. K. Liu, and J. Chai, “Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture,” *ACM Trans. Graph.*, vol. 33, no. 6, pp. 221:1–221:14, Nov. 2014.

[18] D. Grest, V. Kr1ger, and R. Koch, “Single view motion tracking by depth and silhouette information,” in *Scandinavian Conference on Image Analysis*, 2007, pp. 719–729.

[19] G. Ye, Y. Liu, N. Hasler, X. Ji, Q. Dai, and C. Theobalt, “Performance capture of interacting characters with handheld Kinects,” in *Proceedings of the 12th European Conference on Computer Vision*, 2012, pp. 828–841.

[20] T. Helten, A. Baak, G. Bharaj, M. Muller, H.-P. Seidel, and C. Theobalt, “Personalization and evaluation of a real-time depth-based full body tracker,” in *Proceedings of the 2013 International Conference on 3D Vision*, 2013, pp. 279–286.

[21] S. Sridhar, A. Oulasvirta, and C. Theobalt, “Interactive markerless articulated hand motion tracking using RGB and depth data,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2456–2463.

[22] T. Helten, M. Muller, H.-P. Seidel, and C. Theobalt, “Real-time body tracking with one depth camera and inertial sensors,” in *IEEE International Conference on Computer Vision*, 2013, pp. 1105–1112.

[23] I. Baran and J. Popović, “Automatic rigging and animation of 3D characters,” in *ACM Transactions on Graphics*, vol. 26, no. 3, 2007, p. 72.

[24] G. Bharaj, T. Thormählen, H.-P. Seidel, and C. Theobalt, “Automatically rigging multi-component characters,” in *Computer Graphics Forum*, vol. 31, no. 2pt3, 2012, pp. 755–764.

[25] L. Lu, B. Lévy, and W. Wang, “Centroidal Voronoi tessellation of line segments and graphs,” in *Computer Graphics Forum*, vol. 31, no. 2pt4, 2012, pp. 775–784.

[26] L. Lu, Y.-K. Choi, W. Wang, and M.-S. Kim, “Variational 3D shape segmentation for bounding volume computation,” in *Computer Graphics Forum*, vol. 26, no. 3, 2007, pp. 329–338.

[27] D. Brunner and G. Brunnett, “Fast force field approximation and its application to skeletonization of discrete 3D objects,” in *Computer Graphics Forum*, vol. 27, no. 2, 2008, pp. 261–270.

[28] N. D. Cornea, D. Silver, and P. Min, “Curve-skeleton properties, applications, and algorithms,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

[29] P. D. Simari and K. Singh, “Extraction and remeshing of ellipsoidal representations from mesh data,” in *Proceedings of Graphics Interface*, 2005, pp. 161–168.

[30] X. Jia, Y.-K. Choi, B. Mourrain, and W. Wang, “An algebraic approach to continuous collision detection for ellipsoids,” *Computer Aided Geometric Design*, vol. 28, no. 3, pp. 164–176, 2011.

[31] D. Saupe and D. V. Vranić, “3D model retrieval with spherical harmonics and moments,” in *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, 2001, pp. 392–397.

[32] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation invariant spherical harmonic representation of 3D shape descriptors,” in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2003, pp. 156–164.

[33] Y. Fei, G. Rong, B. Wang, and W. Wang, “Parallel L-BFGS-B algorithm on GPU,” *Computers & Graphics*, vol. 40, pp. 1–9, 2014.

[34] M. Harris, S. Sengupta, and J. D. Owens, “Parallel prefix sum (scan) with CUDA,” *GPU Gems 3*, pp. 851–876, 2007.

[35] “Microsoft Kinect for Windows,” http://www.microsoft.com/en-us/kinectforwindows/.

[36] “Berkeley Multimodal Human Action Database,” http://tele-immersion.citris-uc.org/berkeley_mhad/.

[37] E. Auvinet, J. Meunier, and F. Multon, “Multiple depth cameras calibration and body volume reconstruction for gait analysis,” in *IEEE International Conference on Information Science, Signal Processing and their Applications*, 2012, pp. 478–483.

[38] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *IEEE International Conference on Computer Vision*, 1998, pp. 839–846.

[39] “CMU Graphics Lab Motion Capture Database,” http://mocap.cs.cmu.edu/.

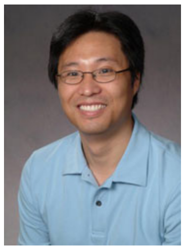
[40] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, “A benchmark for surface reconstruction,” *ACM Transactions on Graphics*, vol. 32, no. 2, p. 20, 2013.

Liang Shuai received his Ph.D. degree in Computer Science from the University of Texas at Dallas in 2014. Before that he received his M.S. degree in Computer Architecture from Graduate University of Chinese Academy of Sciences in 2009 and B.E. degree in Computer Science and Technology from Tsinghua University in 2006. His research interests include computer graphics, animation and visualization, especially on motion tracking, 3D reconstruction, 3D modeling, Voronoi diagram and GPU algorithm.

Chao Li received the B.E. degree in Software Engineering from Beijing Jiaotong University, Beijing, China, in 2009, the M.E. degree from Peking University, Beijing, China, in 2012, and is currently a Ph.D candidate in the Department of Computer Science at The University of Texas at Dallas, Dallas, USA. His research interests include motion tracking, 3D reconstruction, animation and other related topics in the field of Computer Graphics.

Xiaohu Guo received the PhD degree in computer science from the State University of New York at Stony Brook in 2006. He is an associate professor of computer science at the University of Texas at Dallas. His research interests include computer graphics, animation and visualization, with an emphasis on geometric, and physics based modeling. His current researches at UT-Dallas include: spectral geometric analysis, deformable models, centroidal Voronoi tessellation, GPU algorithms, 3D and 4D medical image analysis, etc. He received the prestigious National Science Foundation CAREER Award in 2012.

Balakrishnan Prabhakaran is currently Professor of Computer Science in the University of Texas at Dallas. He received his PhD in Computer Science from Indian Institute of Technology, Madras, India in 1995. He has been working in the area of multimedia systems: animation & multimedia databases, authoring & presentation, resource management, and scalable webbased multimedia presentation servers. Dr. Prabhakaran received the prestigious National Science Foundation (NSF) CAREER Award in 2003 for his proposal on Animation Databases.



Jinxiang Chai is currently an associate professor in the Department of Computer Science and Engineering at Texas A&M University. He received his Ph.D in 2006 from the Robotics Institute at Carnegie Mellon. His primary research is in the area of computer graphics and vision with broad applications in other disciplines such as robotics, human computer interaction, and biomechanics, virtual and augmented reality. He received an NSF CAREER award for his work on theory and practice of Bayesian motion synthesis.