Improved Skeleton Tracking by Duplex Kinects: A Practical Approach for Real-Time Applications

Kwok-Yun Yeung, Tsz-Ho Kwok, and Charlie C. L. Wang*

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract

Recent development of per-frame motion extraction method can generate the skeleton of human motion in real-time with the help of RGB-D cameras such as Kinect. This leads to an economic device to provide human motion as input for real-time applications. As generated by a single-view image plus depth information, the extracted skeleton usually has problems of unwanted vibration, bone-length variation, self-occlusion, etc. This paper presents an approach to overcome these problems by synthesizing the skeletons generated by duplex Kinects, which capture the human motion in different views. The major technical difficulty of this synthesis comes from the inconsistency of two skeletons. Our algorithm is formulated under the constrained optimization framework by using the bone-lengths as hard constraints and the tradeoff between inconsistent joint positions as soft constraints. Schemes are developed to detect and re-position the problematic joints generated by per-frame method from duplex Kinects. As a result, we develop an easy, cheap and fast approach that can improve the skeleton of human motion at an average speed of 5ms per frame.

Keywords: Skeleton, motion, RGB-D camera, real-time, user interface.

1 Introduction

Human motion recognition, as a very natural method for Human-Computer Interaction (HCI), plays an important role in many computer systems and applications (e.g., [5, 6, 22]). Widely used methods for capturing human motion in virtual environment include optical motion capture systems [31,35], inertial systems [36], acoustic systems [9, 21], mechanical motion capture systems [16, 19] as well as hybrid systems combine multiple sensor types [3, 7, 32]. However, most of these systems need specific hardware and software, which are expensive. Moreover, the systems usually have a long installation time and complicated installation steps.

Since the release of Microsoft Kinect [17], this type of RGB-D camera attracts many attentions from a variety of communities as it provides many new opportunities for HCI. The depth information provided by RGB-D cameras facilitate a lot of applications, such as computer games, virtual try-on [8], environmental interaction [34], 3D reconstruction and interaction [12, 20], and human body scanning [30, 33]. The function of real-time human skeleton extraction in Microsoft Kinect SDK [17, 29] is very impressive. Nevertheless, when applying the skeleton generated by Kinect library in *virtual reality* (VR) and robotics applications, some problems are found. Generally, limited by the information provided from a single-view, the skeleton extracted by [29] has the following problems (see also the illustration in Fig.1).

Figure 1: Problems of skeleton tracking by single Kinect – the viewing direction of Kinect sensor is specified by yellow arrows. (Top-left) *Self-occlusion*: the left arm is hidden by the main body so that the positions of elbow and wrist are estimated to incorrect places. (Top-right) *Bone-length variation*: when viewing from a different direction, the length of forearm in an open-arm pose (right) changes significantly from its length that is captured when stands facing the Kinect camera (left). (Bottom) *Artificial vibration*: When viewing from a specific angle, the position of elbow joint has unwanted vibration even if a static pose is kept.

- *Self-occlusion*: This happens when some parts of a human body are hidden. As the RGB-D camera can only report the depth value of a pixel that is nearest to the camera, information at the hidden part will be missed (e.g., the top-left of Fig.1).
- *Bone-length variation*: The method of Shotten et al. in [29] first segments the pixels of a human body into different regions, and the segmentations are used to generate confidence-weighted proposals for the joint posi-

*Corresponding Author; Email: cwang@mae.cuhk.edu.hk

tions. However, as there is no explicit constraint applied for the bone-length, the lengths of bones varied significantly during the motion (e.g., the top-right of Fig.1).

- *Artificial vibration*: Caused by the acquisition error from camera, the segmentation conducted in [29] vibrates near the boundary of regions even when the human body is not moved. This leads to unwanted vibrations on the extracted joint-positions (see the bottom of Fig.1 for an example), which also make the lengths of bones change during the motion. On the other aspect, this artifact vibration is also a factor leading to the bone-length variation in motion.

Although there are some solution (e.g., [1]) to improve the extracted skeleton in a single-view system, a more reliable consideration is to introduce more cameras into the motion extraction system. Especially when the cost of such a camera is low (e.g., the Kinect sensor). For the problem shown in Fig.1 (top-left), a single-view approach like [1] can have trivial chance to fix the incorrect skeleton. In our system, duplex Kinects are used. All above drawbacks are improved by our approach, which is easy, cheap and fast.

1.1 Our system

To overcome the problems of skeleton generated from a single-view, we adopt a system with duplex Kinects for motion capture. A camera facing the user at the beginning of per-frame motion capture is called *principal camera* (denoted by K_A in the rest of this paper), and the other camera is called *secondary camera* (denoted by K_B). Although the artificial variation can be somewhat reduced by averaging the positions of joints reported by two cameras, this setup with two sensors cannot automatically solve the problems of bone-length variation and self-occlusion. In many cases, positions of the same joint reported by K_A and K_B are far away from each other – called *unmatched joints* in the rest of this paper.

As shown in Fig.2, such inconsistent situation occurs even after registering the coordinate systems of two Kinect cameras by a calibration procedure. Specifically, 3D point clouds are used to calibrate the positions of two Kinects by a method similar to [20] so that the 3D environments captured by K_A and K_B overlap. Our analysis finds that the inconsistency of skeletons is generated by the following reasons:

- The 3D regions captured by two Kinects are partially overlapped on the human body (as shown in the top row of Fig.2); therefore, the positions of joints computed independently by the 3D information obtained from K_A and K_B are seldom coincident although they could be very close to each other after 3D registration.
- More significant inconsistency is caused by the misclassification of regions in the 3D data obtained from a single-view. As illustrated in the bottom row of Fig.2, the joint of right elbow is tracked to the waist joint by mistake if only K_A is used. However, this joint is reported as 'tracked' by Kinect SDK [17]. Simply averaging two skeletons will generate an incorrect result.

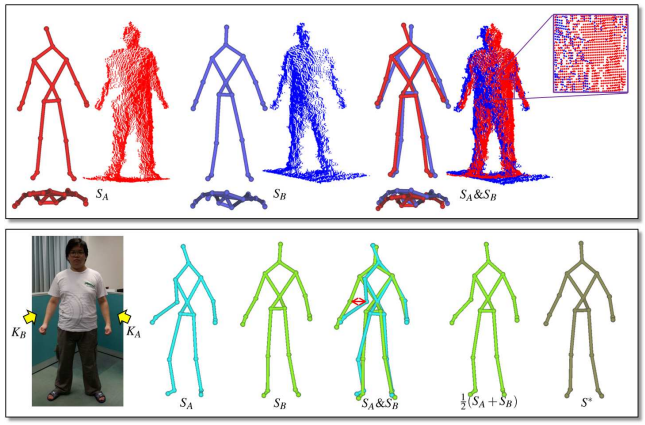

Figure 2: Inconsistent skeletons extracted by two Kinects. (Top row) The 3D information captured by K_A (in red) and K_B (in blue) are only partially overlapped even after carefully applying a registration procedure; as a result, the extracted skeletons, S_A and S_B, can be very close to each other but seldom be coincident. (Bottom row) In the view of K_A, the elbow joint of S_A is misclassified to the region of waist joint; although the position of this elbow joint on S_B is correct, simply computing the average of S_A and S_B (i.e., $\frac{1}{2}(S_A + S_B)$) will not give the result as good as S^* generated by our approach.

- When self-occlusion occurs in the view of any Kinect, inconsistent positions can also be generated as the joint position is estimated (but not being tracked) according to the motion database of Kinect SDK library [17].

The problem of inconsistency can be solved under our optimization framework (details can be found in Sections 2 and 3).

In this paper, we present an approach using the setup of duplex Kinects to enhance the skeletons generated by Kinect system. Two Kinects, K_A and K_B, are placed 'orthogonal' to each other. Note that, it is not necessary to let K_A and K_B exactly orthogonal to each other, where the relative position and orientation between them can be automatically registered at the beginning of skeleton tracking (see Section 4). Microsoft Kinect SDK [17] is used to generate two skeletons S_A and S_B corresponding to K_A and K_B respectively. Under a constrained optimization framework, an optimal skeleton S^* is obtained by using the bone-lengths as hard constraints and the tradeoff between inconsistent joint positions as soft constraints. A practical per-frame method is developed to detect and re-position the problematic joints efficiently. As a result, we are able to compute S^* in real-time applications (i.e., at about 30 fps). The skeleton enhancement itself can be updated at $\sim$ 5 ms for each frame in average when running on our test platform.

1.2 Related work

The research of motion capture system has a long history, where the surveys of vision-based techniques can be found in [18, 25]. The skeleton generation function provided by

Kinect SDK is based on the approach described in [29]. It is a per-frame-based method. A body parts representation is designed to map the pose estimation problem into a per-pixel classification problem. The segmentations are used to generate confidence-weighted proposals for the joint positions. A large and highly varied training data set is employed for the classification problem. As a result, the approach is fast and can predict the joint points position in 3D regardless to body shape or clothing. Our work presented in this paper is based on above pose estimation approach to generate the approximated positions of joints. Differently, we focus on how to improve the skeletons generated by these approaches so that the motion of human body is more realistic to be used in real-time applications.

A thread of research in motion is about how to improve (or correct) joint-positions captured by motion systems. Many existing approaches focus on the problem of occlusion filling. Interpolation methods (e.g., linear or spline interpolation [26]) are commonly used to estimate the missed markers. However, interpolation algorithms cannot be used for on-site applications as they require data sampled both before and after the period of occlusion. Tommaso et al. [23] propose an extrapolation algorithms which only require data sampled before an occlusion. Bone-length constraints are used in BoLeRO [14] for occlusion filling methods. Ho et al. [10] used bone-length constraints in character motion. By keeping the distance between adjacent joints from the original scales to the target scales, the scene semantics is captured. Our approach also takes the bone-length constraints as a basic assumption.

Real-time human skeleton extraction is an impressive function provided by Microsoft Kinect SDK [17]. However, information that can be captured in a single view is not sufficient for many motions. Aforementioned problems occur frequently. Therefore, more attention is paid on multiple Kinect approaches recently (e.g., [4,30]). A survey about 3D human pose estimation from multi-view videos can be found in [11]. Difficulties of using multiple Kinects for motion capture have been analyzed above.

1.3 Main result

Our approach provides the following advantages in skeleton tracking.

- By using the constraints of bone-lengths and the weighting scheme to solve inconsistent joint positions, this approach provides an improved per-frame skeleton tracking solution.
- The joint mistracking problem of Kinect is solved by a new method that can efficiently estimate the reliability of joint positions and then adjust the weights in optimization.
- Since the inconsistency of joint-positions is solved under the constrained optimization framework, two Kinects can be automatically calibrated in a very simple way, which greatly reduces the installation time of system.

As a result, a low-cost ($\sim$ \$100 $\times$ 2) RGB-D camera-based skeleton tracking interface is developed as an input device for real-time applications.

2 Optimization

In this section, we formulate the skeleton enhancement problem under a framework of constrained optimization.

Without loss of generality, we assume that a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ have been obtained during the initialization of the duplex Kinects system to synthesize the 3D scenes captured by two cameras. For any point $\mathbf{q} \in \Re^3$ in the coordinate system of K_B, its corresponding position in the system of K_A is $\mathbf{q}' = \mathbf{R}\mathbf{q} + \mathbf{t}$. Given two skeletons, S_A and S_B that are generated by K_A and K_B respectively, we wish to optimize the position of every joint $\mathbf{p}_i \in S_A$ to $\mathbf{p}_i^*$ in the coordinate system of K_A to satisfy the following requirements:

- The distance between two neighboring joints (e.g., $\mathbf{p}_i^*$ and $\mathbf{p}_j^*$) is expected to be the same as the corresponding bone-length[1] (e.g., $l_{i,j}$).
- When the positions of a joint i obtained in both K_A and K_B are reliable, $\mathbf{p}_i^*$ should be *as-close-as-possible* (ACAP) to $\mathbf{p}_i$ and $\mathbf{R}\mathbf{q}_i + \mathbf{t}$ with $\mathbf{q}_i$ being the position of joint i on S_B.
- When the positions of a joint i obtained by one Kinect is reported as unreliable and an estimated position is given, the position of $\mathbf{p}_i^*$ should be closer to the reliable position.

Among these requirements, the length preservation of bones is set as hard constrains in our optimization framework while the ACAP request is set as soft constraints. In short, joints of a skeleton can be re-positioned by solving the following constrained optimization problem.

$$\begin{array}{ll} \min & \sum_{i\in S_A} w_i^A \|\mathbf{p}_i^* - \mathbf{p}_i\|^2 + w_i^B \|\mathbf{p}_i^* - (\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2 \\ s.t. & \sum_{\{i,j\}\in S_A} (\|\mathbf{p}_i^* - \mathbf{p}_j^*\| - l_{i,j})^2 = 0 \end{array}, \quad (1)$$

where the weights w_i^A and w_i^B are adjusted according to the reliability of $\mathbf{p}_i$ and $\mathbf{q}_i$. Details about weighting will be discussed in Section 3 below. Note that in Eq.(1), $\mathbf{p}_i$ and $\mathbf{q}_i$ could be either 'tracked' positions or 'estimated' positions (reported by Kinect SDK as 'inferred' joints).

The constrained optimization problem defined in Eq.(1) can be solved efficiently. With the Lagrange multiplier λ, an augmented objective function as

$$\begin{array}{rcl} J(\mathbf{X}) & = & \sum_{i\in S_A} w_i^A \|\mathbf{p}_i^* - \mathbf{p}_i\|^2 + w_i^B \|\mathbf{p}_i^* - (\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2 \\ & & +\lambda \sum_{\{i,j\}\in S_A} (\|\mathbf{p}_i^* - \mathbf{p}_j^*\| - l_{i,j})^2 \end{array} \quad (2)$$

can be minimized by using Newton's method. Here, the unknown vector becomes $\mathbf{X} = \{\mathbf{p}_i^*\} \cup \lambda$. The update in each iteration of Newton's approach consists of two steps:

1. Solving $\nabla J^2(\mathbf{X})\mathbf{d} = -\nabla J(\mathbf{X})$;

[1] The bone-length can be obtained during the initialization step.

2. $\mathbf{X} \leftarrow \mathbf{X} + \tau\mathbf{d}$ by using a line search to determine the best τ.

Note that, in the computation of each frame, the optimized joint-positions obtained in previous frames are used as the initial value of $\mathbf{X}$. Using the sequential linearly constrained programming, the second derivatives of the constraint are neglected. The equation $\nabla J^2(\mathbf{X})\mathbf{d} = -\nabla J(\mathbf{X})$ to be solved in each step is simplified to

$$\begin{pmatrix} H & \Lambda^T \\ \Lambda & 0 \end{pmatrix}\begin{pmatrix} \mathbf{d}_p \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{b}_p \\ b_\lambda \end{pmatrix} \tag{3}$$

with $\{\mathbf{p}_i^*\}_{new} = \{\mathbf{p}_i^*\} + \mathbf{d}_p$. Here, the vectors Λ and $\mathbf{b}_p$ are

$$\Lambda = \{\frac{\partial}{\partial \mathbf{p}_i^*} \sum_{\{i,j\}\in S_A} (\|\mathbf{p}_i^* - \mathbf{p}_j^*\| - l_{i,j})^2\} \tag{4}$$

$$\mathbf{b}_p = -\{\frac{\partial}{\partial \mathbf{p}_i^*} \sum_{i\in S_A} w_i^A \|\mathbf{p}_i^* - \mathbf{p}_i\|^2 + w_i^B \|\mathbf{p}_i^* - (\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2\} \tag{5}$$

and the value of b_λ is

$$b_\lambda = -\sum_{\{i,j\}\in S_A} (\|\mathbf{p}_i^* - \mathbf{p}_j^*\| - l_{i,j})^2. \tag{6}$$

H is a diagonal matrix $H = diag\{h_i\}$ that has

$$h_i = \frac{\partial^2}{\partial \mathbf{p}_i^{*2}} (w_i^A \|\mathbf{p}_i^* - \mathbf{p}_i\|^2 + w_i^B \|\mathbf{p}_i^* - (\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2). \tag{7}$$

Efficient Numerical Scheme: By the above formulation, when applying the iterations to find optimal value of $\{\mathbf{p}_i^*\}$, we can actually determine the value of $\mathbf{d}_p$ in a more direct way (i.e., without applying the general numerical solver). Specifically, the value of λ can be computed by

$$\lambda = (\Lambda H^{-1}\mathbf{b}_p - b_\lambda)/(\Lambda H^{-1}\Lambda^T) \tag{8}$$

with $H^{-1} = \{1/h_i\}$ since H is a diagonal matrix. The value of $\mathbf{d}_p$ can be determined by the substitution that

$$\mathbf{d}_p = H^{-1}(\mathbf{b}_p - \Lambda^T\lambda). \tag{9}$$

In short, the optimization for joint-positions in a frame is completed in a very efficient way. The resultant joint-positions can be obtained in less than 5ms in average according to our experimental tests. Here, we stop the iteration of Newton's approach when $\|\mathbf{d}\| < 10^{-5}$ or the update has been taken for more than 50 times.

3 Scheme for Solving Inconsistency

Now we introduce the method about how to determine the values of w_i^A and w_i^B based on the reliability of joint-positions extracted by K_A and K_B. When using a single Kinect setup, whether a joint is 'well' tracked will be reported by the Kinect SDK as 'tracked' or 'inferred'. The position of an 'inferred' joint is estimated by the method proposed in [29] – which will be mentioned as 'estimated' positions in the rest of this paper. When duplex Kinects are employed in the motion capture, for the same joint, it can be reported as 'tracked' by one Kinect while being reported as 'inferred' by another one. A scheme is developed to determine the weights (i.e., w^A and w^B) employed in the above constrained optimization framework to indicate the reliability of a position.

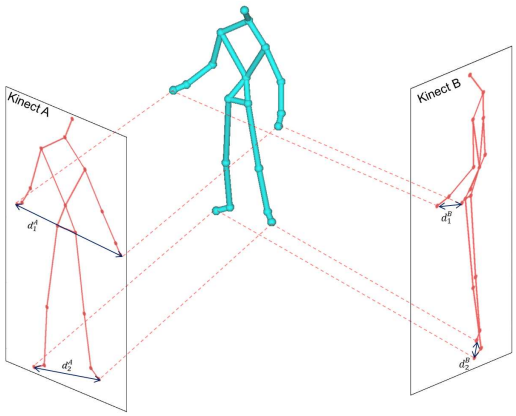

Figure 3: An illustration for explaining the observation that the distance between mis-tracked joints in one viewing plane will generally be much shorter than the distance in another viewing plane.

According to our experimental tests, the positions of a joint i reported by K_A and K_B (e.g., $\mathbf{p}_i$ and $\mathbf{q}_i'$) can be *unmatched* – i.e., the distance between $\mathbf{p}_i$ and $\mathbf{q}_i'$ is large. The unmatched case sometime happens even when both Kinect sensors report 'tracked' joints. Our observation finds that most of the unmatched cases are led by the mis-tracking of joint position in a camera. To resolve this problem, we need to figure out which one is mis-tracked and therefore give a lower weight in optimization. Without loss of generality, mis-tracking occurs in the scenario that a joint i is very close to another joint m in the viewing plane of a camera (e.g., K_A) but they are actually far away from each other in $\Re^3$ – m stands for *misleading* here. This can be detected in the viewing plane of another camera (e.g., K_B) which is placed in a nearly orthogonal orientation. An illustration is shown in Fig.3. For a joint i, the following steps are conducted to figure out which joint m will more likely lead to the mis-tracking of i:

- We first project $\mathbf{p}_i$ into the viewing plane of K_A and search its closest joint j in the viewing plane which has no bone directly linking to i.
- Then, we project $\mathbf{q}_i'$ into the viewing plane of K_B and search its closets joint k – again, there must have no bone linking i and k.
- The distance, $d_{i,j}^A$, between i and j in the viewing plane of K_A and the distance, $d_{i,k}^B$, between i and k in the viewing plane of K_B are compared. If $d_{i,j}^A < d_{i,k}^B$, j is considered as the mis-leading joint, m, that could cause the mis-tracking of i. Otherwise, k is considered as the possible candidate that leads to mis-tracking.

After determining the mis-leading joint m, distances between

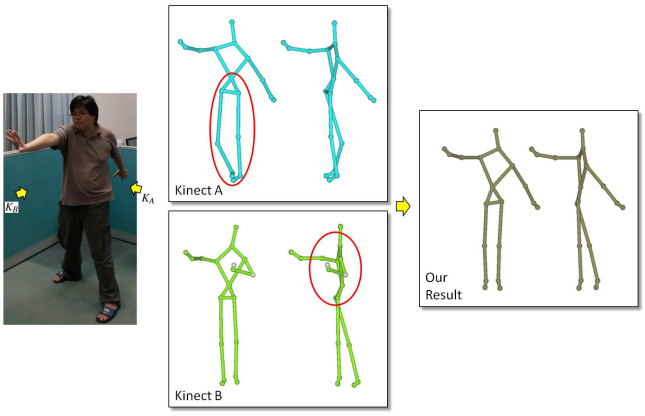

Figure 4: Our algorithm can correct the positions of problematic joints by resolving inconsistency under our constrained optimization framework while preserving the bone-lengths. The joints in the red circles are problematic.

points i and m in the viewing planes of K_A and K_B can be obtained as $d^A_{i,m}$ and $d^B_{i,m}$. If $d^A_{i,m} > d^B_{i,m}$, the position of i provided by K_A, $\mathbf{p}_i$, is more reliable; otherwise, the position generated by K_B, $\mathbf{q}'_i$, is more trustable. Such reliability is incorporated in the formulation of weights computation below.

The distance between i and the mis-leading joint (i.e., m) in the viewing plane of K_A and K_B are used to determine the basic weights of $\bar{w}^A_i$ and $\bar{w}^B_j$. Specifically,

$$\bar{w}^A_i = \frac{d^A_{i,m}}{d^A_{i,m} + d^B_{i,m}}, \quad \bar{w}^B_i = \frac{d^B_{i,m}}{d^A_{i,m} + d^B_{i,m}}. \tag{10}$$

Since the 'inferred' joint position is an estimated result by the Kinect SDK, the 'tracked' one is expected to be more reliable. If $\mathbf{p}_i$ and $\mathbf{q}'_i$ have different tracking states (i.e. 'tracked' and 'inferred'), there exists a reliability difference between the two joint points. In order to reflect this difference, two adjusting coefficients h^A_i and h^B_i are integrated to $\bar{w}^A_i$ and $\bar{w}^B_i$ respectively. Values of the coefficients are assigned to be h for the joint which is reported as 'tracked' and $(1-h)$ for the 'inferred' one, where $h \in (0.5, 1.0)$ is a parameter that can be tuned by users. A larger h is used, the weighting results depend more on the tracking state of Kinect SDK.

Finally, the weights are normalized to

$$w^A_i = \frac{(h^A_i \bar{w}^A_i)^4}{(h^A_i \bar{w}^A_i)^4 + (h^B_i \bar{w}^B_i)^4}, \quad w^B_i = \frac{(h^B_i \bar{w}^B_i)^4}{(h^A_i \bar{w}^A_i)^4 + (h^B_i \bar{w}^B_i)^4}. \tag{11}$$

$h = 0.786$ is adopted for all examples shown in this paper.

Our scheme proposed above can successfully resolve inconsistency. Figure 4 shows the processed result for an example with unmatched joints.

4 Details of Initialization

During initialization of the system, we need to determine a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ to synthesize the 3D information captured by two cameras, K_A and K_B. This is in fact a problem of rigid registration, where a good survey can be found in [27]. However, as shown in the top row of Fig.2, even after applying an accurate calibration step to determine $\mathbf{R}$ and $\mathbf{t}$, the transformed positions of joints obtained from K_B do not keep coincident with joint-positions extracted by K_A. Therefore, a simplified but more effective method is developed in our approach.

After installing and placing the two Kinects appropriately – nearly orthogonal to each other, we let a user to stand at about 45° facing both Kinects and in a pose with two arms and legs open (such as the top row of Fig.2). In such a pose, all joints S_A and S_B will be reported as 'tracked' and will be used to determine $\mathbf{R}$ and $\mathbf{t}$ in a least-square sense by minimizing the following energy function defined on joint positions

$$J_R = \sum_i \|\mathbf{R}(\mathbf{q}_i - \mathbf{c}_q) + \mathbf{t} - (\mathbf{p}_i - \mathbf{c}_p)\|^2, \tag{12}$$

where $\mathbf{c}_p$ and $\mathbf{c}_q$ are the average centers of $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_i\}$ respectively. Specifically, the 3×3 matrix $\mathbf{R}$ can be solved by *Singular Value Decomposition* of the linear equations of $\partial J_R / \partial \mathbf{R} = 0$. The solution of SVD need to be first converted into a quaternion, and then be normalized to finalize the rotation matrix $\mathbf{R}$. Details can be found in [28]. The translation vector $\mathbf{t}$ is determined by $\mathbf{c}_p \equiv \mathbf{R}\mathbf{c}_q + \mathbf{t}$.

Moreover, the update of translation vector $\mathbf{t}$ is also conducted before the computation in every frame. Specifically, the centric positions of joints on the main body (i.e., excluding the joints on limbs and at head) are used to transform the rotated S_B to let its center coincident with the center of S_A.

In the initialization step, the lengths of bones are also computed and stored. The tracked joints of S_A and S_B are first placed to the average position, i.e.,

$$\mathbf{p}^*_i \leftarrow \frac{1}{2}(\mathbf{p}_i + \mathbf{R}\mathbf{q}_i + \mathbf{t}). \tag{13}$$

Then, lengths of bones on this averaged skeleton can be computed. The procedure of bone length initialization can be taken for a few seconds, and the averaged lengths of each bone during this period will be preserved in the later ACAP computation.

5 Results and Discussion

We have implemented the algorithm proposed in this paper by using Visual C++ and the Microsoft Kinect SDK library v1.6 [17] on Window 7 OS. The experiments are performed on real human motions captured by a setup of two Kinect cameras for Xbox360. All the experimental tests are run on a moderate PC with Intel Core i5 CPU 750 at 2.67GHz and 4GB RAM. Benefit from the efficiency of our algorithm, the proposed skeleton enhancement algorithm can generate the results at the speed of 5ms per frame in average, where the exact computation time is in the range from 1 to 7 ms. As a result, the CPU code can achieve a real-time skeleton extraction. Specifically, two different programs are developed in our prototype system. One is used to communicate with the Kinect sensor (K_A or K_B) and extract the skeleton (S_A or

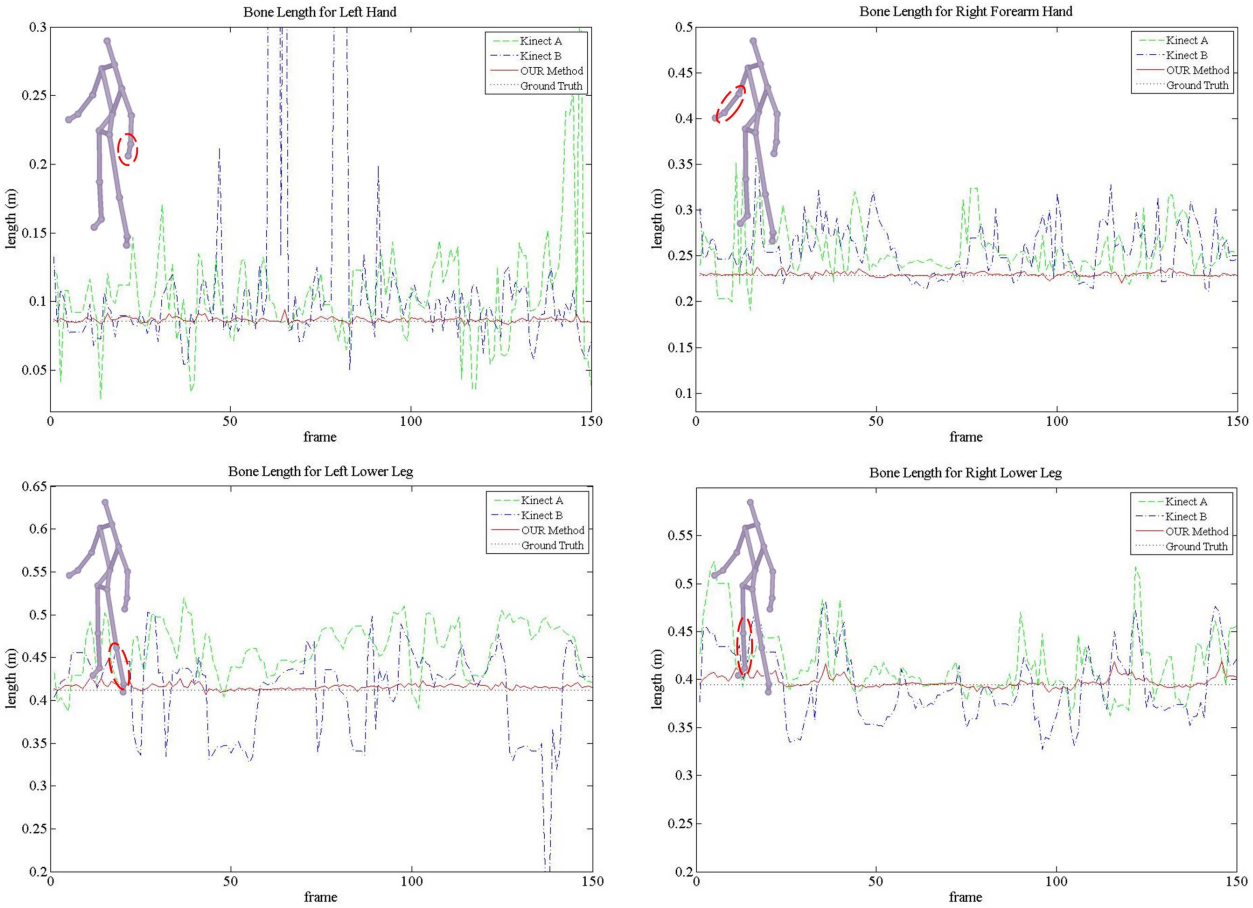

Figure 5: The statistics of bone-length variation at different parts of skeletons in a Badminton playing motion, where the green, blue and red curves are representing the bone-lengths of S_A, S_B and S^* respectively. The target bone-lengths, which are obtained from the initialization step, are displayed as a horizontal dot line in black.

S_B) by calling the functions provided by Microsoft Kinect SDK [17]. Limited by the functionality of Kinect SDK library, one program can only get the skeleton from one Kinect sensor. Therefore, this program has two copies running at the same time on the test platform. Another standalone program communicates with these two programs to collect the data of S_A and S_B, and implements the algorithm proposed in this paper to compute the enhanced skeleton S^*. The single-core CPU implementation of [29] provided by Microsoft Kinect SDK can extract S_A and S_B at around 30fps on two cores of CPU. As the program of our enhancement algorithm takes maximal 7ms for processing the skeleton in a frame, there is almost no delay in the experimental tests, the optimized skeleton S^* can still be obtained at 30fps in average. Therefore, the improved skeleton generated by our system can be used in many real-time applications.

There was a concern about the interference between multiple Kinects. Maimone et al. [15] propose a self-vibration method for reducing interference between multiple Kinects operating in the same spectrum. In motion extraction, Caon et al. [4] verify that the interference caused by two Kinects is insignificant for the skeleton tracking, and the skeleton remains nearly constant no matter how the relative position of the two cameras is.

Our first test shown here is conducted to check how significantly the algorithm proposed in this paper can improve the bone-length. As shown in Fig.5, the statistics and comparisons of bone-lengths are given at different parts of the skeletons extracted from a badminton playing motion (shown in Fig.6). After applying our algorithm, the lengths of bones are all very close to the ideal lengths which are set as the hard constraints in our ACAP optimization framework. In the other aspect, the bone-lengths of skeletons extracted from K_A and K_B independently varies significantly during the motion, and the length variations on S_A and S_B are not compatible to each other. The skeletons of a badminton playing motion are extracted from different frames and displayed in Fig.6 for the comparison. The skeletons processed by our approach can represent the movement of player more naturally. The skeletons in another motion of basket-ball playing are show in Fig.7. More demos can be found in the supplementary video of this paper.

Limitations

The major limitation of this approach is that the algorithm is based on the initial skeletons, S_A and S_B, generated by

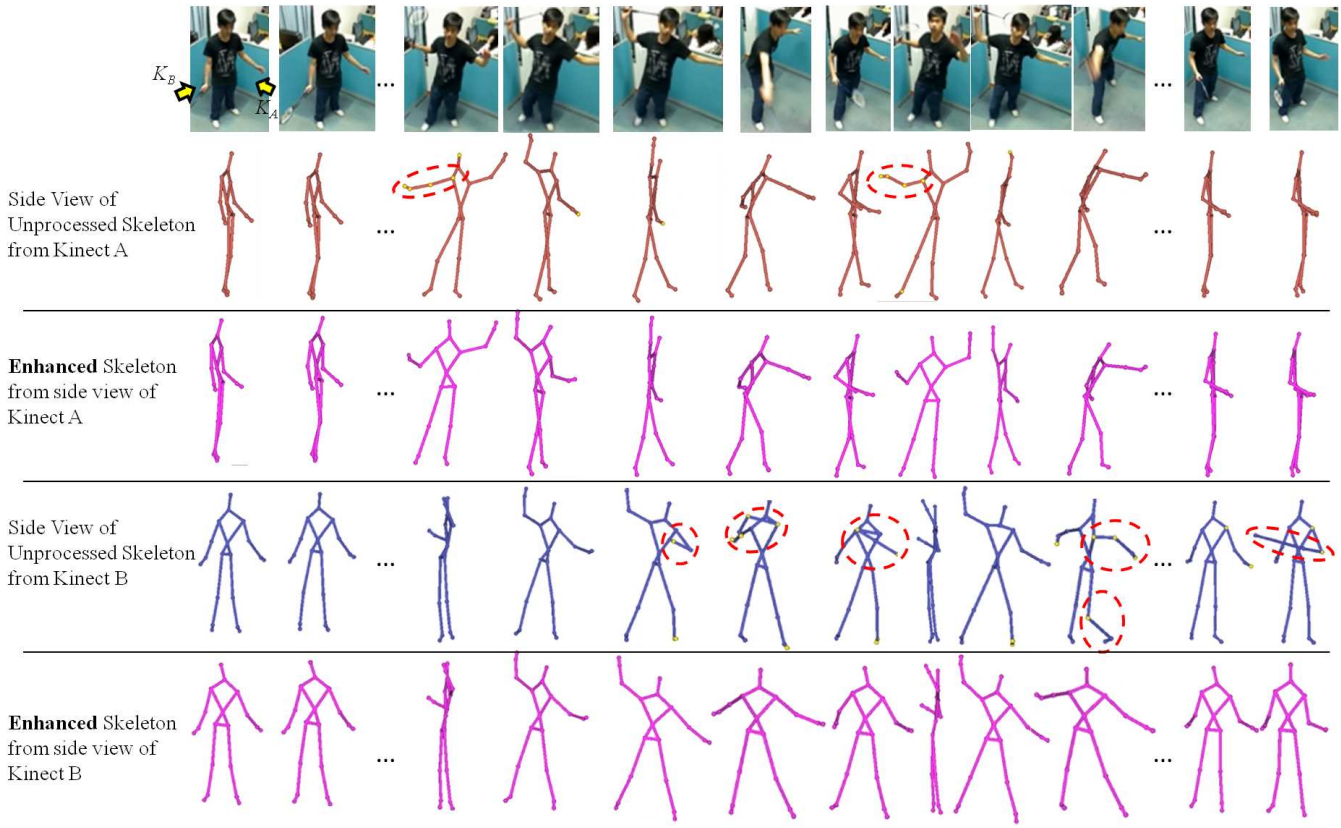

Figure 6: The motion of badminton playing: the enhanced skeletons, S^*, generated by our algorithm are displayed in purple (at the third and the fifth rows), the skeletons generated by Microsoft Kinect SDK, S_A and S_B, are displayed in brown (at the second row) and blue (at the fourth row) respectively. In our tests, we also use a video camera to capture the motion (shown in top row) so that the real motion can be illustrated more clearly. The orientations of two Kinect cameras, K_A and K_B, are also illustrated in the first row – see the yellow arrows.

Microsoft Kinect SDK. In the scenario neither of the two views can generate reliable joint positions (as illustrated in Fig.8), it has difficulty to retrieve correct joint positions.

On the other hand, the correctness of motion extraction by Kinect SDK depends heavily on the database of Kinect motions; therefore, some motions (e.g., squatting shown in Fig.9) cannot be estimated as good as other existing motions in the database. In our future work, we plan to construct a supplementary motion database to cooperate the motion extraction with the motion database of Kinect SDK.

Moreover, we also find that the positions of hands and feet generated by Kinect SDK can have unwanted vibration that cannot be eliminated by our algorithm. Nevertheless, this type of vibration doesn't affect the overall motion of our result.

At last, since the technique used to generate S_A and S_B is per-frame estimation based, it does not provide the ability to distinguish the front/back directions of a human body. This difficulty can be overcome by adding another sensor or attaching one special marker on human body (e.g., at the chest or on one side of the shoulder). Once this special marker can be identified in the images captured by RGB-D cameras, the orientation of a human body can be identified.

6 Conclusion

This paper presents an efficient method to enhance the skeletons of human motion, which can be extracted per-frame with the help of RGB-D cameras such as Kinect. This development leads to an economic device to provide human motion as input for virtual reality systems. The skeleton extracted by a single RGB-D camera using Microsoft Kinect SDK usually has problems of unwanted vibration, bone-length variation, self-occlusion, etc. We develop an approach in this paper to overcome these problems by synthesizing the skeletons generate by duplex Kinects, which capture the human motion in different views. Specifically, we did not change the motion extraction method; but we use two per-frame extracted skeletons as input to generate an optimized skeleton on-site. The major technical difficulty comes from how to evaluate the reliability of each joint's positions reported by two cameras, and how to resolve the inconsistency. Our algorithm is formulated under the constrained optimization framework by using the bone-lengths as hard constraints and the tradeoff between inconsistent joint positions as soft constraints.
In summary, the following advantages can be provided by our approach.

- We develop an approach that can preserve the bone-

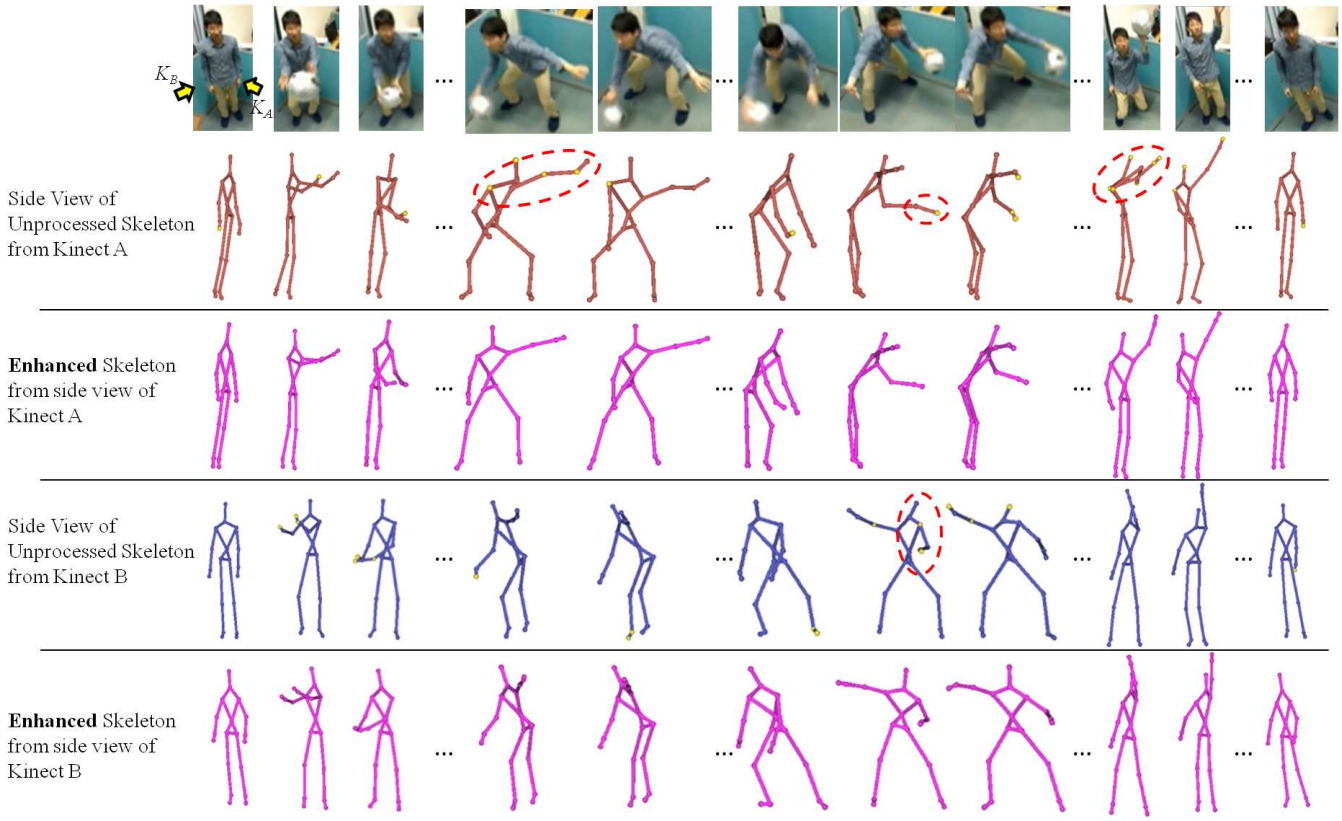

Figure 7: The motion of basket-ball playing: enhanced skeletons in the motion are displayed in the third and fifth rows along the same viewing direction of Kinect cameras (i.e., K_A and K_B), which are shown in the second and the fourth rows. The problematic skeletons in the motion extracted by K_A and K_B independently are circled by red dash lines.

length when taking the on-site skeleton tracking.

- We develop a method to efficiently evaluate the reliability of joint positions that can solve the inconsistent problem of joint positions under our optimization framework.
- A simple but effective method has been provided to take care of the calibration of two Kinect's coordinate system automatically.

All these benefits lead to a low-cost RGB-D camera-based input device for real-time interactive applications. Our future research will focus on how to further enhance the accuracy of this device and overcome the current limitations.

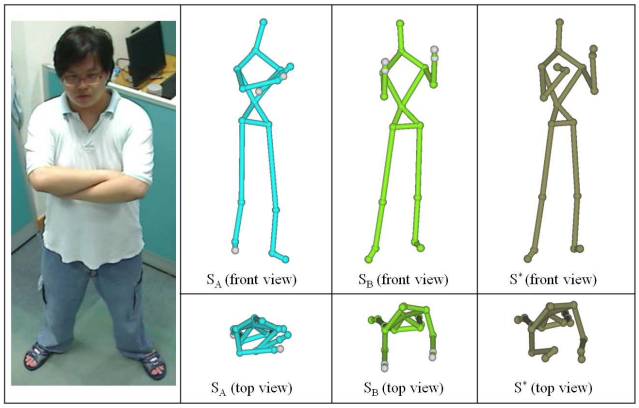

Figure 8: An illustration for limitation: the wrist joints and the elbow joints are hardly separated from the main body in both views of Kinect cameras; therefore, the resultant joint position may not be fixed correctly.

Acknowledgement

This work was partially supported by the Hong Kong RGC/GRF Grant (CUHK/417508 and CUHK/417109) and the Direct Research Grant (CUHK/2050518).

References

[1] Activate3D. Activate3d's intelligent character motion (icm). www.http://activate3d.com/.

[2] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *IEEE 13th International Conference on Computer Vision (ICCV)*, pages 1092–1099, 2011.

[3] E. R. Bachmann, R. B. McGhee, X. Yun, and M. J. Zyda. Inertial and magnetic posture tracking for insert-

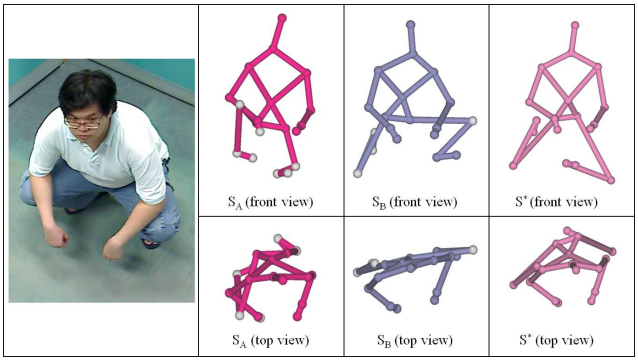

Figure 9: The squatting pose does not included in the database of Kinect SDK, and the positions estimated by Kinect SDK is not good. As a result, our approach cannot generate a reasonable skeleton with those input.

ing humans into networked virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 9–16, 2001.

[4] M. Caon, Y. Yue, J. Tscherrig, E. Mugellini, and O. A. Khaled. Context-aware 3D gesture interaction based on multiple kinects. In *Proceedings of The First International Conference on Ambient Computing, Applications, Services and Technologies*, AMBIENT 2011, Barcelona, Spain, 2011.

[5] M. Cavazza, R. Earnshaw, N. Magnenat-Thalmann, and D. Thalmann. Motion control of virtual humans. *IEEE Comput. Graph. Appl.*, 18(5):24–31, 1998.

[6] J. C. P. Chan, H. Leung, J. K. T. Tang, and T. Komura. A virtual reality dance training system using motion capture technology. *IEEE Trans. Learn. Technol.*, 4(2):187–195, 2011.

[7] E. Foxlin and M. Harrington. Weartrack: A self-referenced head and hand tracker for wearable computers and portable VR. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, pages 155–162, 2000.

[8] S. Hauswiesner, M. Straka, and G. Reitmayr. Free viewpoint virtual try-on with commodity depth cameras. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, pages 23–30, 2011.

[9] M. Hazas and A. Ward. A novel broadband ultrasonic location system. In *Proceedings of the 4th international conference on Ubiquitous Computing*, pages 264–280, 2002.

[10] E. S. L. Ho, T. Komura, and C.-L. Tai. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph.*, 29(4):33:1–33:8, 2010.

[11] M. B. Holte, C. Tran, M. M. Trivedi, and T. B. Moeslund. Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):538–552, 2012.

[12] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[13] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Trans. Graph.*, 29(4):102:1–102:12, 2010.

[14] L. Li, J. McCann, N. Pollard, and C. Faloutsos. BoLeRO: a principled technique for including bone length constraints in motion capture occlusion filling. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 179–188, 2010.

[15] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *2012 IEEE Virtual Reality Conference*, pages 51–54, march 2012.

[16] Measurand. Shapewrap. http://www.metamotion.com.

[17] Microsoft. Microsoft kinect for windows SDK. www.microsoft.com.

[18] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90–126, 2006.

[19] M. Motion. Gypsy. http://www.motion-capture-system.com.

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[21] L. J. Olson, E. and S. Teller. Robust rangeonly beacon localization. *Journal of Oceanic Engineering*, 31(4):949–958, 2006.

[22] Z. Pan, W. Xu, J. Huang, M. Zhang, and J. Shi. Easybowling: a small bowling machine based on virtual simulation. *Computers & Graphics*, 27:231–238, 2003.

[23] T. Piazza, J. Lundström, A. Hugestrand, A. Kunz, and M. Fjeld. Towards solving the missing marker problem in realtime motion capture. In *Proceedings of ASME 2009 IDETC/CIE Conference*, pages 1521–1526, 2009.

[24] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. In *2010 IEEE International Conference on Robotics and Automation*, pages 3108–3113, 2010.

[25] R. Poppe. Vision-based human motion analysis: An overview. *Comput. Vis. Image Underst.*, 108(1-2):4–18, 2007.

[26] C. Rose, B. Bodenheimer, and M. F. Cohen. Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications*, 18:32–40, 1998.

[27] S. Rusinkiewicz, B. Brown, and M. Kazhdan. 3D scan matching and registration. ICCV 2005 Short Course, http://www.cs.princeton.edu/ bjbrown/iccv05_course/.

[28] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Computer Graphics*, 19:245–254, 1985.

[29] 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 *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

[30] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18:643–650, 2012.

[31] Vicon. Vicon motion capture system. www.vicon.com.

[32] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović. Practical motion capture in everyday surroundings. *ACM Trans. Graph.*, 26(3), 2007.

[33] A. Weiss, D. Hirshberg, and M. Black. Home 3d body scans from noisy image and range data. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1951 –1958, nov. 2011.

[34] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 273–282, 2010.

[35] H. J. Woltring. New possibilities for human motion studies by real-time light spot position measurement. *Biotelemetry*, 1(3):132–146, 1974.

[36] Xsens. Xsens MVN. www.xsens.com.