Non-Linear Skeletal Fusion with Multiple Kinects for Unified Skeletal Animation Reconstruction

Naveed Ahmed

***Abstract*—We present a new method for reconstructing a unified skeletal animation with multiple Kinects over 360 degrees using a non-linear skeletal fusion function. Using the skeletal data from multiple Kinects, we use dynamic programming to find the global optimal solution that selects the joints from each camera to create a unified skeleton at each frame such that it results in the correct pose. The unified skeletal reconstruction is constrained by a number of terms that take into account the orientation of the joints, bone lengths, and the temporal smoothness of the joint's motion. We quantitatively validate the goodness of the unified skeleton using two evaluation methods, and also perform qualitative and quantitative comparisons with the sate-of-the-art methods in skeletal reconstruction.. The output of our method is a 360-degree plausible unified skeletal animation that would not be possible with a single Kinect due to occlusions, tracking failures, and field of view constraints.**

***Index Terms*—3D Animation, Kinect, Multi-view Video, Motion Capture, 3D Reconstruction**

I. Introduction

The field of human motion capture has been an active area of research both in computer vision and computer graphics. Specifically, in the past two decades there has been a special focus on the marker-less motion capture. Motion capture in general has a number of applications in a number of areas, e.g., movies, games and robotics etc. Marker-less motion capture on the other hand opens new avenues of application as it can capture a user's motion in a general environment without the need of any special attachment. It has a number of applications in the areas such as natural user interface design, motion analysis, video surveillance, virtual reality etc. In addition to the motion, it also allows to capture the shape, and appearance of the person that can be employed for additional scene analysis and visualizations. One of the earliest work in this area was done by Carranza et al. [1]. They used eight synchronized RGB video cameras and used an analysis-based-synthesis method for the marker-less motion capture. Later, Theobalt et al. [2] extended their work to not only capture the motion, but also the surface reflectance properties. Debevec et al. [3] [4] also captured different aspects of human motion and appearance. Later Aguiar et al. [5] used a template based deformation method to capture the motion of a moving actor, whereas Vlasic et al. [6] used a skeleton based deformation approach to capture non-trivial motion, and Ahmed et al. [7] used shape matching of reconstructed visual hulls to capture the motion. All of these methods used RGB cameras to capture the moving person.

Microsoft introduced Kinect in 2010 as a natural user interface device [8]. It has been widely adopted as a low-cost depth sensor for the acquisition of static or dynamic 3D content. The major benefit of Kinect is that it provides both color and depth data at 30 frames per seconds. Depth data can also be acquired through other type of depth cameras [9] [10], but the main benefit of using Kinect is, that it provides both color and depth data in a single system at a very low price. In addition, Microsoft provides a very comprehensive SDK to access all the functionality of Kinect that makes it a ubiquitous choice for 3D acquisition.

Kinect, being a consumer grade RGB-D camera, is deployed in a diverse range of applications. Some of the examples include: Gesture recognition for novel interfaces [11], human behavior recognition [12], time-coherent 3D animation reconstruction [13], or head tracking for virtual reality applications [14]. Additionally, Kinect [8] has emerged as a standard choice for pose estimation. Since it provides both RGB and depth data, a single Kinect can be used to estimate the pose of the human actor [15] [16] [17]. In addition to the color and depth data, Kinect SDK [8] also allows access to the real-time pose data. Real-time pose estimation with the Kinect SDK is one of its main strength, resulting it being employed in a number of applications ranging from video games to the controlling of robots [18]. Microsoft has been constantly improving the Kinect SDK, which can provide real-time pose estimation of a person in standing or sitting positions.

There has been a number of methods proposed for pose estimation using depth cameras [19]. Many of these methods can be used to get a real-time pose estimation. The benefit of using Kinect SDK is that it is simple to use, and using it makes implementing a motion capture system a simplified process. In contrast to the pose estimation approaches by Wei et al.[20], Ye et al. [16], Baak et al.[17], Yasin et al. [21], Shotton et al. [22], Yueng et al. [23], and Dantone et al. [24], which though may well be more robust, are very difficult to implement for a general user and their adaption rate for motion capture application is very low. On the other hand, pose estimation from Kinect has been employed in a number of applications in many areas [18].

In general, for all the applications that rely on the skeletal data from one Kinect, this single view pose estimation is reliable as long as the user is oriented towards the camera with minimal occlusions [25]. This type of pose estimation works most of the time while geared towards the player facing the television, but it cannot be used for a free-form motion capture over 360 degrees, where the actor can move in any direction. One of the main reason for the pose estimation failure is the occlusion of body parts resulting in the missing depth information.

A straightforward solution for resolving the occlusions would be to use more than one Kinect. Recently, a number of methods have been proposed that make use of multiple Kinects for the pose estimation problem. Even though this

Manuscript received August 31, 2017; revised November 7, 2017

N. Ahmed is with the Department of Computer Science, University of Sharjah, Sharjah, 27272, UAE. E-mail: nahmed at sharjah.ac.ae

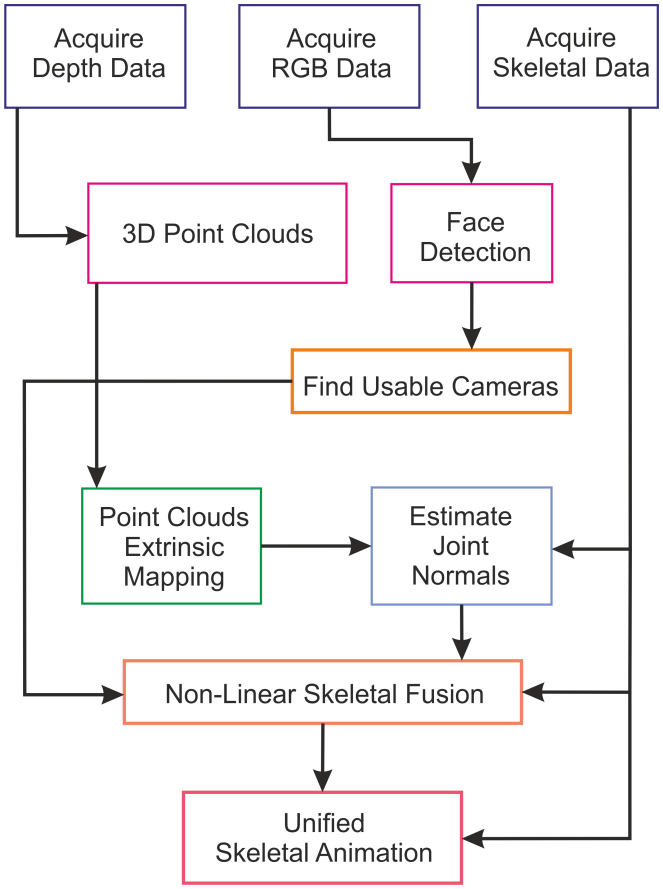

Fig. 1. Flowchart of the proposed method, starting from the acquisition of depth, RGB, and skeletal data to the unified skeletal animation reconstruction.

solves the occlusion problem, but recording with more than one Kinect introduces interference, resulting in the depth data loss. In principle, this is not a big limitation, because the depth data missing from one Kinect could be filled in by the other. Ahmed [26] showed an acquisition system comprising of six Kinects for the 360 degrees acquisition and 3D animation reconstruction. Berger et al. [27] employed four Kinects for unsynchronized marker-less motion capture. Ye et al. [28] used three hand-held Kinects for marker-less performance capture, Yueng et al. [23] employed two Kinects, whereas Caputo et al. [29] employed multiple Kinects for hand gesture recognition. None of these methods used the skeletal data provided by Kinect to create the final pose, but the pose was estimated using an optimization process based on silhouettes, human template matching, or skeleton-based constraints. One of the major benefits of directly using the Kinect-based skeleton is that it does not require any post-processing and is available in real-time along with the RGB and depth data. As shown by the study of Obdrzalek et al. [25], the skeleton data from Kinect compares favorably with established pose estimation techniques and can be reliably used in a number of scenarios as long as the occlusions are minimal, and the actor is facing the camera. Therefore, in this work, we propose a new method that will reconstruct a unified pose over 360 degrees only using the best available joint positions provided by Kinect to maximize the efficiency.

There are a number of challenges in incorporating multiple Kinects and fusing their skeleton data over 360 degrees. First, the Kinect cannot differentiate between the front-facing and the back-facing person. An incorrect inverted posture for all joints is returned for the back-facing person. If one captures an actor over 360 degrees, then there should be a method to automatically detect and discard the incorrect pose. Secondly, there should be a way to fuse the data from the joints that are not occluded as the tracking result from the occluded joints can be completely wrong. In addition, due to the underlying algorithm from Kinect, the tracking can fail due to the sensor noise or very fast motion. Thus there should be a way to fuse the skeleton data from multiple Kinects in such a way so that it can rectify these failures using the best available joints.

Therefore, we propose a new method of fusing the skeleton data from multiple Kinects over 360 degrees. Our method can automatically detect the correct orientation of the actor with respect to each camera, and can fuse the joint data based on our novel non-linear skeletal fusion function that creates a unified skeletal representation at each frame. Our method uses the Microsoft Kinect SDK for acquisition and its implementation is relatively very simple. The result of our method is a unified human motion measurement in the form of a skeletal animation over 360 degrees that is free from the artifacts due to occlusions or tracking failures. Our work does not estimate the pose from the depth data, rather it presents a very simple and effective method to combine the data acquired from multiple low-cost sensors for a reliable 360 degrees motion capture. An algorithmic flowchart of our method can be seen in Fig. 1. In the following sections, we will present each of the algorithmic steps in detail, starting from the discussion of data acquisition in the next section. Afterward, the unified skeletal animation reconstruction algorithm is presented, followed by results and validation, and conclusions.

II. Data Acquisition

Our acquisition system is comprised of four Kinects placed at 90 degrees with respect to each other. Our system is not confined to a fixed camera setup, but can work effectively for a hand-held acquisition, if required. We use a software-based synchronization similar to Ye et al. [28] for the multi-view acquisition. We use the Kinect SDK to acquire RGB, depth and skeleton data. RGB-D streams from Kinect are low resolution (640x480) at 30 frames per second. For each frame, Kinect tracks a skeleton comprising of 20 joints. One frame from our acquisition system showing, RGB, depth and the skeleton data can be seen in Fig. 2a, b.

One of the benefits of using the Kinect SDK is that it circumvents the need of any manual intrinsic camera calibration. The SDK provides the mapping between RGB, depth, and skeleton data. It also maps the depth and skeleton data to a unified three-space coordinate system. Thus, for every depth value the corresponding RGB value is available. Additionally, for every joint position we know its depth value and the mapping to the RGB data. For our work, we only need the mapping between depth and the skeleton data.

The depth to world coordinate mapping allows us to resample the depth data in a 3D point cloud. Thus, for each frame we obtain four 3D point clouds along with their corresponding estimated skeleton data in their local coordinate systems. It is to be noted that we do not simplify the 3D

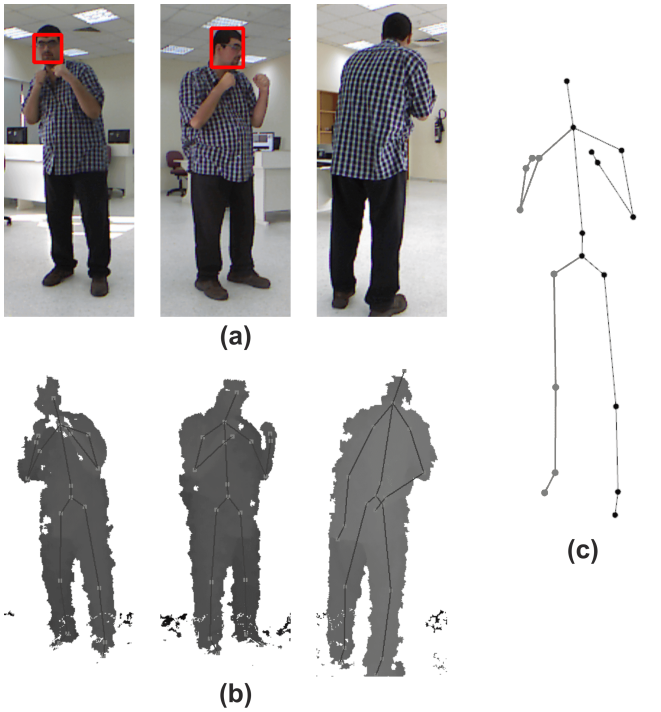

Fig. 2. (a) shows RGB frames from three cameras. Frontal and profile faces are detected in two cameras. (b) shows the depth data with the overlaid skeleton from Kinect. (c) shows the unified skeleton from the two cameras (shown in different shades of grey) towards which the actor's face is oriented.

point clouds to remove the noise or outliers [30], rather we rely on the raw data to speed up the runtime performance. In addition, the Kinect SDK also provides a tracking state for the skeleton and each joint. The joint tracking states are an important part of the error measure while finding the optimal solution using the non-linear fusion function, as discussed in the next section. As our experiments use a static camera setup, we use the same approach as Ahmed [26] for the extrinsic calibration. In general, a dynamic calibration approach can be used to perform the extrinsic calibration at each frame as long as some correspondences between different cameras can be established. Therefore, our method is not limited to a fixed camera setup. 3D point clouds, and their corresponding skeletal data registered in a global coordinate system can be seen in Fig. 3.

III. Unified Skeletal Animation Reconstruction

The fusion of skeleton data from multiple Kinects poses a number of challenges. First, the skeleton data from the Kinect is not usable if the actor is not facing the camera. The Kinect uses the depth data under the assumption that the actor is facing the camera and returns the incorrect pose if the actor is not facing the camera, as seen in Fig. 2b(right). In the first step, for every frame we need to identify which cameras can be used for reconstructing the unified skeleton. As the depth data, or the skeleton and joins tracking states are not helpful in finding the correct orientation of the human actor, we use one of the standard face detection methods [31] over the RGB data to determine the front-facing actors. We use two profiles, one for the frontal face, and one for the profile face to find out which cameras can be used for the fusion (Fig. 2a). Face detection is a standard feature provided

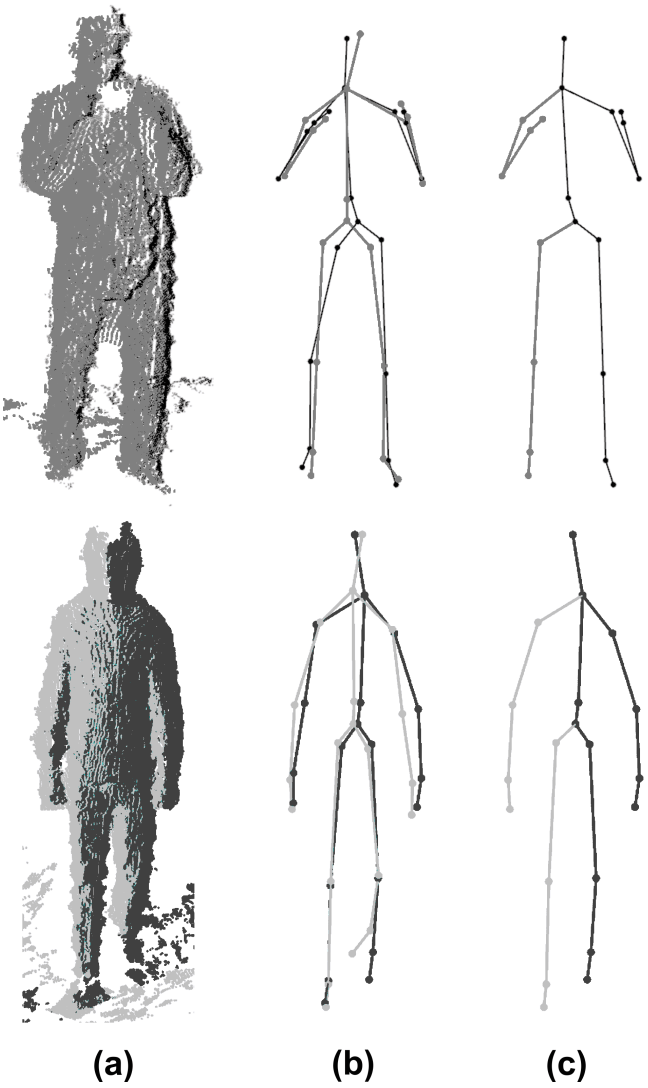

Fig. 3. (a) shows unified two point clouds (shown in different shades of grey), and (b) shows the corresponding two skeletons after the extrinsic calibration. (c) shows the unified skeleton reconstructed from our method.

in nearly all camera systems, ranging from mobile phones to high end DSLRs. It is prone to failure if the actor's face is occluded. Sometimes it can also detect false positives. We used simple sanity checks to circumvent these issues, to be discussed in the Results and Validation section. Additionally, we could also use the face detection API provided with the Kinect SDK, which works robustly in practice, but since it is real-time, we found that it adversely affected the performance of our acquisition system. In principle, as the Kinect already provides the head position in the depth image coordinates, the extrinsic camera parameters can be used to localize the head position in the RGB space. Using the head position, some other image processing algorithm can also be used to detect the front-facing camera.

Once the cameras to be used are identified, we start the fusion process by selecting the 20 joints from the usable cameras that result in the correct pose. Identifying the correct joints can be a challenging problem, and one can opt for a local solution where each joint is selected individually based on some confidence score, or opt for a global solution that takes into account all the joints at the same time. The benefit

of a global solution is that it can employ additional properties of the skeleton, e.g. bone lengths to enforce pose consistency. Therefore, we formulate a non-linear fusion function that finds the optimal skeleton in a global solution. Assuming we are using $\mathcal{C}$ usable cameras, and there are $\mathcal{T}$ frames in the sequence. A skeleton $\mathcal{S}_t$ is defined by the 20 joints j_t^c, where c=1, ..., $\mathcal{C}$, and t=1, ..., $\mathcal{T}$. It is to be noted that within a skeleton the choice of the joints is local, i.e., the right knee joint can only be one of the right knee joints from one of the usable cameras. Thus, by definition $\mathcal{S}_t$ is always a valid skeleton, though its pose could be incorrect based on the choice of the joints. Under these conditions, at any time frame t, the number of possible valid skeleton configurations for $\mathcal{S}_t$ are $\mathcal{C}^{20}$. The non-linear fusion function for each frame is defined as:

$$f(\mathcal{S}_t) = \alpha\Phi(S_t) + \beta\Psi(S_t) + \gamma\Omega(S_t) \tag{1}$$

where $\Phi(S_t)$, $\Psi(S_t)$, and $\Omega(S_t)$ are the error measures terms related to the orientation, bone length, and the temporal smoothness, respectively. These error measures are evaluated for each configuration of $\mathcal{S}_t$, and the joint configuration that results in the minimal value for Eq. 1 is selected as the unified skeleton at that particular time step. α, β, and γ are weighting factors summing to 1.0. Empirically, we found the values of α = 0.46, β = 0.33, and γ = 0.21 (see also the Results and Validation section for a discussion). In the following subsections, we will describe the each error measure term in detail.

A. Orientation Measure

The orientation error measure $\Phi(S_t)$ penalizes the skeleton whose joints are not oriented towards the camera. In order to calculate this measure, we first need to estimate the normal $\mathcal{N}(j_t^c)$ for each j_t^c. $\mathcal{N}(j_t^c)$ is estimated in the global coordinate system using the unified 3D point clouds that is obtained using the extrinsic camera calibration Fig 3a. For each j_t^c, using its three-space position $\mathcal{P}(j_t^c)$, the standard SVD-based plane-fitting algorithm on nearest 20 three-space points in the unified point cloud is used to obtain $\mathcal{N}(j_t^c)$.

The orientation measure $\varphi(j_t^c)$ for each j_t^c is defined as the dot product of :

$$\varphi(j_t^c) = \mathcal{N}(j_t^c) \cdot \mathcal{V}(j_t^c).$$

where $\mathcal{V}(j_t^c)$ is the view vector from $\mathcal{P}(j_t^c)$ to the three-space position of the camera c. Thus, if the joint is completely oriented towards the camera then $\varphi(j_t^c)$ will be 0, otherwise its value will increase. The joint is discarded if $\varphi(j_t^c)$ is less than 0. This is desirable for our error measure, as the joints that face the camera will result in the lower error compared to the joints that are not oriented towards the camera.

Finally for a skeleton $\mathcal{S}_t$, the orientation error measure term $\Phi(S_t)$ for the non-linear fusion function (Eq. 1) is defined as the sum of $\varphi(j_t^c)$ over 20 joints scaled by the joint confidence term $\mathcal{R}(j_t^c)$ and the occlusion term $\mathcal{O}(j_t^c)$:

$$\Phi(S_t) = \sum_{j=1}^{20} \varphi(j_t^c) * \mathcal{R}(j_t^c) * \mathcal{O}(j_t^c) \tag{2}$$

The two additional terms $\mathcal{R}(j_t^c)$ and $\mathcal{O}(j_t^c)$ that scale $\varphi(j_t^c)$ are required to compensate for the shortcomings of the Kinect's underlying skeletal estimation algorithm. It is to be noted that we do not estimate a completely new joint position using the skeletal data from multiple Kinects, rather as explained earlier, the best joints from one of the skeletons is selected to create the unified skeleton. As Kinect estimates a skeleton in real-time using the depth data, the skeleton tracking can fail for one more joints due to a number of reasons. Most common cause for a tracking failure is the occlusion, fast motion, or sensor noise, resulting in missing or low quality of the depth data. Microsoft Kinect SDK provides a joint tracking state for each j_t^c. There are three possible tracking states that are used to define $\mathcal{R}(j_t^c)$ as follows:

$$\mathcal{R}(j_t^c) = \begin{cases} 3 & \text{if the joint data is not available,} \\ 2 & \text{if joint data is calculated from other joints,} \\ 1 & \text{if the joint data is tracked and available.} \end{cases}$$

If the joint data is tracked and available then there is no penalty. If the joint data is calculated from other joints or is simply not available, due to the missing depth information, then the reliability of the joint data is low and the orientation error measure for that particular joint is scaled accordingly. The missing depth information can be due to occlusion, fast motion, or sensor noise. Thus $\mathcal{R}(j_t^c)$ favors the correctly tracked joints.

In addition to $\mathcal{R}(j_t^c)$, we also introduce a complimentary term $\mathcal{O}(j_t^c)$ that specifically checks for joint's occlusion to further penalizes the joints that are occluded by some other body parts. Joint occlusion is one of the most common reasons for tracking failure for a particular joint. We cannot rely on the joint tracking state $\mathcal{R}(j_t^c)$ to report for occlusion because its underlying algorithm for the tracking state is unknown, rather we check for the joint occlusion using our own algorithm.

$\mathcal{O}(j_t^c)$ is calculated by finding if j_t^c is occluded or not by back projecting its three-space value from the skeleton to the depth image, giving its two-space $(j_t^c, x, y)_{back}$ location in the depth coordinates. A lookup in the depth image using $(j_t^c, x, y)_{back}$ provides the back projected depth value $\mathcal{D}_{back}(j_t^c)$. In addition, the Kinect SDK also provides the actual depth value for each joint in the depth space $\mathcal{D}_{SDK}(j_t^c)$. If j_t^c is not occluded then $\mathcal{D}_{back}(j_t^c)$ should be equal to $\mathcal{D}_{SDK}(j_t^c)$, otherwise it must be occluded. Thus $\mathcal{O}(j_t^c)$ is defined as follows:

$$\mathcal{O}(j_t^c) = \begin{cases} 2 & \text{if } \mathcal{D}_{back}(j_t^c) \neq \mathcal{D}_{SDK}(j_t^c), \\ 1 & \text{if } \mathcal{D}_{back}(j_t^c) = \mathcal{D}_{SDK}(j_t^c) \end{cases}$$

There is no penalty if the joint is not occluded otherwise the error is scaled. The term $\mathcal{O}(j_t^c)$ complements $\mathcal{R}(j_t^c)$. For example, in some cases Kinect can estimate a skeleton that have a higher quality with all joints having a higher confidence, whereas the actual subject has occlude body parts from the point of view of the camera. In this scenario, the occlusion term penalizes the joints that are occluded so that joints that are not occluded are assigned a higher score.

B. Bone Length Measure

The bone length error measure $\Psi(S_t)$, penalizes the skeletons configuration that deviate greatly from the ideal bone lengths. In general, there is no notion for ideal bone length for a person as they differ for every person. We adopt the approach similar to [23], and initialize all the bone-lengths manually within a unified skeleton for the first frame and classify their lengths as the ideal lengths. This is only done at the first frame and provides a goodness measure that will later also be used for the quantitative validation of the method.

A joint can be associated with one or more bones. Assuming it can be associated with n bones, we define $\mathcal{B}_i(j_t^c)$ to be the bones associated with j_t^c, where i=1...n. The ideal bones are defined as $\mathcal{B}_i(j_0)$. Using these definitions, the joint's bone length error measure $\psi(j_t^c)$ is defined as:

$$\psi(j_t^c) = \sum_{i=1}^{n} (||\mathcal{B}_i(j_t^c)|| - ||\mathcal{B}_i(j_0)||)^2$$

which is the sum of the square of the difference between the bone lengths associated with each j_t^c and their corresponding ideal bone length. This error measure will be closer 0 if the bone lengths are closer to the ideal bone lengths, otherwise its value will increase. It is to be noted that $\psi(j_t^c)$ is only defined for a valid skeleton, it cannot be calculated for a joint individually. Therefore, for a skeleton $\mathcal{S}_t$, the bone length error measure term $\Psi(S_t)$ for the non-linear fusion function (Eq. 1), is defined as:

$$\Psi(S_t) = \sum_{j=1}^{20} \psi(j_t^c) * \mathcal{R}(j_t^c) * \mathcal{O}(j_t^c) \tag{3}$$

Similar to the orientation error measure, the bone length error measure $\psi(j_t^c)$ is scaled by the joint confidence term $\mathcal{R}(j_t^c)$ and the occlusion term $\mathcal{O}(j_t^c)$. These terms as explained in the previous section, penalize the joints that either have a lower quality of tracking data or they are occluded by some other body parts.

C. Temporal Smoothness Measure

The temporal smoothness error measure $\Omega(S_t)$ penalizes the joints that move abruptly compared to the previous frame. As we rely on Kinect to reconstruct the skeleton, the underlying pose estimation algorithm from Kinect estimates a skeleton at each frame that can behave very differently from the previous frame due to a number of reasons. As we fuse the skeleton data from multiple Kinects, the joints that depict smoother motion should be preferred to the jerky motion as it can be due to a tracking failure or the limitation of the underlying algorithm. Given the three-space position of a joint at the current frame $\mathcal{P}(j_t^c)$ and at the previous frame $\mathcal{P}(j_{t-1}^c)$, its temporal smoothness error measure $\omega(j_t^c)$ is defined as the square of the distance between the two positions:

$$\omega(j_t^c) = (||\mathcal{P}(j_t^c) - \mathcal{P}(j_{t-1}^c)||)^2$$

Please note that Kinect's coordinate system is in meters, thus in general the displacement is always small and less than 1 meter, and this error measure only penalizes very high displacement that results in a sudden jerky motion. As we use the joint positions provided by the Kinect SDK, $\omega(j_t^c)$ is never 0, rather it penalizes if a joint deviates too much from the last frame, especially in case of a tracking failure. Therefore, for a skeleton $\mathcal{S}_t$, the temporal smoothness error measure term $\Omega(S_t)$ for the non-linear fusion function (Eq. 1) is defined as:

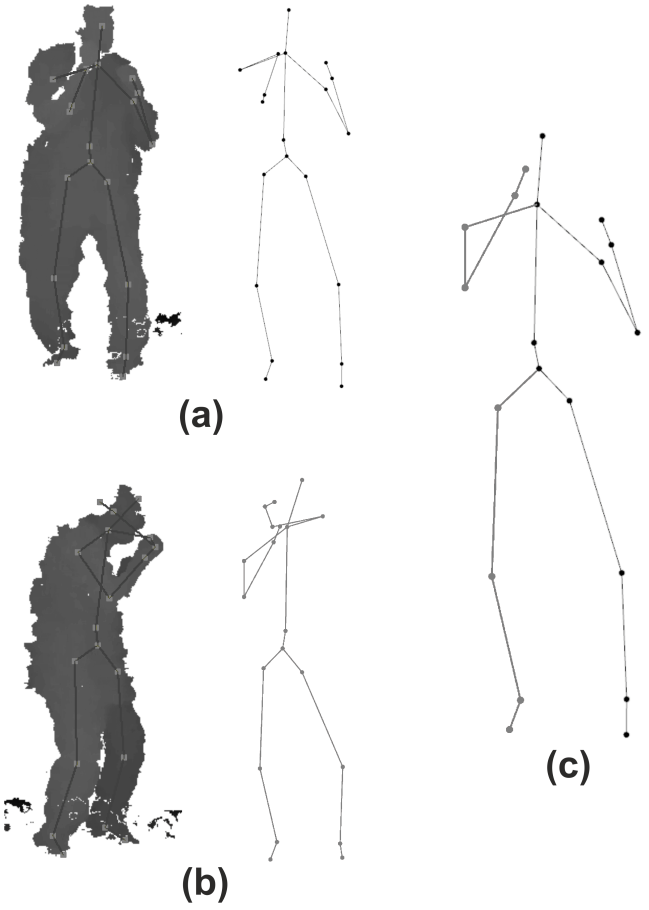

Fig. 4. Tracking fails for the right arm in the front-facing camera due to self-occlusions (a), whereas for the side facing camera the left arm was not tracked because it was not in the view (b). Our method unified the two skeletons (shown in different shades of grey) and reconstructed the correct pose (right).

$$\Omega(S_t) = \sum_{j=1}^{20} \omega(j_t^c) * \mathcal{R}(j_t^c) * \mathcal{O}(j_t^c) \tag{4}$$

Similar to the previous error measures it is again scaled by the joint confidence term $\mathcal{R}(j_t^c)$ and the occlusion term $\mathcal{O}(j_t^c)$. Please note that $\Omega(S_t)$ cannot compensate for the jerkiness if it is present in all skeletons at a particular frame. We do not estimate a new joint position, rather select the best joint from the set of available joints from usable cameras. Even if all joints depict a jerky motion, this term will favor the one with the smoothest motion.

Given the non-linear fusion function, starting from the second frame, we estimate the optimal skeleton by selecting the best 20 joints that minimize Eq. 1 using the dynamic programming. The reconstructed unified skeleton at each frame results in a unified skeletal animation that uses best joints from each Kinect skeleton. The results of our method along with its validation are discussed in the next section.

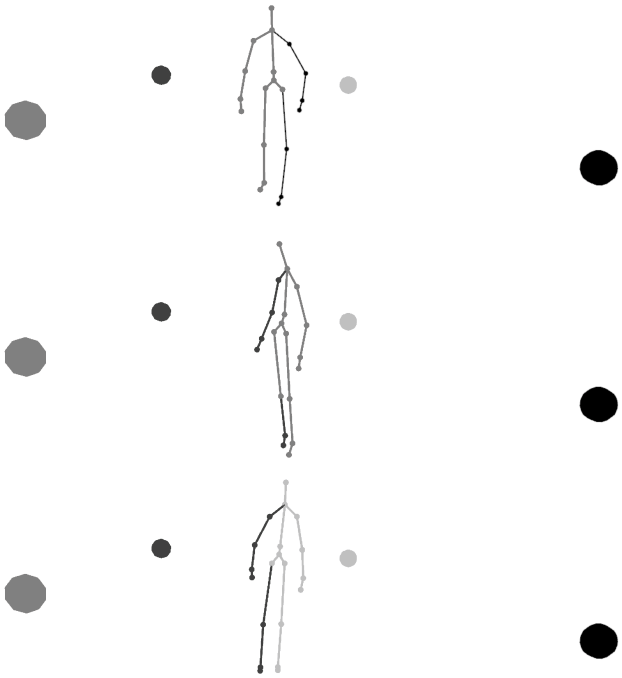

Fig. 5. Three frames of the walking sequence show the fusion of different cameras over 360 degrees. The cameras are shown in different shades of grey, and their three-space location is visualized as circles. Depending on the orientation of the actor, the usable cameras are identified that are used to create the unified skeleton using the non-linear fusion function.

IV. Results and Validation

In order to test our method, we used four Kinect sensors to record three sequences of 200 frames each. The first sequence shows a fast boxing motion, the second sequence is a normal walking motion, while the third sequence is the fast rotation motion of the whole body, while the actor moves the arms and legs on the spot. The results show that the non-linear fusion solution selected the 20 best joints at each frame and managed to reconstruct a unified pose that is free from the failures of individual cameras. The three error measure terms ensure that joints with the wrong pose are replaced by the joints from other cameras that estimate the correct pose, as can be seen in Fig. 4. More results from two of the sequences can be seen in Fig. 2c, 3c, 5. It can be observed in the results that our method can merge the skeleton data from multiple cameras to reconstruct the unified skeletal animation. Also note that the boxing sequence is shown with only three cameras because the actor never turned around to face the fourth camera. It can be seen in the video that because of the faster motion, the boxing sequence has a number of tracking failures, even in the front-facing camera, but our method was able to reconstruct the correct motion by merging data from the other cameras. The walking sequence shows a complete 360-degree reconstructed unified motion.

In addition to the qualitative visual evaluation, we also perform multiple quantitative validations. In general, there is no ground truth data available for us to compare the goodness of our method. In addition, this work is not a direct pose estimation from the depth data [19], rather it uses the estimated pose from each camera and combines them together. Therefore, we do not need to quantify the quality of the individual pose, but need to estimate if the unified skeleton is better than the individual poses from each camera.

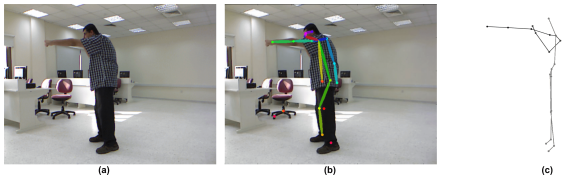

Fig. 6. Skeletal reconstruction comparison with the state-of-the-art method from Cao et al. [32]. The input image with the occluded right arm is shown in (a). The result from [32], directly obtained using their source code [33] is shown in (b). It can be seen that [32] fails badly in this specific situation where the pose of one of the arm and the leg is not estimated correctly. The unified skeleton reconstructed by combining the data from three cameras (shown in different shades of grey) captures the pose correctly, as shown in (c).

To demonstrate the quality and necessity for a unified skeletal reconstruction, we compare the results of our method with the state-of-the-art skeletal reconstruction method from Cao et al. [32]. In [32], a learning method based on Part Affinity Fields is used to estimate the skeleton of one or more person from any camera view. We directly used their publicly available code [33] to estimate the skeleton on the boxing sequence. One input frame can be seen Fig. 6a. The results from [32], directly using their code, can be seen in Fig. 6b. Whereas, the unified skeleton can be seen in Fig. 6c. As can be seen in Fig. 6b that even though [32] can work for most of the cases, but extreme cases with severe occlusion will still result in the incorrect estimation of the pose. Whereas, a multi-view non-linear skeletal fusion (Fig. 6c) will give a better result because it combines the joints with the least error to get an optimal unified skeleton.

Finally, we use two methods that compare the unified skeleton with individual skeletons using the bone-length variation estimation and 3D point cloud overlap to quantify the goodness of the unified skeleton as explained in the following sections.

A. Bone-length Variation Estimation

For the first quantitative analysis, we implement the bone-length variation estimation system that is presented and employed by Yueng et al. [23]. Similar to [23], we initialize all the bone-lengths manually for the first frame and classify them as the ideal lengths. Ideally the bone-lengths of the reconstructed skeleton at each frame should be as close as possible to the ideal lengths. Following [23], we compared bone-lengths at each frame for the unified skeleton and the front-facing cameras at each frame. For all the sequences we found the unified skeleton to be the closest to the ideal lengths compared to individual Kinects.

We computed the bone-length variation statistics for all the sequences. The most challenging sequence was the boxing sequence because it is the most challenging sequence with very fast motion, and with a number of tracking failures for all the cameras. Similar to [23], we show the statistics of bone-length variation for a number of bones for the boxing sequence in Fig. 7, and the walking sequence in Fig. 8. Table I and Table II show the absolute difference of the average bone length and the ideal bone-length for individual Kinects and the unified skeleton for the boxing and walking sequences, respectively. As can be seen in Fig. 7 and 8 and Table I and II that over the course of the entire sequence,

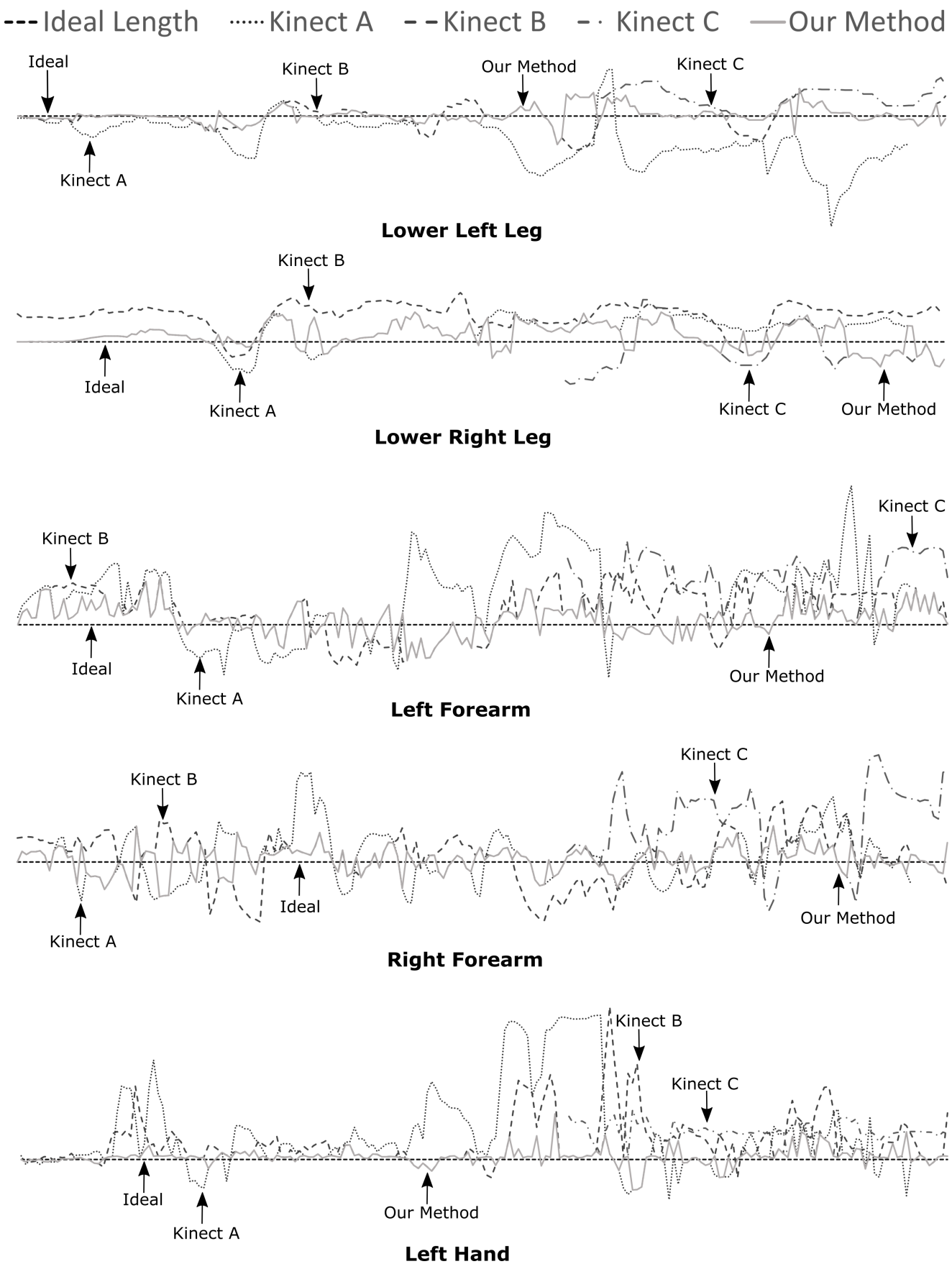

Fig. 7. The statistics of bone-length variation at different parts of skeleton in the **boxing** sequence. The ideal bone length is shown as the black dotted line in the center. The bone-length from individual Kinects and from the reconstructed uniform skeleton are shown over the 200 frames in different patterns and shades of grey. Kindly note that some Kinects (e.g. Kinect C), depending on the orientation of the person, are not used for all the frames in the reconstruction process.

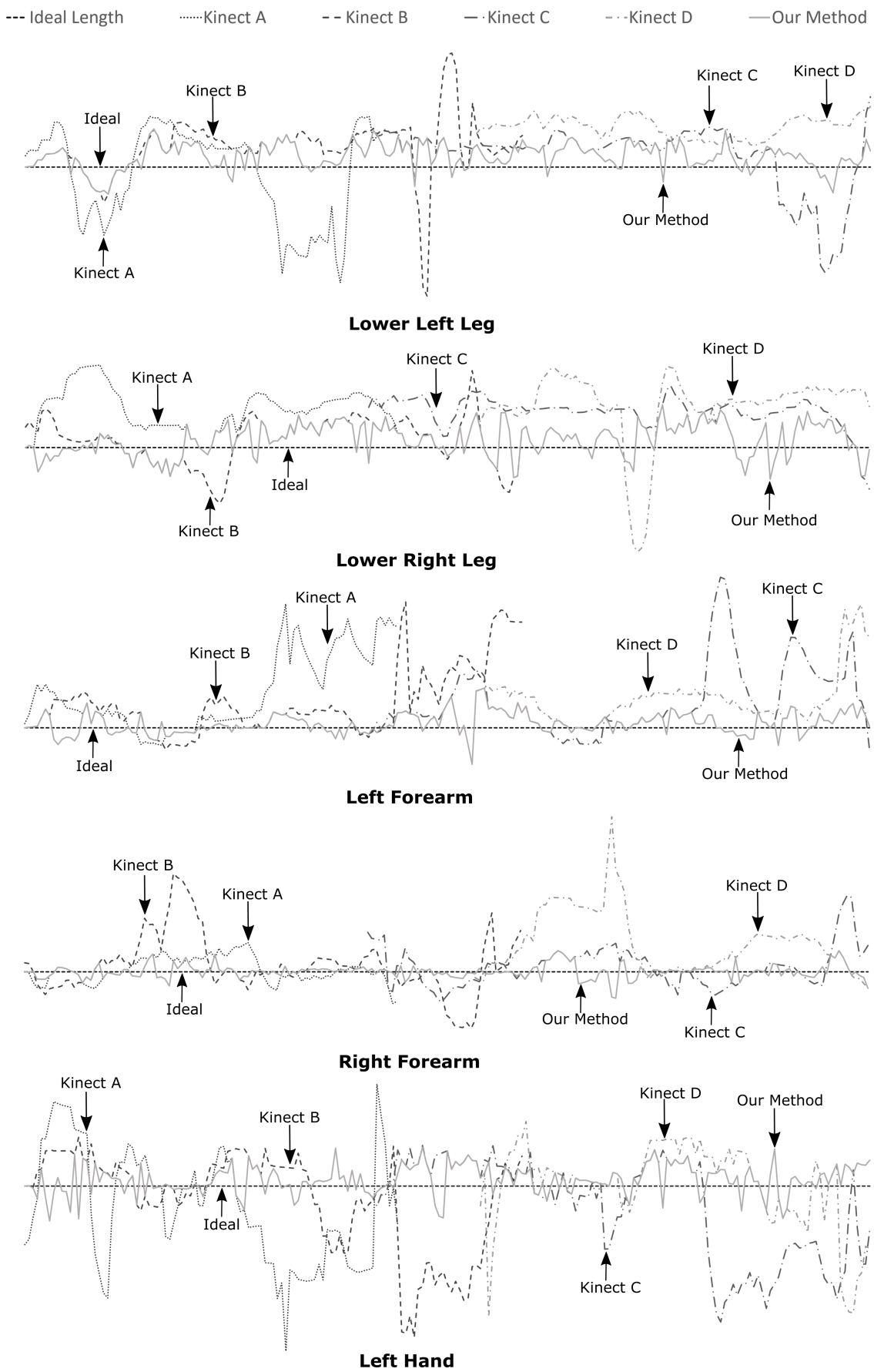

Fig. 8. The statistics of bone-length variation at different parts of skeleton in the **walking** sequence. The ideal bone length is shown as the black dotted line in the center. The bone-length from individual Kinects and from the reconstructed uniform skeleton are shown over the 205 frames in different patterns and shades of grey. Depending on the orientation of the person, not all Kinects are used for all the frames in the reconstruction process.

TABLE I
ABSOLUTE AVERAGE BONE-LENGTH DIFFERENCE FOR EACH KINECT (KINECT A, KINECT B, AND KINECT C) AND OUR METHOD (UNIFIED SKELETON) WITH RESPECT TO THE IDEAL BONE-LENGTH FOR THE **BOXING** SEQUENCE.

Bone	A	B	C	Unified
Pelvis	0.0078	0.0139	0.0096	0.0006
Spine	0.0088	0.0194	0.071	0.0008
Head	0.1090	0.1767	0.0642	0.0035
Left shoulder	0.0222	0.0300	0.0811	0.0071
Left upper arm	0.0175	0.0047	0.0399	0.0009
Left forearm	0.0304	0.0148	0.0684	0.0050
Left hand	0.0303	0.0208	0.0328	0.0033
Right shoulder	0.0105	0.0052	0.0142	0.0005
Right upper arm	0.0161	0.0338	0.0102	0.0046
Right forearm	0.0062	0.0044	0.0044	0.0019
Right hand	0.0221	0.0383	0.0435	0.0032
Left hip	0.0138	0.0073	0.0610	0.0042
Left upper leg	0.0128	0.0113	0.0639	0.0010
Left lower leg	0.0270	0.0012	0.0120	0.0007
Left foot	0.0135	0.0028	0.0087	0.0015
Right hip	0.0037	0.0134	0.0232	0.0005
Right upper leg	0.0186	0.0813	0.0173	0.0060
Right lower leg	0.0137	0.0324	0.0093	0.0058
Right foot	0.0019	0.0097	0.0193	0.0012

the bone lengths of the unified skeleton are always closest to the ideal length, when compared to individual Kinects.

B. Bounding-box and Skeleton Overlap Estimation

The bounding-box-based error measure calculates the overlap of the skeleton and the underlying 3D point cloud. For each bone in the individual skeleton from Kinects and the unified skeleton, a bounding box $\mathfrak{B}^i$ is defined at the first frame, where i=1, ..., 20 is the bone index. The size of each bounding box $\mathfrak{B}^i$ is initialized manually, and remains consistent throughout the sequence. The orientation of each bounding box is automatically determined from the orientation of the bone. The bounding boxes are tracked over the whole sequence using the skeletal animation. An example bounding box of a bone at an arbitrary frame can be seen in Fig. 9.

For each bounding box, the number of overlapping 3D points are calculated for each skeleton. A normalized error measure ξ_t is calculated for a time frame t as follows:

$$\xi_t = \frac{count(\bigcup(\mathfrak{P}(\mathfrak{B}_t^i)))}{count(\mathfrak{P}_t)} \tag{5}$$

Where, $count(\bigcup(\mathfrak{P}(\mathfrak{B}_t^b)))$ is the count of all unique points overlapping the bounding boxes of the bones, and, $count(\mathfrak{P}_t)$ is the count of all the points in the complete 3D point cloud. As shown in Fig. 9(c and d), the bounding box from the unified skeleton completely overlaps the correct region of the merged 3D point clouds, resulting in the higher value of ξ_t. In this particular frame, the unified skeleton has on average 8.95% better quality, compared to the individual cameras.

To compare the quality of the unified skeleton, we also estimate the goodness criteria for each individual front facing camera, ξ_t^c. The results of the bounding-box error measure with respect to ξ_t^c can be seen in Table III. Overall, for all three sequences, we found that on average the goodness of the unified skeleton ξ_t was better than the average goodness

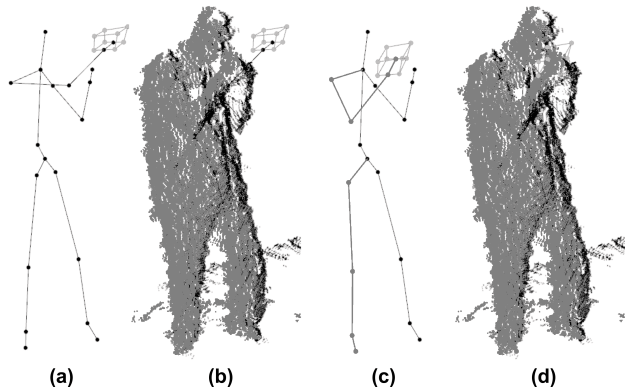

Fig. 9. One example of the bounding box based error calculation is shown. As shown in (a) and (b), the tracking failure causes the complete mismatch of the right arm's joints with respect to the underlying merged 3D point clouds. On the other hand, (c) and (d) show the reconstructed unified skeleton, where the joints are correctly aligned with the underlying merged 3D point clouds (shown in different shades of grey), thus a large of number of 3D points are within the bounding box of the right forearm.

of individual front facing cameras ξ_t^c by a factor of 9% to 12%. In addition, we also compared the bounding box error with two multi-view skeletal reconstruction methods from Yueng et al. [23] and Kitsikidis et al. [34]. The results for each sequence can be seen in Table III. As can be seen in Table III that even with a very low level of complexity in terms of the implementation our method results in a very similar accuracy for all the methods, especially for the boxing sequence where the error rate is 0.5% better. The maximum bounding box error was less than 2% as shown in Table III.

C. Discussion

In order to estimate the weighting factors of the non-linear fusion function (Eq. 1), α, β, and γ, we make use of the bone length, and bounding box error measures. As we do not have any ground truth data available for the sequences, we estimated the unified skeleton by varying these parameters, under the constrain that their sum is 1.0. In the end, the values α = 0.46, β = 0.33, and γ = 0.21, resulted in the unified skeletal animation where the overall error was minimum for both error measures. In principal, we can also use a learning method to improve the initial estimation of the weighting factors [35] [36] [37]. We would like to explore this in the future work.

In terms of computing speed our method runs at a moderate speed, and can estimate 10 frames of uniform skeletons per second. Ignoring the I/O overhead, it runs at an interactive frame rate. We tested the method on a 2.4 Ghz Quad Core i5 system with 4 GB of memory. Our method can easily be parallelized on a cluster as each frame is processed individually.

Our method is subject to a couple of limitations. We employ face detection to find out actor's orientation with respect to the camera. Face detection works well in more than 90% of the frames but it can fail if the face is occluded, for example, in the boxing sequence. We solve this issue in a pre-processing step by analyzing the sequence and if a couple of frames are missing the face. We look at the frames before and after the missing frames under the assumption that the frames were skipped due to occlusion. Additionally, we also

TABLE II
ABSOLUTE AVERAGE BONE-LENGTH DIFFERENCE FOR EACH KINECT (KINECT A, KINECT B, KINECT C, AND KINECT D) AND OUR METHOD (UNIFIED SKELETON) WITH RESPECT TO THE IDEAL BONE-LENGTH FOR THE **WALKING** SEQUENCE.

Bone	A	B	C	D	Unified
Pelvis	0.0067	0.0074	0.0082	0.0101	0.0004
Spine	0.0091	0.0065	0.0054	0.0075	0.0005
Head	0.0812	0.1293	0.0954	0.1421	0.0029
Left shoulder	0.0412	0.0126	0.0315	0.0523	0.0065
Left upper arm	0.0231	0.0125	0.0154	0.0312	0.0012
Left forearm	0.0201	0.0242	0.0543	0.0249	0.0040
Left hand	0.0201	0.0312	0.0126	0.0243	0.0049
Right shoulder	0.0542	0.0341	0.0287	0.0312	0.0009
Right upper arm	0.0091	0.0112	0.0115	0.0268	0.0031
Right forearm	0.0059	0.0067	0.0031	0.0043	0.0021
Right hand	0.0312	0.0498	0.0397	0.0587	0.0041
Left hip	0.0032	0.0053	0.0051	0.0043	0.0035
Left upper leg	0.0123	0.0209	0.0412	0.0251	0.0031
Left lower leg	0.0154	0.0109	0.0201	0.0119	0.0006
Left foot	0.0398	0.0295	0.0401	0.0393	0.0009
Right hip	0.0098	0.0168	0.0145	0.0195	0.0003
Right upper leg	0.0092	0.0451	0.0212	0.0119	0.0045
Right lower leg	0.0114	0.0216	0.0062	0.0098	0.0041
Right foot	0.0102	0.0125	0.0187	0.0203	0.0010

TABLE III
AVERAGE BOUNDING BOX ERROR MEASURE FOR THE FRONT FACING SKELETON AND THE UNIFIED SKELETON. FOR EACH SKELETON, THE PERCENTAGE OF PIXELS THAT ARE OUTSIDE THE BOUNDING BOX ARE SHOWN. THE UNIFIED SKELETON HAS MOST OF THE POINTS INSIDE THE BOUNDING BOXES COMPARED TO AN INDIVIDUAL FRONT FACING SKELETON.

Sequence	Front Facing	[23]	[34]	Our Method
Boxing	13.2%	1.89%	1.78%	1.25%
Walking	10.82%	0.86%	0.79%	0.69%
Rotation	11.34%	1.34%	1.23%	1.11%

use the normal of the root joint from the previous frame to determine if the actor is still oriented towards the camera. For example, in case face detection has failed, but in the previous frame the actor was facing the camera, then it is unlikely that the actor was rotated by 90 degrees in a single frame. Similarly, face detection can also detect false positives, for example, some parts in the surroundings can be incorrectly classified as faces. Again, we make use of the full sequence to determine the correct size and most likely position of the face. Incorrect face rectangles with very small or large areas are immediately discarded.

One can also see some flickering in the video sequences, where one joint switches between two cameras quickly. This is due to very similar error measures for different joint configurations, which can vary according to the normal orientation if both cameras see the joint clearly. The depth data from Kinect is very noisy, and we do not compensate for this noise [30], thus normal orientation can differ slightly in each frame. Additionally, the pose data from Kinect SDK is not time coherent and can result in a flicker from frame to frame, which is not a limitation of our method. In the future, we want to explore smoothing the 3D point clouds [30], and smoothing the skeleton data by reconstructing the joint position from all available cameras by means of learning or probabilistic methods.

Despite the limitations, we show that our method is able to reconstruct a unified 360-degree skeletal motion by a non-linear fusion of skeletal data from multiple Kinects that is free from the artefacts resulting from the orientation of the person and occlusion problems if only a single Kinect is employed for the pose estimation.

V. CONCLUSIONS

We presented a new method to reconstruct a unified skeletal animation from multiple Kinects. Our method can merge the skeleton data directly from Kinects by employing a non-linear fusion function that takes into account the joint's orientation, bone lengths, and their temporal smoothness. We use face tracking to find the cameras towards which the actor's face is oriented. Using the skeletal data from these cameras, the joint configuration that minimizes the non-linear fusion function is used to create a unified skeleton at each frame. Our method can reconstruct a unified 360-degree skeletal animation from multiple Kinects that would not be possible from a single Kinect due to occlusions and tracking failures. We also quantified the goodness of the reconstructed unified skeleton using the bone-length variation calculation and bounding-box overlap ratio methods. In the future, we would like to extend the unified skeletal animation reconstruction algorithm by incorporating a probabilistic model in the confidence measure. In addition, we would also like to work on new methods to quantify the goodness of the reconstructed unified skeleton.

REFERENCES

[1] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 569–577, 2003.

[2] C. Theobalt, N. Ahmed, G. Ziegler, and H.-P. Seidel, "High-quality reconstruction of virtual actors from multi-view video streams," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 45–57, 2007.

[3] P. E. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar, "Acquiring the reflectance field of a human face," in *SIGGRAPH*, 2000, pp. 145–156.
[4] T. Hawkins, P. Einarsson, and P. E. Debevec, "A dual light stage," in *EGSR*, 2005, pp. 91–98.
[5] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.
[6] D. Vlasic, I. Baran, W. Matusik, and J. Popovic, "Articulated mesh animation from multi-view silhouettes," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.
[7] N. Ahmed, C. Theobalt, C. Rössl, S. Thrun, and H.-P. Seidel, "Dense correspondence finding for parametrization-free animation reconstruction from video," in *CVPR*, 2008.
[8] MICROSOFT, "Kinect for microsoft windows and xbox 360. http://www.kinectforwindows.org/," November 2010.
[9] Y. M. Kim, D. Chan, C. Theobalt, and S. Thrun, "Design and calibration of a multi-view tof sensor fusion system," in *CVPR Workshop*, 2008.
[10] T. Fujita and T. Yoshida, "3d terrain sensing by laser range finder with 4-dof sensor movable unit based on frontier-based strategies," *Engineering Letters*, vol. 24, no. 2, pp. 164–171, 2016.
[11] C.-K. Yang and Y.-C. Chen, "A hci interface based on hand gestures," *Signal, Image and Video Processing*, vol. 9, no. 2, pp. 451–462, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11760-013-0462-1
[12] Y. Wu, Z. Jia, Y. Ming, J. Sun, and L. Cao, "Human behavior recognition based on 3d features and hidden markov models," *Signal, Image and Video Processing*, vol. 10, no. 3, pp. 495–502, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11760-015-0756-6
[13] N. Ahmed and S. Khalifa, "Time-coherent 3d animation reconstruction from rgb-d video," *Signal, Image and Video Processing*, vol. 10, no. 4, pp. 783–790, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11760-015-0813-1
[14] A. Amamra, "Smooth head tracking for virtual reality applications," *Signal, Image and Video Processing*, vol. 11, no. 3, pp. 479–486, 2017. [Online]. Available: http://dx.doi.org/10.1007/s11760-016-0984-4
[15] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *ICCV*, 2011.
[16] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 731–738.
[17] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *ICCV*, 2011.
[18] R. Lun and W. Zhao, "A survey of applications and human motion recognition with microsoft kinect," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 29, no. 05, p. 1555008, 2015. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218001415550083
[19] L. Chen, H. Wei, and J. Ferryman, "A survey of human motion analysis using depth imagery," *Pattern Recogn. Lett.*, vol. 34, no. 15, pp. 1995–2006, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2013.02.006
[20] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 188:1–188:12, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366207
[21] H. Yasin, U. Iqbal, B. Krüger, A. Weber, and J. Gall, "3d pose estimation from a single monocular image," *CoRR*, vol. abs/1509.06720, 2015. [Online]. Available: http://arxiv.org/abs/1509.06720
[22] 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*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1297–1304. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2011.5995316
[23] K.-Y. Yeung, T.-H. Kwok, and C. C. L. Wang, "Improved skeleton tracking by duplex kinects: A practical approach for real-time applications," *Journal of Computing and Information Science in Engineering*, vol. 13, no. 04, pp. 041–051, 2013.
[24] M. Dantone, J. Gall, C. Leistner, and L. J. V. Gool, "Human pose estimation using body parts dependent joint regressors." in *CVPR*. IEEE Computer Society, 2013, pp. 3041–3048.
[25] S. Obdrzalek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel, "Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population," in *Engineering in Medicine and Biology Society (EMBC)*, 2012, pp. 1188–1193.
[26] N. Ahmed, "A system for 360 degree acquisition and 3d animation reconstruction using multiple rgb-d cameras," in *Proceedings of the 25th International Conference on Computer Animation and Social Agents (CASA)*, ser. Casa'12, 2012.
[27] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor, "Markerless motion capture using multiple color-depth sensors," in *VMV*, 2011.
[28] G. Ye, Y. Liu, Y. Deng, N. Hasler, X. Ji, Q. Dai, and C. Theobalt, "Free-viewpoint video of human actors using multiple handheld kinects," *IEEE T. Cybernetics*, 2013.
[29] M. Caputo, K. Denker, B. Dums, and G. Umlauf, "3d hand gesture recognition based on sensor fusion of commodity hardware," in *Mensch and Computer 2012*, 2012.
[30] G. Sanchez, E. Leal, and N. Leal, "A linear programming approach for 3d point cloud simplification," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 60–67, 2017.
[31] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.
[32] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
[33] Z. Cao, "$https : //github.com/zhec/realtime\_multi - person\_pose\_estimation$," July 2017.
[34] A. Kitsikidis, K. Dimitropoulos, S. Douka, and N. Grammalidis, "Dance analysis using multiple kinect sensors," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 2, Jan 2014, pp. 789–795.
[35] H. Miyajima and N. Shigei, "Initial setting of effective parameters for fuzzy modeling," *IAENG International Journal of Computer Science*, vol. 44, no. 3, pp. 375–382, 2017.
[36] M. Langovoy, "Machine learning and statistical analysis for brdf data from computer graphics and multidimensional reflectometry," *IAENG International Journal of Computer Science*, vol. 42, no. 1, pp. 22–30, 2015.
[37] H. Zhuang, M. Yang, Z. C. Cui, and Q. Zheng, "A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 52–59, 2017.

Naveed Ahmed is an assistant professor at the Department of Computer Science, University of Sharjah. He received his PhD in computer science from the University of Saarland (Max-Planck-Institute for Informatics), Germany, in 2009. He worked as a research and development engineer at Autodesk in Cambridge, United Kingdom, for 2 years. He is currently working as an assistant professor at the Department of Computer Science, University of Sharjah. His research interests include 3-D animation, dynamic scene reconstruction, vision-based computer graphics, and multi-view video based modeling and rendering.