Journal of Computer Science

Original Research Paper

Multi-View RGB-D Video Analysis and Fusion for 360 Degrees Unified Motion Reconstruction

Naveed Ahmed

Department of Computer Science, University of Sharjah, Sharjah, UAE

Article history
Received: 14-10-2017
Revised: 26-11-2017
Accepted: 23-12-2017

Email: nahmed@sharjah.ac.ae

Abstract: We present a new method for capturing human motion over 360 degrees by the fusion of multi-view RGB-D video data from Kinect sensors. Our method is able to reconstruct the unified human motion from fused RGB-D and skeletal data over 360 degrees and create a unified skeletal animation. We make use of all three streams: RGB, depth and skeleton, along with the joint tracking confidence state from Microsoft Kinect SDK to find the correctly oriented skeletons and merge them together to get a uniform measurement of human motion resulting in a unified skeletal animation. We quantitatively validate the goodness of the unified motion using two evaluation techniques. Our method is easy to implement and provides a simple solution of measuring and reconstructing a 360 degree plausible unified human motion that would not be possible to capture with a single Kinect due to tracking failures, self-occlusions, limited field of view and subject orientation.

Keywords: 3D Animation, Kinect, RGB-D Video, Motion Capture, Multi-View Video

Introduction

The field of marker-less motion capture and 3D or free-viewpoint video has received a lot of interest in the past decade. It has a number of applications in the areas such as natural user interface design, motion analysis, video surveillance, virtual reality etc. Traditionally, multi-view RGB camera systems have been used to capture motion, shape and appearance of a real-world actor. Carranza *et al*. (2003) presented one of the pioneer works in this area by employing eight synchronized RGB video cameras to capture a real-world actor. Using the eight video streams they developed a template-based marker-less motion capture scheme to correctly estimate the motion of the actor. This work was later extend by Theobalt *et al*. (2007), who measured the surface reflectance properties of the actor in addition to its motion. Afterward, de Aguiar *et al*. (2008) presented another template-based deformation framework to capture high quality motion of the real-world actor. In contrast, Vlasic *et al*. (2008) used the skeletal data to deform a template mesh to capture the high quality motion. Ahmed *et al*. (2008) used a shape matching approach over dynamic visual hulls to capture the track a single mesh over the complete sequence. So far, the previously explained methods relied on the RGB data.

Depth cameras, especially consumer-grade depth cameras were made popular by the introduction of Kinect by Microsoft (2010). The major benefit of Kinect is its low cost that allows it to be used a very cheap RGB-D sensor to acquire both the color and depth data at 30 frames per second (Ahmed and Khaifa, 2016). If only the depth data is desired then the Time-of-Flight (TOF) cameras can also be employed (Kim *et al*., 2008). Unlike Kinect, a TOF camera does not provide a unified solution to acquire both depth and RGB data, which is one of the major strengths of Kinect. In addition, using the Microsoft's Kinect SDK, one can also acquire real-time pose estimation or skeletal data of a real-world actor.

Pose estimation from a single camera has been a hallmark feature of Kinect and a number of solutions have been proposed for human pose estimation using a single Kinect (Girshick *et al*., 2011; Ye *et al*., 2011; Baak *et al*., 2011). The real-time skeletal data from Kinect is employed in a number of applications ranging from controlling a robot using the skeletal data or a controller free gaming experience by means of body poses (Lun and Zhao, 2015). The Kinect SDK can provide the skeletal data of multiple actors in a standing or sitting position.

A number of methods have been proposed that only use the depth data for the real-time pose estimation using machine learning or non-linear optimization (Chen *et al*.,

2013). On the other hand, one can use the Kinect SDK directly to get the real-time pose data. Thus Kinect SDK provides a simpler solution of pose retrieval compared to a number of other methods that are comparatively very difficult to implement (Wei *et al*., 2012; Ye *et al*., 2011; Baak *et al*., 2011; Yasin *et al*., 2015; Shotton *et al*., 2011; Dantone *et al*., 2013). Due to the complexity of these methods they are not as widely adapted as Kinect's pose estimation. In practice, Kinect's SDK has been widely adopted for the real-time pose estimation and has been employed in a number of applications in a number of areas (Lun and Zhao, 2015).

Kinect has been developed to be used as a standalone camera in a living room, where the person is always facing the camera. Therefore, the Kinect SDK only captures the correct pose of the person as long as it is facing the camera with the frontal orientation (Obdrzalek *et al*., 2012). If the person is not facing the camera or in case the body parts of the person are occluded due to self-occlusions then the incorrect orientation or the missing depth information result in the incorrect pose estimation. Additionally, due to the field of view limitations of a Kinect combined with the orientation of the person, it is not possible to capture the motion of the person from all sides. Thus a 360 capture of the motion of the person is not possible using a single Kinect.

In order to resolve these shortcomings of pose estimation from a single Kinect, a number of methods have been proposed that employ more than one Kinect for the pose estimation. Viewing a scene from multiple Kinects provide a number of benefits, specifically if a body part is occluded in one camera view will be visible in some other camera. Additionally, if the placement of the cameras is around the person, then the person will be oriented towards at least one of the camera that can correctly estimate its pose. On the other hand using multiple Kinects results in the loss of depth data due to the interference between different depth sensors. As shown by Ahmed (2012), this interference does not result in the loss of quality for a 360 degree 3D animation reconstruction, because the missing information from one depth sensor is filled in by the other sensors. In their work (Ahmed, 2012), employed six synchronized Kinects to reconstruct a 360 degree 3D animation. In contrast, Berger *et al*. (2011) employed four Kinects for unsynchronized marker-less motion capture. Ye *et al*. (2013) employed three hand-held Kinects for marker-less performance capture. Caputo *et al*. (2012) employed multiple Kinects for hand gesture recognition. All of these methods did not use the real-time pose data from Kinect. Rather, all pose estimation methods used an optimization process by means of silhouette-based minimization or template deformation to find the correct pose. Even though these methods work fine in practice, using Kinect SDK for the pose estimation has a number of benefits. In the first place, the pose data is available at 30 frames per second, making it suitable for a number of real-time applications. There is no additional post processing required before using the skeletal data. In addition, the reliability of the skeletal data is good enough to be employed in a number of applications as long as the person is facing the camera and the person's pose does not result in the self-occlusion of body parts (Obdrzalek *et al*., 2012).

If multiple Kinects are used to acquire the real-time pose data, it is not straightforward to fuse these poses together for 360 degree unified motion reconstruction. As Kinect only estimates the correct pose if the person is facing the camera, a completely incorrect pose with inverted joints is estimated for the back-facing camera. To fuse the pose data from Kinects, it is important to first identify the Kinects toward which the person is oriented. In addition, even for those Kinects with the correct person orientation, the joint data should be selected in such a way that the self-occluded joints should be discarded and only be used from the pose data that is estimated from the non-occluded joints. Finally, even if the joints are no occluded, a joint which is oriented more towards a Kinect should be preferred because in general it is better tracked compared to a joint that is not oriented towards Kinect (Obdrzalek *et al*., 2012).

In this study, 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 confidence score to create a unified skeletal representation at each frame. Our method uses the Microsoft Kinect SDK for acquisition and its implementation is relatively 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 and the algorithmic details can be seen in Fig. 2.

In the following sections, we will present each of the algorithmic step in detail, starting from the discussion of data acquisition, followed by the presentation of the unified skeletal animation reconstruction algorithm. Afterward, the results are presented and validated followed by the conclusions.

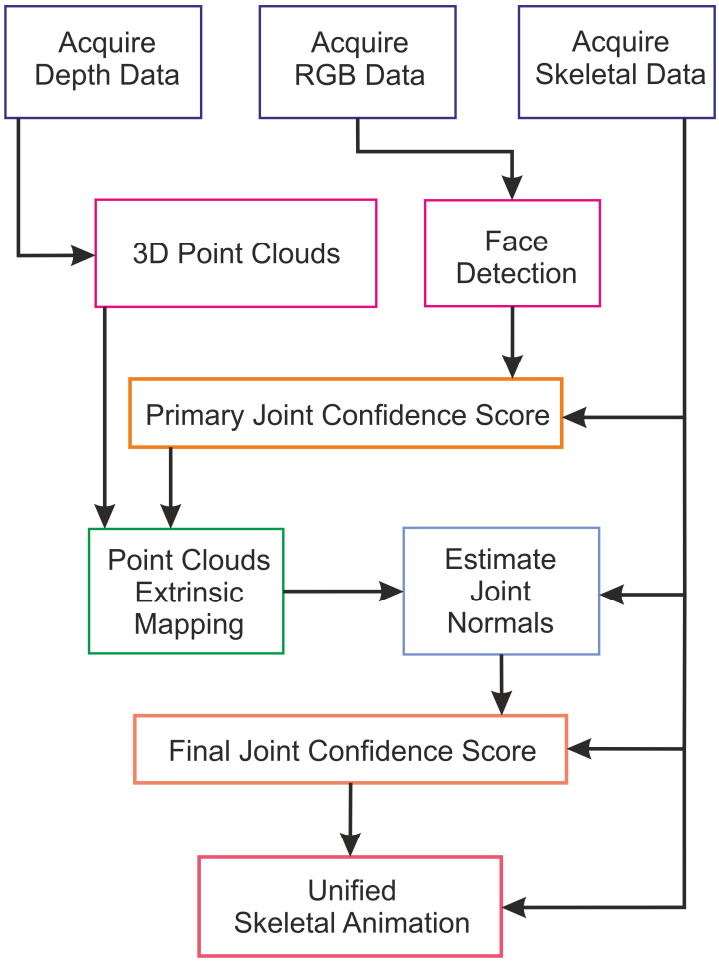

Fig. 1: Flowchart of the proposed method, starting from the acquisition of depth, RGB and skeletal data to the measurement of 360-degree human motion measurement using the novel confidence score, resulting in the unified skeletal animation reconstruction. The algorithmic details from each step can be seen in Fig. 2

Algorithmic Details

1) *Acquire Depth data using multiple Kinects at 30 frames per second and save each frame for each camera as a depth image file.*
2) *Acquire RGB data using multiple Kinects at 30 frames per second and save each frame from each camera as an RGB image file.*
3) *Acquire Skeletal data using multiple Kinects at 30 frames per second and save each frame from each camera in data file storing all the joints information.*
4) *Transform the acquired depth data to a 3D point cloud using the calibration parameters.*
5) *Use a face detection algorithm on the RGB data to find the usable front facing cameras.*
6) *Only using the front facing cameras, use the skeletal data to find the primary joint confidence score that uses the joint tracking state, occlusion status, temporal smoothness and the bone length measure.*
7) *Map the 3D point clouds to a unified 3D coordinate system using the primary joint confidence score and the RGB to depth mapping.*
8) *Use the unified point cloud from all the front facing cameras at each frame to find the orientation of each joint in the skeletal data.*
9) *Use the orientation data from each joint and combine it with the primary joint confidence score to obtain the final joint confidence score. The new score favors the front facing cameras using the orientation data.*
10) *Select the best 20 joints for each joint type from the front facing cameras that have the best final confidence score to reconstruct a unified skeletal animation.*

Fig. 2: Algorithmic details of the method from the acquisition to the unified 360-degree animation reconstruction

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*. (2013) 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. 3a and 3b.

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. In addition, the Kinect SDK also provides a tracking state for the skeleton and each joint. For the skeleton the tracking states are: Not Tracked (did not track anything), Position Only (did not track any joint, only one skeleton position) and Tracked (did track joints). For the joints the tracking states are: Not Tracked (joint data is not available), Inferred (joint data is calculated from other tracked joints), Tracked (joint data is tracked and available).

The joint tracking states are an important part of the confidence score assigned to each joint for our method, as discussed in the next section. Even though our experiments use a static camera setup, our method can also work without a fixed extrinsic parameterization between the cameras for the whole sequence, in case the cameras are not static. We show that the extrinsic parameters can also be calculated dynamically using the skeleton data as explained in the next section.

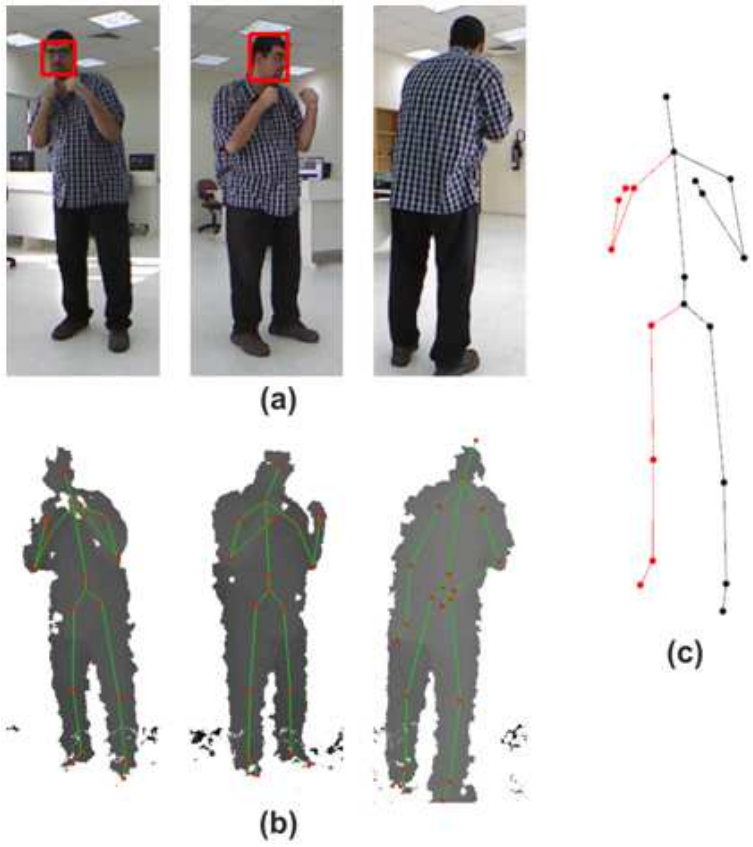

Fig. 3: (a) shows RGB frames from three cameras. Frontal and profile faces are detected in two cameras (b) shows the depth data with the overlaid skeleton from Kinect (c) shows the unified skeleton from the two cameras towards which the actor's face is oriented

Unified Skeletal Animation Reconstruction

The fusion of skeleton data from multiple Kinects poses a number of challenges. First, the skeleton data from 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. 3b(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 (Viola and Jones, 2001) 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. 3a). Face detection is a standard feature provided 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 Validations 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 assigning a confidence score to each of the skeleton joint for each camera. Assuming we are using C cameras and there are $\mathcal{T}$ frames in the sequence, the confidence score S for a joint j_t^c, where j = 1,…,20, c = 1,...,C and t = 1,..., $\mathcal{T}$, is defined by:

$$\mathcal{S}\left(j_t^c\right)=\mathcal{R}\left(j_t^c\right)+\mathcal{O}\left(j_t^c\right)+\mathcal{D}\left(j_t^c\right)+\mathcal{B}\left(j_t^c\right) \quad (1)$$

$\mathcal{R}(j_t^c)$ is the joint tracking state for j_t^c from the Kinect SDK and its possible values are:

$$\mathcal{R}(j_t^c)=\begin{cases}0 & \text{if joint data is not available}\\ 0.5 & \text{if it is calculated from other joints}\\ 1 & \text{if it is tracked and available}\end{cases}$$

It is to be noted that the values 0, 0.5 and 1 are not provided by the Kinect SDK. We convert the joint tracking state to these weights based on its tracking status. A joint that is not tracked should not have any weight. Similarly, if the joint's state is tracked from the other joints, then its weight should be half of the weight of the joint that is independently tracked.

$\mathcal{O}\left(j_t^c\right)$ is the occlusion score for j_t^c, it is 0 if the joint is occluded or 1 otherwise. We find out if the joint is occluded or not by back projecting its depth value to the depth image and comparing the z value of the three-space joint position from Kinect and the depth image. We cannot completely discard a joint if it is occluded because in some cases Kinect can still track the pose even if a joint is occluded for a small number of frames. The term $\mathcal{O}\left(j_t^c\right)$ complements $\mathcal{R}\left(j_t^c\right)$ such that even if the joint tracking state reports a higher confidence in the joint, but it is occluded then it should get a lesser score.

$\mathcal{D}\left(j_t^c\right)$ is the temporal smoothness term for j_t^c, which if a joint is moving, compares its displacement d_t at t with the displacement d_{t-1} at t–1. If the joint is not moving, or if there is very little movement, then it is set to 1. If $d_t \Leftarrow \sigma * d_{t-1}$ then it is set to 1, if $d_t > \sigma * d_{t-1}$ and $d_t \Leftarrow \rho * d_{t-1}$ then it is 0.5, otherwise 0. We found this term to be very important because it penalizes sudden jerky motion of the joints in case of a tracking failure. Skeleton tracking from Kinect can also fail not because of the occlusions but also due to the limitations of the underlying pose estimation algorithm. By introducing this temporal smoothness term, we try to compensate for these failures. It is to be noted that $\mathcal{D}\left(j_t^c\right)$ cannot compensate for the jerkiness if it is present for a joint in a particular frame, in all the cameras. The jerkiness is a shortcoming of the underlying pose estimation algorithm, whereas this term favors the best available joint with the least jerky motion. In this regard, this term compensates for this particular shortcoming of the underlying skeleton estimation algorithm. The parameters σ and ρ are found through experiment, as discussed in the Results and Validation section. For our method, we chose σ = 1.2 and ρ = 2.0 for two slow sequences, while for the faster motion their value was *1.05* and *1.7* respectively.

Finally, $\mathcal{B}\left(j_t^c\right)$, is the bone length score. Similar to Yueng *et al*. (2013), we initialize all the bone-lengths manually for the first frame and classify them as the ideal lengths. As each joint is associated with one or more bones, let $\mathcal{L}(j)$ be the sum of all the bones lengths associated with each joint. Using the sum of ideal bone lengths associated with each joint $\mathcal{L}_{ideal}(j)$, the normalized term $\mathcal{B}\left(j_t^c\right)$ is calculated as follows:

$$\mathcal{B}\left(j_t^c\right)=\begin{cases}\dfrac{\mathcal{L}\left(j_t^c\right)}{\mathcal{L}_{ideal}\left(j_t^c\right)} & \text{if}\mathcal{L}_{ideal}\left(j\right)>\mathcal{L}\left(j_t^c\right)\\ \dfrac{\mathcal{L}_{ideal}\left(j_t^c\right)}{\mathcal{L}\left(j_t^c\right)} & \text{if}\mathcal{L}_{ideal}\left(j\right)\le\mathcal{L}\left(j_t^c\right)\end{cases}$$

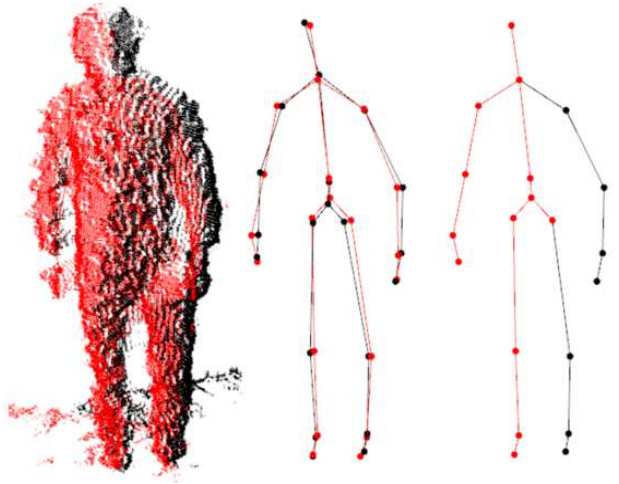

Fig. 4: (a) shows unified two point clouds (shown in black and red) and (b) shows the corresponding two skeletons after the extrinsic calibration. (c) shows the unified skeleton reconstructed from our method

Thus, the maximum value of $\mathcal{B}\left(j_t^c\right)$ is 1.0 and any deviation from the ideal bone length results in a smaller score. We also use the bone length variation to quantitatively validate our method, as discussed in the Results and Validations section.

For all the cameras oriented towards the face of the actor, we use the 3-space mapping of the joints with the confidence score greater than 2 and find the least squares solution to determine the transformation that maps one camera to the other. As explained earlier, given the static camera setup, this is not a required step. It is only performed to demonstrate that our method can also work for moving cameras.

This dynamic extrinsic calibration is done at every frame and if more than two cameras are used, they are mapped to one reference camera. The results of extrinsic calibration can be seen in Fig. 4a and 4b. In practice, we always found 12 or more joints with the confidence value greater than 2. Thus, the linear system was never underdetermined. The confidence score in Equation 1 is one of the ways to perform the dynamic extrinsic calibration. Using the skeletal data is a novel approach in this regard, but one can use also use traditional image processing based methods, similar to Ahmed (2012), to achieve the similar results. To reconstruct the unified skeleton, dynamic extrinsic calibration by means of Equation 1, is the first step. We modify Equation 1 with an additional orientation term to select the best possible joints for the uniform skeletal reconstruction.

Using the extrinsic calibration, we first map 3D point clouds and skeletons to the global world coordinate system. In the next step, we use the unified point cloud (Fig. 4a) to estimate the normal $n\left(j_t^c\right)$ of each j_t^c. The normal orientation is estimated using SVD-based plane fitting on the neighboring 3D points of j_t^c in the unified point cloud. If we do not use the unified point clouds and the normal for j_t^c is only estimated through its corresponding camera point cloud, then the normal orientation will be biased towards that particular camera.

Before merging the skeleton data, we modify our confidence measure (Equation 1) and introduce a new orientation term $\mathcal{N}\left(j_t^c\right)$:

$$\mathcal{S}\left(j_t^c\right)=\mathcal{R}\left(j_t^c\right)+\mathcal{O}\left(j_t^c\right)+\mathcal{D}\left(j_t^c\right)+\mathcal{B}\left(j_t^c\right)+\left(1.0-\mathcal{N}\left(j_t^c\right)\right) \quad (2)$$

$\mathcal{N}\left(j_t^c\right)$ is the dot product of $n\left(j_t^c\right)$ and $v\left(j_t^c\right)$, where $v\left(j_t^c\right)$ is the view vector from j_t^c to the camera c. The maximum value of $\left(1.0-\mathcal{N}\left(j_t^c\right)\right.$ is 1 if $n\left(j_t^c\right)$ is oriented towards c and it decreases as the actor rotates away from the camera. This term increases the confidence score for the joints of the front-facing camera, which is desired, as Kinect best estimates the skeleton if the actor is facing the camera. Finally, we reconstruct the unified skeleton at t by selecting each of the 20 joints from the camera c that has the highest confidence score $\mathcal{S}\left(j_t^c\right)$ presented in Equation 2 for that particular joint.

Results, Validations and Discussions

We recorded three sequences of 200 frames each. 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 whole body. Our method was able to track all sequences successfully and the selected joints from multiple cameras capture the motion accurately. Our confidence measure ensures 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. 5. More results from two of the sequences can be seen in Fig. 3c, 4c. It can be observed in the results that our method can merge the skeleton data from multiple cameras to reconstruct the unified skeletal animation. Please 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 figures 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 (Chen *et al*., 2013), 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. 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:

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*. (2013). Similar to their method, 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 Yeung *et al*. (2013), 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 closest to the ideal lengths compared to individual Kinects.

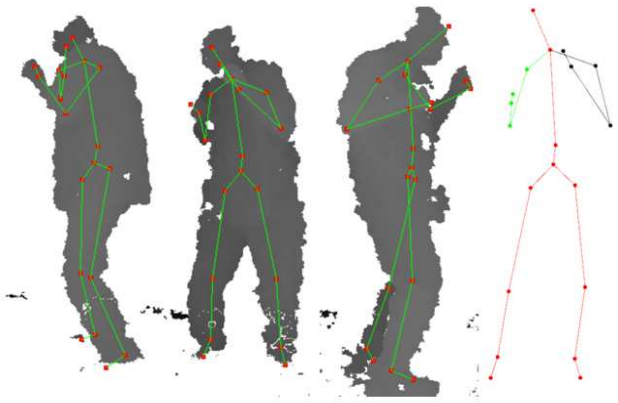

Fig. 5: Merging of three cameras (black, red and green) is shown on the right. As can be seen the algorithm correctly selects the joints from the cameras that depict the most accurate motion

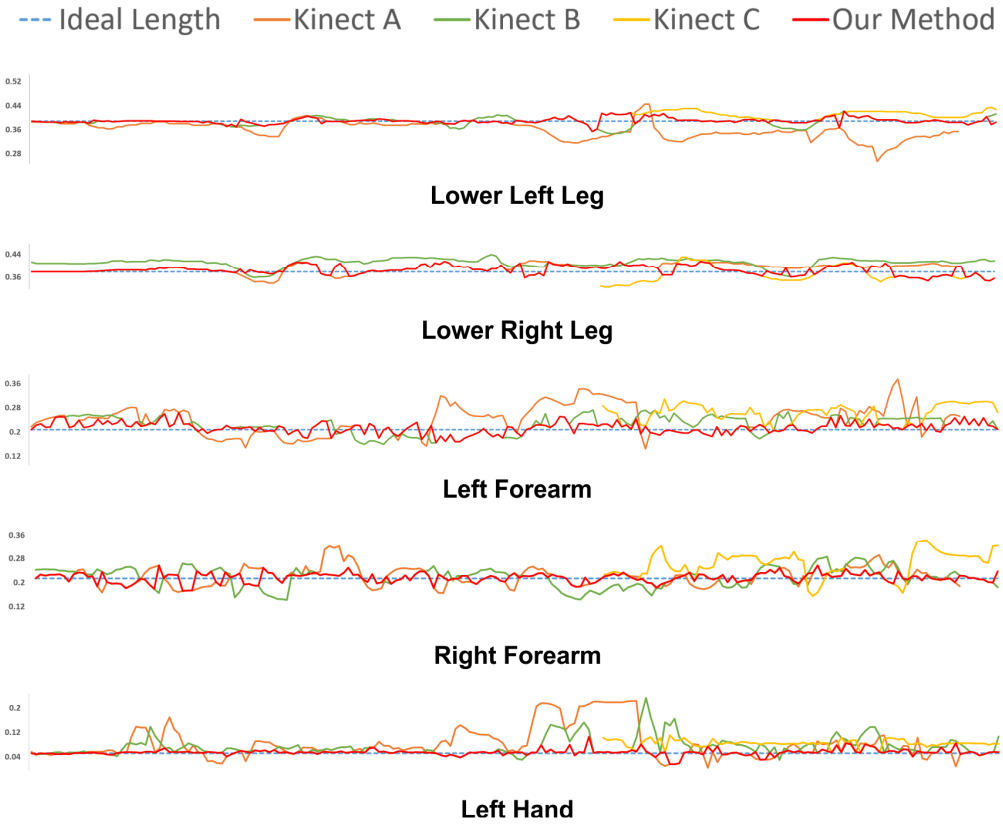

Fig. 6: The statistics of bone-length variation at different part of skeleton in the boxing sequence. The ideal bone length is shown as a dotted line in the center. The bone-length from individual Kinects and from the reconstructed uniform skeleton are shown over the 200 frames. Kindly note that some Kinects (e.g., Kinect C), depending on the orientation of the person, are not used for all the frames in the reconstruction process

In this study, due to size constraints, we are only showing results of the boxing sequence, because it is the most challenging sequence with the very fast motion, with a number of tracking failures for all the cameras. Similar to Yeung *et al*. (2013), we show the statistics of bone-length variation for a number of bones for the boxing sequence in Fig. 6. Table 1 shows the absolute difference of the average bone length and the ideal bone-length for individual Kinects and the unified skeleton for the boxing sequence. As can be seen in Fig. 6 and Table 1, over the course of the sequence, the bone lengths of the unified skeleton are always closest to the ideal length, when compared to individual Kinects.

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 $\mathcal{B}^i$ is defined at the first frame, where i = 1,...,19 is the bone index. The size of each bounding box $\mathcal{B}^i$ 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. 7.

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:

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

where, $count\left(\bigcup\left(\mathfrak{P}\left(\mathfrak{B}_t^i\right)\right)\right)$ is the count of all unique points overlapping 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. 7(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 7.73% better quality, compared to the individual cameras.

For comparison, we also estimate the goodness criteria for each individual front facing camera ξ_t. For all three sequences, we found that on average the goodness of the unified skeleton ξ_t was better than the average goodness of individual front facing cameras ξ_t by a factor of 7% to 10%.

Table 1: Table type styles (Table caption is indispensable)

Bones	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
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

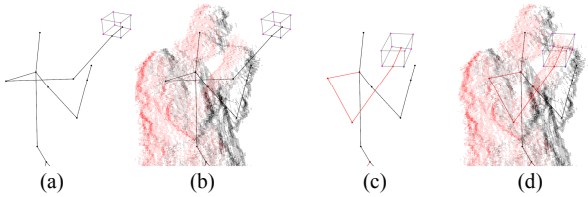

Fig. 7: One example of bounding box based error calculation is shown. As shown in (a) and (b), the tracking failure causes the complete mismatch of the right arm's joints with respect to the underlying merged 3D point clouds. On the other hand, (c) and (d) show the reconstructed unified skeleton, where the joints are correctly aligned with the underlying merged 3D point clouds, thus a large of number of 3D points are within the bounding of the right forearm

Discussion

We used both evaluation methods to estimate the two parameters σ and ρ in the temporal smoothness term $\mathcal{D}\left(j_t^c\right)$ that was used in Equation 1. For each sequence, we reconstructed the unified skeletons with varying parameters and compared each bone-length with the corresponding ideal length and the goodness of the skeleton by calculating ξ_t. We observed that for the sequences with the slower motion the value was higher, whereas for the faster motion it was lower, because the sudden jerky motion is heavily penalized in case of the faster motion.

In terms of computing speed our method runs at a moderate speed and can estimate 12 frames of uniform skeletons per second. Ignoring the I/O overhead and if the extrinsic calibration is pre-established, it runs in real-time at 30 frames per second. 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, then 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 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 reconstructing sequences, where one joint switches between two cameras quickly. This is due to very similar confidence score, which can vary according 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, thus normal orientation can differ slightly in each frame. Additionally, the general flickering in joint positions is not from our algorithm rather it is the raw skeleton data from Kinect, which is not smooth over time. In future, we want to explore smoothing the skeleton data by reconstructing the joint position from all available cameras by means of a weighted average, or incorporate a probabilistic model in the confidence measure.

Despite the limitations, we show that our method is able to reconstruct the human motion over 360 degrees by fusing multiple RGB-D sensors and reconstruct a unified skeletal animation in a plausible way that would not be possible with a single Kinect.

Conclusion

We presented a method to reconstruct human motion over 360 degrees by using data from multiple RGB-D Kinect sensors and reconstruct a unified skeletal animation. Our method can merge the skeleton data directly from Kinects by assigning a confidence score to each joint based on its tracking state, occlusion, displacement, bone length and orientation. The confidence score is then used to select 20 best joints from the cameras towards which the actor's face is oriented. This orientation is found by means of face detection. 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 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.

Acknowledgement

The author would like to thank the students who helped with the recordings by performing specific motion.

Funding Information

The work is funded completely internally by the University of Sharjah.

Author's Contributions

The author has solely worked on the acquisition setup, followed by the design and implementation of the algorithm.

Ethics

This is author's original work and no ethical issues are associated in terms of its publication.

References

Ahmed, N. and S. Khalifa, 2016. Time-coherent 3d animation reconstruction from rgb-d video. Signal, Image Video Processing, 10; 783-790.

Ahmed, N., 2012. A system for 360 degree acquisition and 3d animation reconstruction using multiple rgb-d cameras. Proceedings of the 25th International Conference on Computer Animation and Social Agents, (ASA'12).

Ahmed, N., C. Theobalt, C. Rössl, S. Thrun and H.P. Seidel, 2008. Dense correspondence finding for parametrization-free animation reconstruction from video. Proceedings of the Conference on Computer Vision and Pattern Recognition, Jun. 23-28, IEEE Xplore Press, Anchorage, AK, USA. DOI: 10.1109/CVPR.2008.4587758

Baak, A., M. Muller, G. Bharaj, H.P. Seidel and C. Theobalt, 2011. A data-driven approach for real-time full body pose reconstruction from a depth camera. Proceedings of the International Conference on Computer Vision, Nov. 6-13, IEEE Xplore Press, Barcelona, Spain. DOI: 10.1109/ICCV.2011.6126356

Berger, K., K. Ruhl, Y. Schroeder, C. Bruemmer and A. Scholz *et al*., 2011. Markerless motion capture using multiple color-depth sensors. In: Vision, Modeling and Visualization, Eisert, P., K. Polthier and J. Hornegger (Eds.), The Eurographics Association.

Caputo, M., K. Denker, B. Dums and G. Umlauf, 2012. 3d hand gesture recognition based on sensor fusion of commodity hardware. Proceedings of the Mensch and Computer, (PMC'12), Oldenbourg Verlag, pp: 293-302.

Carranza, J., C. Theobalt, M.A. Magnor and H.P. Seidel, 2003. Free-viewpoint video of human actors. ACM Trans. Graph., 22: 569-577. DOI: 10.1145/882262.882309

Chen, L., H. Wei and J. Ferryman, 2013. A survey of human motion analysis using depth imagery. Pattern Recogn. Lett., 34: 1995-2006.

Dantone, M., J. Gall, C. Leistner and L.J.V. Gool, 2013. Human pose estimation using body parts dependent joint regressors. Proceedings of the Conference on Computer Vision and Pattern Recognition, Jun. 23-28, IEEE Xplore Press, Location: Portland, pp: 3041-3048. DOI: 10.1109/CVPR.2013.391

de Aguiar, E., C. Stoll, C. Theobalt, N. Ahmed and H.P. Seidel *et al*., 2008. Performance capture from sparse multi-view video. Proceedings of SIGGRAPH, Aug. 11-15, ACM, Los Angeles, California, DOI: 10.1145/1399504.1360697

Girshick, R., J. Shotton, P. Kohli, A. Criminisi and A. Fitzgibbon, 2011. Efficient regression of general-activity human poses from depth images. Proceedings of the International Conference on Computer Vision, Nov. 6-13, IEEE Xplore Press, Barcelona, Spain. DOI: 10.1109/ICCV.2011.6126270

Kim, Y.M., D. Chan, C. Theobalt and S. Thrun, 2008. Design and calibration of a multi-view tof sensor fusion system. Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Jun. 23-28, IEEE Xplore Press, Anchorage, AK, USA. DOI: 10.1109/CVPRW.2008.4563160

Lun, R. and W. Zhao, 2015. A survey of applications and human motion recognition with microsoft kinect. Int. J. Pattern Recogn. Artificial Intelligence, 29: 1-50.

Microsoft, 2010. Kinect for microsoft windows and xbox 360.

Obdrzalek, S., G. Kurillo, F. Oi, R. Bajcsy and E. Seto *et al*., 2012. Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. Proceedings of the Annual International Conference of the Engineering in Medicine and Biology Society, Aug. 28-Sept. 1, IEEE Xplore Press, San Diego, CA, USA, pp: 1188-1193. DOI: 10.1109/EMBC.2012.6346149

Shotton, J., A. Fitzgibbon, M. Cook, T. Sharp and M. Finocchio *et al*., 2011. Real-time human pose recognition in parts from single depth images. Proceedings of the Conference on Computer Vision and Pattern Recognition, Jun. 20-25, IEEE Xplore Press, Colorado Springs, CO, USA, pp: 1297-1304. DOI: 10.1109/CVPR.2011.5995316

Theobalt, C., N. Ahmed, G. Ziegler anad H.P. Seidel, 2007. High-quality reconstruction of virtual actors from multi-view video streams. IEEE Signal Processing Magazine, 24: 45-57.

Viola, P. and M. Jones, 2001. Rapid object detection using a boosted cascade of simple features. Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, Dec. 8-14, IEEE Xplore Press, Kauai, HI, USA. DOI: 10.1109/CVPR.2001.990517

Vlasic, D., I. Baran, W. Matusik and J. Popovic, 2008. Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph.

Wei, X., P. Zhang and J. Chai, 2012. Accurate realtime full-body motion capture using a single depth camera. ACM Trans. Graphics.

Yasin, H., U. Iqbal, B. Krüger, A. Weber and J. Gall, 2015. 3d pose estimation from a single monocular image. CoRR, abs/1509.06720.

Ye, G., Y. Liu, Y. Deng, N. Hasler and X. Ji *et al*., 2013. Free-viewpoint video of human actors using multiple handheld kinects. IEEE Trans. Cybernet., 43: 1370-1382. DOI: 10.1109/TCYB.2013.2272321

Ye, M., X. Wang, R. Yang, L. Ren and M. Pollefeys, 2011. Accurate 3d pose estimation from a single depth image. Proceedings of the International Conference on Computer Vision, Nov. 6-13, IEEE Xplore Press, Barcelona, Spain, pp: 731-738. DOI: 10.1109/ICCV.2011.6126310

Yeung, K.Y., T.H. Kwok and C.C.L. Wang, 2013. Improved skeleton tracking by duplex kinects: A practical approach for real-time applications. J. Comput. Inform. Sci. Eng., 13: 041-051.