Unified Skeletal Animation Reconstruction with Multiple Kinects

Naveed Ahmed[†]

University of Sharjah, UAE

Abstract

We present a new method for reconstructing a unified skeletal animation with multiple Kinects. Our method is able to reconstruct the unified skeletal animation from Kinect data over 360 degrees. 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 animation. Our method is easy to implement and provides a simple solution of creating a 360 degree plausible unified skeletal animation that would not be possible to capture with a single Kinect due to occlusions, tracking failures, and field of view constraints.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion, Sensor Fusion, Time-varying Imagery

1. Introduction

The field of marker-less motion capture and free-viewpoint video has received a lot of interest in the past decade. Traditionally, multi-view RGB camera systems have been used to capture motion, shape, and appearance of a real-world actor [dAST*08] [VBMP08]. Most of these methods use some analysis-through-synthesis based approach for reconstructing these properties.

In recent years, Kinect [MIC10] 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 [GSK*11] [YWY*11] [BMB*11]. Microsoft has been constantly improving the Kinect SDK [MIC10], which can provide real-time pose estimation of a person in standing or sitting position. In general, this single view pose estimation is reliable as long as the user is oriented towards the camera with minimal occlusions [OKO*12]. 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 solves the occlusion problem, but recording with more than one Kinect introduces interference, resulting in 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. Berger et al. [BRS*11] employed four Kinects for unsynchronized marker-less motion capture. Ye et al. [YLD*13] used three hand-held Kinects for marker-less performance capture, whereas Caputo et al. [CDDU12] employed multiple Kinects for hand gesture recognition. None of these methods used the skeletal data provided by Kinect, but the pose was estimated using an optimization process based on silhouettes or human template matching. One of the major benefit of using 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. [OKO*12], 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.

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

[†] nahmed@sharjah.ac.ae

DOI: 10.2312/egsh.20141001

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. 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 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 skeletal animation over 360 degrees that is free from the artefacts due to occlusions or tracking failures.

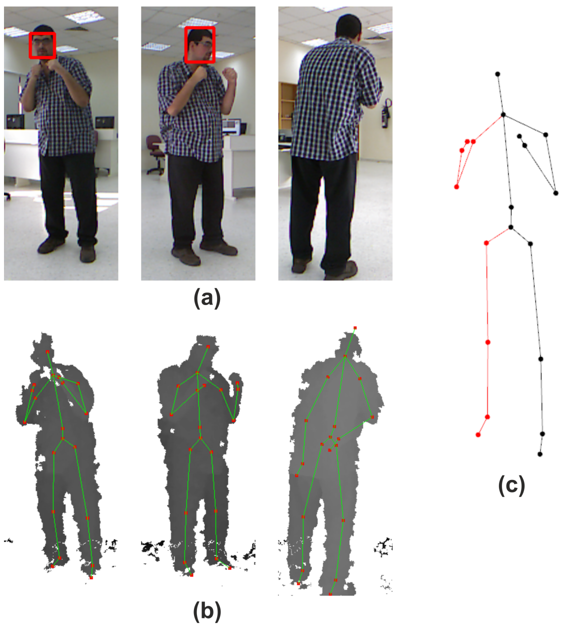

Figure 1: *(a) shows RGB frames from three cameras. Frontal and profile faces are detected in two cameras. (b) shows the depth data with the overlaid skeleton from Kinect. (c) shows the unified skeleton from the two cameras towards which the actor's face is oriented.*

2. 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. [YLD*13] 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. 1a, 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. 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 confidence score assigned to each joint for our method, as discussed in the next section. Our method does not need a fixed extrinsic parameterization between the cameras for the whole sequence. Instead, the mapping is calculated dynamically using the skeleton data as explained in the next section.

3. 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. 1b(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 [VJ01] 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. 1a). 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 Sec. 4. In principle, 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.

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 $\mathcal{C}$ cameras, and there are $\mathcal{T}$ frames in the sequence, the confidence score $\mathcal{S}$ for a joint j_t^c, where j=1,...,20, c=1,...,$\mathcal{C}$, and t=1,...,$\mathcal{T}$, is defined by:

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

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

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

$\mathcal{O}(j_t^c)$ 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. Finally $\mathcal{D}(j_t^c)$ 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 $d_t <= \sigma * d_{t-1}$ then it is set to 1, if $d_t > \sigma * d_{t-1}$ *AND* $d_t <= \rho * d_{t-1}$ 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. For our method, we chose $\sigma = 1.2$ and $\rho = 2.0$. These thresholds can be adjusted depending on the type of the motion.

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. 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. 2a, b. In practice, we always found 12 or more joints with the confidence value greater than 2. Thus, the linear system was never underdetermined.

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. 2a) to estimate the normal $n(j_t^c)$ 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 (Eq. 1) and introduce a new orientation term $\mathcal{N}(j_t^c)$:

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

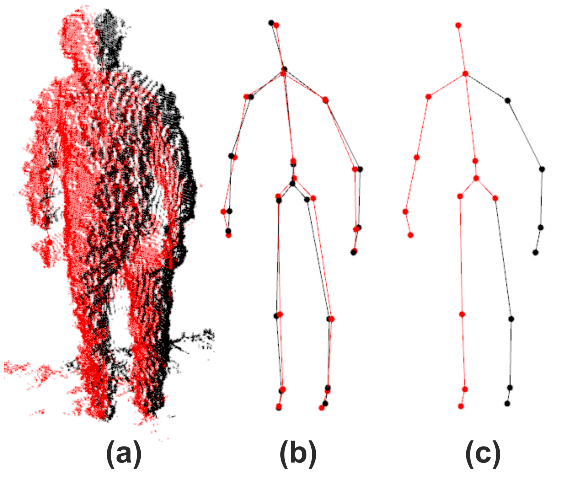

Figure 2: *(a) shows unified two point clouds (shown in black and red), and (b) shows the corresponding two skeletons after the dynamic extrinsic calibration. (c) shows the unified skeleton reconstructed from our method.*

$\mathcal{N}(j_t^c)$ is the dot product of $n(j_t^c)$ and $v(j_t^c)$, where $v(j_t^c)$ is the view vector from j_t^c to the camera c. The maximum value of $(1.0 - \mathcal{N}(j_t^c))$ is 1 if $n(j_t^c)$ 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 $\mathcal{S}(j_t^c)$ for that particular joint (Eq. 2).

4. Results

We recorded two sequences of 200 frames each. First sequence shows a fast boxing motion and the second sequence is a normal walking motion. Our method was able to track both 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. 3. More results can be seen in Fig. 1c, 2c and the accompanying video. 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.

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 video 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.

Despite the limitations, we show that our method is able to reconstruct a unified 360 degree skeletal motion from multiple Kinects in a plausible way that would not be possible with a single Kinect.

5. Conclusions

We presented a method to reconstruct a unified skeletal animation from multiple Kinects. 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, 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. In future, we would like to test our method on varying sequences and perform a quantitative analysis to measure the goodness of the estimated motion.

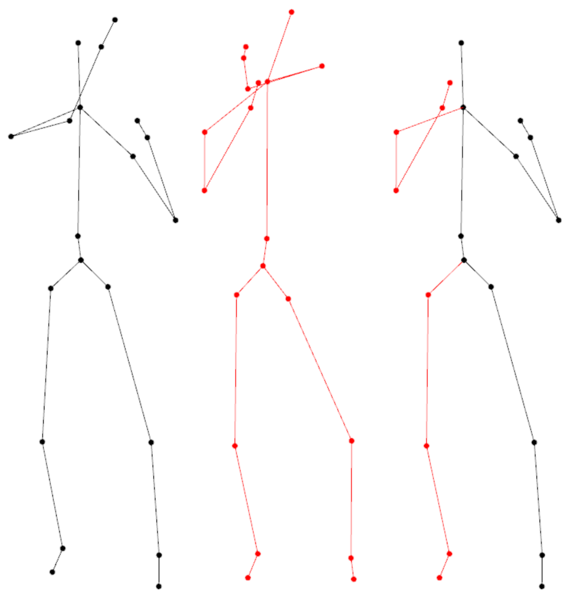

Figure 3: *Tracking fails for the right arm in the front-facing camera due to self-occlusions (left), whereas for the side facing camera the left arm was not tracked because it was not in the view (middle). Our method unified the two skeletons and reconstructed the correct pose (right).*

References

[BMB*11] BAAK A., MULLER M., BHARAJ G., SEIDEL H.-P., THEOBALT C.: A data-driven approach for real-time full body pose reconstruction from a depth camera. In *ICCV* (2011). 1

[BRS*11] BERGER K., RUHL K., SCHROEDER Y., BRUEMMER C., SCHOLZ A., MAGNOR M. A.: Markerless motion capture using multiple color-depth sensors. In *VMV* (2011). 1

[CDDU12] CAPUTO M., DENKER K., DUMS B., UMLAUF G.: 3d hand gesture recognition based on sensor fusion of commodity hardware. In *Mensch and Computer 2012* (2012). 1

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. *ACM Trans. Graph. 27*, 3 (2008). 1

[GSK*11] GIRSHICK R., SHOTTON J., KOHLI P., CRIMINISI A., FITZGIBBON A.: Efficient regression of general-activity human poses from depth images. In *ICCV* (2011). 1

[MIC10] MICROSOFT: Kinect for microsoft windows and xbox 360. http://www.kinectforwindows.org/, November 2010. 1

[OKO*12] OBDRZALEK S., KURILLO G., OFLI F., BAJCSY R., SETO E., JIMISON H., PAVEL M.: 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. 1

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIC J.: Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (2008). 1

[VJ01] VIOLA P., JONES M.: Rapid object detection using a boosted cascade of simple features. In *CVPR* (2001). 2

[YLD*13] YE G., LIU Y., DENG Y., HASLER N., JI X., DAI Q., THEOBALT C.: Free-viewpoint video of human actors using multiple handheld kinects. *IEEE T. Cybernetics* (2013). 1, 2

[YWY*11] YE M., WANG X., YANG R., REN L., POLLEFEYS M.: Accurate 3d pose estimation from a single depth image. In *Proceedings of the 2011 International Conference on Computer Vision* (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 731–738. 1