Real-time Human Motion Tracking using Multiple Depth Cameras

Licong Zhang[1], Jürgen Sturm[2], Daniel Cremers[2], Dongheui Lee[1]

Abstract— **In this paper, we consider the problem of tracking human motion with a 22-DOF kinematic model from depth images. In contrast to existing approaches, our system naturally scales to multiple sensors. The motivation behind our approach, termed Multiple Depth Camera Approach (MDCA), is that by using several cameras, we can significantly improve the tracking quality and reduce ambiguities as for example caused by occlusions. By fusing the depth images of all available cameras into one joint point cloud, we can seamlessly incorporate the available information from multiple sensors into the pose estimation. To track the high-dimensional human pose, we employ state-of-the-art annealed particle filtering and partition sampling. We compute the particle likelihood based on the truncated signed distance of each observed point to a parameterized human shape model. We apply a coarse-to-fine scheme to recognize a wide range of poses to initialize the tracker. In our experiments, we demonstrate that our approach can accurately track human motion in real-time (15Hz) on a GPGPU. In direct comparison to two existing trackers (OpenNI, Microsoft Kinect SDK), we found that our approach is significantly more robust for unconstrained motions and under (partial) occlusions.**

I. Introduction

Motion capture and analysis is a strongly researched field with many applications in areas such as computer animation, video games, medical therapy, tele-presence, surveillance and human machine interaction [1], [2], [3]. Various approaches have been proposed to push the performance gradually towards accurate, stable and real-time motion capturing with simple setup procedures. However, most commercially available real-time motion tracking systems are marker-based, which require the actors to wear obtrusive devices. As a result, such systems are relatively complex, expensive and difficult to maintain. Therefore, marker-less, camera-based tracking systems are in high demand, but are often challenged by problems such as occlusion, ambiguity, lighting conditions and dynamic objects. Depth sensors such as time-of-flight sensors provide additional information about the 3D shape of the scene. Recently, several approaches have demonstrated that reliable, real-time human pose recognition is feasible [10], [12], [13]. However, all of these approaches only use a single sensor and are thus still sensitive to partial (self-)occlusions. With the recent market launch of low-cost depth sensors such as the Microsoft Kinect and Asus Xtion sensor, the question in our point of view is how these methods can be extended to multi-sensor setups.

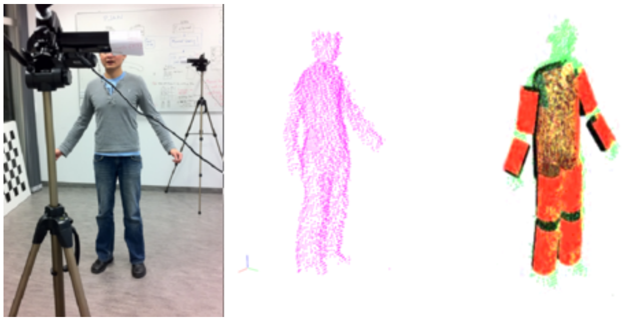

Fig. 1. Motion capture with multiple depth cameras significantly reduces ambiguities and inaccuracies due to occlusions. From left to right: the human actor, the joint point cloud observed by the two Kinect sensors and the estimated human pose.

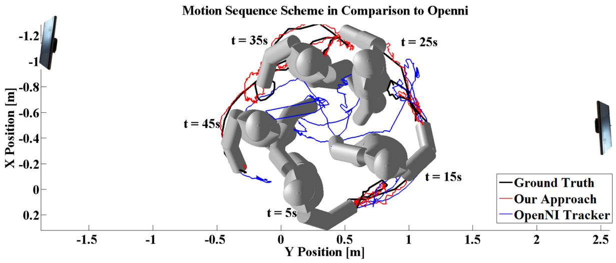

Fig. 2. Setup used for evaluating our approach. Two Kinect sensors are placed in opposite corners of a room. Our approach is robust against (self-)occlusions (red) while existing approaches only use a single sensor (on the right) and are far more sensitive (blue, OpenNI). The ground truth is indicated in black.

In this paper we contribute a novel approach, named Multiple Depth Camera Approach (MDCA), to track human motion from multiple Kinect sensors (Fig. 1). Our approach includes fast data fusion and association, a flexible shape model and an efficient estimation method based on particle filtering. The goal of our approach is to track the human pose with 22 DOF. To efficiently track the human pose, we use annealed particle filtering in combination with partitioned sampling. Furthermore, we implemented a simple coarse-to-fine search to robustly detect the initial pose from a wide range of configurations. Our system is able to track human motion in real-time (15Hz) using a GPGPU. In the experimental evaluation, we found that our tracker clearly outperforms existing state-of-the-art trackers such as the Microsoft Kinect SDK or the OpenNI tracker in terms of robustness against occlusion and ambiguity resolution. In particular, our system can continuously track a 360° human walking motion (shown in Fig. 2) which is not possible with

[1] Licong Zhang and Dongheui Lee are with the Department of Electrical Engineering and Information Technology, Technical University of Munich, Germany. {licong.zhang,dhlee}@tum.de

[2] Jürgen Sturm and Daniel Cremers are with the Computer Vision Group at the Computer Science Department, Technical University of Munich, Germany. {sturmju,cremers}@in.tum.de

This research is partly supported by the DFG excellence initiative research cluster "Cognition for Technical Systems CoTeSys" and Institute of Advanced Study, TUM

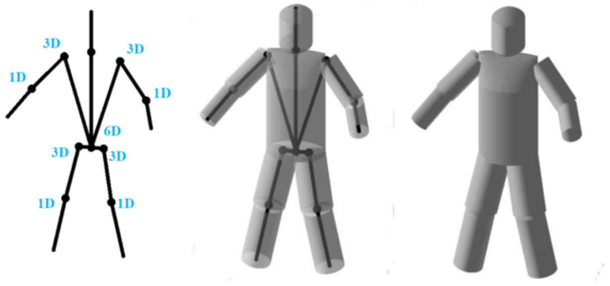

Fig. 3. Our human model. From left to right: the kinematic model, the shape model based on the kinematic model, the shape model.

the other systems.

II. Related Work

Several surveys [1], [2], [3] provide an excellent overview covering the sensors, models and estimation methods in motion tracking. We only focus our review on marker-less approaches. Camera-based methods have a long history [4], [5], [6], [7], [8]. However, monocular setups are highly sensitive to occlusions, while multi-view methods are in general computationally expensive. Recently, depth sensors such as time-of-flight cameras [12] and structured-light systems [10] are gaining interest as they directly provide the 3D geometry of the scene which simplifies segmentation, detection and tracking significantly.

For modeling and state estimation, two alternative approaches exist: Top-down approaches have an explicit model of the human body. Sequential Monte Carlo methods [14] such as particle filters are often used to track the human pose, with several extensions to tackle the high dimensional search space and to enhance the efficiency of the sampling [15], [16], [17]. In contrast, bottom-up approaches [10], [11] aim at the detection of body parts from the sensor data, with no or only a minimal model of the human body. Shotton et al. [10] learn decision trees for pixel-wise labeling of input images. While the training process requires an enormous amount of training images and computational resources, the runtime classification is highly efficient ($>$ 30Hz). Plagemann et al. [11] developed an approach based on the interest point detector. There has also been increasingly more works combining both kinds of approaches [12], [13]. Plagemann et al. integrated the interest point approach with a particle filter [12] and were able to track a 48-DOF human pose at 4 to 10 Hz on a GPGPU. Baak et al. [13] extend this idea by means of a (pre-trained) pose database from which they can efficiently obtain relevant pose estimates.

All approaches mentioned above operate on monocular depth images and it is not clear how they can be applied in a multi-view setup. In contrast, we fuse the individual depth images to a joint point cloud and use an efficient particle filtering approach for pose estimation. In contrast to previous work, this allows us to estimate the human pose from all available data to increase the robustness against occlusions. To the best of our knowledge, this is the first work to use multiple depth cameras for human motion capture.

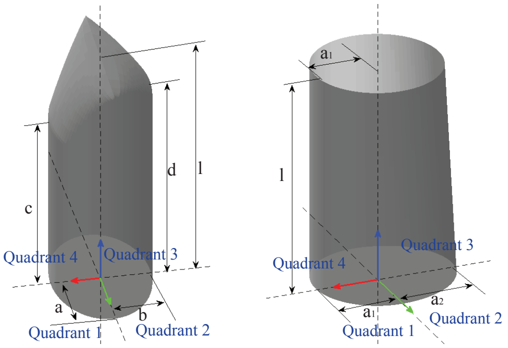

Fig. 4. The parameterizations of the torso and upper leg models from left to right

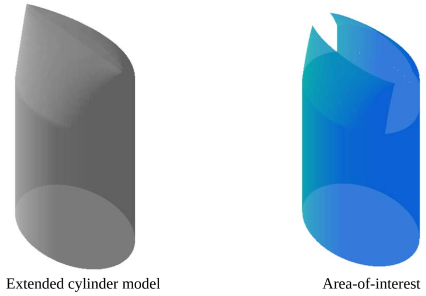
Extended cylinder model
Area-of-interest

Fig. 5. Illustration of the idea of the area-of-interest in torso. The surface area near the shoulder joints are excluded from the original model.

III. Human Model

The first step for detecting and tracking persons from depth data is to define a human model that describes the 3D appearance of the human. This model thus defines both the kinematics of the human body as well as its shape. Fig. 3 schematically depicts our model.

A. Kinematic Model

In particular, we employ a kinematic tree model with 22 DOF, consisting of joints and body parts (torso and limbs), originating from the pelvis. In total, our model has 9 body parts, namely the torso, the right and left upper arm, lower arm, upper leg and lower leg. These body parts are referred to with the abbreviations {t, rua, rla, lua, lla, rul, lul, rll, lll} in the reminder of this paper. The pose of this model is then defined by a 22-dimensional vector $\mathbf{x} \in \mathbb{R}^{22}$ describing the configurations of the joints. Note that most joints have 3 DOF, while others have only one (see Fig. 3). Furthermore, we enforce certain constraints on this configuration vector (e.g. joint limits). Each body part is defined by the static transformation between the two adjacent joints, which is defined by a set of kinematic parameters $\boldsymbol{\theta}_{kin}$.

B. Shape Model

We assign to each body part a local coordinate system which originates at the beginning of the body part (see Fig. 4 for torso and upper leg) with the z-axis on the central axis of the body part and the x-y plane perpendicular to it. Based on this coordinate system, we define the shape of the body part using an extended cylinder model.

Suppose that $\boldsymbol{y} = (x, y, z)^T \in \mathbb{R}^3$ denotes the position of a 3D point expressed in the local coordinate system of a body part. If the body part is a perfect cylinder with height l and radius r, a point lying on the surface is characterized by

$$\frac{x^2}{r^2} + \frac{y^2}{r^2} = 1 \text{ and } 0 \leq z \leq l \quad . \tag{1}$$

To enhance the flexibility, we extend the circle to a shape consisting of 4 quarter ellipses (one in each quadrant). Furthermore, the half-radii of the quarter ellipses $a_j f_j(z)$ and $b_j g_j(z)$ ($j \in [1, 4]$ as quadrant index) depend on the z-value of the point in the local coordinate system. The extended model is thus given by

$$\frac{x^2}{a_j^2 f_j^2(z)} + \frac{y^2}{b_j^2 g_j^2(z)} = 1 \text{ and } 0 \leq z \leq l \quad . \tag{2}$$

The functions $f_j(z)$ and $g_j(z)$ can be customized to fit to different real human body part shapes. Using this model, a large variety of different shapes can be expressed. For example, in the torso of our human model, shown in Fig. 4, $f_j(z)$ and $g_j(z)$ are defined as

$$f_1(z) = f_4(z) = \begin{cases} 1 & \text{if } 0 \leq z < c \\ \sqrt{\frac{l-z}{l-c}} & \text{if } c \leq z \leq l \end{cases} \tag{3}$$

$$f_2(z) = f_3(z) = \begin{cases} 1 & \text{if } 0 \leq z < d \\ \sqrt{\frac{l-z}{l-d}} & \text{if } d \leq z \leq l \end{cases} \tag{4}$$

$$g_1(z) = g_2(z) = g_3(z) = g_4(z) = 1 \tag{5}$$

where c and d are the built-in parameters of the functions $f_j(z)$. In the upper legs, we use a similar model but with $f_1(z)$ and $f_4(z)$ as the constant and $f_2(z)$ and $f_3(z)$ as linear functions (Fig. 4). The arms and lower legs are modeled as perfect cylinders in our human model.

Although this model ensures a high degree of flexibility, we found that the regions close to the joints are difficult to model due to the deformability of the tissue. To tackle this shortcoming, we automatically exclude regions near the joints as illustrated in Fig. 5 by computing areas-of-interest on the surface. In our observation model as described in the next section, only the areas of interest will be considered for the computation of the particle likelihood.

We use $\boldsymbol{\theta}_{shape}$ to refer to the parameters used to define the shape model of all body parts. These parameters are customized to fit to the real shape of the body parts of the motion demonstrator.

For each human motion demonstrator, a specific set of human model parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{kin}^T, \boldsymbol{\theta}_{shape}^T)^T$ is needed. Finding these parameters is beyond the scope of this paper. However both manual calibration and automatic shape fitting can be used for this aim.

Compared to cylindrical or mesh-based shape modeling, we found that our shape model provides a much larger flexibility to adapt to the human body with different build and shape, as illustrated in Fig. 6. Furthermore, the mathematical definition of the surface shape is still very simple and can thus be efficiently evaluated (and parallelized).

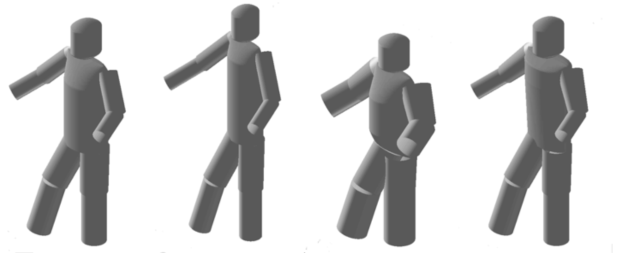

Fig. 6. Examples of kinematic and shape parameters adapted to different body types.

IV. Pose Estimation

Our pose estimation framework is based on particle filtering [14]. The objective in pose estimation is to estimate the 22-DOF configuration of the human model according to the sensor data. We define the configuration of the human model as the state $\mathbf{x} \in \mathbb{R}^{22}$. Further, we define a point cloud $\mathbf{y}$ as an unordered sequence of 3D points, i.e., $\mathbf{y} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n)})$ with $\mathbf{y}^{(i)} \in \mathbb{R}^3$. Further, the state is represented by an unordered set of particles $\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)})$, each of which with an associated importance weight $w^{(j)} \in \mathbb{R}^+$.

In each time step, the particles from the previous step are propagated further by the *motion model* $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. Subsequently, the importance weights are updated with the *observation model* $p(\mathbf{y}_k|\mathbf{x}_k)$. Both steps will be discussed in detail in the remainder of this chapter.

To tackle the problem of high dimensionality, we combine annealed particle filtering [15], [18] with partitioned sampling [16]. In particular, we split the 22-dimensional search space hierarchically into 5 subspaces, and run individual particle filters in each subspace. The torso ($\mathbb{R}^6$) forms the root, after which the right arm ($\mathbb{R}^4$), the left arm ($\mathbb{R}^4$), the right leg ($\mathbb{R}^4$) and the left leg ($\mathbb{R}^4$) can be estimated in parallel. In each subspace, we use an annealed particle filter, which employs a multi-layered search to gradually concentrate the particle density on areas with the highest probability.

A. Motion Model

From time $k-1$ to k, we propagate all particles according to the motion model

$$\mathbf{x}_k^{(j)} = \mathbf{x}_{k-1}^{(j)} + \delta^{(j)} \tag{6}$$

where $\delta^{(j)} \sim \mathcal{N}(0, \Sigma)$ is normally distributed and Σ is a diagonal matrix with its diagonal entries taking the values σ_j^2 corresponding to the (assumed) agility of each joint. Despite the simplicity of the motion model, we found in our experiments that Gaussian white noise works well in practice. We also experimented with a constant velocity model, but achieved worse results. We believe that the reason for this is twofold: first, humans can too quickly (de-)accelerate their

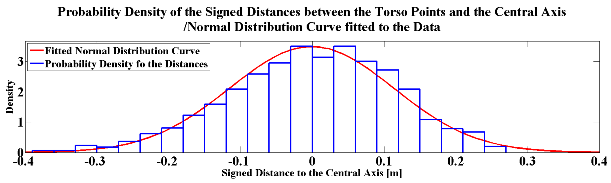

Fig. 7. Empirical validation of the probabilistic model. The columns show the probability density of the distance observed for the torso. The sample distribution (blue bins) is well approximated by a normal distribution (red).

limbs and second, the estimated joint-space velocities are (due to the particle filter sampling) relatively noisy.

B. Observation Model

The objective of the observation model is to compare the pose estimates with the sensor data and calculate a likelihood $p(\mathbf{x}_k^{(j)} \mid \mathbf{y}_k)$ for each particle j. Note that in the remainder of this section, we omit the subscripts k and $k-1$ to improve readability. Based on the shape model derived in the previous section, the likelihood can be evaluated very efficiently as follows. For the moment, we only consider a single point $\mathbf{y}^{(i)}$ from the point cloud and a single body part $q \in \{t, rua, rla, lua, lla, rul, lul, rll, lll\}$. Further, we assume that the transformation between the global coordinate frame and the limb is given by a 4×4 matrix $M_q(\mathbf{x}^{(j)}) \in \mathrm{SE}(3)$ so that the point can be expressed in the limb coordinate system as $\mathbf{y}_q^{(i)} = M_q(\mathbf{x}^{(j)})\mathbf{y}^{(i)} = (x, y, z)^T$. Our goal is now to compute the *signed distance* of this point to surface of the body part. Here, *signed* means that we assume the distance is negative when the point lies within the body part, zero exactly on its surface and positive on the outside. This signed distance can be computed using (we omit the quadrant index for readability)

$$d(\mathbf{y}_q^{(i)}; q) = \sqrt{\frac{x^2}{a^2 f^2(z)} + \frac{y^2}{b^2 g^2(z)}} - 1 \quad . \tag{7}$$

Note that (7) is not the Euclidean distance, but the distance scaled according to the semi-axes of the corresponding ellipse. Also, the range of the signed distance is $d(\mathbf{y}^{(i)}) \in [-1\ \infty)$. By construction, $d(\cdot)$ is monotonically increasing if a point is moving away from the central axis and takes the value 0 if the point lies on the surface.

In the ideal case, an observed (noise-free) point should be located exactly on the surface ($d(\cdot) = 0$) if the corresponding pose estimate is optimal. However, discrepancies between the real surface of the human body part and the modeled one exist since our shape model is not perfect. Moreover, all real sensor data will always contain noise and outliers. As a consequence, the distance will behave according to some noise distribution depending on the sensor type and the accuracy of the human model. In our system, we assume that this probability distribution is a zero-mean Gaussian. Given this, we can compute the likelihood of point $\mathbf{y}^{(i)}$ being observed from body part q and with a given particle $\mathbf{x}^{(j)}$ as

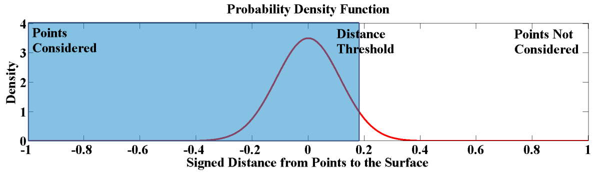

Fig. 8. Robust observation likelihood. Only points in the blue area are considered during likelihood computation.

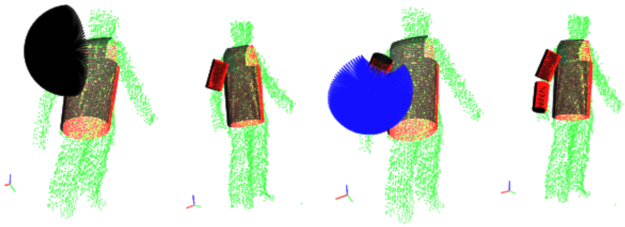

Fig. 9. Our coarse-to-fine scheme allows us to initialize the tracker automatically from a wide range of initial configurations. Here, the configuration of the upper arm is determined in two steps (left) and subsequently the lower arm (right).

$$p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(j)}, q) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{d^2(M_q(\mathbf{x}^{(j)})\mathbf{y}^{(i)}; q))}{2\sigma^2}\right) \tag{8}$$

where σ^2 is the variance. Fig 7 shows the empirical validation of this assumption on real sensor data. It can be observed from the figure that for the torso part, the probabilistic density function can be decently approximated by a zero-mean Gaussian distribution.

C. Data Association

The prerequisite of the validity of this probabilistic model is that the points of concern actually belong to the corresponding body part. Without data association, it is unknown to which body part a point actually belongs. While the problem of data association is very difficult in the general case, we pursue a simple strategy where we treat points with a signed distance above a certain threshold as unrelated points that can be ignored. Provided the tracking result of the previous frame is acceptable, we can reasonably assume that the shape model of a specific body part rendered by the pose estimate is near the real human body part of the current frame, in which case the absolute value of the relative distance of the points belonging to the corresponding body parts are close to 0. In contrast, points of other body parts have much larger values. Therefore we can implement data association via a distance threshold and consider only the points within the valid range.

$$\begin{aligned} A_q := \{\ i \quad | \quad & i \in \{1, \dots, n\} \text{ with } d(\mathbf{y}^{(i)}) < \alpha \\ & \text{and } i \notin \cup_{s=1}^{i-1} A_s \} \quad . \end{aligned} \tag{9}$$

Here we choose the value of α manually according to experimental experience. This idea is also shown in Fig. 8.

The likelihood that the shape model of a specific body part match the sensor data can be obtained as

$$p(\mathbf{y} \mid \mathbf{x}^{(j)}, q) = \prod_{i \in A_q} p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(j)}, q) \tag{10}$$

where only the related points are considered, i.e., points within the area of interest and the valid range of the signed distance. Finally, this gives us the full observation model required to compute the likelihood of a particle

$$p(\mathbf{y} \mid \mathbf{x}^{(j)}) = \sum_{q} p(q \mid \mathbf{x}^{(j)}) p(\mathbf{y} \mid \mathbf{x}^{(j)}, q) \tag{11}$$

with which the particle weights can be updated accordingly.

The computational cost of this algorithm depends on the number of points in the joint point cloud, i.e., it is $\mathcal{O}(\beta \times n)$, with a setup consisting of β cameras and an average number of points in a point cloud from a single camera of n.

D. Tracking Initialization

To find the initial pose of a motion sequence, we developed a coarse-to-fine search paradigm that can detect the pose of a human from a single frame of data. To this aim, we implemented a hierarchical search down the kinematic tree. To assist detection of the torso orientation, we employ recursive principal components analysis. Then for each subspace of the configuration, we divide the search into two phases. In the coarse search phase, we run a particle filter on a uniform prior distribution, in which case the estimates generated span the entire search space uniformly. Subsequently, we resample the resulting particle set using a Gaussian distribution to further optimize the pose estimates. This method is able to detect the configuration of a wide range of human poses from a single frame of sensor data, provided that the human is standing relatively straight and no ambiguous poses are involved. Fig. 9 visualizes the basic idea of the coarse-to-fine search of the right arm configuration.

V. Experiment

We conducted a series of experiments to verify that (a) our approach can accurately and robustly track human motions, (b) tracking is possible in real-time and (c) that a multi-camera setup is significantly more robust than approaches using only single depth camera.

A. Data Acquisition

Our experimental setup is depicted in Fig. 1. Two Kinect sensors were placed face-to-face and extrinsically calibrated using a checkerboard. With this setup, we recorded several motion sequences with increasing complexity.

To evaluate our approach and compare it to other systems, we selected four distinct points on the human actors, namely, the right wrist (RW), left wrist (LW), right ankle (RA) and left ankle (LA). The positions of these marker points are provided by all three tracking systems (MDCA, OpenNI tracker, Microsoft SDK), and moreover, can easily be identified manually. The positions of the markers on the three systems are shown in Fig. 10. For our evaluation sequences, we manually tracked these points in every tenth frame using a GUI. The OpenNI tracker only offers one marker at the end of each limb (hand and foot). The Microsoft SDK offers two markers, the hand(foot) and wrist(ankle) marker. In the comparison, we choose the wrist(ankle) marker since this corresponds to our ground truth marker and the kinematics of our approach. Experimental results show that average error of the wrist(ankle) marker is smaller than the hand(foot) marker in Microsoft SDK.

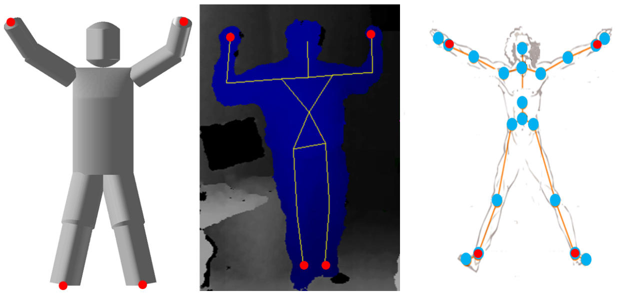

Fig. 10. Position of the corresponding markers on the three systems, from left to right, our approach, the OpenNI tracker and the Microsoft SDK.

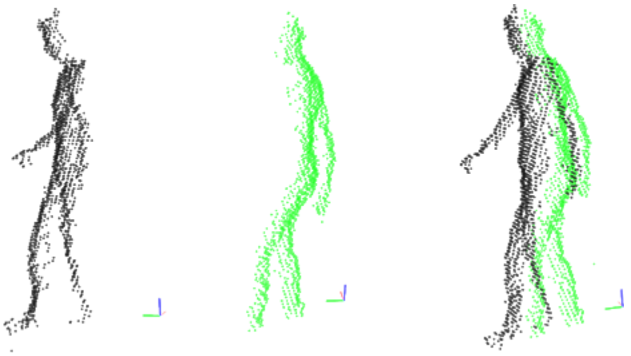

Fig. 11. Joint point cloud. The first two figures shows the point clouds from each individual Kinect sensor. The last figure shows the merged point cloud.

Our own tracking algorithm is implemented as a C++ program which can run either on a single CPU core or parallelized on a GPGPU. In our approach, we down-sample the depth images by a factor of 16 and then apply a simple segmentation to obtain the human point cloud. The segmented depth images were then converted to point cloud and transformed into the global coordinate system as shown in Fig 11.

B. Tracking Accuracy

We tested our program on different real life motion sequences with normal motion speed (e.g. walking, exercising, Taiji). The experiments show that the tracker can successfully track all these motion sequences with none or few loss of track in certain frames.

Fig. 12 shows the tracking accuracy of the positions of the 4 markers in a motion sequence of a human walking in circles. The plot shows that the tracking error rarely exceeds 0.2m and is most of the time clearly below 0.1m. This result can also be confirmed in Table. I, where the average

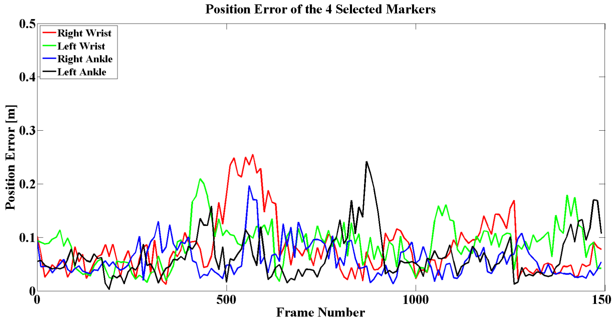

Fig. 12. The position error of the 4 markers compared to the ground truth in a motion sequence of a human walking in circles.

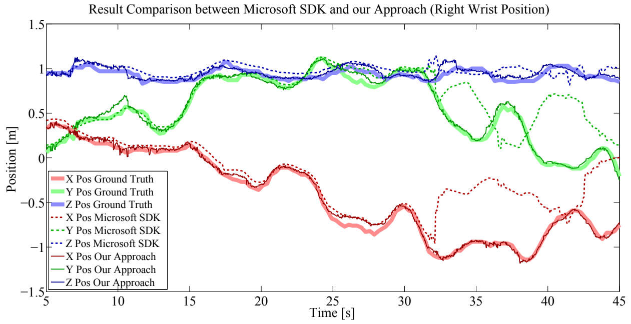

Fig. 13. The position trajectory of the Microsoft SDK and our approach compared to the ground truth of the right wrist position.

error over all markers is 0.073m. Interestingly, the ankles are tracked by 15% more accurate than the wrists. One reason for this might be that more intensive arm movements than leg movements are involved in this motion pattern. Furthermore, tracking is on average only lost in 2.1% of the frames, despite the occlusions due to the 360° walking pattern. From these results, we conclude that our approach is both accurate and robust.

C. Tracking Speed

The average processing time per frame of our program on the CPU is 0.5s. If the program is run on GPGPU (a NVidia GeForce GTX 480 card), this number is reduced to 0.05s. Stable tracking can thus be guaranteed at 15Hz. The processing time also depends on the parameters in the tracking algorithm, primarily on the number of particles used. The more particles used, the longer the processing time and vice versa. We use 2000 particles in the experiments. It is also influenced by the number of sensors used. Using more sensors should improve the tracking performance, but it also increases the computational cost and thus reducing the operating speed.

D. Comparison with other Approaches

Furthermore, we compare our tracking approach (MDCA) with two state-of-art tracking systems using single sensors: the Microsoft Kinect SDK [1] and the PrimeSense OpenNI tracker[2].

Because of the given limitations of the underlying APIs, we were only able to compare our approach to one system at a time. Therefore, the two similar but not identical motion sequences of a human walking in cycles were used for comparison. One of them is shown in Fig. 2.

Fig. 13 shows the comparison between the results of the Microsoft Kinect SDK and our approach for the right wrist position. We can observe from this figure that both approaches can successfully capture motion at the beginning. But from the 30th second, the result of the Microsoft SDK deviates from the ground truth and loses track completely, while our approach continues to track successfully. This is an expected result, since from $t = 30s$, the right arm is occluded in certain frames by the torso and no longer observable from the camera on the right (see Fig. 2). As a result, tracking temporarily fails. The Microsoft approach, however, quickly recovers as soon as the occluded parts become observable again. This experiment shows that our approach outperforms in dealing with occlusion problems and unconstrained (360°) motions. In particular, as shown in Table I, our approach looses on average track only in 1.3% of all frames, while the Microsoft tracker has lost in more than 25% of the trajectory. Moreover, we also (qualitatively) observe that our approach yields similar tracking accuracies, if not better, with the Microsoft approach in case of no occlusions.

Similar results can be observed in Fig. 14 and Table II, which show the same comparison but now between our approach and the OpenNI tracker. It clearly shows that in this particular motion sequence, where many occlusion scenarios in all the markers are involved, our approach has better performance both in the average tracking errors (0.073m vs. 0.33m) and less loss (2.1% vs. 42.7%).

It should be noted that two factors may affect the motion capture result collected from the Microsoft SDK and OpenNI tracker. Firstly, the results of both approaches are transformed into the coordinate system used by our approach. The transformation we obtained with a standard checkerboard may contain minor inaccuracies. Secondly, there could be small discrepancies between the positions of the wrists and ankles on the human body in different systems. However, these two factors can only have minor influence on the accuracies in final result, and certainly cannot explain large tracking errors of above 0.2m in 25% and 42% of all frames, respectively. Therefore, draw two conclusions from these results: First, existing single-sensor trackers are highly sensitive to occlusions, and second, multi-sensor setups, as implemented by our approach, are significantly more robust.

From this analysis we can see that our approach is better equipped to deal with occlusion problems, especially when the body parts are completely blocked from one view point. This is mainly because we use two Kinect sensors and thus can obtain more information on the human motion. The temporal continuality of the tracking approaches also contributes to this aspect. Moreover, compared to others, our approach can easily deal with the situation of the data fusion

[1] http://www.microsoft.com/en-us/kinectforwindows/a
[2] http://openni.org/docs2/Reference/smpl_user_tracker.html

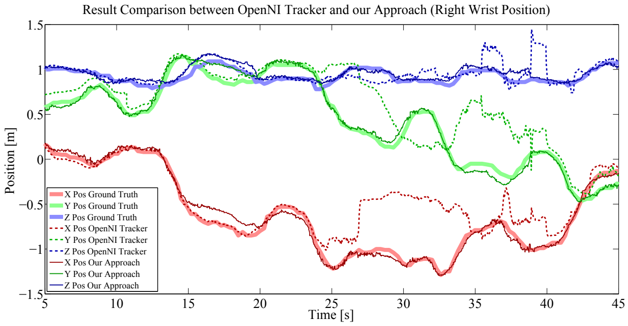

Fig. 14. The position trajectory of the OpenNI Tracker and our approach compared to the ground truth the right wrist position.

TABLE I
COMPARISON OF MDCA WITH MICROSOFT KINECT SDK.

	Error	<0.1m	<0.15m	<0.2m	≥0.2m	Avg.
Marker	Approach	[%]	[%]	[%]	[%]	[m]
RW	MDCA	84.0	98.0	100.0	0.0	0.063
	MS-Kinect	49.3	61.3	66.0	34.0	0.310
LW	MDCA	62.6	97.3	100.0	0.0	0.087
	MS-Kinect	33.3	67.3	70.0	30.0	0.226
RA	MDCA	94.0	98.7	100.0	0.0	0.056
	MS-Kinect	29.3	54.6	77.3	22.7	0.168
LA	MDCA	81.3	92.6	94.6	5.4	0.076
	MS-Kinect	8.0	67.3	83.3	16.7	0.159
Avg.	MDCA	80.5	96.7	98.7	1.3	**0.071**
	MS-Kinect	30.0	62.6	74.2	25.8	**0.216**

TABLE II
COMPARISON OF MDCA WITH THE OPENNI TRACKER.

	Error	<0.1m	<0.15m	<0.2m	≥0.2m	Avg.
Marker	Approach	[%]	[%]	[%]	[%]	[m]
RW	MDCA	77.3	89.3	94.0	6.0	0.081
	OpenNI	27.3	53.3	62.0	38.0	0.308
LW	MDCA	63.3	94.0	98.7	1.3	0.086
	OpenNI	4.7	34.0	43.3	56.7	0.479
RA	MDCA	90.0	98.0	100.0	0.0	0.059
	OpenNI	8.7	36.7	63.4	36.6	0.292
LA	MDCA	83.3	94.7	98.7	1.3	0.066
	OpenNI	0.7	16.0	60.7	39.3	0.254
Avg.	MDCA	78.5	94.0	97.9	2.1	**0.073**
	OpenNI	10.4	35.0	57.3	42.7	**0.333**

of two or more Kinect sensors.

VI. CONCLUSIONS

In this paper, we proposed a motion tracking approach that can be used with any number of Kinect sensors. We proposed a novel, flexible human surface modeling method that we integrated into a probabilistic observation model. We use annealed particle filtering in combination with partition sampling to track the 22-DOF human pose in real-time using GPGPU support. Furthermore, we proposed a simple coarse-to-fine scheme for finding the initial pose of a motion sequence from a wide range of initial poses. In an extensive set of experiments on real data, we demonstrated that our approach is robust and accurate, and performs significantly better in the presence of occlusions than current state-of-the-art implementations of single-sensor trackers.

REFERENCES

[1] T. Moeslund, A. Hilton, V. Krueger. A Survey of Advances in Vision-based Human Motion Capture and Analysis. Journal on Computer Vision and Image Understanding, Volume 104 Issue 2, 2006.

[2] R. Poppe. Vision-based Human Motion Analysis: An Overview. Journal on Computer Vision and Image Understanding, Volume 108 Issue 1-2, 2007.

[3] C. Cedras, M. Shah. Motion-based Recognition a Survey. Journal on Image and Vision Computing, Volume 13, Issue 2, Pages 129-155, 1995.

[4] C. Sminchisescu, B. Triggs. Covariance Scaled Sampling for Monocular 3D Body Trakcing. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Volume 1, Pages I-447-I-454, 2001.

[5] A. Agarwal, B. Triggs. Recovering 3D Human Pose from Monocular Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 28, Issue 1, Pages 44-58, 2006.

[6] D. Lee and Y. Nakamura. Motion Capturing from Monocular Vision by Statistical Inference Based on Motion Database: Vector Field Approach. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Pages 617-623, 2007.

[7] D. Gavrila, L. Davis, 3-D Model-based Tracking of Humans in Action: A Multi-view Approach. In Proceedings of the Conference on Computer Vision and Pattern Recognition, San Francisco, Pages 73-80, 1996.

[8] R. Kehl, L. Van Gool, Markerless Tracking of Complex Human Motions from Multiple Wiews. Journal on Computer Vision and Image Understanding, Volume 104, Issue 2, Pages 190-209, 2006.

[9] D. Glas, T. Miyashita, H. Ishiguro, N. Hagita, Laser Tracking of Human Body Motion Using Adaptive Shape Modeling. In Proceedings of IEEE Conference on Intelligent Robots and Systems, Pages 602-608, 2007.

[10] J. Shotton, A. Fitzgibbon, M.t Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. IEEE Conference on Computer Vision and Pattern Recognition, Pages 1297-1304, 2011.

[11] C. Plagemann, V. Ganaphathi, D. Koller, S. Thrun. Real-time Identification and Localization of Body Parts from Depth Images. IEEE Conference on Robotics and Automation, Pages 3108-3113, 2010.

[12] V. Ganaphathi, C. Plagemann, D. Koller, S. Thrun. Real Time Motion Capture Using a Single Time-of-flight Camera. IEEE Conference on Computer Vision and Pattern Recognition, Pages 755-762, 2010.

[13] A. Baak, M. Mueller, G. Bharaj, H. Seidel, C. Theobalt. A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera. IEEE International Conference on Computer Vision, Pages 1092-1099, 2011.

[14] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo Sampling Methods for Bayesian filtering. Journal on Statistics and Computing Volume 10, Issue 3, Pages 197-208, 2000.

[15] J. Deutscher, A. Blake, I. Reid. Articulated body Motion Capture by Annealed Particle Filtering. IEEE Conference on Computer Vision and Pattern Recognition, Volume 2, Pages 126-133, 2000.

[16] J. MacCormick , M. Isard, Partitioned Sampling, Articulated Objects and Interface-quality Hand Tracking. In Proceedings of the 6th European Conference on Computer Vision-Part II, Pages 3-19, 2000.

[17] J. Mitchelson, A. Hilton. Simultaneous Pose Estimation of Multiple People Using Multiple-view Cues with Hierachical Sampling. In Proceedings of the British Machine Vision Conference, Pages 67.1-67.10, 2003.

[18] J. Bandouch, F. Engstler, M. Beetz. Evaluation of Hierarchical Sampling Strategies in 3D Human Pose Estimation. In Proceedings of the 19th British Machine Vision Conference, 2008.

[19] T. Brox, B. Rosenhahn, J. Gall, D. Cremers. Combined region- and motion-based 3D tracking of rigid and articulated objects. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 32, Issue 3, 2009.

[20] A. Johansen. SMCTC: Sequential Monte Carlo in C++. Journal of Statistical Software, Volume 30, Issue 6, 2009.