Personalization and Evaluation of a Real-time Depth-based Full Body Tracker

Thomas Helten[1] Andreas Baak[1] Gaurav Bharaj[2] Meinard Müller[3] Hans-Peter Seidel[1] Christian Theobalt[1]

[1]MPI Informatik
Saarbrücken, Germany

[2]Harvard University
Cambridge, MA, USA

[3]International Audio Laboratories Erlangen
Erlangen, Germany

{thelten,abaak,theobalt}@mpi-inf.mpg.de gaurav@seas.harvard.edu meinard.mueller@audiolabs-erlangen.de

Abstract

Reconstructing a three-dimensional representation of human motion in real-time constitutes an important research topic with applications in sports sciences, human-computer-interaction, and the movie industry. In this paper, we contribute with a robust algorithm for estimating a personalized human body model from just two sequentially captured depth images that is more accurate and runs an order of magnitude faster than the current state-of-the-art procedure. Then, we employ the estimated body model to track the pose in real-time from a stream of depth images using a tracking algorithm that combines local pose optimization and a stabilizing database look-up. Together, this enables accurate pose tracking that is more accurate than previous approaches. As a further contribution, we evaluate and compare our algorithm to previous work on a comprehensive benchmark dataset containing more than 15 minutes of challenging motions. This dataset comprises calibrated marker-based motion capture data, depth data, as well as ground truth tracking results and is publicly available for research purposes.

I. Introduction

Tracking 3D human motion data constitutes an important strand of research with many applications to computer animation, medicine or human-computer-interaction. In recent years, the introduction of unexpensive depth cameras like Time-of-Flight cameras [1] or the Microsoft Kinect has boosted the research on monocular tracking since they constitute comparably cheap to obtain so-called 2.5 dimensional depth maps. Tracking from such depth input is especially appealing in home consumer scenarios, where a user controls an application only by using his own body as an input device and where complex hardware setups are not feasible.

While depth data facilitates background subtraction compared to pure image based approaches, tracking still remains challenging because of the high dimensionality of the pose space and noise in the depth data. Currently, there exist two different strategies to harness depth data for tracking human motions. Bottom-up approaches detect body parts or joint-positions directly from the depth images. Such approaches often neglect the underlying skeletal topology of the human which may lead to improbable joint locations and jitter in the extracted motion. Top-down approaches fit a parametric model to the depth data using an optimization scheme. Here, the accuracy of the final tracking result is dependent on the degree to which the body model matches the true body shape of the person. In practice, such models are often obtained in a preprocessing step, *e. g.*, using laser scanners which are not available in home consumer scenarios.

Recently, first attempts have been made to obtain the shape of a person by fitting a parametric model to a set of depth images of a strictly defined calibration pose. However, the runtime in the orders of one hour as well as the requirement of a fixed calibration pose limit the applicability in a practical scenario.

Contributions.: We contribute with algorithmic solutions that improve the performance of a combined discriminative and generative real-time tracker. Firstly, we present a new shape estimation method that makes model fitting an order of magnitude faster compared to previous approaches [2] at no loss of quality. Secondly, we extend an existing tracking algorithm by [3] to obtain a personalized version that works with arbitrary body shapes. As another contribution, we deployed an extensive dataset of 15 minutes of calibrated depth and marker-based motion capture (mocap) data which was used to evaluate our proposed tracker and which will be made publicly available to the research community. We also contribute with suitable error metrics to make different trackers comparable on our data set.

The remainder of the paper is organized as follows. After discussing related work, we present our novel shape

978-0-7695-5067-1/13 $26.00 © 2013 IEEE
DOI 10.1109/3DV.2013.44

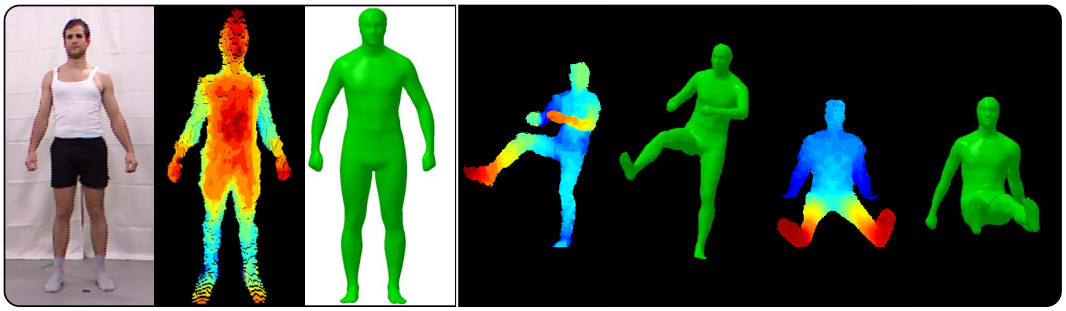

Figure 1. **(From left to right):** Actor standing in the front of a single Kinect camera. Color coded depth data (red is near, blue is far) as obtained from the Kinect. Automatically estimated body shape of the actor. Two complex poses reliably tracked with our algorithm (left: input depth, right: estimated pose).

estimation method in Sect. III. Then, in Sect. IV, we describe our personalized tracker and evaluate it with respect to previous approaches. Finally, we conclude in Sect. V with a discussion of future work.

II. Related Work

Shape Estimation.: Parametric shape models, as for example described in [4], [5], provide an easy way to represent the complex shape of the human body with only a small set of parameters. The estimation of their parameters from various kinds of input data constitutes a challenging problem. Several approaches, such as [6], [7], solve this task by estimating shape parameters from images of a person. In contrast, [2] show a first approach to obtain such shape parameters from a small set of depth images. However, the runtime of such approaches is still very long and problematic for home consumer scenarios. Our approach closes this gap and enables fast and accurate fitting of a body model to depth data.

Real-time Pose Estimation.: Marker-less pose estimation from multi-view video has been a long-standing problem in computer vision, and nowadays mature solutions exist, see [8] for an overview. Usually, these approaches do not run in real-time and require studio environments with complex multi-camera setups. Real-time skeletal pose estimation has come into reach by making use of depth sensors like time-of-flight (ToF) cameras [1] or the Microsoft Kinect. By using kinematic body models with simple shape primitives, the pose of an actor can be found by fitting the model to depth data or a combination of depth and image features [9], [10]. Body part detectors and a mapping to a kinematic skeleton are used in [11] to track full-body poses at interactive frame rates. Recently, data-driven methods to perform 3D human pose tracking based on a single depth image stream have become an important research topic [3], [12], [13], [14]. Another approach was proposed by [15] where regression forests were used to obtain model-to-depth data correspondences which are used for single frame optimization of body

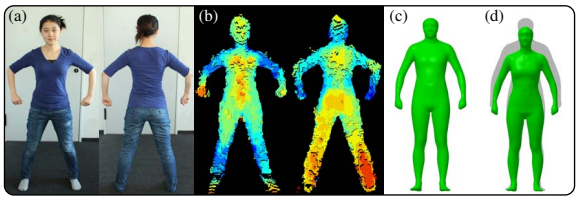

Figure 2. Shape estimation. **(a)**: Calibration poses. **(b)**: Depth input of poses. **(c)**: Initial shape. **(d)**: Estimated shape.

pose and rough estimation of the overall body size. We improve over the work by Baak *et al.* enabling a combined generative and discriminative tracker with a personalized shape model.

III. Personalized Body Shape Estimation

Many tracking procedures based on depth camera input rely on the availability of an accurate model of the actor's 3D body shape, which is often obtained by manual modeling or by direct measurement using specialized body scanners [3]. Such a method for generating a shape model is impractical for home applications, where the user has to deal with the following conditions and requirements. Firstly, in most cases there will not be any other sensor available for the shape estimation than for the actual tracking. Secondly, the whole shape estimation procedure should run in a reasonable amount of time, which means in the order of one minute rather than one hour. Thirdly, the procedure should impose as little additional constraints and efforts on the user as possible. In particular, manual interventions by the user, *e. g.*, to adjust parameters, should be minimized. Finally, the body shape estimation should be robust to possible inaccuracies such as imprecise poses assumed by the user during the calibration stage.

A first approach that tries to meet these requirements while estimating the body shape from depth camera input is described in [2]. Based on example shapes obtained from laser scans, the authors construct a model that parametrizes pose and shape of the human body. This model is then fit

to four depth images, captured from four different views that are 90 degrees apart. After capturing the depth images, the shape estimation takes about one hour.

In this section, we introduce a novel procedure for estimating the body shape from a single depth camera using only two different calibration poses and within only a minute of fitting time, see Fig. 2 for an overview. In addition, even if the user only roughly matches the required calibration poses, our shape estimation algorithm achieves accurate results. We propose two innovations to achieve high speed and high accuracy. Firstly, our optimization scheme works purely in the 3D domain and does not revert to 2D data representations as silhouettes or contours as used in [2]. However, note that the richer 3D contour is implicitly represented in the 3D-domain. Using 3D cues instead of 2D cues typically results in fewer ambiguities and occlusion problems such as an arm in front of the observed body, which would be invisible in the observed contour. Secondly, in our optimization scheme we use a local cost function that is not only based on distances of corresponding points, but also considers normal-based distances between points and planes. As a result, the optimization is less likely to get stuck in local minima and the speed of convergence is increased significantly.

A.Shape Model

Mathematically, our shape model is given as a mesh consisting of vertices and triangular faces. Let P be the number of vertices and, as explained below, let $\boldsymbol{\varphi}$ be a vector of shape parameters. Henceforth, we assume that the mesh is rigged with a kinematic skeleton which is driven by a pose parameter vector $\boldsymbol{\chi}$ using linear blend skinning. Hence, the 3D coordinates of the mesh depend on both $\boldsymbol{\varphi}$ and $\boldsymbol{\chi}$ and can be represented as the stacked vector $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}} \in \mathbb{R}^{3\cdot P}$. Furthermore, let $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)$ denote the 3D coordinate of the p^{th} vertex, $p \in [1:P] := \{1, 2, \ldots, P\}$. Finally, from the triangulation one can derive a normal vector $\mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p) \in \mathbb{R}^3$ for each vertex.

Our body model is a statistical model of human pose and body shape similar to [16]. The statistical model is a simplified SCAPE model [4], where we omit the terms responsible for modeling muscle bulging in order to speed up computations. Our model is generated from scans of 127 young male and female persons [5]. This certainly limits the expressiveness of the model to a certain extent. However, as our experiments will show, even with a model generated from a relatively small number of scans we achieve better accuracy than [2] where 2 500 scans were used. By suitably averaging the available scans, a base shape consisting of $P = 6449$ vertices was generated. Fig. 2 (c) shows the base shape in the standard pose given by the parameter $\boldsymbol{\chi}_0$. Let $\mathcal{M}_{0,\boldsymbol{\chi}_0}$ be the vertex coordinates of the base shape in the standard pose. Furthermore, let $\boldsymbol{\varphi}$ be a shape parameter vector that linearly influences the size and shape of the mesh. More precisely, from a shape database, a suitable eigenvector matrix $\Phi \in \mathbb{R}^{3\cdot P\times|\boldsymbol{\varphi}|}$ is determined, encoding the statistical shape variations. The details to compute the eigenvector matrix can be found in the supplemental material. This yields a family of different body shapes in the following way:

$$\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0} = \mathcal{M}_{0,\boldsymbol{\chi}_0} + \Phi \cdot \boldsymbol{\varphi} \tag{1}$$

In [5] it was shown that by using dimensionality reduction techniques, one obtains already a wide range of naturally looking shapes of different people for a low-dimensional $\boldsymbol{\varphi}$. In our experiments, we use the 13 most significant Eigenvectors.

As for the underlying skeleton, we use a model containing 51 joints similar to [17]. Not all joints possess a full degree of freedom (DoF). For example, the spine is represented by several coupled joints that are parameterized by only 3 DoFs, which results in a smooth bending of the whole spine. In our experiments, we represent the pose of a person with 31 DoFs (3 translational and 28 rotational) encoded by the pose parameter vector $\boldsymbol{\chi}$. The skeleton was once manually fitted to the base shape corresponding to the parameter vector $\boldsymbol{\varphi} = 0$ in the pose $\boldsymbol{\chi}_0$. To be able to transfer the skeleton to other shapes, we represent the position of each joint as a linear combination of its surrounding vertices.

B.Optimization

Our shape estimation problem can be formalized as follows. First, we assume a target point cloud is given T consisting of points $T(q) \in \mathbb{R}^3$ for $q \in [1:Q]$, where Q denotes the number of points. In our setting we assume that T is a depth image as supplied by a Kinect camera, but point clouds from other sources could also be used. The goal is to jointly optimize the shape and pose parameters of our shape model to best explain the given target point cloud.

Firstly, the shape and pose parameter vectors are initialized by $\boldsymbol{\varphi} = \boldsymbol{\varphi}_{\text{init}}$ and $\boldsymbol{\chi} = \boldsymbol{\chi}_{\text{init}}$. In our scenarios, we set $\boldsymbol{\varphi}_{\text{init}} = 0$ and $\boldsymbol{\chi}_{\text{init}}$ to the standard pose parameter $\boldsymbol{\chi}_0$ translated to the mean center of the point cloud T. In order to make the shape model compatible with the target point cloud T, we transform the shape model surface into a mesh point cloud. To this end, we basically consider the 3D coordinates $M(p) := \mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)$, $[1:P]$, of the mesh vertices. Since in our setting the target point cloud T comes from a depth image and hence only shows one side of the actor, we also restrict the mesh point cloud to the points that are visible from the depth camera's perspective (the rough orientation of the body is assumed to be known in the calibration phase). To simplify notation, we still index the restricted point cloud by the set $[1:P]$.

We establish correspondences between the target point cloud and the mesh point cloud based on closest points. For each point $M(p)$, we define the corresponding point $T(q_p)$ to be the point that minimizes the Euclidean distance between $M(p)$ and the point cloud T. Similarly, for each point $T(q)$ the point $M(p_q)$ is defined.

Based on these correspondences, we now introduce our optimization scheme. It is well known from the literature that one obtains faster convergence rates in rigid shape registration based on iterative closest points (ICP) when using point-plane constraints instead of point-point constraints, see [18] and references therein. Furthermore, such constraints are more robust to noise from depth sensors leading to a more stable convergence. On the other hand, point-to-plane constraints are problematic when correspondences are far apart. Therefore, we design an energy functional that incorporates both point-point as well as point-plane constraints. First, for a pair $(p, q) \in [1 : P] \times [1 : Q]$ let

$$d_{\text{point}}(p, q) = ||M(p) - T(q)||_2 \tag{2}$$

denote the Euclidean distance between the points $M(p)$ and $T(q)$. Next, we use the normal information supplied by the mesh to define a point-plane constraint. Let $N(p) = \mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)$, $p \in [1 : P]$, be the normal vector at the p^{th} vertex. Then, the distance between the point $T(q)$ and the plane defined by the normal $N(p)$ that is anchored at the point $M(p)$ is given by

$$d_{\text{normal}}(p, q) = \langle M(p) - T(q), N(p) \rangle. \tag{3}$$

Next, we fix a suitable threshold τ (in our experiments $\tau = 15\,\text{mm}$) to decide which of the distances should be considered depending on how far the two corresponding points are apart and we define

$$d_\tau(p, q) := \begin{cases} d_{\text{point}}(p, q), & \text{if } \|M(p) - T(q)\|_2 > \tau, \\ d_{\text{normal}}(p, q), & \text{otherwise.} \end{cases} \tag{4}$$

Finally, in the definition of the energy functional $E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T)$ we consider all correspondences from the mesh point cloud to the target point cloud and vice versa:

$$E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T) := \sum_{p \in [1:P]} d_\tau(p, q_p) + \sum_{q \in [1:Q]} d_\tau(p_q, q). \tag{5}$$

To minimize Eq. (5), we use a conditioned gradient descent solver as described in [17]. To this end, we compute the analytic partial derivatives of $E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T)$ with respect to the shape parameters $\boldsymbol{\varphi}$ and the pose parameters $\boldsymbol{\chi}$ and solve until convergence. Note that in contrast to numeric differentiation, analytic derivatives enable faster and more stable convergence. The analytic derivatives can be found in the supplemental material of this paper. We repeat the process in an ICP fashion, where between two iterations, the correspondences are updated using the newly estimated parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\chi}$. We further speed up the overall optimization procedure by using a multi-scale approach, where we start with only a small number of correspondences and successively increase the number of correspondences until we use one correspondence for every point in T and for every vertex in M.

Finally, we want to note that our optimization procedure can be easily extended to consider several target point clouds to be jointly optimized against. More precisely, having K target point clouds $T_1, \ldots, T_K$, the objective is to estimate K pose parameter vectors $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_K$, but one joint shape parameter vector $\boldsymbol{\varphi}$. In the optimization, the energy functional is defined as the sum $\sum_{k \in [1:K]} E(\boldsymbol{\varphi}, \boldsymbol{\chi}_k|T_k)$, see Eq. (5). Our experiments show that using only $K = 2$ different depth images (one from the front of the body and one from the back) are already sufficient to obtain an accurate shape estimate, see Fig. 2.

C.Evaluation

To evaluate the accuracy of our proposed method and to compare it with previous results, we conducted similar experiments as reported in [2]. As for the test data, we considered the body shapes of six different persons of different size and gender (three males, three females), see also Fig. 3. For each person, we recorded two depth images, one showing the front and the other the back of the body, see Fig. 2. Furthermore, using a full-body laser scanner, we generated for each person a surface point cloud with a resolution of about 350 000 vertices. These scans serve as ground-truth (GT).

Now, let $\boldsymbol{\varphi}^*$ be the optimized shape parameter vector obtained by our algorithm when using the two depth images as target point clouds (the pose parameter vectors $\boldsymbol{\chi}_1$ and $\boldsymbol{\chi}_2$ are not used in the evaluation). Furthermore, to obtain a ground-truth shape, we use the same algorithm as before, however, this time using the laser scanner point cloud as target. Let $\boldsymbol{\varphi}^{\text{GT}}$ denote the resulting optimized shape parameter vector. To compare the shapes resulting from $\boldsymbol{\varphi}^*$ and $\boldsymbol{\varphi}^{\text{GT}}$, one needs to generate the corresponding meshes. However, to this end, one also requires pose parameters, and simply taking the standard pose parameter vector $\boldsymbol{\chi}_0$ is usually not the right choice, since the different shape parameters may also have a substantial influence on the assumed pose. Therefore, we compensate for this effect by taking the standard pose for the laser scan shape and by suitably adjusting the pose parameters for the estimated shape. To this end, we again apply our optimization algorithm using $\mathcal{M}_{\boldsymbol{\varphi}^{\text{GT}},\boldsymbol{\chi}_0}$ as target point cloud and only optimize over the pose parameter vector $\boldsymbol{\chi}$ leaving $\boldsymbol{\varphi} = \boldsymbol{\varphi}^*$ fixed. Let $\boldsymbol{\chi}^*$ denote the result. As for the final evaluation, we then compare the mesh $\mathcal{M}_{\boldsymbol{\varphi}^*,\boldsymbol{\chi}^*}$ (representing our shape estimation result) with $\mathcal{M}_{\boldsymbol{\varphi}^{\text{GT}},\boldsymbol{\chi}_0}$ (representing the ground truth shape). Since

	M_1	M_2	M_3	F_1	F_2	F_3	$\varnothing$
μ	5.1	18.7	9.1	6.8	11.4	9.2	10.1
σ	2.5	9.5	4.0	3.7	4.9	4.4	4.8
max	14.1	46.3	20.5	18.7	30.1	19.4	24.9

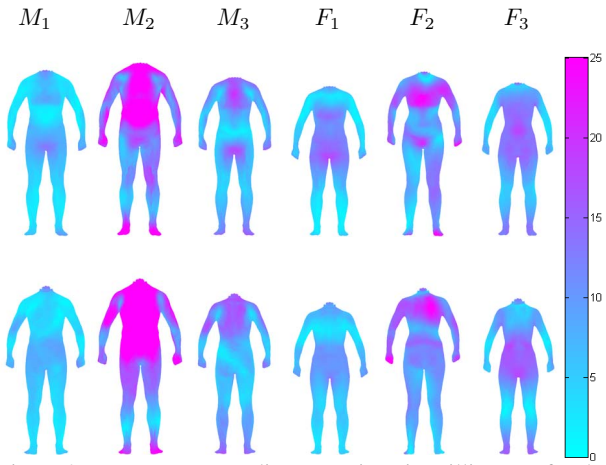

Figure 3. Vertex-to-vertex distances given in millimeters for three male (M_1–M_3) and three female (F_1–F_3) subjects shown from the front (**middle**) and from the back (**bottom**). The table (**top**) shows the mean μ, standard deviation σ, and maximum max over all vertices. The heads where removed from the error calculation because of their bad representation in the shape model.

vertex correspondences of these two meshes are trivial (based on the same index set $[1 : P]$), one can directly compute the vertex-to-vertex Euclidean distances in the same way as Weiss *et al.* [2].

The vertex-to-vertex distances are indicated in Fig. 3, which also shows the mean, variance and maximum over these distances. For example, for the first male actor M_1, the mean average is $5.1\,\mathrm{mm}$ and the maximal distance is $14.1\,\mathrm{mm}$. Overall, the achieved accuracies (in average $10.1\,\mathrm{mm}$) are good and comparable to (in average $10.17\,\mathrm{mm}$) reported in Weiss *et al.* [2]. There are various reasons for inaccuracies. In particular, using only 13 of the most significant Eigenvectors in Eq. (1) does not allow us to capture all shape nuances which may lead to higher errors, such as for the actors M_2 and F_2. In these cases, either similar shapes might be not spanned by the training data of the shape model or the 13-dimensional approximation of shape variations might be too coarse. Furthermore, note that the depth image resolution (which is roughly $20\,\mathrm{mm}$ at the used distance of $2.6\,\mathrm{m}$) as well as the mesh resolution (where neighboring vertices often have a distance of $20\,\mathrm{mm}$) puts limits on the achievable accuracy. Nonetheless, overall good accuracy is achieved with a compact model.

Besides its accuracy, our approach has two further main benefits: efficiency and robustness. It only requires 50–60 seconds to estimate the shape parameter vector (and the two pose parameter vectors) from two target depth point clouds. This is substantially faster than the 3 900 seconds (65 minutes) reported by Weiss *et al.* [2]. The running times were measured using a C++ implementation of our algorithm executed on an Intel Xeon CPU @ 3.10 GHz. Furthermore, jointly optimizing for shape and pose introduces a high degree of robustness and allows us to use only two different depth images to obtain accurate shape estimates. Actually, an additional experiment, where we used four target point clouds (using two additional depth images) only slightly improved the overall accuracies (from $10.1\,\mathrm{mm}$ when using two poses to $8.4\,\mathrm{mm}$ when using four poses). Besides implementation issues, these substantial improvements in running time and robustness are the result of using a relatively small number of optimization parameters, reverting to reliable 3D correspondences, using a more effective parametrization of the body model, and combining point and plane constraints.

IV. Personalized Depth Tracker

The tracker of Baak *et al.* [3] combines a generative with a discriminative approach. The discriminative tracker finds closest poses in a database, but that database is specific to an actor of a certain body shape. To perform best, that database would need to be regenerated for each body shape, which they do not do. They merely use a crude heuristic and scale the depth point cloud along fixed axes to match the database model. Therefore, in this paper, we suggest a different strategy by recomputing the entire set of poses in the database using the estimated personalized mesh. The database needs to be computed only once for each actor, which takes around 12 minutes for 50 000 poses using unoptimized code. An efficient GPU implementation would yield further speedups.

The resulting personalized depth tracker captures even fast and complex body poses (jumping jack, sitting down) reliably and in real-time, see Fig. 1 and also the accompanying video for some qualitative results. In the following, we will give some quantitative results with comparison to other approaches.

A. Evaluation on the Standford Dataset

In a first experiment, we compare our personalized tracker to previous approaches based on the dataset and error metrics described in [12]. The results of this evaluation are depicted in Fig. 4. One can see that our tracker gives comparable results to the previous approaches presented by Ganapathi *et al.* [12] and Baak *et al.* [3] and excedes the results of the previous approaches in many cases. Please note that for this evaluation marker positions of markers attached to the actor's body are predicted and compared to ground truth marker positions obtained with an optical marker based mocap system. We think that this way of

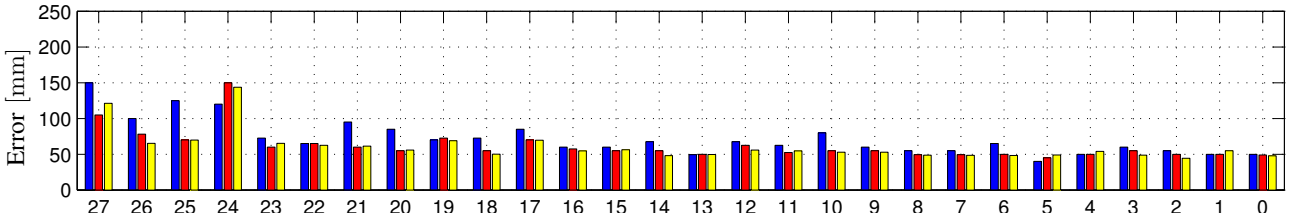

Figure 4. Average tracking error of sequences 0 to 27 of the dataset provided by [12]. The sequences were tracked using the tracker proposed by Ganapathi *et al.* [12] **(blue)**, Baak *et al.* **(red)**, and our tracker **(yellow)**.

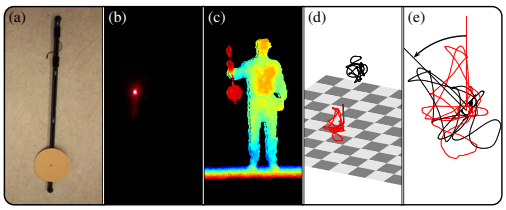

Figure 5. **(a):** Modified calibration wand with a cardboard disc around one marker. **(b):** Illuminated marker shown in an image from the RGB-camera of the Kinect. **(c):** Cardboard disk is clearly visible in the Kinect's depth image. **(d):** Reconstructed marker trajectories from Kinect **(red)** and optical mocap system **(black)**. **(e):** Estimation of the rotational offset between both trajectories after centering at their mean.

evaluating the tracking accuracy is not well suited for the specific requirements in home consumer scenarios. For example, in some reconstruction scenarios one is only interested in reconstructing the joint positions of the user, as it is done for example in many Kinect applications. On the other hand, when it comes to augmented reality scenarios, such as virtual try-on applications, one is rather interested in tightly approximating the depth image of the user to get a well fitting overlay of simulated objects such as cloths. In order to address these two evaluation aspects, we recorded a dataset with ground truth tracking results.

B.Our Evaluation Dataset

For our evaluation, we recorded a dataset [19] using both a Microsoft Kinect as well as a Phasespace active marker-based mocap system simultaneously. It comprises various kinds of motion performed by five actors (three male: M_1, M_2, and M_3 and two female: F_1 and F_2). The body models for each actor were estimated with the method from Sect. III. We defined four groups of motions of different difficulties D. They range from easy to track motion sequences (D_1), simple arm and leg motions (D_2), fast movements such a kicking and jumping (D_3), to very hard to track motions such as sitting down, walking in circles, or rotating in place (D_4). In total we recorded a set of 40 sequences, 2 takes from every of the 4 difficulties performed by each of the 5 actors. We used half of the recorded motions to build the pose database of the tracker, the other half is used for evaluation and is referred to as *evaluation dataset*. We use the notation *<actor><difficulty>* to refer to a specific sequence from the evaluation dataset, *e. g.* M_2D_4 refers to the sequence of difficulty D_4 performed by actor M_2.

Calibration.: In order to make the tracking results from the depth trackers comparable to the ground truth data we need to calibrate the Kinect with respect to the marker-based system. Since the location of the Kinect camera is unknown a priori and the frame capturing of the Kinect cannot be externally synchronized, such a calibration consists of two parts, a temporal calibration and a spatial calibration. While the spatial calibration only needs to be done once, the temporal calibration must be done for every captured sequence. We perform the temporal calibration by calculating a space invariant but time varying feature for two corresponding trajectories from both the marker-based and the Kinect recording. The temporal offset is then determined by identifying the lag that maximizes the cross correlation of both features. In our case, it turned out that the absolute velocities of the trajectories are a robust feature for temporal calibration even under the presence of tracking errors. A suitable trajectory could, for instance, be the position of a joint or another well defined point over a period of time.

For spatial calibration of both the Kinect and the marker-based system, we use a calibration wand with a single active LED (see Fig. 5 (a)). Here, the idea is to determine the trajectory of the marker using both recording devices, and to register the trajectories to each other. While the marker-based system provides the marker's trajectory in a straight forward way, we need some additional processing to obtain the trajectory from the Kinect. The Kinect records depth and video simultaneously, see Fig. 5 (b) & (c), and both streams are calibrated relative to each other. We can thus get the LED trajectory from the Kinect by recording in a dark room, thresholding the intensity image to identify the pixel position of the LED, and extracting corresponding depth information from the depth channel. Using the intrinsic parameters of the Kinect, we calculate the 3D position of the marker from the 2D position and depth value. Fig. 5 (d) shows a reconstructed marker trajectory (red) from Kinect footage. Now, we temporally align the trajectories with the method described

above. The resulting trajectories are then aligned spatially by determining a rigid transform for point correspondences (Fig. 5 (e)).

Joint Tracking Error.: In a first experiment, we want to evaluate how accurate the various depth-based trackers capture the joint positions of an actor. To this end, we used the marker data from the phase space system to animate a kinematic skeleton using inverse kinematics. We consider the resulting joints positions as ground truth data for the evaluation. In the following we assume that the sequences of the trackers and the ground-truth data have been temporally and spatially aligned using the procedure described above.

Since all trackers use a slightly different set of joints, we select for each tracker a subset of 20 joints that are close to semantic positions in the body such as the lower back, the middle of the back, the upper back, the head, the shoulders, the elbows, the wrists, the hands, the hips, the knees, the ankles, and the feet. We now measure for every frame the distance between the tracked joints and the ground truth joints. Since the corresponding joints from the different trackers do not lie at the exact same positions we need to normalize for an offset. Therefore, we calculate the average local displacement of the joint relative to the corresponding ground-truth joint, and subtract this offset from the position of the tracked joint. Here, local displacement means that we consider the displacement within the local coordinate frame of the ground truth joint.

The average errors—over all joints and frames of one sequence—for the various actors and sequences are shown in Fig. 6. One can see that the tracker of the Kinect SDK performs worst with an average error of 95.8 millimeters over all sequences. The tracker presented by Baak *et al.* [3] shows an average error of 82.6 millimeters over all sequences, while our tracker performs best with an error of 73.8 millimeters.

Surface Tracking Error.: In a second experiment, we assess the quality of the tracker by quantifying how well the tracked mesh at each frame approximates the point cloud recorded by the Kinect, referred to as *surface tracking error*. To this end, we first calculate a so-called *distance map* for every frame of a tracked sequence, by determining for every foreground point in the depth image of the Kinect the distance to the closest point on the mesh. Now, the straightforward way to compute a suitable surface tracking error would be to take the maximum distance from each distance map. Unfortunately, it turns out that the maximum is very unstable due to noise in the depth image and inaccuracies of the background subtraction. Here, a quantile value is better suited since it filters out influences of noise. We tested several quantiles and it turned out that a 97%-quantile is a good compromise between robustness to outliers and responsiveness to tracking errors. Please note that since the Kinect SDK does not provide a tracked mesh, we cannot calculate this error for the tracker of the Kinect SDK.

Table I. Averaged surface tracking errors in millimeters for each sequence of the evaluation dataset that were tracked by Baak *et al.*, and our tracker.

M_1	D_1	D_2	D_3	D_4	∅
Baak *et al.*	66	84	139	138	106
Ours	61	81	116	102	90
M_2	D_1	D_2	D_3	D_4	∅
Baak *et al.*	54	84	71	153	91
Ours	56	77	75	110	80
M_3	D_1	D_2	D_3	D_4	∅
Baak *et al.*	59	88	104	108	90
Ours	56	76	89	93	79
F_1	D_1	D_2	D_3	D_4	∅
Baak *et al.*	74	102	172	129	119
Ours	64	84	115	97	90
F_2	D_1	D_2	D_3	D_4	∅
Baak *et al.*	49	66	82	117	79
Ours	46	62	80	105	73

Fig. 7 (top) shows the surface tracking error over sequence F_1D_1. The red curve represents the error of the tracker by Baak *et al.* [3] while the yellow curve is the result of our personalized tracker. The black vertical line at 22.7 seconds indicates a point in time where the surface tracking error of Baak *et al.* is significantly higher than that of our tracker. Fig. 7 (b)–(f) shows that this corresponds to a notable tracking error. In the middle, Fig. 7 (b) displays the depth image recorded by the Kinect. In the distance map, cyan colors depict small distances around 0 millimeters while magenta colors represent high distance values of 25 millimeters and up. On the right, Fig. 7 (c) & (d) shows the distance map (left) and the tracked mesh of their tracker, Fig. 7 (e) & (f) depicts the distance map and the tracked mesh of our tracker. Our tracker tracks the right arm of actor F_1 correctly while it was merged with the upper body by the tracker of Baak *et al.*.

Table I lists the average surface tracking errors of the different sequences, actors and trackers. Our tracker performs significantly better than the tracker of Baak *et al.* [3]. Especially sequence M_2D_4—which is one of the hardest sequences—is tracked considerably better by our tracker (average error of 110 mm) than by the tracker by Baak *et al.* (average error of 153 mm) Of course our tracker also has limitations, *e. g.*, when the actor does not face the camera (as in sequences of difficulty D_4) or when parts of the body are occluded or outside of the recording volume of the Kinect—which occasionally happens during all sequences.

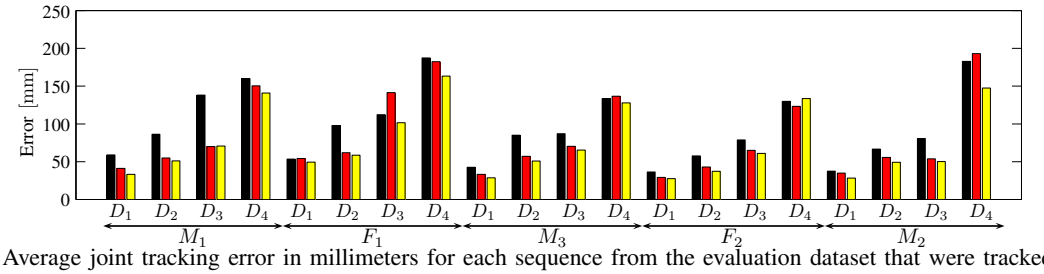

Figure 6. Average joint tracking error in millimeters for each sequence from the evaluation dataset that were tracked by the tracker of the Kinect SDK **(black)**, Baak *et al.* **(red)**, and our tracker **(yellow)**.

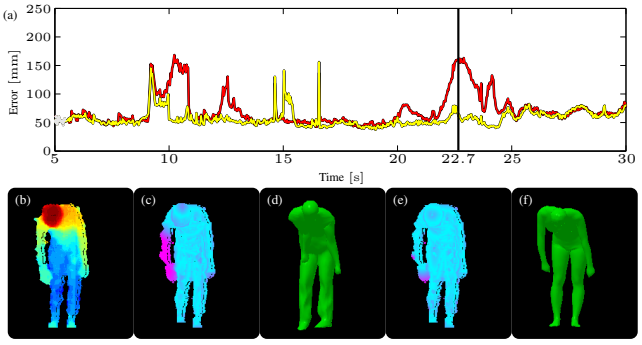

Figure 7. **(a):** Surface tracking error in millimeters for sequence F_1D_1 tracked by of Baak *et al.* **(red)** and our tracker **(yellow)**. **(b)-(f):** Status at 22.7 seconds. **(b):** Depth image at (red front, blue back). **(c):** Distance map of tracker of Baak *et al.*. **(d):** Tracked mesh for tracker of Baak *et al.*. **(e):** Distance map for our tracker. **(f):** Tracked mesh for our tracker.

V. Conclusion and Future Work

In this paper, we presented a personalized real-time tracker of human body poses from single depth images that is more accurate than related approaches from the literature. Key to its success is personalization. We developed a new approach to estimate the personalized shape of an actor based on a parametric body model, which is much faster and more accurate than previous methods. We also presented a new real-time pose tracker that exploits this model and automatically adjusts to every actor. In conjunction, these two contributions allow us to track both skeletal joint locations as well as the shape of the body more accurately than with previous methods. We confirm this through extensive evaluations against ground truth on a comprehensive test dataset which is publicly available.

References

[1] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-flight sensors in computer graphics," *CGF*, vol. 29, no. 1, pp. 141–159, 2010.

[2] A. Weiss, D. Hirshberg, and M. Black, "Home 3D body scans from noisy image and range data," in *ICCV*, Nov. 2011.

[3] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *ICCV*, Nov. 2011.

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM TOG*, vol. 24, no. 3, pp. 408–416, Jul. 2005.

[5] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel, "A statistical model of human pose and body shape," *CGF*, vol. 28, no. 2, 2009.

[6] P. Guan, A. Weiss, A. Bălan, and M. J. Black, "Estimating human shape and pose from a single image," in *ICCV*, 2009, pp. 1381–1388.

[7] N. Hasler, C. Stoll, B. Rosenhahn, T. Thormhlen, and H.-P. Seidel, "Estimating body shape of dressed humans," *Computers & Graphics*, vol. 33, no. 3, pp. 211–216, 2009.

[8] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010.

[9] A. Bleiweiss, E. Kutliroff, and G. Eilat, "Markerless motion capture using a single depth sensor," in *SIGGRAPH ASIA Sketches*, 2009.

[10] S. Knoop, S. Vacek, and R. Dillmann, "Fusion of 2D and 3D sensor data for articulated body tracking," *Robotics and Autonomous Systems*, vol. 57, no. 3, pp. 321–329, 2009.

[11] Y. Zhu, B. Dariush, and K. Fujimura, "Kinematic self retargeting: A framework for human pose estimation," *CVIU*, vol. 114, no. 12, pp. 1362–1375, 2010.

[12] V. Ganapathi, C. Plagemann, S. Thrun, and D. Koller, "Real time motion capture using a single time-of-flight camera," in *CVPR*, 2010.

[13] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *CVPR*, Jun. 2011.

[14] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM TOG*, vol. 31, no. 6, Nov. 2012.

[15] J. Taylor, J. Shotton, T. Sharp, and A. W. Fitzgibbon, "The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation," in *CVPR*, 2012.

[16] A. Jain, T. Thormählen, H.-P. Seidel, and C. Theobalt, "Moviereshape: Tracking and reshaping of humans in videos," *ACM TOG*, vol. 29, no. 5, 2010.

[17] C. Stoll, N. Hasler, J. Gall, H.-P. Seidel, and C. Theobalt, "Fast articulated motion tracking using a sums of gaussians body model," in *ICCV*, 2011.

[18] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, pp. 145–155, Apr. 1992.

[19] T. Helten, A. Baak, G. Bharaj, M. Müller, H.-P. Seidel, C. Theobalt, "Personalized Depth Tracker Dataset", http://resources.mpi-inf.mpg.de/PersonalizedDepthTracker/