A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera

Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, Christian Theobalt
Saarland University & MPI Informatik, Saarbrücken, Germany
{abaak,meinard,gbharaj,hpseidel,theobalt}@mpi-inf.mpg.de

Abstract

In recent years, depth cameras have become a widely available sensor type that captures depth images at real-time frame rates. Even though recent approaches have shown that 3D pose estimation from monocular 2.5D depth images has become feasible, there are still challenging problems due to strong noise in the depth data and self-occlusions in the motions being captured. In this paper, we present an efficient and robust pose estimation framework for tracking full-body motions from a single depth image stream. Following a data-driven hybrid strategy that combines local optimization with global retrieval techniques, we contribute several technical improvements that lead to speed-ups of an order of magnitude compared to previous approaches. In particular, we introduce a variant of Dijkstra's algorithm to efficiently extract pose features from the depth data and describe a novel late-fusion scheme based on an efficiently computable sparse Hausdorff distance to combine local and global pose estimates. Our experiments show that the combination of these techniques facilitates real-time tracking with stable results even for fast and complex motions, making it applicable to a wide range of interactive scenarios.

1. Introduction

In recent years, several approaches for marker-less human pose estimation from multiple video streams have been presented [2, 4, 7, 9]. While multi-view tracking already requires solving challenging non-linear optimization problems, monocular pose estimation puts current technology to its limits since, with intensity images alone, the problem is considerably underconstrained, see *e.g.* [16, 21]. Here, non-trivial inference or optimization steps are needed in combination with strong priors in order to have a chance to reconstruct human movements. In general, real-time reconstruction of complex human motions from monocular intensity image sequences can still be considered an open problem. New depth sensors such as time-of-flight (ToF) cameras capture 2.5D scene geometry [14] at video frame rates. Therefore, they promise to deliver data that permits more reliable 3D pose reconstruction from a single viewpoint. Some previous papers have already shown that based on ToF data, reconstruction of even complex poses comes into range at interactive frame rates [3, 8, 10, 13, 19, 30]. In this paper, we contribute to this area of research by presenting a tracking framework that not only yields more robust pose estimates from monocular depth image sequences but also enables significant speed-ups of an order of magnitude compared to previous approaches.

Our procedure follows a hybrid strategy combining generative and discriminative methods, which is an established paradigm for pose estimation and tracking problems. While local optimization strategies have proven to yield high frame rates, see *e.g.* [13], fast motions and noisy data lead to tracking errors that are hard to recover from. Algorithms using global optimization could prevent this, but are typically slow and prohibitive for real-time scenarios. Various data-driven approaches have also been suggested to overcome some of these issues, enabling fast yet robust tracking from intensity image streams, see [18, 22, 26, 29]. These approaches, however, fail on poses that are not contained in the database and thus rely on a dense sampling of the pose space to reconstruct. Hybrid strategies that combine generative and discriminative methods have proven to be a suitable methodology for pose estimation and tracking procedures, see *e.g.* [1, 6, 10, 23, 25, 28]. In these works, the main idea is to stabilize generative optimization algorithms by a discriminative component which is implemented as a database lookup or a classification scheme. Using this strategy, the risk of getting stuck in local minima is significantly reduced, while time-demanding global optimization methods are avoided.

In our approach, we employ a data-driven hybrid strategy conceptually similar to [6], where local optimization is combined with global retrieval techniques, see Fig. 1 for an overview. In our scenario, an actor may perform even complex and fast motions in a natural environment facing a

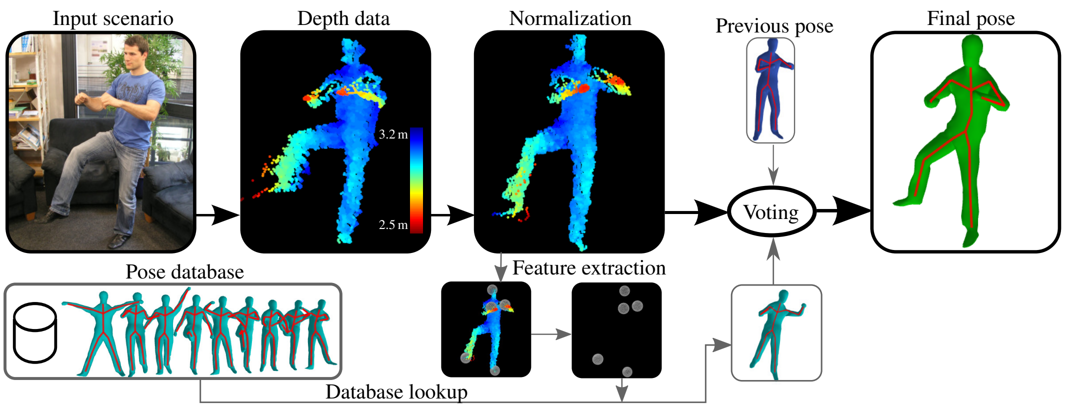

Figure 1. Overview of our proposed pose estimation framework.

single depth camera at a reasonable distance. Similar to [6], we retrieve pose hypotheses from a large database of 3D poses using sparse features extracted from the input data. Additionally, a further hypothesis is generated based on the previously tracked frame. After a local optimization of both hypotheses, a late-fusion voting approach combines the hypotheses to yield the final pose. While the overall procedure follows previous work [6, 10], we introduce a number of novel techniques to add robustness and significantly speed up computations at various stages including efficient feature computation, efficient database lookup, and efficient hypothesis voting. In our experiments, we also compare with previous work using the publicly available benchmark data set [10]. We gain significant improvements in accuracy and robustness (even for noisy ToF data and fast motions) while achieving frame rates of up to 100 fps (opposed to 4 fps reported in [10]).

Contributions. In this paper, we present a system for full-body pose estimation from monocular depth images that requires only 10 to 16 ms per frame on a standard single-core desktop PC, while being able to track even fast and complex full-body motions. Following a data-driven hybrid strategy that combines local pose estimation with global retrieval techniques, we introduce several technical improvements. Firstly, in the feature extraction step, we introduce a variant of Dijkstra's algorithm that allows for efficiently computing a large number of geodesic extrema. Secondly, in the retrieval step, we employ an efficient database lookup scheme where semantic labels of the extrema are not required. Thirdly, we describe a novel late-fusion scheme based on an efficiently computable sparse Hausdorff distance. It is the combination of all these techniques that avoids computational bottlenecks while still providing robust tracking results.

The remainder of this paper is organized as follows. In Sect. 2, we describe the input data and preprocessing steps. We define the parts of the pose reconstruction framework in Sect. 3 and describe our extensive experiments in Sect. 4, before we conclude in Sect. 5. Further related work is discussed in the respective sections.

2. Acquisition and Data Preparation

Depth data. A ToF camera captures depth/distance data at video frame rates by measuring the round trip time of infrared light emitted into and reflected from the scene [14]. Opposed to stereo systems, current ToF cameras are robust to background illumination and yield stable distance values independent of the observed textures. Unfortunately, they have challenging data characteristics, exhibiting low resolution, strong random noise and a systematic bias [14]. In particular, dynamic scenes with fast motions lead to strong artifacts in the depth measurements such as so-called flying mixed pixels at occlusion boundaries.

The camera returns a distance image $I := \mathbb{Z}^2 \to \mathbb{R}$. with $\mathbb{Z}^2$ being the pixel domain. We transform the per-pixel distances into a metric 3D point cloud $\mathcal{M}_I \subseteq \mathbb{R}^3$ for every input frame. We then perform background subtraction using a static prerecorded background model and delete contour pixels to remove the influence of mixed pixels. Then, a 3×3 median filter is used to reduce noise in the measurements.

Model of the actor. The motion of the actor is modeled as a kinematic chain [17]. It contains a set of $J = 20$ joints that are connected by rigid bones. We define one distinguished joint as the root of the kinematic chain. A pose is fully determined by the configuration of a kinematic chain specified by a pose χ containing the position and orientation of the root joint as well as a set of joint angles. Through forward kinematics [17] using χ, joint positions $P_\chi \in \mathbb{R}^{3\times J}$ can be computed. Using linear blend skinning [15], we attach a surface mesh with a set of 1170 vertices $\mathcal{M}_\chi \subseteq \mathbb{R}^3$ to the kinematic chain to model the deforming body geometry. Initializing the body model to the shape of a specific actor is beyond the scope of this paper. Methods exist to solve this task using a large database of scanned humans, see *e.g.* [11, 12]. As shown in the experiments (Sect. 4), even with a fixed body model we can track people for a range of different body sizes.

Pose database. In the proposed algorithm, we use a pose database to overcome limitations of local optimization. To create the pose database, we record human motions using a commercial marker-based motion capture system. The actor performs a variety of motions typical for our scenario including hand gestures and foot motions, etc., to span a large range of different poses. The obtained poses χ_i are then normalized according to the positions of the root joint and the viewing direction to enable invariance under global transformations. To maximize the variety and minimize the number of poses in the database, we select a subset of the recorded poses using a greedy sampling algorithm [29]. To this end, the distance of two poses χ_1 and χ_2 is measured by the average Euclidean distance of the corresponding joint positions $d_P(\chi_1,\chi_2) := {}^1/_J \cdot \|P_{\chi_1} - P_{\chi_2}\|_2$. In contrast to [29], we truncate the sampling as soon as a the minimal distance between all pairs of selected poses reaches a certain threshold. Using the truncated sampling, we obtain roughly $25,000$ poses in which any two selected poses have a pose distance d_P larger than 1.8 cm.

For each selected pose, we then consider end effector positions of the left/right hand, the left/right foot, and the head, modeled as $E^5_\chi := (e^1_\chi, \ldots, e^5_\chi) \in (\mathcal{M}_\chi)^5$. The following three reasons motivate the usage of these features. Firstly, end effector positions can be efficiently estimated for a large set of different poses even from ToF data alone, see Sect 3.2. Secondly, for many poses these positions are characteristic, thus yielding a suitable representation for cutting down the search space. Thirdly, they lead to low-dimensional feature vectors which facilitates the usage of efficient indexing methods. In conclusion, end effector positions constitute a suitable mid-level representation for full-body poses that on the one hand abstract away most of the details of the noisy input data, yet on the other hand retain discriminative power needed in the pose estimation. For indexing, we use a *kd*-tree [5] on the 15-dimensional stacked vectors E^5_χ. Since skeleton size (*e.g.* body height or arm span) varies with different actors, the pose database has to be adapted to the actor. While not implemented in the presented system, this task can be solved using a retargeting framework. Even without retargeting, by manipulating $\mathcal{M}_I$ we are able to track motions of people if body proportions are not too far off the database skeleton, see Sect. 4.

Normalization. To cope with global rotations, one could augment the database to contain pose representations from several viewing directions [6, 26, 29]. In this case, the retrieval time and also the risk of obtaining false poses would increase. Instead, in our framework, we normalize the point cloud according to an estimated viewing direction. To this end, we compute a least-squares plane fit to the points corresponding to the torso, which we assume to lie within within a sphere of 0.15 m radius around the center of $\mathcal{M}_I$. The normal of the plane corresponds to the Eigenvector with the smallest Eigenvalue of the covariance matrix of the points. The viewing direction is its projection onto the horizontal plane. To cope with frames in which the direction cannot be estimated because, *e.g.*, the torso is occluded, we use exponential smoothing over time with an adaptive smoothing factor. Body parts that occlude the torso lead to a less planar plane fit (smallest Eigenvalue is relatively large) or a less circular fit (largest Eigenvalues are not similar). The smoothing factor minimizes the influence of normals corresponding to a less planar or circular fit. As a consequence, the estimation remains stable even if the arms occlude the torso or the center of $\mathcal{M}_I$ does not correspond to the torso.

3. Pose Reconstruction Framework

In the offline preprocessing phase of the framework, the camera matrix is obtained and the background model is created. Hereafter, our proposed online framework is described which can be divided into five main components, see also Fig. 1. As explained in the previous section, at a given frame t, the first steps are to compute the point cloud $\mathcal{M}_I$ from the distance image I, to perform background subtraction, to filter and to normalize according to the viewing direction. Let χ^*_{t-1} be the final pose estimate of the previous frame $t-1$. From χ^*_{t-1}, we obtain a pose hypothesis χ^{LocOpt}_t by refining χ^*_{t-1} with the input data using local optimization (Sect. 3.1). A second pose hypothesis is obtained as follows. We extract a 15-dimensional feature vector from $\mathcal{M}_I$, representing the 3D coordinates of the first five geodesic extrema (Sect. 3.2). Being a low-dimensional yet characteristic pose representation, the features allow for rapid retrieval of similar full-body poses from a large pose database (Sect. 3.3). From the set of retrieved poses we choose a single pose hypothesis χ^{DB}_t using a distance function that is regularized by χ^*_{t-1}. Based on a voting scheme that combines two sparse Hausdorff distances, our algorithm decides between χ^{DB}_t and χ^{LocOpt}_t to find the final pose χ^*_t, see Sect. 3.4.

3.1. Local Optimization

In our local pose optimization, we follow a standard procedure as described in *e.g.* [24]. Here, the goal is to modify an initial pose χ such that the modified pose χ' fits to the point cloud $\mathcal{M}_I$ more accurately. To this end, we seek correspondences between vertices in $\mathcal{M}_\chi$ and points in $\mathcal{M}_I$. Finding correspondences for all $v \in \mathcal{M}_\chi$ is not meaningful for three reasons. Firstly, many vertices do not have semantically meaningful correspondences in $\mathcal{M}_I$, *e.g.* the back vertices. Secondly, the number of correspondences for the torso would be much higher than the number of correspondences in the limbs, disregarding the importance of the limbs for pose estimation. Thirdly, the computation time

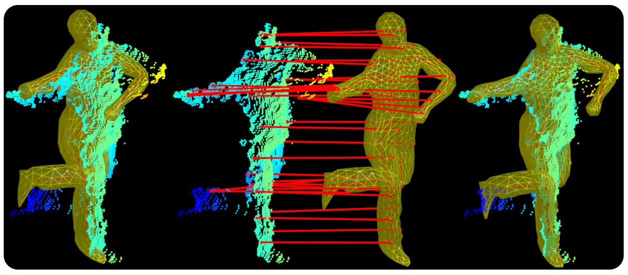
Figure 2. From pose χ (left), correspondences for mesh vertices in C_χ are estimated (middle). Local optimization using the correspondences yields an updated pose χ' (right).

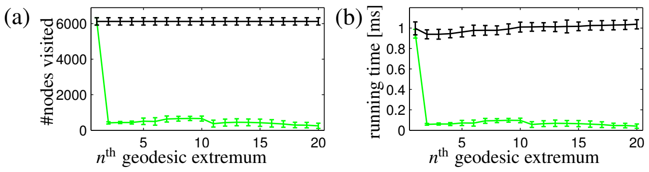

Figure 3. **(a)** Number of nodes visited and **(b)** running time in milliseconds to find the n^{th} geodesic extreme point for (black) the unoptimized and (green) our optimized algorithm. Average values and standard deviation bars for a sequence of 400 frames from the data set of [10] are reported.

of local optimization increases with the number of correspondences. To overcome these shortcomings, we use a predefined set $C_\chi \subseteq \mathcal{M}_\chi$ of mesh vertices (each body part should be assigned some correspondences) and find correspondences in $\mathcal{M}_I$ for all $v \in C_\chi$ (Fig. 2). Using these correspondences in an optimization framework similar to the one in [24], we obtain updated pose parameters χ'.

3.2. Feature Computation

To obtain a sparse yet expressive feature representation of the input point cloud $\mathcal{M}_I$, we revert to the concept of geodesic extrema as introduced in [20]. Such extrema often correspond to end effector positions, yielding characteristic features of many poses, see Fig. 4. Following [20], we now summarize how to obtains such features. Then, we introduce a novel variant of Dijkstra's algorithm that allows for efficiently computing a large number of geodesic extrema.

We model the tuple of the first n geodesic extreme points as $E_I^n := (e_I^1, \ldots, e_I^n) \in (\mathcal{M}_I)^n$. To compute E_I^n, the point cloud data is modeled as a weighted graph where each point in $\mathcal{M}_I$ represents a node in the graph. To efficiently build up the edge structure, we exploit the neighborhood structure in the pixel domain $\mathbb{Z}^2$ of the underlying distance image as follows. For all neighbors q in the 8-neighborhood of $p \in \mathcal{M}_I$, we add an edge between p and q of weight $w_{pq} := \|p - q\|_2$ if w_{pq} is less than a distance threshold τ and $q \in \mathcal{M}_I$. In our approach, in contrast to the method in [20], we need to ensure that the obtained graph does not separate into more than one connected component. In practice, however, the obtained graph is not fully connected due to sensor noise and occlusions (*e.g.* Fig. 4 (c) and (d)). Using an efficient union-find algorithm [27], we compute the connected components and discard all components that occupy a low number of nodes. The connected component with the largest number of nodes is assumed to be the torso. Finally, all remaining components are connected to the torso by adding an edge between the respective closest pair of pixels.

We now show how a large number of extrema can be computed efficiently. To find the first geodesic extreme point $e_I^1 \in \mathcal{M}_I$, we run Dijkstra's algorithm [5] with the centroid point $\bar{p} \in \mathcal{M}_I$ as source. The first pass of Dijkstra stores the shortest geodesic distances from the source to any other node in the graph in an array Δ of distances having $|\mathcal{M}_I|$ entries. After the first pass of Dijkstra, the node with the largest distance in Δ is taken as the first geodesic extreme point e_I^1. According to [20], the next step is to add a zero-cost edge between $\bar{p}$ and e_I^1 and then to restart Dijkstra's algorithm to find e_I^2, and so on. This leads to a running time of $O(n \cdot D)$ for n extrema with D being the running time of Dijkstra's algorithm for the full graph. Note, however, that the second run of Dijkstra's algorithm shows a high amount of redundancy: the entries in the array Δ corresponding to all nodes in the graph that are geodesically closer to $\bar{p}$ than to e_I^1 do not change. Therefore, to compute the 2nd pass, we keep the distance values of the 1st pass and set e_I^1 as the new source. The value in Δ corresponding to the new source is set to 0 and Dijkstra's algorithm is started. Then, we pick e_I^2 as the point with the maximal distance in the updated Δ. For the 3rd pass we set e_I^2 as the new source, set the value in Δ corresponding to the new source to 0, and run Dijkstra. This way, in the 3rd pass, only nodes in the graph that are nearer to e_I^2 than to all other previously used source nodes are touched, leading to drastic improvements in running time for each pass, see Fig. 3. We proceed iteratively to compute the subsequent extreme points.

Using this computational scheme, end effector positions are detected efficiently even in difficult scenarios where *e.g.* a foot is bent to the back or where a hand occludes parts of the body (Fig. 4 (b)), or the arms are outstretched to the camera (Fig. 4 (c)). In poses where the end effectors are very near to other parts of the body, the topology of the graph may change and the detected extrema may not correspond to the set of end effectors any longer (Fig. 4 (e) and (f), right knee and left elbow are selected as e_I^5, respectively). Note, however, that by means of following a combined generative and discriminative approach our framework can circumvent the influence of false detections.

3.3. Database Lookup

In this section, we show how to employ an efficient lookup scheme that does not need a priori semantic labels of the extracted geodesic extrema. The goal is to iden-

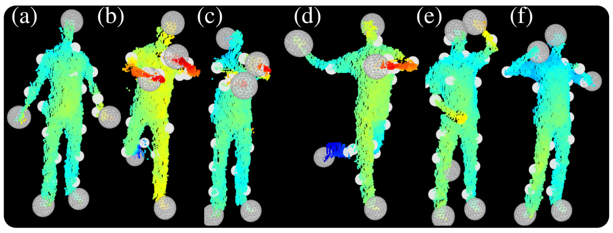

Figure 4. Example poses with the first five geodesic extrema drawn as big spheres, and extrema 6 to 20 drawn as smaller blobs.

tify a suitable full-body pose χ_t^{DB} in our pose database by searching the *kd*-tree of the database using the recovered geodesic extrema E_I^5 as the query input. However, opposed to the database motions where the semantics of the end effector positions are known, the semantic labels of the extrema on the query side are not known. To partially solve for missing semantics, the method of [10] uses a classifier trained on 'hand', 'head', and 'foot' patches of distance images. This process, however, is relatively expensive for a real-time scenario (60 ms per frame according to [20]). In our approach, we circumvent the classification problem by searching the database for a number of permutations of E_I^5. More precisely, let $\mathbb{S}_5$ be the symmetric group of all five-permutations and let $\mathcal{S} \subseteq \mathbb{S}_5$ be a subset containing permutations σ such that the positions in σE_I^5 are close to the end effectors of the previous frame χ_{t-1}^*, *i.e.*

$$\mathcal{S} := \{\sigma \in \mathbb{S}_5 | \forall n \in [1:5] : \|e_I^{\sigma(n)} - e_{\chi_{t-1}^*}^n\| < \mu\}. \tag{1}$$

We fix a conservative distance threshold $\mu = 0.5$ meters to effectively and conservatively prune the search space while still allowing for large jumps in the end effector positions which may be present in fast motions. To further introduce robustness to a possible false estimation in the previous frame, we augment $\mathcal{S}$ if we detect jumps in the positions of the geodesic extrema. The additional permutations are chosen by the weak assumption that the lowest extrema correspond to the feet. In frames with clear geodesic extrema, the number of considered permutations typically drops to one.

By querying the *kd*-tree of the pose database for K nearest neighbors for each permutation in $\mathcal{S}$, we obtain $K \cdot |\mathcal{S}|$ pose candidates $\chi_{k,\sigma}$ with $k \in [1:K], \sigma \in \mathcal{S}$, and associated distance values to the database defined through the average end effector distances of the corresponding stacked vectors

$$\delta(\chi_{k,\sigma}, E_I^5) := 1/5 \cdot \|E_{\chi_{k,\sigma}} - \sigma E_I^5\|_2. \tag{2}$$

The result of the database lookup χ_{k^*,σ^*} for frame t is then chosen by integrating temporal consistency using

$$(k^*, \sigma^*) = \underset{(k,\sigma)}{\mathrm{argmin}}\, \lambda \cdot \delta(\chi_{k,s}, E_I^5) + (1-\lambda) \cdot d_P(\chi_{k,s}, \chi_{t-1}^*) \tag{3}$$

with a weighting factor λ. We fix $\lambda = 0.5$ such that temporal continuity in the full-body poses is preserved, while still allowing to recover from possible false pose estimates of the previous frame. Finally, we refine $\chi_{k^*,s*}$ to the hypothesis χ_t^{DB} using local optimization (Sect. 3.1).

3.4. Hypothesis Voting

At this stage, two alternative pose hypotheses have been derived, *i.e.* χ_t^{LocOpt} from a generative component, and χ_t^{DB} from a discriminative approach. For the fusion of both, we propose a novel voting scheme based on an efficiently computable sparse Hausdorff distance. By fusing the hypotheses, the local optimization and database lookup schemes benefit from each other. On the one hand, local optimization can loose track, and the database lookup can reinitialize the tracking. On the other hand, the database lookup can fail, in particular if the end effectors are not revealed. Then, local optimization continues to track the motion. We use distances that revert to the original input point cloud $\mathcal{M}_I$ rather than to derived data in order to avoid a dominant influence of potential errors in the feature extraction or in the database lookup.

One possible distance measure could be defined by rendering $\mathcal{M}_\chi$ into a distance image and comparing it to I. In practice, however, because of the relatively low number of pixels in the limbs, such a distance measure is dominated by the torso. For this reason, we propose a novel distance metric that can be computed efficiently and accounts for the importance of the limbs for pose estimation. To this end, we combine two sparse Hausdorff distances. The first distance expresses how well the mesh is explained by the input data:

$$d_{\mathcal{M}_\chi \to \mathcal{M}_I} := \frac{1}{|C_\chi|} \sum_{v \in C_\chi} \min_{p \in \mathcal{M}_I} \|p - v\|_2. \tag{4}$$

Likewise, the second distance measures how well $\mathcal{M}_I$ is explained by $\mathcal{M}_\chi$:

$$d_{\mathcal{M}_I \to \mathcal{M}_\chi} := \frac{1}{20} \sum_{n \in [1:20]} \min_{v \in \mathcal{M}_\chi} \|e_I^n - v\|_2. \tag{5}$$

Here, to emphasize the importance of the limbs, we take the first 20 geodesic extrema of the input depth data, which largely correspond to points on the limbs rather than the torso, see Fig. 4. In practice, 20 to 50 geodesic extrema yield similarly stable results. Both distance measures can be computed efficiently for two reasons. Firstly, we showed that geodesic extrema can be extracted very efficiently (Sect.3.2). Secondly, the small number of points leads to a low overall computation time for both distance measures. Since we choose only a subset of points in the distance measure, the distances are sparse. By means of the specific choice of the points, the distances capture the important parts of the pose. The final pose χ_t^* is then given through

$$\chi_t^* := \underset{\chi \in \{\chi_t^{\mathrm{DB}}, \chi_t^{\mathrm{LocOpt}}\}}{\mathrm{argmin}} (d_{\mathcal{M}_\chi \to \mathcal{M}_I} + d_{\mathcal{M}_I \to \mathcal{M}_\chi}). \tag{6}$$

4. Experiments

We implemented the proposed hybrid strategy in C++ and ran our experiments on a standard off-the-shelf desktop PC with a 2.6 GHz CPU. To numerically evaluate and to compare our hybrid strategy with previous work, we use the publicly available benchmark data set of [10]. In this data set, 28 sequences of ToF data (obtained from a Mesa Imaging SwissRanger SR 4000 ToF camera) aligned with ground truth marker positions obtained from a marker-based motion capture system are provided, comprising 7900 frames in total. In addition to numerically evaluating on this data set, we demonstrate the effectiveness of the proposed algorithm in a real-time scenario with fast and complex motions captured from a PMD Camcube 2 in a natural and unconstrained environment, see Fig. 6, 7. In the accompanying video, we show that the same framework also seamlessly works with the Microsoft Kinect depth sensor.

Feature extraction. First, we evaluate the effectiveness of the proposed feature extractor on the benchmark data set. Not all ground truth markers in all frames are visible, thus, for this evaluation, we use only the 3992 frames in which all five end effector markers are visible. A good recognition performance of the feature extractor is needed for a successful subsequent database lookup. In 86.1% of the 3992 frames, each of the found 5 geodesic extrema E_I^5 is less than 0.2 meters away from its corresponding ground truth marker position. This shows that we can effectively detect the end effector positions for most motions contained in the test data set.

Quantitative evaluation. We run our pose reconstruction algorithm on the benchmark data set. Since the surface mesh of the actor is not part of the data set, we scale the input point cloud data so that it roughly fits the proportions of our actor. We manually fix correspondences between each motion capture marker and a mesh vertex. For a test sequence with T frames, let M_t be the number of visible motion capture markers in frame t, let $m_{t,i}$ be the 3D position of the i^{th} visible marker in frame t and $\tilde{m}_{t,i}$ the position of the corresponding mesh vertex of the reconstructed pose.

Then, the average pose error for a sequence is computed as

$$\bar{\epsilon}_{\text{avg}} := \frac{1}{\sum_{t=1}^{T} M_t} \sum_{t=1}^{T} \sum_{i=1}^{M_t} \|m_{t,i} - \tilde{m}_{t,i}\|_2. \qquad (7)$$

The evaluation measure as used in [10] can only be used to show tendencies in accuracy, since only the average distances of the markers to corresponding mesh vertices over all frames of a sequence are regarded, thus possibly averaging away strong local tracking errors. We compare different pose estimation strategies using the evaluation measure for all benchmark sequences, see Fig. 5. To this end, we report on how the components of our algorithm perform individually, without being combined with the hypothesis voting. Using only local optimization (1st bar) has the problem of getting stuck in local minima and often loses track. Using only a database lookup (2nd bar), poses where the end effectors are not revealed by the first five geodesic extrema may yield a false lookup result. Thus, in terms of the average pose error, both methods in isolation do not perform well on all sequences. The 3rd bar shows the result of the proposed hybrid strategy, which is significantly more accurate. Also in comparison to [10] (last bar, std. dev. values were not available), we achieve comparable results for basic motions and perform significantly better in the more complex sequences 20 to 27. Only for sequence 24 we perform worse than [10]. Here, the reason is that this sequence contains a 360° rotation around the vertical axis, which cannot be handled by our framework. However, rotations in the range of ±45° can be handled, since we normalize the input data based on the estimated viewing direction.

Our hypothesis voting decided in 22.5 % of the frames for the retrieval component, and in 77.5 % for the local optimization from the previous frame. As a result, we significantly reduced the average pose error of the final pose estimate in comparison to either method ran individually.

	Total	Prep	Loc.Opt	E_I^{20}	Lookup	Voting
Full res.	16.6 ms	1.2 ms	5.7 ms	6.2 ms	1.2 ms	0.9 ms
	100%	7%	34%	37%	7%	5%
Half res.	10.0 ms	1.1 ms	4.6 ms	1.5 ms	1.2 ms	0.9 ms
	100%	11%	46%	15%	12%	9%

Table 1. Average running times in milliseconds over all frames of the benchmark data set.

Running time. In Tab. 1, we report the average running times in milliseconds per frame. In [10], the authors report a performance of 4 FPS on downsampled input data. By contrast, with our proposed algorithm, we achieve 60.4 FPS (16.6 ms per frame) on average on the full resolution input data, and 100 FPS (10.0 ms per frame) with half of the resolution, which we track with the same accuracy. We also give the running time of each algorithmic component, namely the data preparation phase (Sect. 2), the local optimization component (Sect. 3.1), the feature extraction (Sect. 3.2), the database lookup (Sect. 3.3), and the voting (Sect. 3.4). For the full resolution, the running time of local optimization and the feature extraction are approximately equal. The latter benefits most from downsampling the data. Note that because of our efficient algorithmic components no clear bottleneck is present.

Qualitative evaluation. In Fig. 6, we show example results of fast and complex motions captured in an unconstrained environment. Note that the considered motions

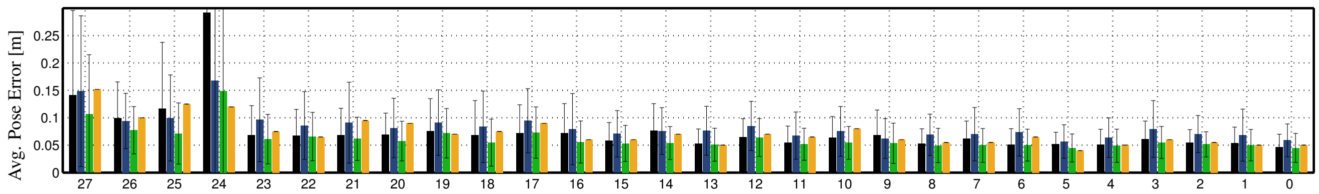

Figure 5. **Left**: Average pose error and standard deviation of sequences 27 to 0 in the data set of [10]. Bars left to right: Using only local optimization, only the database lookup, our results, and values reported by [10] (without std. dev.).

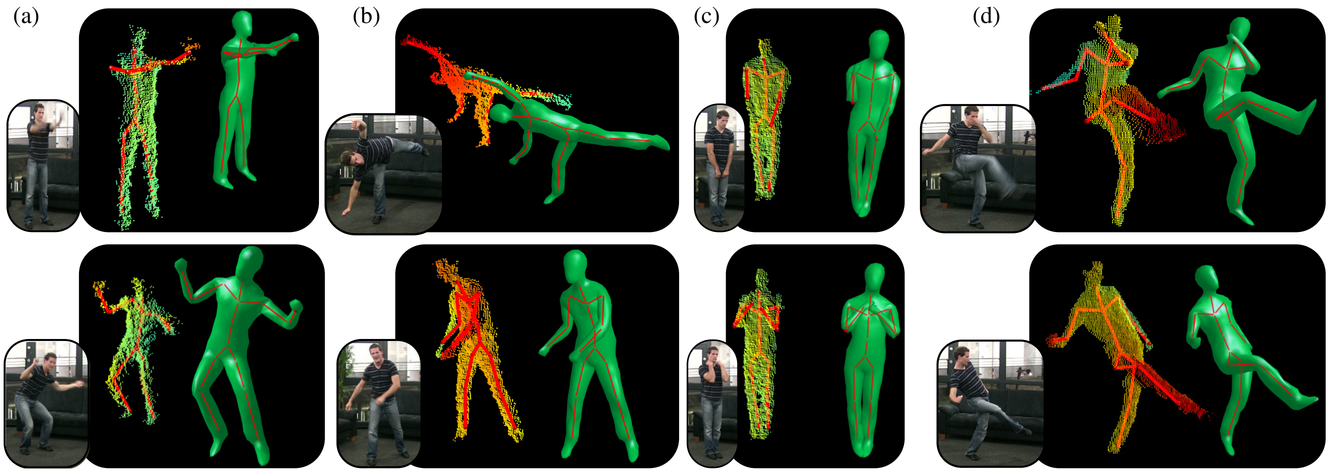

Figure 6. Snapshots of our results on fast and complex motions. For every motion we show a picture of the actor (not used for tracking), ToF data overlayed with the reconstructed skeleton, and a rendering of the corresponding mesh.

are much faster and contain more challenging poses than the ones used in [10]. Each image shows a video frame of the pose captured from a second video camera not used for tracking, the depth data from a different angle overlayed with the estimated skeleton of the pose, and the corresponding mesh. The first column of Fig. 6 (a) shows a very fast arm rotation motion in a pose where the arms are close to being outstretched to the camera, as well as a vivid dancing pose. In (b), we show that because of the normalization step, non-frontal complex poses can also be tracked. Also, a motion where the arm separates upper and lower body in the distance image is successfully tracked. In (c), we show that poses with severe self-occlusion are still a challenge for pose reconstruction. Nonetheless, the overall pose is reliably captured and arm tracking quickly recovers once the occlusion is resolved. The last column (d) shows a sequence of fast and complex kicking motions. Here, the algorithm successfully distinguishes the right and left leg through temporal consistency, which would be difficult by analyzing individual frames alone. Moreover, the arms partly occlude the upper body, and still are reflected well in the estimated pose. In the accompanying video we show the performance of our prototype implementation also with the Microsoft Kinect depth sensor.

First experiments show that actors with different body proportions can be tracked if they are not too different from our body model. To this end, we scale the input data to roughly match the proportions the model, see Fig. 7 and the accompanying video.

Limitations. If the end effectors are not revealed by the geodesic extrema, our algorithm continues to track using local optimization. Then, fast motions lead to unstable poses, which are resolved as soon as the end effectors are detected again. Rotations around the vertical axis within a typical range for interaction ($\pm 45^\circ$) are handled by normalizing the viewing direction. Results become unstable once the actor leaves that range, since on the one hand, local optimization does not find meaningful correspondences anymore, and on the other hand, the database priors can no longer stabilize the pose estimation. To overcome this limitation, one could employ a dynamic model for simulating hidden limbs.

5. Conclusions

In this paper, we showed how we obtain robust and efficient full-body pose estimates from noisy depth image streams within a combined generative and discriminative framework. Here, we contributed with an efficient algorithm for computing robust and characteristic features, enabling real-time performance of the whole framework. Furthermore, by employing an efficient database lookup scheme, we make use of the detected features without hav-

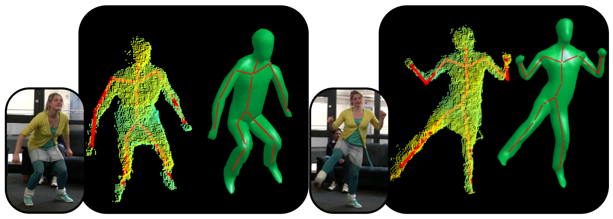
Figure 7. In case of slightly different body proportions, we can simply scale $\mathcal{M}_I$ to roughly match our body model.

ing to rely on a priori semantic labels. Finally, by introducing a robust and efficient sparse Hausdorff distance for fusing local optimization and a database lookup, we further increased efficiency with a distance measure that accounts for the importance of the limbs. In our experiments we go far beyond results of previous work, both in terms of efficiency and robustness of the algorithm, as well as complexity of the tracked motions. In future, we plan to integrate a dynamic model for continuing stable pose estimates for 360°-rotations and for occluded limbs. Furthermore, the low computational complexity of our method will allow us to capture several interacting people.

Acknowledgments. This work was supported by the German Research Foundation (DFG CL 64/5-1) and by the Intel Visual Computing Institute. Meinard Müller is funded by the Cluster of Excellence on Multimodal Computing and Interaction.

References

[1] A. Baak, B. Rosenhahn, M. Müller, and H.-P. Seidel. Stabilizing motion tracking using retrieved motion priors. In *ICCV*, pages 1428–1435, 2009.

[2] A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *CVPR*, pages 1–8, 2007.

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

[4] C. Bregler, J. Malik, and K. Pullen. Twist based acquisition and tracking of animal and human kinematics. *IJCV*, 56(3):179–194, 2004.

[5] T. H. Cormen, C. Stein, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. B&T, 2001.

[6] D. Demirdjian, L. Taycher, G. Shakhnarovich, K. Graumanand, and T. Darrell. Avoiding the streetlight effect: tracking by exploring likelihood modes. In *ICCV*, volume 1, pages 357–364, 2005.

[7] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR*, volume 2, pages 126–133, 2000.

[8] R. Friborg, S. Hauberg, and K. Erleben. GPU accelerated likelihoods for stereo-based articulated tracking. In *ECCV Workshops (CVGPU)*, 2010.

[9] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *CVPR*, pages 1746–1753, 2009.

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

[11] P. Guan, A. Weiss, A. O. Bălan, and M. J. Black. Estimating human shape and pose from a single image. In *ICCV*, 2009.

[12] N. Hasler, T. Thormählen, and H.-P. Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *CVPR*, pages 1823–1830, 2010.

[13] S. Knoop, S. Vacek, and R. Dillmann. Fusion of 2D and 3D sensor data for articulated body tracking. *Robotics and Autonomous Systems*, 57(3):321–329, 2009.

[14] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-flight sensors in computer graphics. *CGF*, 29(1):141–159, 2010.

[15] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH*, pages 165–172, 2000.

[16] T. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *CVIU*, 104(2):90–126, 2006.

[17] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[18] R. Okada and B. Stenger. A single camera motion capture system for human-computer interaction. *IEICE*, E91-D:1855–1862, 2008.

[19] Y. Pekelny and C. Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *CGF*, 27(2):399–408, 2008.

[20] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-time identification and localization of body parts from depth images. In *ICRA*, 2010.

[21] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[22] R. Rosales and S. Sclaroff. Inferring body pose without tracking body parts. In *CVPR*, pages 721–727, 2000.

[23] R. Rosales and S. Sclaroff. Combining generative and discriminative models in a framework for articulated pose estimation. *IJCV*, 67:251–276, 2006.

[24] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H.-P. Seidel. Markerless motion capture of man-machine interaction. In *CVPR*, pages 1–8, 2008.

[25] M. Salzmann and R. Urtasun. Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *CVPR*, 2010.

[26] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–757, 2003.

[27] L. Shapiro and G. Stockman. *Computer Vision*. PH, 2002.

[28] L. Sigal, L. Balan, and M. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *NIPS*, pages 1337–1344, 2008.

[29] R. Y. Wang and J. Popovic. Real-time hand-tracking with a color glove. *TOG*, 28(3), 2009.

[30] Y. Zhu, B. Dariush, and K. Fujimura. Kinematic self retargeting: A framework for human pose estimation. *CVIU*, 114(12):1362–1375, 2010. Special issue on Time-of-Flight Camera Based Computer Vision.