Real-Time Human Pose Tracking from Range Data

Varun Ganapathi[1,2], Christian Plagemann[1,2],
Daphne Koller[1], and Sebastian Thrun[1,2]

[1] Stanford University, Computer Science Department, Stanford, CA, USA
[2] Google Inc., Mountain View, CA, USA
{varung,plagemann,koller,thrun}@stanford.edu

Abstract. Tracking human pose in real-time is a difficult problem with many interesting applications. Existing solutions suffer from a variety of problems, especially when confronted with unusual human poses. In this paper, we derive an algorithm for tracking human pose in real-time from depth sequences based on MAP inference in a probabilistic temporal model. The key idea is to extend the iterative closest points (ICP) objective by modeling the constraint that the observed subject cannot enter *free space*, the area of space in front of the true range measurements. Our primary contribution is an extension to the articulated ICP algorithm that can efficiently enforce this constraint. The resulting filter runs at 125 frames per second using a single desktop CPU core. We provide extensive experimental results on challenging real-world data, which show that the algorithm outperforms the previous state-of-the-art trackers both in computational efficiency and accuracy.

1 Introduction

Tracking human pose in real-time has been a computer vision challenge for decades. Automatic tracking is useful in a variety of domains including human computer interaction, surveillance, gait analysis, the film industry and entertainment. Yet, existing marker-less solutions are limited. They have difficulty with occlusion, unusual poses, sensor noise, and the challenging computational constraints of operation on embedded systems, running side-by-side with actual user applications.

In this paper we focus on improving model-based tracking. Existing model-based trackers use one of two likelihood formulations. ICP-style trackers treat the depth image as a point cloud. They alternate between updating pose to minimize the distance between points and their corresponding locations on the model, and updating the correspondences themselves. The other type of tracker treats the sensor as a camera, and models the ray casting process directly. Their goal is to find the pose that minimizes an error metric based on comparing the observed depth image to the expected depth image derived from the pose estimate. These two algorithms suffer opposing drawbacks: on the one hand, the ray-casting likelihood function is more complete, but is difficult and slow to

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 738–751, 2012.

optimize because of local maxima and plateaus; On the other hand, the ICP-style algorithm, while fast to converge, often converges to the wrong answer.

We propose an algorithm that combines elements of the ray-casting likelihood, specifically knowledge about free space, with the ICP likelihood. By using particular data structures and a suitable model representation, we are able to retain the speed advantages of the ICP method while achieving more accurate tracking. We also contribute a new pose tracking dataset that features more subjects, acrobatic maneuvers, and unusual movements than the currently used Stanford TOF benchmark [1]. As we demonstrate, our new algorithm is able to robustly track widely varying poses at 125 fps using one CPU core.

2 Related Work

Human motion tracking from camera data is a widely studied problem in computer graphics, computer vision and related fields [2]. One class of existing approaches requires multiple cameras to constrain the pose of the tracked person [3], while our approach tracks with a single depth camera. Off-line multi-camera approaches typically aim at achieving highly accurate pose reconstruction up to the surface mesh level [4] at the expense of long processing and inconvenient setups. Usually such systems require an accurate mesh model of the subject before it can be tracked [5].

Learning-based approaches [6–8] require a large amount of training data to sample the large space of body configurations. Many papers have proposed to detect body parts from images as a preprocessing step for full-body pose reconstruction. Plagemann *et al.* [9], Shotton *et al.* [10] and Girshick *et al.* [11] give algorithms for detecting parts from single depth images. These approaches are complementary to ours in that they focus on the part detection problem without exploiting temporal and skeletal constraints. As a result, they do not necessarily output consistent poses at each frame because they might include multiple detections of the same part or miss them entirely. These failures are problematic in a large variety of applications, and therefore such algorithms benefit from being combined with a tracking algorithm like the one we present in this paper.

A related class of algorithms are variants of the iterative closest point (ICP) algorithm, such as, Articulated ICP (AICP) [12–14], Nonrigid ICP [15], or those specifically focused on depth data [16]. To the best of our knowledge, none of the existing algorithms has been shown to be robust enough to track difficult maneuvers like hand-stands, cart-wheels, or even fast full-body movements. The most similar in flavor to our approach is that of Demirdjian *et al.* [14], which updates the pose of individual body parts using ICP and applies constraint projection to enforce articulation and body pose constraints. Our algorithm goes further by introducing a more realistic likelihood function by adding free-space and self-intersection constraints to the ICP likelihood. We apply Chamfer distance transforms, as seen in algorithms that recover pose from silhouettes [17], to efficiently enforce free-space constraints.

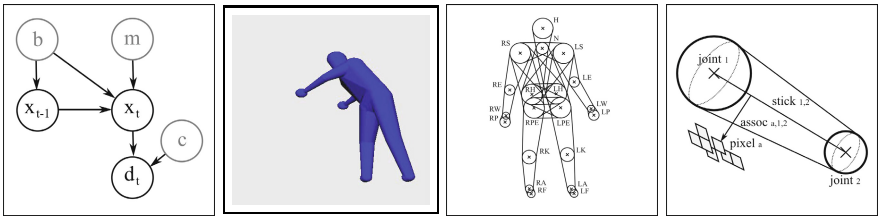

Fig. 1. (a) Graphical model for the tracking algorithm, (b) Human body model, (c) Model schema, (d) Capsule model and pixel correspondence

Combining detection with tracking has been explored in order to avoid local minima and increase the robustness of global optimization of likelihood. Zhu and Fujimura [18] combine articulated ICP with part detection for improved robustness. Ganapathi *et al.* [19] and Siddiqui *et al.* [20] approach the markerless human tracking problem from a hypothesize-and-test angle, with hypotheses additionally generated from part detectors. Ganapathi *et al.* implement this using a GPU-accelerated generative model for synthesizing and evaluating large sets of hypothetical body configurations. Baak *et al.* [21] perform nearest-neighbor search in a database that maps features computed on depth images to poses. These poses are used as additional hypotheses for local optimization.

Compared to Ganapathi *et al.* [19], our approach does not require a detailed mesh model of the person. Though our algorithm can benefit from part detectors, we are able to achieve better performance than Ganapathi *et al.* on the Stanford TOF dataset [19] even without detectors.

3 Human Motion Tracking

The objective is to track the pose of a human over time using a stream of monocular depth images $\{D^1, D^2, \dots\}$. Each image D consists of scalar distance measurements $D = \{d_1, d_2, \dots, d_N\}$, where N is the number of image pixels.

We model our system as the dynamic Bayesian network (DBN) depicted in Fig. 1(a). Here, the state $\mathbf{x}$ denotes the pose of the human at time t (for ease of notation, the time index is omitted for the most recent frame) and the measurement D is the corresponding depth image. Our DBN encodes the Markov independence assumption that $\mathbf{x}$ is independent of $D^1 \dots D^{t-1}$ given $\mathbf{x}^{t-1}$. The model also contains the latent auxiliary variables b, m and $\mathbf{c}$, which will be introduced later in this section.

Our goal is to estimate the most likely state $\hat{\mathbf{x}}$ at time t given only the MAP estimate of the previous frame, that is, $\hat{\mathbf{x}}^{t-1}$, which is determined by solving the inference problem

$$\hat{\mathbf{x}} = \text{argmax}_{\mathbf{x}} \log P(D|\mathbf{x}) + \log P(\mathbf{x}|\mathbf{x}^{t-1}) \ . \tag{1}$$

In this paper, we do not deal with the image segmentation problem and rather assume that a pixel-wise background mask is given. We now describe the measurement model $P(D|\mathbf{x})$ and transition model $P(\mathbf{x}|\mathbf{x}^{t-1})$ in detail.

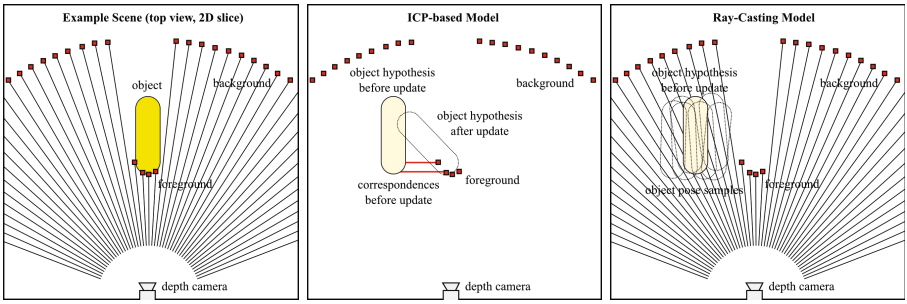

Fig. 2. The two predominant measurement models for depth sensing and their effects on an object tracking process. Left: Example scene with the true object pose. Middle: ICP-based models "pull" the object towards the foreground points. Right: Ray casting models evaluate pose likelihoods based on individual pixel likelihoods.

3.1 Measurement Model

The choice of measurement model has a particularly strong influence on (a) how accurately the DBN is able to represent the human motion sequence and (b) how efficiently Eq. 1 can be solved. Before introducing our proposed measurement model, let us discuss the two most prominent alternatives from the literature as well as their respective shortcomings. As a running example, consider the situation illustrated in the left diagram of Fig. 2. The task is to infer the pose of an object measured by a depth camera from a given initial estimate. Both measurement models make the assumption that the measurements are independent given state, $P(D|\mathbf{x}) = \prod_i P(d_i|\mathbf{x})$.

ICP-Based Model. Given the camera calibration, each distance measurement d_j in the depth image can be converted to a point $\mathbf{d}_j \in \mathcal{R}^3$ in the coordinate system of the camera. This makes it possible to understand the data as a 3D point cloud (rather than as an image-array of depth values). A well-known class of tracking algorithms called *Iterative Closest Point (ICP)* take such 3D point clouds as input and seek to fit a geometric body model to the data by minimizing the length of a correspondence c_j between model surface and data points; see the middle diagram in Fig. 2 for an illustration. We define $S(\mathbf{x})$ to be a geometric representation of the object surface at pose $\mathbf{x}$. In addition, $S(\mathbf{x}, c_j) \to \mathcal{R}^3$ returns the closest point on $S(\mathbf{x})$ according to the data correspondence c_j. ICP-based approaches solve Eq. 1 by alternating between optimizing the correspondence variables and optimizing the state variables. Concretely, the measurement model that leads to the ICP algorithm is defined as

$$p_{\text{ICP}}(\mathbf{d}_k|\mathbf{x}, c_k) \propto \begin{cases} \exp\left(-\frac{1}{2}||S(\mathbf{x}, c_k) - \mathbf{d}_k||^2_\Sigma\right) & \text{, if } d_k \text{ in foreground,} \\ 1 & \text{, if } d_k \text{ in background.} \end{cases}$$

While ICP-based models allow for efficient gradient-based optimization, they fail to model an important aspect of the data. Each measurement d_j not only

provides evidence for the existence of model surface at point $\mathbf{d}_j$, but also for the non-existence of surface between $\mathbf{d}_j$ and the camera. This "negative" information or *free space* constraint is non-trivial to use in ICP-based approaches and, thus, typically omitted. The middle diagram in Fig. 2 illustrates how ICP manages to "explain" the foreground points, but does not utilize the free space information provided by the measurement rays to infer the correct pose of the object.

Ray Casting Model. So-called *Ray Casting Models* are more flexible in this respect, since they model the image formation process directly, that is, they define how a certain scene and body configuration influences each pixel in the depth image. Concretely, for each hypothesized body pose $\mathbf{x}$, a synthetic depth map $\mathbf{r}(\mathbf{x})$ is constructed by casting rays from the camera to $S(\mathbf{x})$. The measurement model in these models is typically defined per-pixel in a form similar to

$$p_{\text{RC}}(d_k|\mathbf{x}) \propto \exp\left(-\frac{1}{2\sigma^2}|r_k(\mathbf{x}) - d_k|^2\right).$$

Here, $r_k(\mathbf{x})$ is the hypothetical (or synthesized) depth measurement for pixel k. While ray casting-based approaches model the object-measurement relationship more faithfully than ICP (by including free space information provided by the rays), they are much harder to optimize.

As the right diagram in Fig. 2 illustrates, approaches using ray casting [19, 20] often make use of the hypothesize-and-test paradigm, that is, they sample model configurations, synthesize the respective depth measurements and compare them to the actual measurements. Such a system is typically hard to optimize, because the *direction* of improvement of the optimization objective has to be estimated indirectly from samples in a high-dimensional space. In the depicted example, all sampled object poses have a similar measurement likelihood.

Ray-Constrained ICP Model. One of the main contributions of this paper is a novel measurement model, termed *Ray-Constrained ICP Model* that combines aspects of each model above. As with ICP, we instantiate a hidden correspondence variable c_j for each measured point $\mathbf{d}_j$. Then we define our measurement likelihood for a given measurement d_k as

$$p_{\text{RC-ICP}}(d_k|\mathbf{x}, c_k) \propto \exp\left(-\frac{1}{2}||S(\mathbf{x}, c_k) - \mathbf{d}_k||^2_\Sigma\right) \cdot I_{\{r_k(\mathbf{x}) \geq d_k\}}, \tag{2}$$

where I_E denotes the indicator function, which is 1 for all points that satisfy an expression E and 0 otherwise.

This measurement likelihood essentially multiplies into the ICP likelihood an approximation of the ray-casting likelihood. This approximation assigns zero probability to poses that would receive extremely low probability according to the ray-casting likelihood, and uniform probability to all other states. As we show in Sec. 4, we can derive an efficient optimization scheme for this objective, even though it contains the valuable non-linear free space information acquired through the depth image.

3.2 Body Model and Pose Parametrization

The discussion so far applies to arbitrary 3D objects. For the purpose of tracking human subjects, we model the human body as a collection of linked 3D volumes; see Fig. 1(b) and (c). The body parts take the form of 3D capsules $\{s_{ij}\}$, each of which connects a joint $\mathbf{x}_i$ to another one, $\mathbf{x}_j$, see Fig 1 (b,c,d). The resulting body model is flexible enough to represent a wide variety of human body shapes, while allowing for fast geometric operations such as collision checks and closest point calculations. Note that, unlike the ray-casting model, our measurement model introduced in the previous section does not require a highly accurate body model. The pose is parametrized by a set of *joint positions* $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_J\}, \mathbf{x}_i \in \mathcal{R}^3$, in the camera coordinate system. Each joint has an associated radius r_i that defines the form of the connecting capsules.

3.3 Transition Model

The last component of the DBN is the transition model $P(\mathbf{x}|\mathbf{x}^{t-1})$, which defines how states are allowed to evolve over time. Apart from motion constraints, such as how far a limb can move within one frame, this term also includes constraints on the intrinsic body configuration, such as limb lengths, and a component to allow larger jumps in state space to regain tracking after partial occlusions or after complete track loss.

Physical Constraints. We do not require the body model to be known precisely ahead of time. Rather, we constrain the tracked model to remain similar in shape to a prototypical human model scaled by a single dynamically estimated latent variable b; see Fig. 1(a). Let $l_{ij}(\mathbf{x}) = ||\mathbf{x}_i - \mathbf{x}_j||$ denote the length of a link s_{ij} between joint i and j in our model. We then define $\bar{l}_{ij}$ to be this quantity calculated on the reference human model and $\hat{l}_{ij} = b\bar{l}_{ij}$ to be the scaled version. Our transition model enforces the constraint that $\forall s_{ij} : (1-\epsilon)\ \hat{l_{ij}} \leq l_{ij}\ (\mathbf{x}) \leq (1+\epsilon)\ \hat{l_{ij}}$, where ϵ is a fraction encoding our uncertainty on link length.

Since multiple rigid objects cannot occupy the same region in space, the transition model also enforces the constraint that capsules do not intersect one another. We prohibit two capsules from penetrating each other if they are not connected by a shared joint.

Motion Model. Apart from self-collision and limb constraints, the transition model includes a discrete set of admissible state transitions selected by a switching variable m; see Fig. 1(a). $m = 0$ indicates regular, smooth human motion following a normal distribution on joint positions, $P(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto \mathcal{N}(0, \sigma^2 I)$. The modes $m > 0$ represent non-smooth state transitions from a library of partial poses followed by the regular smooth update. The non-smooth transitions, which are sampled with a low, uniform probability, ensure that the tracker can recover from partial track loss, e.g. due to temporary occlusion of body parts.

In our experiments, we use a library of seven partial poses, including three configuration changes to the arms (left and right) as well as a full pose reset to the stored prototypical pose. For ease of notation, we denote with $\mathbf{x}_m$ the pose after the potential non-smooth update that is applied to $\mathbf{x}^{t-1}$.

4 Inference

We now describe how to perform efficient MAP inference at each frame, that is, how to solve eq. 1 given the definition of its components in the previous section. We observe that both the measurement model as well as the transition model contain a likelihood term and a set of constraints. By denoting with $\mathcal{C}$ the intersection of all constraints and by making the latent switching variable m explicit, we can rewrite the optimization objective to

$$\begin{aligned} G(\mathbf{x}, \mathbf{c}) &= \log P(\mathbf{x}|\mathbf{x}_m) + \log P(D|\mathbf{x}) + \log P(m) \\ &= \frac{1}{\sigma^2}||\mathbf{x_t} - \mathbf{x}_m||^2 + \sum_k -\frac{1}{2}||S(\mathbf{x}, c_k) - \mathbf{d}_k)||^2_\Sigma + \log P(m)\,. \end{aligned} \tag{3}$$

Our goal is to find $\mathbf{x}, \mathbf{c}, m$ that maximize G subject to the constraint that the state satisfies the physical and ray-casting constraints, that is, $\mathbf{x} \in \mathcal{C}$. Since our library of partial poses is small, we employ a deterministic, frame-wise sampling strategy for m, i.e., in each frame we evaluate the setting $m = 0$ as well as exactly one setting from $m > 0$ using a fixed schedule.

Even after fixing m, the remaining optimization problem is not solvable in *closed form* for the following three reasons: (1) the search space over data associations $\mathbf{c}$ is large, (2) the articulation and self-collision constraints are non-convex, and (3) the free space constraint requires ray casting the model, which depends on $\mathbf{x}$. The first two reasons apply to existing articulated ICP approaches [14], while the last is unique to our approach.

Fortunately, well justified approximations exist that allow us to optimize the problem efficiently and accurately, as detailed in the following.

4.1 Constrained Optimization

Articulated ICP-style algorithms maximize log likelihood by calculating a sequence of pose estimates $\{\mathbf{x}^{[0]}, \mathbf{x}^{[1]}, \dots\}$. The basic algorithmic steps are estimating correspondences, updating the pose and projecting onto the feasible set [14]. We apply projected gradient ascent to interleave maximizing pose and satisfying constraints.

The first estimate is obtained from the prior $\mathbf{x}^{[0]} = \mathbf{x}_m$. The $(n+1)$-th estimate is computed from the n-th one by the following 3-step algorithm.

Step 1. Holding $\mathbf{x}$ constant, maximize the objective with respect to correspondences $\mathbf{c}$. We observe that the terms involving only $\mathbf{c}$ in the objective decomposes into a sum of terms each involving only one correspondence variable c_k. Thus we apply the update $c_k := \text{argmin}_{\hat{c}_k}[S(\mathbf{x}, \hat{c}_k) - \mathbf{d}_k]$ for each foreground measurement

$\mathbf{d}_k$. This calculation is equivalent to finding the closest point in the model to the measurement (hence the name of the algorithm). If a measurement is too far from all existing points in the model, it is temporarily discarded as an outlier, which is equivalent to using a heavy-tailed Gaussian for the measurement likelihood $P(\mathbf{d}_k|\mathbf{x}, c_k)$.

Step 2. The pose estimate is updated by gradient ascent on the likelihood, holding $\mathbf{c}$ fixed,

$$\mathbf{x}^{[k+1]} \leftarrow \mathbf{x}^{[k]} + \gamma(\mathbf{x}^{[k]} - \mathbf{x}^{[k-1]}) + \lambda \nabla_{\mathbf{x}} G(\mathbf{x}^{[k]}, \mathbf{d}; \mathbf{c}) \ ,$$

where γ gives the strength of the "momentum" term and λ is the size of the gradient step.

Step 3. As a result of the gradient update, the state $\mathbf{x}$ may no longer satisfy the physical and ray-casting constraints. We therefore apply constraint projection. Formally, a projection operator for the constraint $\mathcal{C}$, $\text{proj}_{\mathcal{C}}$ is defined as $\text{proj}_{\mathcal{C}}(\mathbf{x}) = \text{argmin}_{\mathbf{x}' \in \mathcal{C}} ||\mathbf{x}' - \mathbf{x}||^2$, taking a configuration $\mathbf{x}$ as argument and returning the closest configuration within $\mathcal{C}$.

While it is difficult to project onto the intersection of all the physical constraints in closed form, it is easier to derive a projection operator for any individual constraint. In order to project onto the entire intersection of the articulation, self-collision and ray-casting constraints, we iterate through each constraint $\mathcal{C}_i$ in a fixed order and apply the update $\mathbf{x} := \alpha\mathbf{x} + (1-\alpha)\ \text{proj}_{\mathcal{C}_i}(\mathbf{x})$, where $\alpha \in [0, 1]$. This iterative algorithm is related to well-known cyclical projection algorithms for finding the intersection of convex sets which are known to converge to the optimal value [22].

This leaves the question of the individual projection operators. Consider an articulation constraint constraining the distance between two joints. To project into the constraint space for this constraint, the two affected joint positions can be moved in 3D space along the capsule center line until the constraint is met.

The projection operator for the ray-casting constraint is more involved. As a reminder, the function $\mathbf{r(x)}$ maps a pose estimate to a synthetic range map. This is the set of measurements a perfect sensor would produce for pose $\mathbf{x}$. The ray-casting constraint states that $r_k(\mathbf{x}) \geq d_k$. We wish to design a data structure that helps us to efficiently project $\mathbf{x}$ onto this constraint.

The first key idea is that the constraint is equivalent to constraining *all* points on the model surface to lie in the 3D region of space behind the measurements, which we denote by $\mathcal{A}$. Since our model surface is defined by symmetric capsules, we can reformulate this surface constraint in terms of joint positions: instead of constraining the surface to $\mathcal{A}$, we constrain the line segments between joints to lie inside $\mathcal{A}$ *shrunk* by the radius of the capsule. We approximate the projection of entire line segments by projecting a finite set of points lying on the line segments.

One way to approach this problem would be to construct a 3D voxel grid and mark all cells inside the allowed space, and perform a 3D distance transform.

Each voxel would then contain the distance and direction to the closest point of the allowed space. For our purposes, this is prohibitively expensive in both space and computational time.

Our approach instead splits the constraint into the intersection of two constraints with simpler projection operators, termed *Silhouette* and *Z-Surface* constraints. Embedded within the overall cyclical projection loop, this will achieve the effect of projection onto $\mathcal{A}$, but much more efficiently. The first set is the foreground silhouette (*Silhouette* constraint) on which we compute a 2D distance transform. If a model point projects outside the silhouette, we project it to the closest point *inside*. If the model point lies inside the silhouette, we check if it is closer than the observed depth and project it backwards (*Z-Surface* constraint).

Using this approximation, the projection into the admissible space of the free space constraint becomes as efficient as an array lookup on a precomputed distance transform. As we show in the following experimental evaluation, enforcing this constraint leads to a large reduction in tracking error without causing substantial runtime overhead.

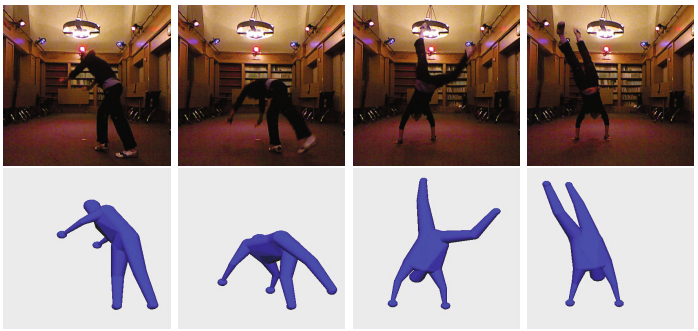

Fig. 3. Sample frames from the `EVAL` test data set. Our system tracks the human motion in real-time from just the stream of depth images.

5 Experiments

The full tracking system has been implemented in `C/C++` and evaluated on a desktop computer running Linux using the following benchmark data sets:

- `SMMC-10` : the Stanford markerless motion capture data set 2010 [1] and
- `EVAL` : a new evaluation data set (3 subjects, 8 sequences each).

`EVAL` was recorded using the Microsoft Kinect camera at approximately 30 fps. The recording includes precise locations of artificial markers attached to the subjects body using the Vicon motion capture system. `SMMC-10` has been recorded using a Mesa SwissRanger time-of-flight sensor.

The goal of this experimental evaluation is (a) to verify that our system is able to track a broad range of human motion in real-time using a single depth camera,

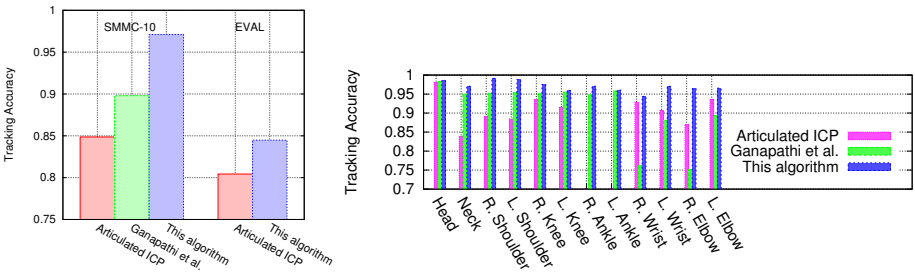

Fig. 4. Tracking performance on `SMMC-10` and `EVAL` as well as comparison to Articulated ICP and Ganapathi *et al.*. The tracking accuracy is significantly higher than for the other approaches – especially for hard-to-estimate joint locations, such as, the elbows.

(b) to show that the system outperforms Articulated ICP [12, 16] and existing ray-casting based approaches [19] on an evaluation data set in accuracy and speed by a large margin and (c) to analyze the practical role of individual elements incorporated in our algorithm, including the ability to make use of body part detections. Articulated ICP has been implemented by deactivating the free space constraints and jump proposals in our system. All depth sequences start with the human subject facing the camera. The system is able to initialize tracking autonomously, because the person starts in a pose that is reasonably close to the standard reset pose.

5.1 Quantitative Tracking Performance

We compared our algorithm quantitatively to state-of-the-art approaches from the literature. Ganapathi *et al.* [19] introduced a model-based tracking algorithm for monocular depth data and published the evaluation data set [1], which we call `SMMC-10` in this paper. We compare experimentally against their results as well as against Articulated ICP on the same data set.

`SMMC-10` contains 28 depth image sequences of human motion and time-synchronized marker positions from PhaseSpace, a professional marker-based motion capture system. We calculated the true joint positions in 3D from the motion capture markers and compare these to the estimated 3D joint positions by our algorithm. Our evaluation metric is the *joint prediction accuracy* per joint. This is defined as the number of correctly predicted joints divided by the number of joints. We count a joint as *detected correctly*, if the algorithm estimates the 3D joint location within a 10cm sphere of the true joint location and as *detected incorrectly*, otherwise. The results are shown in Fig. 4. Our algorithm outperforms the model-based tracker presented by Ganapathi *et al.* and Articulated ICP by a large margin. The prediction error is reduced by 71.6% w.r.t. Ganapathi *et al.* and by 80.8% w.r.t. Articulated ICP. Our algorithm performs inference at 125 frames per second (fps) on a standard desktop computer running Linux on a single CPU core (Intel i7) using no GPU, compared to 230 fps for Articulated ICP. In comparison, Ganapathi *et al.* report inference speed of 10

Algorithm Step (one frame)	Time (ms)	Rate (fps)
Data Preprocessing	1.1	915.2
Subsampling, 160 points	0.54	1859.5
Free Space Precomputation	.90	1101.9
Opt. 1 (no reset) (200 steps)	2.7	184.3
Opt. 2 (partial reset)	2.7	184.3
Total	7.9	125.7

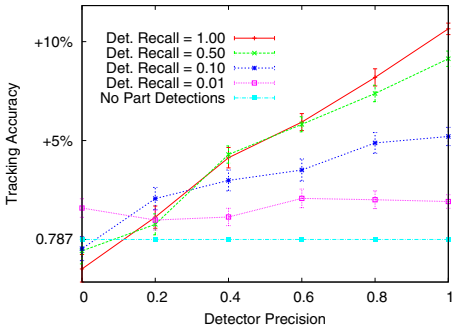

Fig. 5. Left: Run-time performance, single Intel i7 core. Right: Study of how the accuracy of available body part detections influences the tracking accuracy of our algorithm.

frames per second on a desktop computer using a GPU. The left table in Fig. 5 shows the distribution of runtime requirements over the different stages of the algorithm.

5.2 Comparison to Real Part Detectors

We compared the tracking accuracy against the state-of-the-art body part detector by Shotton *et al.* [10] as well as [11] on the `SMMC-10` data set. In principle, these approaches cannot be compared directly to ours, since the purpose differs fundamentally (part detection vs. tracking) and they should be seen as complementary rather than competing. Nevertheless, in a comparison favorable to their approaches we can consider the tracking output of our tracker as a set of part detections (at confidence 1). Comparing our metric from above (tracking accuracy) with their *average joint prediction precision* (which again is disadvantageous for our approach since their average (a) includes low recall regimes while our measure just contains the one actual recall number (b) does not consider occluded parts), we find that our algorithm decreases the error by 44% for [10] (0.971 vs. 0.948) and 26% for [11] (0.971 vs. 0.961). At the same time, our algorithm runs approximately 16 times faster (based on the 50 fps reported in [10] using an 8-core CPU implementation).

5.3 Analysis of Model Components

In this experiment, we analyzed how important the individual model components are for tracking human motion successfully. To this aim, we ran the algorithm on a set of depth sequences and evaluated the tracking accuracy using the *joint prediction accuracy* as described for the previous experiment. We tested the influence of the model components *Z-Surface constraints*, *Silhouette constraints*, and *Jump proposals* by running the algorithm with all possible combinations of these settings. The data set used, termed `EVAL` in this paper, includes artistic motions, such as, hand stands, kicks, sitting down on

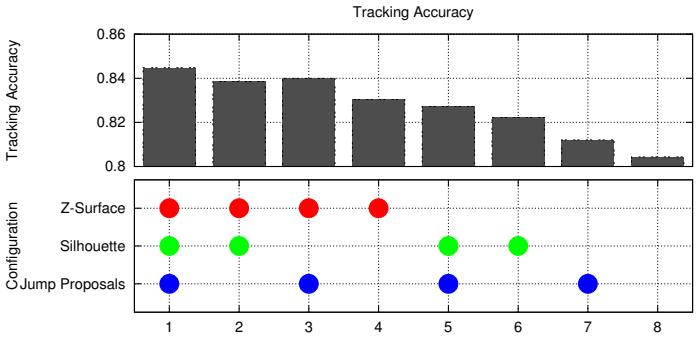

Fig. 6. Top: Tracking accuracy of our algorithm (upper bar plot) for different configurations of the algorithm (below) on the `EVAL` data set

the floor, see Fig. 3 and Fig. 7. This new benchmark data set is available at `http://ai.stanford.edu/~varung/eccv12`.

The quantitative results are given in Fig. 6. Each component adds tracking accuracy to the results, whereby *Z-Surface constraints* provide the largest benefits, followed by *Silhouette constraints*, and *Jump proposals*.

5.4 Integration of Part Detectors

Our framework makes it easy to include bottom-up evidence about body part locations by simply defining an appropriate partial pose update to be considered as a jump proposal, see Sec. 3.3. Such a part detection can then be accepted or rejected by the algorithm automatically. In this experiment, we analyzed how the accuracy of a part detection algorithm influences the tracking accuracy. We consider Sequence 27 of the `SMMC-10` data set, which is the most challenging in the set, featuring a fast *tennis swing*, partial body rotation and a significant amount of self occlusion. To analyze a broad spectrum of part detector quality levels (defined by two constants $P_{\text{precision}}$, P_{recall}), we sample body part detections from the set of available motion capture markers. For each motion capture marker, we sample a detection proportional to the probability P_{recall} and for each chosen detection, we sample a correctness variable according to $P_{\text{precision}}$. If the detection has been sampled as *incorrect*, we assign it a wrong, randomly chosen marker location. This simulates the nature of a detector to confuse body parts and makes it especially difficult for our algorithm to filter out.

Fig. 5 shows the quantitative results. The individual graphs represent recall levels ranging from 0.0 (no part detections) to 1.0 (all parts detected). Precision on the x-axis ranges from 0.0 (all part labels wrong) to 1.0 (all part labels correct) and the y-axis gives the resulting tracking accuracy of our full algorithm. While precise detectors (precision>0.6) lead to a large increase in tracking accuracy as expected, it should be noted that precisions as low as 0.2 (which means that up to 80% of parts are detected at wrong locations) still lead to improved performance. This shows that the algorithm is able to filter out bad detections.

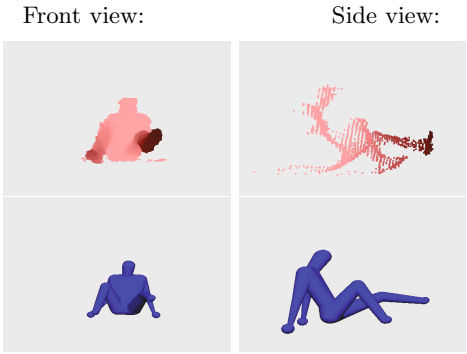

Fig. 7. Sample frames from the `EVAL` test data set. The side view shows that the arms of the subject are not visible due to occlusion. Nevertheless, the algorithm outputs reasonable estimates for the arms, as other constraints limit the feasible set.

6 Conclusion

This paper proposed a technique for tracking human pose in real-time from range data through likelihood maximization. We parameterize human pose through the deformation of a flexible capsule model and derive a measurement model that represents free space constraints and can be optimized efficiently. We also describe a set of natural physical constraints on pose that limit the deformation of the body model and prevent self collision. To tackle the resulting constrained optimization problem efficiently we described an algorithm with the following key elements: data association, accelerated gradient descent, constraint projection, and reinitialization. Experimental results obtained on both a standard data set and a new data set show the speed and accuracy of our approach. We also performed experiments to show that our algorithm can benefit from body part detectors of varying quality.

As our more difficult data set shows, there is still room for improvement. More than one parameter of the body model could be adapted to the specific subject being tracked. Discriminative partial pose detectors could be learned specifically for integration with this framework to focus on the remaining error modes. The framework could also be extended towards fully deformable surfaces meshes which can potentially capture finer configuration details such as cloth deformation.

References

1. Ganapathi, V., Plagemann, C.: Project website and data sets (March 2010), `http://ai.stanford.edu/~varung/cvpr10`
2. Pons-Moll, G., Rosenhahn, B.: Model-based pose estimation. In: Visual Analysis of Humans, pp. 139–170 (2011)
3. Stoll, C., Hasler, N., Gall, J., Seidel, H.P., Theobalt, C.: Fast articulated motion tracking using a sums of gaussians body model. In: IEEE International Conference on Computer Vision, ICCV (2011)

4. de Aguiar, E., Theobalt, C., Stoll, C., Seidel, H.P.: Marker-less deformable mesh tracking for human shape and motion capture. In: CVPR, pp. 1–8 (2007)
5. Corazza, S., Mundermann, L., Chaudhari, A., Demattio, T., Cobelli, C., Andriacchi, T.: A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. Annals of Bio. Eng. (2006)
6. Van den Bergh, M., Koller-Meier, E., Van Gool, L.: Real-time body pose recognition using 2D or 3D haarlets. Int. Journal of Computer Vision 83, 72–84 (2009)
7. Agarwal, A., Triggs, B.: 3D human pose from silhouettes by relevance vector regression. In: Computer Vision and Pattern Recognition (CVPR) (2004)
8. Sun, Y., Bray, M., Thayananthan, A., Yuan, B., Torr, P.H.S.: Regression-based human motion capture from voxel data. In: British Machine Vision Conf. (2006)
9. Plagemann, C., Ganapathi, V., Koller, D., Thrun, S.: Realtime identification and localization of body parts from depth images. In: IEEE Int. Conference on Robotics and Automation (ICRA), Anchorage, Alaska, USA (2010)
10. Shotton, J., Fitzgibbon, A.W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR (2011)
11. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV (2011)
12. Grest, D., Woetzel, J., Koch, R.: Nonlinear Body Pose Estimation from Depth Images. In: Kropatsch, W.G., Sablatnig, R., Hanbury, A. (eds.) DAGM 2005. LNCS, vol. 3663, pp. 285–292. Springer, Heidelberg (2005)
13. Plankers, R., Fua, P.: Articulated soft objects for multiview shape and motion capture. Pattern Analysis and Machine Intelligence 25(9), 1182–1187 (2003)
14. Demirdjian, D., Ko, T., Darrell, T.: Constraining Human Body Tracking. In: IEEE International Conference on Computer Vision, vol. 2 (2003)
15. Hähnel, D., Thrun, S., Burgard, W.: An extension of the ICP algorithm for modeling nonrigid objects with mobile robots (2003)
16. Knoop, S., Vacek, S., Dillmann, R.: Sensor fusion for 3D human body tracking with an articulated 3D body model. In: ICRA (2006)
17. Balan, A., Sigal, L., Black, M., Davis, J., Haussecker, H.: Detailed human shape and pose from images. In: Computer Vision and Pattern Recognition (CVPR), pp. 1–8. IEEE (2007)
18. Zhu, Y., Fujimura, K.: Bayesian 3D Human Body Pose Tracking from Depth Image Sequences. In: Zha, H., Taniguchi, R.-I., Maybank, S. (eds.) ACCV 2009, Part II. LNCS, vol. 5995, pp. 267–278. Springer, Heidelberg (2010)
19. Ganapathi, V., Plagemann, C., Thrun, S., Koller, D.: Real time motion capture using a single time-of-flight camera. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA (June 2010)
20. Siddiqui, M., Medioni, G.: Human pose estimation from a single view point, real-time range sensor. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–8 (June 2010)
21. Baak, A., Müller, M., Bharaj, G., Seidel, H.P., Theobalt, C.: A data-driven approach for real-time full body pose reconstruction from a depth camera. In: IEEE 13th International Conference on Computer Vision (ICCV), pp. 1092–1099. IEEE (November 2011)
22. Bregman, L.M.: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Computational Mathematics and Mathematical Physics 7(3), 200–217 (1967)