Accurate Realtime Full-body Motion Capture Using a Single Depth Camera

Xiaolin Wei Peizhao Zhang Jinxiang Chai*
Texas A&M University

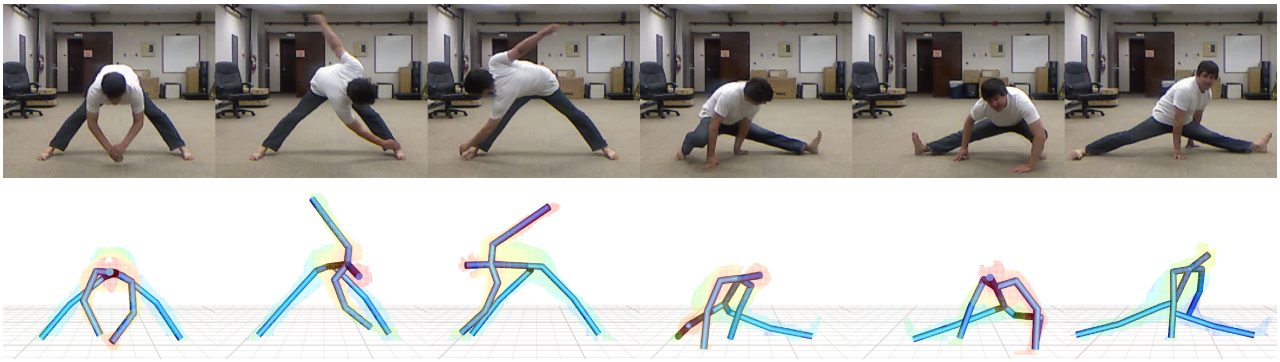


Figure 1: *Our system automatically and accurately reconstructs 3D skeletal poses in real time using monocular depth data obtained from a single camera. (top) reference image data; (bottom) the reconstructed poses overlaying depth data.*

Abstract

We present a fast, automatic method for accurately capturing full-body motion data using a single depth camera. At the core of our system lies a realtime registration process that accurately reconstructs 3D human poses from single monocular depth images, even in the case of significant occlusions. The idea is to formulate the registration problem in a *Maximum A Posteriori* (MAP) framework and iteratively register a 3D articulated human body model with monocular depth cues via linear system solvers. We integrate depth data, silhouette information, full-body geometry, temporal pose priors, and occlusion reasoning into a unified MAP estimation framework. Our 3D tracking process, however, requires manual initialization and recovery from failures. We address this challenge by combining 3D tracking with 3D pose detection. This combination not only automates the whole process but also significantly improves the robustness and accuracy of the system. Our whole algorithm is highly parallel and is therefore easily implemented on a GPU. We demonstrate the power of our approach by capturing a wide range of human movements in real time and achieve state-of-the-art accuracy in our comparison against alternative systems such as *Kinect* [2012].

Keywords: motion capture, performance-based animation, human motion tracking, 3D pose detection, vision-based motion modeling

Links: DL PDF

*e-mail: xwei|stapz|jchai@cs.tamu.edu

ACM Reference Format
Wei, X., Zhang, P., Chai, J. 2012. Accurate Realtime Full-body Motion Capture Using a Single Depth Camera. *ACM Trans. Graph. 31* 6, Article 188 (November 2012), 12 pages. DOI = 10.1145/2366145.2366207 http://doi.acm.org/10.1145/2366145.2366207.

1 Introduction

The ability to accurately reconstruct 3D human poses in real time would allow interactive control of human characters in games, virtual environments, and teleconferences. Such a system would also facilitate natural user interaction with computers, robots, and machines. A major milestone in realtime full-body pose reconstruction is achieved by the recent launch of Microsoft *Kinect* [2012], which automatically infers 3D joint positions from a single depth image. The *Kinect* system is appealing because it is robust, superfast, low-cost, and fully automatic. In addition, it is non-intrusive and easy to set up because the system requires no markers, no motion sensors, and no special suits.

Kinect, despite its high popularity and wide applications, does not provide an accurate reconstruction for complex full-body movements. In particular, the system often produces poor results when significant occlusions occur. This significantly limits the application of *Kinect* because depth data from a single camera frequently contains significant occlusions. The primary contribution of this paper is a robust full-body motion capture system that accurately reconstructs 3D poses even in the case of significant occlusions (see Figure 1). Our system advances the state of the art because it shares the same advantages of *Kinect* but significantly improves the accuracy of the capturing system.

At the core of our system lies a realtime full-body motion tracking process that accurately reconstructs 3D skeletal poses using monocular depth images obtained from a single camera. The idea is to formulate the tracking problem in a *Maximum A Posteriori* (MAP) framework and iteratively register a 3D articulated human model with depth data via linear system solvers. The system is accurate because we integrate depth data, silhouette information, full-body geometry, and temporal pose priors into a unified framework. In addition, we incorporate occlusion reasoning into MAP estimation in order to handle significant occlusions caused by a single camera.

Our 3D pose tracking process, however, requires manual initialization and might be prone to local minima because it initializes current poses with previously reconstructed poses. We address this challenge by complementing tracking with 3D pose detection. 3D pose tracking and detection are complementary to each other. At

one end, 3D pose tracking can produce more accurate results but often requires manual initialization and recovery. At the other end, 3D pose detection can automatically infer 3D human poses from single depth images but often with less accurate results. An appropriate combination of both techniques provides benefits at both ends. To achieve this, we introduce a hybrid motion capture scheme that automatically switches between *3D pose tracker* and *3D pose detector*. We apply the *3D pose tracker* to reconstruct the poses for all the frames except the starting frame and failure frames, which are initialized/reset by the *3D pose detector* automatically. Such a combination not only automates the whole capturing process but also improves the accuracy and robustness of the capturing system.

We implement our hybrid algorithm on GPU to ensure the whole system runs in real time (44 frames per second). In addition, we introduce a simple yet effective skeleton calibration process that automatically computes 3D human skeleton models from depth images of a single reference pose. This enables the system to work for subjects of different skeletal sizes. Our final system for full-body motion capture is robust and fully automatic, runs in real time, and allows for accurate motion capture for different subjects.

We demonstrate the power and effectiveness of our system by capturing a wide range of human movements in real time using a single depth camera. Our system achieves state-of-the-art accuracy in our comparison against alternative systems such as [Kinect 2012; Shotton et al. 2011; Ganapathi et al. 2010]. We assess the quality of reconstructed motions by comparing them with ground truth data simultaneously captured with a full marker set in a commercial motion capture system [Vicon Systems 2011]. We also evaluate the importance of each key component of our 3D pose tracking process. And lastly, we evaluate the robustness of our system and analyze possible failure modes of the entire system.

1.1 Related Work

Our work builds upon a rapidly growing body of recent literature on 3D pose tracking and detection with depth data. One possibility to track 3D human poses is to sequentially register an articulated human body model with depth data via *Iterative Closest Point* (ICP) techniques. Grest and colleagues [2007] explored how to apply ICP techniques to register an articulated mesh model with monocular depth data along with silhouette correspondences between the hypothesized and observed data. The ICP algorithm was also been used in [Knoop et al. 2006] to fit a degenerated cylindrical 3D body model to depth data. However, the ICP algorithm is often sensitive to initial poses and prone to local minima, thereby failing to track 3D human poses from noisy depth data obtained from a single camera (for details, see comparison results in Section 7.2).

An alternative solution is to take a bottom-up approach to predict/estimate 3D joint positions directly from a single depth image. This approach is appealing because it does not assume any 3D human body model, does not require any pose initialization, and does not get trapped in any local minima. For example, Plagemann and colleagues [2010] constructed 3D meshes to detect interest points in depth images and then learned patch classifiers in order to automatically label them as "head", "hands" or "feet". Shotton et al. [2011] formulated 3D pose estimation as a per-pixel classification problem and trained randomized decision trees based on a large database of synthetic depth images rendered from prerecorded motion data for automatic pixel labeling. More recently, Girshick and colleagues [2011] presented an efficient regression forest-based method for detecting 3D human poses from single depth or silhouette images.

Among all the systems, our work is most similar to [Shotton et al. 2011], which forms a core component of the *Kinect* system [2012]. Both systems focus on automatic realtime reconstruction of full-body poses using a single depth camera. Like their system, we formulate 3D pose detection as a per-pixel classification problem and apply randomized decision trees to automatic pixel labeling. Our work, however, is significantly different from theirs in that we develop a realtime 3D tracking process to complement 3D pose detection. Our system, therefore, shares the same advantages of *Kinect* but significantly improves the accuracy of the reconstructed poses. In addition, our solution of reconstructing 3D poses from classified pixels is different from theirs. We explicitly remove the misclassified pixels and apply our 3D tracking algorithm to reconstruct 3D poses in joint angle space, while their system estimates 3D joint positions using probabilistic clustering techniques via meanshift. The comparison results show that our system produces much more accurate results than [Shotton et al. 2011] as well as the *Kinect* system [2012] for depth data with or without significant occlusions (see Section 7.2).

Our work builds upon the success of combining 3D pose tracking and detection to benefit from each other. Siddiqui and Medioni [2010] combined bottom-up detection results and top-down likelihood in a data-driven MCMC framework for upper-body pose estimation. Ganapathi and colleagues [2010] developed an interactive online motion capture system using a single depth camera. Their algorithm used body part proposals from [Plagemann et al. 2010] to track the skeleton with kinematic and temporal information in a probabilistic framework. Baak et al. [2011] proposed to use detection results obtained from [Plagemann et al. 2010] to search similar poses in a prerecorded motion database and combined pose hypothesis with local optimization to reconstruct the final solution. Ye and colleagues [2011] utilized a data-driven pose detection technique to first identify a similar pose in the pre-captured motion database and then refined the pose through non-rigid registration techniques. Our system is unique because it builds upon our highly accurate 3D tracking algorithm and state-of-the-art 3D pose detection algorithm. Our system is advantageous because it is robust and fully automatic, runs in real time, and allows for accurate reconstruction of full-body poses even under significant occlusions.

Our work is related to techniques of tracking 3D human poses using conventional intensity/color cameras (for more details, we refer the reader to [Moeslund et al. 2006]). One notable solution is to perform sequential pose tracking based on 2D image measurements (e.g., [Bregler et al. 2004]), which initializes 3D human poses at the starting frame and sequentially updates 3D poses by minimizing the inconsistency between the hypothesized poses and observed measurements. This approach, however, is often vulnerable to occlusions, cloth deformation, illumination changes, and a lack of discernible features on the human body because 2D image measurements are often not sufficient to determine high-dimensional 3D human movement.

Our work is also relevant to recent efforts in realtime full-body motion capture using only a few sensors [Chai and Hodgins 2005; Slyper and Hodgins 2008; Liu et al. 2011; Tautges et al. 2011]. Reconstructing human poses from a small number of sensors is often an ill-posed problem. One good way to reduce the ambiguity is to utilize prior knowledge embedded in prerecorded motion data. A number of researchers have explored how to utilize prerecorded motion capture data to reduce the reconstruction ambiguity from low-dimensional control signals obtained from a sparse number of markers [Chai and Hodgins 2005] or inertial sensors [Slyper and Hodgins 2008; Liu et al. 2011; Tautges et al. 2011]. Our work is different because we focus on markerless motion capture using a single depth camera. Our system, therefore, is easy to set up and requires no markers, no motion sensors, and no special suits.

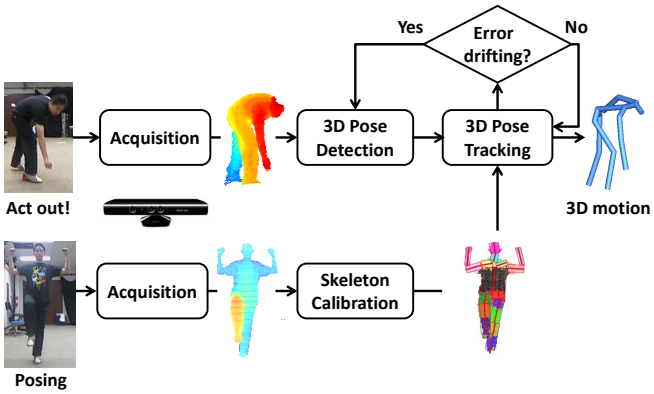


Figure 2: *System overview.*

1.2 Overview

We develop an automatic realtime system that accurately captures 3D full-body motion data using a single depth camera. Here we highlight the issues that are critical for the success of this endeavor and summarize our approach for addressing them (see Figure 2).

3D pose tracking. The core of our system is a realtime tracking process that accurately reconstructs 3D skeletal poses using input data obtained from a single depth camera. Our idea is to formulate the problem in a *Maximum A Posteriori* (MAP) framework and incrementally register 3D skeletal poses with monocular depth data via linear system solvers. One unique property of our tracking process is its sensible reasoning about occlusions in depth data. This allows us to reconstruct 3D human poses under significant occlusions, a capability that has not been demonstrated in the state-of-the-art systems such as *Kinect* [2012].

3D pose detection. 3D pose tracking often requires manual initialization and recovery. This motivates us to develop a 3D pose detection algorithm, which automatically infers 3D poses from single depth images, to complement our tracking process. We formulate 3D pose detection as a per-pixel classification problem and apply randomized decision trees [Amit and Geman 1997] to associate each depth pixel with particular bone segments. In addition, we have introduced an efficient reconstruction technique to estimate 3D joint angle poses from classified depth pixels.

Combining tracking with detection. Our final system combines the advantages of 3D pose tracking and detection. Specifically, we design a hybrid motion capture scheme that automatically switches between 3D pose *tracker* and 3D pose *detector*. We apply the 3D pose *tracker* to reconstruct the poses for all the frames except the starting frame and failure frames, which are initialized/reset by the 3D pose *detector* automatically. Such a combination not only automates the whole capturing process but also significantly improves the accuracy and robustness of the system.

Skeleton calibration. Skeleton calibration ensures the system works for users of different skeletal sizes. We approximate the geometry of each bone segment with a cylindrical model. The calibration process automatically estimates the length and radius of cylindrical models for each bone segment. Each user needs to perform the skeleton calibration step only once.

We describe these components in detail in the next sections.

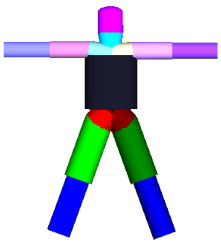


Figure 3: *Our human skeleton models contain 15 bone segments. We define a full-body pose using a set of independent joint coordinates $\boldsymbol{q} \in R^{34}$, including absolute root position and orientation as well as the relative joint angles of individual joints.*

2 Depth Data Acquisition and Preprocessing

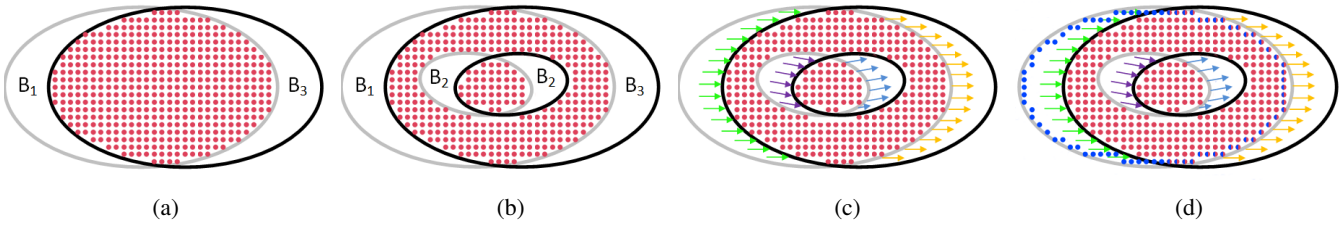
Current commercial depth cameras are often low-cost and can record 3D depth data at a high frame rate. For example, Microsoft *Kinect* cameras, which cost roughly one hundred dollars, give a 320×240 depth image at 30 frames per second (fps) with depth resolution of a few centimeters. In our experiment, we have tested the system on depth data obtained from two types of depth cameras, although our system is applicable to other types of depth cameras as well. We recorded depth image data using a *Swissranger* SR4000 Time-of-Flight camera, which was set to capture depth data at 25 fps with a resolution of 176×144 pixels. We also employed the *Kinect* camera for full-body motion capture. Pixels in a depth image store calibrated depth data in the scene, rather than a measure of intensity or color. In our experiment, each pixel $\mathbf{x} = [u, v]^t$ in the depth image stores not only the depth value D($\mathbf{x}$) but also the corresponding x-y-z coordinates $\mathbf{p} = [x, y, z]^T$ in the 3D space.

Silhouette extraction. The system requires segmenting foreground pixels from background pixels in depth images, though clean segmentation results are not required for system success. We apply the background subtraction algorithm in the *OpenCV* library to remove background pixels for the *Swissranger* camera. For the *Kinect* camera, we use the *Kinect API* [2012] to extract foreground pixels. The system then converts foreground/background information into a binary silhouette image $S(\mathbf{x})$ by setting foreground and background pixels to ones and zeros, respectively. The silhouette images $S(\mathbf{x})$ will later be used for 3D tracking and detection.

Full-body representation. We approximate full-body geometry with a human skeleton model of 15 rigid body segments (see Figure 3). We approximate geometry of each bone segment with a cylindrical model except that the torso is modeled by an elliptic cylinder. The full-body geometry model of a human figure can thus be defined by the radius and length of each bone segment. We describe a full-body pose using a set of independent joint coordinates $\mathbf{q} \in R^{34}$, including absolute root position and orientation as well as the relative joint angles of individual joints. These joints are the head (3 Dof), lower neck (2 Dof), lower back (3 Dof), and left and right clavicle (2 Dof), humerus (3 Dof), radius (1 Dof), femur (3 Dof), tibia (1 Dof).

3 Realtime 3D Pose Tracking

This section describes our tracking algorithm which sequentially reconstructs 3D human poses $\mathbf{q} \in R^{34}$ from input depth data $D(\mathbf{x})$ and silhouette data $S(\mathbf{x})$ obtained from a single depth camera. Our idea is to formulate the tracking problem in the MAP framework and iteratively register a 3D articulated human geometry model with observed data via linear system solvers. In the following,



(a) (b) (c) (d)

Figure 4: *Automatic labeling of depth pixels for likelihood evaluation. Black and gray circles represent the silhouettes of "observed" and "rendered" depth images, respectively. The pixels used for evaluating the **depth image term** are visualized in red. (a) Without self-occlusion, the **depth image term** is evaluated by pixels located in the overlapping region. (b) When self-occlusion occurs, pixels in region B_2 need to be excluded from the **depth image term** evaluation because they might be associated with different bone segments in the "rendered" and "observed" images. (c) For all the non-red pixels which were excluded for the **depth image term** evaluation, we categorize them into four different groups and use them to evaluate the **extra depth term**. Note that pixels in type "1", "2", "3" and "4" are visualized in green, orange, magenta, and blue, respectively. (d) To evaluate the **silhouette image term**, we discard pixels with zero gradients and focus the evaluation on the pixels with non-zero gradients in the foreground of the "rendered" region (see solid-blue pixels and half-blue pixels).*

we explain how to incorporate depth data, silhouette information, full-body geometry information, and pose priors into the MAP estimation framework. We also discuss how to incorporate occlusion reasoning into MAP estimation in order to handle significant occlusions caused by a single camera.

Let C_i be the input data at the current frame i consisting of a depth map D_i and a binary silhouette image S_i. We want to infer from C_i the most probable skeletal poses $\mathbf{q}_i \in R^{34}$ for the current frame given the sequence of m previously reconstructed poses, denoted as $Q_m^i = [\mathbf{q}_{i-1}, ..., \mathbf{q}_{i-m}]$. Dropping the index i for notational brevity, we aim to estimate the most likely poses $\mathbf{q}^*$ by solving the following MAP estimation problem:

$$\mathbf{q}^* = \arg\max_{\mathbf{q}} \; Pr(\mathbf{q}|C, Q_m), \tag{1}$$

where $Pr(\cdot|\cdot)$ denotes the conditional probability.

Using Bayes' rule, we obtain

$$\mathbf{q}^* = \arg\max_{\mathbf{q}} \; Pr(C|\mathbf{q}, Q_m) \cdot Pr(\mathbf{q}|Q_m). \tag{2}$$

Assuming that C is conditionally independent of Q_m given $\mathbf{q}$, we can write

$$\mathbf{q}^* = \arg\max_{\mathbf{q}} \; \underbrace{Pr(C|\mathbf{q})}_{\text{Likelihood}} \cdot \underbrace{Pr(\mathbf{q}|Q_m)}_{\text{Prior}}, \tag{3}$$

where the first term is the *likelihood* term which measures how well the reconstructed pose $\mathbf{q}$ matches the current observation data C, and the second term is the *prior* term which describes the prior distribution of the current pose $\mathbf{q}$ given the previous reconstructed poses Q_m. Maximizing the posteriori produces a most probable skeletal pose $\mathbf{q}^*$ which is consistent with both observation data C and previous reconstructed poses Q_m.

We adopt an "analysis-by-synthesis" strategy to measure how well the reconstructed poses $\mathbf{q}$ match observed data C, including depth data D and silhouette data S. By assuming conditional independence, we can model the likelihood distribution in Equation 3 as the product $Pr(C|\mathbf{q}) = Pr(D|\mathbf{q})Pr(S|\mathbf{q})$. The two factors capture the alignment of reconstructed poses with depth data and silhouette data, respectively. The rest of this section focuses on discussion on how to model $Pr(D|\mathbf{q})$ and $Pr(S|\mathbf{q})$, as well as the prior term $Pr(\mathbf{q}|Q_m)$. We model the distribution of each term as a product of Gaussians, treating each pixel from input data independently.

Depth image term. In general, the evaluation of the likelihood term $Pr(D|\mathbf{q})$ requires identifying the correspondences between the "hypothesized" depth data and "observed" depth data. Previous approaches (e.g., [Knoop et al. 2006; Grest et al. 2007]) often apply Iterative Closest Points (ICP) techniques to find the correspondences between the two. However, ICP techniques often produce poor results for human pose registration (for details, see Section 7.2). To address this challenge, we project 3D depth data onto the 2D image plane and register the "hypothesized" and "observed" depth images via image registration techniques, which avoids building explicit correspondences between the two.

In an analysis-by-synthesis loop, we first use the "hypothesized" joint angle pose $\mathbf{q}$ along with the 3D human body model to render all the surface points on the human body. We define the 3D position of a surface point as $\mathbf{p} = \mathbf{f}(\mathbf{q}; k, \mathbf{p}_0)$, where the vector function $\mathbf{f}$ is the forward kinematic function which maps the local coordinates of the surface point $\mathbf{p}_0$ on the k-th bone segment to the global 3D coordinates $\mathbf{p}$. For notational brevity, we denote the 3D global coordinates of the "rendered" surface point as $\mathbf{p}(\mathbf{q})$. We further project all the "rendered" 3D points $\mathbf{p}(\mathbf{q})$ onto 2D image space with the calibrated camera parameters to obtain a "rendered" depth image D_{render} under the current camera viewpoint.

Assuming Gaussian noise with a standard deviation of σ_{depth} for each depth pixel $\mathbf{x}$, we obtain the following likelihood term for depth image registration:

$$Pr(D|\mathbf{q}) = \prod \frac{1}{\sqrt{2\pi}\sigma_{depth}} \exp\left(-\frac{\|D_{render}(\mathbf{x}(\mathbf{q}), \mathbf{q}) - D(\mathbf{x})\|^2}{2\sigma_{depth}^2}\right), \tag{4}$$

where $\mathbf{x}(\mathbf{q})$ is a column vector containing the pixel coordinates of the "rendered" depth image. Note that unlike color image registration, pixel values in the "rendered" depth image D_{render} are not fully dependent on pixel coordinates $\mathbf{x}(\mathbf{q})$ because the depth values of pixels also directly vary with the reconstructed pose $\mathbf{q}$. This is due to reparameterization of 3D point data $\mathbf{p}(\mathbf{q})$ on 2D image space.

A critical issue for the *depth image* term evaluation is to determine which pixels in the "rendered" image should be included for evalution. This is very important to our tracking process because we adopt gradient-based methods for MAP estimation. Including inappropriate pixels into the *depth image* term will mislead MAP estimation and cause the optimizer to fall in local minima. A simple solution is to use all the foreground pixels, denoted as $\mathbf{x} \in R$, in the "rendered" depth images (Figure 4(a)). This solution, however, often causes inappropriate updates to the current poses because a foreground pixel in the "rendered" depth image might be located in the background region of the "observed" depth image. To address the issue, our system automatically excludes those pixels from the

depth term evaluation by checking whether a foreground pixel in the "rendered" image is located in the background region of the "observed" image. In Figure 4(a), those pixels are located in region $B_1 = \{\mathbf{x} \in R; \mathbf{x} \notin O\}$, where region O includes all foreground pixels in the "observed" image.

When self-occlusion occurs, a pixel located in the overlapping region $R \cap O$ might associate with different bone segments in the "rendered" and "observed" images. Imagine crossing the left arm in front of the chest. Some pixels on the "left arm" in the "rendered" image might inevitably overlap the "chest" pixels in the "observed" image. We illustrate the situation in Figure 4(b), where the "left arm" and "chest" are visualized as the "small" elliptical region and "large" elliptical region, respectively. Similarly, those pixels would mislead MAP estimation and should be excluded from evaluation because they are associated with different bone segments in the "rendered" and "observed" depth images. In our experiment, we adopt a simple yet effective idea to detect those pixels, denoted as $\mathbf{x} \in B_2$ (Figure 4(b)). We calculate the depth difference for each pixel in the overlapping region and exclude the pixels from the *depth image* term evaluation if their depth differences are larger than a specific threshold, which is experimentally set to 8 cm. In Figure 4(b), we visualize all the pixels $G = \{\mathbf{x} \in R; \mathbf{x} \notin B_1 \cup B_2\}$ used for the *depth image* term evaluation in red.

Extra depth term. In practice, even with ground truth poses, the "rendered" depth images might not precisely match the "observed" depth images due to camera noise, cloth deformation, and an approximate modeling of full-body geometry. As a result, the *depth image* term alone is often not sufficient to produce satisfactory results, particularly when significant occlusions occur. This motivates us to introduce an *extra depth* term to evaluate all the depth pixels that were excluded from the *depth image* term evaluation. Besides the region B_1 and B_2, we will also consider the pixels in the region $B_3 = \{\mathbf{x} \in O; \mathbf{x} \notin R\}$.

We categorize all the pixels that were excluded from the *depth image* term evaluation, denoted as $\mathbf{x} \in B_1 \cup B_2 \cup B_3$, into four different groups and use them to obtain a number of correspondences between the "rendered" depth data and the "observed" depth data as follows (Figure 4(c)):

Type 1. For a "rendered" pixel $\mathbf{x}(\mathbf{q}) \in B_1$, we find the closest point $\mathbf{x}_{c1} \in O$ in the "observed" data and push the "rendered" pixel $\mathbf{x}(\mathbf{q})$ towards the "observed" pixel $\mathbf{x}_{c1}$ ("green" arrow).

Type 2. For an "observed" pixel $\mathbf{x} \in B_3$, we search the closest point $\mathbf{x}_{c2}(\mathbf{q}) \in R$ in the "rendered" depth image and pull the "rendered" pixel $\mathbf{x}_{c2}(\mathbf{q})$ to the "observed" pixel $\mathbf{x}$ ("orange" arrow).

Type 3. For a "rendered" pixel $\mathbf{x}(\mathbf{q}) \in B_2$, if its "rendered" depth value is smaller than its "observed" depth value, then the "observed" pixel is occluded by the "rendered" pixel. Therefore, we use the "rendered" pixel $\mathbf{x}(\mathbf{q})$ to find the closest point $\mathbf{x}_{c3} \in O$ in the "observed" image and push the "rendered" pixel $\mathbf{x}(\mathbf{q})$ towards the "observed" pixel $\mathbf{x}_{c3}$ ("magenta" arrow).

Type 4. For a "rendered" pixel $\mathbf{x}(\mathbf{q}) \in B_2$, if its "rendered" depth value is larger than the "observed" depth value, then the "rendered" pixel is occluded by the "observed" pixel. We thus use the "observed" pixel $\mathbf{x}$ to search the closest point $\mathbf{x}_{c4}(\mathbf{q}) \in R$ in the "rendered" depth image and pull the "rendered" pixel $\mathbf{x}_{c4}(\mathbf{q})$ to the "observed" pixel $\mathbf{x}$ ("blue" arrow).

We define the *extra depth* term in 3D space instead of 2D image space. This is because the *extra depth* term is defined for pixels located in non-overlapping regions while correspondence information in non-overlapping regions cannot be estimated on 2D image space via image gradients. We find the correspondences by directly searching the closest points in 3D space as depth pixels in the "rendered" and "observed" depth images also contain corresponding x-y-z coordinates in 3D space. In our implementation, we find the closest points with bidirectional distance measurement to ensure one-to-one correspondences. Meanwhile, we exclude the correspondences from the *extra depth* term evaluation if their depth differences are larger than a specific threshold (8 cm). The "arrows" in Figure 4(c) visualize the correspondences associated with each type of pixels.

We now can define the *extra depth* term as follows:

$$Pr_{extra}(D|\mathbf{q}) = \prod \frac{1}{\sqrt{2\pi}\sigma_{extra}} \exp\left(-\frac{\|\mathbf{p}(\mathbf{q}) - \mathbf{p}^*\|^2}{2\sigma_{extra}^2}\right), \quad (5)$$

where $\mathbf{p}(\mathbf{q})$ is a 3D point in the "rendered" depth image and $\mathbf{p}^*$ is the closest point in the "observed" depth image.

Silhouette image term. The *silhouette image* term $Pr(S|\mathbf{q})$ ensures that the "rendered" silhouette map S_{render} matches the "observed" silhouette map S extracted from input depth data. This term complements the *depth image* term and *extra depth* term, neither of which penalizes the mismatch between the "rendered" and "observed" silhouette images. The *silhouette image* term is particularly helpful for tracking human poses under significant occlusions. Figure 11 in Section 7.3 shows the importance of the *silhouette image* term.

Similarly, evaluating the *silhouette image* term requires building the correspondences between the "rendered" and "observed" silhouettes. We choose to evaluate the *silhouette image* term on 2D image space in the same way as the *depth image* term. This avoids finding explicit correspondences between the "rendered" and "observed" silhouettes, and therefore is less prone to local minima. Assuming Gaussian noise with a standard deviation of $\sigma_{silhouette}$ for each pixel $\mathbf{x} \in R$, we obtain the likelihood term for silhouette registration:

$$Pr(S|\mathbf{q}) = \prod \frac{1}{\sqrt{2\pi}\sigma_{silhouette}} \exp\left(-\frac{\|S_{render}(\mathbf{x}(\mathbf{q})) - S\|^2}{2\sigma_{silhouette}^2}\right), \quad (6)$$

where $\mathbf{x}(\mathbf{q})$ represents 2D pixel coordinates in the "rendered" silhouette image. Since our system applies iterative gradient solvers to estimate the reconstructed poses, we can discard pixels with zero gradients and focus the evaluation on the pixels with non-zero gradients (see solid-blue and half-blue pixels in Figure 4(d)).

Prior term. We evaluate the *prior* term $Pr(\mathbf{q}|Q_m)$ by measuring how well the current pose $\mathbf{q}$ is placed after previous m poses $[\tilde{\mathbf{q}}_{-1}, ..., \tilde{\mathbf{q}}_{-m}]$. We assume that the pose at the current time depends only on the poses at previous two frames. The *prior* term is

$$Pr(\mathbf{q}|Q_m) = \frac{1}{\sqrt{2\pi}\sigma_s} \exp\left(\frac{-\|\mathbf{q} - 2\tilde{\mathbf{q}}_{-1} + \tilde{\mathbf{q}}_{-2}\|^2}{2\sigma_s^2}\right), \quad (7)$$

where $\tilde{\mathbf{q}}_{-1}$ and $\tilde{\mathbf{q}}_{-2}$ are the reconstructed poses in the previous two frames. The prior term assumes a constant acceleration model and thus penalizes the velocity change of 3D human poses.

3.1 Realtime Optimization

We solve the MAP estimation problem defined in Equation (3) by minimizing the negative logarithm of the posteriori probability density function, yielding the following energy minimization problem:

$$\mathbf{q}^* = \arg\min_{\mathbf{q}} \quad E_{depth} + E_{extra} + E_{silhouette} + E_{prior} \quad (8)$$

where

$$E_{depth} = \frac{\|D_{render}(\mathbf{x}(\mathbf{q}), \mathbf{q}) - D\|^2}{2\sigma_{depth}^2}, \quad (9)$$

$$E_{extra} = \frac{\|\mathbf{p}(\mathbf{q}) - \mathbf{p}^*\|^2}{2\sigma^2_{extra}}, \tag{10}$$

$$E_{silhouette} = \frac{\|S_{render}(\mathbf{x}(\mathbf{q})) - S\|^2}{2\sigma^2_{silhouette}}, \tag{11}$$

$$E_{prior} = \frac{\|\mathbf{q}_i - 2\tilde{\mathbf{q}}_{i-1} + \tilde{\mathbf{q}}_{i-2}\|^2}{2\sigma_s^2}. \tag{12}$$

This requires minimizing a sum of squared nonlinear function values. Our idea is to extend the Lucas-Kanade algorithm [Baker and Matthews 2004] to solve the above non-linear least squares problem. Lucas-Kanade algorithm, which is a Gauss-Newton gradient descent non-linear optimization algorithm, assumes that a current estimate of $\mathbf{q}$ is known and then iteratively solves for increments to the parameters $\delta\mathbf{q}$ using linear system solvers.

In our implementation, we initialize the current pose using the previously estimated pose and iteratively perform the following steps until the change of the pose is smaller than a specified threshold:

- Step 1: Given the current pose $\mathbf{q}$ and the full-body human mesh model, we render a depth image $D_{render}(\mathbf{x}(\mathbf{q}), \mathbf{q})$ and a silhouette image $S_{render}(\mathbf{x}(\mathbf{q}))$ under the current camera viewpoint.
- Step 2: For a pixel $\mathbf{x} \in R$ in the "rendered" depth image, we use OpenGL's selection buffer to determine which bone segment (k) the pixel is associated with as well as the local coordinates of the corresponding surface point ($\mathbf{p}_0$) on the k-th bone segment. This step is necessary for evaluating the partial derivatives $\partial\mathbf{p}/\partial\mathbf{q}$ because the global coordinates of surface points $\mathbf{p} = \mathbf{f}(\mathbf{q}; k, \mathbf{p}_0)$ are dependent on the local coordinates $\mathbf{p}_0$ as well as the associated bone segment k.
- Step 3: We calculate the gradients of the "rendered" depth image and the "rendered" silhouette image and other partial derivatives in Equations (9), (10), (11), and (12) to form linear equations (for details, see Appendix A).
- Step 4: We compute the optimal increment $\delta\mathbf{q}$ using linear system solvers and update the current pose: $\mathbf{q} = \mathbf{q} + \delta\mathbf{q}$.

The algorithm usually converges within five iterations as we initialize the solution using the previous reconstructed poses.

Realtime GPU implementation. The fact that each step in the tracking algorithm can be executed in parallel allows implementing a fast solver on modern graphics hardware. By using CUDA to implement our tracking algorithm, the current system runs in real time (48 frames per second) on a machine with Intel Core i7 3.40GHz CPU and GeForce GTX 580 graphics card. For each iteration, the typical computational times for rendering depth/silhouette images, forming and solving linear questions are about 1ms, 1.6ms, and 1.4ms, respectively.

4 3D Pose Detection

One limitation of the tracking process is that it requires manual initialization of the starting poses. In addition, it cannot automatically recover from failures once the system gets stuck in the local minimum. This section describes an efficient 3D pose detection process for automatic initialization/reset of our 3D pose tracking process.

Similar to [Shotton et al. 2011], we formulate the 3D pose detection problem as a per-pixel classification problem. During training, we construct a set $K = \{k_1, ..., k_N\}$ of N classes of pixels. Each class corresponds all pixels located on a particular bone segment except for low-body bone segments, where upper legs and lower legs are divided into two classes to allow for more accurate detection of 3D poses. At runtime, given an input patch $\mathbf{w}(\mathbf{x}_{input})$ centered at a "foreground" depth pixel $\mathbf{x}_{input} \in O$, we want to decide whether or not its measurement matches one of the N classes of pixels. In other words, we want to find for $\mathbf{w}(\mathbf{x}_{input})$ its class label $Y(\mathbf{w}) \in C = \{1, 2, ..., N\}$.

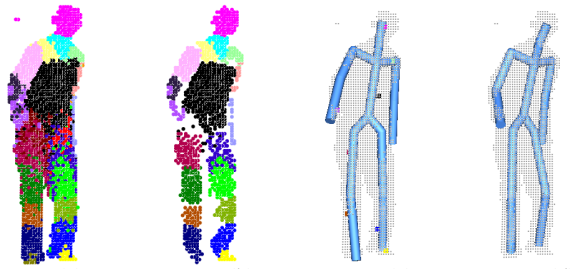


Figure 5: *3D pose detection and reconstruction: (a) classified pixels; (b) classified pixels after outlier removal; (c) pose reconstruction with inverse kinematics; (d) 3D pose refinement via tracking.*

In practice, no classifier is perfect. The classified pixels are often noisy and frequently corrupted by outliers due to classification errors (see Figure 5(a)). We, therefore, develop an efficient technique to automatically remove misclassified pixels (Figure 5(b)). We then process the classified depth pixels to compute the centroid of each labeled cluster and reconstruct the pose in joint angle space with inverse kinematics techniques (Figure 5(c)). Lastly, we apply the 3D tracker described in Section 3 (Figure 5(d)) to refine the reconstructed poses. We discuss each step in detail in the rest of this section.

4.1 Automatic Pixel Labeling with Randomized Trees

We use randomized decision trees [Amit and Geman 1997] to train a classifier for automatic labeling of depth pixels. We advocate the use of randomized trees because they naturally handle multi-class problems and are robust and fast, while remaining reasonably easy to train. A randomized forest is an ensemble of L decision trees $T_1, ..., T_L$. Each node in the tree contains a simple test that splits the space of data to be classified, in our case the space of depth patches. Each leaf contains an estimate based on training data of the posterior distribution over the classes. A new patch is classified by dropping it down the tree and performing an elementary test at each node that sends it to one side or the other. When it reaches a leaf, it is assigned probabilities of belonging to a class depending on the distribution stored in the leaf.

Once the trees $T_1, ..., T_L$ are built, their responses are combined during classification to achieve a better recognition rate than a single tree could. More formally, the tree leaves store posterior probabilities $Pr_{\lambda(l,\mathbf{w})}(c|\mathbf{w})$, where c is a label in C and $\lambda(l, \mathbf{w})$ is the leaf of tree T_l reached by the patch $\mathbf{w}$. Such probabilities are evaluated during training as the ratio of the number of patches of class c in the training set that reach λ and the total number of patches that reach λ. The whole forest achieves an accurate and robust classification by averaging the class distributions over the leaf nodes reached for all L trees:

$$\tilde{c} = \arg\max_c \frac{1}{L} \sum_{l=1,...,L} Pr_{\lambda(l,\mathbf{w})}(c = Y(\mathbf{w})). \tag{13}$$

Node testing. The tests performed at the nodes are simple binary

tests based on simple functions of raw pixels taken in the neighborhood of the classification pixel. Similar to [Lepetit and Fua 2006], our feature function calculates the difference of depth values of a pair of pixels taken in the neighborhood of the classification pixel $\mathbf{x}$: $D(\mathbf{x}+\Delta\mathbf{x}_1) - D(\mathbf{x}+\Delta\mathbf{x}_2)$. We normalize the offset of each pixel (i.e., $\Delta\mathbf{x}_1$ and $\Delta\mathbf{x}_2$) by its depth value $D(\mathbf{x})$ to ensure the features are depth invariant. If the value of a splitting function is larger than a threshold, go to left child and otherwise go to right child. In all our experiments, the patches are of size 50×50. And the optimal threshold for splitting the node is automatically determined by maximizing the information gain for particular features.

Training data. To learn a classifier for automatic depth pixel labeling, we need to construct a training database containing a large set of synthetic depth images. Every pixel in the depth images needs to be annotated with an appropriate class label $c \in 1, ..., N$. To construct the training database, we first use a polygon mesh model to render "synthetic" depth images corresponding to different poses, camera viewpoints, and human skeletal models. Synthetic depth pixels are automatically annotated with class information because every pixel in the "rendered" depth images is associated with a particular bone segment thus a class label. Our training database consists of approximately 20k poses from 65 sequences of human actions such as dancing, sitting, kicking, running, boxing, and tennis.

Randomized trees learning. We use the randomized trees algorithm to learn binary forests. Each tree is trained separately on a small random subset of the training data. The trees are constructed in the classical, top-down manner, where the tests are chosen by a greedy algorithm to best separate the given examples. At each node, several candidates for a feature function are generated randomly, and the one that maximizes the expected gain in information about the node categories is chosen. The process of selecting a test is repeated for each nonterminal node, using only the training examples falling in that node. The recursion is stopped when the node receives too few examples, or when it reaches a given depth.

4.2 Automatic 3D Pose Reconstruction

Our next task is to infer a 3D joint angle pose from classified depth pixels. One way to achieve this is to estimate the centroid of each class and use them for finding the inverse kinematics solution. In practice, no classifier is perfect, so kinematic constraints derived from the classified pixels are often noisy and frequently corrupted by outliers due to classification errors. Figure 5(a) shows the classification result on a test depth image. It is noticeable that some pixels on the "left arm" are misclassified to the "left leg" while some pixels on the "left leg" are incorrectly labeled as the "torso". In practice, direct use of the classified pixels for joint angle pose estimation often produces noisy reconstruction results.

Outlier removal. To address the issue, we formulate the problem as a robust fitting problem and apply random sampling techniques to automatically detect and remove misclassified depth pixels. We choose random sampling techniques because it allows for robust fitting of a model (i.e. a rigid-body bone segment in our application) in the presence of a high percentage of outliers. Specifically, we parameterize the location and orientation of each bone segment with two end points (i.e. inboard and outboard joints) and fit to the classified depth pixel by randomly selecting a pair of depth pixels associated with the same class and counting the number of inliers (i.e. the number of "foreground" depth pixels that are located on or within the bone segment). To speed up the robust fitting process, we discard random samples without counting the number of inliers if the Euclidean distance between the selected depth pixels is larger than the bone length or smaller than one fifth of the bone length. The exact number of samples for each bone heavily depends on the total number of pixels within the projection region of each bone, but we found 2500 samples to be more than enough for all of the testings we tried. Figure 5(b) shows the classified depth pixels after outlier removal.

3D pose reconstruction and refinement. We now can apply inverse kinematics techniques to transform classified depth pixels to a 3D joint angle pose. We calculate the centroid of all the inlier pixels categorized into each of the classes and use them as kinematic constraints for 3D pose estimation. However, when only a small percentage of inliers are detected or bone segments are self occluded, the solution becomes noisy due to inaccurate centroid locations (see Figure 5(c)). We apply the 3D tracker described in Section 3 to refine the solution. Figure 5(c) and (d) show the reconstructed pose before and after 3D pose refinement.

Realtime GPU implementation. We achieve realtime performance (about 44 fps) by executing every step of 3D pose detection process in CUDA. The computational times for pixel classification, outlier removal, and 3D pose reconstruction and refinement (three iterations) are about 3.3ms, 6.7ms, and 13ms, respectively.

5 Combining Tracking with Detection

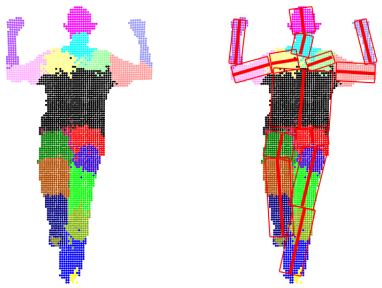
This section discusses how to combine 3D pose *tracker* with 3D pose *detector* to improve the robustness and accuracy of our full-body motion capture system. One possible solution is to apply the 3D pose *detector* to every single depth image and then refine the reconstructed poses using the 3D pose *tracker*. While such a combination produces a fully automatic reconstruction system, the quality of the reconstructed poses is highly dependent on the accuracy of 3D pose detectors. When significant occlusions occur, the system often produces poor results because current 3D pose *detectors* often fail to handle occlusions effectively.

In practice, a majority of frames can be successfully tracked by our tracking process while only a few frames require 3D pose re-initialization. This observation leads us to apply the 3D pose *tracker* to reconstruct the poses for all the frames except the starting frame and failure frames, which can be initialized/reset by the 3D pose *detector*. Briefly, the system first uses the *detector* to initialize the starting pose of the *tracker* and invokes the *tracker* to sequentially register 3D skeletal poses to input data. The system automatically monitors the status of the *tracker*. Once it detects the "failure" mode, the system switches back to the *detector*, uses the *detector* to re-initialize the *tracker*, and then starts a new *tracker* for online motion reconstruction.

A remaining issue is to determine when the *tracker* fails and when to switch to the *detector*. The system automatically identifies tracking failures by measuring how well the reconstructed poses match the current input data, which is achieved by evaluating the sum of the likelihood terms defined in Equation (9), (10) and (11). To be specific, the system automatically switches to the *detector* if one of the following two conditions is satisfied:

- when the "rendered" depth data is not consistent with the "observed" depth data. More specifically, if the average error for the *depth image* term and *extra depth* term is higher than a specific threshold ϵ_1, the system will automatically switch to the "detector".
- when the *silhouette image* term is not well satisfied. On other words, the number of pixels in the overlapping region ($R \cap O$) is smaller than a certain percentage ϵ_2 of the total number of pixels in the region $R \cup O$.

The specific thresholds often depend on sensor noise, the size and proportions of full body geometric models, and the accuracy of



(a) (b)

Figure 6: *Automatic skeleton calibration. (a) the per-pixel labeling result on the calibration pose; (b) the calibrated bone segments (before averaging over symmetric bones).*

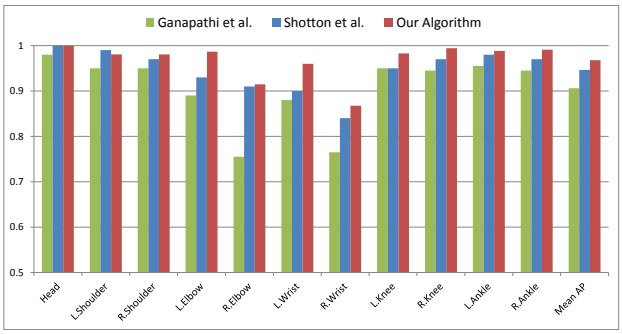


Figure 7: *Comparison against Shotton et al. [2011] and Ganapathi et al. [2010]. The vertical bars show the estimation accuracy of different joints from three different algorithms.*

skeleton geometric models. Since our system provides realtime feedback, we can experimentally determine suitable values that achieve stable performance. For all our results we use the same settings: $\epsilon_1 = 2.5$ cm and $\epsilon_2 = 90\%$.

6 Automatic Skeleton Calibration

In this section, we introduce an automatic skeleton calibration method to ensure the system works for users of different skeletal sizes. We approximate geometry of each bone segment with a cylindrical model except the torso, which is modeled by an elliptic cylinder. We choose to approximate geometry of each bone segment with cylindrical models because they are easy to generalize to full-body geometry of different users. The calibration process automatically estimates the length and radius of cylindrical models for each bone segment (Figure 6). Each user needs to perform the skeleton calibration step only once.

The whole process consists of two steps:

- We instruct the subject to perform a reference pose shown on the screen. We apply the randomized decision trees described in Section 4.1 to automatically label input depth pixels. Unlike the 3D pose detection process, here we restrict the training data sets of randomized trees to the calibration pose only and therefore achieve more accurate pixel labeling results (Figure 6(a)).
- To remove the effect of misclassified pixels, we adopt a similar random sampling technique described in Section 4.2. Briefly, for each bone segment, we randomly sample a radius value and select a pair of depth pixels associated with the bone. The sampled radius value and the pair of depth pixels define a cylinder in 3D space. We evaluate each sample by counting the number of inliers N_i and outliers $\tilde{N}_i$ within the 2D projection of the sampled cylinder. Mathematically, we define the following cost function to evaluate each sample:

$$Q = N_i - \tilde{N}_i \tag{14}$$

  where N_i are the number of depth pixels associated with bone i (i.e. inliers) and $\tilde{N}_i$ is the total number of pixels which are not associated with bone i, including both misclassified pixels and background pixels.

We implement the whole skeleton estimation process in CUDA, which provides realtime feedback on the calibrated skeleton. To improve the accuracy of our calibration process, we assume symmetric human skeletons and average the parameters estimated from the left and right limbs.

7 Results

In this section, we demonstrate the power and effectiveness of our system by capturing a wide range of human movements using a single depth camera (section 7.1). Our comparison against alternative methods shows the system achieves state-of-the-art accuracy (Section 7.2). We assess the performance of our tracking process by dropping off each term in the cost function (Section 7.3). Finally, we evaluate the robustness of the entire system and examine its potential failures by testing on the user playing a number of *Kinect* games (Section 7.4).

7.1 Test on Real Data

We have evaluated our system on two types of depth cameras: *Kinect* camera and *SwissRanger SR4000* Time-of-Flight camera. We test the system on capturing a wide range of human activities, including locomotion such as *walking* and *running*, sports activities such as *warming up exercises*, *golf*, *tennis*, *soccer*, and *Tai-Chi*, and everyday actions such as *picking*, *sitting on ground*, *kicking*, *dancing*, *arm crossing*, and *hand clapping*. Our results are best seen in video form, although we show several frames of a few results here. In the video, we annotate the results obtained from *Kinect* and *SwissRanger* with "KS" and "SR", respectively.

The system works for users with different body size and proportions. The accompanying video includes the reconstruction results from six different subjects, including five males and one female. Another notable feature of the system is to handle significant occlusions caused by a single depth camera. We show the system can accurately capture a wide range of human movements even in the case of significant occlusions, including *sitting down on the ground*, *standing up*, *Tai-Chi*, *kicking*, *picking*, *walking with 360-degrees rotation*, *arm crossing*, and so on. For example, in the "arm crossing" example, we demonstrate that the system is able to track complex arm movements even in the case of significant occlusions. Note that the "detector" in our system was triggered only once to initialize the starting pose for capturing the "arm crossing" action.

7.2 Comparisons Against Alternative Methods

We have evaluated the performance of our system by comparing against alternative methods.

Comparison against [Shotton et al. 2011; Ganapathi et al. 2010]. The authors of [Ganapathi et al. 2010] provided their test data and results for direct comparison. Shotton and colleagues [2011] also evaluated their algorithm on the same data sets.

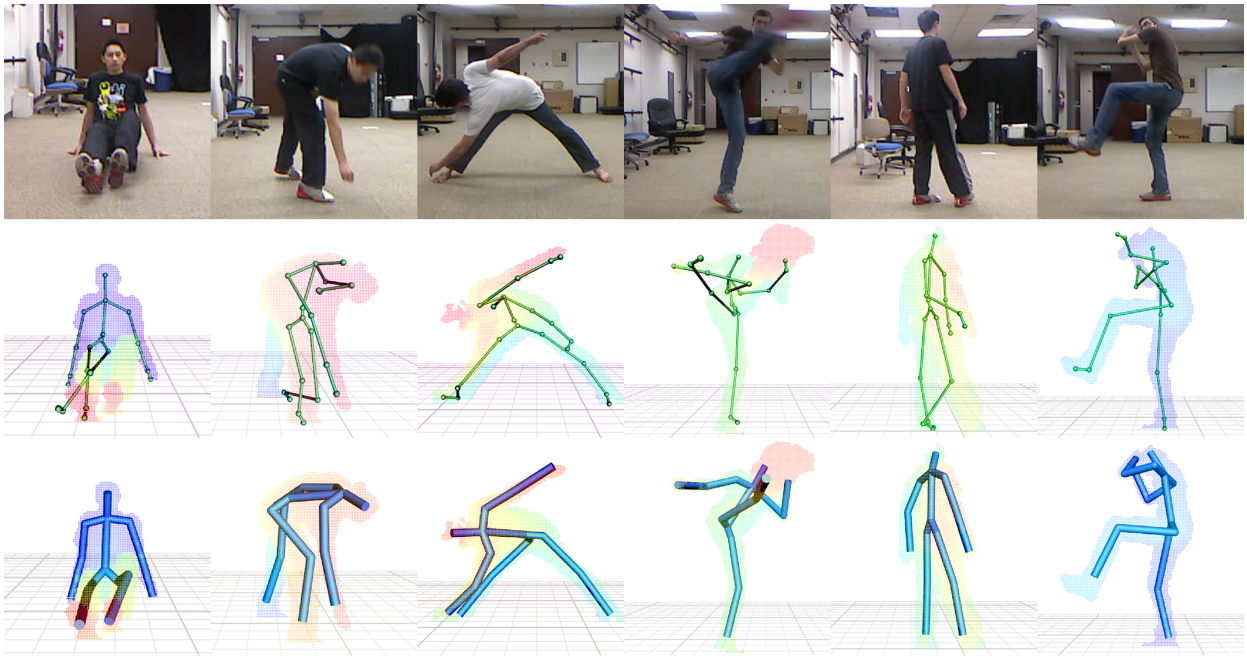


Figure 8: *Comparison against Kinect [2012]. (top) reference image data; (middle) Kinect results; (bottom) our results.*

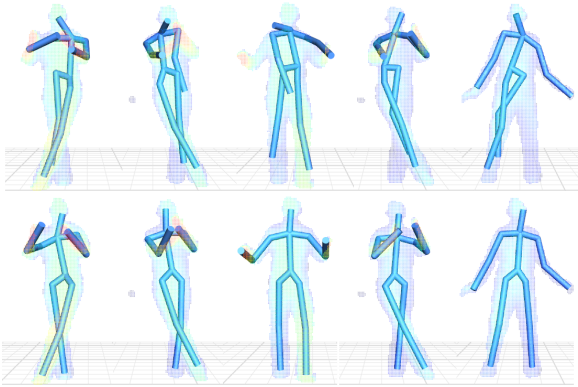


Figure 9: *Comparison against ICP. (top) the result obtained from the ICP algorithm; (bottom) our result.*

In this experiment, we make a comparison between our method and [Shotton et al. 2011; Ganapathi et al. 2010]. Figure 7 shows our algorithm significantly improves average precision of reconstruction results. Our comparison is based on the same evaluation metric as in [Ganapathi et al. 2010; Shotton et al. 2011]. In particular, the bars in Figure 7 show estimation accuracy of different joints. The joint position within D meters of the ground truth is counted as "true positive" (correct). Otherwise, it is counted as "false positive". The precision of one joint is the fraction of joint positions that are correct (within D meters of the ground truth). We set D to *0.1m*. This is because Ganapathi and colleagues [2010] found that individual marker errors of 10 cm or lower can be interpreted as perfectly tracked markers, since this corresponds to the approximate accuracy of the recorded ground truth data.

Comparison against *Kinect* [2012]. We compare the system against the state-of-the-art in motion capture using a single depth camera [Kinect 2012]. We download the most recent version of Microsoft *Kinect* for windows [2012] and test both systems on capturing a wide range of human movements. Here, we focus our comparison on capturing human movements in the presence of significant occlusions because almost all the benchmark data sets [Ganapathi et al. 2010] in the previous comparison do not involve significant occlusions. The accompanying video highlights a side-by-side comparison between our system and *Kinect*. Figure 8 shows several sample frames for a side-by-side comparison between our result and the result obtained by *Kinect*. The comparison results clearly show the advantage of our system over *Kinect*. Note that the *Kinect* system [2012] builds on Shotton et al. [2011].

Comparison against ICP techniques. We also compare our 3D pose tracking process described in Section 3 with Iterative Closest Point (ICP) techniques [Knoop et al. 2006; Grest et al. 2007]. We start both algorithms with the same initial pose. The comparison result shows that our tracking process is much more robust and accurate than the ICP algorithm. In the *hand clapping* example shown in Figure 9, our tracking process successfully tracks the entire motion sequence while ICP fails to track most of frames. This is because ICP is often sensitive to initial poses and prone to local minimum, particularly involving tracking high-dimensional human body poses from noisy depth data.

Comparison against *Vicon* [2011]. In this experiment, we quantitatively assess the quality of the captured motion by comparing with ground truth motion data captured with a full marker set in a twelve-camera *Vicon* system [2011]. The average reconstruction error, which is computed as average 3D joint position discrepancy between the estimated poses and the ground truth mocap poses, was about 5.0 cm per joint per frame. Figure 10 shows a side-by-side comparison between our result and the result obtained by *Vicon*.

7.3 Evaluation on Tracking Process

We have evaluated the performance of our tracking process by dropping off each term of the cost function described in Equation 8.

The importance of occlusion handling. We compare results obtained by the tracking process with or without "occlusion handling" (See Figure 11(a)). Tracking without "occlusion handling" includes all the pixels in $\mathbf{x} \in B_2$ to evaluate the *depth image* term. In con-

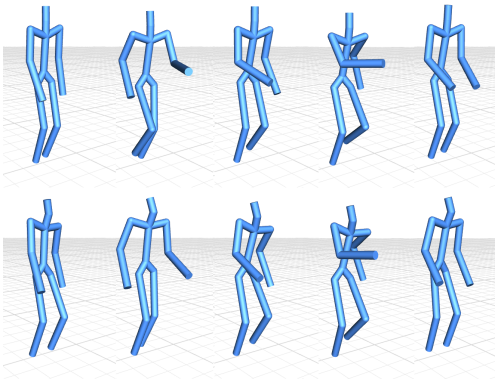
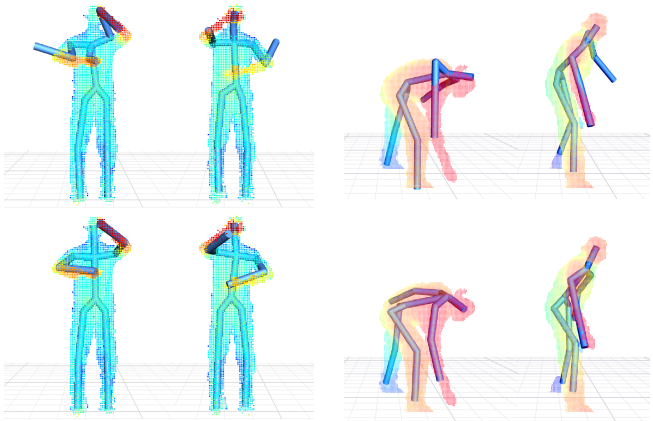


Figure 10: *Comparison against Vicon [2011]. (top) ground truth data captured with a full marker set in a twelve-camera Vicon system; (bottom) our result.*



(a) (b)

Figure 11: *Evaluation on the tracking process. (a) the importance of occlusion handling: the top and bottom rows show the tracking results with and without occlusion handling. (b) the importance of the silhouette image term: the top and bottom rows show the tracking results with and without the silhouette term.*

trast, tracking with "occlusion handling" excludes all the pixels in $\mathbf{x} \in B_2$ from the *depth image* term evaluation and instead includes them into the *extra depth* term evaluation. Our video shows that tracking without "occlusion handling" gets stuck in wrong results several seconds after the tracking starts. With "occlusion handling," the algorithm can accurately reconstruct 3D poses across the entire sequence.

The importance of silhouette image term. We evaluate the importance of the *silhouette image* term. The side-by-side comparison in Figure 11(b) shows that the *silhouette image* term ensures better silhouette registration, thereby producing more accurate results for the reconstructed poses.

7.4 Evaluation of System Robustness

We have evaluated the robustness of the system by testing on real subjects playing five *Kinect* games, including *Adventure, Star War, Fruit Ninja, Dance Central (easy), and Dance Central (hard)*. Table 1 reports the statistics of our analysis on system failure.

Tracking failure analysis. For each game, we compute the percentage of the total frames which were switched to the *detector*. The *tracking failure percentage* indicates the effectiveness of the tracking process. On average, 0.46% of the total frames were re-initialized by the *detector* while the rest of the frames were automatically reconstructed by the *tracker*. This confirms our assumption that a majority of frames can be successfully reconstructed by 3D *tracker* while only a few frames require 3D pose re-initialization. We have observed two typical failure modes in the tracking system. The tracking system often gets stuck in local minima and fails to produce good results when tracking extremely fast movement due to poor initialization of the tracking poses and noisy depth measurement caused by fast motion. In addition, the system might fail to reconstruct accurate joint angle poses for disoccluded bone segments. This is mainly because of poor initialization of the current poses. Note that we initialize the current poses using the previous poses. However, when a bone segment is not visible to the current camera, its 3D joint angle value is often ambiguous and cannot be reliably estimated.

Kinect games	Tracking failure pct	System failure pct
Adventure	0.17%	0
Star War	0.26%	0
Fruit Ninja	0.23%	0.01%
Dance Central (easy)	0.35%	0.01%
Dance Central (hard)	1.28%	0.52%
Average	0.46%	0.11%

Table 1: *Evaluation of system robustness.*

System failure analysis. We evaluate the robustness of the entire system by calculating the percentage of the total failure frames. In our evaluation, we ask the user to manually label all failure frames across the entire sequence of each testing motion. The *system failure percentage* measures the effectiveness of the whole system as neither tracking process nor detection process is able to estimate the pose reliably. The average *system failure percentage* is about 0.1% (about 1 failure pose every 33 seconds). The whole system fails when neither tracking process nor detection process is able to produce good results. The detection system often fails to generate good results when the testing pose is very different from training data sets and/or when human bodies are under significant occlusions.

8 Conclusion

In this paper, we have developed an end-to-end full-body motion capture system using a single depth camera. Our system is appealing because it is low-cost and fully automatic, runs in real time, and can accurately reconstruct 3D poses even in the case of significant occlusions. The system is also non-intrusive and easy to set up because it requires no markers, no sensors, and no special suits. We have demonstrated the power of our approach by capturing a wide range of complex human movements. The system achieves state-of-the-art accuracy in our comparison against *Kinect*.

Complementing tracking with detection not only automates the capturing process but also improves the accuracy and robustness of the whole system. Our framework for combining human pose tracking and the detection process is very flexible. We believe that any efficient 3D pose detector such as [Shotton et al. 2011] can be plugged into our framework to allow for automatic and robust reconstruction of 3D human poses from single-camera depth data.

Our detection process follows the same pixel classification framework as described in [Shotton et al. 2011]. However, our solution of reconstructing 3D poses from classified pixels is different from

theirs. Unlike their system, which estimates 3D joint positions using probabilistic clustering techniques via meanshift, we utilize a 3D human body mesh model obtained from our skeleton calibration process to remove the misclassified pixels and apply inverse kinematic techniques to reconstruct human poses in joint angle space. We also employ our 3D tracking process to refine the reconstructed poses in joint angle space, thereby further improving the accuracy of 3D pose estimation.

Our skeleton calibration process enables our system to work for human subjects of different skeletal sizes. We choose to approximate the geometry of each bone segment with cylindrical models because they are easy to calibrate and easy to generalize to different body size and proportions. In the future, we plan to model geometry of human bodies with skinned mesh models. This will certainly improve the performance of our 3D pose tracking and detection process. We also plan to increase the size of training datasets to improve the generalization ability of our pose detector, as the current training datasets are much smaller than those used in [Shotton et al. 2011].

Another way to improve the accuracy and robustness of the system is to combine depth data with color image data. We are particularly interested in incorporating color and texture information obtained from a video camera in the current tracking framework. One possible solution is to integrate the model-based optical flow equations derived from color image data into the cost function. In addition, as noted by prior research [Chai and Hodgins 2005; Liu et al. 2011], kinematic priors learned from prerecorded motion data could be used to constrain the pose in the solution space of natural appearance to further improve the quality of reconstruction poses.

APPENDIX

A Linear Approximation

In this section, we show how to linearize the nonlinear expressions in Equation (9), (10), (11) and (12) so that the nonlinear least squares problem can be iteratively solved via linear system solvers. Similar to Lucas-Kanade algorithm, we perform a first-order Taylor expansion for nonlinear expressions and obtain the following linear equations on $\delta\mathbf{q}$:

$$\frac{1}{\delta_{depth}}(\nabla D_{render}\frac{\partial \mathbf{x}}{\partial \mathbf{p}}+\frac{\partial D_{render}}{\partial \mathbf{p}})\frac{\partial \mathbf{p}}{\partial \mathbf{q}}\delta\mathbf{q} = \frac{1}{\delta_{depth}}(D-D_{render}), \tag{15}$$

$$\frac{1}{\delta_{silhouette}}\nabla S_{render}\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{q}}\delta\mathbf{q} = \frac{1}{\delta_{silhouette}}(S-S_{render}), \tag{16}$$

$$\frac{1}{\delta_{extra}}\frac{\partial \mathbf{p}(\mathbf{q})}{\partial \mathbf{q}}\delta\mathbf{q} = \frac{1}{\delta_{extra}}(\mathbf{p}^{*}-\mathbf{p}), \tag{17}$$

$$\frac{1}{\delta_{s}}\delta\mathbf{q} = \frac{1}{\delta_{s}}(2\tilde{\mathbf{q}}_{i-1}-\tilde{\mathbf{q}}_{i-2}-\mathbf{q}), \tag{18}$$

where image gradients ∇D_{render} and ∇S_{render} are evaluated on the "rendered" depth and silhouette images, respectively. Note that the standard deviations δ_{depth}, $\delta_{silhouette}$, δ_{extra} and δ_s are used to control the weights for each term are experimentally set to 1, 100, 0.29, and 14.3, respectively.

Acknowledgement

The authors would like to thank Hui Lou for her assistance of experiments. We would also like to thank all of our motion capture subjects. This work was supported in part by the National Science Foundation under Grants No. IIS-1065384 and IIS-1055046.

References

AMIT, Y., AND GEMAN, D. 1997. Shape quantization and recognition with randomized trees. *Neural Computation*. 9(7):1545–1588.

BAAK, A., MÜLLER, M., BHARAJ, G., SEIDEL, H.-P., AND THEOBALT, C. 2011. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *IEEE 13th International Conference on Computer Vision (ICCV)*, 1092–1099.

BAKER, S., AND MATTHEWS, I. 2004. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*. 56(3):221–255.

BREGLER, C., MALIK, J., AND PULLEN, K. 2004. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision*. 56(3):179–194.

CHAI, J., AND HODGINS, J. 2005. Performance animation from low-dimensional control signals. In *ACM Transactions on Graphics*. 24(3):686–696.

GANAPATHI, V., PLAGEMANN, C., KOLLER, D., AND THRUN, S. 2010. Real time motion capture using a single time-of-flight camera. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 755–762.

GIRSHICK, R., SHOTTON, J., KOHLI, P., CRIMINISI, A., AND FITZGIBBON, A. 2011. Efficient regression of general-activity human poses from depth images. In *Proceedings of IEEE 13th International Conference on Computer Vision*, 415–422.

GREST, D., KRUGER, V., AND KOCH, R. 2007. Single view motion tracking by depth and silhouette information. In *Proceedings of the 15th Scandinavian Conference on Image Analysis (SCIA)*, 719–729.

KINECT, 2012. Microsoft Kinect for Xbox 360.

KNOOP, S., VACEK, S., AND DILLMANN, R. 2006. Sensor fusion for 3D human body tracking with an articulated 3D body model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1686–1691.

LEPETIT, V., AND FUA, P. 2006. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 28(9): 1465–1479.

LIU, H., WEI, X., CHAI, J., HA, I., AND RHEE, T. 2011. Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*, ACM, I3D '11, 133–140.

MICROSFT KINECT API FOR WINDOWS, 2012. http://www.microsoft.com/en-us/kinectforwindows/.

MOESLUND, T. B., HILTON, A., AND KRUGER, V. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*. 104:90–126.

PLAGEMANN, C., GANAPATHI, V., KOLLER, D., AND THRUN, S. 2010. Realtime identification and localization of body parts

from depth images. In *Proceedings of International Conferences on Robotics and Automation (ICRA 2010)*, 3108–3113.

SHOTTON, J., FITZGIBBON, A., COOK, M., SHARP, T., FINOCCHIO, M., MOORE, R., KIPMAN, A., AND BLAKE, A. 2011. Real-time human pose recognition in parts from a single depth image. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1297–1304.

SIDDIQUI, M., AND MEDIONI, G. 2010. Human pose estimation from a single view point, real-time range sensor. In *CVCG at CVPR*.

SLYPER, R., AND HODGINS, J. 2008. Action capture with accelerometers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 193–199.

TAUTGES, J., ZINKE, A., KRÜGER, B., BAUMANN, J., WEBER, A., HELTEN, T., MÜLLER, M., SEIDEL, H.-P., AND EBERHARDT, B. 2011. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics*. 30(3): 18:1–18:12.

VICON SYSTEMS, 2011. http://www.vicon.com.

YE, M., WANG, X., YANG, R., REN, L., AND POLLEFEYS, M. 2011. Accurate 3D pose estimation from a single depth image. In *Proceedings of IEEE 13th International Conference on Computer Vision*, 731–738.