Accurate 3D Pose Estimation From a Single Depth Image

Mao Ye[1] Xianwang Wang[2]
Ruigang Yang[1] Liu Ren[3] Marc Pollefeys[4]
University of Kentucky[1] HP Labs, Palo Alto[2] Bosch Research[3] ETH Zürich [4]

Abstract

This paper presents a novel system to estimate body pose configuration from a single depth map. It combines both pose detection and pose refinement. The input depth map is matched with a set of pre-captured motion exemplars to generate a body configuration estimation, as well as semantic labeling of the input point cloud. The initial estimation is then refined by directly fitting the body configuration with the observation (e.g., the input depth). In addition to the new system architecture, our other contributions include modifying a point cloud smoothing technique to deal with very noisy input depth maps, a point cloud alignment and pose search algorithm that is view-independent and efficient. Experiments on a public dataset show that our approach achieves significantly higher accuracy than previous state-of-art methods.

1. Introduction

Human motion modeling has many applications in digital entertainment, health care, and surveillance. While marker-based motion capture (mocap) and analysis systems have received commercial success; the need for markers, the technical expertise required for using the system, and the high hardware cost limit its adoption in many practical applications. Therefore markerless mocap has been an active research topic in computer vision for the last two decades.

Markerless mocap research has always focused on using one or more regular video cameras (e.g., [14]). In this paper we present a novel approach that uses only the depth information to reconstruct articulated body configurations. The motivation for our approach is two-fold. First, depth measurement avoids the ambiguity caused by perspective projection in 2D images; and is invariant to lighting conditions, therefore background objects can be easily segmented. Secondly, low-cost active depth sensors are becoming more and more available. These sensors, based on the principle of time-of-flight(ToF) [10] or active lighting [13], provide more stable depth maps than passive approaches. Nevertheless, using depth for motion capture is not as simple as it appears at first blush. First, a depth sensor only generates a point cloud with noise and outliers; semantic information about which part corresponds to which joint must be extracted. Secondly, there exists large occlusion: at least 50% of the body is not observable in any single view.

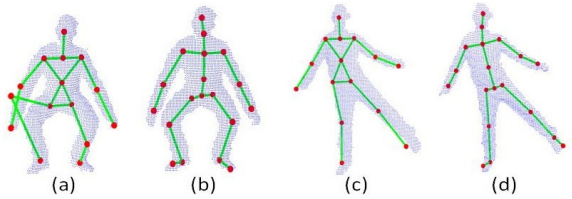

Figure 1. Examples of estimation results using pose tracking algorithms in [20]((a) and (c)) and our method ((b) and (d)), from depth images captured by Kinect.

We use a pre-captured motion database to constrain the possible body configuration space. The motion database contains both body surface models and corresponding skeleton body configurations. When an input depth map is matched with a body surface model, it obtains not only the semantic labeling from the surface model, but also the underlying body configuration. We then further optimize the body configuration using a non-rigid point registration process. It serves two purposes: first to account for the difference between the sample surface model; and second to fill in the missing regions that are occluded in the input. In this two-stage process, we avoid the typical problem of a huge motion database associated with directly mapping input data to body configuration, while we also provide good initial estimation so that optimization-based refinement is unlikely to be trapped in local minima. This is the most novel aspect of our system.

An important technical contribution of this paper is a new *view-independent* matching algorithm between a 3D full-body surface mesh and a depth map. A new challenge in any single-view setup is that the input is *view-dependent* and incomplete. Matching the depth map directly with a complete surface mesh, *with non-rigid deformation*, can lead to inaccurate or even wrong initial body configuration estimation and eventually reduce the final accuracy. Toward this end, we apply PCA to both the input depth map and the motion database; first aligning them in the three prin-

cipal axes, then searching in a reduced space to both accelerate the computation and remove ambiguity caused by small variations in postures. Our method can effectively handle body-size variations across subjects, benefiting from the matching approach and the non-rigid point registration. In addition, we extend a point denoising scheme to significantly reduce the noise and outliers in the input depth map – a problem that is detrimental in practice.

Using active depth sensors for mocap has been a new topic in the last few years. In particular Microsoft has announced its Kinect camera [13, 21] that can capture body movement for computer-human interaction. Qualitative evaluations using kinect are conducted on our method as well as the algorithms in [20], and will be discussed in Section 7; while some examples are shown in Figure 1. In addition, we validate our approach using a publicly available dataset and achieve an average accuracy of 38mm, which is significant compared to previous state of the art (100mm in [9]). Quantitative comparisons between our method and [21] on the this dataset are also discussed in Section 7. With further research and development in this direction, we believe that low-cost motion capture and analysis software can have enormous impact beyond computer gaming, such as socially important problems in health care for which quantifiable accuracy is the key.

2. Related Work

Human motion capture has been a highly active research area in computer vision and graphics, due in part to its many applications. Such applications span fields as diverse as security surveillance, medical diagnostics, games, movies and sports. Traditional marker-based motion capture systems provide a feasible solution, however such systems carry the burden of specially designed equipment or suits, which is inconvenient for many practical applications. Therefore, non-invasive marker-less approaches are the main focus of research in recent years. It is beyond the scope of this paper to provide a comprehensive review. Interested readers are referred to an excellent recent survey by Moeslund et al. [14].

While the prevailing methods in marker-less mocap require a surrounding camera array, solutions using a single video camera have been explored [6, 22]. "However, they are not always robust, in part because image data is inherently noisy and in part because it is inherently ambiguous" [15, 19]. We believe the use of depth information can significantly reduce the ambiguity and make one-camera solutions practical. In this regard, our approach is mostly related to methods that use stereovision for motion capture [19, 2]. However these methods use relatively simple motion models and do not provide a quantitative study of their accuracy. Compared to stereovision (in particularly passive ones), an active depth sensor can provide more accurate and robust depth estimation.

Related to motion capture, SCAPE [1] and its extensions (e.g., [4, 3]) generate high quality human shape models with pose information. It uses a data-driven approach to find a low-dimensional parametric human shape and pose description, so the fitting of image data to 3D model can be simplified. In particular, it has been extended to estimate joint centers [16, 7] by using the semantic information in the training model. The main difficulty with SCAPE is that it needs a high-quality initialization shape (either from marker-based motion capture or visual hull from a surrounding camera array) and its quality depends on the training data. We demonstrate in this paper that our approach is still effective when using a sparse set of motion exemplars.

There are several new papers focusing on the problem of single camera motion capture. Fossati et al. [8] combine detection and tracking techniques to recover 3D motion using a single moving camera. They demonstrate the effectiveness of their approach with two types of motion, namely golf swings and walking. However, no body size variation is considered. Wei and Chai [24] combine motion capture with physically-based simulation to obtain high quality results from a single video sequence. Nonetheless, this technique requires manual labeling of key frames, while we seek a fully automatic approach. The work by Ganapathi et al. [9] is most closely related to our approach. They also use a single depth of camera (based on the principle of time-of-flight) to estimate full-body motion. A probabilistic method is used with a human body model as template in their work. While their algorithm can achieve real-time performance, the accuracy, reported as around 100mm, leaves something to be desired. Besides, their method does not handle body-size variations across subjects. We trade off real-time performance for high accuracy, achieving an average of 38mm using the data set provided in [9].

Just recently, Microsoft released the Kinect Sensor [13]. Performance evaluation of the underlying pose tracking algorithm was reported in [21]. PrimeSense (which provided the reference hardware design for the Kinect Sensor) has released an SDK that preforms pose tracking with Kinect input [20]. Different from these methods is our emphasis on accuracy, rather than real-time operation.

3. Overview

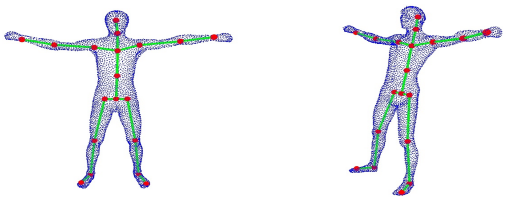

Figure 2. Two views of our mesh model and its underlying skeleton

In our method, a motion database is utilized, which is generated by driving a *generic* human mesh model with

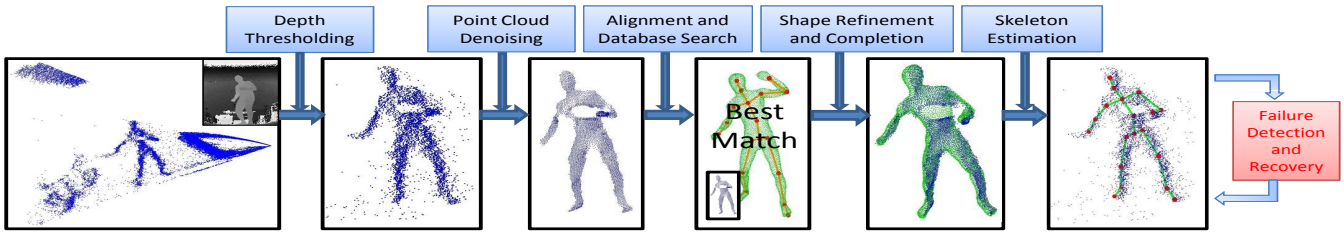

Figure 3. The outline of our processing pipeline. The leftmost image is a typical depth map, we define distance thresholds to cut the subject out. It goes through a number of processing stages, generating the estimated skeleton embedded in the input point cloud.

movements captured from an eight-camera optical motion capture system [23] operating at 120Hz. Around 19300 poses are recorded, including walking, running, bending, etc. The human model includes both a surface mesh and an embedded skeleton that contains 19 joints, as shown in Figure 2. The mesh model is animated with linear blending technique according to the recorded motions [18]. We denote a deformed mesh under a certain pose as $\mathcal{M}_l$. Four *synthesized* depth images, denoted as $\{\mathcal{P}_l^i\}_{i=1}^4$, are also rendered from four different perspectives. These depth images will be used for view-independent shape matching explained in Section 5.1.

The input to our approach is one or more depth images $\{\mathcal{X}_j\}_{j=1}^N$, or equivalently point clouds, from a single depth sensor (we will use these two terms interchangeably in this paper). Our goal is to estimate the configuration $\hat{\Upsilon}_j$ given a depth image $\mathcal{X}_j$ based on our motion database $\{\mathcal{M}_l, \mathcal{P}_l^1, \mathcal{P}_l^2, \mathcal{P}_l^3, \mathcal{P}_l^4, \Upsilon_l\}$.

Figure 3 shows the outline of our processing pipeline. Given a point cloud, we first remove irrelevant objects based on distance information, for which we use two fixed distance thresholds representing the interested distance range throughout our test. A modified surface reconstruction algorithm is applied to remove noise. Then the cleaned point cloud is transformed into a canonical coordinate frame in order to remove viewpoint dependency, and a similar pose is identified in our motion database. Then a refined pose configuration is estimated through non-rigid registration between the input and the rendered depth map for the corresponding pose. We rely on database exemplars and a shape completion method to deal with large occlusions, i.e., missing body parts. Finally a failure detection and recovery mechanism is adopted to handle occasional failures from previous steps, using the temporal information.

4. Point Cloud Segmentation and Denoising

The input depth map first needs to be processed to remove background and noise. Given the depth information, background objects can be easily removed by defining a bounding box or through background subtraction. However, the noise level in the depth map from typical ToF sensor is quite significant as shown in Figure 4(a). This is most likely due to the long range (for full body capture) between the subject and the camera. Since our subsequent processing requires finding point correspondences between the input point cloud and database exemplar, we need to overcome this obstacle of noisy input. Here we modify a surface reconstruction algorithm–*Locally Optimal Projection* (LOP) [12] for denoising.

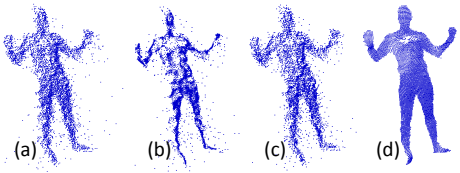

Figure 4. Comparison of original LOP and our modified LOP applied on a point cloud from depth sensor. from left to right (a) the input; (b) LOP with support radius of $0.2m$; (c) LOP with support radius of $0.1m$; (d) with our modified LOP.

LOP is a parameterization-free operator that, given a target point-set $\mathcal{P} = \{p_i\}_{i\in I}$, projects an arbitrary point-set $\mathcal{X} = \{x_j\}_{j\in J}$ onto the data $\mathcal{P}$, to reconstruct the underlying geometry structure in the data $\mathcal{P}$. The criteria is to minimize the sum of weighted distances between the projected point-set and $\mathcal{P}$, meanwhile preventing points in the projected point-set being too close to each other.

Applying LOP directly to our point cloud is problematic though. The amount of smoothness is controlled by a radius value h, which determines how big a neighborhood a projected point can contribute to the objective distance function. As shown in Figure 4, if h is too large, it leads to obvious shrinkage of the point cloud; if h is too small, there is little effect of denoising.

We thereby seek a solution that offers both smoothness and the preservation of geometric structure. With a closer look into applying this method to the depth map, we realize that shrinkage could be avoided by projecting only z coordinates of the point-set, which contains the most important depth information; while x and y can then be calculated through re-projection.

Therefore, given a point-set $\mathcal{P} = \{p_i\}_{i\in I}$, where $p_i = [x_i, y_i, z_i] \in R^3$ are the 3D coordinates, following the original derivation in [12], our modified LOP algorithm is initialized as follows:

$$z_i^{(1)} = \frac{\sum_{s\in I} z_s \theta(\|p_s - p_i\|)}{\sum_{s\in I} \theta(\|p_s - p_i\|)} \quad (1)$$

$$x_i^{(1)} = z_i^{(1)} \cdot x_i/z_i,\ y_i^{(1)} = z_i^{(1)} \cdot y_i/z_i \quad (2)$$

Figure 5. Comparison of point cloud smoothing using bilateral filtering and our modified LOP algorithm. From left to right: input; after Bilateral Filtering; after our modified LOP.

where $\theta(\cdot)$ is a fast decreasing function controlled by h, $\theta(r) = e^{-16r^2/h^2}$. Then for each iteration $k = 1, 2, \dots K$, the point is updated as

$$z_i^{(k+1)} = \frac{\sum_{s\in I} \alpha_s^i z_s}{\sum_{s\in I} \alpha_s^i} + \mu \frac{\sum_{s\in I\setminus\{i\}} \|p_i^{(k)} - p_s^{(k)}\| \beta_s^i}{\sum_{s\in I\setminus\{i\}} \beta_s^i} \quad (3)$$

$$x_i^{(k+1)} = z_i^{(k+1)} \cdot x_i/z_i,\ y_i^{(k+1)} = z_i^{(k+1)} \cdot y_i/z_i \quad (4)$$

with

$$\alpha_s^i = \frac{\theta(\|p_s^{(k)} - p_i\|)}{\|p_s^{(k)} - p_i\|},\ \beta_s^i = \frac{\theta(\|p_s^{(k)} - p_i^{(k)}\|)}{\|p_s^{(k)} - p_i^{(k)}\|} \left|\frac{\partial \eta}{\partial r}(\|p_s^{(k)} - p_i^{(k)}\|)\right| \quad (5)$$

where $\eta(r) = 1/3r^3$ and $\mu \in [0, 1/2)$ is a repulsion parameter that leverages between smoothness and surface geometry accuracy. In practice we found that setting $h = 0.5$ and $\mu = 0.35$ usually obtains the best results within $K = 5$ iterations. After applying LOP, we also remove isolated outliers since these points do not have effective supporting neighborhood and remain unchanged after projection. These points are identified by thresholding distance to their nearest point. The effectiveness of our denoising scheme is shown in Figure 5, in which we also compare it with bilateral filtering (BF) on the depth map. BF generates undesirable points connecting disjoint parts, probably due to the low-resolution of the depth map and too many stray points on occlusion boundaries.

5. Model Based Motion Estimation

After the depth map has been segmented and cleaned, our next step is to search for a similar pose in the motion database. Directly measuring similarity between the complete mesh model and a depth map is difficult, since a depth map is incomplete (at least 50% of a subject's information is missing) and there is no prior knowledge about from what viewpoint a depth map is captured.

Our solution to this search problem involves two steps. First we generate several synthesized depth maps from representative viewing directions, and align the input point cloud to these representative views, in this way removing the **viewpoint dependency** of the input point cloud. Then we apply dimension reduction techniques to find the most similar depth map (and body configuration) efficiently.

5.1. Point Cloud Alignment

We address this viewpoint dependency problem with the observation that principal axes of a point cloud provide robust characteristics of a pose, through which a transformation can be constructed and applied on the point cloud to rectify it to a canonical view.

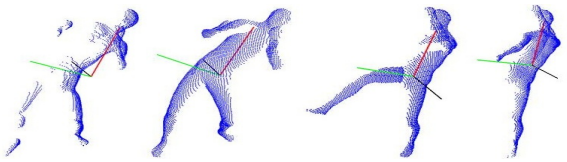

Figure 6. Visualization of the three principal axes (color-coded) of different point clouds. They are quite stable across different viewpoints. Two different poses are shown. (The right-most one is rendered from back-view with the entire right leg occluded.)

More specifically, given a point cloud $\mathcal{X}$ of dimension $N \times 3$, the principal axes are the eigenvectors of the 3×3 covariance matrix that represents the three major directions the point cloud spans. As we can see in Figure 6, they generally provide sufficient match across different view points for our purpose; while our neighbor search and non-rigid registration approach described in the following sections can deal with remaining small misalignment. Therefore we can define a local coordinate frame based on the mean value of the point cloud and the three principle axes. Note that we do not know the positive direction of the principal axes, therefore we pick the largest principle axis and define its positive ("up") direction as the up-direction of the camera coordinate–assuming the camera is usually not up-side-down. Using the point cloud's mean value as the origin, we define four canonical coordinate frames by alternating the positive directions of the remaining two axes. For each canonical coordinate frame, we define a virtual depth camera that is away from the origin and looking into the depth direction. Each mesh model $\mathcal{M}_l$ is transformed into its own canonical coordinate frames and four synthesize depth maps are rendered, denoted as $\{P_l^i\}_{i=1}^4$.

For an input point cloud $\mathcal{X}$, we also compute such a transformation T but with the difference that we just pick a random positive direction for the remaining two axes. The transformed point cloud $\mathcal{X}^c = T(\mathcal{X})$ is in a view-independent canonical coordinate frame.

5.2. Nearest-Neighbor Search in Low-Dimensional Subspace

In $\mathcal{X}^c$ the viewpoint dependency is mostly removed. We can now search the synthesized depth maps to find the most similar pose. One could search in 3D space by comparing the point distances in 3D, we instead search in the PCA space of the image space and look for a P_l^i that is closest to $\mathcal{X}^c$ in the PCA space. More specifically, all synthesized depth maps $\{\mathcal{P}_l^i\}$ are vectorized and stacked into a matrix from which a PCA subspace and corresponding coefficient vectors $\{\lambda_l^i\}$ are learned.

$\mathcal{X}^c$ is re-synthesized as a standard depth image in the canonical view and vectorized to calculate PCA coefficients

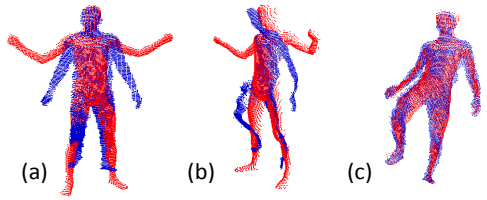

Figure 7. Nonrigid registration between a test point cloud (blue) and one of our database exemplars (red) using CPD. (a) and (b) show different views of these two point clouds after global alignment; (c) is registration result. Notice both body shape and pose differences.

in this subspace. Finally, a most similar pose is identified by finding P_l^i closest to $\mathcal{X}^c$ in the PCA space, i.e. finding $< l, i >$ that satisfies

$$< l, i >= \underset{1 \le l \le N_c, 1 \le i \le 4}{\arg\min} \{\|\lambda_l^i - \gamma\|\} \quad (6)$$

Up to now we have obtained a point cloud $\mathcal{P}_l^i$ and its corresponding surface model $\mathcal{M}_l$ that are in similar pose as $\mathcal{X}^c$. Since $\mathcal{M}_l$ has a known joint configuration, an initial estimation of the joint configuration of $\mathcal{X}^c$ is obtained. Note everything is now defined in a canonical coordinate frame, we need to apply the inverse transform T^{-1} to $\mathcal{M}_l$ and joint configurations so that they are in the input coordinate frame. Such transformation facilitates the refinement process discussed in the next section. For the sake of simplicity in notation, we assume $\mathcal{M}_l, \mathcal{P}_l^i$ are defined in the input coordinate frame from this point on.

6. Pose Refinement

Although a similar pose has been identified from our motion database for the input point cloud, naively outputting that pose as the final result yields poor results due to personal shape variations and small pose differences. We further refine the pose through non-rigid registration between $\mathcal{X}$ and $\mathcal{M}_l$. Such a registration should maintain consistent movement of nearby points, so that the geometry of body shape is preserved. We adopt the Coherent Drift Point (CPD) algorithm [17] to establish point correspondences between $\mathcal{P}_l^i$ and $\mathcal{X}$. The global alignment discussed above ensures correct initializations. Since $\mathcal{P}_l^i$ is rendered from $\mathcal{M}_l$, the point correspondences between $\mathcal{M}_l$ and $\mathcal{X}$ are also established. Note that we cannot use CPD to register $\mathcal{M}_l$ and $\mathcal{X}$ directly since its formulation assumes roughly one-to-one point correspondence that is not valid given the incompleteness of $\mathcal{X}$.

One example of the registration results is shown in Figure 7. (Note that this example is artificially chosen for illustration purpose, i.e. this database exemplar is not the identified neighbor pose for this input.) Joint positions can be extracted from the deformed point cloud, but only for the visible part. We would like to provide a reasonable estimate for the occluded part as well. We therefore perform a shape completion, by deforming the complete mesh $\mathcal{M}_l$ to fit to the input $\mathcal{X}$. Here we apply the method in [11] that maintains point correspondences between $\mathcal{X}$ and $\mathcal{M}_l$, and at the same time preserves geometry structure of $\mathcal{M}_l$ through Laplacian coordinates. This is formulated as solving the following least-squares system:

$$\hat{\mathcal{M}}_l = \underset{\mathcal{M}}{\arg\min} \{\|L\mathcal{M} - L\mathcal{M}_l\|_2^2 + w \sum_{k \in I} \|m_{l,k} - x_k\|_2^2\} \quad (7)$$

Here L is the cotangent Laplacian matrix; I is set of indices of the points in $\mathcal{M}_l$ that have correspondences with $\mathcal{X}$; $m_{l,k}$ and x_k are the corresponding points. The first term of Equation 7 aims to preserve the geometric properties of $\mathcal{M}_l$ after deformation, while the second term enforces point correspondences obtained from CPD registration. w is a user defined weighting term, we typically set it to 1.

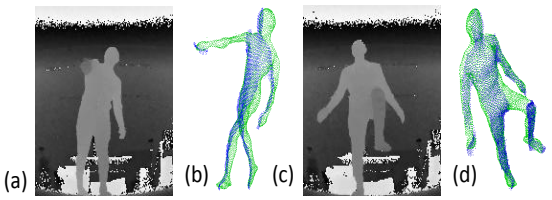

Figure 8. Two examples showing our approximation of the geometry using the mesh completion method for input point clouds with missing body parts.(a) and (c) are input depth maps.(Here we use depth map to better illustrate the pose). In (b) and (d), blue points are input point clouds and green points are our estimated mesh models.

We use two examples in Figure 8 to show the results of this shape completion method and illustrate how missing body parts in the partial observations are naturally handled.

With the deformed mesh model $\hat{\mathcal{M}}_l$ in hand, we can proceed to estimate the desired joint positions. We assume a joint center can be determined by a set of mesh vertices around it. A set of control vertices $\{q_i^k | q_i^k \in \mathcal{M}_l; i \in \{1, 2, \cdots, n\}\}$ are pre-defined for each joint $\mathcal{J}_k$. Then a transformation T_k is computed between this vertex set and its counterpart $\{\hat{q}_i^k\}$ in deformed mesh $\hat{\mathcal{M}}_l$ using ICP [5], such that

$$\hat{q}_i^k \approx T_k(q_i^k), i \in \{1, 2, \cdots, n\} \quad (8)$$

Finally the joint location is computed as $\hat{\mathcal{J}}_k = T_k \mathcal{J}_k$. Since we have a complete mesh, all joint locations can be calculated and they are the final body configuration output for this frame. If the input is a sequence, we can also further apply low-pass filtering to the joint locations to smooth out the trajectory–something commonly done as a postprocessing step for mocap.

Failure Detection and Recovery: Though our approach performs well in most cases, a few failure cases still exist. Such a case generally happens when there is no similar pose in our database for which reasonable point correspondence can be estimated using CPD. In order to handle such cases, a failure detection and recovery mechanism is embedded in our pipeline. It is observed that in case of failure, in general either the length of adjacent joints undergoes large sudden changes or the configuration of joint angles violates

practical limits. Therefore, by checking for the occurrence of either one, a failure case can be identified. In terms of recovery, since no assumption is made about motion model, we rely on temporal consistency. When a failure is detected, a non-rigid registration is performed between $\mathcal{X}^{t-1}$ and $\mathcal{X}^{t}$, which are inputs of frame $t-1$ and t respectively. After that a similar shape completion procedure is preformed to compute a complete mesh $\hat{\mathcal{M}}_l^t$ from that in previous frame $\hat{\mathcal{M}}_l^{t-1}$ and the registered point cloud $\mathcal{X}^{t-1}$. Subsequently joint locations can be estimated accordingly.

Since we rely on database exemplars to separately estimate motions for each frame and temporal registration is only required when failure is detected, our approach does not suffer from drift error as in typical temporal tracking methods for long video sequences. Moreover we do not require piece-wise rigid assumption on the model, instead we directly deform the entire model for estimation.

7. Experiments and Results

Our system is implemented in Matlab and evaluated mainly based on the public dataset provided by Stanford [9]. This dataset consists of 28 sequences of motions, of which about half contain 100 frames and the others have 400 frames, all recorded at 25fps with a depth camera of resolution 176×144. The motions range from simple movements such as lifting a hand to a very challenging tennis swing with severe occlusion and simultaneous movements of several body parts. Locations of 3D markers attached to the subject's body measured using a commercial active marker motion capture system are provided as ground truth. In order to compare with this ground truth, we choose a set of patches on our mesh model to approximate the position of markers according to their markers setup, as was done in [9]. Then estimation error is measured as

$$\bar{e} = \frac{1}{N_f}\sum_{k=1}^{N_f}\frac{1}{N_m}\sum_{i=1}^{N_m}\|m_i - \hat{m}_i\| \qquad (9)$$

where N_f and N_m are number of frames and markers respectively. m_i is measured ground truth of marker location and $\hat{m}_i$ is our estimation. Throughout our evaluation we use *the single motion database* described in the beginning of Section 3. Figure 9 shows our estimation error for all 28 test sequences, compared with the best result reported in [9], which combines Hill Climbing(HC) search and Evidence Propagation(EP). As we can see, our method achieves substantially higher accuracy, with a total mean error around $38mm$, compared to their $100mm$. It should be emphasized that we use a generic human model to generate our own database, no sequence from the test data is in our motion database; and we always use the entire database (e.g., the nearest neighbor search is global, instead of sequence specific).

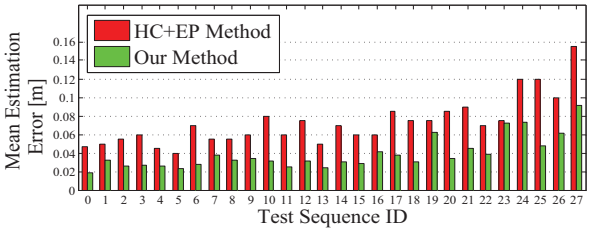

Figure 9. The mean estimation error of our approach on Stanford's public test dataset that consists of 28 sequences of motions, compared with the results reported in [9] (HC+EP)

Next we will show the effectiveness of various components of our pipeline. We pick two representative sequences from the Stanford dataset: sequence 21 is of moderate complexity that includes mainly hand and feet movement and sequence 27 is the most challenging tennis swing motion.

The Effectiveness of Denoising: The point cloud denoising procedure is important for our method to correctly estimate poses, due to the way it is designed. Simply applying this approach to an original point cloud with background objects removed gives unsatisfactory results as shown in Figure 10. The importance of our smoothing module arises from two reasons: in the presence of severe noise, neighbor search is erroneous and CPD is strongly perturbed.

The Effectiveness of Pose Refinement: Here we show both the effectiveness of our neighbor search approach and the necessity of pose refinement. In Figure 10 we see that directly representing the pose of an input point cloud by that of the identified neighbor sample results in higher estimation error. On the other hand, the error for sequence 21 is still acceptable, meaning that similar poses are in general correctly localized. Meanwhile, the reason we have larger errors for sequence 27 is that the pose of the neighbor exemplar cannot properly approximate the input, since our database might not contain such substantially similar poses. In general, pose refinement is required for accurate estimation.

Viewpoint Independency: The test sequences in Stanford's dataset were basically captured from the frontal view, and re-rendering from those partial point clouds with different viewpoints would result in even more incomplete data. To verify the issue of view independency, we test on three synthetic sequences rendered with our mesh model walk-

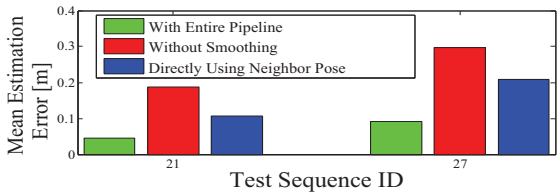

Figure 10. Comparison of estimation errors of three methods: using our entire pipeline, without smoothing and directly using neighbor pose without refinement. The results demonstrate the necessity of such processing components.

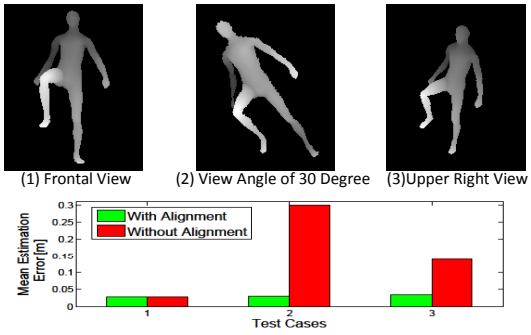

Figure 11. Viewpoint independency test. First row: examples of synthetic inputs (see text for explanation). Second row shows quantitative evaluations.

ing. Shown in the first row of Figure 11, the first sequence is captured from then normal frontal view; the second one with a roll angle of 30 degrees; and the last case with the camera looking from the upper right direction, tilting down. The quantitative results are shown in the second row of Figure 11. It can been seen easily what the detrimental effect of view-dependent input can be, if not handled.

Failure Detection: In this experiment we use the input from frame 301 to frame 400 of sequence 27 as an example. Among these test frames, for around 35 frames the subject remains in a relative static pose for which our database does not have a similar one. The difference is shown in Figure 12. Under this situation, our failure detection and recovery mechanism takes effect and re-estimates poses for input through temporal information. The comparison of results with and without such detection and recovery is shown in Figure 13. As we can see, failure poses were effectively recovered. However, notice that for a majority of the test sequence in this dataset, our regular pipeline can generate good results.

Database Dependency: In this test, we aim at quantitatively determining the relationship between our estimation result and the number of database samples. Originally our database contains around 19000 samples with the sampling rate of 120fps. We sub-sample it with different ratios up to 100, which corresponds to a minimum sampling rate of

Figure 12. An example of our failure case (left), for which the most similar pose in our database (right) exhibits a very large difference in pose.

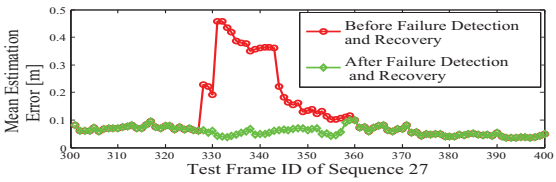

Figure 13. Estimation errors, with and without failure detection and recovery.

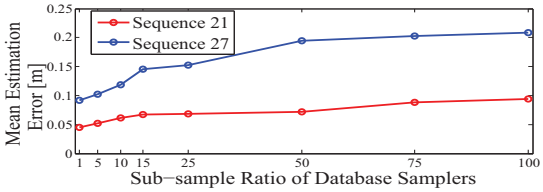

Figure 14. Estimation errors of our approach with a set of motion samples which is sub-sampled from our original database, with different ratios.

1fps, and show the corresponding estimation errors in Figure 14. For sequence 21, only a small set of samples are sufficient. On the other hand, for the complicated movements in sequence 27, denser samples are required. Overall our method is quite insensitive to sampling rate.

Qualitative Evaluation using Kinect We have performed a qualitative comparison using the pose tracking algorithm in [20]. The primary reason that we didn't compare quantitatively is the lack of ground-truth data. The markers used in a typical optical motion capture systems (which are considered as golden standard) interferes with the Kinect Sensor. Nevertheless even just visual inspection can clearly demonstrate the improved accuracy in our method. The deficiencies visually identified in [20] include (1) joint positions are not consistent with input depth maps when the subject is moving; (2) joint centers are unrealistically located on the surface, especially for the arms; and (3) catastrophic failure in simple motion (such as a crouch). A visual side-by-side comparison is presented in the supplementary video.

Quantitative Comparison with [21]: For the purpose of quantitative comparison between our method and the underlying algorithm of Kinect [21], mean average precision (mAP) [21] is calculated based on the public dataset [9]. As reported in their paper, the mAP is around 0.9 with the true positive distance threshold set to 0.1m. Under the same condition, our method achieves higher mAP, which is 0.95. In addition, they reported the effect of threshold on true positives in mAP using synthetic data. We also carried out similar experiments, however based on the real dataset [9], which is much noisier. The comparisons shown in table 1 further demonstrate our higher accuracy (the numbers below for [21] are estimated from their Fig. 4):

Body-size Invariance: Our method is capable of handling large body-size variations across subjects. Notice that the subject in the test above ($\approx 1.7m$) is different from both the subject in the public dataset and our template model. In order to further demonstrate this capability, we perform tests on a higher subject ($\approx 1.9m$) and a child ($\approx 1.2m$). The results are shown in our supplemental video.

In summary, our pipeline performs well based on the

Threshold (m)	0.02	0.03	0.05	0.07	0.1	0.15	0.20
mAP: [21]	0.02	0.06	0.30	0.53	0.73	0.82	0.85
mAP: Ours	0.36	0.57	0.79	0.89	0.95	0.98	0.99

Table 1. Comparison of mAP with [21] on public dataset [9]

tests in the public dataset. **Some of the results would be shown in supplemental video by embedding our estimated skeleton with the input point clouds** [25]. As a tradeoff for the high accuracy, the computational time is currently non-real time, mainly due to the non-rigid registration process. With our implementation in Matlab, the running time for each frame is between 60s and 150s, depending on the number of 3D points and pose differences.

8. Conclusion

In this paper we present an effective pipeline that achieves highly accurate and robust pose estimation from a single depth image. The key insight is to combine data-driven pose detection with pose refinement. By using a prior database, we not only reduce the possible joint configuration space, but also provide an effective way to fill in the unobservable parts. Our pose refinement scheme can accommodate both pose difference and body-size difference. In addition we carefully design our pose detection algorithm to be view-independent. All these together dramatically reduce the size of the motion database – we only need motion samples synthesized from one generic human model. Quantitative evaluation shows that we achieve more than two times better accuracy than previous state-of-the-art (38mm vs. 100mm).

Looking into the future we would like to spend more time accelerating the computation. We believe some of the operations can be effectively accelerated through GPU. We also plan to invest on non-optic mocap systems for more quantitative comparison with Kinect. There are more challenging mocap cases such as the interferences from loose clothes, multiple interacting persons, etc. With all these improvements, a single-camera mocap solution will find its application in many new frontiers.

Acknowledgement This work is supported in part by University of Kentucky Research Foundation, US National Science Foundation award IIS-0448185, CPA-0811647, MRI-0923131, Microsoft's ETH-EPFL Innovation Cluster for Embedded Software (ICES), as well as the EC's FP7 European Research Council grant 4DVIDEO (n° 210806).

References

[1] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24, 2005. 2

[2] P. Azad, A. Ude, T. Asfour, and R. Dillmann. Stereo-based markerless human motion capture for humanoid robot systems. In *ICRA*, 2007. 2

[3] A. Balan and M. J. Black. The naked truth: Estimating body shape under clothing. In *ECCV*, 2008. 2

[4] A. Balan, L. Sigal, M. Black, J. Davis, and H. Haussecker. Detailed human shape and pose from images. In *CVPR*, 2007. 2

[5] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE PAMI*, 1992. 5

[6] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR*, 1998. 2

[7] S. Corazza, E. Gambaretto, L. Mundermann, and T. Andriacchi. Automatic generation of a subject specific model for accurate markerless motion capture and biomechanical applications. *IEEE Trans. Biomedical Engineering*, 2008. 2

[8] A. Fossati, M. Dimitrijevic, V. Lepetit, and P. Fua. From canonical poses to 3-d motion capture using a single camera. *IEEE PAMI*, 2010. 2

[9] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *CVPR*, 2010. 2, 6, 7

[10] A. Kolb, E. Barth, R. Koch, and R. Larsen. Time-of-Flight Sensors in Computer Graphics. *Eurographics State of the Art Reports*, 2009. 1

[11] M. Liao, Q. Zhang, H. Wang, R. Yang, and M. Gong. Modeling deformable objects from a single depth camera. In *ICCV*, 2009. 5

[12] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.*, 2007. 3

[13] Microsoft. Kinect camera. `http://www.xbox.com/en-US/kinect/default.htm`, 2010. 1, 2

[14] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 2006. 1, 2

[15] D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *CVPR*, 1998. 2

[16] L. Mundermann, S. Corazza, and T. Andriacchi. Accurately measuring human movement using articulated icp with soft-joint constraints and a repository of articulated models. In *CVPR*, 2007. 2

[17] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE PAMI*, 2010. 5

[18] M. Pharr and R. Fernando. *GPU Gems 2: Programming techniques for high-performance graphics and general purpose computataion*. 2005. 3

[19] R. Plankers and P. Fua. Articulated soft objects for multiview shape and motion capture. *IEEE PAMI*, 2003. 2

[20] Primesense. OpenNI. `http://www.openni.org/`. 1, 2, 7

[21] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011. 2, 7

[22] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. In *CVPR*, 2001. 2

[23] Vicon. Optical Motion Capture System. `http://www.vicon.com/`. 3

[24] X. Wei and J. Chai. Videomocap: Modeling physically realistic human motion from monocular video sequences. In *SIGGRAPH*, 2010. 2

[25] M. Ye and R. Yang. Project website. `http://vis.uky.edu/~gravity/Research/Mocap/Mocap.htm`, 2011. 8