Tracking of Deformable Human Avatars through Fusion of Low-Dimensional 2D and 3D Kinematic Models

Ningjian Zhou
S. Shankar Sastry

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-87
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-87.html

May 19, 2019

Tracking of Deformable Human Avatars through Fusion of Low-Dimensional 2D and 3D Kinematic Models

Bill Zhou
UC Berkeley
billzhou@berkeley.edu

Abstract

We propose a method to estimate and track the 3D posture as well as the 3D shape of the human body from a single RGB-D image. We estimate the full 3D mesh of the body and show that 2D joint positions greatly improve 3D estimation and tracking accuracy. The problem is inherently very challenging because due to the complexity of the human body, lighting, clothing, and occlusion. The solve the problem, we leverage a custom MobileNet implementation of OpenPose CNN to construct a 2D skeletal model of the human body. We then fit a low-dimensional deformable body model called SMPL to the observed point cloud using initialization from the 2D skeletal model. We do so by minimizing a cost function that penalizes the error between the estimated SMPL model points and the observed real-world point cloud. We further impose a pose prior define by the pre-trained mixture of Gaussian model to penalize out unlikely poses. We evaluated our method on the Cambridge-Imperial APE (Action Pose Estimation) dataset showing comparable results with non-real time solutions.

1. Introduction

The understanding 3D human pose is a fundamental problem human-computer interaction and a relatively unexplored area in computer vision. Unfortunately, the human body is a hugely complex system and generating a biologically accurate computer model is currently an intractable problem. In the field of digital modeling, we have three major ways of representing the human body: skeletal, articulated, and deformable. The skeletal model parameterized the body joints positions and bone length [14]. The articulated model builds on top of the skeletal model. It adds volume information to the body through a combination of geometric shapes (rectangular prism, frustums, spheres, etc.)[11]. The deformable model eschews geometric shapes and uses organic contours to represent the body surface. The deformable model is the most realistic and expressive

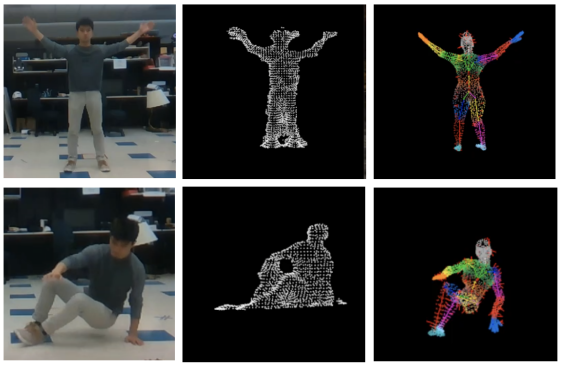

Figure 1. Model fitted to various observed posture. Left is the RGB camera input. Center is the observed depth camera input as a point cloud. Left is the estimated deformable model pose.

of the three models [29]. A fast and accurate methodology of recovering a deformable human model would enable applications such as virtual closets, photorealistic telepresence conferencing, and remote physiatry which require the doctor to observe minute body language cues during therapy.

This paper introduces an approach to track human avatars by fitting a deformable model in near real time. Fitting a deformable model can be a daunting task. Depending on the complexity of the chosen model, we may have to perform non-linear optimization over thousands of parameters. This requires significant runtime and can easily fail because of local minima. Further, we face ambiguities caused by the loss of 3D geometry due to self-occlusion. This ambiguity is compounded by the non-Gaussian noise characteristics of the depth camera. Our methodology poses the fitting as a convex optimization over the parameters of the deformable model.

We start with a low dimensional model of the human body. The model we chose is SMPL[13] which has 178 parameters and models most features of the body except for fingers and facial expressions. We feel that this is a reasonable compromise between realism and efficiency. Next, we

compute the 2D skeletal model and 3D head pose model inferred from the RGB image. Finally, we construct a convex optimization problem to fit the SMPL model to the observed data given constraints imposed by the 2D skeletal model.

2. Related Works

The recovery of 3D human pose from incomplete data is by nature an ambiguous problem. Over the past 10 years, different methods deal with this ambiguity in different ways. These include reducing the scope of the problem by only tracking the skeleton of the body. Other approaches discard the semantic of the human body by reconstructing the surface using fusion methods.

2.1. Skeletal Fitting

The first camp of approach is skeletal fitting. OpenPose[5][9] proposes a deep convolutional neural network that estimates the joint positions of multiple people in the image[16]. Microsoft Kinect proposes a random forest classifier on 4D HoG features to estimate the 3D position of human joints[20]. The drawback with this class of approach is that a skeleton by itself is not enough to drive a fully deformable model. The body surface contours are left unmodeled.

2.2. Fusion

The second group of approaches is fusion where human tracking is treated as a 3D reconstruction problem. DynamicFusion[18] and DoubleFusion[24] leverage multiple camera views to construct a full point cloud of the human body. Fusion is very accurate and is capable of producing real-time scans. However, to do real-time fusion, we need camera views from all perspective which often require a calibrated stationary system. Further, we lose the semantic understanding of the human body. For instance, there is no way for the computer to tell which part of the body is the hand versus the torso.

2.3. Parameter Fitting

The third group of approaches is parameter fitting on a deformable kinematic model. The current literature on the subject is mostly geared towards animations[21][6]. Methods proposed intakes a single RGB image and attempt to fit the highly parameterized model[12]. Guan et al[17]. take manually marked 2D joints and first estimates skeletal joint positions. Then they use this estimate to pose a SCAPE model. In similar work, Hasler et al[25]. fit the SCAPE model to human silhouettes[1] with a known segmentation of the human and a few manually labeled correspondences. While these methods are highly accurate, none are scalable to tracking situations.

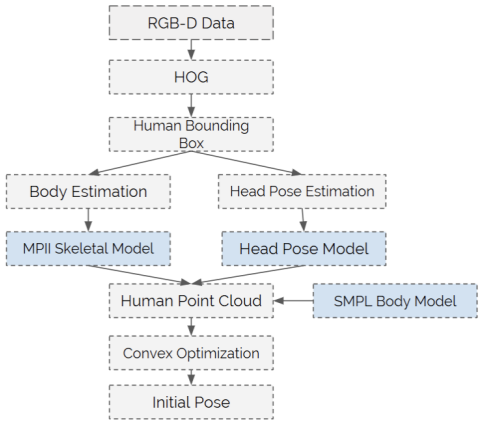

Figure 2. System pipeline that takes in a single view RGB-D and fits an initial pose to the observed data. In this pipeline, three models (MPII Skeletal Model, Head Pose Model, and SMPL Body Model) are built and fused via a convex optimization.

3. Method Overview

Figure 2 shows an overview of our system. We acquire time and spatially synchronized depth and RGB frames from our sensor. We begin with a preprocessing step that segments the human body. We remove the ground plane as well as objects not in direct contact with the human. From the segmented human, we leverage the OpenPose CNN[9] to construct a 2D skeletal model of the body. In the next stage, we construct a convex optimization problem $C(\theta, \beta)$ where is the pose of the joints and are coefficients of a low-dimensional shape space, learned from a set of thousands of labeled body scans. We define $C(\theta, \beta)$ to be how closely the body model parameterized by θ and β matches the observed human body. The objective is to minimize C. To solve this problem we rely on a reasonably good estimate of the 2D joint positions. We observed that the 2D skeletal model initialization significantly improved accuracy and convergence speed.

4. Data Processing

We used the Intel Realsense D435 to acquire data. We chose the Realsense D435 over the popular Microsoft Kinect for three reasons. First, it is being actively developed by Intel versus the now discontinued Kinect line which means more developer will likely have access to the Realsense D435 in the future. Secondly, the D435 combines a short-range structure light sensor and a long-range time of flight sensor allow it to achieve an impressive operating range of 0.2 meters to 10 meters. Lastly, it is light, portable, and does not require an external power source which means it can be deployed in a variety of environments. The raw input data from the D435 first undergo a 3-step preprocessing

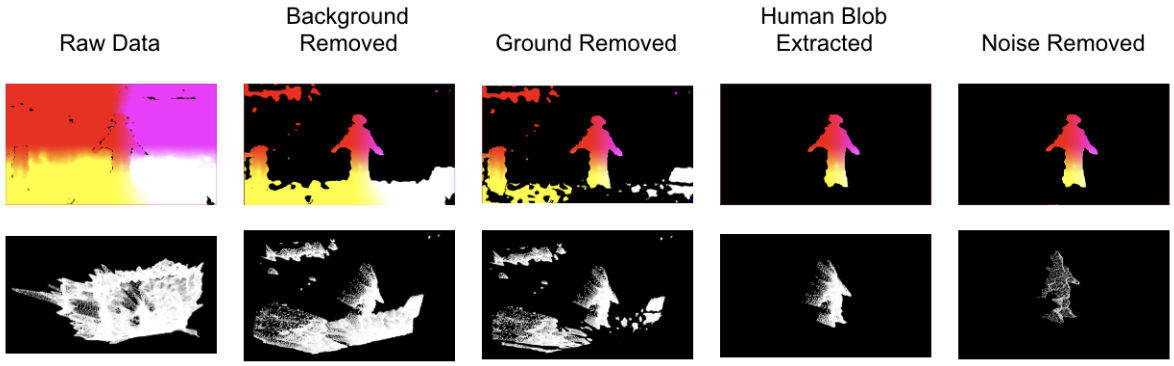

Figure 3. The raw image is acquired from the depth camera the human point cloud is extracted through the preprocessing procedure. First, the background and ground plane are removed. Next, small blobs are the human are removed. Finally, the data is downsampled, averaged, and outlier points are removed.

stage.

4.1. Background Removal

We use a Histogram of Oriented Gradients (HoG)[15] classifier to compute a 2D bounding box around the human on the RGB image. We project the bounding box on the depth image and computed the median depth of the points inside the frustum. That computed depth is believed to be the distance the human is away from the camera. Subsequently, we remove all points with depth outside 1 meter radius from the body center.

4.2. Ground Removal

Next, we remove the floor by fitting a plane to the ground via RANSAC method[8]. We take extra precaution in this step since the human body can be misinterpreted as a plane in certain poses. We begin by removing all the points in the human bounding box. Next, we fit a plane model to the remaining points. Subsequently, we remove all points belongs to the plane within a threshold of 0.1 meters.

4.3. Noise Removal

The Realsense D435 has higher noise around edges of objects. This causes a lot of points at the edge of the human body to be especially erroneous. We apply a statistical outlier removal method[7]. For each point, we compute the mean distance from its 30 nearest neighbors. The nearest neighbors are determined through a KD-tree search. By assuming that the resulted distribution is Gaussian with a mean and a standard deviation, all points whose mean distances are outside an interval defined by the global distances mean and standard deviation can be considered as outliers and trimmed from the dataset. This noise removal method significantly improves the quality of our human point cloud.

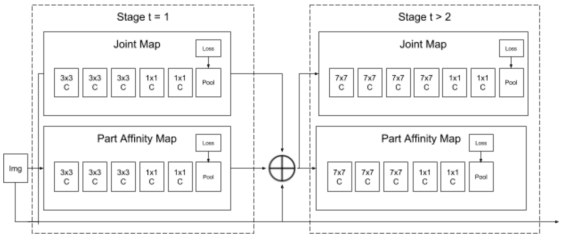

Figure 4. Two stage neural network proposed by OpenPose. Modification are made to the part affinity map to use less convolutional layers since our tracking is constrained to a single person.

5. Model Construction

5.1. 2D Skeletal Model

We adopt the neural network from OpenPose to train a model to compute 2D joint positions from a single RGB image. The convolutional neural network consists of two branches. The top branch produces a probability heat map of joint locations. Each pixel on the heatmap represents the likelihood of a joint at that pixel. The bottom branch produces a part affinity map (PaF)[9]. The PaF predicts which joints are associated with each other. The joint mapping and the PaF are concatenated and interpolated on top of the input data and sent through the next stage which is the two branches repeated.

We modified the original OpenPose implementation to use depth-wise separable convolution methods as described in MobileNet[3]. Our implementation of OpenPose in MobileNet reduced the number of trainable parameters by 89%. We further reduced the footprint of the network by using 15 PaF unit rather than 20 PaF units.

We trained the model on the MPII human joint dataset containing 25K human image performing 410 classes of activities. We chose to train the network on the

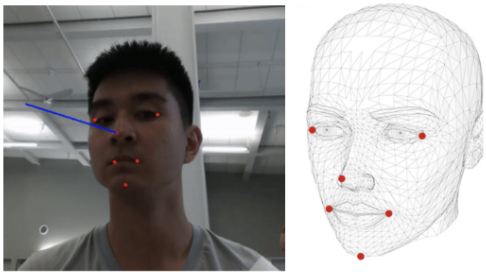

Figure 5. Positions of 6 corresponding facial landmarks are computed both on the RGB and synthetic face mesh. We solve 2D-3D correspondence problem[28] to recover the head direction from the RGB view,

MPII dataset[12] rather than the more common COCO dataset[26] for two reasons. First, MPII contains fewer joints which meant we could use a simpler neural network architecture. Second, the MPII joint annotations are closely matched with the SMPL model. This meant we use the MPII model to initialize the SMPL model.Our formulation greatly reduced the footprint of OpenPose while maintaining acceptable accuracy.

5.2. 3D Face Modeling

The depth data from the Intel Realsense D435 is too noisy to distinguish facial features. As a result, we decided to infer the head direction from the RGB image which contains clear facial features. We use the pre-trained Linear Binary Features (LBF)[19] classifier available in OpenCV[4] to track the 2D locations of 6 facial features (right eye right corner, left eye left corner, nose, mouth left corner, mouth right corner) as shown in Figure 5. We pick the same 6 features on a 3D model of a generic human face and record their 3D coordinates. We solve the 2D-3D correspondence problem[28] to recover the pose of the face in the RGB image.

5.3. 3D Body Modeling

We chose the SMPL model[13] to represent the deformable human body. SMPL provides a good compromise between efficiency and realism. It contains 24 joint positions and 6890 surface points. Each surface point position is computed through a linear combination of its 4 nearest joint positions. In other words, by moving the joints, we can drive the surface points movement. Additionally, it contains 80 shape keys that control the deformation of various parts of the body. We chose 10 shape keys in our model to reduce computation cost.

6. Model Fusion

The 2D and 3D body models are fitted through the convex optimization problem.

$$C(\theta, \beta) = \lambda_s E_s(\theta, \beta) + \lambda_J E_J(\theta) + \lambda_p E_p(\theta)$$

θ is a vector of bone parameters. Naively, each bone should have 9 degrees of freedom (3-axis position, 3-axis rotation, and 3-axis scale). In our formulate we reduced each bone to 3 degrees of freedom by making the following two observations. First, the relative position of each bone can be related to the global positioning of the body center. After all, bones in the body have to stay in their relative arrangements. Secondly, we found that the scale can be determined by the body shape coefficient that controls overall body scale. We observed that in different height humans, bone length tend to change proportionally. By representing the bones only by their rotation quaternion, we reduced the number parameters to optimize from 240 to 144.

β which is a vector of 80 floating decimal shape keys. However, we observed that the majority of shape keys only marginally impact the shape of the body. Thus, we chose the top 10 keys that have the most impact on the body contour to be included in the convex optimization. We used the Google's Ceres Solver as our convex optimizer[2].

6.1. ICP Error

The ICP error measure how closely the SMPL model matches the observed depth data.

$$E_s(\theta, \beta) = \Sigma_i ||p_i - S(p_i; \theta, \beta)||^2$$

So here p_i is the observed point on the sensor, the function $S(pi; \theta, \beta)$ returns the closest point on the SMPL model given a pi and the pose parameters (θ and β). The ICP error is the sum of the squared distances.

6.2. Joint Prior

The Joint Prior measures how far the SMPL joints are away from the neural net joint positions.

$$E_j(\theta) = \Sigma_i ||\hat{J}_i - J_i(\theta)||^2$$

The $\hat{J}_i$ represents the pose of the ith joint as predicted by the neural network. The function $J_i(\theta)$ returns the pose of the ith joint in the SMPL model. We sum over the squared errors over 24 joints in the pose.

6.3. Pose Prior

The goal of the pose prior is to provide a measure of how likely is pose is done by a real human. It is possible to use joint-angle limits to verify whether two connected bones are valid or not. However, the problem becomes complex as there are dependencies in joint-angle limits between certain

Figure 6. SMPL model[13] parameterized to fit various poses and body shapes. The first three shows the skeletal of the body driving the surface point movements. The last two shows the shape keys affecting the surface contours of the body.

pair of bones. For example, how much a person can flex their elbow depends their arm is in front or behind them. Thus constructing a rule based reverse kinematic model is unfeasible.

$$E_p(\theta) = min(-log(c_i N_i(\theta; u_{\theta,i}, \Sigma_i))$$

We used a pretrained pose prior from the CMU MoCap Dataset[10] to enforce the realism of human poses. Each distribution $N_i(\theta; u_{\theta,i}, \Sigma_i)$ gives a likelihood of a pose parameterized by θ and β. c_i is the weight assigned to the ith distribution and is determined during training.

7. Frame to Frame Tracking

Our tracking is relatively simple. For every new frame, we initialize our convex optimization from the previous frame. We only optimize for joint positions and ignore body shape during tracking. During the tracking stage, we add an extra loss term that exponentially penalizes poses that are dissimilar to the previous pose.

$$C(\theta, \beta) = \lambda_s E_s(\theta, \beta) + \lambda_J E_J(\theta) + \lambda_p E_p(\theta) + \lambda_n e^{||\theta_t - \theta_{t-1}||}$$

Although the human body pose within 1/30th second intervals remains relatively consistent, random noise from the depth sensor may introduce rapid changes to the body contour.

8. Experiments and Results

8.1. Dataset

We evaluated on the Cambridge-Imperial APE (Action Pose Estimation) dataset[23]. The APE dataset contains 245 sequences from 7 subjects performing 7 different categories of actions. Videos of each subject were recorded in different unconstrained environments, changing camera poses and moving background objects. The dataset contains fully 3D point clouds of human subjects and labeled 3D position for 20 joints. We evaluated the accuracy of 12 joints (list out what the joints are) since they matched up with the SMPL model. We computed the cumulative the drift of the 12 joints between our predicted 3D joint positions and the ground truth provided by APE[23]. This gives us a per frame error. We computed the average per frame error over 4 sequences and reported them below.

Method	Wave 1	Wave 2	Bend	Balance
Ours	32.2	35.8	68.1	45.2
Eichner et. al[22]	30.1	34.6	58.8	37.8
Yang et. al[30]	33.2	40.4	56.7	34.4
Tsz-Ho et al[27]	35.6	37.1	44.5	43.5

Table 1. Comparison between our method and state of the art fitting methods designed for animation and none-realtime fitting. Our accuracy is competitive with non realtime methods

8.2. Evaluation

The algorithm performed relatively well on stationary poses sequences such as waving with one hand and waving with two hands. The average drift is between 60-70 millimeters which is not noticeable by the eye. However, for moving poses such as balancing on one leg and bending the body, the drift becomes more apparent.

These results represent the accuracy of our algorithm on a near perfect point cloud. The APE dataset differ from observations made by the RealSense D435 in three ways. First, the RealSense provides one camera view of the point cloud so, in practice, half of the human would be missing due to self-occlusion. Conversely, the APE dataset is captured by multiple depth cameras producing complete point cloud. Secondly, the APE dataset has 20 times the resolution compared to the single D435 camera. The APE dataset captures distinguishable features on the D435 such as feet direction and hands. Finally, the APE dataset only label 3D joint locations. Currently, we have no way of evaluating body surface tracking accuracy.

We compared the results to the state of the art approaches used in animation. While our performance is not much worse in simple movements, error started accumulating in the more complex poses.

Configuration	Wave 1	Wave 2	Bend	Balance
Ours	32.2	35.8	68.1	45.2
No Pose Prior	56.7	68.9	140.4	179.2
No Joint Prior	55.2	40.5	80.0	103.6
No MobileNet	44.2	37.1	65.7	42.9

Table 2. Comparison of accuracy by excluding certain parameters from the convex optimization. Without imposing a pose prior, the accuracy quickly deteriorated. We observe very minor accuracy gains by using the fully OpenPose CNN rather than the MobileNet implementation

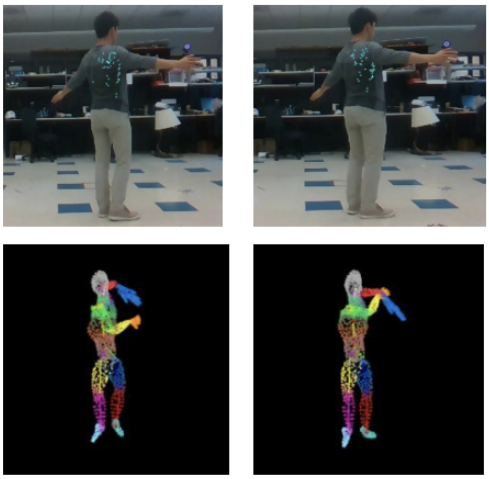

Figure 7. Poses that introduces significant self occlusion are problematic for the current approach. We hypothesis this may be due excessive sensor noise in these fast moving cases

8.3. Ablation Studies

We did experts on the effect of various parameter on both the accuracy our system.Without a pose prior, we perform especially poorly on complex poses such as the Wave 2 and Bend. This is likely due to noisy data and the algorithm is unable to estimate a realistic pose. Without the joint prior, the algorithm performs worse overall in all categories. Finally, we measured the effect of using a full OpenPose network to estimate the pose prior. The accuracy gain is minimal.

8.4. Failure Postures

In our experiments, we noticed one major class of failure the spinning pose. We noticed that when the side profile is exposed to the D435 camera, we failed to fit the pose completely. The main source of error stems from incorrect joint prior when the camera doesnt see enough information.

Configuration	Wave 1	Wave 2	Bend	Balance
Intel i7 + TitanX	25 FPS	24 FPS	24 FPS	24 FPS
Intel i7 + 1060	25 FPS	24 FPS	24 FPS	24 FPS
Intel i7 + No GPU	14 FPS	14 FPS	14 FPS	14 FPS
Intel i5 + No GPU	11 FPS	11 FPS	10 FPS	10 FPS

Table 3. Comparison of performance under various hardware configurations. A Intel i7 paired with a NVIDIA 1060 GPU can achieve real time performance. A CPU only configuration can run in near real time.

8.5. Performance

We ran our algorithm on a combination of CPUs (Intel i7-9700k, Intel i5-8600k) and GPUs (NVIDIA TitanX, and NIVIDA 1060). We chose the Intel i7 + Nivdia TitanX to represent a top end enterprise server configuration. We chose the Intel i7 + Nvidia 1060 to represent a mobile GPU configuration. Finally, we chose the Intel i5 by itself to represent an embedded device.

Currently, the only component that takes advantage of the GPU is the neural network that predicts the 2D joint positions. All other computations are CPU bound. Our algorithm is able to run in real time (24 FPS) with a NVIDIA 1060 or better. We noticed no performance gain with GPUs better than NVIDIA 1060. We believe this is due to the bottleneck being CPU bound computations.

However, even without a GPU, we are able achieve 10-14 FPS. Our tracking solution can be deployed to embedded devices for non-real time applications.

9. Conclusion

We present a fully automated method to estimate 3D body posture from a single RGB-D camera view in real time. Our method first uses a CNN to estimate 2D joint positions then initializes the 3D fitting with these joint locations. We use the recently proposed SPML body model which linearly related joint positions and body shape, allowing us to construct a highly constrained fitting process. We define our objective function and optimize the joint pose and body shape directly by minimizing the error between the model points and the real world point cloud observed by the depth camera. This gives us a simple and efficient solution to estimate pose and shape at the same time. We track the pose changes between frames by initializing each subsequent convex optimization with the posture from the previous frame. We evaluated the accuracy on the newest APE dataset and found that it performs comparably with non-real time methods.

Our formulation opens up many directions for future work. We will immediately address the failure cases by reexamining the pose prior and imposing a constraint on frame-to-frame changes in pose. We are currently explor-

ing use cases of this technology in driver safety.

References

[1] A. Agarwal and B. Triggs. 3d human pose from silhouettes by relevance vector regression. *IEEE CVPR*, 2004. 2

[2] S. Agarwal, K. Mierle, and Others. Ceres solver. 4

[3] B. C. D. K. W. W. T. W. M. A. H. A. Andrew G. Howard, Menglong Zhu. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *IEEE CVPR*, 2017. 3

[4] Authors. Frobnication tutorial, 2014. Supplied as additional material `tr.pdf`. 4

[5] A. Bulat and G. Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. *ECCV*, 2016. 2

[6] M. L. F. Bogo, M. J. Black and J. Romero. Detailed full-body reconstructions of moving people from monocular rgb-d sequences. *IEEE ICCV*, 2015. 2

[7] C. K. K. Wolff and H. Zimmer. Point cloud noise and outlier removal for image-based 3d reconstruction. *3DV*, 2016. 3

[8] f. Y. L. Li and D. Li. An improved ransac for 3d point cloud plane segmentation based on nomal distribution transformation cells. *IEEE CVPR*, 2016. 3

[9] S. T. B. A. M. A. P. G. L. Pishchulin, E. Insafutdinov and B. Schiele. Joint subset partition and labeling for multi person pose estimation. *IEEE CVPR*, 2016. 2, 3

[10] C. G. Lab. Cmu graphics lab motion capture database. 5

[11] J. L. S. Q. Little. Real-time tracking of articulated human models using a 3d shape-from-silhouette method. *Robovision*, 2001. 1

[12] S. R. M. Andriluka and B. Schiele. Monocular 3d pose estimation and tracking by detection. *CVPR*, 2010. 2, 4

[13] N. M. M. Loper and J. Romero. Smpl: A skinned multi person linear model. *SIGGRAPH Asia*, 2015. 1, 4, 5

[14] S. M. P. Mikhail Pavlov, Sergey Kolesnikov. Run, skeleton, run: skeletal model in a physics-based simulation. *ICML*, 2017. 1

[15] B. T. N. Dalal. Histogram of oriented gradients for human detection. *IEEE CVPR*, 2005. 3

[16] R. Okada and S. Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. *ECCV*, 2008. 2

[17] A. B. P. Guan, A. Balan. Estimating human shape and pose from a single image. *IEEE ICCV*, 2009. 2

[18] D. F. R. Newcombe and S. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. *IEEE CVPR*, 2016. 2

[19] X. C. S. Ren and Y. Wei. Face alignment at 3000 fps via regressing local binary features. *IEEE CVPR*, 2014. 4

[20] A. F. Shotton, Jamie and M. Cook. Real-time human pose recognition in parts from single depth images. *IEEE Transactions on Visualization and Computer Graphics*, 2016. 2

[21] C. T. T. Alldieck, M. Magnor and G. Pons-Moll. Video-based reconstruction of 3d people models. *IEEE CVPR*, 2018. 2

[22] S. B. L. B. T. Eichner, Michael Maire. Iterated second-order label sensitive pooling for 3d human pose estimation. *IEEE CVPR*, 2014. 5

[23] T. K. T. Yu and R. Cipolla. Unconstrained monocular 3d human pose estimation by action detection and cross-modality regression forest. *IEEE CVPR*, 2013. 5

[24] Z. Z. T. Yu and J. Zhao. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. *IEEE CVPR*, 2015. 2

[25] S. B. L. B. Tsung-Yi Lin, Michael Maire. Estimating anothropometry and pose from a single uncalibrated image. *IEEE ECCV*, 2014. 2

[26] S. B. L. B. Tsung-Yi Lin, Michael Maire. Microsoft coco: Common objects in context. *IEEE ECCV*, 2014. 4

[27] G. F. Tsz-Ho, Pons-Moll. Posebits for monocular human pose estimation. *IEEE CVPR*, 2014. 5

[28] P. F. V. Lepetit, F. Nogner. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 2008. 4

[29] V. P. K. V. Vegeshna. Deformable modeling for human body acquired from depth sensors. *IEEE CVPR*, 2017. 1

[30] K. M. G. Yang, S. Chan. Flow convnets for human pose estimation in vidos. *IEEE CVPR*, 2014. 5