Multi-View 3D Pose Estimation from Single Depth Images

Boya Peng
Stanford University
353 Serra Mall, Stanford, CA
boya@stanford.edu

Zelun Luo
Stanford University
353 Serra Mall, Stanford, CA
zelunluo@stanford.edu

Abstract

In this paper, we investigate the problem of multi-view 3D human pose estimation from depth images using deep learning methods. We utilize an iterative approach that progressively makes changes to an initial mean pose by feeding back error predictions. Our model is evaluated on a newly collected dataset (ITOP) that contains 30K annotated depth images from top-down and frontal views. Experiments show that our model achieves competitive results compared with current state-of-the-art models that use non-deep learning methods.

1. Introduction

Human pose estimation involves the identification of the keypoint locations of the body, which includes major body parts and joints. It has various applications including action classification and body movement prediction. It is also one of the basic building blocks for marker-less motion capture (MoCap) technology. With depth sensors becoming ubiquitous in applications ranging from gaming to security and smart spaces, accurately estimating human pose from these depth signals is a key component to leveraging the potential of these sensors.

Given an input depth image of a human, we aim to output a set of 3D coordinates corresponding to real world joint locations of the person's body. Current state-of-the-art results for 3D human pose estimation from depth images are given by traditional discriminative models. Shotton et al. [21] trained a random forest classifier for body part segmentation from a single depth image and used mean shift to estimate joint locations. Jung et al. [29] trained a regression tree to estimate the probability distribution to the direction toward the particular joint, relative to the current position.

Despite the success of deep learning in the RGB space [2, 25], for human pose estimation, its use has largely remained unsolved in the 3D depth-space. In addition, modern 3D solutions often constrain the problem to full-body frontal views, making it infeasible for real-world applications where a clean frontal view is not available. Another challenge is that existing depth datasets are often small in size, both in terms of number of frames and number of classes, which limits the use of deep learning methods. To address these challenges, our contributions are as follows:

1. We propose a model with iterative error feedback to predict 3D human joint positions from single depth images from multiple viewpoints.

2. To evaluate our model from challenging viewpoints, we introduce a new dataset of 30K depth images with annotated body part labels and 3D human joint locations. The dataset consists of front, top, and side views of people performing actions with occluded body parts.

2. Related Work

Our work draws on recent human pose research in generative models, discriminative part-based methods, and deep representation learning that embeds both local and global features in a shared embedding.

Generative Models. The human skeleton is a strong prior that can be leveraged for pose estimation. Generative models fit a human body template, either parametric or non-parametric, to the input data. These include variants of iterative closest point [9, 11, 12, 17], graphical models [13, 8], and pictoral structures [7, 3, 5, 20]. Other methods have attempted to use point clouds with database lookups [27], template fitting with Gaussian mixture models [28], and kernel methods with kinematical chain structures [4].

Discriminative Models. Instead of fitting a skeleton, discriminative models attempt to detect instances of body parts. In [21], Shotton et al. trained a random forest classifier for body part segmentation from a single depth image and used mean shift to estimate joint locations. This work inspired an entire line of pose estimation research investigating the use of regression tree methods: Hough forests [10], random ferns [14], and random tree walks [29].

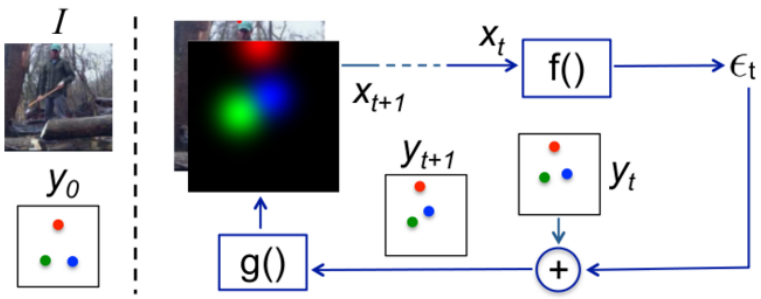


Figure 1: An implementation of the Iterative Error Feedback (IEF) model[2].

Representation Learning. Convolutional networks have seen widespread use as discriminative methods for solving the human pose estimation task. Since valid poses represent a much lower-dimensional manifold in the high-dimensional input space, it is difficult to directly regress from input image to output poses with a classical convolutional network. As a solution to this, researchers framed the problem as a multi-task learning problem where human joints must be first detected then precisely localized[18, 6, 19]. To further constrain the space of valid poses, Jain et al. [15] enforce global pose consistency with a Markov random field representing human anatomical constraints. Follow up work by Tompson et al.[24] improve the model and learning procedure.

Because human pose estimation is ultimately a structured prediction task, it is difficult for convolutional networks to correctly regress the full pose in a single pass. This problem was addressed through the use of iterative refinement techniques. In [23], Sun et al. proposed a multi-stage system of convolutional networks for predicting facial point locations. Each stage refines the output from the previous stage given a local region of the input. Building on this work, Toshev et al. [26] adapted this approach for full human pose estimation. Instead of predicting absolute human joint locations, Carreira et al. [2] refine the pose estimate by predicting pose offsets at each iteration. However, these iterative deep learning methods have only been used in RGB space. Building on this, we design a model that predicts 3D pose offsets on depth images from multiple viewpoints.

3. Approach

3.1. Overview

We formulate the human pose estimation problem as a regression problem. Inspired by DeepPose [26] and Iterative Error Feedback (IEF) [2], we adopt an iterative approach to learn pose correction offsets for each step of the error correction process. Instead of directly predicting joint locations, our model progressively make changes to an initial mean pose by feeding back error predictions. Unlike IEF, our model takes a depth image as the first channel of the input tensor which provides depth information. We encode 2D spatial information by mapping 3D joint coordinates into 2D image plane using known camera parameters, and generating heatmaps from estimates of 2D joints in each error correction step. During testing, we map the resulting 2D joint estimates back to 3D space to get real world 3D coordinates

3.2. Model Architecture

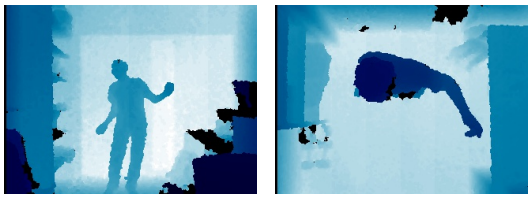
Input Representation During the t^{th} error correction, the input for our CNN model is a depth image I concatenated with J heatmaps generated from current estimated pose y_t, where J is the number of joints. y_t is a $J \times 2$ vector, containing 2D image plane coordinates for each joint. We map each row of y_t into a 2D Gaussian heatmap centered at the joint location with a fixed standard deviation. The J heatmaps are stacked with the depth image, resulting in a $H \times W \times (J+1)$ input tensor to our deep network, where H, W are input image height and width.

Iterative Error Feedback For the t^{th} error correction step, our CNN receives the input x_t – depth image I stacked with J heatmaps rendered from current estimates of joint positions y_t. The CNN outputs a correction ϵ_t. This correction is added to y_t, resulting in new joint position estimates y_{t+1}. New heatmaps are generated from y_{t+1} and are stacked with image I, resulting in x_{t+1}, and so on iteratively. Our model can be represented mathematically as follows:

$$\epsilon_t = f(x_t)$$
$$y_{t+1} = y_t + \epsilon_t$$
$$x_{t+1} = I \oplus g_{t+1}(y_{t+1})$$

where f represents our CNN model, g is the function that converts each 2D joint location into one Gaussian heatmap. Figure 1 shows the overall architecture of our model.

CNN Architecture We use the VGG-19 architecture



(a) Frontal view (b) Top down view

Figure 2: Examples images from ITOP dataset, containing challenging frontal and top down view images.

[22], with input images and heatmaps resized to 224 pixels by 224 pixels. We change the size of the first convolution layer filter from $3 \times 3 \times 3 \times 64$ into $3 \times 3 \times (J+1) \times 64$ to account for the depth channel plus J heatmaps. The network consists of 13 layers of 3×3 convolutions interspersed with 5 layers of 2×2 max-pooling operations. Additionally, we reduce the number of neurons in the last dense layers to $2J$. We replace the final softmax layer with a euclidean loss layer. The L_2 regression loss is formulated as follows:

$$L = \sum_{t=1}^{T} \sum_{i=1}^{M} (\epsilon_t^i - e(y^i, y_t^i))^2$$

where T is the number of correction steps and M gives the number of training data. ϵ_t is the predicted error correction in the t^{th} step while $e(y - y_t)$ gives the ground truth error correction. The bounded error correction for the t^{th} step and k^{th} joint is calculated as follows:

$$e(y^k, y_t^k) = \min(L, ||u||) \cdot \hat{u}$$

where $u = y^k - y_t^k$ is the error correction vector, $\hat{u} = \frac{u}{||u||_2}$ is the corresponding unit vector, and L gives the upper bound of correction for each joint location. We minimize the loss using stochastic gradient descent (SGD) with every correction step being an independent training example. We grow our training set progressively: for the $(t+1)^{th}$ step, we add x_{t+1} to our training set with corresponding ground truth labels, $e(y, y_{t+1})$. In this way, early steps can be optimized longer and get consolidated.

3.3. Training and Optimization

We train the full model end-to-end in a single step of optimization. We train the CNN from scratch with all weights initialized from a Gaussian with $\mu = 0, \sigma = 0.005$. We use the Adam optimizer [16] with a learning rate of 1×10^{-5}, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Our training batches consist of depth images resized to 224 pixels by 224 pixels. Heatmaps are then generated from current joint estimates to form the input tensor $\mathbf{x}$ as outlined in Section 3.2.

4. Datasets

To more rigorously evaluate our model, we collect a new dataset consisting of varied camera viewpoints.

4.1. Invariant-Top View Dataset (ITOP)

Existing depth datasets in the front view are often small in size. Besides, the benchmark for the evaluation of top-view human pose still does not exist. To resolve these issues, we collected our own dataset consisting of 30K depth images. The dataset is two orders of magnitude larger than previous depth-based pose estimation datasets and contains images from both frontal and top views of 20 people performing 15 action sequences each. Each image is labeled with 15 precise 3D joint locations from the point of view of the respective camera. For both frontal and top-down views, we use the video sequences of the first 4 people as our test set, and the remaining 8 for training.

Data Collection. We collect our data using two kinect depth cameras, with one facing front and the other facing top-down. Since the Kinect camera is unable to perform top-down human pose estimation, we use a front-facing camera to estimate 3D joint locations and transform these coordinates to a top-view coordinate system shared by the top-view camera based on the translational offsets and relative Euler angles of the two cameras. This method can be extended to n cameras from arbitrary viewpoints. Figure 2 shows example images from the ITOP dataset.

Ground Truth Error Correction. The aforementioned coordinate transformation is successful when front-facing skeleton is accurately generated – this is a strong assumption that does not always hold true. To solve the problem of noisy skeleton estimates, we propose an approach that iteratively refines the initial estimation of both the front and top-view human pose estimates by performing local mode-finding on an intermediate body part representation (see Figure 3).

In the initial step, we perform a background subtraction on the depth images using Gaussian mixture-based foreground-background segmentation on sequential frames. The background-subtracted images are then segmented into body parts using the following procedure:

1. Generate the human pose skeleton by connecting adjacent joints in topological order. Each pixel on the skeleton is assigned a label based on the relative weights of joints on both ends (e.g. torso is physically larger than the shoulders and therefore carries a greater relative weight).

2. Pixels on the skeleton and their assigned body part labels are used as training examples and class labels for a k-nearest neighbor model, respectively.

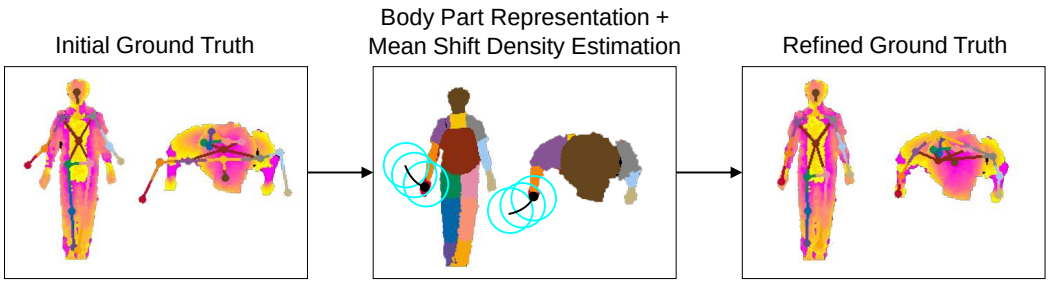
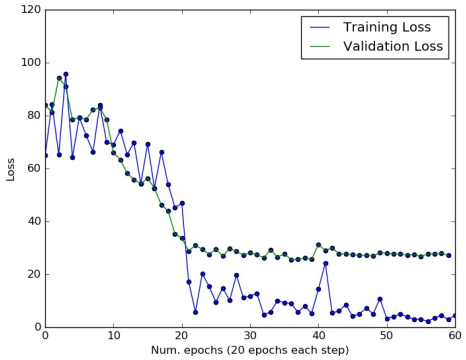
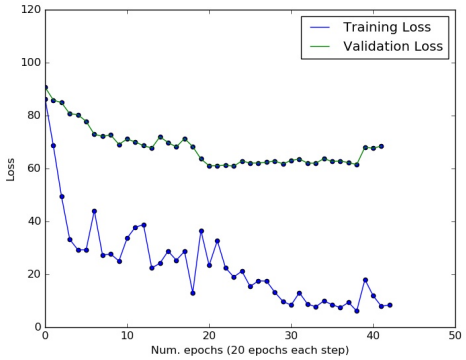


Figure 3: Ground truth correction for front and top views. In the event that ground truth skeleton data is noisy, we correct the ground truth using an iterative refinement technique.



(a) Frontal view



(b) Top down view

Figure 4: Training and validation loss for IEF from frontal and top-down views.

3. Pixels not directly on the skeleton are assigned body part labels based on the resulting k-NN model.

Joint positions are re-calculated from the estimated body parts by accumulating the global 3D centers of probability mass for each part. As the final step, we manually go over each frame and remove frames that present inaccurate joint positions.

5. Experiment

5.1. Baselines

Our comparison with state-of-the-art methods is split into two parts: (i) training and testing on frontal views, (ii) training and testing on top views. We compare our model to Shotton et al. [21] and Random Tree Walks (RTW) [29]. We implemented both models from scratch, as their source code are not publicly available.

The Shotton model consists of multiple decision trees that traverse each pixel to find the body part labels for that pixel pixel. Once the pixels are classified into body parts, the possible joint positions are found with multiple mean-shifts. RTW trains a regression tree to estimate the probability distribution to the direction toward the particular joint, relative to the current position. At test time, the direction for the random walk is randomly chosen from a set of representative directions.

5.2. Implementation Details

To aid in reproducibility, we provide a detailed list of preprocessing, hyperparameters, and model settings.

Iterative Error Feedback. Our model from section 3 is implemented using TensorFlow[1]. We use mini-batches of size 10. To generate heatmaps at the first error correction step, the mean 2D pose of the training set is used. Heatmaps are 224 pixels by 224 pixels and centered at joint locations. Each image is concatenated with J heatmaps where J is the number of human joints. Our model takes a $224 \times 224 \times (J+1)$ tensor as input, and is trained for 4 error correction steps, using 20 epochs for each step. Other information can be found in section 3.3.

Shotton et al. We implement the model from Shotton et al.[21] as its original implementation is not publicly available. We train and test our implementation on ITOP (top-view) and are able to produce comparable results. We set the model parameters as follows: a random forest of 3 trees,

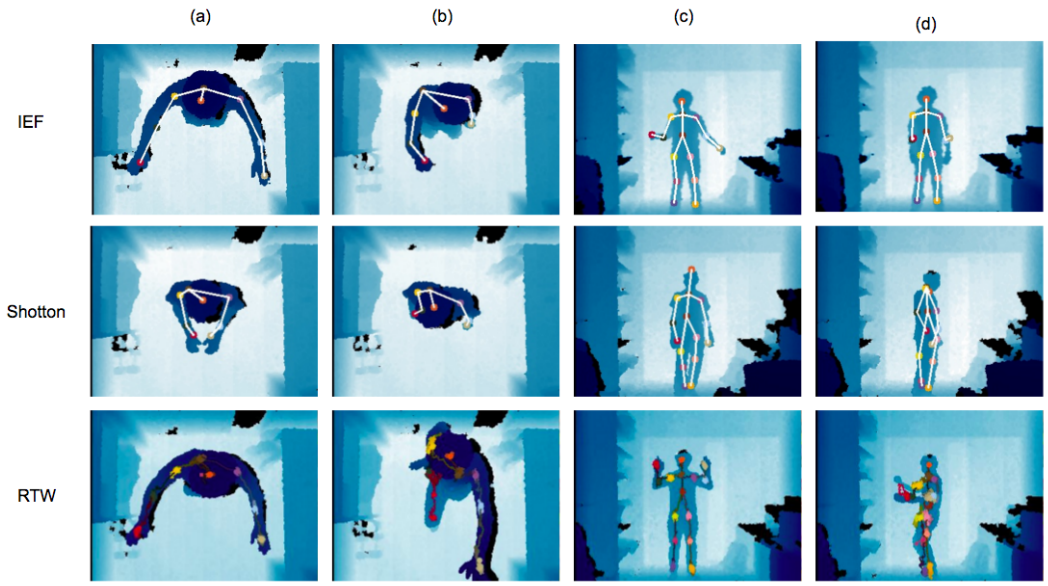


Figure 5: Qualitative results from frontal and top down views for Shotton and RTW. Column (a) and (c) are good prediction examples, (b) and (d) are bad ones.

15 deep, approximately 3,000 training pixels per image, 2,000 candidate features, and a probe offset of 100 pixels. The same local mode-finding approach is employed to generate joint proposals from body part labeling predictions. We use a bandwidth of 0.065m, a probability threshold of 0.14, and a z offset of 0.039m.

Random Tree Walks. Random tree walks [29] is a refined model based on the model from Shotton et al. and enjoys significantly faster runtime by training a regression tree for each joint instead of each pixel. Five hundred sample points are randomly generated with a maximum offset of 30 pixels along each axis. For each sample point, we generate 500 features in random directions with a probe offset of 100 pixels.

5.3. Evaluation Metric

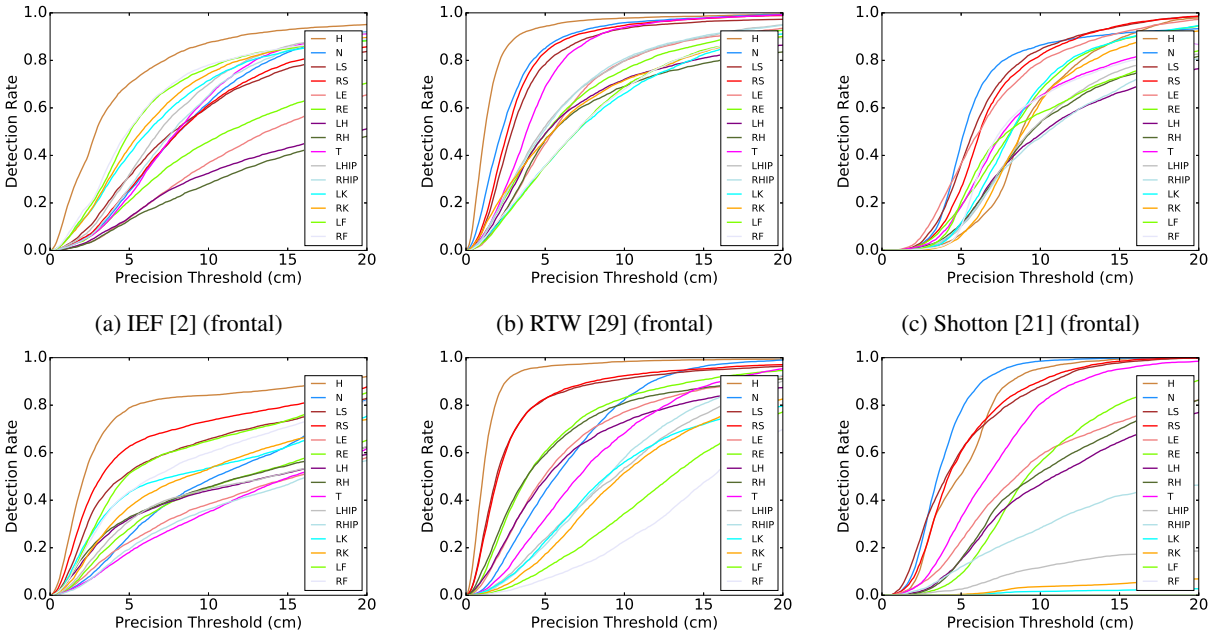
For summary tables and figures, we use the mean average precision (MAP), which is simply the mean of the average precision scores for all human body parts.

5.4. Results

Iterative Error Feedback Figure 4 shows training and validation loss for our Iterative Error Feedback model. From Figure 4 (a) we can see that for frontal view, both training and validation losses drop exponentially during the first error correction step (i.e. first 20 epochs) but slow down after the first step. The small but noticeable gap between training and validation loss indicates that the model slightly overfits. For top-down view, there is a bigger gap between

	ITOP (front-view)			ITOP (top-view)		
Parts	RTW	S	IEF	RTW	S	IEF
H	0.978	0.638	0.944	0.983	0.954	0.851
N	0.958	0.864	0.866	0.822	0.985	0.492
LS	0.934	0.845	0.801	0.911	0.878	0.657
RS	0.947	0.821	0.776	0.924	0.901	0.738
LE	0.797	0.791	0.541	0.771	0.589	0.379
RE	0.761	0.673	0.566	0.830	0.559	0.458
LH	0.720	0.489	0.339	0.729	0.466	0.441
RH	0.690	0.537	0.285	0.809	0.516	0.462
T	0.938	0.650	0.890	0.681	0.805	0.363
LHIP	0.798	0.542	0.840	0.536	0.117	0.451
RHIP	0.807	0.474	0.796	0.578	0.283	0.364
LK	0.660	0.690	0.798	0.561	0.016	0.547
RK	0.716	0.623	0.837	0.516	0.036	0.531
LF	0.690	0.579	0.842	0.344	0.000	0.664
RF	0.678	0.646	0.804	0.227	0.000	0.601
mAP, U	0.848	0.707	0.751	0.847	0.731	0.560
mAP, F	0.805	0.658	0.728	0.682	0.474	0.533

Table 1: Comparison of results with different methods using a 10 cm threshold. U and F stand for upper body and full body; S stands for Shotton et al. The body parts we evaluate on include head (H), neck (N), shoulders (LS/RS), elbows (LE/RE), hands (LH/RH), torso (T), hips (LHIP/RHIP), knees (LK/RK), and feet (LF/RF).



(a) IEF [2] (frontal) (b) RTW [29] (frontal) (c) Shotton [21] (frontal)

(d) IEF [2] (top down) (e) RTW [29] (top down) (f) Shotton [21] (top down)

Figure 6: Detection rates for various detection thresholds for body parts: head (H), neck (N), shoulders (LS/RS), elbows (LE/RE), hands (LH/RH), torso (T), hips (LHIP/RHIP), knees (LK/RK), and feet (LF/RF).

training and validation loss, indicating more severe overfitting (as Figure 4 (a) shows). To address this, we could do data augmentation such as random crops to increase the training set size, or/and increase regularization strength. In addition, due to limited time and computation resources, we were only able train the top-down view model for 2 error correction steps, 20 epochs each step. As the model has not yet converged, we believe it can achieve better performance with more error correction steps.

Quantitative Analysis Table 1 shows the average precision for each joint using a 10 cm threshold and the overall mean Average Precision (mAP) for Iterative Error Feedback (IEF) [2], Random Tree Walk (RTW) [29] and Shotton et al [21]. RTW achieves the best overall accuracies for both full and upper body joints from frontal and top-down view. IEF is less robust on hands and elbows, but achieves the best performance on lower body joints, and gives competitive results for upper body joints. As occlusions happen most often with lower body joints in the ITOP dataset (in particular from top down view), IEF appears to be the most robust among three methods in the case of occlusion. Overall, Shotton gives the lowest mAP. This is due to the limitation of the size of our training set (9K images for each viewpoint), as the original model was trained on 900K images with 300K images per tree. Figure 6 shows the precision results of each methods in detail.

Qualitative Analysis Figure 5 shows examples of qualitative results from frontal and top down views for all three methods. For top down view, we show only 8 joints on the upper body (i.e. head, neck, left shoulder, right shoulder, left elbow, right elbow, left hand, and right hand) as the lower body joints are almost always occluded. Shotton and RTW give reasonable results when all joints are visible (as in figure 5 (a) and (c)) but do not perform well when occlusion happens (Figure 5 (b) and (d)). For Shotton, we can see from figure 5 (b) that the prediction for the occluded right elbow is topologically invalid though both right shoulder and hand are visible and correctly predicted. This is because the model doesn't take into account the topological information among joints, so it is not able to modify its prediction for one joint base on the predicted positions of neighboring joints. For RTW, figure 5 (b) shows that the predicted position for right hand goes to the right leg. Though legs and hands possess very different depth information, the model mistook the right leg for right hand when the hand is occluded and the leg appears in the common spatial location of a hand. Figure 5 (d) shows that the prediction results for visible joints are affected due to partial occlusions. IEF also performs well when all joints are visible (as in figure 5 (a) and (c)). In addition, figure 5 (b) shows that IEF is able to give reasonable predictions for the occluded left shoulder, elbow and hand.

6. Conclusion and Future Work

We introduced a multi-view model that estimates 3D human pose from a single depth image. Instead of directly regressing to joint locations, we adopt an iterative approach that progressively make changes to an initial pose by feeding back error corrections. We compared our model with two state-of-the-art models that we implemented from scratch, and showed that our model achieves competitive performance. All three models were trained and tested on a newly-collected depth dataset with 30K images from multiple viewpoints. Qualitative results shows that our model is able to achieve better performance in the case of occlusion.

For future work, we would like to train and test on a mixture of frontal and top down view poses, in order to further explore our model's ability to handle multiple viewpoints. In addition, due to the limitation of computing power, we are only able to train the IEF model with 9,000 images for 4 error correction steps with 20 epochs each per viewpoint. We believe that the model can achieve better performance when trained on a larger dataset for more epochs and error correction steps.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*.

[2] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. 2015.

[3] M. Dantone, J. Gall, C. Leistner, and L. Gool. Human pose estimation using body parts dependent joint regressors. *CVPR*, 2013.

[4] M. Ding and G. Fan. Articulated gaussian kernel correlation for human pose estimation. *CVPR Workshops*, 2015.

[5] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *IJCV*, 2012.

[6] X. Fan, K. Zheng, Y. Lin, and S. Wang. Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation. *CVPR*, 2015.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.

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

[9] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-time human pose tracking from range data. *ECCV*, 2012.

[10] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. *ICCV*, 2011.

[11] D. Grest, J. Woetzel, and R. Koch. Nonlinear body pose estimation from depth images. Springer, 2005.

[12] D. Haehnel, S. Thrun, and W. Burgard. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. *IJCAI*, 2003.

[13] L. He, G. Wang, Q. Liao, and J.-H. Xue. Depth-images-based pose estimation using regression forests and graphical models. *Neurocomputing*, 2015.

[14] N. Hesse, G. Stachowiak, T. Breuer, and M. Arens. Estimating body pose of infants in depth images using random ferns. *CVPR Workshops*, 2015.

[15] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. *ICLR*, 2013.

[16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

[17] S. Knoop, S. Vacek, and R. Dillmann. Sensor fusion for 3d human body tracking with an articulated 3d body model. *ICRA*, 2006.

[18] S. Li, Z.-Q. Liu, and A. Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. *IJCV*, 2015.

[19] S. Li, W. Zhang, and A. B. Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. *ICCV*, 2015.

[20] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet conditioned pictorial structures. *CVPR*, 2013.

[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*. IEEE, June 2011.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[23] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. *CVPR*, 2013.

[24] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *NIPS*, 2014.

[25] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. volume abs/1312.4659, 2013.

[26] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. *CVPR*, 2014.

[27] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys. Accurate 3d pose estimation from a single depth image. *ICCV*, 2011.

[28] M. Ye and R. Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. *CVPR*, 2014.

[29] H. Yub Jung, S. Lee, Y. Seok Heo, and I. Dong Yun. Random tree walk toward instantaneous 3d human pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.