3D Pose Estimation using Synthetic Data over Monocular Depth Images

Wei Chen
cwind@stanford.edu

Xiaoshi Wang
xiaoshiw@stanford.edu

Abstract

We proposed an approach for human pose estimation over monocular depth images. We augment the data by sampling from existing dataset and generate synthesized images. The generated dataset covers a more continuous pose space than the existing one. We use the generated dataset to train a multi-pathway neural network. We also introduced an orientation and translation invariant embedding for poses within the network.

1. Introduction

Pose estimation of human has been recognized as an important vision task by artificial intelligence community. As human poses provide direct indications of their actions, correctly evaluating human pose from a single image provides solid information for various vision and learning tasks including detection, tracking, scene analysis and social-interaction understanding, etc.

Pose estimation has been explored for several years. Recent researches show special interests on neural-network based approaches, which turns out to have supreme performance on other vision tasks. For monocular image pose estimation, most previous works refer to RGB data [14] [10]. The community is also working towards large-scale dataset incorporating with depth images. Especially for pose datasets, Human 3.6M [12], CDC4CV[2] and CMU MOCAP [4] become available recently.

We are strongly convinced that depth images provide more abundant information than RGB images. In particular for pose estimation, depth information helps us to better address issues like 3D symmetry. To better understand depth images, we propose to apply a neural-network based algorithm which evaluates human poses with only depth images.

2. Approach

2.1. Previous Works

Pose data with depth image and MOCAP labelings become available in recent years. [2] builds a dataset which is limited in scale (692 images). [4] and [12] has significantly larger scale. They contain poses collected under various scenes. However, human skeleton has a number of freedom joints. The depth images of poses are also subject to variances in scales, rotations, translations, and background noises. It turns out that currently available pose datasets with depth images are not abundant enough [7] to cover the whole space of available poses. [7] uses synthesized depth images to reinforce the training process. [9] discussed a Bayesian-Network based approach that learns prior distribution from real pose samples, which provides us the potential to sample a large-scale data set that covers the whole space of available poses.

Depth image/video based pose estimation is an attractive area where few previous work [7] is available. Nevertheless, there are various work in related fields. For pose estimation on monocular RGB images, [10] explores a multi-pathway CNN approach for pose estimation; [14] introduces a hourglass architecture. For depth related task, [17] proposed a multi-pathway CNN for detection task over RGB-D images; this work includes 3D convolution layer for the pathway associated with depth images.

To enhance our network architecture, we are interested in looking at the pose space. Existing works including [10] and [14] trains the CNN that maps images to poses directly. In other words, the final layer of CNN in [10] are predicted 3D positions of all joints; and similarly, the final layer of [14] is the heatmap of 2D positions of joints. [16] and [11] discussed metric learning approaches which trains a CNN that embeds images to a space specifically designed for target labels (shape[11], class [16], etc.). [5] applies a metric learning approach on monocular image pose estimation task; it trained a CNN that clusters similar poses using contrastive loss [13].

2.2. Problem Statement and Methods

In our project, we would like to address the problem of human pose estimation over monocular depth images. Our proposed approach intend to combine the advantages of the previous works. We would like to utilize the recognized power of deep CNN to achieve this task. We will augment the existing dataset using synthesized data and use the large-scale data to train a multi-pathway deep CNN. We also give a closer look to metric learning approaches. We propose an embedding of 3D poses which allows us to train the CNN in a translation and rotation invariant way.

3. Implementation Details

3.1. Overview

Our project include two parts. First, we propose to augment the existing datasets using synthesized data so that poses in the training set covers the pose space better. Specifically, we will apply a Bayesian-Network based approach which learns the skeleton structure and then use KDE to sample from prior probability. Second, we propose to apply a neural-network approach for pose-estimation over a single depth image. Our neural-network will embed depth image to a pose space invariant to translation and rotation. Given that we have only about 6 weeks, the major focus of our project will be the first part. As a part a larger project, a portion of our tools refer to some recent works [6] on the same topic.

3.2. Datasets

For this project, we refer to a human pose dataset, HUMAN 3.6M [12]. The dataset collects 3.6 million 3D human poses and corresponding images from a total of 11 professional actors, under 17 different scenarios.

Pose information is collected by videos. The recording system contains four RGB cameras, one Time-of-flight (ToF) cameras, and a 10-camera 3D motion capture system. We will use ToF data and motion capture data for our project.

ToF data provides depth images. As motion caputure system record the accurate 3D position of each joint, the data will be used as ground-truth labelings for poses.

3.3. Data Synthesizing

Our synthesized data is generated using parameters learned from real dataset. Specifically, we would like to generate new pairs of depth images and 3D joint labelings with synthesized data.

Although real dataset might not cover combinations of all available poses, it provides information on ”partial” poses. For example, real dataset might not have combinations of all pairs of arm positions AND leg positions; nevertheless it might be sufficient to cover individually all arm positions OR let positions. We propose an approach that will retrieve relatively independent parameters from real data, and generate synthesized data from combinations of independent parameters.

It is worth mentioning that the motion of human body can be viewed as a combination of motions of highly independent joints. Intuitively, human poses can be generated from a set of relatively independent random variables. We can learn prior probability of those variables from real data.

Nevertheless, determining variables to be learned is a crucial issue. One naive way is to assume that all joints are constrained by a given skeleton structure. The advantage of this assumption is that joints used in motion camera are closely mapped to biological joints. However, the skeleton is joint-specific. Which require us to provide different skeletons for different sets of joints. As different dataset ususally use different joints for motion capture, it makes this approach poorly generalizable and strongly biased.

Therefore, we would like to learn the skeleton information over the real dataset. We follow the method proposed in [9]. The skeleton is computed as a maximum spanning tree over the weighted completed graph of all joints, where edge weights are given by pairwise mutual information values between joints.

Specifically, we denote n the number of joints and N the number of samples. For $i \leq N$ and $j \leq n$, we denote $x_j^{(i)}$ the 3D position of the jth joint of the ith sample. We define a graph $G(V, E)$ with n vertices corresponding to n joints. For any j and k, the edge weight is the mutual information of vertices v_j and v_k, given by

$$MI(v_j, v_k) = H(v_j) + H(v_k) - H(v_j, v_k),$$

where H denotes the entropy of the distribution of position of joints. [9] and [8] discussed estimating the entropy using nearest-neighbour approach. For $x_j^{(i)}$, denote $d_j^{(i)} = \min_{x_j^{(k)}, k \neq i} \|x_j^{(i)} - x_j^{(k)}\|$ the distance of the nearest neighbour of $x_j^{(i)}$. The entropy of the jth joint, up to

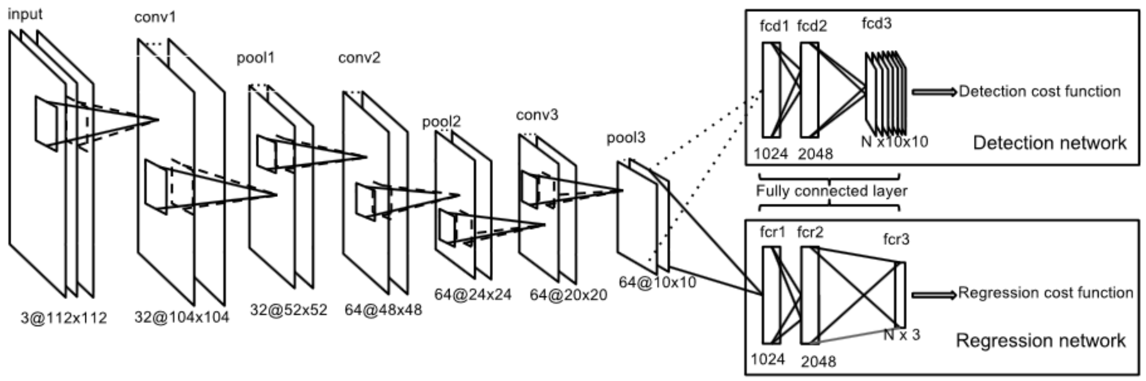

Figure 1. The 2-pathway CNN Architecture Proposed by [10]. The architecture contains 7 convolutional layers followed by two separate pathways for detection and regression. Each pathway contains 3 FC layers.

constant scale and constant shift, is given by

$$H(v_j) = \frac{1}{N}\sum_{i=1}^{N} \ln \|d_j^{(i)}\|.$$

The skeleton is obtained by computing the maximum spanning tree over G.

After we obtain the skeleton, we orient G and sample each joint from the root to leaves. Each joint's relative position to its parent is sampled independently using KDE with gaussian kernel. Specifically, let $y_j^{(i)}$ denotes the relative position of joint j to its parent in the ith sample, new samples are generated from a mixture of Gaussian

$$p_j(y) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(y; y_j^{(i)}, b_j^2 I),$$

where b_j is the bandwidth given by $b_j \approx 1.06\sigma_j N^{-1/5}$ with σ_j being the standard deviation of y_j's[15].

This method is proven to be able to generate a more continuous pose space as discussed in [3].

After generating poses, we use MakeHuman software [1] to generate various 3D human models. We create depth images by rendering those models with various camera angles. We also introduce random noise and backgrounds to rendered depth images so that the synthesized data appears like real data.

3.4. Models

For the model, we refer to the architecture proposed by [10]. The model shown in figure 1 includes a number of convolutional layers followed by two pathways. The first pathway is the detection path which outputs the bounding box of the human for input images. The second pathway is the pose estimation (regression) pathway which outputs the 3D positions of each joint of the predicted pose for the input 2D image.

Training the model includes two steps. First, we train only with the detection pathway. We then fix all convolutional layer and fine-tune the regression pathway using the same training data.

[10] uses the 3D position of each joint (relative to the root joint) as the output of the regression model. It uses relative position between joints to define the loss function. For each joint i, let J_i^w denote the absolute position of the joint and J_i^p being the relative position to its parent; we also denote $\hat{J}$ be corresponding value from ground-truth labelings. [6] proposed a loss function which handles translation better and has experimentally better performance. The loss is defined as

$$L = 0.027L_d + 0.833L_s + 0.139L_r,$$

where L_d is the sum of error in absolute joint positions:

$$L_d = \sum_i \|J_i^w - \hat{J}_i^w\|^2,$$

L_s is the sum of limb-length errors:

$$L_s = \sum_i \|J_i^p - \hat{J}_i^p\|^2,$$

and L_r is sum of rotation error:

$$L_r = \sum_i \left(1 - \frac{J_i^p \cdot \hat{J}_i^p}{\|J_i^p\|\|\hat{J}_i^p\|}\right).$$

The loss function L is defined on the last layer of regression path, which contains the predicted position of all 3D joints. The disadvantage of this approach is that although the loss function is invariant of translation, the output of the CNN, as the direct position of each joint, is variant to translation, rotation, etc. This requires our model to adapt poses subjected to translations and rotations.

[11] considered embedding of 3D CAD models into a space invariant to rotation. Intuitively, human pose as a 3D description, should also be considered as invariant under rigid transformations. Therefore, it is worthy considering an rotation invariant embedding for poses. In other words, we would like the last CNN layer output the same pose for images taken at different camera angles of the pose. Given that our time is limited, we propose a case-specific embedding which align x-axis of the chest plane and orient the head on the y-axis.

Specifically, we first put the "body" joint at the origin, and define the chesk plane as the plane defined by "body", "Left-Shoulder" and "Right-Shoulder". Let the chest plane have unit norm $N_1 = (n_1, n_2, n_3)^T$. To align this norm to x-axis, we define

$$N_2 = \frac{(-n_3, 0, n_1)^T}{\sqrt{n_1^2 + n_3^2}}$$

and $N_3 = N_1 \times N_2$. Notice that $R = (N_1, N_2, N_3)$ defines a orthonormal matrix which rotate x-axis to N_1. We define $R_n = R^T$. After we apply R_n to the pose, we compute the projection of "head" joint on y-z plane and apply the rotation around x-axis to orient the project on the y-axis.

From two rotations discussed above, we are able to embed poses into a space invariant to rotations and translations. We apply loss function L as previously discussed on normalized poses.

Our naive embedding is very problem-specific. Generalized and more powerful embedding is possible, and it serve as a part of a larger project which exceed the scope of this report.

4. Experiments

4.1. Data Generation

Computing mutual information requires us to take nearest neighbours of each sample at each joint. The process is implemented in a naive way which takes $O(N^2)$ time complexity for each joint. As we need to compute the joint entropy $H(v_j, v_k)$ for all joint pairs, overall it takes

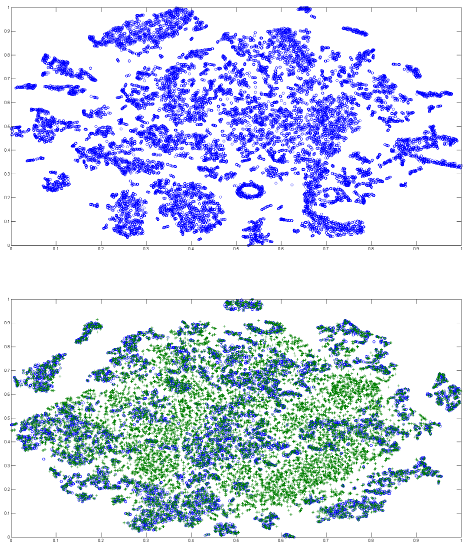

Figure 2. The t-SNE embedding of real data. Straight lines at the out bound implies possible continuous translation (e.g. walking) in the pose space. The ring in the lower bottom indicates possible rotation in the pose space. It is also obvious that there are many small, scattered clusters.

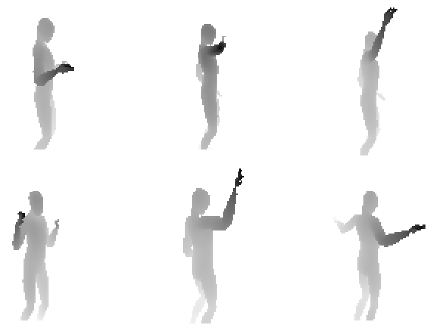

Figure 3. Examples of Synthesized Depth Images. Each image is obtained by rendering randomly chosen human model, with a sampled pose and camera angles.

$O(n^2 N^2)$ time complexity to learn the whole skeleton structure. It turns out that for a large dataset (large N), we must limit the number of joint used and the number of samples to achieve considerably acceptable run time. Instead of using all 32 joints provided by HUMAN 3.6M, we only used a subset of 10 joints, including head, arms, hands, body, legs, and feet. It turns out that limited number of joints is able to generate sufficient variety of poses from our sampling process. Figure 3 shows six examples of synthesized depth images.

We sampled 100,000 poses from 18,072 existing ones. We

	Left-Hand	Left-Elbow	Body	Head	Right-Elbow	Right-Hand	Mean
(Syn,Real)/Syn	68.7254	70.1336	0	50.1225	63.2041	71.354	64.7079
(Syn,Real)/Real	154.257	163.4265	0	138.5861	158.3504	152.4464	153.4133
Embed+(Syn,Real)/Syn	55.5436	62.7590	0	48.1183	54.4399	54.1073	54.9936
Embed+(Syn,Real)/Real	132.8012	149.2931	0	141.0282	160.3875	147.2454	146.1511

Table 1. Prediction Error in mm of Joints. The last column is the mean of all joints. Also notice that since error are taken over normalized poses, the error of body is zero, since we put the origin at the body. The mean value of joints are taken over all joints other than the body joint.

visualized the data by t-SNE [18] embedding as shown in figure 2. The raw real data is scattered into small clusters. Our synthesized data filled in the gap between those clusters and combines them into a large cluster. Furthermore, the synthesized pose space appears to be more symmetric. As the space for available human poses is continuous (as we can move from one pose to another one) and symmetric (as joints are symmetric), we conclude that poses we generated are covers the space of available poses better than the original dataset.

We use MakeHuman software [1] to obtain 12 human models with various body shapes. We render those shapes to generate synthesized depth images from previously sampled poses. Figure 3 shows six example of rendered depth images.

4.2. Pose Estimation

We first train the detection pathway and fine-tune the regression pathway. We trained our model twice, once with the embedding of the normalized pose, once without the embedding. Both models are trained with a combination of synthesized and real data.

We use the mean distance between absolute position of corresponding joints as the evaluation metric of our experiments. Since the output of experiments with embedding are normalized poses, we normalized both ground truth and predicted poses in all experiments before computing the error. This makes our experiment results cross-comparable to each other. Figure 4 and table 1 includes the error computed at each joint. We are able to achieve satisfying results on validation using also synthesized images. The accuracy improves with our proposed embedding. However, the accuracy drops on tests over real images, comparing to validations on synthetic images.

One reason that our embedding provides better performance is that it simplifies the variety of CNN output. As a various number of poses map to one single normalized pose in the embedding space, weights would converge faster on the same amount of training data. We could notice

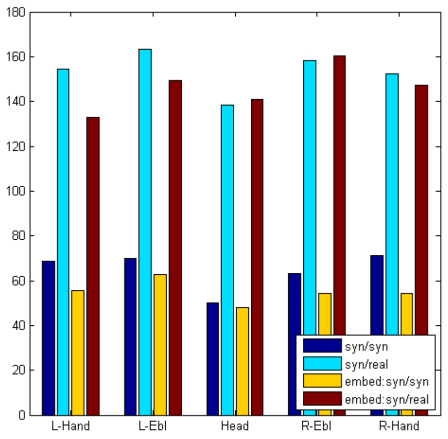

Figure 4. Mean Prediction Errors over Each Joint in Different Experiments. Errors are measured in millimeters. The lower the error, the better the results.

that the errors of all joints, except head, drop significantly in all four sets of experiments. One explanation is that the position of head is more likely to be rotation invariant comparing to other joints. Therefore, applying the embedding does not help improving the prediction of the position of head much.

Figure 5 and figure 6 show visualizations of predicted poses. Figure 5 are synthesized images and figure 6 are real images. We can see that our predictions of joint positions on synthesized images is closer to ground truth than predictions on real images. The last row of figure **??** also shows that when testing over real images, our model would take a part incorrectly as a joint (the right elbow and left hand).

One major reason that we have significantly worse result on real data could be that our synthesized data appears very differently to the real data. Although we add random background and noise to the synthesized image, the background in HUMAN3.6M dataset is relatively clean and invariant. In other words, the test set itself is strongly

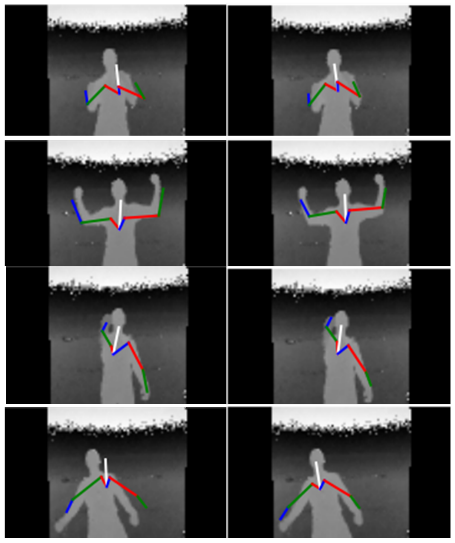

Figure 5. Visualization of Results over Synthesized Images. The left column contains predictions and the right column contains ground truth labels. The first three row contains qualitatively good results. The last row is a failed result.

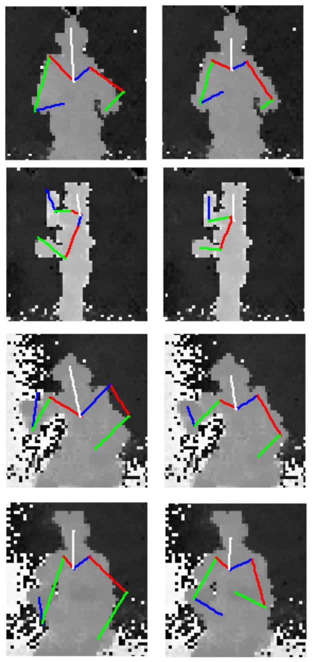

Figure 6. Visualization of Results over Real Images. The left column contains predictions and the right column contains ground truth labels. The first three row contains qualitatively good results. The last row is a failed result.

biased in background. The model could be misled by features introduced by the background. In addition, as the performers in HUMAN3.6M wear MOCAP devices when the video is taken, their appearance in ToF significantly differ from the model we used to render depth images. Since MOCAP sensors are placed at joints, they introduce strong distortion to the depth data as our model is expecting smoother joints.

5. Conclusions and Future Works

In this project, we explored human pose estimation on monocular depth images. We contributed to this new area by combining advantages of works in other related fields. We augmented real-world dataset by synthesized images. We use the synthesized images to train a deep network with our newly proposed pose embedding layer. The embedding is proven to be able to improve our models in experiments.

Our visualizations and quantitative results shows that distortion in real data could introduce potential problems into our models. It is worth exploring domain transformation and low-level feature fine-tuning to enforce our model to adapt such distortions.

As we are referring to a neural network which works well on RGB images, we will also explore other architectures designed specifically for depth or 3D inputs [17][14].

References

[1] M. Bastioni. Makehuman software. `http://www.makehuman.org`.

[2] H. C. Brian Holt, Eng-Jon Ong and R. Bowden. Putting the pieces together: Connected poselets for human pose estimation. *ICCV*, 2011.

[3] W. Chen, H. Wang, Y. Li, H. Su, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen. Synthesizing training images for boosting human 3d pose estimation. *arXiv preprint arXiv:1604.02703v1*, 2016.

[4] CMU. Carnegie-mellon graphics lab motion capture database. `http://mocap.cs.cmu.edu/`.

[5] N. K. T. L. G. T. A. T. Greg Mori, Caroline Pantofaru and W. Yang. Pose embeddings: A deep architecture for learning to match human poses. 2015.

[6] D. Held and J. Huang. Pose estimation on depth images with convolutional neural network. 2016.

[7] M. C. T. S. M. F. R. M. A. K. Jamie Shotton, Andrew Fitzgibbon and A. Blake. Real-time human pose recognition in parts from single depth images. *CVPR*, 2011.

[8] K. L and L. N. Sample estimate of the entropy of a random vector. *Probl. Peredachi Inf.*, pages 9–16, 1987.

[9] A. Lehrmann, P. Gehler, and S. Nowozin. A non-parametric bayesian network prior of human pose. pages 1281–1288, 2013.

[10] S. Li and A. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. *Asian Conference on Computer Vision*, 2014.

[11] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas. Joint embeddings of shapes and images via cnn image purification. *ACM Transactions on Graphics (TOG)*, 34(6):234, 2015.

[12] C. Lonescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[13] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *NIPS*, 2013.

[14] A. Newwell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937v1*, 2016.

[15] B. Silverman. Density estimation for statistics and data analysis. page 48, 1998.

[16] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. *arXiv preprint arXiv:1511.06452*, 2015.

[17] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. *CVPR*, 2016.

[18] L. Van Der Maaten and G. Hinton. Visualizing data using t-sne. *The Journal of Machine Learning Research*, pages 2579–2605, 2008.