3D Human Pose Estimation based on Center of Gravity

1st Hao Xu
School of Information Engineering
Ningxia University
YinChuan, China
Hao_Xu321@163.com

2nd Suping Wu
School of Information Engineering
Ningxia University
YinChuan, China
wspg123@163.com

Abstract—In this paper, we propose a method about 3D human pose estimation with only 2D joints as input. Previous methods generally lift 2D poses to 3D space through a single mapping function, in which case some large-pose samples far away from the majority distribution may not be well concerned. To address the issue above, we design a multi-branch network based on the human center of gravity (COG) to enhance the robustness of the model to large-pose samples. Specifically, noticing the correspondence between the COG and human pose, by clustering the COG, we separate the large-pose samples from the normal ones in an unsupervised pattern, and lift them with separate branch network. In addition, we introduce a global loss function to regularize the integrality of 3D joints. Extensive experiments on the largest publicly available dataset demonstrate the validity and efficiency of our method.

Index Terms—3D human pose estimation, center of gravity, clustering, multi-branch network

I. INTRODUCTION

3D human pose estimation is a fundamental problem in computer vision thanks to many potential useful real-world applications such as human-computer interaction, virtual reality and sports analysis. Great progress has been made in 3D human pose estimation based on deep learning in recent years. Generally, 3D human pose estimation can be categorized into 2D joints detection and 3D pose regression. While the former step aims to locate accurate joints from a monocular RGB image [1], the latter aims to infer human depth information for each joints [2]–[4], [11], [12]. The merit of the two-stage approach is that it decomposes a complex problem into a combination of two simple problems, so that the pose regression task only needs to analyze 2D joints without considering interference factors such as background, lighting and clothing.

There have been extensive studies on two-stage methods in recent years, mainly focusing on the second step. Martinez et al. [3] propose a simple baseline for 3D human pose regression, proving 'lifting' 2D ground truth joints to 3D space is a task that can be conducted with a remarkably low error rate only utilizing a simple network architecture. Since the 2D joints are stored in a non-square matrix form, which can only be represented in the form of irregular structure, the regression method in [3] only utilizes linear structure without convolutional layer. To address this limitation, Zhao et al. [2] utilize Graph Convolutional Networks [5] to directly deal with the irregular data, boosting the performance of 3D human pose regression with the help of non-local module. Despite the significant progress with deep learning, 3D human pose estimation is still a challenging task due to the variety of human poses. The most widely used human dataset Human3.6M [7] consists of 3.6 million images with 15 actions, whose poses hold a variety of different actions (e.g. Walking and Sitting-Down), and therefore it is hard to fit multiple poses with large deviations using a single mapping function. When dealing with the samples from various distributions, the network will tend to fit majority distribution, while some samples with a small proportion and deviating from the majority distribution may not obtain enough attention, this part of samples will bring a large error, we call them large-pose samples. To address the impact of large-pose samples, learning from the idea of the Gaussian Mixture Model, Li et al. [8] utilize multiple models to predict multiple candidate 3D human poses for each 2D pose, their weighted sum is considered as the most appropriate pose. However, estimating multiple candidate poses for each 2D pose is time consuming. Besides, in the case that large-pose samples are not separated and treated specially, using multiple models does not solve the problem mentioned above, due to the task of each model is still to fit the majority distribution.

Fig. 1. The COG distributions of four action sets in Human3.6M, the pose on the left is a sample from each action sets, We calculate the position of the COG (red point) relative to the root joint (blue point). The red star represents the distribution center.

* This work was supported by the National Science Foundation of China under Grant 61660205. ** represents the corresponding author.
In this paper, we propose a multi-branch network CoGN (Center of Gravity Network) for 3D human pose regression. Unlike the previous methods processing the all 2D poses equally, we utilize an unsupervised method to separate large-pose samples from the normal ones and reason them through separate branch network. To be more specific, we notice that the body’s COG is above the belly button when standing, if performing an action, the COG will shift with the body movement. By following the method in [9], we get the distribution of the COG under four different action sets in Human3.6M as shown in Fig. 1. It is obvious that different actions have different distributions of COG. Based on this observation, we categorize the 2D poses into several classes by adopting K-Means on the COG calculated from 2D joints. Samples from each class have similar COG distributions and pose, and this step requires no additional supervision. Then the samples in each class are mainly processed through a separate branch network. We also customize an initial 3D pose for each class to improve the efficiency and accuracy. In addition, we introduce a COG loss function to constraint the integrality of the 3D joints. Our main contributions lie in the following aspects:

- We control the exploration scope of each branch network by providing an exclusive initial 3D pose respectively, so that each branch can accurately and efficiently lift one class of 2D poses.
- Extensive evaluation on the largest publicly available dataset Human3.6M demonstrates our method can achieve higher estimation precision than the state-of-the-art methods with less training episode.

II. METHOD

A. Overview

Our goal is using 2D input to infer the coordinates of human body joints in three-dimensional space, i.e., to learn a mapping function \( f : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{3n} \). But only using a single mapping function is hard to accurately lift multiple 2D inputs from variety of actions. To this end, inspired by Boosting Method, we utilize multiple mapping functions to deal with the 3D pose regression problem. Each mapping function mainly focus on lifting one class of pose. Finally, all the mapping functions are combined linearly to form a strong regressor \( F \), which is competent for the pose estimation task of all classes:

\[
F = \sum_{j=1}^{C} w_j * f_j^* \tag{1}
\]

s.b. \( f_j^* = \min_{f_j} \frac{1}{N} \sum_{i=1}^{N} \ell(f_j(x_i) - y_i) \)

Where \( x_i \) represents the location of 2D joint, \( y_i \) is the corresponding 3D label, \( C \) is the number of categories we set, and \( f_j^* \) is the corresponding \( C \) mapping functions. We
design a multi-branch network CoGN to implement this idea, multiple branches act as different mapping functions. The workflow of CoGN is shown in Fig. 2. Specifically, at the beginning of the training, we categorize the training dataset by clustering its COG, for each class we calculate an initial 3D pose. During training, each batch of samples will be classified through KNN. Before sending into each branch, the weight of the samples will be dynamically modified according to the task of the current branch. Then each branch performs pose estimation on the basis of initialization. Finally, the results of all branches are combined to give the eventual prediction.

B. Dataset Classification and Preprocessing

Before training, the entire train dataset must be reasonably classified. We observe that the COG shifts with the movement of the body, its position can roughly reflect the human pose, so that we can achieve unsupervised pose classification by clustering the COG. The more categories obtained by clustering means that the classification is more refined, but it will also increase the cost of the multi-branch network. By the way, although Human3.6M has 15 action sets, different action sets have some poses in common (e.g. Sitting and Eating in Fig. 1). So as is shown in Fig. 3, we just cluster the 2D poses into 5 categories according to COG distribution. It can be seen that the COG of the common pose locates around the center, which is shifted to the surroundings in large-pose. Then, we calculate an average 3D pose for each class as the initialization:

\[ \text{InitPose}_j = \frac{1}{N_j} \left( \sum_{i=1}^{N_j} 3D\text{Pose}_i^j \right) \]  

\[ \text{s.b. } j \in \{1, 5\}, \sum_{j=1}^{5} N_j = N \]

where \(3D\text{Pose}_i^j\) is the 3D label corresponding to the \(j\)-class sample, \(N_j\) is the number of samples in the \(j\)-class. It should be noted that during the evaluation, we still use the initial pose generated from the training data without recalculating.

C. Weight Modification

In this part, we make weight adjustments to the sample sent into each branch network. We aim at forcing each branch to focus on processing a specific class of pose, but if only one class of pose is used to train a branch, it will inevitably lead to overfitting. So we still train each branch with the poses from all classes, while adjusting the weight of samples sent to each branch according to its task. For example, as shown in Fig. 2(c), the first branch is mainly responsible for regressing the sitting pose (with yellow border), so the weight of sitting poses in the gradient calculation is increased, while reducing the weight of all other samples:

\[ \text{grad} = \frac{1}{N} \left( \sum_{i=1}^{N^*} W_h * \text{grad}_i + \sum_{j=1}^{N-N^*} W_i * \text{grad}_j \right) \]  

Equation (3) shows how to calculate the gradient of a batch in which \(N^*\) samples belong to the class which the current branch is focus, and the weight of these samples is increased, while the other \(N - N^*\) samples is reduced.

D. Multi-Branch Network

We design a multiple branch network to act as multiple mapping functions for handling different classes of pose, shown in Fig. 2(d). Each branch is similar to the network in [3], as shown in Fig. 4. We set a customize 3D pose as the initialization for each branch respectively, so that the task of each branch changes from reasoning 3D coordinates to fine-tuning the initial 3D pose:

\[ \text{Pose}_{3D} = \lambda * \text{InitPose} + (1 - \lambda) * \text{Pre} \]  

where \( \text{Pre} \) is the prediction of the branch, and the \( \lambda \) is used to weight the proportion of the initial pose in the final prediction. The smaller the \( \lambda \) is, the more the initialization can be fine-tuned.

E. Mixture Module

In this part, we model a linear combination of multiple branch networks to get a strong regressor. For example, when processing a sample, the branch which is focus on its class should present the most accurate prediction result. However, when dealing with some samples which are close to the classification hyperplane or misclassified due to wrong COG calculation, it is helpful to refer to the prediction results of other branches to reduce the prediction error. Therefore, for
each sample, all the branches will be integrated to infer the most accurate pose:

\[
Pose_{3D} = \sum_{j=1}^{5} W_j \ast Pose_j
\]

s.b. \( j \in Z[1, 5], \sum_{j=1}^{5} W_j = 1 \)

where \( Pose_j, j \in Z[1, 5] \) represents the prediction of each branch, \( W_j \) represents the weight of each branch in the current sample prediction.

\[ F. \text{ Loss Function} \]

We train our network on the fully-annotated 3D dataset, so the training loss can be a simple standard Euclidean Loss using ground-truth depth label \( Y_{dep} \), let \( Y_{dep}^{p} \) denote the predicted depth, the loss of joints is:

\[
Loss_j = \left\| Y_{dep}^{p} - Y_{dep} \right\|^2
\]

However, this loss function only constrains the accuracy of each joint but ignores the integrity of all the joints. As mentioned above, the position of the COG can reflect the overall distribution of joints, so we introduce the COG loss as an overall constraint on 3D joints:

\[
Loss_g = \left\| \sum_{i=1}^{N} W_i \ast \text{Joint}_i - \sum_{i=1}^{N} W_i \ast \text{Joint}_i^{p} \right\|^2
\]

where \( W_i \) is the weight of \( \text{Joint}_i \) in COG calculation. In summary, the final loss of our network is:

\[
Loss = Loss_j + \lambda \ast Loss_g
\]

\( Loss_j \) is a direct constraint on each joint, which is the main constraint, while \( Loss_g \) does not act on each joint directly, so its weight \( \lambda \) will be adjusted when it is used.

III. EXPERIMENT

In this section, we first introduce settings and implementation details for training and evaluation, then report our results and compare with state-of-the-art methods.

A. Dataset and Protocol

Our proposed approach is comprehensively evaluated on Human3.6M [7], the largest publicly available dataset for 3D human pose estimation, following the standard protocol.

**Dataset.** Human3.6M is a large-scale dataset consisting of 3.6 million RGB images of 11 different professional actors performing 15 everyday activities, both 2D and 3D ground truth are available for supervised learning. Our method only leverages 2D joints of the human pose as inputs, the ground truth (GT) 2D joints are calculated by camera parameters and 3D ground truth. We also evaluate our approach on the detected 2D joints which are gained by HourGlass (HG) [1], the HG is first pre-trained on MPII [21] and then fine-tuned on Human3.6M. We process the dataset following Zhao et al. [2].

**Protocol.** We follow the standard Protocol#1, using all 4 camera views in subjects S1, S5, S6, S7 and S8 for training, and the same 4 camera views in subjects S9 and S11 for evaluation. Errors are calculated after alignment of the root between ground truth and our prediction.

We utilize the accepted evaluation metric Mean Per Joint Position Error (MPJPE), which is calculated in millimeter between the predicted 3D coordinates and the ground truth after aligning the root joint.

B. Implementation Details

We train our network for 100 epochs using Adam [10], adopting a starting learning rate of 0.001 and exponential decay every four epochs, using mini-batches of size 256. We implement our code using python2.7 on Pytorch, which takes around 10 minutes for one epoch training on the entire Human3.6M dataset in one Titan Xp GPU with CUDA 9.0 and cudnn7. Each branch contains a linear module, as shown in Fig. 4, the keeping probability of dropout is set to 0.5 during training. The \( W_h \) and \( W_l \) in equation (3) are set to 1 and 0.5 respectively. The \( \lambda \) in equation (4) and equation (8) are set to 0.9 and 0.5 respectively.

C. Evaluation on 3D Human pose Regression

Our method uses only 2D joints as input to complete 3D pose regression. We conduct experiments on ground truth 2D joints and 2D detections created by HG respectively. We show the evaluation results under Protocol#1. We compare ours with previous methods on the Human3.6M, some of them aim to exploit temporal information [2], [16], while others simply learn a mapping function to complete the 2D to 3D regression [3]. These methods have their own strengths, some are good at estimating large-pose sample, while others focus on processing in-the-wild pose. Therefore, comparing with them can examine the performance of our method comprehensively. **TABLE I** report the results.

![Fig. 5. Test errors curves of our network with different settings on Sitting-Down action set in Human3.6M.](image-url)
Therefore, we set \( \text{Loss}_{g} \) performs as an overall constraint on joints. We test the \( \text{Loss}_{g} \) on Human3.6M, using GT 2D joints as input, Fig. 6 shows the effect of \( \text{Loss}_{g} \) on network performance. It can be seen that after the introduction of \( \text{Loss}_{g} \), the average MPJPE and the minimum MPJPE in the training phase are reduced, while the stability of training is not affected.

E. comparison with state-of-the-art methods

We compare our method with the previous ones from four aspects: the ability to solve large-pose sample, the robustness of multi-action, the performance upper bound and the stability in the training process.

**Large-pose.** We select the top four action sets of MPJPE in previous methods for comparison: SittingD., Sitting, Phone, Smoke. In two of the above actions, our method works best, and the other two are close to the first place. Our network separate these large-pose poses from the common works best, and the other two are close to the first place. Our network separate these large-pose poses from the common ones according to the COG distribution, therefore, we can make special treatment to the large-pose poses to improve the prediction accuracy.

**Robustness.** Our average MPJPE across all action sets is the lowest among all the mentioned methods, proving that our method is more robust when dealing with variety of actions. Since our network is a linear combination of several weak regressors, and each weak regressor is good at reasoning a specific category of pose. After combination, when processing different poses, the weight of each regressor will be modified dynamically to obtain the best performance.

**Upper bound.** We evaluate our method on the 2D ground truth joints to show an upper bound performance. Our method not only shows a better performance compared with the baseline method [3], but even exceeds the state-of-the-art method. Compared with using 2D detection as input, the promotion is more significant, suggesting the detection error of 2D joints affects the calculation of COG, thus leading to pose...
misclassification, which is not conducive to pose estimation. However, 2D ground truth can ensure the correctness of classification and bring significant performance improvement.

**Stability.** We also show the test errors curves of our networks with different settings and other two methods, as shown in **Fig. 6.** Compared with the other two, our method is significantly more stable, the test error does not show a fluctuation as [2] does. Moreover, the error is relatively low at the beginning of the training (49.35mm vs 64.71mm), which proves that the initial 3D poses we set are reasonable and effective, which can accelerate the training speed.

**F. Qualitative results**

In **Fig. 7**, we select three large-pose action sets from Human3.6M to visualize our results: Posing, SittingDown and Sitting from top to bottom. As seen, our method is able to accurately predict those large-pose samples. It indicates that our CoGN can pay close attention to such actions which are markedly different from the common pose.

**IV. CONCLUSIONS**

In this paper, we propose a novel approach based on COG for 3D human pose estimation. Our method has addressed the large-pose problem which is not well concerned by previous methods. By utilizing COG to classify poses and mixing multiple mapping functions reasonably, our method is competent for handling the multi-pose regression task. Extensive experiments demonstrate our method can boost the training speed and the estimation precision.

**REFERENCES**


