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 SittingDown), 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.

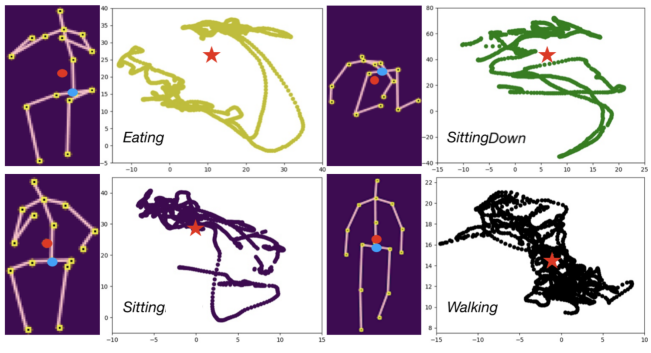

Fig. 1. The COG distributions of four action sets in Human3.6M, the pose on the left is a sample from each action sets, We calculate the position of the COG (red point) relative to the root joint (blue point). The red star represents the distribution center.

* This work was supported by the National Science Foundation of China under Grant 61662059.*(✉) represents the corresponding author.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

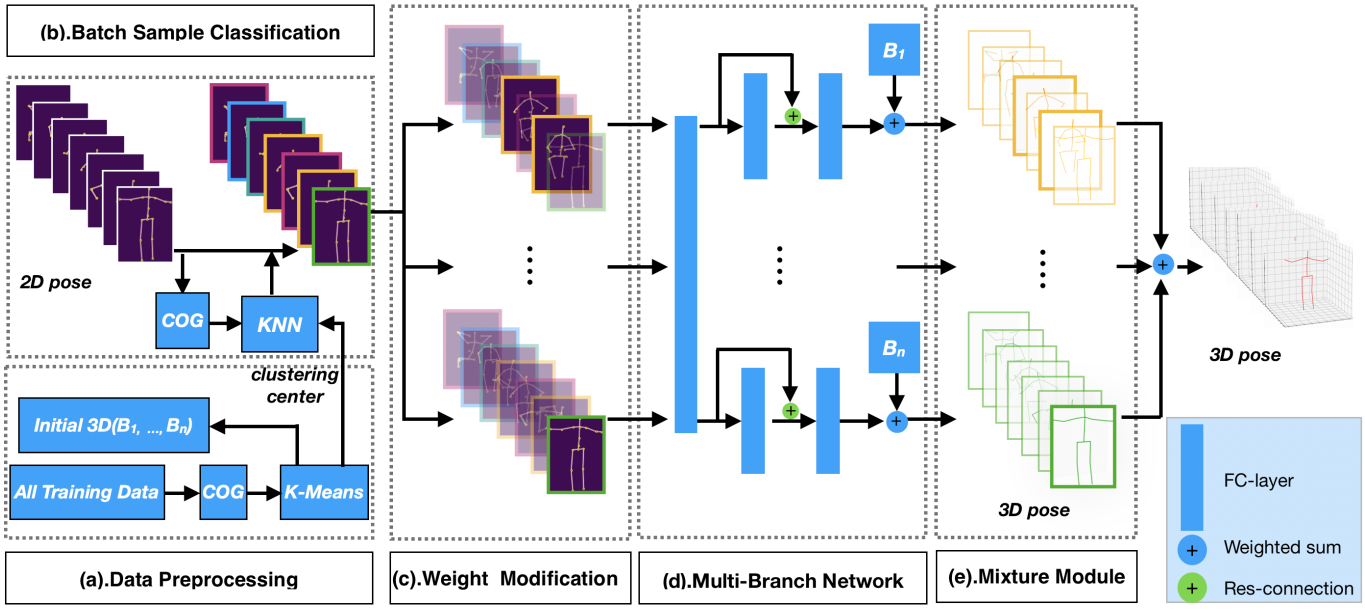

Fig. 2. The workflow of the proposed CoGN framework. (a) Before training, calculate the COG of all samples in the training dataset and cluster them into n categories, then obtain an initial 3D pose for each class. (b) During training, each batch of samples is first classified utilizing KNN according to the clustering center gained in step (a). The color of the border represents the class to which the pose belongs. (c) According to the classification results in (b), the weights of the samples sent to each branch are adjusted separately. The transparency of the sample indicates the level of weight. (d) Each branch according to its own input combined with the initial 3D pose to give a prediction result. (e) The final prediction is a weighted sum of the outputs of each branch.

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 bellybutton 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 customise 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:

- Unlike previous methods lifting all the 2D poses to 3D space through a same mapping function, we utilize an unsupervised method to separate the large-pose samples from the normal ones and lift them with separate branch network.
- To the best of our knowledge, we are the first to classify human pose in virtue of the correspondence between human pose and the COG, so that we can conduct pose estimation for each class separately.
- 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 $f : R^{2n} \rightarrow R^{3n}$. 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:

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

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

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.

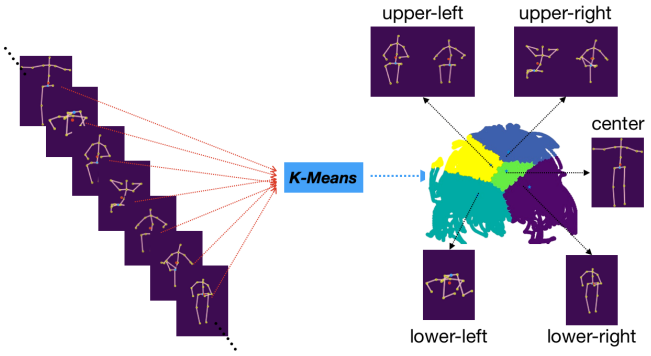

Fig. 3. The training dataset is classified by K-Means according to the relative position of the COG (the red point). The five classes represent the five distribution region of the COG.

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:

$$InitPose_{\,j} = \frac{1}{N_{\,j}} \left(\sum_{i=1}^{N_{\,j}} 3DPose_{\,i}^{\,j} \right) \tag{2}$$

$$s.b. \quad j \subset Z[1,5], \sum_{j=1}^{5} N_{\,j} = N$$

where $3DPose^{\,j}$ 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:

$$grad = \frac{1}{N} * \left(\sum_{i=1}^{N^*} W_{\,h} * grad_{\,i} + \sum_{j=1}^{N-N^*} W_{\,l} * grad_{\,j} \right) \tag{3}$$

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.

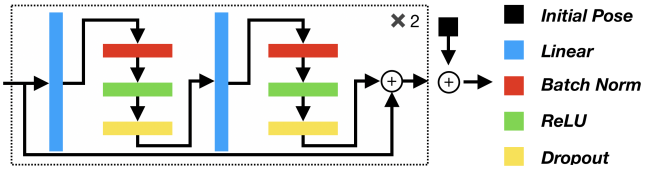

Fig. 4. The architecture of each branch network

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:

$$Pose_{\,3D} = \lambda * InitPose + (1 - \lambda) * Pre \tag{4}$$

where Pre is the prediction of the branch, and the λ is used to weight the proportion of the initial pose in the final prediction. The smaller the λ 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:

$$Pose_{3D} = \sum_{j=1}^{5} W_j * Pose_j \tag{5}$$

$$s.b. \quad j \subset Z[1,5], \sum_{j=1}^{5} W_j = 1$$

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

F. 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^p_{dep} denote the predicted depth, the loss of joints is:

$$Loss_j = \left\| Y_{dep} - Y^p_{dep} \right\|^2 \tag{6}$$

However, this loss function only constraints 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:

$$Loss_g = \left\| \sum_{i=1}^{N} W_i * Joint_i - \sum_{i=1}^{N} W_i * Joint^p_i \right\|^2 \tag{7}$$

where W_i is the weight of $Joint_i$ in COG calculation. In summary, the final loss of our network is:

$$Loss = Loss_j + \lambda * Loss_g \tag{8}$$

$Loss_j$ 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 λ 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 λ 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.

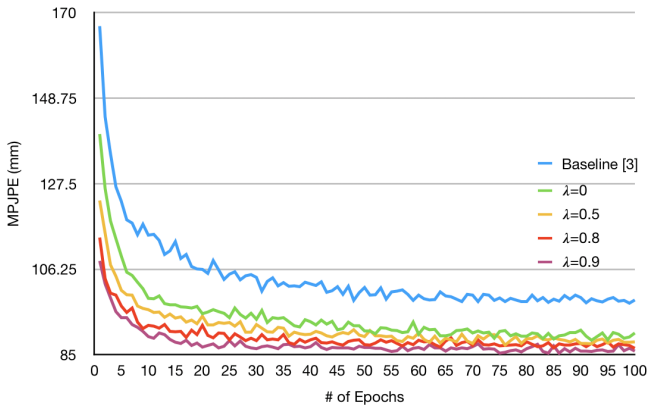

Fig. 5. Test errors curves of our network with different settings on Sitting-Down action set in Human3.6M.

TABLE I

Quantitative evaluations on the Human3.6M under Protocol1(no rigid alignment or similarity transform is applied in post-processing). GT and HG indicate our model is trained with the ground truth 2D pose and 2D detections of HourGlass respectively. The Bold-faced numbers represent the best results.

Protocol#1 (HG)	Direct.	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD.	Smoke	Wait	WalkD.	Walk	WalkT.	Average
Tome *et al.* [11] CVPR'17	65.0	73.5	76.8	86.4	86.3	110.7	68.9	74.8	110.2	173.9	84.9	85.8	86.3	71.4	73.1	88.4
Metha *et al.* [20] TOG'17	62.6	78.1	63.4	72.5	88.3	93.8	63.1	74.8	106.6	138.7	78.8	73.9	82.0	55.8	59.6	80.5
Lin *et al.* [12] CVPR'17	58.0	68.2	63.3	65.8	75.3	93.1	61.2	65.7	98.7	127.7	70.4	68.2	72.9	50.6	57.7	73.1
Pavlakos *et al.* [14] CVPR'17	67.4	71.9	66.7	69.1	72.0	77.0	65.0	68.3	83.7	96.5	71.7	65.8	74.9	59.1	63.2	71.9
Tekin *et al.* [19] ICCV'17	54.2	61.4	60.2	61.2	79.4	78.3	63.1	81.6	70.1	107.3	69.3	70.3	74.3	51.8	63.2	69.7
Jahangiri *et al.* [17] CVPR'17	63.1	55.9	58.1	64.5	68.7	61.3	55.6	86.1	117.6	71.0	71.2	66.3	57.1	62.5	61.0	68.0
Martinez *et al.* [3] ECCV'17	51.8	56.2	58.1	59.0	69.5	78.4	55.2	58.1	74.0	94.6	62.3	59.1	65.1	49.5	52.4	62.9
Zhao *et al.* [2] CVPR'19	**48.2**	60.8	**51.8**	64.0	64.6	**53.6**	**51.1**	67.4	88.7	**57.7**	73.2	65.5	**48.9**	64.8	51.9	60.8
Fang *et al.* [13] AAAI'18	50.1	54.3	57.0	57.1	66.6	73.3	53.4	55.7	72.8	88.6	60.3	57.7	62.7	47.5	50.6	60.4
Sun *et al.* [15] ICCV'17	52.8	54.8	54.2	**54.3**	**61.8**	67.2	53.1	**53.6**	71.7	86.7	61.5	**53.4**	61.6	47.1	53.4	59.1
Our	50.2	**53.2**	55.4	56.2	64.1	72.1	51.9	53.8	**70.0**	87.4	**59.0**	56.3	60.3	**45.8**	**49.3**	**59.0**
Protocol#1 (GT)	Direct.	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD.	Smoke	Wait	WalkD.	Walk	WalkT.	Average
Moreno. *et al.* [18] CVPR'17	53.5	50.5	65.7	62.4	56.9	80.8	60.6	50.8	55.9	79.6	63.6	61.8	59.4	68.5	62.1	62.1
Martinez *et al.* [3] ECCV'17	37.7	44.4	40.3	42.1	48.2	54.9	44.4	42.1	54.6	58.0	45.1	46.4	47.6	36.4	40.4	45.5
Zhao *et al.* [2] CVPR'19	37.8	49.4	37.6	40.9	45.1	**41.4**	**40.1**	48.3	50.1	**42.2**	53.5	44.3	40.5	47.3	39.0	43.8
Lee *et al.* [16] ECCV'18	**34.6**	**39.7**	37.2	40.9	45.6	50.5	42.0	39.4	47.3	48.1	39.5	**38.0**	**31.9**	41.5	37.2	40.9
Wang *et al.* [6] CVPR'19	36.5	42.7	38.2	39.6	45.3	50.8	40.2	**34.8**	**45.0**	50.3	39.4	39.9	42.5	32.2	**33.8**	40.8
Our	35.3	40.8	**35.3**	**38.0**	**40.4**	47.6	41.5	36.3	48.2	49.0	**39.1**	40.3	40.7	**31.0**	33.9	**39.8**

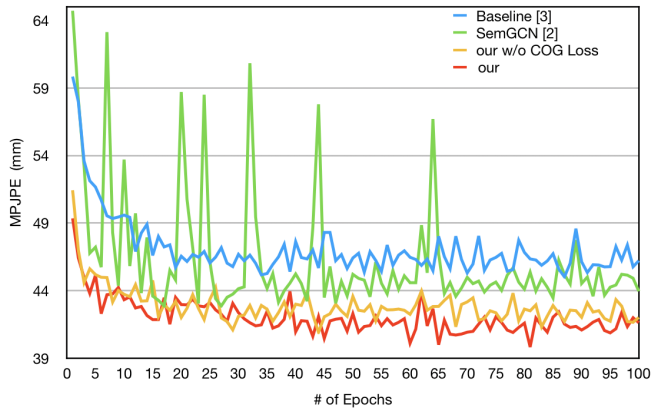

Fig. 6. Test errors curves of our networks with different settings and compare with other two methods.

D. Ablation Study

We calculate the ablation study on the proposed method, mainly focus on the effects of the initial 3D pose $InitPose$ (Sec.II.B) and the integrity constraint $Loss_g$ (Sec.II.E).

InitPose. We set up an initial pose for each class to achieve rapid exploration. To explore the effect of initial pose on network performance, we test our approach with different settings on the SittingDown action set in Human3.6M, using HG 2D joints as input, the result is shown in **Fig. 5**. A larger λ indicates the greater impact the initial pose has on the final result. As λ increases, MPJPE decreases throughout the whole training phase, which proves that a reasonable initial pose can improve the prediction accuracy. However, a large λ also means that the network relies too much on the initial pose and lacks its own exploration which may lead to overfitting. Therefore, we set λ to 0.9, which preserves the exploration ability of the network.

$Loss_g$. Considering the integrity of joints, we utilize $Loss_g$ performs as an overall constraint on joints. We test the $Loss_g$ on Human3.6M, using GT 2D joints as input, **Fig. 6** shows the effect of $Loss_g$ on network performance. It can be seen that after the introduction of $Loss_g$, 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 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

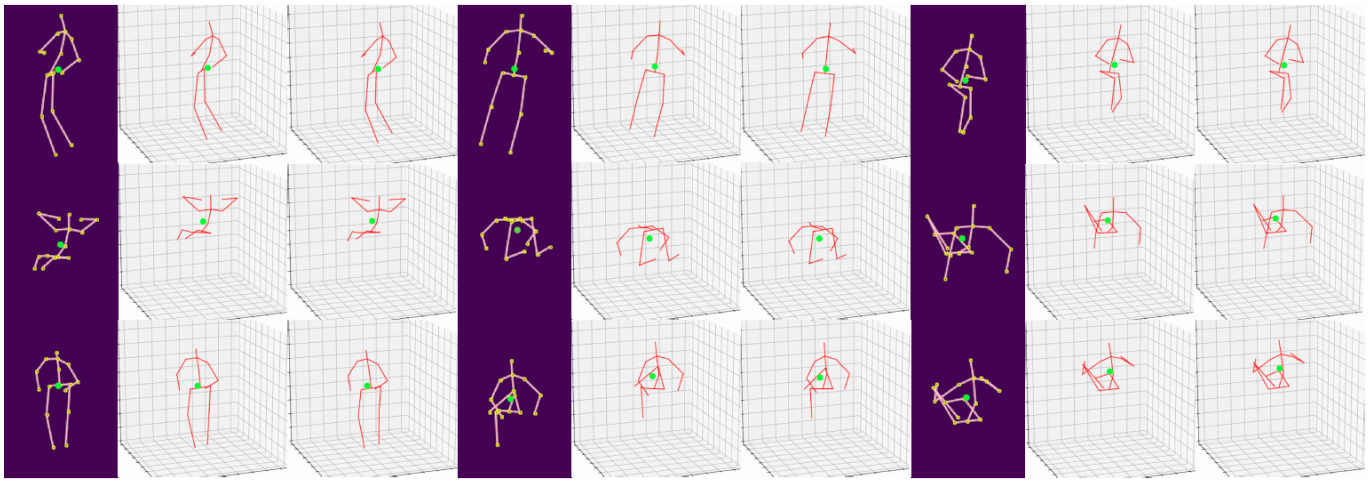

Fig. 7. Example output on the test set of Human3.6M. left: 2D detection. Middle: 3D ground truth. Right: our 3D predictions. The green joints indicates the COG.

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

[1] Newell, Alejandro, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation." European conference on computer vision. Springer, Cham, 2016.

[2] Zhao, Long, et al. "Semantic Graph Convolutional Networks for 3D Human Pose Regression." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[3] Martinez, Julieta, et al. "A simple yet effective baseline for 3d human pose estimation." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[4] Zhou, Xingyi, et al. "Towards 3d human pose estimation in the wild: a weakly-supervised approach." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[5] Gori, Marco, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains." Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. Vol. 2. IEEE, 2005.

[6] Wang, Luyang, et al. "Generalizing Monocular 3D Human Pose Estimation in the Wild." arXiv preprint arXiv:1904.05512 (2019).

[7] Ionescu, Catalin, et al. "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments." IEEE transactions on pattern analysis and machine intelligence 36.7 (2013): 1325-1339.

[8] Li, Chen, and Gim Hee Lee. "Generating multiple hypotheses for 3d human pose estimation with mixture density network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

[9] Wang Wei, Feng Yaqin, Yang Zaixing, et al. "Calculation Method for Human Center of Gravity Based on Somatosensory Interaction". Journal of Data Acquisition and Processing, 2018, 33(4): 595-602.

[10] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[11] Tome, Denis, Chris Russell, and Lourdes Agapito. "Lifting from the deep: Convolutional 3d pose estimation from a single image." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[12] Lin, Mude, et al. "Recurrent 3d pose sequence machines." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[13] Fang, Hao-Shu, et al. "Learning pose grammar to encode human body configuration for 3d pose estimation." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[14] Pavlakos, Georgios, et al. "Coarse-to-fine volumetric prediction for single-image 3D human pose." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[15] Sun, Xiao, et al. "Compositional human pose regression." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[16] Lee, Kyoungoh, Inwoong Lee, and Sanghoon Lee. "Propagating lstm: 3d pose estimation based on joint interdependency." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[17] Jahangiri, Ehsan, and Alan L. Yuille. "Generating multiple diverse hypotheses for human 3d pose consistent with 2d joint detections." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[18] Moreno-Noguer, Francesc. "3d human pose estimation from a single image via distance matrix regression." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[19] Tekin, Bugra, et al. "Learning to fuse 2d and 3d image cues for monocular body pose estimation." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[20] Mehta, Dushyant, et al. "Vnect: Real-time 3d human pose estimation with a single rgb camera." ACM Transactions on Graphics (TOG) 36.4 (2017): 44.
[21] Andriluka, Mykhaylo, et al. "2d human pose estimation: New benchmark and state of the art analysis." Proceedings of the IEEE Conference on computer Vision and Pattern Recognition. 2014