Multi-Task and Multi-Level Detection Neural Network Based Real-Time 3D Pose Estimation

Dingli Luo*† , Songlin Du* and Takeshi Ikenaga *
* Graduate School of Information, Production and Systems, Waseda University,
Kitakyushu 808-0135, Japan
† University of Electronic Science and Technology of China,
Chengdu 611731, China
luodingli@toki.waseda.jp

Abstract—3D pose estimation is a core step for human-computer interaction and human action recognition. However, time-sensitive applications like virtual reality also need this task to achieve real-time speed. This paper proposes a multi-task and multi-level neural network architecture with a high-speed friendly 3D human pose representation. Based on this, we build a real-time multi-person 3D pose estimation system with a single RGB image as input. The network estimates 3D poses from the input image directly by the multi-task design and keeps both accuracy and speed by the multi-level detection design. By evaluation, we show our system achieves the 21 fps on RTX 2080 with only 33 mm accuracy lose compared with related works. We also provide network visualization to prove our network work as we design. This work shows the possibility for a single RGB image based 3D pose estimation system to achieve real-time speed, which is a basement for building a low-cost 3D motion capture system.

I. Introduction

Human pose estimation is an important step for the computer to understand human behavior. Because compared with image or video, abstract 2D or 3D human poses are much easier to analyze. Some works [1], [2] estimate human 2D pose by marking joints' 2D positions for each human on the given input image. Farther more, estimating human pose by giving joints' 3D position in 3D space is a more difficult task. However, human poses in 3-dimensional space contain more information and are not affected by camera projection. Considering this, there are also many works that try to do this. Recent approaches estimate the human's 3D pose in 4 kinds of ways: based on single 2D image input [3], based on multiple images with calibration [2], based on video sequences [4], based on other 2D pose estimation system [5]. Image and video-based methods are more complex and hard to design. However, compared with 2D estimation system based methods, they avoid the time cost and accuracy loss from other systems. For the systems using more than one camera, they always provide more accurate result because of less occlusion. However, this also limited by cost consideration and usage environment.

Our approach only needs one monocular RGB camera without depth and focus on game or entertainment usage like Kinect from Microsoft. This means we want to provide a real-time multi-person 3D pose estimation system with a single RGB image as input. Although this system has a limited detection range and do not perform well on a complicated situation which is the same as Kinect, we only need a regular RGB camera, and our system is much cheaper for customers.

To achieve this goal, a new multi-task and multi-level pose estimation neural network and its corresponding representation is proposed. For the multi-task, it takes RGB image as input and produces 2D joint position, depth information and connection information in the same process. And the multi-level structure avoids repeat process and keep the accuracy. Then a post-processing system generates all human poses based on this information. We proposed a representation of joints as a bridge between neural network and the traditional algorithm, which is easier for both learning and post-processing. After the design part, in order to train this neural network and overcome the lack of 3D dataset in wild space, we provide a mixed training design with a real-life 2D dataset and virtual 3D dataset captured from the game. Finally, we validate the effectiveness by using a virtual dataset to train a task for real-life. We also provide the light-weight neural network design for real-time speed. Our post-processing system design overcomes the lost accuracy.

Our system is general for using in house and the wild. We trained this system on the MSCOCO 2014 [6] and JTA dataset [7]. Then we evaluated that and it can achieve 112mm MPJPE in 21 FPS on GTX 2080ti. We also provide detail profile for an explanation of our neural network. This is important for industry usages.

II. Related Work

A. 2D Real-Time Human Pose Estimation

Although we design for 3D human pose estimation system, we learn many ideas from current 2D human pose estimation systems like the encoding method and the network design.

2D human pose estimation system takes image or video as input and gives the information of each human and its 2D positions of each human's joints. In the beginning, there are image feature based pose estimation systems like [8]. But nowadays, deep learning based methods like

DeepPose [9] has been widely used to achieve higher accuracy. Not only this, but some of these methods also consider to achieve real-time speed. In order to achieve both high speed and high accuracy, researchers think from two directions.

a) Bottom-top solution: This kind of solutions detect each joint first, then connects corresponding joints into humans. These methods are summarised into 'Bottom-up solutions.' Most of the works treat joint position detection as given the probability in the current pixel position. The difficult problem is that the method to connect joints belongs to one person. For example, Open Pose [2] from CMU uses part affinity fields to help the connection. Another one [10] uses the parent joint offset to solve this, which is inspired us to design our representation. Some researchers [11] also use clustering to merge all joints belongs to the same person. Besides, by treating human pose as Graphical Model, deep graph neural networks [12] are used to analyze the possible relationship between joints to find out the belonging [13]. Bottom-top solutions are naturally easy to detect multiple persons in image and can achieve stable high speed.

b) Top-bottom solution: This kind of solutions detect each human in input first by object detection methods like Mask-RCNN [14] and crop each person into image parts. Then it does pose estimation for each human individually. These methods are called "Top-bottom" solutions. Since there is only one person in an image, the connection of joints is easier to deal with. So these methods are used to achieve high accuracy requirement. However, the speed depends on the object detection system and also be affected by human numbers. Alpha Pose [15], [16] is a famous top-bottom solutions.

B. 3D Real-time Human Pose Estimation

Compared with image-space 2D human pose estimation, 3D human pose estimation is a much harder problem. In industry, markers and multi-camera calibration based 3D human pose systems [17] are widely used in motion capture by CG companies. Since the requirement of clothes, markers, spaces and camera number, this is really hard to use as a customer-level solution. Many works try to provide solutions for lower cost, less camera and faster speed. Depending on the input, they can be divided into four kinds.

a) Single 2D image based: These methods only need one camera as input, which means less cost and easy deployment. However, estimating 3D joint position based on 2D image space is actually an under-determined problem because there is more than one answer. Using binocular camera [18] or depth camera [19] to get depth information is one solution. On the other hand, considering the human body structure, even only one RGB camera can calculate the human 3D pose with the highest probability. For example, single-shot multi-person 3D pose estimation [3] system provides an end-to-end neural network which can provide 3D joints positions. Also some works [20] provide 3D geometry models based on parametric human 3D models [21]. All these works show the possibility of estimating 3D human pose based on monocular RGB camera input.

b) Camera calibration based: Camera calibration [22] calculates 3D position based on captured images from different directions. This is widely used in SLAM systems [23]. Since this method is much easier for extending existed 2D human pose estimation system into 3D pose estimation, many 2D estimation works [2] [16] choose this to provide a 3D pose estimation solution. However, this kind of method needs more than one cameras from more than one angles, which is hard to use in the wild environment.

c) Video sequence based: The movement also provides depth information because movement speed is variant based on the distance to the camera. Also, an action sequence contains more information for neural networks to understand human body structure. So many works [24] choose video as input. Some works need another 2D estimation network to transfer the video into a 2D motion sequence, then use to estimate 3D pose. VideoPose3D [4] from Facebook AI is a recent example. These methods can achieve higher accuracy compared with image-based methods. But the processing time of pre-processing and the latency from collecting frames for input makes the speed hard to achieve real-time.

d) 2D pose based: To avoid the complexity of colorful image, many works take 2D poses as input which is produced from other 2D human pose estimation methods. For example, 3D pose baseline [5] takes one human's 2D pose directly then output the 3D pose with joints 3D position. Both the input and output is vector instead of image. To improve the quality. Since the pose vector samples is easy to construct automatically, self-supervised [25] and unsupervised methods [26] are also wildly used to improve the accuracy. However, Converting from the image to the abstract 2D pose causes color information lost. This makes it difficult to benefit from color information to fix the 3D position.

III. Multi-Task and Multi-Level Network for Real-Time Multi-Person 3D Pose Estimation

We design a simple pipeline with one RGB image as input and the 3D pose as output. As shown in Fig. 1, our total pipeline contains only 2 step: a neural network processes the input RGB image and output 2D joint position, connection information and joint depth information in the same time, then a post-processing system combine all things together to generate 3D pose result. Compared with a typical 3D pose estimation system '3D pose baseline' [5], our work uses a specially designed middle representation to avoid losing depth information. This design makes our system finish both 2D and 3D detection in one network which means the speed is higher.

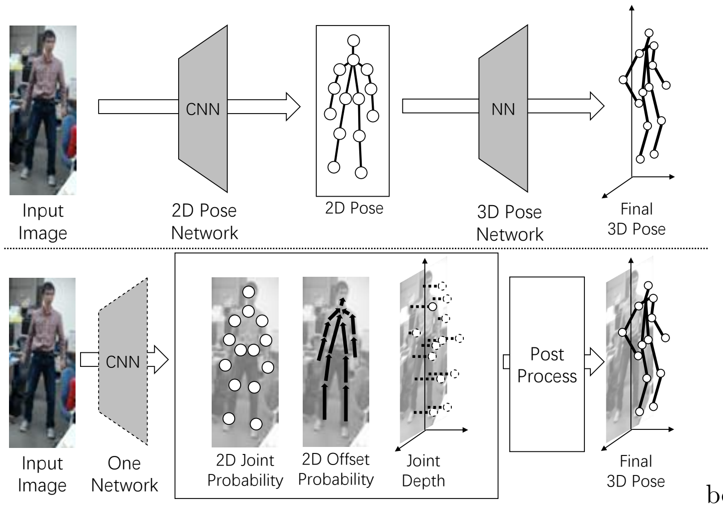

Fig. 1. Concept difference between ours and a typical solution

In this section, we first talk about the representation of pose in our system which determines neural network and post-processing design. Then we talk about each step in our pipeline. A multi-task neural network is used to output 2D pose information and 3D information in the same time. And a multi-level detection structure is applied to keep the speed and accuracy. The second post-processing step deals with detection, linking and automatically matching the pose in 2D image space and depth in 3D world space.

A. Relative Depth Based Pose Representation

The pose representation is like a bridge between the neural network and the post-processing system. So the definition of representation highly effect both the neural network design and the post-processing system design. In our situation, we use CNN as basic module which means the outputs are images. So we define an image based representation for all human poses.

There are many kinds of representation. Some researches [5] directly represent each 3D joint position in the unified 3D space. "unified" means the positions are not related with camera position. Some other researchers consider the pose as a set of parameters to a parametric body model like SMPL [21].

In our case, we consider a human pose as a directional graph. In this graph, each node is corresponding to a human joint j which contains these information:$(Class_j, X_j, Y_j, D_j, OffsetX_j, OffsetY_j)$ as shown in Fig. 2, The $Class_j$ means the current joint's class. The (X_j, Y_j) means the position of current joint$j$. The D_j shows the joint depth which is the distance to camera. The $(OffsetX_j, OffsetY_j)$ is target position of current joint's parent joint. In order to simplify and speed up, we want to make the task of the network as easy as possible. So in our case, we encode the 3D joint positions by two parts: 2D positions(X_j, Y_j) in image space and relative depth(D) in world space. The 2D positions can

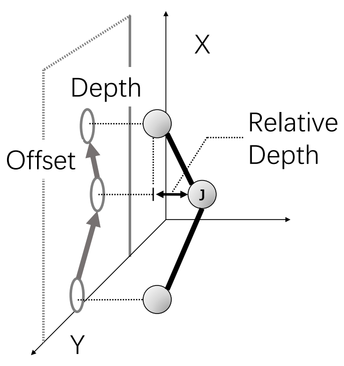

Fig. 2. The encoding method of our pose.

be reused from pose linking process. So the one channel depth(D) is the only additional information needed. Based on the network output, the post-process does a mapping F to transfer to the final camera-related 3D space:

$$F(X_j, Y_j, D) \longrightarrow (X_{3D}, Y_{3D}, Z_{3D}). \quad (1)$$

For the depth, we encode depth value as relative depth. We consider the human pose as a tree structure with the head as the root node. For each joint, the connected joint which is near to head will be the parent node, and the far joints will be the children. Then the relative depth is encoded as the current joint's depth to the parent joint's depth. We make sure each joint only have one parent joint, otherwise, the relative depth is not uniqueness. Compared with the absolute depth, the relative depth is much easier to learn: it is not related to the human position and only needs local area information.

B. Multi-Task and Multi-Level 3D Detection Network

The design target of our network is keeping the accuracy and achieve high speed. Based on this, we provide a new multi-task and multi-level 3d detection network architecture. As we show in Fig. 3, our network architecture contains three parts: the feature pyramid network, 2D detection branch and depth detection branch. The depth detection branch looks almost the same with 2D detection except for the map generation module. In each detection branch, we do detection in different feature level then concatenate together. Finally, a map detection module analysis the concatenate result and output the final map.

1) Multi-Level Detection Network Backbone: The convolutional neural network usually has the best detection target size. So in order to adapt different target size, a normal way is processing the input image many times with different scales. However, in our case, processing the input image more than one time takes a huge time cost. Instead, we use feature pyramid network [27] base multi-level architecture to detect in multiple scales by the network structure. This structure can adapt sizes of the target without higher the calculation cost too much. This is saved without causing accuracy loss because the saved

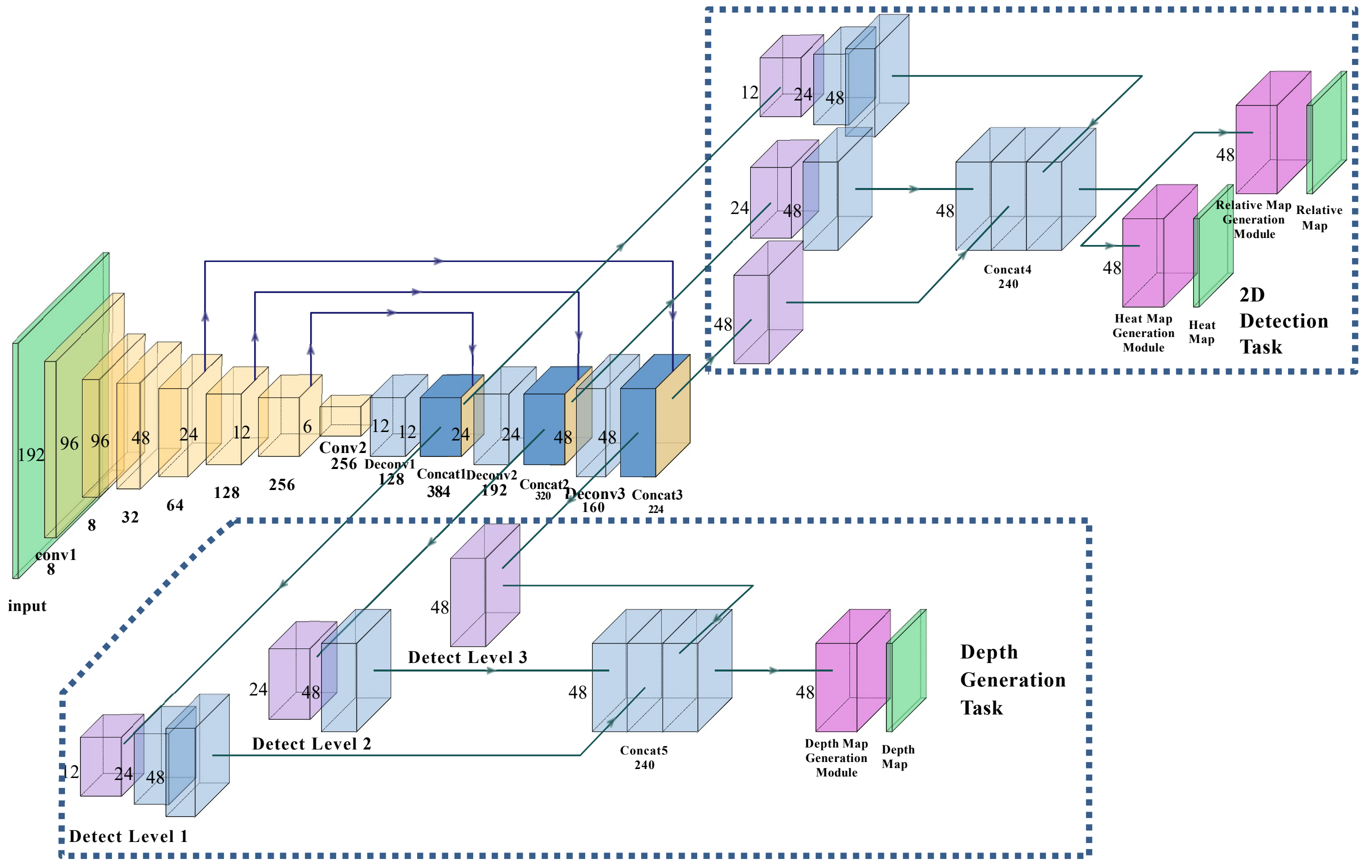

Fig. 3. The overall map of our network architecture.
This shows our network architecture. Firstly, a ResNet34-like backbone processes the input image. Then it is followed by a Deconvolution and concatenate layer. These two structures build a U-Net structure. Then there are two separate branches. One is for 2D detection to output 2D probability map and 2D offset map. The other one is for the 3D depth map. Each detection branch is constructed by a multi-level detection module, a concatenate layer and a map generation module. For the detailed structure, please check Fig. 4

calculation cost is redundancy. For each area in the input image, there is only one best detection level, which means the calculation of other detection levels is not needed.

In detail, we use ResNet34 [28] as the front part but compress the channels into a half. The detailed structure is shown in Table. I. After this, there is an additional convolution layer. Then it is followed by three pairs of deconvolution layer and concatenate layer.

2) Detection Module and Map Generate Module: We design a common structure to use in different levels called detection module and another small module called Map Generate Module to generate the target map. The structure is shown in Fig. 4. Take the map generate module as an example. A convolution layer is applied to the input features in order to decrease the channel number. Then it is processed by a residual structure to extract high-quality features. Finally, in order to merge feature maps from different levels, we must make the size of feature maps become the same. We use deconvolution layer or convolution transpose layer. Each time the feature map processed by deconvolution layer, the map's channel size becomes a half. Compared with using pooling layers, this saves calculation.

All these modules are designed based on the same idea of ResNet [28]. It starts with a 1×1 convolution to smaller the channel number. Then it uses a 3×3 convolution. Finally, it uses a 3×3 convolution to recover the original channel number. The cost is lower than directly doing a 3×3 convolution.

3) Loss Design: As a multi-task neural network, we need to carefully design the loss function in order to balance each different branch. Otherwise, one branch may be weaker than others.

In our case, for classification branch, we directly use L2 loss instead of soft maxed cross entropy loss. Then for the other two branches, we only focus on the position where has a joint instead of the background. For the background area, we directly ignored the loss and allowed the branches to output any result. We find out this makes the task easier and the network can achieve lower loss.

IV. Post-Processing

The output of the network is three multi-channel maps. We need to use post-processing to get the final 3D pose.

Our post-processing progress constructed by four steps:

1) Joint Detection: Use a convolution pass to find out the local maximum point in the output heat map.
2) Joint Linking: Generate a set of joint pairs based on a heat map and relative map. Then sort the joint pairs and calculate the possible joint connection with the highest total probability.
3) Depth Adapting: Calculate the relative scale of world space depth. This process makes the mismatch coordinate space to the same coordinate space.
4) Temporal filter: Depending on the application, if needed, a temporal filter will be used to smooth the 3D pose result.

A. Distance Based Joint Linking

In order to link the joints together, we first calculate the target position:

$$(TargetX_j, TargetY_j) = (X_j + OffsetX_j, Y_j + OffsetY_j)$$

. Then we search a circle area with $(TargetX_j, TargetY_j)$ as center and R_{search} as radius. If there is a corresponding joint with the right class inside the circle, we link current joint to this joint. We calculate the search radius based on the target distance because the farther the parent joint, the higher the error will be. We calculate the R_{search} based on

$$l = \sqrt{OffsetX_j^2, OffsetY_j^2}, \tag{2}$$

$$R_{search} = \max(l * \alpha + (1 - \alpha), 1). \tag{3}$$

In practice, we set the α as 0.3. We find out this makes higher accuracy compared with the fixed joint search radius.

TABLE I
ResNet-like Backbone Structure

layer name	output size	34-layer
input	384×384	
conv1	192×192	7×7, 8, stride 2
conv2_x	96×96	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$
conv3_x	48×48	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$
conv4_x	24×24	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$
conv5_x	12×12	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$

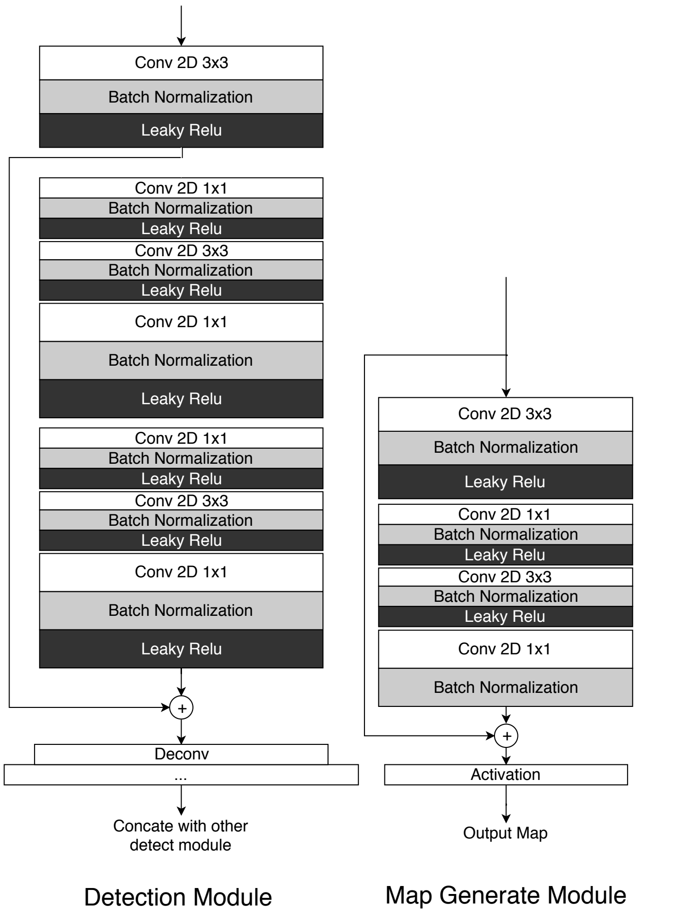

Fig. 4. Detection module and map generate structure
The detection module and the map generate module is constructed by the same residential structure. This structure is made by a Conv 2D 1×1 to decrease the channels into a half. Then a Conv 2D 3×3 is applied to do convolution. Finally, a Conv 2D 1×1 is used to recover the original channel number. The detection module uses this basic structure twice and the map generate module only use once. In order to concatenate with other levels output, a set of Deconv layers are used to make the output feature map have the same width and height.

B. Regression Based Automatic Depth Scale

A core part is how to adapt the depth scale based on 2D detected result. We want to scale the depth with a value $Scale_{depth}$ to make the coordinate in the same unit. In theory, the depth is based on camera position. And the 2D coordinate of joint also needs camera information. This means without camera information, we cannot map the world space depth to camera space.

$$depth = \sqrt{(P_x - C_x)^2 + (P_y - C_y)^2 + (P_z - C_z)^2}, \tag{4}$$

$$\begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 1 & {}_r 12 & r_1 3 & t_1 \\ r_2 1 & r_2 2 & r_2 3 & t_2 \\ r_3 1 & r_3 2 & r_3 3 & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{5}$$

However, we find out a map between the human size and depth scale. This assumes the human size is normal. The relationship between human 2D size ($Width, Height$) and $Scale_{depth}$ is strong both proved by linear regression and visualization.

Then we use a fast calculation to get the $Scale_{depth}$ and multiply to the $Depth$. Then we can get the right human joint position in camera space without getting camera information.

V. Experiments

A. Training

A huge problem for training 3D pose estimation neural network is the dataset. Capturing high-quality 3D joints data usually need to do in the studio and the characters need to wear special clothes with markers. This means it is tough to get 3D pose dataset in the wild environment. However, it is much easier to get the 3D pose from the game because it is needed to render the character. So we choose to use a virtual 3D pose dataset called JTA dataset [7]. Although the input image's realistic is not as good as the dataset from real life, we proved that with carefully designed training progress and data augmentation, a game dataset is also suitable for training a neural network which will be used in real life.

We train the 2D detection branch of the network with MSCOCO 2014 [6] dataset first to initialize the network. Then, we randomly choose a sample from 2D dataset or 3D dataset each time. We recognized this training progress makes the network training more stable and decrease the training time. We train the whole network on one Nvidia GTX 1080ti for 72 hours.

B. Results

We evaluated our method on JTA dataset test set. We use MPJPE (mean per joint position error) to measure our network's performance. The result is shown in Table. II. Since we want to test the image based performance, we do not use temporal smoother. Since there are not many image-based 3D pose solution also considering speed, we compared with some other solutions. For other RGB image based solutions, they do not target for speed and cannot achieve real-time. Some methods also used 2D pose, but it takes a 2D pose as input. This means it needs a pre-processing by another 2D pose network. This means the quality and the speed of the pre-processing method highly affect the performance of the followed 3D pose estimation methods. We test these methods by adding a typical 2D pose estimation system OpenPose [2] to measure the speed and accuracy.

However, many cases of the test dataset are not suitable for our design target. Like there are crowd peoples in a different size. So we also test our work on a test set more similar to game or entertainment. The result images are listed in Fig. 5.

By analysis, we find out most error is happened on the foot. Considering about our relative depth encoding method, the farther the joint to the root node, the higher the error will be.

C. Network profile and visualization

We design this neural network architecture based on the idea of multi-level detection. However, it is hard to prove the neural network acts as we designed because there isn't any manually writing code or logic. In order to prove the neural network is acted as we designed, we use network profile and visualization to prove this.

To prove different level have different focus area, we dump the output feature maps in each level, then find out the area with the highest output in total.

$$Map_{Activation}(x, y) = \frac{\sum Map_{Feature}(x, y, i)}{\max(Map_{Feature}(x, y))}. \quad (6)$$

The higher activation means the current level has higher interest on this area. Like we show in Fig. 6, we find out a different level of detection path do as we want. The high level with small detection receptive field focuses on small targets like necks and human heads far away from the camera. The low level with large detection receptive field focuses on large body parts near to the camera. The middle level takes responsibility for other areas. The interest detects areas in different level do not cover each other. This proves our network act as we design to do.

VI. Conclusions

This paper proposed a system to do real-time multi-person 3D pose estimation. It contains a multi-task and multi-level detection neural network to directly take an

TABLE II
Performance compare

Methods	Input	MPJPE (mm)	Speed (ms)
Zhou et al.[24]	Video	64.9	-
Martinez et al.[5]	2D Pose	62.9	-
Open Pose[2] (high accuracy mode) + Martinez et al.[5]	RGB Image	70.5	833.3
Open Pose[2] (high speed mode) + Martinez et al.[5]	RGB Image	82.7	113.6
LCR-Net[29]	RGB Image	87.7	-
Ours	RGB Image	112.9	46.5

- Means this work is not designed for speed or cannot measure by FPS. For example, the 2D pose based work's speed depends on the pre-processing system's speed.

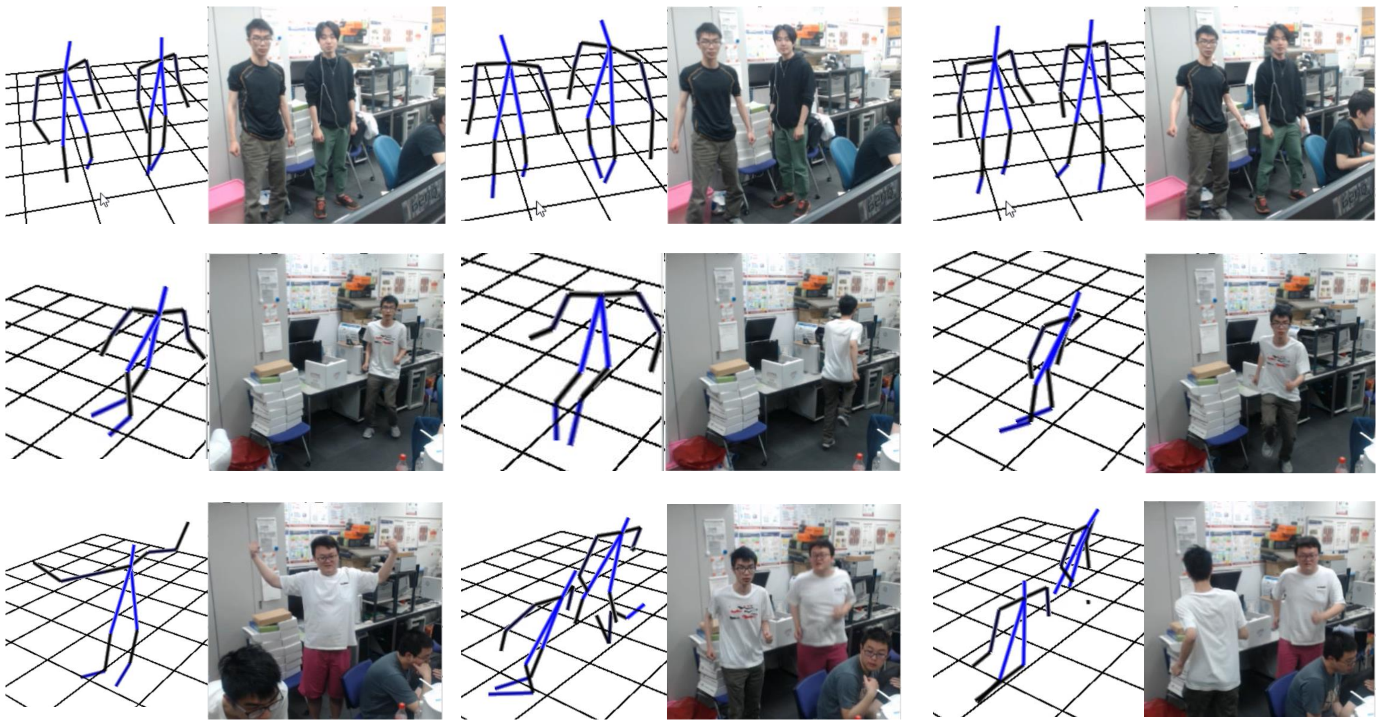
Fig. 5. Some results of our work, include single-person and multi-person pose estimation result.

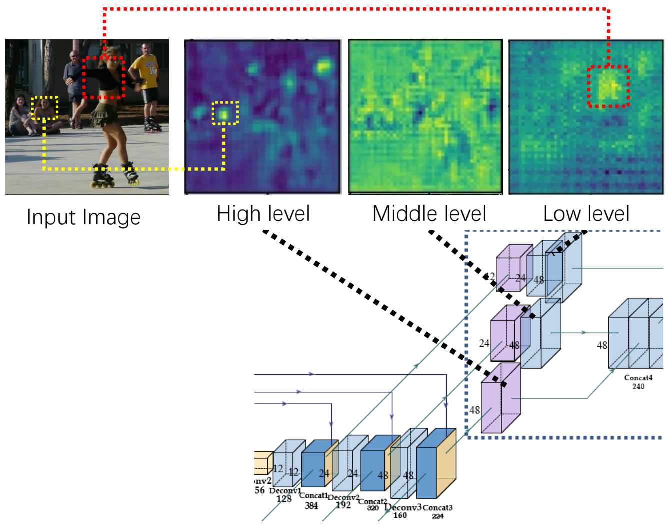

Fig. 6. Activation area for different detection level.
This image shows the division of work for different levels: The high-level focus on small detection targets like neck and human head far away from the camera. The middle-level focus on most parts of the image. The low-level focus on the large parts in the image like the human arm of the person near to the camera.

image as input and output 2D joint position, 2D joint linking and 3D joint depth information. Then a post-processing progress is used to construct 3D pose for each human. With only 33 mm accuracy down, our RGB image based solution achieves 21 fps speed on RTX 2080. This is an affordable accuracy for game and entertainment. Our system shows the potential of using a normal RGB camera to do real-time 3D pose estimation for multiple people is possible.

Acknowledgment

This work was supported by Waseda University Grant for Special Research Projects (2019C-581)

References

[1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in CVPR, 2017.
[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields,” in arXiv preprint arXiv:1812.08008, 2018.
[3] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, “Single-shot multi-person 3d pose estimation from monocular rgb,” in 2018 International Conference on 3D Vision (3DV), pp. 120–130, IEEE, 2018.
[4] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, “3d human pose estimation in video with temporal convolutions and semi-supervised training,” in Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[5] J. Martinez, R. Hossain, J. Romero, and J. J. Little, “A simple yet effective baseline for 3d human pose estimation,” in ICCV, 2017.
[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in Computer Vision – ECCV 2014 (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.
[7] M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara, “Learning to detect and track visible and occluded body joints in a virtual world,” in European Conference on Computer Vision (ECCV), 2018.

[8] T. Vatahska, M. Bennewitz, and S. Behnke, "Feature-based head pose estimation from images," in 2007 7th IEEE-RAS International Conference on Humanoid Robots, pp. 330–335, IEEE, 2007.

[9] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[10] S. D. Dingli Luo and T. Ikenaga, "End-to-end feature pyramid network for real-timemulti-person pose estimation," in MVA, 2019.

[11] X. Nie, J. Feng, J. Xing, and S. Yan, "Pose partition networks for multi-person pose estimation," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 684–699, 2018.

[12] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[13] X. Chen and A. L. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 1736–1744, Curran Associates, Inc., 2014.

[14] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," 2017.

[15] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in ICCV, 2017.

[16] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose Flow: Efficient online pose tracking," in BMVC, 2018.

[17] A. Sharifi, A. Harati, and A. Vahedian, "Marker based human pose estimation using annealed particle swarm optimization with search space partitioning," in 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 135–140, Oct 2014.

[18] D. C. Blumenthal-Barby and P. Eisert, "High-resolution depth for binocular image-based modeling," Computers & Graphics, vol. 39, pp. 89–100, 2014.

[19] Z. Zhang, "Microsoft kinect sensor and its effect," IEEE MultiMedia, vol. 19, pp. 4–10, Feb 2012.

[20] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele, "Neural body fitting: Unifying deep learning and model based human pose and shape estimation," in 2018 International Conference on 3D Vision (3DV), pp. 484–494, IEEE, 2018.

[21] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," ACM Trans. Graphics (Proc. SIGGRAPH Asia), vol. 34, pp. 248:1–248:16, Oct. 2015.

[22] M. Drennan, "An implementation of camera calibration algorithms," Clemson University, 2010.

[23] D.-x. Zhu, "Binocular vision-slam using improved sift algorithm," in 2010 2nd International Workshop on Intelligent Systems and Applications, pp. 1–4, IEEE, 2010.

[24] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3d human pose estimation from monocular video," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4966–4975, 2016.

[25] M. Kocabas, S. Karagoz, and E. Akbas, "Self-supervised learning of 3d human pose using multi-view geometry," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[26] C.-H. Chen, A. Tyagi, A. Agrawal, D. Drover, R. MV, S. Stojanov, and J. M. Rehg, "Unsupervised 3d pose estimation with geometric self-supervision," arXiv preprint arXiv:1904.04812, 2019.

[27] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125, 2017.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[29] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3433–3441, 2017.