Ego3DPose: Capturing 3D Cues from Binocular Egocentric Views

Taeho Kang
Seoul National University
Seoul, South Korea
taeho.kang@hcs.snu.ac.kr

Kyungjin Lee
Seoul National University
Seoul, South Korea
jin11542@snu.ac.kr

Jinrui Zhang
Central South University
Changsha, China
zhangjinruicsu@gmail.com

Youngki Lee
Seoul National University
Seoul, South Korea
youngkilee@snu.ac.kr

ACM Reference Format:
Taeho Kang, Kyungjin Lee, Jinrui Zhang, and Youngki Lee. 2023. Ego3DPose: Capturing 3D Cues from Binocular Egocentric Views. In *Proceedings of SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers'23)*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3610548.3618147

A DATA PREPROCESSING

A.1 UnrealEgo

The full dataset of UnrealEgo [Akada et al. 2022] is utilized. We use the publicly available preprocessing, data loading, and training code. The dataset provides ground truth 2D and 3D poses, which we utilize to generate the ground truth for Perspective Embedding Heatmaps.

A.2 EgoCap

We only use the validation dataset for our evaluation. The 3D annotations for the training dataset are not publicly available.

The validation set contains 2D and 3D versions. The 2D set contains all of the images in the 3D set. Thus, we use images from the 3D set, and the ground truth 2D joint position is obtained from the 2D set annotation, while the ground truth 3D pose is gathered from the 3D set.

The EgoCap dataset's original image dimension is 1280×1024 as shown in Fig.1. A large portion of the image is empty. The full view of the camera approximately covers a circle of 512-pixel radius. In the prepossessing, images are horizontally cropped, to discard the out-of-view area. The horizontal focal center is placed to be the center of the cropped image, resulting in a 1024×1024 image. To fit our system's input size, the cropped image is then downsampled to a 256×256 image.

The ground truth pose is adjusted to fit the unit used by the UnrealEgo. The UnrealEgo uses a unit length of centimeters for the ground truth pose, while the EgoCap dataset uses millimeters. EgoCap dataset consists of poses for 18 joints, including the head. But following the EgoCap paper, the head 3D pose is not estimated, resulting in a total of 17 joints, and 16 limbs. Note that the UnrealEgo dataset uses a different set of joints, which consists of 16 joints including the head.

In the real-world setting, the relative 3D transform of two cameras consists of the rotational component, thus the local pose is not the same for the two cameras. However, the 3D pose is provided only for the first camera's coordinates, so we use the same view-plane angle and 3D orientation computed in the first camera's coordinate system for both views.

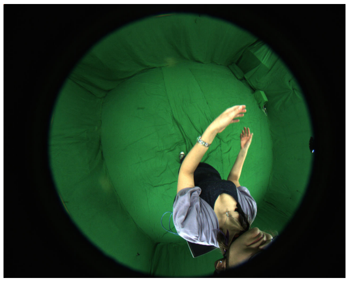


Figure 1: Sample of 1280×1024 EgoCap dataset image.

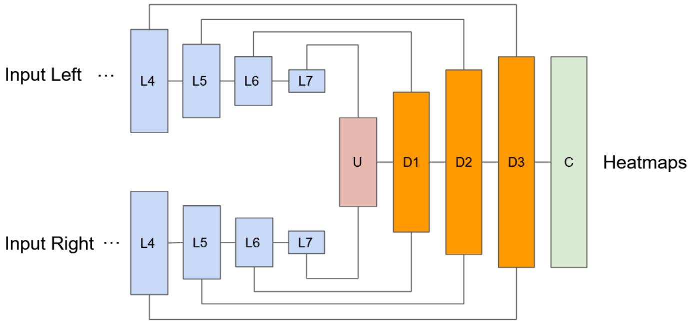


Figure 2: The architecture of the Optical Feature Extractor

B NETWORK DETAILS

Our implementation is based on the open-sourced UnrealEgo's implementation. Each of the layers described as a convolutional, deconvolutional[Zeiler et al. 2010], and fully connected layer is followed by a batch normalization layer and a leaky ReLU with a negative slope of 0.2.

SA Conference Papers'23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12…$15.00
https://doi.org/10.1145/3610548.3618147

Table 1: The hyperparameters for the decoder layers in the optical feature extractor.

Layer	Dimension	Input Channels	Output Channels
D1	16×16	3072	2048
D2	32×32	2560	1024
D3	64×64	1280	1024

B.1 Optical Feature Extraction

Two networks for the Joint Position Heatmaps and the Perspective Embedding Heatmap with the same architecture are trained for optical feature extraction. The ResNet-18 [He et al. 2016] in the U-Nets [Ronneberger et al. 2015] of the optical feature extractors are initialized with pre-trained weights ImageNet1K_1[Deng et al. 2009] available on PyTorch [Paszke et al. 2019].

Two ResNet are used in one U-Nets for each optical feature extractor to take stereo input images. In the torch's ResNet-18 implementation, the output of base layers of index 4, 5, 6, and 7 are concatenated to the U-Net's decoder parts, as shown in Fig. 2. The output is processed by an 1 by 1 convolutional layer, to the same number of channels. In the decoder, the features from two ResNet base layer 7 are concatenated and upsampled after the process. The next layers take the concatenated processed output from two ResNets and the upsampled features from the previous layer. Each layer consists of one convolutional layer with the specified number of channels of kernel size 3, and an upsample layer following that. Table 1 describes the total input channels (the upsampled feature channels and the concatenated ResNet features) and output channels. Finally, one convolutional layer (C in Fig. 2) takes the D3 layer's output and outputs heatmaps, using a kernel size of 1.

B.2 Heatmap Encoder

The heatmap encoder consists of 3 convolutional layers and 3 fully-connected layers. The first convolutional takes all of the heatmaps. Each convolutional layer has 64, 128, and 256 channels of features, using a kernel size of 4, stride of 2, and padding of 1. The output of 256 channels of features is flattened and processed by the following fully connected layers. Each fully connected layer has an output size of 2048, 512, and 20. The 20 is the size of the embedding vector used by the 3D Pose Decoder, and the Heatmap Reconstructor.

B.3 3D Pose Decoder

The 3D pose decoder consists of 3 fully connected layers. The first layer takes 20-dimensional embedding from the Heatmap Encoder and 14 by 3 estimated orientations from the Stereo Matcher, flattened and concatenated together as a vector. The first two layers output 32-dimensional embeddings, and the last layer outputs 16 by 3 estimated 3D pose.

B.4 Stereo Matcher

The stereo matcher module has a similar architecture to the combination of the Heatmap Encoder and the 3D Pose Decoder, with different input, intermediate embedding, and output sizes. The first difference is that it takes only 4 channels of heatmaps, the one set of Perspective Embedding Heatmaps. The output embedding size, from the Heatmap Encoder-like architecture is 10. The final decoder's output is a 3-dimensional vector, which corresponds to the estimated 3D orientation.

B.5 Heatmap Reconstructor

The Heatmap Reconstructor consists of 3 fully-connected layers and 3 deconvolutional layers. The fully connected layers take 20-dimensional embeddings and output 512, 2048, and one last 16384-dimensional vector that is reshaped to 256 channels of 8 by 8 features. The deconvolutional layer outputs 128, 64, and the total number of heatmaps channels in order. All of the deconvolutional layer uses a kernel size of 4, stride of 2, and padding of 1. The deconvolutional layer corresponds to PyTorch's torch.nn.ConvTranspose2d module.

C LIMITATIONS AND FUTURE WORKS

There still remain several limitations. Occlusion in the egocentric pose estimation is still a challenging problem as many motions suffer from high occlusion, especially in the lower body. To deal with it, temporal optimization of the output poses is an important direction [Wang et al. 2021]. Additional inverse kinematics methods can be also useful for virtual character applications.

Secondly, the trained network can overfit the camera's distortion used in the training dataset. The 2D-to-3D lifting is an inherently ambiguous problem, without given camera parameters. Many egocentric methods focus on shared camera setups for 3D pose estimation. However, individual camera's distortion patterns may vary and the method can exhibit larger errors on cameras with different distortions. The Stereo Matcher network can introduce reliance on the binocular camera's configuration, as it attempts to estimate 3D pose from stereo correspondences. Recently, a 2D-to-3D lift-up model applicable for different camera optics and setups is suggested [Miura and Sako 2022]. Such a generalizable framework that applies to various egocentric camera setups is a promising direction for future work.

Lastly, our real-world evaluation has several limitations in its variety, as a result of experimenting only on publicly available dataset. Evaluation in the real-world setting EgoCap dataset has a limited number of subjects and frames since we evaluated only the portion of the dataset with 3D pose annotations. It also lacks a variety of motion types, as it consists of activities while standing. Finally, due to the difficulty of egocentric dataset collection, the dataset is captured only in a lab environment with a green screen. The performance can further be experimented with more comprehensive real-world datasets.

D ADDITIONAL EXPERIMENTS

D.1 Impact of Using Joint Position Heatmap

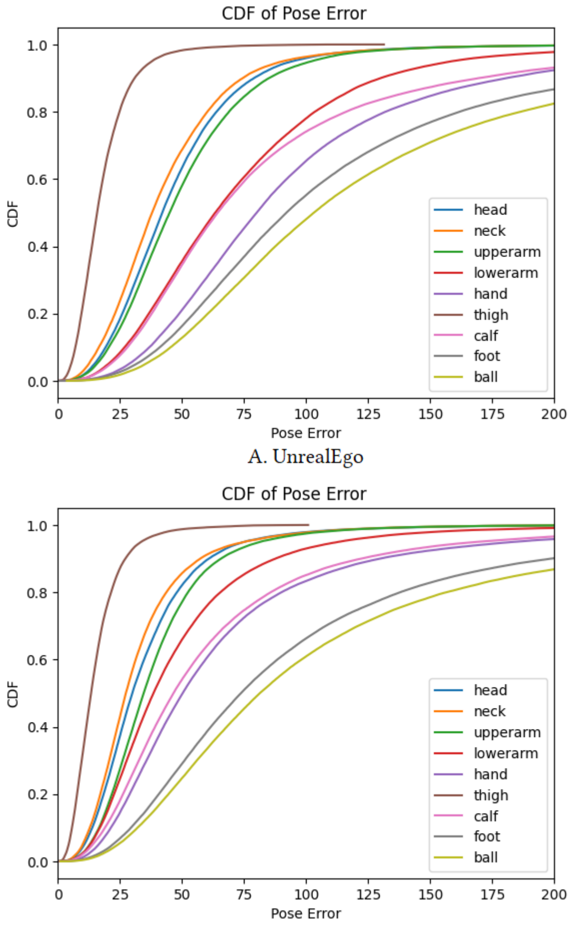
We show the impact of using the Joint Position Heatmap when it is used together with the Perspective Embedding Heatmap. We experimented with our system with only one type of heatmap and both. For the Joint Position Heatmap-only experiment, the result in *Effectiveness of Perspective Embedding Heatmap* section is used. The result in Table 2 reveals that using only Perspective Embedding Heatmap outperforms the method using only Joint Position Heatmaps by 4.7% in MPJPE, and using both outperforms the latter by 8.8%. Perspective Embedding Heatmap contains joint position information when it successfully estimates the 3D information. Its confidence is directly connected to its estimate of 3D angle. Thus, if the estimation of the 3D angle fails, the network may not output meaningful values for the positional estimate, even though the visual cue is available for the 2D position. In those cases, the traditional Joint Position Heatmap provides a fall-back option by focusing on extracting 2D information.

Table 2: Comparison of results on UnrealEgo dataset, using our system with Joint Position Heatmap (JH) only, Perspective Embedding Heatmap (PH) only, and both.

Heatmaps Error	JH	PH	JH + PH
MPJPE	66.72	63.60	**60.82**
PA-MPJPE	52.29	49.49	**48.47**

D.2 Per Joint Error Distribution

We plot the distribution of pose estimation error on the UnrealEgo dataset for two systems, UnrealEgo [Akada et al. 2022] and our Ego3DPose, in Figure. 3. In this plot, we combined the results of corresponding joints on the left and right as distinct samples for one category. "upperarm", "lowerarm", and "hand" corresponds to upper body joints, and "thigh", "calf", "foot", and "ball" are lower body joints. The distribution is visualized as a Cumulative Distribution Function (CDF). As previous works [Tome et al. 2019][Zhao et al. 2021] suggest, lower body parts generally had larger estimation errors. In the experiment, however, the estimation of "thigh" appears to be accurate. This is due to the local pose's definition of the UnrealEgo dataset. The local pose is relative to the pelvis' position, and since the thighs are directly connected to the pelvis, it is easy to estimate its position. Our method shows more improvement on the upper body, which has more visibility than the lower body, particularly noticeable when comparing "lowerarm", "hand", and "calf". It indicates that our method is more effective at extracting visible cues.



A. UnrealEgo

B. Ego3DPose

Figure 3: CDF of joint pose estimation error of the UnrealEgo (A) and Ego3DPose (B) in mm unit.

REFERENCES

Hiroyasu Akada, Jian Wang, Soshi Shimada, Masaki Takahashi, Christian Theobalt, and Vladislav Golyanik. 2022. UnrealEgo: A New Dataset for Robust Egocentric 3D Human Motion Capture. In *European Conference on Computer Vision (ECCV)*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV, USA) *(CVPR '16)*. IEEE, 770–778. https://doi.org/10.1109/CVPR.2016.90

Teppei Miura and Shinji Sako. 2022. Simple yet Effective 3D Ego-Pose Lift-up Based on Vector and Distance for a Mounted Omnidirectional Camera. *Applied Intelligence* 53, 3 (may 2022), 2616–2628. https://doi.org/10.1007/s10489-022-03417-3

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. http://arxiv.org/abs/1505.04597 cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015.

Denis Tome, Patrick Peluse, Lourdes Agapito, and Hernan Badino. 2019. xR-EgoPose: Egocentric 3D Human Pose from an HMD Camera. In *Proceedings of the IEEE International Conference on Computer Vision*. 7728–7738.

Jian Wang, Lingjie Liu, Weipeng Xu, Kripasindhu Sarkar, and Christian Theobalt. 2021. Estimating Egocentric 3D Human Pose in Global Space. arXiv:2104.13454 [cs.CV]

Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. 2010. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2528–2535. https://doi.org/10.1109/CVPR.2010.5539957

Dongxu Zhao, Zhen Wei, Jisan Mahmud, and Jan-Michael Frahm. 2021. EgoGlass: Egocentric-View Human Pose Estimation From an Eyeglass Frame. In *2021 International Conference on 3D Vision (3DV)*. 32–41. https://doi.org/10.1109/3DV53792.2021.00014