Solution of Multiview Egocentric Hand Tracking Challenge ECCV2024

Minqiang Zou , Zhi Lv, Riqiang Jin, Tian Zhan, Mochen Yu, Yao Tang, Jiajun Liang*
Jiiov Technology
{minqiang.zou, zhi.lv, riqiang.jin, tian.zhan
mochen.yu, yao.tang, jiajun.liang}@jiiov.com

Abstract

Multi-view egocentric hand tracking is a challenging problem and plays an important role in VR interaction. In this report, we introduce a method that utilizes multi-view input images and the extrinsic parameters of the cameras to estimate both the hand shape and hand pose. To alleviate the overfitting to the camera layout, we adopt crop jittering and extrinsic parameter noise augmentation. Moreover, we propose an off-line neural smooth post-process method to further improve the position and pose accuracy. Our method achieves 13.92mm MPJPE on Umetrack dataset and 21.66mm MPJPE on HOT3D dataset.

1. Introduction

Commercial headsets for virtual reality usually have multiple cameras for hand tracking task to get a larger interacting area. In the overlap filed of view between cameras, it's able to obtain multi-view images of hands. While in some area near the border of the interacting area, we can only get a single-view image of hands. So it's important for hand tracking system to be able to handle both cases.

We design an architecture including a feature extracting part, a feature fusion part and several regression parts to realize processing both the input types. Specifically, images are fed into the feature extractor and the feature lifting module separately getting the 3d features. For single-view input, the feature is directly passed to the regression heads to estimate the hand pose and hand shape. For multi-view input, the features are firstly fused with feature transform layer (FTL) module and then continue the remaining regressions.

We find that the 2d landmark estimation is usually better than the 3d estimations like hand position and hand pose. We propose an auxiliary post-process method to optimize the 3d estimations based on this observation.

In a world, our work can be summarized into two parts: a unified architecture which handles both single-view input and multi-view inputs; a offline optimizing post-process step which further improves the hand tracking performance.

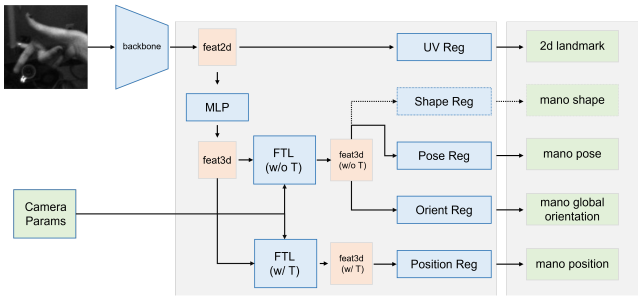

Figure 1. Our method is flexible and supports both monocular and multi-view inputs. The figure shows the monocular process. After feature extraction by the backbone, 2D features (feat2d) are obtained and fed into the UV regressor to get 2D points. Simultaneously, feat2d is processed through an MLP to generate 3D features (feat3d). Then we use the FTL module proposed in [2] to produces features without translation (feat3d(w/o T)) for predicting shape, pose, and global orientation, and features with translation (feat3d(w/ T)) for predicting position.

More details will be explained in the following section.

2. Method

Datasets and Preprocessing In our approach, we exclusively utilized the Umetrack[2] and HOT3D[1] datasets provided for the competition. Preprocessing involved the use of the hand_tracking_toolkit[2] to perform perspective cropping. To enhance the diversity of viewpoints in our dataset, we applied perturbations to the field of view (FOV) and rotation parameters specified in the provided perspective crop camera parameters. Additionally, to facilitate the training process for both single-hand and hand-hand interaction scenarios, all left-hand data were flipped to appear as right-hand data.

Our data augmentation strategy primarily included the following techniques: crop jittering, random brightness and contrast adjustments, blurring, and Gaussian noise addition. Additionally, to enhance the model's generalization capability to different extrinsic camera parameters, we introduced random noise in the range of $(-0.5, 0.5)$ to the

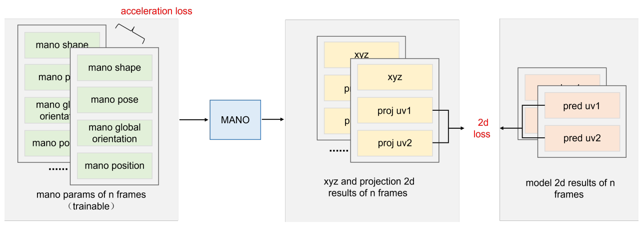

Figure 2. To maintain temporal consistency, we optimize the model's predictions over a sequence as trainable parameters. Simultaneously, we address the issue of inaccurate positioning caused by poor generalization to different extrinsic parameters by employing a 2D projection loss.

translation component of the camera extrinsics.

Model Structure Our Model is highly flexible, accommodating both monocular and multi-view inputs. The Fig1 illustrates a monocular workflow: after extracting features with the backbone, we obtain 2D features (feat2d). These are then fed into the UV regressor to derive 2D points. Concurrently, feat2d is processed through an MLP to produce 3D features (feat3d).

Due to the significant impact of large variations in the translation component(T) of the extrinsic parameters on model predictions, we have decomposed the FTL module into two parts: translation-invariant FTL and translation-variant FTL. Using the FTL module, we generate both translation-invariant features (feat3d w/o T) and translation-variant features (feat3d w/ T). The feat3d w/o T is utilized for predicting shape, pose, and global orientation, while the feat3d w/ T is used to predict position.

For multi-view input, after obtaining feat3d w/o T and feat3d w/ T from each view, we employ a concatenation operation to fuse them into a unified multi-view feat3d w/o T and feat3d w/ T. These fused features are then passed through subsequent modules to generate the final predictions.

Neural Smooth Given that 2D points can be derived solely from image information, models generally exhibit better generalization in predicting 2D points. However, we have identified significant differences in camera parameters, particularly in translation, between the Umetrack and HOT3D datasets. These discrepancies introduce challenges in accurately predicting positions. To address the poor generalization to varying camera parameters and leverage temporal information for optimization, we propose a method called Neural Smooth.

Initially, we utilize a pre-trained model to obtain 2D landmarks, MANO shape, pose, global orientation, and position for each frame. These predictions form a time series, representing the hand movements across frames, referred to as the MANO[4] parameters. Using these predictions as the initial values, we compute the corresponding 3D joints (xyz) and the projected 2D points (proj uv) through the MANO model. The loss is then calculated between the projected 2D points and the model's predicted 2D results.

To ensure temporal consistency, we also incorporate an acceleration error loss to constrain the consistency between frames. This additional loss helps maintain smooth transitions and reduces jitter in the predicted hand movements.

By iteratively optimizing these predictions, Neural Smooth enhances the model's ability to generalize across different camera parameters, thereby improving the accuracy of position predictions.

The final loss function in Neural Smooth can be formulated as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{acce\_pose}} + \lambda_2 \mathcal{L}_{\text{acce\_orients}} + \lambda_3 \mathcal{L}_{\text{acce\_position}} + \lambda_4 \mathcal{L}_{\text{2D}}$$

where

$$\mathcal{L}_{\text{acce\_pose}} = \frac{1}{N-2} \sum_{t=3}^{N} |Pose_t - 2Pose_{t-1} + Pose_{t-2}|$$

The $\mathcal{L}_{\text{acce\_orients}}$ and $\mathcal{L}_{\text{acce\_position}}$ can also be written in a similar form. In our experiments, λ_1 to λ_4 are set to 0.5, 0.5, 0.5, and 1, respectively.

3. Experiments

In terms of model architecture, we selected Hiera-base[5] as the backbone. For training, we used binocular data with an image resolution of 224x224. The training was conducted on 8 NVIDIA 2080TI GPUs, with each GPU processing 16 images, resulting in a total batch size of 128. Each epoch consisted of 256 iterations, and the model was trained for a total of 300 epochs. We use AdamW[3] optimizer with an initial learning rate of 1e-4 scheduled by cosine scheduler and weight decay 1e-2.

After obtaining the initial results from the model, we apply the Neural Smooth method for optimization. We utilize the AdamW optimizer with cosine decay, setting the learning rate to 1e-2 and the maximum number of iterations to 500.

Table 1 shows several methods that significantly improved performance in our pose estimation for this challenge.

ID	method	Umetrack	HOT3d
1	Baseline	23.82	184.27
2	ID1 + extrinsic perturbations	17.91	36.93
3	ID2 + neural smooth	13.92	21.66

Table 1. MPJPE(mm) results on Umetrack/HOT3D test set.

For shape estimation, we did not perform any additional optimizations. We simply added a shape prediction module to the aforementioned model Figure 1 to make the corresponding predictions.

4. Conclusion

we presented the data, models, and post-processing optimization methods we employed in this challenge. Specifically, the neural smooth method significantly enhanced our performance in the competition.

References

[1] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. Introducing hot3d: An egocentric dataset for 3d hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024. 1

[2] S. Han, P. Wu, Y. Zhang, et al. Umetrack: Unified multi-view end-to-end hand tracking for vr. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 1

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[4] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017. 2

[5] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. *ICML*, 2023. 2