Supplementary Material for
AssemblyHands: Towards Egocentric Activity Understanding via 3D Hand Pose Estimation

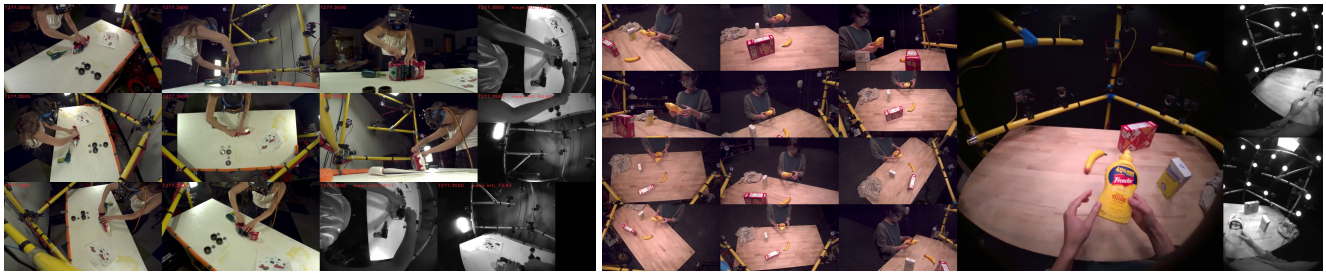

Figure 1. **Multi-view camera rigs used in our experiments.** Left: Assembly101. Right: Desktop Activities.

1. Camera rigs

Please see example captured frames from both camera rigs used in our experiments in Fig. 1.

AssemblyHands uses the same camera rig as Assembly101 [4], which consists of 8 RGB cameras of 1080p resolution mounted on a static scaffold, and a synchronized headset with 4 monochrome cameras of VGA resolution, arranged similarly to the Oculus Quest VR headset.

We also use another multi-camera setup from the *Desktop Activities* subset in the recent Aria Pilot Dataset [3], which has 12 RGB cameras of 1080p resolution, synchronized to the Project Aria glasses. The glasses are equipped with one egocentric RGB camera and two monochrome cameras, but we only use the static exocentric RGB cameras for the purpose of evaluating our multi-view automatic annotation method.

2. Implementation details

2.1. Processing of exocentric images

For all training and evaluation images, we pre-process them to remove the lens distortion from both exocentric images. This significantly simplifies geometric operations such as triangulation.

At training time, we first project the annotated 3D hand keypoints to each camera's image plane, then use the resulting 2D keypoints to define hand bounding boxes. Given a set of keypoint coordinates $\{(x_i, y_i), \forall i \in I\}$ on a single hand (or both hands), we create a square bounding box centered on the geometric center of all keypoints, with the side length L defined as

$$L = \gamma \cdot \max\left(\max_{i \in I} x_i - \min_{i \in I} x_i, \max_{i \in I} y_i - \min_{i \in I} y_i\right), \quad (1)$$

where γ is an expansion coefficient. It is randomized during training with a mean of 1.5. For the 3D feature volume, the size is 300 mm on each side, centered on the the third MCP joint. The root position is also augmented during training by adding a small random noise in the range of $[-5\text{ mm}, 5\text{ mm}]$ to all axes; we have observed that without this augmentation, the network can learn a trivial solution $(0, 0, 0)$ for the root joint.

At test time, we need to crop hands based on the output of a hand detector. However, we found it challenging to directly apply 2D object detectors for hands: off-the-shelf models do not achieve very good accuracy in our setup due to the challenging occlusions, and their detections from different views are not necessarily geometrically consistent. Instead, we use a more robust heuristic based on the triangulation of body keypoints. Specifically, we first detect 3D body keypoints from multi-view exocentric images, using the "2D + Triangulation" approach with a full-image body keypoint detector trained on MS COCO. Then, we create a virtual "hand center" keypoint in 3D, by extending 1/3 of the forearm length (defined from the elbow and wrist keypoints) out from the wrist. Finally, for every camera view, we can derive an image-space bounding box, centered on the 2D projection of the "hand center", with a heuristic size. We note that a similar heuristic is employed in the open-source implementation of OpenPose [1].

This initialization of hand bounding boxes is crucial for the quality of automatically generated annotations. We

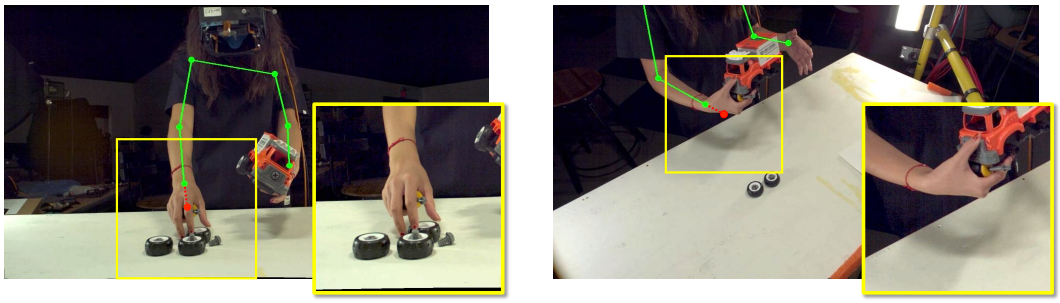

Figure 2. **Visualization of hand detection on two different frames.** Green: detected body keypoints (shown: shoulder, elbow, wrist). Red: estimated hand center. Yellow: hand bounding boxes derived from our heuristic, and the resulting cropped images. Left: the hand is well centered in the bounding box when the wrist is not bent. Right: hand is not centered when the wrist is bent; this can be corrected by our iterative refinement at inference time.

found that our iterative refinement during the annotation pipeline is a key step to reduce the annotation error. However, when wrong hand boxes are given in the initialization due to misidentifying left or right hand, the refinement further accumulates the error because a new hand crop is generated from the keypoint prediction on the misidentified hand crop. The introduced hand detection based on 3D body keypoints is more robust to such failures as the hand identity is determined from the skeleton structure of human body, not hand image itself.

Fig. 2 shows examples of detected body keypoints and derived hand bounding boxes. By construction, the bounding boxes in different views of the same frame are geometrically consistent, and can be directly used in creating the 3D feature volume. On the other hand, a drawback of this approach is that the heuristic guess of "hand center" can be inaccurate, especially when the wrist is significantly bent. As a result, the crop is not necessarily centered on the hand, and we need to use a larger size to ensure coverage. However, this can be corrected by our iterative refinement scheme during inference; see Fig. 4 in the paper.

2.2. Processing of egocentric images

Based on the annotated keypoints, we generate hand clips for input to the egocentric hand pose estimator. Given the 3D world-space coordinates of all joint locations (21 per hand) on two hands and an egocentric camera, we project the keypoints to the 2D image space, and then crop the image using bounding boxes that enclose the 2D keypoints. We remove lens distortion from the original fisheye cameras so that the images correspond to simple pinhole cameras. We also remove crops that are too close to the image boundary or do not contain hands in a given image. This preprocessing is separately done for each right or left hand, so we have two input crops when the two hands are shown in an image.

Then, given a single cropped image where either the left or right hand appears, we predict the 3D coordinates of 21 joints in the wrist-relative space. In post-processing, using the predictions for each hand, we convert a single-hand prediction to two-hand prediction by merging two predictions on different hand crops generated from the same single image. This two-hand format is used for the pose evaluation with the action recognition model.

3. Annotation quality

3.1. Comparison to annotation using OpenPose

The open-source OpenPose [1] has been used to automatically annotate hand poses in several existing datasets, *e.g.*, H2O. We compare to this annotation approach, by running a 2D + Triangulation baseline with the hand keypoints predicted by OpenPose.

Table 1 shows the comparison between the OpenPose baseline and our proposed methods. First, we note that 2D + Triangulation with OpenPose fails to triangulate on 40.2% of all the annotated keypoints in our test set, due to either too few 2D detections or large triangulation error. While it does achieve a reasonable 5.15 mm MPJPE on the successfully triangulated predictions, the high number of missing annotations is undesirable for the purpose of training an egocentric model. Note that we vary the distance threshold between 0 and 20 mm when computing PCK-AUC, and we observe OpenPose's PCK value at the maximum cutoff is 59.2%. The rest either do not receive a valid prediction, or have a prediction error larger than 20 mm.

Both our 2D + Triangulation baseline and final model MVExoNet-R3 significantly outperform OpenPose. First, our 2D + Triangulation model significantly increases the ratio of successful triangulations, at the cost of slightly higher MPJPE. Then, MVExoNet-R3 has both the lowest MPJPE and a much higher successful annotation rate, with a PCK value over 85% at the 20 mm threshold.

Annotation method	MPJPE	PCK-AUC
2D + Triangulation (OpenPose)	5.15*	48.1
2D + Triangulation (Ours)	7.97	63.8
MVExoNet-R3 (Ours)	4.20	83.4

* Evaluated on valid predictions only.

Table 1. **Comparison to OpenPose-based annotation.** While 2D + Triangulation with OpenPose is relatively accurate on the valid predictions, it fails to triangulate on 40.2% of the keypoints, leading to a very low PCK-AUC value. We use a maximum distance threshold of 20 mm for PCK-AUC. Our proposed 2D + Triangulation and MVExoNet-R3 both outperform OpenPose by a significant margin.

3.2. Verb-wise annotation error

We evaluate our annotation method for each verb category on manually annotated data. As shown in Fig. 3, we plot relative improvement of our final annotation method (*i.e.*, MVExoNet-R3) to the original annotation based on egocentric hand tracker [2] in Assembly101. For all the verbs appearing in the manually annotated set, our method further reduces the error by more than 10 mm compared to the original annotation. Our annotation particularly improves the quality when two hands are interacting with objects intricately, such as *position screw on*, *screw*, *push*, and *position*. This indicates that our multi-view annotation is more effective during such heavy hand-object occlusion than the annotation from egocentric views only.

Owing to this error reduction in pose annotations, in the downstream task of verb classification (see Table 5 in the paper), the performance of our SVEgoNet for *position* is improved by a large margin of 13%. Considering the quality of the annotation even for each verb and its support for verb classification, more refined pose annotation helps accurately recognize user's hand actions.

4. Action recognition

4.1. Verb label selection

In Assembly101 [4], fine-grained actions are defined as the combination of a verb and an interacting object, which consists of 24 classes. Since the verb labels follow a long-tailed distribution, we select the six most frequent verbs out of the full list of 24 for our study. These include three main assembly verbs: *pick up*, *position*, *screw*, and three disassembly verbs: *put down*, *remove* and *unscrew*, which altogether cover 70% of the verb labels introduced in Assembly101.

4.2. Object cues in verb recognition

As noted in the future work (Section 6), we believe using object information could further help in recognizing the user's actions. Since object annotation (*e.g.*, box and pose) for small parts is challenging for this dataset nowadays, we try to use object labels for verb recognition. We incorporate object class labels as one-hot encoded frame-level features into our pose-based verb classifier. As we reported in Table 5, the pose-only recognition using SVEgoNet has a verb recognition accuracy of 54.7%. In contrast, the classifier based on pose + object labels achieves a higher accuracy of 56.1%. This improvement further inspires us to explore the use of object bounding boxes and object poses.

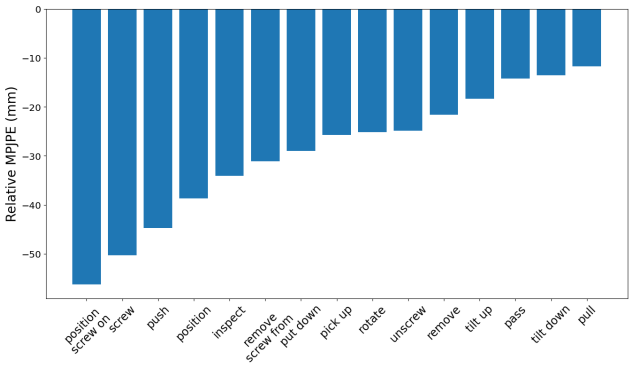

Figure 3. **Verb-wise hand pose annotation error reduction in AssemblyHands.** We show relative MPJPE (mm) of our annotations to the original annotations provided by Assembly101 [4]. We observe large reductions in annotation error across all verbs.

References

[1] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1, 2

[2] S. Han, P.-C. Wu, Y. Zhang, B. Liu, L. Zhang, Z. Wang, W. Si, P. Zhang, Y. Cai, T. Hodan, R. Cabezas, L. Tran, M. Akbay, T.-H. Yu, C. Keskin, and R. Wang. UmeTrack: Unified multi-view end-to-end hand tracking for VR. In *Proceedings of the ACM SIGGRAPH Asia Conference*, pages 50:1–50:9, 2022. 3

[3] Z. Lv, E. Miller, J. Meissner, L. Pesqueira, C. Sweeney, J. Dong, L. Ma, P. Patel, P. Moulon, K. Somasundaram, O. Parkhi, Y. Zou, N. Raina, S. Saarinen, Y. M. Mansour, P.-K. Huang, Z. Wang, A. Troynikov, R. M. Artal, D. DeTone, D. Barnes, E. Argall, A. Lobanovskiy, D. J. Kim, P. Bouttefroy, J. Straub, J. J. Engel, P. Gupta, M. Yan, R. D. Nardi, and R. Newcombe. Aria pilot dataset. https://about.facebook.com/realitylabs/projectaria/datasets, 2022. 1

[4] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21096–21106, 2022. 1, 3