UmeTrack: Unified multi-view end-to-end hand tracking for VR

SHANGCHEN HAN, PO-CHEN WU, YUBO ZHANG, and BEIBEI LIU, Meta Reality Labs, USA
LINGUANG ZHANG, ZHENG WANG, WEIGUANG SI, and PEIZHAO ZHANG, Meta Reality Labs, USA
YUJUN CAI, TOMAS HODAN, RANDI CABEZAS, and LUAN TRAN, Meta Reality Labs, USA
MUZAFFER AKBAY, TSZ-HO YU, CEM KESKIN, and ROBERT WANG, Meta Reality Labs, USA

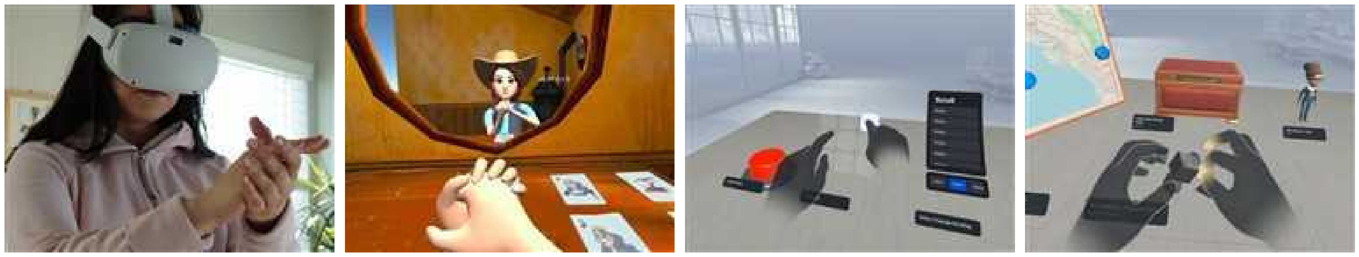
Fig. 1. The proposed hand tracking method can effectively incorporate information from multi-view image sequences captured by cameras mounted on a VR headset (left) and has been successfully deployed in various VR applications (right).

Real-time tracking of 3D hand pose in world space is a challenging problem and plays an important role in VR interaction. Existing work in this space are limited to either producing root-relative (versus world space) 3D pose or rely on multiple stages such as generating heatmaps and kinematic optimization to obtain 3D pose. Moreover, the typical VR scenario, which involves multi-view tracking from wide field of view (FOV) cameras is seldom addressed by these methods. In this paper, we present a unified end-to-end differentiable framework for multi-view, multi-frame hand tracking that directly predicts 3D hand pose in world space. We demonstrate the benefits of end-to-end differentiabilty by extending our framework with downstream tasks such as jitter reduction and pinch prediction. To demonstrate the efficacy of our model, we further present a new large-scale egocentric hand pose dataset that consists of both real and synthetic data. Experiments show that our system trained on this dataset handles various challenging interactive motions, and has been successfully applied to real-time VR applications.

CCS Concepts: • **Computing methodologies**;

Additional Key Words and Phrases: motion capture, hand tracking, virtual reality

ACM Reference Format:
Shangchen Han, Po-chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomas Hodan, Randi Cabezas, Luan Tran, Muzaffer Akbay, Tsz-Ho Yu, Cem Keskin, and Robert Wang. 2022. UmeTrack: Unified multi-view end-to-end hand tracking for VR. In *SIGGRAPH Asia 2022 Conference Papers (SA '22 Conference Papers), December 6–9, 2022, Daegu, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3550469.3555378

1 INTRODUCTION

Commercial headsets for virtual and augmented reality (VR/AR), including Meta Quest, Pico Neo, HTC Vive and the HP Reverb G2 deliver immersive experiences for gaming, communication, productivity and fitness. While the first generation of these headsets primarily relied on controllers for gaming, recent headsets have shifted towards interactions based on hand tracking to deliver a more natural experience and cater to applications outside of gaming.

Classical methods for hand tracking, e.g. [Han et al. 2020; Sharp et al. 2015; Taylor et al. 2016], typically employ multiple stages, first predicting heatmaps or correspondences and then solving for the 3D hand pose with a separate optimization. However, these multi-stage methods are supervised with proxy losses (e.g. heatmap loss) rather than the actual target metrics (e.g. 3D pose accuracy). A recent line of research [Cai et al. 2018; Spurr et al. 2021; Zimmermann and Brox 2017] explores single-stage architectures that can be trained end-to-end to directly predict hand pose. However, these approaches only make root-relative 3D pose predictions. Many VR interactions (examples in Figure 1) require absolute 3D root poses and these methods need to adopt an additional absolute root pose recovery stage to be applicable for these types of interactions.

Moreover, the hand tracking systems deployed on VR headsets operate on multi-view image sequences captured by wide FOV RGB or monochrome cameras. However, an architecture that unifies multi-view fusion, temporal fusion and handling of wide FOV cameras is still missing. [Han et al. 2020] proposed a relative distance parameterization to handle images from wide FOV cameras but their system is designed as a multi-stage pipeline. [Boukhayma et al. 2019; Zhou et al. 2016, 2020] designed their networks for single-view hand pose estimation. Incorporating multi-view or temporal information has been separately studied in previous work [Cai et al. 2019; Chen et al. 2021; Yang et al. 2020], but never fully unified.

In this paper, we propose an end-to-end differentiable architecture that can predict the absolute 3D hand pose and handle both single- and multi-view temporal data from wide FOV fisheye cameras. The end-to-end differentiability allows us to optimize not only pose accuracy, but also other key aspects of the user/developer experience including jitter and pinch detection, both of which are critical to VR interaction. With an end-to-end differentiable framework, we achieve superior jitter metric through a temporal loss and more accurate pinch detections through a pinch loss than the stage-of-the-art multi-stage hand tracking system.

As the proposed method is based on a deep neural network, a dataset for training and evaluation is essential. To this end, we introduce a large-scale egocentric hand tracking dataset. This dataset was collected using 4 fisheye monochrome cameras featuring both real and synthetic data with large variations. Both single-hand motions as well as challenging hand-hand interactions are included in the dataset. The dataset also contains dedicated pinch sequences with annotated pinch events to study the interplay between action recognition and pose estimation tasks. Our proposed method trained on this dataset shows robust performance on challenging hand motions and has been successfully deployed in several VR applications (Figure 1).

This work makes the following contributions:

(1) An end-to-end differentiable architecture that unifies multi-view fusion, temporal fusion and handling of wide FOV images while making absolute 3D hand pose predictions. This unification has only been achieved using multi-stage methods in previous work.
(2) By leveraging the end-to-end differentiability, we achieve superior jitter metric through a temporal loss and more accurate pinch detections through a pinch loss than the state-of-the-art multi-stage hand tracking system.
(3) A new large-scale egocentric dataset featuring single-hand motions and hand-hand interactions with 1397 real and 1397 synthetic sequences from 53 users. The dataset will be publicly released.

2 RELATED WORK

2.1 Pose estimation using neural networks

Many hand pose estimation methods start by predicting heat maps to estimate 2D keypoints, typically from a single view [Cai et al. 2020; Iqbal et al. 2018]. To better handle self-occlusion and depth ambiguity, multi-view data is integrated through triangulation [Simon et al. 2017] or post-inference optimization [Han et al. 2020; Simon et al. 2017]. Recent work on multi-view fusion within the network can be achieved using latent features [He et al. 2020; Iskakov et al. 2019; Remelli et al. 2020]. For instance, Remelli et al. [2020] used the Feature Transform Layer (FTL) to learn camera geometry-aware latent features and supervise 3D keypoint positions in an end-to-end differentiable manner. However, reconstructing the hand pose from keypoints usually requires an additional optimization stage.

Benefiting from the fast development of learning-based framework, many recent papers [Boukhayma et al. 2019; Spurr et al. 2018; Theodoridis et al. 2020] provided promising results for the root-relative 3D hand pose from single images with end-to-end differentiable architectures. However, estimating absolute 3D pose using a single network still remains limited. For instance, Boukhayma et al. [2019]; Kulon et al. [2020]; Yang et al. [2020] use a weak perspective projection camera model which inherently carries depth ambiguity. [Cai et al. 2019; Chen et al. 2021; Zhou et al. 2020; Zimmermann and Brox 2017] predict root-relative 3D hand poses and a global alignment to ground truth is performed before making evaluations. Moon et al. [2019] proposed a distance-aware architecture to predict absolute root locations but it is later shown to be insufficient to give accurate depth predictions [Moon et al. 2020]. Recent work [Cai et al. 2019; Chen et al. 2021; Hasson et al. 2020; Yang et al. 2020] also explored using mutli-view information or temporal information to address ambiguities in single images but they are still limited to root-relative 3D pose estimation. The architecture proposed in this paper is designed to incorporate multi-view, temporal information while predicting absolute 3D hand pose.

2.2 Hand pose datasets

Table 1 presents a summary of the different datasets used for hand pose estimation. Moon et al. [2020] introduced a large high quality dataset with challenging hand motions labeled with both manual annotations and bootstrapping methods using multiple high resolution RGB cameras. However, the dataset has minimal background and lighting variation. Christian Zimmermann and Brox [2019] also used a multi-view capture cage and included more background variation but the released dataset is restricted to training and evaluating single-frame, single-view pose estimation methods. Due to the difficulty of obtaining high quality data with large variations, synthetic datasets [Mueller et al. 2018; Zimmermann and Brox 2017] based on rendering or image-to-image translations have been proposed. All the above datasets are limited to outside-in views captured by narrow FOV cameras, which are not suitable for egocentric hand tracking for VR/AR.

The available egocentric datasets are either instrumented with highly visible markers [Garcia-Hernando et al. 2018] or have limited environmental variation [Kwon et al. 2021]. For collecting multi-view egocentric dataset, Han et al. [2020] introduced two approaches: (1) a mobile setup equipped with a single depth sensor where ground truth is obtained using a depth based hand tracker, and (2) a lab setup with many motion capture cameras where ground truth is obtained with a marker-based hand tracker [Han et al. 2018]. Annotations of both approaches are obtained automatically, making them suitable for large scale data collections. However, the ground truth quality of (1) is limited by the single-view depth based hand tracker (which struggles with the ambiguity of hand-hand occlusions) and (2) contains limited environment and lighting variations. In this work, we adopt the same marker-based hand tracker for uncompromised ground truth but use synthetic data to improve environment and lighting variations. Altogether, we contribute the largest and most diverse egocentric multi-view, multi-frame dataset to date (as shown in Table 1).

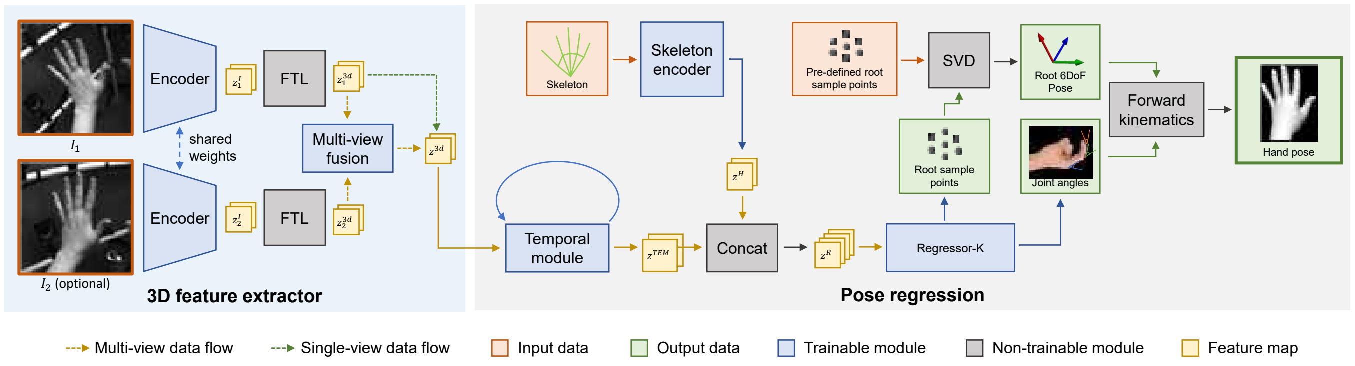

Fig. 2. This figure shows our architecture for the case of a known hand skeleton. The **3D feature extractor** block can take either single-view or multi-view input data and produce 3D features z^{3d} via a feature transform layer (FTL). The **pose regression** block first generates z^R which contains 3D information, temporal context and skeleton features. The known-skeleton regressor (Regressor-K) takes z^R as input and predicts the absolute 3D hand pose. The whole network can be trained end-to-end as all the modules and operators are differentiable.

Table 1. Comparing the proposed datasets with existing datasets.

Dataset	# frames	ego	bg variation	markerless	real	large fov	hand-hand
FreiHAND [2019]	37K	✗	✓	✓	✓	✗	✗
HOnnotate [2020]	78K	✗	✗	✓	✓	✗	✗
InterHand2.6M [2020]	2,590K	✗	✗	✓	✓	✗	✓
GANerated Hands [2018]	331K	✗	✓	✓	✗	✗	✗
H2O [2021]	571K	✓	✗	✓	✓	✗	✗
FPHA [2018]	105K	✓	✓	✗	✓	✗	✗
Ours (real)	839k	✓	✗	✗	✓	✓	✓
Ours (synth)	839k	✓	✓	✓	✗	✓	✓

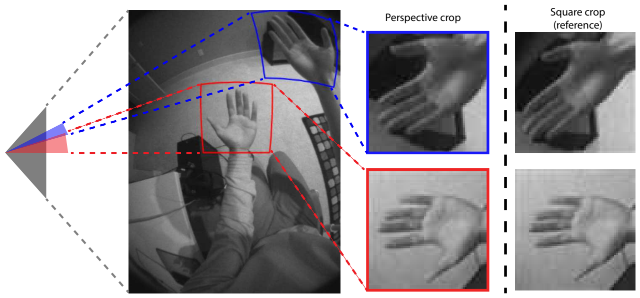

Fig. 3. The perspective cropping method generates a virtual perspective camera for each hand. In this figure, each camera is represented with a different color: grey for original camera, red for left hand virtual camera and blue for right hand virtual camera. The solid lines around each hand outline the crop boundaries. On the right, we show the square crops commonly used in previous methods for comparison. Note that the square crop of the right hand is visually more distorted whereas perspective cropping can correct this distortion.

3 METHOD

Preliminaries. Our hand skeleton H consists of 20 joints that can be articulated by a 20 dimensional joint angle vector θ. Each joint is defined by a joint position and rotation axis. The global transformation of the hand is represented by the root transformation T_H consisting of 6 degrees of freedom (DOFs). With hand skeleton and predicted joint angles and root transformation, we can animate the corresponding hand mesh via linear blend skinning (LBS).

We adopt the multi-camera layout using wide FOV fisheye cameras placed on a VR headset by Han et al. [2020]. In this layout, a hand can be seen by a single camera or multiple cameras depending on the hand location. As a result, we designed our network to jointly handle single-view and multi-vew input data. For each frame, the input to our model is either single-view or multi-view images where the region of interest (ROI) around a hand is provided.

For real-world usage, our model supports both known and unknown hand skeletons. If a user has a hand skeleton generated in advance (i.e. by a scanning system), our model is capable of utilizing the skeleton information and the output is $\{\theta, T_H\}$ in this case. The corresponding architecture is shown in Figure 2. For new users without prior knowledge of hand skeletons, we calibrate the unknown hand skeletons by predicting hand scale from multi-view images and the model output becomes $\{\theta, T_H, H\}$.

In the following sections, Section 3.1 describes how we prepare input images for our model. Section 3.2 gives an overview of our model when a known hand skeleton is provided. Section 3.3 details how skeleton calibration is performed when the user's hand skeleton is unknown. Section 3.4 introduces the loss terms we use for training our model.

3.1 Perspective cropping for input images

Most existing works [Han et al. 2020] utilize a square crop around a hand region as the input. Unlike images captured by a physical camera, these cropped images essentially lose the corresponding camera geometry information (i.e. intrinsics and extrinsics), making them infeasible for directly estimating absolute 3D pose. To tackle this issue, inspired by the effectiveness of perspective cropping [Remelli et al. 2020; Yu et al. 2021], we adopt the same cropping approach (Figure 3) for generating input images for our model. Specifically, given a ROI in the original image, we first create a virtual perspective camera at the same location as the original camera. The extrinsics matrix is constructed such that the virtual camera's z-axis points at the center of the ROI. At train time, the virtual camera's intrinsics matrix is set such that all the ground truth hand keypoints are projected within the image boundary. At inference time when ground truth is not available, we rely on the previously tracked pose to estimate the intrinsics matrix. Once the virtual camera is created, the warping technique described by [Yu et al. 2021] can be used to generate the cropped image. By using perspective cropping, we achieved 2 goals: (1) distortions from fisheye images can be corrected (as shown in Figure 3); (2) the virtual camera provides essential geometry information for 3D space, which enables direct hand scale prediction and absolute 3D pose prediction.

3.2 Architecture with known hand skeleton

In this section, we discuss the network architecture with known hand skeleton, as is shown in Figure 2.

3D feature extractor. Given the input cropped images $\{I_i\}_{i=1}^n$, where N represents the number of hand crops, we first feed them into a fully-convolutional encoder to generate the latent features z_i^I. Here z_i^I is extracted from the image space without knowing the camera geometry, which is insufficient for scale prediction or absolute 3D pose prediction. To incorporate 3D structures into the network, we leverage FTL [Remelli et al. 2020; Worrall et al. 2017] to generate camera-geometry-aware latent features:

$$z_i^{3d} = \mathrm{FTL}(z_i^I | T_{i,1} * K_i^{-1}). \tag{1}$$

Here, the image features z_i^I is first unprojected to 3D space using the virtual camera intrinsics matrix K_i, and then transformed from the 3D space of virtual camera i into the 3D space of the reference camera. In practice, we set virtual camera 1 as the reference camera and $T_{i,1}$ denotes the transform from camera i to camera 1 (so $T_{1,1}$ is identity in particular). By doing so, we integrate the 3D structural information into the image features and the transformed features $\{z_i^{3d}\}_{i=1}^n$ are all in the same 3D space.

Our model can handle both single-view ($n = 1$) and multi-view ($n > 1$) data. We denote the output of *3D feature extractor* as z^{3d}. For single-view input, the features can be directly set with: $z^{3d} = z_1^{3d}$. For multi-view input, a multi-view fusion module MVF is deployed to fuse features from different camera views:

$$z^{3d} = MVF(concatenate(z_i^{3d}, ..., z_n^{3d})), i \in [1, n]. \tag{2}$$

The output z^{3d} of MVF has the same shape as z_i^{3d} so that the following modules in our model that consume z^{3d} as input can be agnostic to the number of input images. Since z_i^{3d} carries the inter-camera relationships thanks to FTL, MVF is capable of performing learnable triangulations in the feature space [Iskakov et al. 2019] and is the key to our ability to predict hand scale (See Section 3.3 for more details).

Temporal Module. Given a set of (single-view or multi-view) spatial features z^{3d}, we start the pose regression step with a temporal module TEM, which fuses z^{3d} with the temporal context: $z^{TEM} = TEM(z^{3d}, h_{TEM})$. The temporal module is a recurrent neural network with hidden states denoted as h_{TEM}. It aims to learn temporal consistency, and serves as the key to handling hands under severe occlusions.

Skeleton Encoder. The task of pose regression is ambiguous without the skeleton data: (1) if the input is a single image, depth ambiguity cannot be resolved without knowing the hand scale, and (2) joint angles need to be coupled with joint positions and rotation axes to define hand articulations. When having the prior knowledge of the skeleton data, we introduce a skeleton encoder SE to implicitly incorporate hand skeleton information into the network. Specifically, we turn the joint positions (3-dimentional) and rotation axes (3-dimensional) at the rest hand pose into a feature vector. With 20 joints, we can obtain a 120 dimentional feature vector. The skeleton encoder SE takes a skeleton H as input, performs featurization and encodes the skeleton features into feature maps: $z^H = SE(H)$.

Regressor-K. The last learnable module is Regressor-K ("K" represents "**k**nown hand skeleton"). The input to Regressor-K is:

$$z^R = concatenate(z^{TEM}, z^H), \tag{3}$$

where the concatenated feature z^R contains 3D space information, temporal context and hand skeleton data. Regressor-K takes z^R as input and predicts 20 joint angles $\hat{\theta}$ and 3D root points $\hat{v}$ which encodes the root transform $\hat{T}_{H,1}$ in the reference camera space. To decode $\hat{T}_{H,1}$ from $\hat{v}$, we use Singular Value Decomposition [Sorkine-Hornung and Rabinovich 2016] to align pre-defined 3D root points in the local hand coordinate system to the predicted 3D root points $\hat{v}$. More details on decoding $\hat{T}_{H,1}$ can be found in the supplementary material. Then we recover the root transform $\hat{T}_H$ into the world coordinate system using the extrinsics of the reference camera. The output $\{\hat{\theta}, \hat{T}_H\}$ can be used to render the hand and drive applications in VR.

3.3 Calibration for unknown hand skeleton

We pose the hand skeleton calibration problem as a hand scale calibration task similar to [Han et al. 2020]. The calibrated hand scale can be used to scale a reference hand skeleton. Calibration of more parameters for a hand similar to [Romero et al. 2017] will be a future extension of the proposed method. For hand scale calibration, our method relies on Regressor-U ("U" represents "**u**nknown hand skeleton"). The differences between Regressor-K and Regressor-U are shown in Figure 4. As shown in Figure 4 (b), Regressor-U requires the temporal features z^{TEM} being computed from multi-view data since single-view data inherently carries scale ambiguity, making scale calibration an ill-posed problem. Regressor-U outputs an additional hand scale estimation which can be used to generate the calibrated hand skeleton $\hat{H}$. The final output of Regressor-U is $\{\hat{\theta}, \hat{T}_H, \hat{H}\}$. During training, both Regressor-K and Regressor-U can be trained jointly in an end-to-end manner. During inference, our model can dynamically decide which regressor to use depending on whether the skeleton data is provided.

3.4 Loss functions

The network is trained using a combination of three loss terms: pose loss, temporal loss and pinch loss.

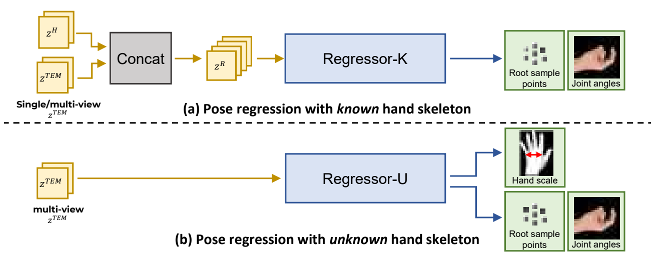

Fig. 4. we provide two regressors for (a) known and (b) unknown hand skeletons. (a) Regressor-K is used when a known hand skeleton is provided. Regressor-K is capable of handling z^{TEM} computed from either single-view or multi-view data and requires a known hand skeleton for computing z^H. (b) Regressor-U is used when the hand skeleton is unknown. Regressor-U requires z^{TEM} being computed from multi-view data and predicts an additional hand scale that can be used to generate the estimated skeleton.

Pose loss. The pose loss consists of a set of regression losses to supervise the hand poses:

$$L_{pose} = \sum_j^J ||p_j(\theta, T_H, H) - p_j(\hat{\theta}, \hat{T}_H, \hat{H})||_1 + \quad (4)$$

$$\lambda_\theta ||\theta - \hat{\theta}||_1 + \lambda_w ||w(T_H) - w(\hat{T}_H)||_1, \quad (5)$$

where $p_j(.)$ is a function that computes the j-th keypoint position using forward kinematics and w is a function that gets the translation component from T_H. We use the hat symbol to denote the network prediction. In the case a hand skeleton is known, $\hat{H}$ is the same as the groud truth skeleton H. If the hand skeleton is unknown, $\hat{H}$ is predicted by Regressor-U. We set the loss weights $\lambda_\theta = 0.05, \lambda_w = 0.5$ in our experiments.

Temporal loss. The temporal loss enforces smoothness of the predicted pose sequence by penalizing acceleration:

$$L_{temp} = \sum_t^T (||acc(\hat{\theta}, t))||_1 + ||acc(\hat{T}_H, t)||_1) \quad (6)$$

where $acc(x, t) = x_{t+1} + x_{t-1} - 2x_t$ is a function computing the acceleration of a given signal x at frame t. Similar losses are harder to incorporate for previous end-to-end methods that predict root-relative hand poses since root transformation is unavailable in those methods.

Pinch loss. Reliable detection of pinch is critical because pinch is used for selection in VR interaction. The design of our pinch loss is based on the observation that the distance between the thumb and index fingertips is small while pinching:

$$L_{pinch} = l \cdot \min(d(\hat{\theta}, \hat{T}_H, \hat{H}) - \epsilon_1, 0) + \quad (7)$$

$$(1 - l) \cdot \min(\epsilon_2 - d(\hat{\theta}, \hat{T}_H, \hat{H}), 0)), \quad (8)$$

where l is an annotated binary pinch label and d is a function that computes distance between the thumb and index fingertips. ϵ_1 is a distance threshold that corresponds to a high chance of a pinch event. ϵ_2 is a “safe" distance threshold that means a pinch event is unlikely to happen. ϵ_1 and ϵ_2 are chosen to be 0.01 and 0.02 meters respectively.

The final loss is a linear combination of the above losses:

$$L = L_{pose} + \lambda_t L_{temp} + \lambda_p L_{pinch}, \quad (9)$$

where we set weights $\lambda_t = 0.05, \lambda_p = 0.4$ in experiments.

4 DATASET

For training our pipeline in a supervised manner, a large scale multi-view egocentric dataset with challenging interactive hand motion is required. As seen in Table 1, existing datasets for hand tracking tasks are either infeasible for egocentric viewpoints or lack the challenging hand-hand motions, which are crutial for VR applications. To this end, we propose a new real-world large-scale egocentric dataset, with 1397 sequences for 53 users. For each user, we get the ground truth hand skeleton and mesh from a scanning system. Each sequence has 15 seconds and is captured at 30fps. Each frame contains 4 VGA images captured from 4 wide FOV monochrome cameras on a VR headset. We set up a motion capture system with 36 cameras to track marker locations. Markers (3mm) were placed on each user’s hands and a marker based hand tracker [Han et al. 2018] was used to obtain the ground truth motions. In each frame, a hand with ground truth label could appear in 1 or 2 views.

The dataset is divided into two protocols: a *separate-hand* protocol focusing on individual hand motions and a *hand-hand* protocol focusing on inter-hand interactions. Within the *separate-hand* protocol, 192 sequences contain pinch motion. Pinch events are manually annotated, resulting in 38003 pinch labels.

In addition to the real data, we rendered a synthetic dataset with the same hand motions as the real dataset using the Unity game engine. Each sequence is rendered using a different background and each frame is rendered using a different lighting configuration. As a result, the synthetic dataset provides much larger variations in environment and lighting. Samples of the real and synthetic data can be found in our supplementary video. As shown in Table 1, this dataset is the largest egocentric dataset to our knowledge dedicated to hand tracking in VR.

5 EVALUATION

To evaluate the accuracy of our system, we use mean-per-joint-position-error (MPJPE) which computes the average 3D Euclidean distance in millimeters between the estimated and ground truth keypoints in world space. We use the mean-per-joint-position-acceleration (MPJPA) metric to measure tracking jitter similar to [Han et al. 2020] using the following equation:

$$mpjpa(\hat{\theta}, \hat{T}_H, \hat{H}) = \frac{1}{T \cdot J} \sum_t^T \sum_j^J ||acc(p_j(\hat{\theta}, \hat{T}_H, \hat{H}), t)||_2 \quad (10)$$

Lower MPJPE indicates better accuracy and lower MPJPA indicates less jittery tracking. We compare our method to the state-of-the-art multi-stage hand tracking method for VR by [Han et al. 2020] to understand the benefit of end-to-end differentiability brought by our pipeline.

5.1 Implementation details

Data augmentation. We perform data augmentation by perturbing the camera intrinsics and extrinsics during training: (1) adding noise to the look-at direction when creating extrinsics of the virtual camera, (2) randomly applying in-plane rotation to the camera extrinsics, and (3) random scaling to the focal lengths. Note that we can't use the commonly used affine transforms to augment input images since they would cause a mismatch between virtual camera parameters and image data.

We implemented our method in PyTorch [Paszke et al. 2017]. For fair comparisons with Han et al. [2020], we use the same input resolution (96×96 monochrome image) and the same backbone for image encoder. All the modules in our model are jointly trained in an end-to-end manner using 9 GPUs with a batch size of 144. We first train the network for 200 epochs using the Adam optimizer with a learning rate of 0.0002 without temporal or pinch loss. The network is then trained for another 200 epochs with all the loss terms enabled. Inference with our model using 4 images (both hands are seen by 2 cameras) takes ~10ms on a PC with a NVIDIA GTX 2080 Super. We provide more details on the layers used for each module in the supplementary material.

Empirically, we found that a network trained on synthetic data shows better robustness to challenging environments whereas a network trained on real data gives better metrics on the test split of the real dataset. To leverage the benefits of both, we train both our method and the keypoint network by [Han et al. 2020] on the combined real and synthetic data for fair comparison. Metrics are reported on the test split of the real data.

5.2 Ablation study

In this section, we perform ablation study on the modules and loss functions. Pinch loss is not included and will be discussed later in section 5.4. We compare results under MPJPE and MPJPA metrics on the *separate-hand* protocol, which are summarized in Table 2. Without FTL, our model can barely learn to predict 3D hand pose and the corresponding model performs the worst (49.8mm MPJPE). The skeleton encoder extracts critical hand scale and joint features, without which, the network accuracy degrades (13.4mm MPJPE). Without the temporal module, the network loses the temporal context and performs slightly worse in MPJPE and much worse in MPJPA. Adding L_{temp} without the temporal module improves MPJPA (5.10 to 4.39) but leads to noticeable degradation in MPJPE (9.5 mm to 10.0mm). When using the full model, the model trained with L_{temp} shows slightly degraded MPJPE (9.4mm vs. 9.3mm) but performs much better in MPJPA metric (2.61 vs. 3.52) than the model trained without L_{temp}. To further validate the benefit of L_{temp}, we compare the model trained with L_{temp} to using a one-euro filter [Casiez et al. 2012] for post-processing. The one-euro filter parameters were tuned to produce identical MPJPA metric and it performed much worse in MPJPE metric (10.1mm vs. 9.4mm). Based on these observations, we consider L_{temp} brings enough benefit to MPJPA metric and we adopt the full model trained with both L_{pose} and L_{temp} for hand tracking task.

Figure 5 visualizes some sample results using our method. In particular, hands that are severely occluded by each other can be reasonably tracked (row 3, 5 in Figure 5). Another observation we made was the temporal module is the key to handling challenging occlusions by another object. To show this, we provide an example in Figure 6 where the full model is able to maintain a plausible pose when the hand is occluded by a paper whereas the model without the temporal module completely fails.

Table 2. Ablation study for different modules and loss functions. For each model, "|" separates the model architecture (left) and loss functions used for training (right). Model annotated with "(One-Euro)" refers to using one-euro filter to post-process the tracked poses. Best model is highlighted in bold.

model	MPJPE	MPJPA
w/o FTL | L_{pose}	49.8	6.29
w/o skeleton encoder | L_{pose}	13.4	4.07
w/o temporal module | L_{pose}	9.5	5.10
w/o temporal module | $L_{pose} + L_{temp}$	10.2	4.39
full model | L_{pose}	**9.3**	3.52
full model | L_{pose} (One-Euro)	10.1	2.67
full model | $\mathrm{L_{pose}} + \mathrm{L_{temp}}$	9.4	**2.61**

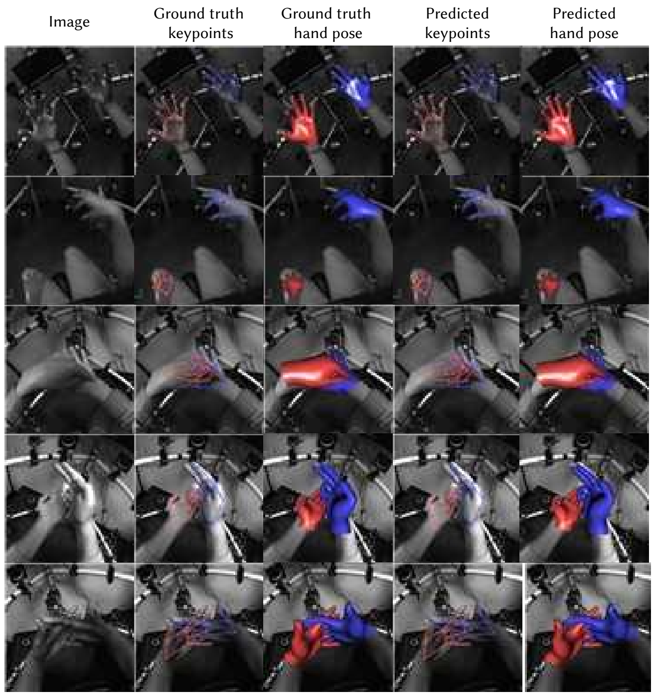

Fig. 5. Qualitative results on our evaluation dataset.

5.3 Comparison to previous multi-stage method

Known hand skeleton. We compare our model trained with both L_{pose} and L_{temp} to Han et al. [2020] using the ground truth skeletons provided by the dataset. Metrics on *separate-hand* and *hand-hand* protocols are reported in Table 3. On both protocols, our method outperforms [Han et al. 2020] in MPJPE (9.4mm vs. 9.9mm on *separate-hand*; 10.5mm vs 10.8mm on *hand-hand*). More details on the performance of each method at different error thresholds can be found in Figure 7(a) and (b). In particular, in Figure 7(a), the curve for our method crossed with the curve for [Han et al. 2020] at 9.5mm error threshold. This suggests our method produces

Table 3. Comparison with the multi-stage method [Han et al. 2020] on *separate-hand* and *hand-hand* protocols. For each of our models, loss functions used for training are specified after "|".

Method	Known hand skeleton				Unknown hand skeleton			
	separate-hand		*hand-hand*		*separate-hand*		*hand-hand*	
	MPJPE	MPJPA	MPJPE	MPJPA	MPJPE	MPJPA	MPJPE	MPJPA
[Han et al. 2020]	9.9	3.48	10.8	3.33	12.9	3.46	13.6	3.33
Ours | $L_{pose} + L_{temp}$	**9.4**	**2.61**	**10.5**	2.73	**11.2**	**2.57**	**12.0**	2.69
Ours | $L_{pose} + L_{temp} + L_{pinch}$	**9.4**	2.65	10.6	**2.68**	11.4	2.60	12.2	**2.65**

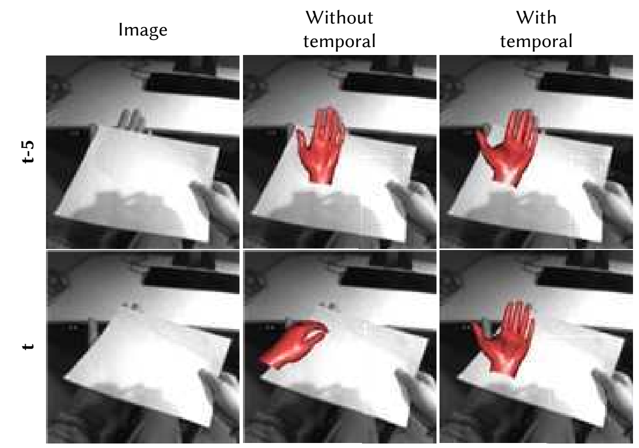

Fig. 6. The model without temporal module predicts a reasonable pose at frame (t-5) but completely fails at frame t due to severe occlusion. In contrast, the model with temporal module predicts a plausible hand pose at frame t by leveraging the temporal context.

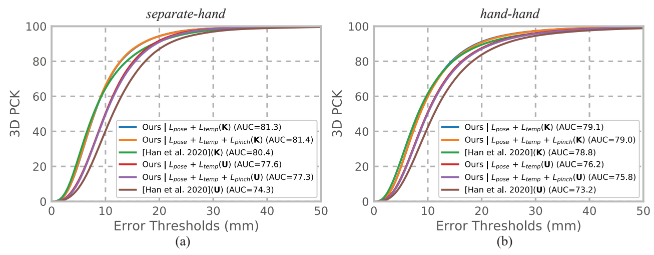

Fig. 7. PCK curves on *separate-hand* and *hand-hand* protocols. Legends including "(**K**)" refer to evaluation for **k**nown hand skeletons and legends with "(**U**)" refer to evaluation for **u**nknown hand skeletons.

fewer outliers (error > 9.5mm) at the cost of degraded capability in precise localization (error < 9.5mm). A similar phenomenon can also be seen in Figure 7(b). With respect to the MPJPA metric, our method consistently outperforms Han et al. [2020] (2.61 vs. 3.48 on *separate-hand*; 2.68 vs. 3.33 on *hand-hand*).

Unknown hand skeleton. When the user's hand skeleton is not provided, we perform a hand scale calibration for both our method and [Han et al. 2020]. Hand scale calibration in [Han et al. 2020] is achieved by gathering multiple multi-view frames and using an optimization method to solve for the hand scale. In our work, we calibrate the hand scale by averaging scale predictions of the first 30 frames. To gather metrics, we first run both methods to perform scale calibration to obtain calibrated hand skeletons. We then re-run inference using the calibrated hand skeletons. In Table 3, our method consistently outperforms [Han et al. 2020] in MPJPE metric by a large margin (11.2mm vs. 12.9mm on *separate-hand*; 12.0mm vs. 13.6mm on *hand-hand*), indicating the effectiveness of our end-to-end differentiable architecture.

Table 4. Comparison on pinch metrics. For each of our models, loss functions used for training are specified after "|".

model	precision (%)	recall (%)
[Han et al. 2020]	96.5	95.0
Ours | $L_{pose} + L_{temp}$	92.2	90.8
Ours | $L_{pose} + L_{temp} + L_{pinch}$	**97.3**	**97.3**

5.4 Pinch evaluation

In this section, we compare the precision and recall metrics [LeCun et al. 2015] for pinch detection. A pinch detector based on index-thumb fingertip distance thresholding is used for both methods. The threshold is the same as used in our designed L_{pinch}. As shown in Table 4, our model trained without L_{pinch} gives worse metrics than Han et al. [2020]. This could be related to the observation made in Section 5.3 that our method is less capable of precise localization. After incorporating L_{pinch} for training, though the MPJPE is slightly degraded as shown in Table 3, the pinch detection accuracy is significantly boosted. In Table 4, our model trained with L_{pinch} outperforms Han et al. [2020] (97.3% vs. 96.5% in precision; 97.3% vs. 95.0% in recall). We'd like to emphasize our model provides a framework for studying the interplay between pose estimation and many other downstream tasks such as pinch detection. These tasks are much harder to be jointly optimized in multi-stage methods.

6 CONCLUSION

We have presented an end-to-end differentiable architecture designed for hand tracking in VR. It unifies multi-view, temporal fusion and handling of wide FOV images while making absolute 3D hand pose predictions. We demonstrate end-to-end differentiability makes it easier to optimize for downstream tasks like jitter reduction and pinch detection compared to multi-stage non-differentiable pipelines. A new large-scale egocentric dataset is introduced. We demonstrate compelling VR experiences in the supplementary video with our model trained on this new datset.

Limitations and future work. Hand-hand interactions remain a challenge for our method. A potential solution is joint regression of both hand poses and designing hand-hand interaction losses (i.e. loss that prevents inter-penetration). Compared to [Han et al. 2020], we discovered our method is less capable of precise localizations. We hypotheize this to be a limitation with the direct pose regression approach. By borrowing ideas from the multi-stage method, our model can potentially incorporate heatmap regression and numerical optimization as differentiable components during training to achieve more precise hand tracking.

REFERENCES

Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 2019. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10843–10852.

Yujun Cai, Liuhao Ge, Jianfei Cai, Nadia Magnenat Thalmann, and Junsong Yuan. 2020. 3D hand pose estimation using synthetic data and weakly labeled RGB images. *IEEE transactions on pattern analysis and machine intelligence* 43, 11 (2020), 3739–3753.

Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. 2018. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 666–682.

Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. 2019. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 2272–2281.

Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2012).

Liangjian Chen, Shih-Yao Lin, Yusheng Xie, Yen-Yu Lin, and Xiaohui Xie. 2021. MVHM: A Large-Scale Multi-View Hand Mesh Benchmark for Accurate 3D Hand Pose Estimation. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2021), 836–845.

Jimei Yang Bryan Russel Max Argus Christian Zimmermann, Duygu Ceylan and Thomas Brox. 2019. FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape from Single RGB Images. In *IEEE International Conference on Computer Vision (ICCV)*. "https://lmb.informatik.uni-freiburg.de/projects/freihand/"

Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. 2018. First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. 2020. HOnnotate: A Method for 3D Annotation of Hand and Object Poses. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shangchen Han, Beibei Liu, Randi Cabezas, Christopher Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics* 39 (07 2020). https://doi.org/10.1145/3386569.3392452

Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D. Twigg, and Kenrick Kin. 2018. Online Optical Marker-based Hand Tracking with Deep Labels. *ACM Trans. Graph.* 37, 4, Article 166 (July 2018), 10 pages. https://doi.org/10.1145/3197517.3201399

Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. 2020. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 571–580.

Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. 2020. Epipolar Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7779–7788.

Umar Iqbal, Pavlo Molchanov, Thomas Breuel, Juergen Gall, and Jan Kautz. 2018. Hand Pose Estimation via Latent 2.5 D Heatmap Regression. In *Proceedings of European Conference on Computer Vision*.

Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. 2019. Learnable Triangulation of Human Pose. In *International Conference on Computer Vision (ICCV)*.

Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M. Bronstein, and Stefanos Zafeiriou. 2020. Weakly-Supervised Mesh-Convolutional Hand Reconstruction in the Wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Taein Kwon, Bugra Tekin, Jan Stuhmer, Federica Bogo, and Marc Pollefeys. 2021. H2O: Two Hands Manipulating Objects for First Person Interaction Recognition. In *ICCV 2021*. https://www.microsoft.com/en-us/research/publication/h2o-two-hands-manipulating-objects-for-first-person-interaction-recognition/

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. 2019. Camera Distance-aware Top-down Approach for 3D Multi-person Pose Estimation from a Single RGB Image. In *The IEEE Conference on International Conference on Computer Vision (ICCV)*.

Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. 2020. InterHand2.6M: A Dataset and Baseline for 3D Interacting Hand Pose Estimation from a Single RGB Image. In *European Conference on Computer Vision (ECCV)*.

Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 11 pages. https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

Edoardo Remelli, Shangchen Han, Sina Honari, Pascal Fua, and Robert Wang. 2020. Lightweight Multi-View 3D Pose Estimation Through Camera-Disentangled Representation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (Nov. 2017).

Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. 2015. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 3633–3642.

Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *CVPR*.

Olga Sorkine-Hornung and Michael Rabinovich. 2016. Least-Squares Rigid Motion Using SVD. Technical note.

Adrian Spurr, Aneesh Dahiya, Xi Wang, Xucong Zhang, and Otmar Hilliges. 2021. Self-Supervised 3D Hand Pose Estimation from monocular RGB via Contrastive Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11230–11239.

Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. 2018. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 89–98.

Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. 2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

Thomas Theodoridis, Theocharis Chatzis, Vassilios Solachidis, Kosmas Dimitropoulos, and Petros Daras. 2020. Cross-modal variational alignment of latent spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 960–961.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. 2017. Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 5726–5735.

John Yang, Hyung Jin Chang, Seungeui Lee, and Nojun Kwak. 2020. SeqHAND: RGB-Sequence-Based 3D Hand Pose and Shape Estimation. *CoRR* abs/2007.05168 (2020). arXiv:2007.05168 https://arxiv.org/abs/2007.05168

Frank Yu, Mathieu Salzmann, P. Fua, and Helge Rhodin. 2021. PCLs: Geometry-aware Neural Reconstruction of 3D Pose with Perspective Crop Layers. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 9060–9069.

Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. 2016. Deep Kinematic Pose Regression. *CoRR* abs/1609.05317 (2016). arXiv:1609.05317 http://arxiv.org/abs/1609.05317

Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. 2020. Monocular Real-Time Hand Shape and Motion Capture Using Multi-Modal Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Christian Zimmermann and Thomas Brox. 2017. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE International Conference on Computer Vision*. 4903–4911.

A NETWORK ARCHITECTURE DETAILS

The input shape, output shape, hidden state shape and the layers used for each module are shown in Table 5. The encoder uses the same resnet as [Han et al. 2020] to ensure fair comparisons. The last layer of the encoder is a 1×1 convolution layer for dimensionality reduction purpose. Multi-view fusion uses multiple 1×1

Table 5. Architecture table

Module	Input	Output	Hidden state	Layers
Encoder	$1 \times 96 \times 96$	$72 \times 6 \times 6$	NA	resnet + Conv11
Multi-view fusion	$144 \times 6 \times 6$	$72 \times 6 \times 6$	NA	(Conv11 + ReLU) ×2 + Conv11
Temporal module	$72 \times 6 \times 6$	$72 \times 6 \times 6$	$18 \times 6 \times 6$	(Conv11 + ReLU) ×2 + Conv11
Skeleton encoder	120	$4 \times 6 \times 6$	NA	linear + reshape
Regressor-K	$76 \times 6 \times 6$	41	NA	residual blocks × 2 + Pool
Regressor-U	$72 \times 6 \times 6$	42	NA	residual blocks × 2 + Pool

convolutions and ReLU layers. Each 1×1 convolution serves the purpose of feature fusion and dimensionality reduction. The output shape of the multi-view fusion module is the same as the output shape of the encoder. The temporal module is a recurrent neural network with a hidden state using 1×1 convolution and ReLU as the building blocks. Both Regressor-K and Regressor-U are built from residual blocks. The output of Regressor-K contains 20 dimensional joint angles and 21 dimensional root point coordinates. Regressor-U outputs a 1 dimensional hand scale parameter in addition to joint angle and root point outputs.

For root transform prediction, we pre-define 7 points for representing a transformation in the hand local space: $v_H = \{[0,0,0]^T, [1,0,0]^T, [0,1,0]^T, [0,0,1]^T, [1,1,0]^T, [1,0,1]^T, [0,1,1]^T\}$. And the task of a regressor is to predict the location of these points denoted as $\hat{v}$ in the reference camera space. The root transformation can be recovered using Singular Value Decomposition [Sorkine-Hornung and Rabinovich 2016] by solving the following equation:

$$\hat{T}_H = \min_{\hat{T}_H} \sum_i ||\hat{T}_H * v_{H,i} - \hat{v}_i||_2^2 \tag{11}$$