MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality

SHANGCHEN HAN, BEIBEI LIU, RANDI CABEZAS, CHRISTOPHER D. 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, Facebook Reality Labs

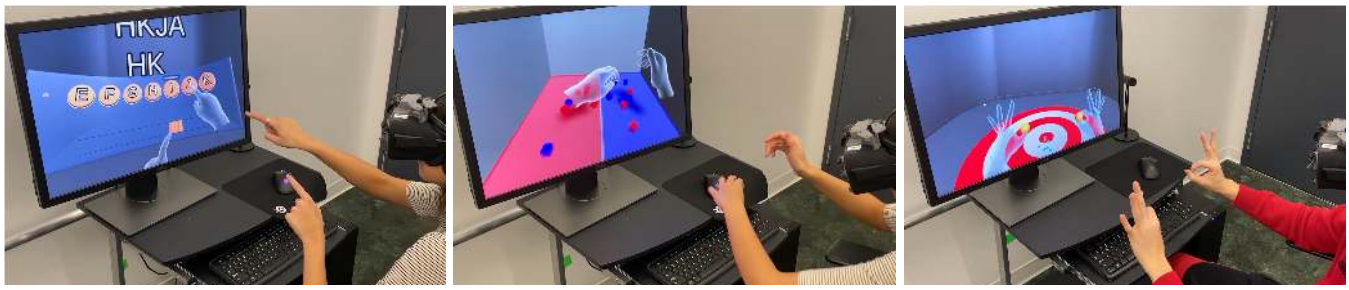

Fig. 1. We present a real-time hand-tracking system using four monochrome cameras mounted on a VR headset. We output the user's skeletal poses and rigged hand model meshes. Here we show some snapshots of users using our system to drive interactive VR experiences.

We present a system for real-time hand-tracking to drive virtual and augmented reality (VR/AR) experiences. Using four fisheye monochrome cameras, our system generates accurate and low-jitter 3D hand motion across a large working volume for a diverse set of users. We achieve this by proposing neural network architectures for detecting hands and estimating hand keypoint locations. Our hand detection network robustly handles a variety of real world environments. The keypoint estimation network leverages tracking history to produce spatially and temporally consistent poses. We design scalable, semi-automated mechanisms to collect a large and diverse set of ground truth data using a combination of manual annotation and automated tracking. Additionally, we introduce a detection-by-tracking method that increases smoothness while reducing the computational cost; the optimized system runs at 60Hz on PC and 30Hz on a mobile processor. Together, these contributions yield a practical system for capturing a user's hands and is the default feature on the Oculus Quest VR headset powering input and social presence.

CCS Concepts: • **Computing methodologies** → **Computer vision**.

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

ACM Reference Format:
Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. 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 Trans. Graph.* 39, 4, Article 87 (July 2020), 13 pages. https://doi.org/10.1145/3386569.3392452

Authors' address: Shangchen Han, shchhan@oculus.com; Beibei Liu, beibeiliu@fb.com; Randi Cabezas, rcabezas@fb.com; Christopher D. Twigg, cdtwigg@fb.com; Peizhao Zhang, stzpz@fb.com; Jeff Petkau, jpet@fb.com; Tsz-Ho Yu, thyu@fb.com; Chun-Jung Tai, btai@fb.com; Muzaffer Akbay, muzoakbay@fb.com; Zheng Wang, wangz@fb.com; Asaf Nitzan, asafn@fb.com; Gang Dong, gang.dong@fb.com; Yuting Ye, yuting.ye@fb.com; Lingling Tao, linglingt@fb.com; Chengde Wan, vgrasp@fb.com; Robert Wang, rywang@fb.com, Facebook Reality Labs.

0730-0301/2020/7-ART87
https://doi.org/10.1145/3386569.3392452

1 INTRODUCTION

Recent improvements in VR/AR technology have led to the mainstream adoption of commercial headsets such as the Oculus Quest, Microsoft HoloLens and HTC Vive for applications including gaming, virtual training and socializing in virtual worlds. As a new computing platform, VR/AR is still experimenting with various input modalities, including mouse and keyboard, game controllers, 6 degree of freedom (DOF) motion controllers and wearable gloves. Vision-based hand-tracking can potentially provide more convenient and lower-friction input than these peripherals. For instance, users may not need to carry or charge an additional device or put on a wearable. However, to be a truly convenient input modality, hand-tracking must also be robust to environmental and user variations, support a generous working volume and produce responsive and precise (low-jitter) motions for targeting and selection. As VR/AR headsets are increasingly mobile, a hand-tracking input system must also run on a low-compute budget.

Most previous work on hand-tracking has focused on outside-in depth or RGB cameras. A depth camera provides hand geometry in terms of a 2.5D point cloud. However, depth cameras impose extra requirements on hardware design and power usage. In comparison, RGB cameras are easier to integrate and their utility continues to improve as deep learning techniques advance. As a result, predicting hand pose from a single RGB camera, typically with the help of a neural network, has become a popular research topic.

Despite continued progress, several remaining issues have held back RGB-based hand-tracking from being applied in VR/AR. First, predicting 3D hand pose from a single RGB camera is inherently

ill-posed due to scale ambiguities, making it not directly suitable for the 3D pose estimation needed to drive VR input. While scale ambiguity can be resolved with stereo, existing methods cannot be easily adapted to achieve consistent predictions in multi-view settings. Second, most RGB-based methods either focus on keypoint regression [Cai et al. 2018; Iqbal et al. 2018; Spurr et al. 2018; Yang and Yao 2019; Zhang et al. 2016; Zimmermann and Brox 2017a] or simultaneous pose and shape reconstruction [Boukhayma et al. 2019; Ge et al. 2019; Hasson et al. 2019; Zhang et al. 2019b], neither of which are well-suited to maintain self-presence requirements inside VR/AR applications. The former cannot be directly used to render an actual hand mesh or re-target hand motion. The latter does not guarantee consistency of the hand shape over time. Third, prior work on RGB-based tracking does not use or evaluate against temporal information, leading to jittery predictions unsuitable for targeting tasks. To our knowledge, no existing work quantifies this issue or provides insight on how to resolve it.

Finally, perhaps the most critical issue for a learning-based hand-tracking system is acquiring sufficient high-quality ground truth data. Hand keypoints are impractical to manually annotate in images due to frequent self-occlusions. Several existing methods use a multi-view capture system to alleviate these challenges, but these systems are not easily portable, which limits the ability to capture a diverse set of background and lighting environments. As a workaround, [Zimmermann et al. 2019] uses a green screen for background replacement as a post-processing, but cannot support variation in lighting. [Zhang et al. 2016] uses a depth-based hand-tracking method to generate ground truth for RGB views. While this is a more scalable solution, ensuring that the depth tracker is sufficiently accurate during heavy occlusion is itself a difficult problem. Another workaround to generating real-world training data is to synthesize images [Mueller et al. 2018]. However, it's still unclear how to generate synthetic data that is realistic enough to train networks that generalize across users and environments.

The main contributions of this work are as follows:

- We present a tracking system that uses four egocentric monochrome fisheye cameras to produce 3D hand pose estimates. To the best of our knowledge, it is the first hand-tracking system that tracks robustly across different environments and users, supports a large working volume and allows for real-time performance not only on PC, but also on a mobile processor.
- We introduce a new keypoint estimation architecture that leverages tracking history, which improves both temporal smoothness across frames and spatial consistency across multiple views.
- Furthermore, we describe how to efficiently generate diverse, high quality labels using portable and lightweight data collection methods with minimal manual annotations.

2 RELATED WORK

RGB-based approaches. Early RGB-based methods [de La Gorce et al. 2008; Prisacariu and Reid 2011; Stenger et al. 2006] follow an analysis-by-synthesis paradigm, fitting a hand model to low level visual cues such as edges, skin color, silhouettes and optical flow. This is extremely challenging given the self-similarity, subtle color variation, and severe self-occlusion exhibited by hands. Later work resolve such challenges by augmenting the tracking system with either a color glove [Wang and Popović 2009] or multi-camera rigs [Ballan et al. 2012]. An alternative line of work uses nearest neighbors to find a matching image/pose pair in the dataset [Athitsos and Sclaroff 2003].

The current state-of-the-art is dominated by deep learning based approaches, which directly regress coordinates of hand skeleton keypoints [Cai et al. 2018; Iqbal et al. 2018; Spurr et al. 2018; Tekin et al. 2019; Yang and Yao 2019; Zhang et al. 2016; Zimmermann and Brox 2017a] and can be highly accurate when fed sufficient training data. Recent work [Cai et al. 2018; Iqbal et al. 2018; Spurr et al. 2018; Tekin et al. 2019; Yang and Yao 2019] has focused on frame-wise estimation. Because motion cues are largely ignored, the result can be jittery, especially when fingers are occluded. One notable exception [Cai et al. 2019] leverages temporal information with graph convolution but ignores hand shape consistency. Most recently, [Boukhayma et al. 2019; Ge et al. 2019; Hasson et al. 2019; Zhang et al. 2019b] extend deep neural networks to directly recover both hand shape and pose. However, no personalization process is involved and there is no guarantee that the hand shape of the same user will not change across frames.

Our approach differs from prior work due our focus on egocentric AR/VR input. Instead of RGB cameras, we use monochrome cameras, which exhibit superior signal-to-noise ratio in low light compared to their equivalent RGB counterparts. We use fisheye lenses, which expand the interaction volume, but also require a re-parameterization of the hand keypoint estimation problem to predict distance instead of depth. Input applications are also more sensitive to jitter and tracking from the egocentric viewpoint of an AR/VR headset is more prone to self-occlusions. To address this, we propose a keypoint regression network that incorporates tracking history, which provides temporally smooth results and helps resolve self-occlusion.

Depth-based approaches. Depth sensors have been widely applied to hand-tracking [Mueller et al. 2019; Oikonomidis et al. 2012; Sharp et al. 2015; Tagliasacchi et al. 2015; Taylor et al. 2016, 2017; Zhang et al. 2019a]. Model-based approaches can reliably fit a hand mesh to the reconstructed point cloud provided by the depth sensor, but this approach does not generalize to RGB images.

With a few exceptions [Zhang et al. 2016], most depth-based approaches use active illumination [Mueller et al. 2019; Sharp et al. 2015; Tagliasacchi et al. 2015; Taylor et al. 2016, 2017; Zhang et al. 2019a], which provides more accuracy in textureless environments. However, active illumination limits the field of view (FOV) and in turn, the interaction volume, due to the difficulty of building wide FOV projectors. Moreover, depth cameras are larger and use more power than monochrome cameras, which makes integration more challenging on an VR/AR headset.

While we use monochrome cameras at inference time, we use depth-based hand-tracking [Taylor et al. 2017] to generate accurate ground truth for training our neural networks, leveraging the benefits of both systems.

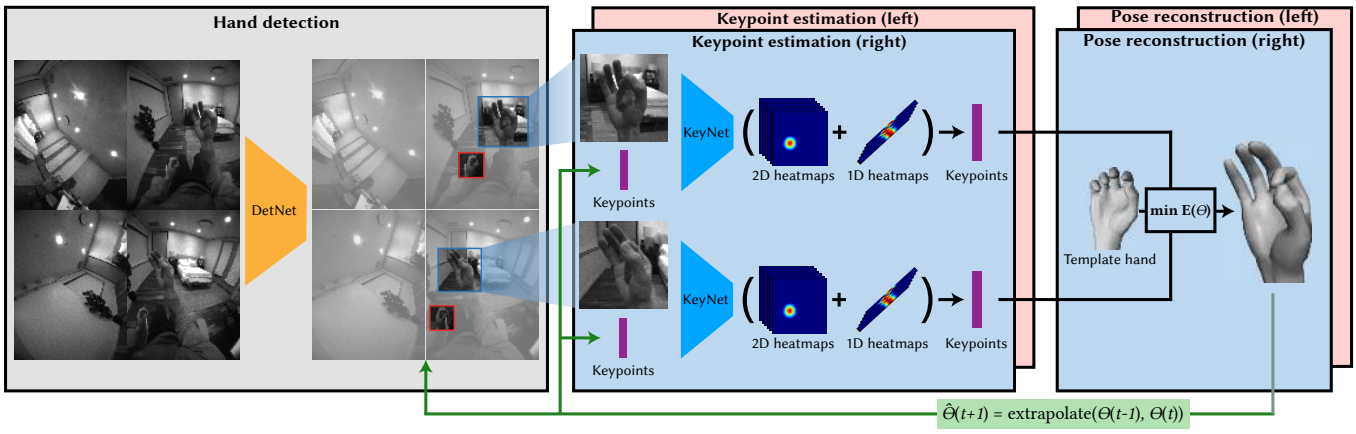

Fig. 2. **Hand-tracking pipeline.** Our system takes inputs from 4 monochrome VGA (640x480) cameras. We first compute a per-hand, per-image bounding box using either the hand detection network (DetNet) or by extrapolating the hand pose from the previous frames (Section 3.3). We crop each image using the bounding box and feed the resulting hand image into the hand keypoint network (KeyNet). The input is further augmented using keypoint features generated from the extrapolated hand pose (Section 3.4). The network predicts a 2D location heatmap and 1D relative distance heatmap for each of the predefined 21 hand keypoints. The keypoints are used to fit a template hand model for each hand (Section 3.5). Finally the output of the previous two frames are used to predict the pose in the next frame, which is used to improve both the bounding boxes and keypoint estimation.

Training data generation. Manually annotating 3D hand pose can be tedious and unreliable. During self-occlusion, manual labeling can suffer from temporal jitter and a lack of consensus between annotators [Supancic et al. 2015]. To minimize human effort, [Oberweger et al. 2016; Tompson et al. 2014] propose to couple human annotation with model based tracking, either by annotating a subset of key frames [Oberweger et al. 2016] or by manually initializing the tracker [Tompson et al. 2014]. [Simon et al. 2017] use a multi-camera system with bootstrapping where an annotator needs only to provide a binary indication of whether the estimated pose is accurate.

Another line of work leverages synthesized data. To bridge the distribution gap between real and synthesized data, [Mueller et al. 2018; Shrivastava et al. 2017] improve the realism of the synthesized data with a generative adversarial network (GAN). Alternatively, [Dibra et al. 2017; Wan et al. 2019] initialize the pose estimation network with synthesized data. The network is then fine-tuned on unlabeled real data by minimizing the model fitting error.

Utilizing cameras from other modalities, *e.g.* , IMU [Huang et al. 2018] or magnetic 6D sensors [Yuan et al. 2017] are also promising paths to generate accurate hand pose annotation. However, wearing the sensors can change the appearance of the hand, especially for RGB or monochrome inputs, making these approaches unsuitable for training deep learning based systems.

In this work, we use a depth tracker to generate ground truth hand poses for training the keypoint estimation network. This model-based tracker requires minimal human intervention (2D bounding boxes) if the tracker fails. Our system maximizes the quality of the ground truth data without sacrificing mobility. As a result, our training set is larger and more diverse in terms of hand shape, pose and background variation than any previously proposed RGB datasets [Zhang et al. 2016; Zimmermann and Brox 2017a].

3 HAND-TRACKING USING MONOCHROME CAMERAS

3.1 Overview

We outline our hand-tracking system in Figure 2. We start with the images from four monochrome cameras (Section 3.2) and detect the left and right hands in each image, producing a set of bounding boxes (Section 3.3). We crop each bounding box from the image and pass it to a network that detects 21 keypoints on the hand (Section 3.4). The resulting keypoints are then used to fit a 3D hand pose (Section 3.5).

Our hand model is defined in two parts, a kinematic hand skeleton $\boldsymbol{S}$ and a mesh model $\boldsymbol{M}$. The hand skeleton $\boldsymbol{S}$ consists of 26 degrees of freedom (DOFs) – 6 DOFs representing the global transformation $\{\mathbf{R}, \mathbf{t}\}$ and 4 rotational DOFs per finger representing finger articulations. Additionally, we define 21 keypoints on the hand – one for each finger's MCP, PIP, DIP and fingertip as well as a wrist and palm center (note that the thumb has 3 keypoints instead of 4). We can compute the global position of a keypoint using,

$$p_i(\theta) = T_{b_i}(\theta) \cdot p_i^{b_i}$$

where b_i denotes the index of the bone associated with keypoint i, T_{b_i} is the world space bone transform which is a function of the hand pose θ and $p_i^{b_i}$ is the keypoint position in the local space of bone b_i. Lastly, the mesh model is rigged using traditional linear blend skinning.

3.2 Camera configuration

We use four VGA, synchronized, global-shutter cameras to drive our hand tracker (Figure 3). Each camera covers a 150° (width), 120° (height) and 175° (diagonal) FOV. Figure 3 shows the coverage from this camera configuration. Regions in the lower center are covered by two or more cameras (STEREO, red/orange/green), ensuring that the most accurate tracking is available in the regions

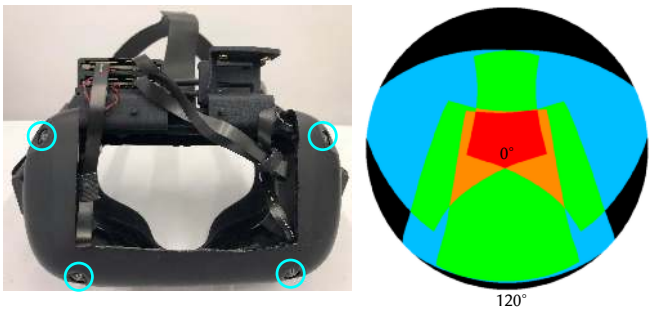

Fig. 3. **Camera configuration.** On the left is a frame holding the 4 monochrome VGA fisheye cameras (circled in blue) that we use for hand-tracking. The frame has a hollow front plate so users can see their hands for data collection purposes. On the right we plot the combined field of view at 50cm distance from the center of the 4 cameras. The angle increases linearly as we move out from the center of the plot. 0° corresponds to the forward-facing (imagine looking forward and extending a ray from the bridge of your nose). We color code the areas by how many cameras can see them: 4(red), 3(orange), 2(green), 1(blue), 0(black).

where ergonomic interactions are likely to occur. Hands are tracked with a single camera in the side regions (MONOCULAR, blue), though with degraded accuracy (Section 5). This ensures the widest possible working volume.

3.3 Hand detection

The task of hand detection is to find the bounding box of each hand in every input image. A key challenge is to ensure robustness to a variety of real world environments. To tackle this challenge, we collect a large and varied hand detection dataset specific to this camera configuration using a semi-automatic labeling method (Section 4.2) and propose a simple and efficient CNN architecture which we name DetNet.

DetNet is inspired by lightweight single shot detectors (SSD)[Liu et al. 2016; Redmon and Farhadi 2017] which simultaneously localize and classify. We further leverage the fact that there is a fixed number of outputs (at most two hands) for any input. Hence, we design DetNet to directly regress the 2D hand center and scalar radius for each hand from the VGA-resolution input image. In addition, a scalar "confidence" output indicates whether the given hand is present in the image. Please see the supplementary section for more details on the network architecture.

During training, we apply the loss function,

$$L = \sum_{i \in \{\text{left,right}\}} L_{\text{loc},i} + \lambda L_{\text{conf},i} \, ,$$

where $L_{\text{loc},i}$ supervises the bounding circles using an MSE loss, $L_{\text{conf},i}$ supervises the hand confidence loss using the standard binary cross entropy loss, and the coefficient λ balances the contribution of the two terms ($\lambda = 100$ when we train DetNet). The losses are summed over the left and right hands. Our detection bounding boxes are the squares that minimally inscribe the circles produced by the network. We only generate boxes when the confidence is above 0.5.

Detection-by-tracking. In general, it is necessary to run DetNet on all four images to guarantee finding both hands in all views. Since we target mobile architectures, this is too expensive to run at every frame, even for our power-optimized DetNet architecture. To overcome this limitation we employ a *detection-by-tracking* approach when a tracked hand is available. We first extrapolate the current hand pose from the previous two tracked poses, $\hat{\theta}_t = 2\theta_{t-1} - \theta_{t-2}$. Detection-by-tracking leverages this extrapolated pose $\hat{\theta}_t$ by projecting the hand keypoints into each camera and computing the minimum enclosing circle as the detection result. In the case that no current hand is tracked, we run DetNet for the next frame. We further reduce compute of this evaluation by running DetNet only on *one* of the cameras (in a round robin fashion). Once the hand has been detected and tracked in a single image, bounding boxes in the remaining cameras in the next frame can be obtained using the tracked pose, allowing for subsequent STEREO tracking. The resulting system can (re-)acquire the hands almost instantly (within two timesteps) while incurring the cost of only a single evaluation of DetNet (or none, when both hands are being tracked). Detection-by-tracking has the added benefit that the resulting bounding boxes are temporally smooth; jittering of the bounding boxes from running DetNet at every frame would otherwise cause more noise later in the pipeline. Please see Section 5.4 for detailed quantitative evaluation of the improvements afforded by these choices.

3.4 Keypoint estimation

Our keypoint estimation network, KeyNet, predicts the 21 keypoints on the hand from a crop of the image based on the predicted bounding box from the hand detection step. Previous work on keypoint estimation typically treats each image independently. This has several drawbacks for real-time, multi-camera systems. First, the quality of predictions will degrade when the hand moves between overlapping camera views as each view is handled independently and views where the hand is partially out-of-frame are problematic for keypoint estimation. Second, the keypoints tend to jitter, particularly for occluded fingers, because temporal consistency is not enforced.

To resolve both problems, we structure our network to explicitly incorporate the extrapolated keypoints as an additional network input. In Section 5, we demonstrate how this architecture can effectively promote both spatial and temporal consistency and lead to a low-jitter tracking system.

The input to KeyNet is the square bounding box crop from the hand detection step. To ensure the entire hand is visible, we increase the crop size by 20%. Similarly to the hand detection step, if a hand is actively tracked, we extrapolate the previous 3D hand pose and project the 21 keypoints to the image. We then augment the input to KeyNet with the 21 keypoint coordinates. Each keypoint coordinate consists of a 2D keypoint coordinate (normalized to $[0, 1]$) and a 1D relative distance coordinate, which we introduce in the next section. If the hand is not actively tracked, this channel is replaced with all zeroes.

The network outputs a 2D heatmap for each of the 21 predicted keypoints, constructed by evaluating a Gaussian following [Tompson et al. 2014]. In addition, we predict a 1D heatmap that encodes the relative distance of each keypoint (see next section). To train

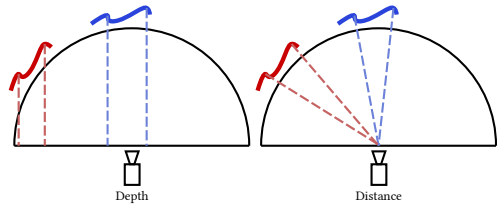

Fig. 4. We use a 2D example to demonstrate the advantage of predicting relative distance over relative depth. The red and blue curves correspond to the same shape appearing in different locations. The cropped view of these two curves will be identical for an equidistance (fisheye camera) projection. We consider two points on the curve as keypoints. **Left:** while the cropped images of the two curves are identical, the depth values are different. A network that uses the image to predict the relative depth between the keypoints will struggle to resolve the ambiguity in the ground truth. **Right:** the two images have the same relative distance values, so the keypoint network can reliably map the input to a single relative distance prediction.

the network, we use MSE loss to supervise both the 2D and the 1D distance heatmaps

$$L_{2D} = \|H_{2D} - \hat{H}_{2D}\|_2^2 + \alpha\|H_{1D} - \hat{H}_{1D}\|_2^2$$

Here H_{2D}, $\hat{H}_{2D}$, H_{1D}, $\hat{H}_{1D}$ are the ground truth and predicted 2D heatmaps and the ground truth and predicted distance heat maps respectively. α is a parameter to balance the two loss terms during training and is set to 0.05 in practice. Our network is designed to only predict keypoints for a left hand. To predict keypoints for the right-hand, we mirror the network inputs and outputs along the x-axis.

Predicting depth vs. distance. Previous work on predicting 3D pose have used 2D image space keypoints and 1D relative *depth* values [Zimmermann and Brox 2017b]. The choice of depth works well for orthographic or weak perspective cameras where depth and image space predictions are orthogonal. However, fisheye cameras are more accurately modeled using the *equidistance projection* [Kannala and Brandt 2006], where rays are parametrized by their angle θ with the camera's principle axis. Predicting depth orthogonal to the image plane is therefore ambiguous for cropped images near the edge of the frame; see Figure 4 (left).

As shown in Figure 4 (right), predicting relative distance instead ensures that a given input image maps to a single output value,

$$d_i^{\text{rel}} = \frac{d_i - \bar{d}}{\phi}$$

Here d_i is the distance of the keypoint i to the camera center, $\bar{d} = \sum_i d_i/n$ is the mean distance across all keypoints and ϕ is the global scale of the hand skeleton (Section 3.6).

3.5 Model based pose estimation

Once we have obtained the 21 3D keypoints of the hand, we solve for the pose of the hand,

$$\theta = \min_{\theta}(E_{2D} + w_1 E_{\text{dist}} + w_2 E_{\text{temporal}}).$$

The 2D error term E_{2D} enforces agreement with the detected 2D keypoints,

$$E_{2D} = \sum_{i,j} \|\Pi_j(p_i(\theta)) - \hat{p}_{i,j}\|_2^2,$$

where Π_j is the function that projects a point in 3D space to the jth camera's image space and $\hat{p}_{i,j}$ is the ith predicted keypoint in the jth camera's image space.

The 1D error term for the relative distance is,

$$E_{\text{dist}} = \sum_{i,j} ||(dist_j(p_i(\theta)) - dist_j(p_0(\theta)) - \phi \cdot (\hat{d}_{i,j}^{rel} - \hat{d}_{0,j}^{rel})||_2^2,$$

where $dist_j$ is a function that computes the distance between a point to the jth camera and $\hat{d}_{i,j}^{rel}$ is the predicted relative distance coordinate for the ith keypoint in the jth camera.

The temporal term is to ensure smoothness of the tracked hand poses

$$E_{\text{temporal}} = ||\theta - \theta_{t-1}||_2^2$$

θ_{t-1} is the hand pose from the previous frame when available. Note that this term assumes a constant position motion model, as we found that a constant velocity model tends to overshoot during fast hand motion. w_1 and w_2 are set to 0.04 and 20 respectively in practice. We use a Levenberg-Marquardt solver and initialize with a valid hand pose from the previous frame when available, or a neutral hand pose otherwise.

3.6 Solving for hand scale

In a single image, there is an inherent ambiguity between the size of a user's hand and its distance from the camera. We resolve this by running an automatic calibration process to estimate the size of the user's hand when the hand is in the STEREO region. We use the same keypoint predictions from KeyNet, but to improve the estimate we incorporate constraints across images from different capture times. We start by sampling frames where either the left or right hand is visible in STEREO. We augment the pose with an additional global scale parameter ϕ (shared between left and right hands) and minimize a multi-frame version of E_{2D}

$$\min_{\theta_t,\phi} \sum_{t=1}^{n} E_{2D}(\theta_t, \phi)$$

where θ_t represents the hand pose at a particular time instance t and $E_{2D}(\theta_t, \phi)$ is the keypoint error across all images captured at time t. Note that solving for pose and scale jointly is key here so the optimizer can adjust the distance to compensate for scale. The block-diagonal structure of the Jacobian matrix allows us to compute the Levenberg-Marquardt search direction in $O(n)$.

4 DATA GENERATION AT SCALE

Similar to [Zimmermann et al. 2019], we find that networks trained on existing RGB hand pose datasets do not generalize well to new environments and camera configurations. We thus found it necessary to generate our own datasets to train both DetNet and KeyNet. The two tasks have different requirements: hand detection requires only that the annotated bounding boxes be accurate, but needs a camera configuration as similar as possible to the target configuration, while keypoint estimation is less sensitive to the camera

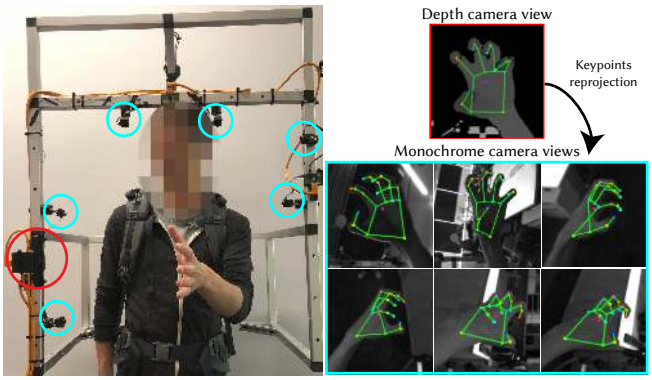

Fig. 5. **Multi-camera system for keypoint annotation.** Left image shows the multi-camera rig with a single depth camera (in red circle) and 6 monochrome cameras (in blue circles). The rig is attached to a backstrap so that user can put it on and walk out to environments with various lightings and backgrounds. Right images show captured frames and generated ground truth. We intentionally place left hand in front of depth camera with minimum occlusion. To this end, the ground truth is generated based on depth camera and projected to other monochrome views.

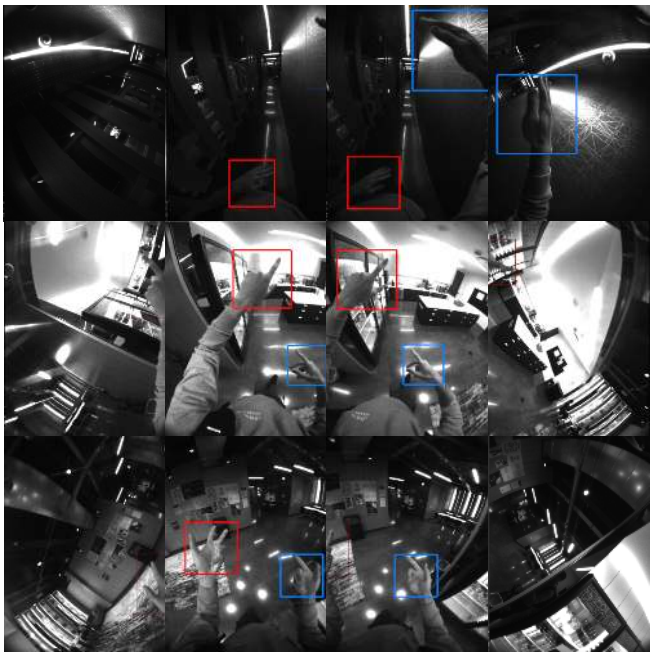

Fig. 6. **Examples of semi-automatically labelled bounding boxes.** Each row corresponds to images from different views captured at the same time. Accurately annotating hand region in images at low light (see Row 1), partially observed hands (see Row 2) and disambiguating left versus right hand is a challenging task for human annotator. On the contrary, our system only needs manual annotation at few frames. Annotation will then propagate to other views and subsequent frames. The examples here are sampled from a sequence of 10k frames where the annotation was done at about 100 bounding boxes per second. Given the capture system is mobile, the sequence features large background and lighting variations.

configuration (since the network sees only a crop) but needs accurate annotation of the entire hand pose. We therefore developed two scalable methods for generating high quality data for training DetNet and KeyNet in real-world scenarios.

4.1 Keypoint labels from depth-based hand-tracking

To generate keypoint labels for training KeyNet, we generate ground truth keypoint annotations using a depth-based hand-tracking system and project the resulting keypoints to several calibrated monochrome views. Fig. 5 shows our hardware setup. Six 60Hz monochrome fisheye cameras, placed to emulate egocentric viewpoints, are attached to a rigid frame. A single depth camera running at 50Hz is used to capture and label the subject's hand motion. The cameras are both spatially and temporally registered to each other, so keypoints generated by the depth-based hand-tracker can be reprojected and interpolated to the monochrome views. Our capture setup is mobile, allowing us to capture lighting and environment variations efficiently.

To ensure high-quality keypoint annotations, it is important to maximize the quality of the depth-based tracking. We therefore try to limit occlusion by instructing users to hold their hand so that the palm always points toward the depth camera, ensuring the fingers stay in view. Because we place the monochrome cameras around the frame, the resulting monochrome frames still capture a variety of more challenging, occlusion-heavy viewpoints that can be used to train KeyNet (Figure 5, right).

We track with a calibrated user-specific template hand model fitted to high quality hand scans [Romero et al. 2017]. We manually label the hand bounding box in the first frame of each sequence and track hands in future frames. Hand poses are estimated using a combination of keypoint estimation [Tompson et al. 2014] and model-based tracking [Tagliasacchi et al. 2015]. After an initial track of a sequence, we re-reprocess it by jointly optimizing the 3D depth error, keypoint error and temporal smoothness across the full sequence. Finally, sequences are manually inspected again in order to discard any problematic frames.

4.2 Semi-automatic bounding box labeling

To train a sufficiently accurate DetNet, we need bounding box labels for data that closely match the camera configuration and environments that we see in the wild. We find therefore that specialized setups as in Section 4.1 are inadequate, and it is better to label images captured from the headset itself (Figure 3). To maximize the throughput and efficiency of the labeling task, we use a novel semi-automatic solution for bounding box labeling, leveraging the detection-by-tracking approach described in Section 3.3. After a user manually labels the hand bounding box for an initial frame, we use a trained KeyNet and our tracking pipeline to propagate the hand pose and thus the resulting boxes for the remaining frames. If the annotator notices that the tracker fails, she simply annotates a new box and the tracked hand is updated automatically. Figure 6 shows some example hand detection data to demonstrate the efficiency of this labeling method.

5 EVALUATION

In this section, we first provide further details on how we train DetNet and KeyNet using the data annotated by the methods we introduced in Section 4. We then describe an evaluation of our pipeline on a separate dataset annotated with a different ground truth mechanism to avoid bias. We also evaluate our pipeline on three public datasets to gain more insights on the strengths and weaknesses of our system.

5.1 Implementation details

Training DetNet. We used the labeling tool introduced in Section 4.2 to generate 2.6 million images for training DetNet. We train the network for 75 epochs using a stochastic gradient descent optimizer with a learning rate of 0.001 and momentum of 0.9. We apply random intensity scaling to the input images as data augmentation to simulate environment and lighting variations.

Training KeyNet. KeyNet takes a 96×96 monochrome image and a 63 dimensional vector (of extrapolated keypoints) as input and outputs $21 \times 18 \times 18$ 2D heat maps and 21×18 1D heat maps. To train KeyNet to handle fast motion and tracking failures, we use a mixture of extrapolation strategies for data augmentation. For 90% of the training data, we compute keypoints from an extrapolated pose calculated from the two previous ground truth hand poses, similarly to what we do at runtime. For the remaining 10%, we compute keypoints from hand poses from 20 frames ago. As indicated by our preliminary experiments (see Section 5.3), directly feeding keypoint features leads to a degenerate network that simply duplicates results of the input keypoints and fails to learn features from image inputs. Hence, we augment the keypoint features by adding Gaussian noise.

To further make KeyNet robust to various real world scenarios, several augmentation strategies are carefully applied during training. First, to simulate partially out-of-frame hands, we randomly wipe out rectangular regions at the image boundary by setting pixel intensity to zero. In addition, we apply random intensity scaling and random geometrical translation, rotation and scaling to the input image and keypoints, to simulate lighting and view-point variations. To simulate the initialization phase when no keypoint features are available, we set the keypoint branch to all zeroes with a certain probability. This augmentation strategy further facilitates the network to learn features from the images without relying on input keypoint features. Our training data set contains ˜2 million examples and we apply all combinations of the aforementioned augmentation strategies at every training iteration. We train the network for 75 epochs using a stochastic gradient descent optimizer with a learning rate of 0.025 and momentum of 0.9.

Note that while many detection and keypoint estimation architectures [He et al. 2017] are trained end-to-end, we trained KeyNet and DetNet separately. This has the advantage of allowing us to use separate datasets for KeyNet and DetNet with different properties as described in Section 4. Moreover, because KeyNet does not rely on evaluation of DetNet, we are able to use detection-by-tracking, which we show is both more efficient and more accurate in Section 5.4.

Hand scale calibration. VR headsets include hardware to detect when they have been placed on or taken off a user's head. We therefore know when we are potentially tracking a new user and thus begin scale calibration. The process takes ˜5 seconds. For offline evaluation, we run calibration using the first 100 frames of a sequence and re-track using the updated hand model to generate results.

Inference. When running our system in real-time, we limit DetNet to evaluate only on one image, and KeyNet to evaluate on no more than two images per hand at any given frame to reduce compute. Our system runs at 60Hz on a PC with an NVIDIA GTX 1080 GPU with a total system of latency of ~60 ms, including rendering in VR. We further trained two models, KeyNet-F and DetNet-F, that require significantly lower compute, which enable us to run on a Snapdragon 835 mobile processor with a Hexagon v62 DSP at 30Hz.

5.2 Evaluation dataset

We collected four evaluation sequences including various finger motion with both slow and fast hand movements and simple hand-hand interactions. To avoid bias, we adopt a different ground truth method than was used during our training. Instead of relying on depth-based hand-tracking, we use a marker-based hand-tracking system [Han et al. 2018] which can achieve sub-millimeter accuracy. We directly placed 3mm semi-sphere markers on the user's skin to minimize appearance modification. Note that these markers only appear in the evaluation set so the trained KeyNet cannot over-fit to the marker appearance.

5.3 Ablative studies of KeyNet

To evaluate the individual contribution of the proposed KeyNet architecture, *3* KeyNet variants are trained,

- KeyNet-S only takes a single image as input without keypoint features.
- KeyNet-N uses the same architecture as the KeyNet but without adding Gaussian noise to the keypoint features during training.
- KeyNet-F is a fast model similar to KeyNet but can be run on a Hexagon V62 DSP in real-time.

Metrics. To measure tracking accuracy, we use mean-keypoint-position-error (MKPE) which computes the average 3D Euclidean distance in millimeters between estimation and ground truth keypoints over all frames. We further measure the temporal smoothness of the estimation by estimating the keypoint acceleration as follows:

$$acc_{i,t} = p_{i,t-1} + p_{i,t+1} - 2p_{i,t}$$

where $acc_{i,t}$ and $p_{i,t}$ are acceleration and keypoint position for the ith keypoint in the tth frame. Given the absence of abrupt finger motion for common VR/AR interaction cases, acceleration can be a good approximate measure of temporal smoothness. We report mean-keypoint-acceleration (MKA) which computes the average of the acceleration of all the keypoints.

Quantitative evaluation. We divide tracking performance into two scenarios: tracking in STEREO and tracking in MONOCULAR. Since the distribution of hand poses is different in the stereo versus monocular tracking regions, we simulate MONOCULAR by running our pipeline

Table 1. Ablative study by using different variants of proposed KeyNet and hand model obtained from different sources. We use generic, calibrated and scanned to refer to the default hand model, the hand model obtained by solving for the hand scale in STEREO and the hand model obtained from a scanning system respectively. We highlight the rows corresponding to our proposed methods in the table in each section.

Method	STEREO		MONOCULAR	
	MKPE	MKA	MKPE	MKA
Ground truth	NA	1.95	NA	1.95
KeyNet-S + scanned	11.4	5.47	15.1	6.49
KeyNet-N + scanned	17.6	2.13	29.7	2.50
KeyNet-F + scanned	11.3	3.07	14.9	3.79
KeyNet + scanned	**11.0**	**2.96**	**15.7**	**3.72**
KeyNet + generic	19.7	2.85	38.8	3.36
KeyNet + calibrated	**11.4**	**2.94**	**15.6**	**3.65**
KeyNet + scanned	11.0	2.96	15.7	3.72

on the same frames of STEREO, except with one view dropped. This produces a fair comparison as the same hand poses and backgrounds are used for both conditions.

The middle section in Table 1 shows the performance of different network architectures when using the hand model obtained from a scanning system. Our proposed KeyNet generated similar MKPE compared to the baseline KeyNet-S but with significantly lower MKA in both STEREO and MONOCULAR. This shows that our proposed KeyNet architecture effectively improves temporal smoothness by incorporating keypoint features without compromising accuracy. Moreover, we verify the effectiveness of data augmentation, which reduces the MKPE in STEREO/MONOCULAR from 17.6mm/29.7mm (KeyNet-N) to 11.0mm/15.7mm (KeyNet). KeyNet-N generates lower MKA by overly relying on temporal information and thus fails to faithfully predict the hand poses. Notably, our KeyNet-F uses much fewer weights (about 7% of the original size) compared to KeyNet but still generates comparable accuracy and smoothness to KeyNet.

The bottom section in Table 1 shows the importance of solving for the hand scale (Section 3.6). When a generic hand model is used, the system accuracy is greatly degraded compared to using the hand model obtained from a scanning system. This degradation is even worse when the tracker runs in MONOCULAR since resolving depth ambiguity in a single view depends heavily on the accuracy of the hand model scale. Using our proposed method for resolving hand scale, tracking accuracy approaches that of the hand model fitted to a 3D scan.

Qualitative evaluation. Figure 7 shows a challenging case where a hand is partially out of view. The baseline KeyNet-S model can't make meaningful prediction for the thumb tip. However, our proposed KeyNet can effectively leverage tracking history to make a better prediction. Figure 8 shows keypoint predictions by the baseline KeyNet-S and proposed KeyNet in two different views. Our proposed KeyNet again leverages the tracking history so that the predictions are consistently correct across the two views.

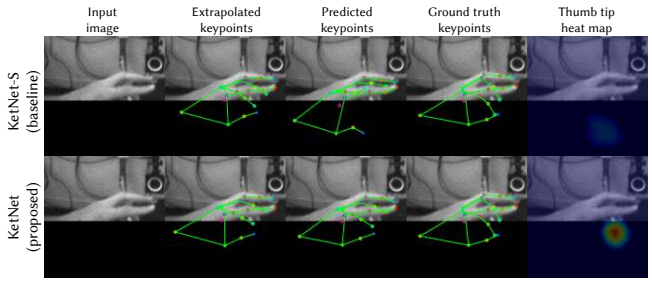

Fig. 7. **Temporally consistent predictions by KeyNet.** The hand is partially out of the image boundary in the input image but our proposed KeyNet can still utilize tracking history to produce a plausible hand pose compared to the baseline KeyNet-S.

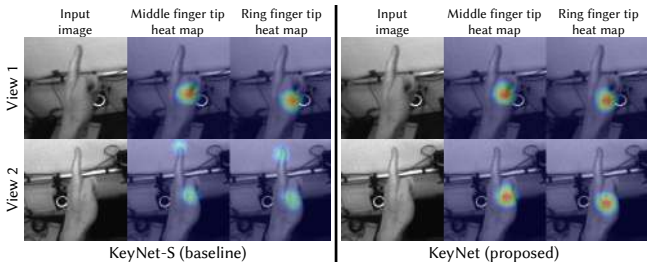

Fig. 8. **Spatially consistent predictions by KeyNet.** Baseline model KeyNet-S produces correct predictions for the middle and ring finger tips in view 1 but produces 2 modes for each keypoint in view 2 due to more severe occlusion. Our proposed KeyNet leverages the input keypoint features and produces consistent predictions across the 2 views.

Table 2. We evaluate variants of our hand detection method through changing the number of views per frame available to DetNet (1 or 4) and toggling detection-by-tracking (track or no-track). DetNet-F corresponds to the fast model that can be run on a mobile processor. This table shows the precision and recall metrics using different hand detection methods for each of the 4 camera views: top left (TL), bottom left (BL), bottom right (BR), top right (TR).

Detection method	TL	BL	BR	TR
DetNet x 4 + no-track	0.94/0.93	1/0.99	1/0.97	0.93/0.95
DetNet x 4 + track	0.98/0.99	1/1	1/1	0.99/1
DetNet x 1 + track	0.98/0.99	1/1	1/1	0.99/1
DetNet-F x 1 + track	0.98/0.99	1/0.99	1/1	0.99/1

5.4 Ablative studies of hand detection

Precision and recall metrics. We use precision and recall to measure the accuracy of each hand detection method. Specifically, we define a positive prediction if the hand confidence predicted by DetNet is greater than 0.5. We define a true positive prediction if it satisfies three criteria: (1) the width of the predicted bounding box is within 20% difference from that of the ground truth bounding box; (2) all the projected ground truth keypoints lie within the cropped image boundary; and (3) more than 16 ground truth keypoints are within the full image boundary.

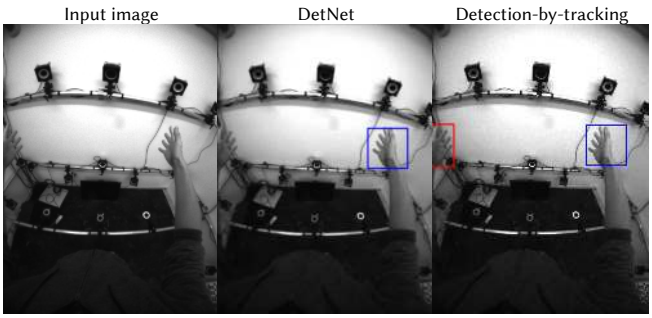

Fig. 9. We follow a detection-by-tracking paradigm to localize hand. Leveraging results from tracking history makes our method robust to challenging cases, *e.g.* , partial occlusion.

Quantitative evaluation. In our system, the views of the hands in different cameras could be drastically different. For example, the bottom cameras tend to capture more of the user's body and the surroundings as the background whereas top cameras tend to capture more of the ceiling. This poses a problem for how hand detection methods perform in each different view. Table 2 shows precision and recall metrics using different hand detection methods for each camera view. When we run DetNet for all the images at every single frame, the performance of hand detection for the top two cameras are noticeably worse than the bottom two cameras, suggesting different performance of DetNet on different camera views. Using detection-by-tracking approach as described in Section 3.3, performance improved for all views and the performance difference between different views becomes much smaller. This shows that detection-by-tracking is more robust than relying on DetNet at every frame. We also show that if we limit DetNet to run on only one view per frame in the optimized runtime setting, the metrics are not obviously affected. This suggests our optimization strategy has negligible impact on the system performance. Finally, our fast model DetNet-F performs similarly to the default DetNet when we enable detection-by-tracking.

Qualitative evaluation. Figure 9 shows a hand at the image boundary where DetNet fails to detect the left hand, but the proposed detection-by-tracking approach succeeds. More broadly, DetNet has different performance in different image regions, especially if the training data distribution is biased, whereas detection-by-tracking is not sensitive to where the hand appears in an image and generates consistent predictions across different image regions.

5.5 Evaluation on different hand motions

Our dataset consists of several types of hand motion designed to realistically stress the tracking system. The finger motion sequence is intended to test how well our system captures subtle finger motion in STEREO. The hand-hand sequence is intended to test how our system handles mild inter-hand occlusions in STEREO. The slow and fast motion sequences are intended to test how our system captures hand motion in a large volume (both in STEREO and MONOCULAR). Table 3 shows the system performance for each sequence. Our system generates the highest accuracy for the finger motion sequence.

Table 3. Tracking accuracy and temporal smoothness of our system evaluated on different motion sequences. MKA GT measures MKA of the ground truth keypoints.

sequence	MKPE	MKA	MKA GT
finger motion	9.4	2.1	1.4
hand-hand	11.4	2.2	1.4
slow motion	11.8	3.8	2.3
fast motion	14.7	5.9	3.5

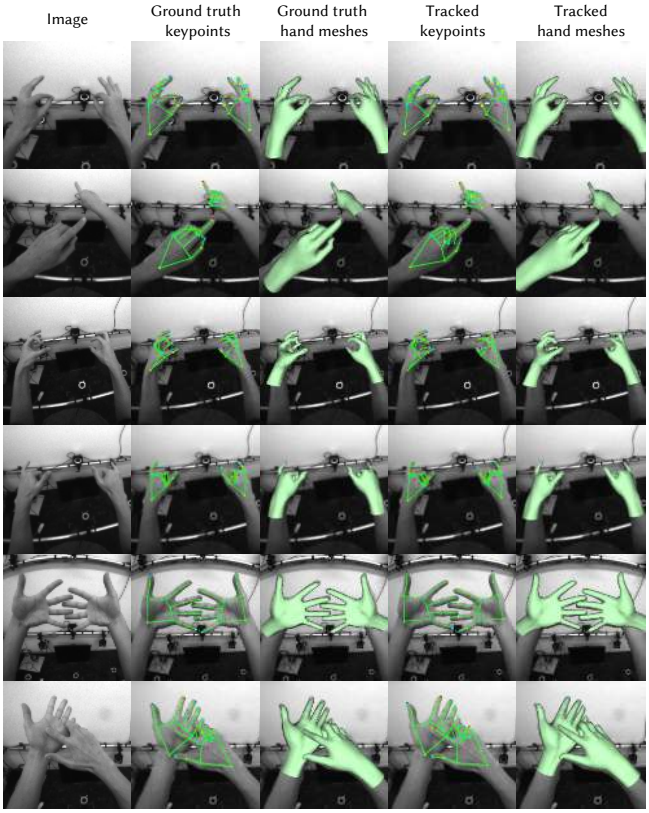

Fig. 10. **Qualitative results on our evaluation dataset.** Each row shows the input image, ground truth keypoints, reconstructed hand mesh and corresponding estimations (from left to right).

Inter-hand occlusion in the hand-hand sequence is more challenging and the accuracy of our system drops slightly compared to the finger motion sequence. Notably while we did not train our KeyNet with any hand-hand data, our system still handles some challenging cases reasonably well. Range of motion is much larger in slow motion and fast motion sequences. As a result, more frames are tracked in MONOCULAR only and the overall tracking accuracy dropped compared to the other two sequences where hands are always tracked in STEREO. Figure 10 shows some examples from each sequence.

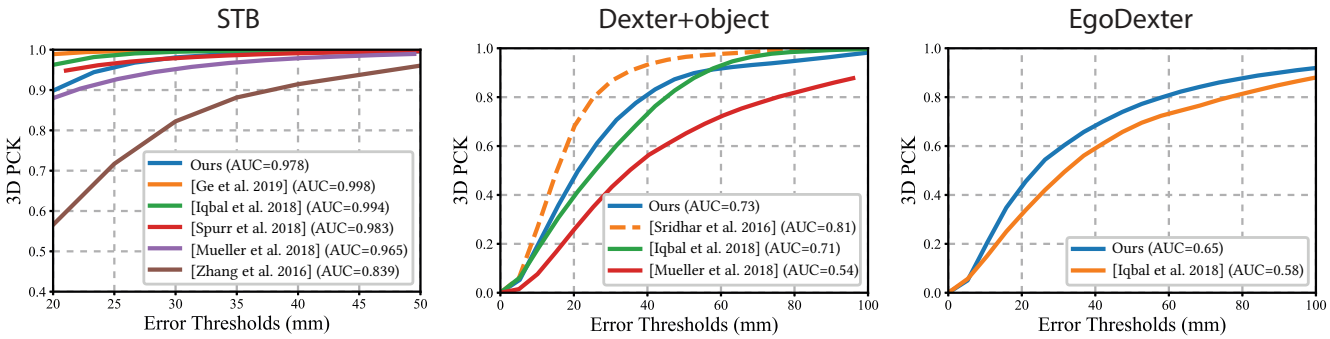

Fig. 11. **PCK plots on public benchmarks.** From left to right, we show the percentage of correct keypoints with respect to different error thresholds across all frames on STB[Zhang et al. 2016], Dexter+object[Mueller et al. 2017] and EgoDexter[Sridhar et al. 2016] benchmarks. The AUC is shown in the legend. The depth-based method [Sridhar et al. 2016] is drawn with dashed line in the middle plot.

5.6 Evaluation on public data sets

In addition to evaluating on the test set that we collected, we also show evaluation results on three public data sets: Stereo Hand Pose Tracking Benchmark (STB) [Zhang et al. 2016], EgoDexter (ED) [Mueller et al. 2017] and Dexter+Object (D+O) [Sridhar et al. 2016]. Each data set contains RGB images only, and we convert these RGB images to monochrome images before running our tracker. Our DetNet is designed specifically for our own camera configuration and therefore doesn't generalize to these data sets. Hence, we manually provide the bounding box either for the first frame or when the tracking fails similar to the labeling method we introduced in Section 4.2 and rely on detection-by-tracking for the rest of the frames. We take the resulting keypoints from the skeleton and report Percentage of Correct Keypoints (PCK) curves. The intent of this exercise is to better understand whether our system generalizes to non-egocentric viewpoints and hand-object interactions since neither is explicitly handled either in our pipeline or data collection.

Evaluation on STB. We follow the split used in [Zimmermann and Brox 2017a] resulting in 15000/3000 training/testing stereo image pairs. We solve for the user hand scale on a training sequence and use the resulting hand model for tracking in the evaluation sequences. Our tracker always takes two views simultaneously for both training and evaluation. To resolve the difference in keypoint definitions, we further fit a linear mapping from the keypoints tracked by our tracker to the ground truth keypoints provided by the dataset, using the training sequences. The 3D PCK curve is shown in Figure 11. From the PCK curve, our result is slightly worse than the state-of-the-art method today. Figure 12 shows some example frames from STB. These poses are commonly seen from egocentric cameras despite being captured from outside-in cameras. We observed most failure cases happen in the last 600 frames in the test set, which contain poses that are not commonly observed from an egocentric point of view (See Figure 13(b)). If we evaluate only on the the first 2400 frames, the AUC reaches 0.992 while the AUC is only 0.929 on the last 600 frames. We further incorporated STB training data to train another KeyNet. When we use this new KeyNet, the AUC score increases to 0.987 with 0.995 on the first 2400 frames and 0.954 on the last 600 frames. .

Evaluation on D+O. D+O contains data from a single RGB camera with 2 users interacting with objects. Due to the inherent scale ambiguity of a monocular viewpoint, we generate a hand model for each user by taking a base model and sweep over the hand scale, selecting the model with the best tracking accuracy according to the ground truth labels. By doing this, we hope to understand the *upper-bound* of our tracking accuracy on this data set. Comparing to methods that do not use depth, we achieve state of the art results on this data set (see Figure 11). Some example results can be seen in Figure 12. Because this dataset only contains sparse labels for the visible fingertips, the most challenging frames with heavy occlusion have no labels and hence, do not affect the overall metric. For example our system generates an incorrect pose (problematic index and ring finger positions) for the case shown in Figure 13(d) but the PCK curve is not affected by this frame.

Evaluation on ED. ED contains fast free hand motion and complex hand-object interactions from an egocentric camera. We get hand models using the same method as for D+O. We again achieve state-of-the-art results on this data set (see Figure 11). However, this dataset has the same issues as D+O in that challenging frames do not contain ground truth labels. We show some example results in Figure 12. Our tracker performs well on the free hand motion sequences and some frames of the hand-object interactions. However, it fails catastrophically when an object severely occludes a hand (see Figure 13).

5.7 Typical failures and future work

Figure 13 summarizes some typical failure cases of our system. The problem of poor performance on uncommon viewpoints of the hand could be resolved with better sampling of those viewpoints in the training set. The failures for hand-hand and hand-object interactions reflect both the limitation of our system design and the fundamental difficulty of these tasks. We believe jointly reasoning about the two hands and handheld objects is an important direction for a better hand-tracking system.

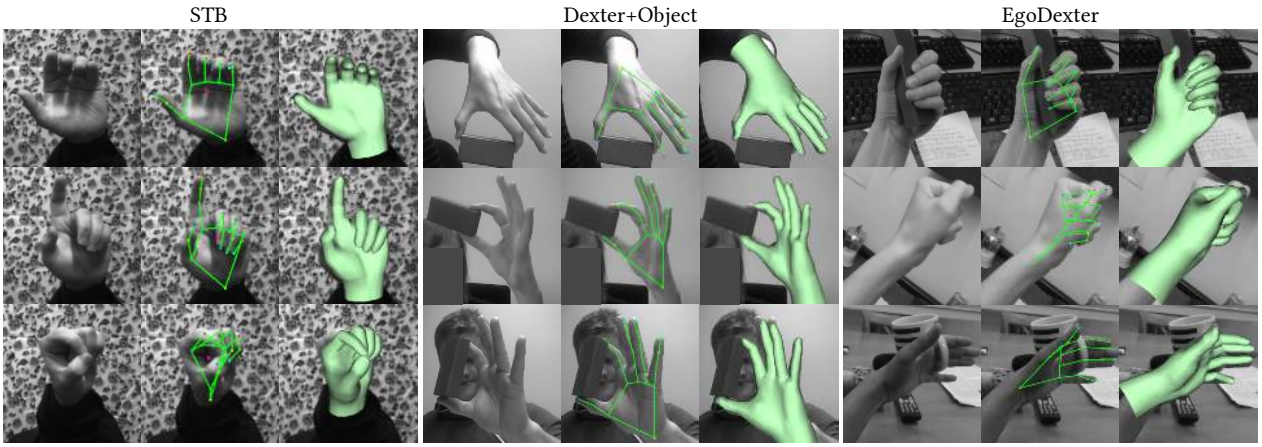

Fig. 12. Tracking results using our system on examples from STB, Dexter+Object and EgoDexter datasets.

Fig. 13. Our system usually fails under complicated hand-hand interactions (a), uncommon view of the hand for egocentric cameras (b) and hand-object interactions (c), (d).

6 CONCLUSION

We have presented a practical real-time hand-tracking system for VR/AR interaction. The system uses four egocentric fisheye cameras with partially overlapping FOV, enabling a large tracking volume. Our proposed hand detection network, DetNet, combined with a detection-by-tracking strategy gracefully handles hands moving between the multiple cameras. Our proposed keypoint estimation network, KeyNet, also leverages tracking information to achieve spatially and temporally consistent keypoint predictions, enabling our system to generate accurate, low-jitter hand motion suitable for interaction. Our model-based pose estimation and hand scale calibration use a traditional linear blend skinning rig, making it easy to incorporate into interactive experiences. Most previous work has only focused on individual components, making it hard to understand how a hand-tracking system can perform in practice. In our work, we demonstrate that enabling experiences using hand-tracking requires a system level design from camera configuration to output representation.

There remain many limitations of our work. Our system can handle interactions driven by a single hand in the air, but the proposed architecture is not designed to reason about hand-hand or hand-object interactions. Both hand-hand and hand-object interactions are critical for immersion and are a direction for future work. Our hand scale calibration is limited to varying a single parameter of the hands (scale), which may not deliver sufficient accuracy for some scenarios.

ACKNOWLEDGMENTS

We thank Kyle Sorge-Toomey, Jenny Spurlock, Matt Longest, Jordan Massey and Kenrick Kin for designing and building the VR experiences we showed in the supplementary video.

REFERENCES

Vassilis Athitsos and Stan Sclaroff. 2003. *Estimating 3D hand pose from a cluttered image.* Technical Report. Boston University Computer Science Department.

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *Proceedings of European Conference on Computer Vision*. Springer, 640–653.

Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 2019. 3D Hand Shape and Pose from Images in the Wild. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR).*

Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. 2018. Weakly-supervised 3d hand pose estimation from monocular rgb images. *ECCV, Springer* 12 (2018).

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 (ICCV).*

Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, Peter Vajda, Matt Uyttendaele, and Niraj K. Jha. 2019. ChamNet: Towards Efficient Network Design Through Platform-Aware Model Adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Martin de La Gorce, Nikos Paragios, and David J Fleet. 2008. Model-based hand tracking with texture, shading and self-occlusions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR).* IEEE, 1–8.

Endri Dibra, Thomas Wolf, Cengiz Oztireli, and Markus Gross. 2017. How to Refine 3D Hand Pose Estimation from Unlabelled Depth Data?. In *3DV.*

Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 2019. 3D Hand Shape and Pose Estimation from a Single RGB Image. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR).*

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, Gül Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J. Black, Ivan Laptev, and Cordelia Schmid. 2019. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*. https://www.di.ens.fr/willow/research/obman

K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2980–2988.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* (2018), 185:1–185:15. Two first authors contributed equally.

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*.

J. Kannala and S. S. Brandt. 2006. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 8 (Aug 2006).

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*.

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 IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A. Otaduy, Dan Casas, and Christian Theobalt. 2019. Real-time Pose and Shape Reconstruction of Two Interacting Hands With a Single Depth Camera. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).

Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In *Proceedings of International Conference on Computer Vision (ICCV)*. 10. https://handtracker.mpi-inf.mpg.de/projects/OccludedHands/

Markus Oberweger, Gernot Riegler, Paul Wohlhart, and Vincent Lepetit. 2016. Efficiently creating 3D training data for fine hand pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2012. Tracking the articulated motion of two strongly interacting hands. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*. IEEE, 1862–1869.

Victor Adrian Prisacariu and Ian Reid. 2011. Robust 3D hand tracking for human computer interaction. In *Face and Gesture*. IEEE, 368–375.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *The IEEE 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).

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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 *CHI*.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. 2017. Learning from Simulated and Unsupervised Images through Adversarial Training.. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Tomas Simon, Hanbyul Joo, Iain A Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images Using Multiview Bootstrapping.. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. 2018. Cross-modal deep variational hand pose estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. 2016. Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input. In *Proceedings of European Conference on Computer Vision (ECCV)*. 17. http://handtracker.mpi-inf.mpg.de/projects/RealtimeHO/

Björn Stenger, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. 2006. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9 (2006), 1372–1384.

James S Supancic, Grégory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. 2015. Depth-based hand pose estimation: data, methods, and challenges. In *Proceedings of International Conference on Computer Vision(ICCV)*.

Andrea Tagliasacchi, Matthias Schroeder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. 2015. Robust Articulated-ICP for Real-Time Hand Tracking. *Computer Graphics Forum (Symposium on Geometry Processing)* 34, 5 (2015).

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), 143.

Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2017. Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Transactions on Graphics (TOG)* (2017).

Bugra Tekin, Federica Bogo, and Marc Pollefeys. 2019. H+O: Unified Egocentric Recognition of 3D Hand-Object Poses and Interactions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Transactions on Graphics (TOG)* (2014).

Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. 2019. Self-supervised 3D hand pose estimation through training by fitting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Robert Y Wang and Jovan Popović. 2009. Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)* 28, 3 (2009), 63.

Linlin Yang and Angela Yao. 2019. Disentangling Latent Hands for Image Synthesis and Pose Estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. 2017. BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recogntion(CVPR)*.

Hao Zhang, Zi-Hao Bo, Jun-Hai Yong, and Feng Xu. 2019a. InteractionFusion: real-time reconstruction of hand poses and deformable objects in hand-object interactions. *ACM Transactions on Graphics (TOG)* 38, 4 (2019).

Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 2016. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214* (2016).

Xiong Zhang, Qiang Li, Wenbo Zhang, and Wen Zheng. 2019b. End-to-end Hand Mesh Recovery from a Monocular RGB Image. In *Proceedings of International Conference on Computer Vision(ICCV)*.

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

Christian Zimmermann and Thomas Brox. 2017b. Learning to Estimate 3D Hand Pose from Single RGB Images. In *IEEE International Conference on Computer Vision (ICCV)*. https://lmb.informatik.uni-freiburg.de/projects/hand3d/ https://arxiv.org/abs/1705.01389.

Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. 2019. FreiHAND: A Dataset for Markerless Capture of Hand Pose and Shape from Single RGB Images. arXiv:1909.04349 [cs.CV]

A NETWORK ARCHITECTURES

DetNet. The input to DetNet is a VGA (640x480) resolution image. Table 4 shows the DetNet-F architecture. The backbone contains a 4×4 average pooling layer downsampling the image to $160x120$ resolution. We use the efficient inverted residual building block (IRB) [Dai et al. 2019; Sandler et al. 2018] to build our network. This efficient architecture takes 5ms to evaluate a single image on a Hexagon v62 DSP. Our default DetNet uses Resnet34 [He et al. 2016] as the backbone with the rest of the network the same as DetNet-F. This architecture takes 2ms to evaluate a single image on a NVIDIA GTX 1080 GPU.

KeyNet. The iput to KeyNet is a 96x96 resolution image and a 63 dimensional keypoint feature vector. Table 5 shows the KeyNet-F architecture. It uses the same IRB as used in DetNet. This efficient architecture takes 14ms to evaluate 4 images on a Hexagon v62 DSP. Our default KeyNet again uses a variant of Resnet34 as the backbone with conv2_x, conv3_x and conv4_x as the "backbone image layers" and conv5_x as the "backbone fused layers". The resuling network takes 8ms to evaluate 4 images on a NVIDIA GTX 1080 GPU.

Table 4. Architecture for DetNet-F

Stage/Output	Input	Operator	Exp size	Out Channels	Stride	Repeat
Backbone	$1 \times 640 \times 480$	AvgPool2d 4×4	-	1	4	1
	$1 \times 160 \times 120$	Conv2d 3×3, BN, Relu	-	32	2	1
	$32 \times 80 \times 60$	IRB 3×3	96	32	2	1
	$32 \times 40 \times 30$	IRB 3×3	96	32	1	1
	$32 \times 40 \times 30$	IRB 3×3	192	64	2	1
	$64 \times 20 \times 15$	IRB 3×3	384	64	1	2
	$64 \times 20 \times 15$	IRB 3×3	384	64	2	1
	$64 \times 10 \times 8$	IRB 3×3	384	64	1	3
	$64 \times 10 \times 8$	IRB 3×3	384	96	1	1
	$96 \times 10 \times 8$	IRB 3×3	576	96	1	2
	$96 \times 10 \times 8$	IRB 3×3	576	128	2	1
	$128 \times 5 \times 4$	IRB 3×3	768	128	1	2
$160 \times 5 \times 4$	$128 \times 5 \times 4$	IRB 3×3	768	160	1	1
Hand center	$160 \times 5 \times 4$	Conv2d 1×1, BN	-	4	1	1
	$4 \times 5 \times 4$	AvgPool2d 5×4	-	4	1	1
2×2	$4 \times 1 \times 1$	Reshape 2×2	-	1	-	1
Hand radius	$160 \times 5 \times 4$	Conv2d 1×1, BN	-	2	1	1
$2 \times 1 \times 1$	$2 \times 5 \times 4$	AvgPool2d 5×4	-	2	1	1
Hand cls	$160 \times 5 \times 4$	Conv2d 1×1, BN	-	2	1	1
	$2 \times 5 \times 4$	AvgPool2d 5×4	-	2	1	1
$2 \times 1 \times 1$	$2 \times 1 \times 1$	Sigmoid	-	2	-	1

Table 5. Architecture for KeyNet-F

Stage/Output	Input	Operator	Exp size	Out Channels	Stride	n
Backbone image	1×96^2	Conv2d 3×3, BN, Relu	-	32	2	1
	32×48^2	IRB 3×3	1	32	1	1
	32×48^2	IRB 3×3	96	32	2	1
	32×24^2	IRB 3×3	192	32	1	1
	32×24^2	IRB 3×3	192	64	2	1
64×12^2	64×12^2	IRB 3×3	384	64	1	2
Backbone keypoints	63	Linear, Relu	-	4608	-	1
32×12^2	4608	Reshape 12×12	-	32	-	1
Backbone fused	$64 \times 12^2, 32 \times 12^2$	Concat	-	96	-	1
	96×12^2	IRB 3×3	384	64	1	2
	64×12^2	IRB 3×3	384	64	1	3
	64×12^2	IRB 3×3	384	96	1	1
	96×12^2	IRB 3×3	480	96	1	2
	96×12^2	IRB 3×3	576	128	2	1
	128×6^2	IRB 3×3	768	128	1	2
128×6^2	128×6^2	IRB 3×3	768	160	1	1
Keypoint heatmap	160×6^2	Conv2d 3×3 pad2, BN, Relu	-	63	1	1
	63×8^2	ConvTranspose2d 2×2	-	42	2	1
21×18^2	42×16^2	Conv2d 3×3 pad2, BN, Relu	-	21	1	1
Keypoint distance	160×6^2	AvgPool2d 6×6	-	160	6	1
	160×1^2	Conv2d 1×1, Relu	-	378	1	1
21×18	378×1^2	Reshape 18	-	21	-	1