DeepFisheye: Near-Surface Multi-Finger Tracking Technology Using Fisheye Camera

Keunwoo Park[1] **Sunbum Kim**[1] **Youngwoo Yoon**[2,1] **Tae-Kyun Kim**[3] **Geehyuk Lee**[1]

[1]HCI Lab, School of Computing, KAIST, Daejeon, Republic of Korea
[2]HMI Research Group, ETRI, Daejeon, Republic of Korea
[3]Imperial College London, London, United Kingdom
{keunwoo, ksb4587}@kaist.ac.kr, youngwoo@etri.re.kr, tk.kim@imperial.ac.uk, geehyuk@gmail.com

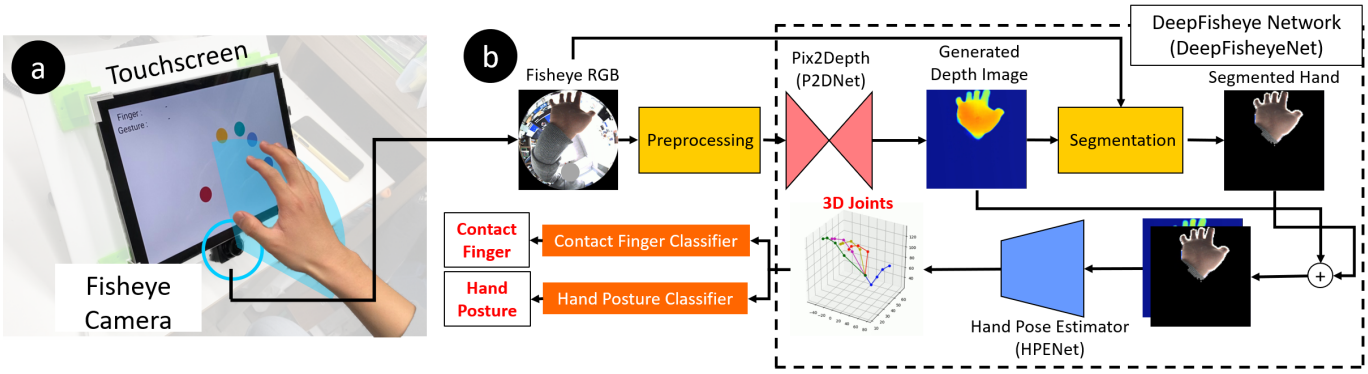

Figure 1. a) DeepFisheye tracks 3D joint locations of multiple fingers near a touchscreen using fisheye camera. Colored points on the touchscreen indicate the projected locations of the fingertips. b) Pipeline of DeepFisheye. Accurate hand postures were estimated by using intermediate depth images. The estimated hand postures were used for mid-air interaction, contact finger classification, and hand posture recognition.

ABSTRACT

Near-surface multi-finger tracking (NMFT) technology expands the input space of touchscreens by enabling novel interactions such as mid-air and finger-aware interactions. We present DeepFisheye, a practical NMFT solution for mobile devices, that utilizes a fisheye camera attached at the bottom of a touchscreen. DeepFisheye acquires the image of an interacting hand positioned above the touchscreen using the camera and employs deep learning to estimate the 3D position of each fingertip. We created two new hand pose datasets comprising fisheye images, on which our network was trained. We evaluated DeepFisheye's performance for three device sizes. DeepFisheye showed average errors with approximate value of 20 mm for fingertip tracking across the different device sizes. Additionally, we created simple rule-based classifiers that estimate the contact finger and hand posture from DeepFisheye's output. The contact finger and hand posture classifiers showed accuracy of approximately 83 and 90%, respectively, across the device sizes.

Author Keywords

Touchscreen; Finger Tracking; Near-Surface; Computer Vision; Deep Learning

UIST'20, October 20–23, 2020, Minneapolis, MN, USA

ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

CCS Concepts

•**Human-centered computing** → **Touch screens;**

INTRODUCTION

Research on mid-air and finger-aware interactions has continued to expand the input space of a touchscreen. The near-surface multi-finger tracking (NMFT) technology tracks the 3D location of each fingertip near an interactive surface. Although it enables mid-air [13, 61, 75] and finger-aware interactions [61, 75, 79], there is no practical solution for NMFT that can be applied to mobile touchscreen devices. An NMFT system should satisfy the following requirements. It should handle various device sizes, work in various environments, have a small physical footprint that can fit into mobile devices, and should not require users to wear and additional accessory. Even though various technologies have been proposed in the human-computer interaction (HCI) field, we could not find one that satisfies all the aforementioned requirements. Vision-based approaches are popular for NMFT because they are more user-friendly than the wearable systems such as magnetic field sensing [9, 11], wherein users are required to wear devices on their hands. However, the hardware configurations of vision-based approaches [20, 61, 75] are not compact, because the camera, which has a limited field of view (FOV), must be distanced from a the hand to capture its sufficient portion. Yu et al. [75] attached a prism to the front camera of a smartphone to solve this problem; however, their solution could not distinguish and track multiple fingers.

We propose DeepFisheye, an NMFT system that can be applied to various mobile devices. It tracks all the fingertips of a

hand above a touchscreen. The system uses a single RGB camera with a 180° fisheye lens (fisheye camera) attached to the bottom center of a touchscreen, as shown in Figure 1. Owing to its wide FOV, the fisheye camera can capture the whole hand above the touchscreen even though the hand and camera are within close distance. Such a compact hardware configuration is favored for mobile devices because no separate hardware is required. DeepFisheye estimates the 3D locations of the full joints of a hand from a fisheye image using deep learning. It enables mid-air interactions by tracking the fingertip locations. Moreover, it can recognize which finger has contacted the touchscreen and the hand posture by using the estimated joint from DeepFisheye.

We designed DeepFisheyeNet, the deep learning model component of DeepFisheye that estimates hand postures through depth estimation. In numerous previous studies [7, 46, 54, 55, 81], hand postures were estimated from cropped images containing only a hand for accurate pose estimation. However, in a fisheye image, the appearance of a hand changes considerably according to its location on the image, owing to distortion. Therefore, to achieve improved accuracy, DeepFisheyeNet generates a depth image from a whole RGB fisheye image, instead of cropping hand images. Furthermore, a large-scale dataset is essential to train the deep learning network; however, no dataset is available for hand pose estimation from fisheye images. Given the significant differences in the appearance of fisheye and flat images (i.e., images following the pin-hole camera model), training on the existing hand pose datasets was not possible. Therefore, we collected two new datasets to train the network. The first dataset, the synthetic fisheye hand dataset comprises a large number of synthetic images of virtual hands with various backgrounds and lighting conditions, and the second, real fisheye hand dataset, comprises real-world images. The latter is smaller but more realistic than the synthetic dataset. These datasets were effectively used by training the network on the synthetic dataset first and then on the real dataset to fine-tune the network.

The contribution of this work is two-fold. First, we suggest a novel NMFT technology, DeepFisheye, that has a practical form factor comprising a single color fisheye camera and a touchscreen. Second, we provide novel hand datasets for fisheye images, which can accelerate vision-based mobile interaction research. Additionally, through a painting application that worsk in real time, we demonstrated how DeepFisheye can expand the touch interface. The codes and datasets will be open to public[1].

RELATED WORK

Near-Surface Finger Tracking

Various approaches have been proposed for finger tracking. One of the most popular approaches is to use glove type devices [11, 16, 19, 30, 42]. However, this approach is not appropriate for mobile interactions, because wearing gloves can be burdensome, especially in mobile situations. Therefore, most near-surface finger tracking technologies for mobile devices used a camera to track single or multiple fingers. Recently, Yu et al. [75] suggested HandSee, that tracks fingertips' 3D locations using a smartphone with a prism attached to its front camera. The prism is used to acquire a reflected image from the touchscreen. The reflected and non-reflected images are used to create a stereo image, and a depth image is estimated from it. Their system can differentiate between the thumb, index, and other fingers. Li et al. [37] attached two wide-angle cameras besides a display. The cameras faced the same direction as that of the display, which cannot directly see a hand above the screen. Therefore, they attached a mirror in front of the camera to shift its viewing direction. Their prototype estimated the distance from the display to a fingertip by utilizing the disparity in two images. WatchSense [61] tracked thumb and index fingers by using a depth camera placed on the user's forearm. These research projects were used for mid-air and finger-aware interactions because they could track multiple fingers. Air+Touch [13] can track a single fingertip near a touchscreen by using a depth camera attached to a smartphone to obtain the image of a fingertip.

However, methods using a camera face a limitations owing to the camera's narrow FOV. The limited FOV of a camera makes it difficult to capture proper hand images when the camera is integrated into a mobile device. Therefore, WatchSense [61] and Air+touch [13] attached the camera apart from the interaction area. Li et al. [37] used mirrors to solve this problem. However, their hardware could not acquire the image of a whole hand when some parts of the hand were not above the display. This makes finger type identification (e.g. index, middle) from a hand image difficult. HandSee's [75] approach failed to cover the whole touchscreen area owing to the camera's limited FOV. Furthermore, owing to the prism's structure, the sensing volume near the prism was very narrow.

3D Hand Pose Estimation Using a Camera

Hand pose estimation has been a major research area in the computer vision field, and it has benefited considerably from the rise of deep learning. Especially, hand pose estimation using a single color image as DeepFisheye, is an active research area [3, 47, 63, 72, 78, 82]. Zimmermann et al. [81] suggested a deep learning pipeline that segments a hand in an image, identifies keypoints, and finally estimates the most likely hand pose. Mueller et al. [46] considered more extreme cases, wherein parts of a hand are occluded. One of the most challenging issues in hand pose estimation is collecting large amount of data. Therefore, Mueller et al. created a large-sized synthetic image dataset. It includes hand images, wherein some parts of the hand are occluded. Based on this dataset, they demonstrated state of the art hand pose estimation performance. Ge et al. [17] proposed a deep learning pipeline that creates a 3D mesh of a hand from a color image. Besides 3D hand pose estimation, CyclopesRing [8] classified hand pose gestures with a fisheye camera located between fingers.

Depth images are commonly used for hand pose estimation [12, 41, 45, 67, 80]. Oberweger et al. [53] estimated hand poses with multi-layer convolution networks from depth images. They used ResNet [25] for their following work [52] to achieve improved accuracy in addition to using data augmentation techniques such as image rotation and translation. Qi and Kim [74] solved the occlusion problem by handling

[1] https://github.com/KAIST-HCIL/DeepFisheyeNet

visible and occluded parts of the hand differently. They allowed multiple states for the occluded joints and designed probabilistic loss. Mueller et al. [48] proposed a method that utilizes both color and depth images.

Mid-Air Interaction Technology

Mid-air interaction is an important research stream in the HCI field. Some related studies used a camera to track fingers [1, 70] and recognize gestures [4, 28, 59]. However, these methods are not suitable for near-surface interactions. Yang et al. [73] used a fisheye camera on a smartphone and a fisheye lens to track fingers near the device. However, they could not track the 3D location of a fingertip. Besides a camera, different types of sensing methods: magnetic field [10, 24, 44], acoustic [50, 77], and electric field [26, 33, 38, 69] have been proposed. However, magnetic field sensing methods require magnets to be attached on the fingertips. The other methods were unable to discriminate the finger type.

Finger Identification Technology

There are various approaches to finger identification, e.g., using color markers [20, 68], wearable sensors [6, 23, 43], magnet [56]. However, wearing sensors or markers would be burdensome during daily activity. Computer vision methods [14, 27, 29, 79] and capacitive sensors [18, 32] were utilized to solve the problem. However, these methods were unsuitable for mobile devices [14, 27, 79], worked only for certain postures [18, 32], and were evaluated with only a single person [29].

HARDWARE CONFIGURATION: FISHEYED SURFACE

We designed a hardware prototype called *FisheyedSurface*. It is a Microsoft Surface device with a fisheye camera attached at its bottom, as shown in Figure 1a. The fisheye camera comprises a fisheye lens with 220° FOV, and a machine vision camera (FLIR CM3-U3-13Y3C-CS). Our goal was to maintain the original form factor of the Microsoft Surface device as much as possible because we wanted to explore the possibility of integrating DeepFisheye into existing smart devices. The fisheye camera was attached to the bottom center of the Microsoft Surface device as closely as possible. It was aligned to match its 180° FOV line with the touchscreen surface. We selected the bottom center location because, if the camera is placed on the top side, it cannot capture the complete shape of the hand, as illustrated in Figure 2b. The bump of the fisheye lens was approximately 3 mm. It may be possible to decrease the bump by scaling down the whole camera system, similar to that in smartphones.

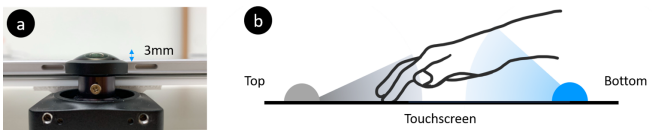

Figure 2. a) Close up view of the fisheye lens. b) Comparison of the two fisheye camera attachment options: top and bottom. Camera when attached to the bottom side is more advantageous to capture the whole hand.

An alternative choice could be a depth camera, which was widely used in hand pose estimation research [2, 52, 53, 76, 74]. However, using a depth camera on a mobile device to capture an interacting hand is not adequate because depth cameras, which use the time-of-flight method are unable to capture a nearby object correctly. Moreover, there is no depth camera with 180° FOV, to the best of our knowledge.

DEEPFISHEYE HAND DATASET

A large-scale dataset is essential for the learning-based hand pose estimation model. Although several hand datasets are available [5, 46, 48, 58, 64, 76], we created our own because the existing ones comprise only flat images that follow the pinhole camera model. Unlike flat images, fisheye images are highly distorted, and the hands are occluded at some locations, as shown in Figure 3. This type of distortion and occlusion can lead to poor performance if we input the fisheye images into a hand pose estimator trained on flat images. Alternatively, we considered dewarping fisheye images, e.g., converting them to equirectangular images. However, polar parts of a fisheye image get stretched and distorted when the image is dewarped to an equirectangular image [15, 35, 62].

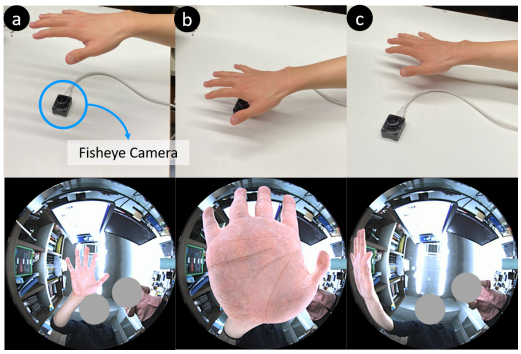

Figure 3. Appearance of a hand in fisheye images differs according to its location relative to the camera. a) Distortion is not severe when the hand is far. b) Considerable amount of distortion occurs when the hand is close to the camera. The fingers look shorter and the palm bigger. c) When the hand moves to the side of the touchscreen, the shape of the hand becomes curved and the fingers are occluded between them.

Creating a dataset is expensive because it requires collecting real images in diverse environments and annotating accurate 3D positions of the hand joints. In fact, other studies have created hand datasets synthetically with virtual hand models [46, 48] for cost effectiveness. However, real and synthetic images differ considerably in several aspects, e.g., texture, lighting, shadows. Owing to this "domain gap," if a deep learning model is trained only with synthetic images, it may not perform satisfactorily with real images [46]. Therefore, we created two datasets. The first, DeepFisheye synthetic dataset, comprises pairs of hand images and 3D joint data. The second is the DeepFisheye real dataset that comprises pairs of real hand images and 3D joint data. The synthetic dataset was used for initial training of a network estimating 3D hand pose, and the real dataset for decreasing the domain gap by fine-tuning the network.

DeepFisheye Synthetic Dataset

DeepFisheye synthetic dataset comprises three types of data: fisheye color image, fisheye depth image, and 3D joint data. During dataset creation, we first collected the hand postures

because we aimed to consider realistic postures that a user can actually assume. We collected this data for four individuals using a Leap Motion device. Among the four individuals, two were the authors of this paper, and the other two were recruited. The participants performed the requested motions above the Leap Motion device. The motions were: (i) folding only one finger from the thumb to pinky finger, (ii) unfolding only one finger from thumb to pinky finger, (iii) folding two consecutive fingers, (iv) unfolding two consecutive fingers, (v) folding and unfolding all the fingers, and (vi) free movement. The participants were asked to slowly assume the postures, while joint data were continuously collected at 120Hz. Approximately 5 s were required to assume a posture.

We created a virtual space and hand models to generate synthetic images, as shown in Figure 4. In the virtual space, color fisheye and depth fisheye cameras were placed at the center. This setting is different from that of the FisheyedSurface, wherein the camera was placed at the bottom side. This is because we aimed to make our dataset generic so as to be used by other researchers. The fisheye camera used the equidistance model that many fisheye lenses follow [36].

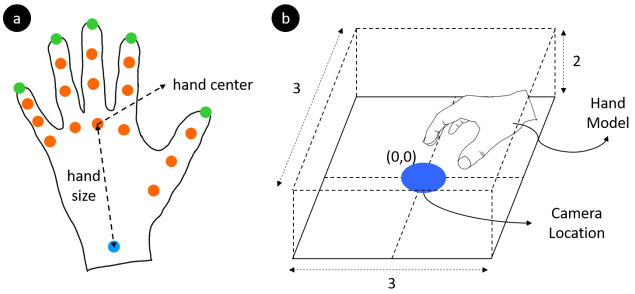

Figure 4. a) Hand joint model. Center of the hand was defined as the middle finger's metacarpophalangeal joint, and size of the hand was defined as the distance between the wrist and the center point. b) Virtual space where the synthetic data were collected.

The hand model has 21 joints as shown in Figure 4a. The center of the hand was defined as the middle finger's metacarpophalangeal (MCP) joint, and the size of the hand as the distance between the wrist and the center of the hand. All the hands were normalized to the unit hand size of 1. There were two right-hand models–one each for the white and black skin colors. Even if we collected the data only for the right hands, we flipped the images and joint data during training to consider both the right and left hands. The size of the virtual space was $3 \times 3 \times 2$, where the center of the hand model could be located. The size of the space was determined by considering the possible hand movement range for the touchscreen of a large tablet, such as Microsoft Surface.

Outlined is the procedure for the dataset creation: (i) we set up a hand model to assume one of the postures that we collected, (ii) the hand model was placed and rotated randomly in the space, and the fingers were stretched with a random ratio between 0.8 and 1.2, (iii) finally, we captured the hand images with the two aforementioned cameras.

The images were augmented through post-processing. The virtual space had no background, which is an unrealistic situation. Therefore, we added background images to the collected hand images. We used 10,000 random images from the Flickr website as backgrounds. Moreover, we randomized the hand color to consider various skin colors and simulate different lighting conditions. For color randomization, the gamma values of the hand images were randomly varied between 0.25 and 2.0. These values were suggested by Mueller et al. [46]. Figure 5 shows sample images from the dataset.

Figure 5. Synthetic image samples. Color fisheye images on the left side and depth fisheye images on the right side of each pair.

DeepFisheye Real Dataset

This dataset comprises pairs of real color fisheye images and 3D hand joint data. We were unable to collect real fisheye depth images because we could not find a depth camera with 180° FOV. We built a device called *FishLeapFrame* to collect the pairs. This apparatus uses the fisheye camera and the Leap Motion device, as shown in Figure 6. We used the Leap Motion device to collect the 3D joint data and assumed the data as ground truth because we could not use data gloves or marker-based tracking systems. These attachments affect the hand appearance, which is crucial in training the model. In previous studies, Leap Motion devices showed fingertip tracking errors from 5 [66] to 17 mm [65], which were accurate for dynamic [40] and static [57] gesture recognition and finger classification [14].The fisheye camera was the same as that in FisheyedSurface.

We defined an imaginary space called the "hand moving space," where a participant moves her hand. Its bottom plane was considered to be a touchscreen, and we captured the hand movement above this imaginary touchscreen. The area started right above the frame, and its height was approximately 20 cm. This was not a strict restriction, but a rough guideline. When a participant moved her hand in the hand moving space, fisheye images were captured by the fisheye camera and the 3D hand joints were collected by the Leap Motion device. The fisheye camera was placed at the lowest surface of the hand moving space. The Leap Motion device was positioned 22 cm below the hand moving space to ensure that the distance between the hand and the device was larger than the minimum sensing range.

Four people, including two authors of this paper participated in the real dataset collection. The participants were required to move their hands to every possible location in the hand moving space for 20 min. Furthermore, they were required to modify hand postures freely and differently. Out of the 20 minutes, participants kept their arms covered with sleeves for 10 min, and left them uncovered by rolling up the sleeves for the remaining 10 min. The purpose of this was to not train the model to not recognize parts of an arm as the hand. We collected all the data at the same location. The collected data comprised 42,000 pairs of images and 3D ground truth joint

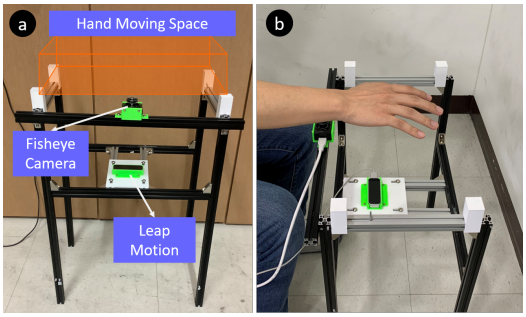

Figure 6. a) FishLeapFrame to collect pairs of fisheye images and 3D hand joint data. b) Participants moved their hands in the "hand moving space," located above the frame.

data. Hand postures in the two datasets may not be as diverse as those in the existing datasets [46, 76] for general hand pose estimation. However, our datasets cover most of the hand postures for multi-finger touchscreen interactions suggested by previous studies [21, 49, 79].

DEEPFISHEYE PIPELINE

DeepFisheye pipeline first preprocesses a fisheye image, and then estimates the 3D locations of all hand joints with DeepFisheye network (DeepFisheyeNet). Then, two rule-based classifiers classify contact fingers and hand postures. Contact fingers and hand postures are two interaction elements defined by us. Figure 1b shows the complete pipeline of DeepFisheye, wherein the final outputs are highlighted in red color.

Preprocessing

The goal of this preprocessing is to reduce the domain gap between real and synthetic images as much as possible using simple computation. Therefore, this step was performed only when the input was a real image. There are two parts in this step. The first part involves fisheye image calibration. Usually, virtual (ideal) and real cameras have different projection model parameters because the curvature of a real lens and alignment of a real image sensor are different than those of an ideal camera. The calibration process compensates these errors by remapping real images to follow an ideal camera's projection model parameters. The calibration process centers the circular image on the actual center of the image, finds the direction to the 3D point with the real fisheye projection parameters, and reprojects the 3D point with the ideal fisheye projection parameters. We used OCamCalib [60] to obtain projection parameters of the real fisheye camera. We verified this calibration process with a checkerboard. The image size was 720×720 px. The calibration mean error was 8.2 px (STD: 1.1 px). It was approximately 1% of the image length.

The next part of preprocessing is image sharpening, or "unsharp masking." One of the major differences between the real and synthetic images as per our observation was sharpness. Therefore, we made the edges in the real images clearer. The image sharpening process was adding a high pass filtered image to the original image.

DeepFisheye Network

DeepFisheyeNet comprises two sub-networks: Pix2Depth (P2DNet) and hand pose estimator (HPENet) networks. Figure 7 is a simple illustration of DeepFisheyeNet. The complete illustration is provided in the supplementary material. Every input image is resized to 256×256 px. Then, P2DNet generates a depth image of the hand from the fisheye color image input. Next, hand parts of the color image are segmented using the generated depth image. Segmentation is a simple process that selects pixels from the color image only when the value of the corresponding pixel on the depth image is larger than 0. The depth image assists HPENet to achieve more accurate hand pose estimation. We also used intermediate hand pose estimators (IHPEs) that estimate joint coordinates from the intermediate feature maps. HPENet estimates the final hand pose, relative to the fisheye camera. Some studies generated depth images of a hand from a color image [51], and then used them to improve the hand pose estimation performance [7, 34]. The key difference between DeepFisheyeNet and the existing networks is that the former generates depth images from a color images and uses both color and depth images for hand pose estimation. Regarding HPENet, it is a modified version of Mueller et al.'s network [48]. The joint decoder in HPENet decodes joints from the encoded features, wherein the modification was made.

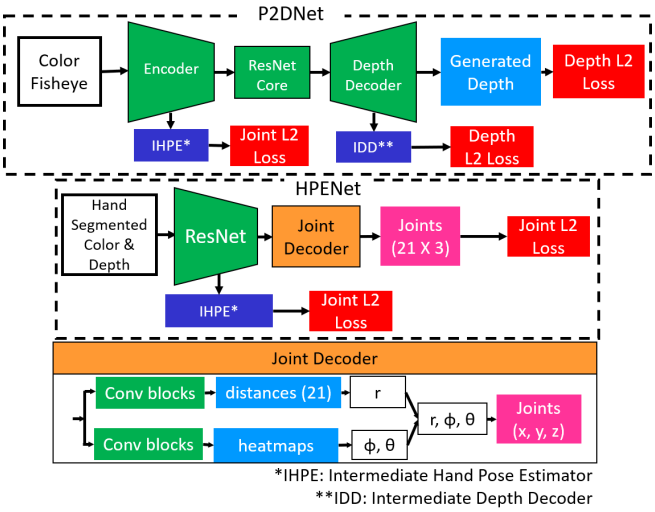

Figure 7. Simple illustration of DeepFisheyeNet. P2DNet generates a depth image, and HPENet estimates 3D joint locations using both the color and depth images. Joint decoder estimates the direction and distance of joint locations separately. Heatmap losses are not included in this figure.

The joint decoder is a specially designed module for DeepFisheyeNet. It outputs 3D joint data represented as spherical coordinates (r, θ, ϕ), and then converted to Cartesian coordinates (x, y, z). Predicting spherical coordinates is preferred than directly predicting Cartesian coordinates because fisheye projection models follow spherical coordinates and the direction and distances are disentangled as shown in Figure 8. This representation is also used in other fisheye image based body pose estimation [71]. Unlike previous networks [48, 71], we used global average pooling layers instead of fully connected layers to increase the network's speed.

The joint decoder has two subparts–one estimates a heatmap for each joint, and the other estimates the distance for each joint, as shown in Figure 7. A heatmap is a 2D probability

Table 1. Training steps and hyperparameters.

Step	Training module	Dataset	Batch size
	Learning rate		Iteration (epoch)
1	P2DNet	Synthetic	32
	1e-3		69,880 (12)
2	DeepFisheyeNet	Synthetic	16
	P2DNet: 1e-4, HPENet: 1e-2		69,880 (6)
3	DeepFisheyeNet	Real	16
	P2DNet: 1e-5, HPENet: 1e-4		31,690 (12)

map of a joint on the image. From the heatmap's maximum point, the direction to the joint, (θ, ϕ), can be calculated using the fisheye projection model, as illustrated in Figure 8. By combining the estimated direction and distance (r), the 3D location of the joint is determined. The IHPEs use the same approach.

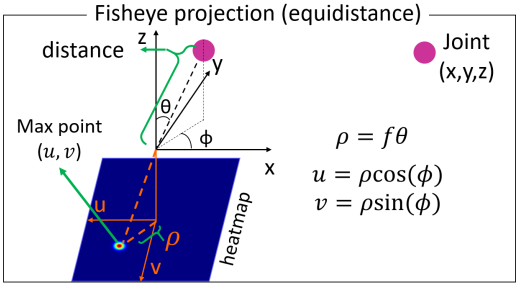

Figure 8. Fisheye (equidistance) projection model. Points with the same direction from the camera (θ, ϕ) are projected on the same point. f is a constant.

Training

The training comprises three steps. First, P2DNet is trained with the synthetic dataset to be used for the next step. Second, the whole DeepFisheyeNet is trained with the synthetic dataset. This step mainly trains the HPENet part. Third, DeepFisheyeNet is fine-tuned with the real dataset. We used Adam optimizer [31] and the parameters listed in Table 1. We used 80% (~186,000) of the synthetic dataset for the training and 20% (~47,000) for testing. The whole real dataset was used for the training.

There were three types of outputs, including intermediate outputs, namely, heatmap, depth, and distance. We calculated L2 losses for all the three types during training, as shown in Figure 7. The ground truth heatmaps were created by applying a 2D Gaussian filter centered at the joint coordinates. In the case of distance outputs, we did not directly calculate the loss; instead, we calculated the loss with the joint output (joint loss) and the loss was backpropagated. Moreover, different weights were applied to the losses (joint loss = 1.0, intermediate joint loss = 0.5, heatmap loss = 250, intermediate heatmap loss = 125).

One of the challenges that we faced was the absence of fisheye depth images in the real image dataset. Therefore, depth losses that comprise an important type of loss for P2DNet were not available for step 3 of the training. However, the early part of P2DNet could be trained because of IHPEs that helped to fine-tune P2DNet with real images. Without IHPEs, the P2DNet could not generate proper depth images.

We transformed the images during training in all the steps. We randomly flipped them and randomized their brightness, contrast, saturation, and hue. Moreover, we blurred the synthetic images with a Gaussian filter because they were considerably sharp compared to the real images. We rescaled the hand joint data so that the hand size was 1. This was done to match the scales of synthetic and real data. Therefore, when DeepFisheye is used in real images, a user's hand size must be multiplied with the DeepFisheyeNet's output.

NETWORK VERIFICATION: ABLATION STUDY

DeepFisheyeNet is a result of our iterative process to improve hand pose estimation accuracy. Our method, that generates a depth image from a color fisheye image and utilizes both images for hand pose estimation is originally introduced in this research. Therefore, we aimed to evaluate the importance of P2DNet by answering the following question: "Does P2DNet improve the 3D joint estimation performance?" To answer this question, we compared two different HPENets, and the synthetic dataset was used for this comparison. The first HPENet was trained with both color fisheye and generated depth images. It was same as the DeepFisheyeNet trained until step 2 of the training. The second HPENet was trained only with fisheye images with the exact same setting as the first.

The percentage of correct keypoints (PCK) of all the joints were calculated and plotted in Figure 9. PCK represents the ratio of correct keypoints for different thresholds. A keypoint is considered to be correct when its error is less than a certain threshold. The result in Figure 9 shows that DeepFisheyeNet outperformed the HPENet for every threshold. Therefore, the answer to the aforementioned question is affirmative–"P2DNet improves the joint estimation performance."

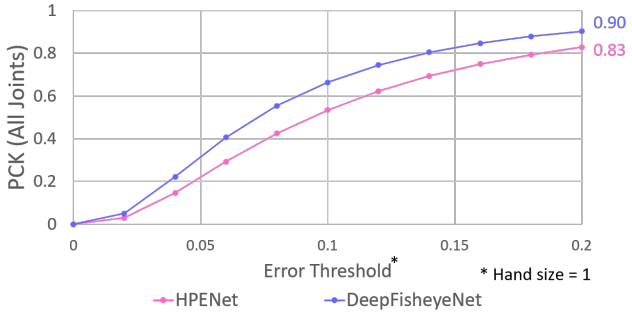

Figure 9. Result of the ablation study. Generated depth images improved the hand pose estimation performance.

INTERACTION ELEMENT CLASSIFIERS

We designed simple rule-based classifiers for contact finger and hand posture classifications to explore DeepFisheye's potential for other interactions. The classifiers use 3D joint output of DeepFisheyeNet as an input.

Contact Finger Classifier

The contact finger classifier classifies which finger among the five has contacted on a touchscreen. It can be used for various finger-aware interfaces [6, 22, 23]. In its simple algorithm, the finger with its fingertip closest to a touch point is selected. The classifier calculates the distances between the fingertips and a touch point in spherical coordinates and applies different weights for each axis because the joint decoder in DeepFisheyeNet separately estimates the direction and distance to a 3D joint location. Therefore, each axis in the spherical coordinate may contribute differently to the accuracy. The classifier calculates the distances between the fingertips and a touch point using Equation 1. Then it selects the finger with the smallest distance. This classifier has two parameters: the weight of θ_{diff} (α) and the weight of r_{diff} (β). The parameters were decided by the pilot test, which are explained in the following section.

$$distance = \phi_{diff} + \alpha \cdot \theta_{diff} + \beta \cdot r_{diff} \quad (1)$$

Hand Posture Classifier

The interaction space would be expanded by using hand posture information as demonstrated in [79, 39]. The hand posture classifier estimates the hand posture when it touches the touchscreen. The classifier discriminates three postures as shown in Figure 10, and classified them as open, pinch, and point postures. The posture classifier is activated only when the touchscreen is tapped with the index finger.

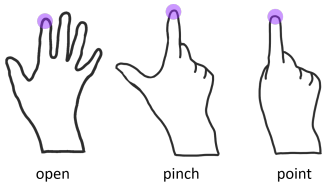

Figure 10. Hand postures. a) Open, b) pinch, and c) point postures.

Hand postures were recognized from the folding states that indicate if a finger is folded or opened. The classifier chooses the posture with the most similar states to the current fingers as the estimated posture. When a finger is opened, its fingertip is more far from the wrist compared to its MCP joint. The classifier uses this geometrical characteristic to identify whether a finger is folded or not. The classifier first normalizes the joint data by dividing it by the hand size. Next, it projects all the joints to the touchscreen plane. Then, it calculates the distance from the wrist to the fingertip and from the wrist to the finger's MCP. If the difference of the two distances is larger than a certain threshold, the finger is considered as opened ($distToTip - distToMCP > threshold$). In case of the thumb, we used the pinky finger's MCP instead of the wrist, because the thumb folds in the horizontal direction of the palm, whereas the other fingers in the vertical direction. Therefore, the classifier has two threshold parameters: first for the thumb and second for the other fingers. The parameters were decided by a pilot test.

Pilot Test

We collected two types of data: contact finger and hand posture data, for optimizing the contact finger and hand posture classifiers, respectively. FisheyedSurface was used for the data collection. The participants in the real dataset collection also participated in this dataset collection.

FisheyedSurface was set on a desk and it was tilted at approximately 40°. There were two sessions, wherein contact finger data were collected in the first. A participant sat on a chair and drew zigzag lines from the left top to the right bottom corner with one finger. The participants were asked to draw lines evenly on the touchscreen. While drawing the lines, the participants kept all the fingers open. The reason for this is while using finger-aware interface, opening the fingers in more preferable than closing the hand [79]. A touch point and an image were recorded for 5 frames-per-second (FPS), and we verified that $500 \sim 700$ samples were collected for every finger of each participant. Participants repeated this process for all the fingers, from the thumb to pinky finger.

Hand posture data were collected in the second session. Only the pinch posture and point posture data were collected, because open posture data were already collected in the first session. The procedure was exactly the same as in the first session, except that the hand postures were changed, but not the touching finger.

By using the collected data, we identified the best parameters for the contact finger and hand posture classifiers. We applied various combinations of the parameters and chose the one with the best accuracy. We estimated 3D joint coordinates for every image by using DeepFisheyeNet that was completely trained, and used the coordinates as input to the classifiers. The search ranges of the parameters were $0 \sim 2$ and $0 \sim 0.02$ for α and β, respectively, in the contact finger classifier. We set the search range of β smaller than α by considering the value ranges of r (in mm) and θ (in radian). The search space of $-1 \sim 1$ was used for the hand posture classifier thresholds.

For the contact finger classifier, ($\alpha = 0.4$, $\beta = 0.004$) showed the best result. For the hand posture parameter, using thresholds of 0.0 for the thumb and 0.6 for the other fingers showed the best result. We used the selected parameters for the user test and our real-time prototype.

USER TEST

We tested DeepFisheye with different people in various environments to evaluate its performance. We measured hover fingertip tracking error, contact finger classifying accuracy, and hand posture classifying accuracy. We collected three types of data for this test. The mid-air hand data were collected for measuring tracking error. This data comprises pairs of real fisheye images and 3D ground truth joint data. The touch hand and hand posture data were collected to evaluate the contact finger and hand posture classification accuracies, respectively. Pairs of real fisheye images and touch points were included for both the types of data.

Data Collection

We recruited 12 participants from an online school community. Their average age and average hand size was 22.8 (STD = 3.6) and 73.9 mm (STD = 4.7 mm), respectively. Four different places: a seminar room, workshop, cafe, and outdoor terrace

were chosen. We selected these places by considering various backgrounds and lighting conditions. Similar to the pilot study, we set open hand posture as a default posture. To this end, we asked the participants to comfortably open all fingers while collecting data, except while collecting hand posture data.

First, we collected mid-air hand data. We used FishLeapFrame that was also used for collecting the real image dataset. A participant sat in front of the frame and moved their right hand in the hand moving space for two minutes. In this situation, fisheye images and ground truth joint data were collected. The participant was asked to move their hand to every part of the hand moving space. Moreover, they were requested to slightly tilt their hand toward the ground so that the hand posture would be similar to that as when using a touchscreen. After the data collection session, we checked if the Leap Motion device properly tracked the hands by projecting the joint data to the real images. Next, we collected touch hand and hand posture data at the same place. The procedure for this was the same as that of the pilot test.

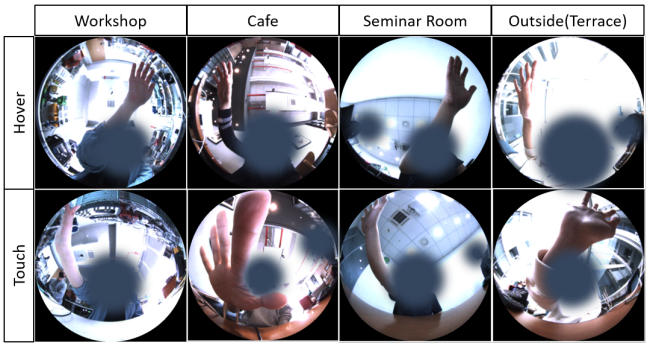

Figure 11. Sample images from the user test for different test locations and hand locations. The test locations had various background and lighting conditions.

Samples of the collected images are presented in Figure 11. More images are in our supplementary document. Images of different places and hand locations are presented in Figure 11. The workshop and cafe had complex backgrounds. In contrast, the seminar room had simple white background. The terrace that was lighted only by the sunlight was brighter than the other places. Some edges of the hand were not visible owing to saturation caused by the sunlight. We would like to emphasize that none of the data collected from the user test were used for training the deep learning model or optimizing the classifier parameters.

Offline Evaluation

One of the motivations for this research is to create a system that can be applied to various device sizes. Therefore, we analyzed the errors for different areas. We considered three sizes: that of a large tablet, small tablet, and smartphone, and the reference devices were Microsoft Surface, iPad Mini, and iPhone 11, respectively. Figure 12 illustrates the screen sizes of the devices relative to FisheyedSurface. We defined the terms "touch area" and "hover space" as follows. Touch area is a subarea of the FisheyedSurface touchscreen with the size of its reference device's touchscreen, as shown in Figure 12. These areas were used for evaluating the contact finger and hand posture classifiers. Only data inside the area of

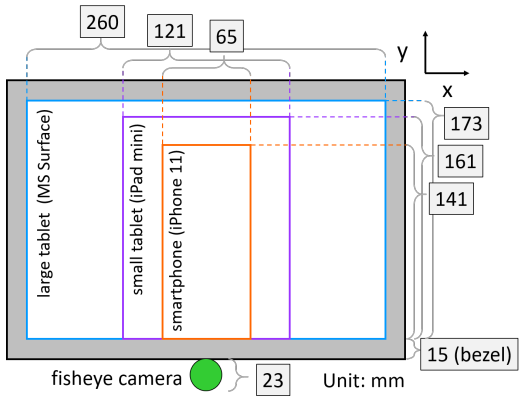

Figure 12. Target device sizes.

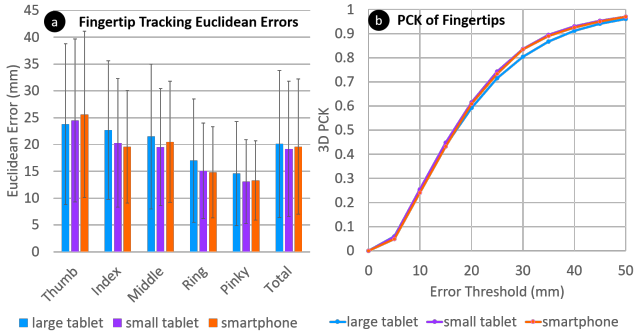

Figure 13. a) Fingertip tracking errors for each finger. Error bar represents standard deviation. b) PCK for different thresholds.

each device size were included for the evaluation. Regarding finger tracking accuracy, we defined the "hover space" for each device size. Hover space is a space above FishLeapFrame with the same area as the touch area with 100 mm height. For the evaluation, we filtered out data that were outside the hover spaces.

We calculated Euclidean distance errors for all the fingertips, and the average errors for each device sizes were: large tablet = 20.1 mm (STD: 13.7 mm), small tablet = 19.2 mm (STD: 12.6 mm), smartphone = 19.6 mm (STD: 12.6 mm), as shown in Figure 13a. The PCKs for each device size are shown in Figure 13b. It can be seen from the figure that the tracking accuracy for the large tablet was lower than that of the others. The spatial distribution of errors is plotted in Figure 14a. The location of a point is the ground truth location of a fingertip. The graph was divided into 5 mm grids, and errors in the same grid were averaged.

The finger classification accuracy distribution on the touchscreen is presented in Figure 14b. The graph is divided into 96×96 px size cells, and each cell shows the average accuracy of the respective area. Figure 14c shows the confusion matrices for each device size. The accuracies for the 5 fingers were 81, 85, and 84% for the sizes of large tablet, small tablet, and smartphone, respectively.

Furthermore, accuracy of the hand posture was evaluated for different touch areas. The distributions of the accuracies are shown in Figure 14d. The confusion matrices for different device sizes are shown in Figure 14e. The accuracies were

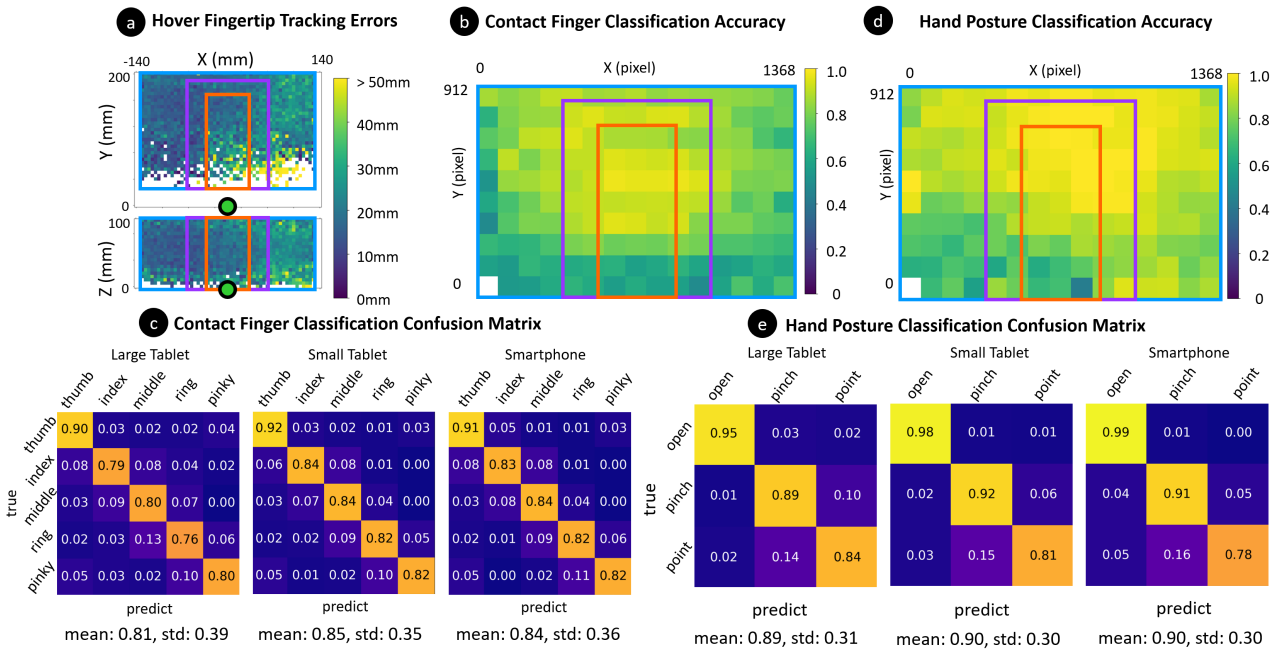

Figure 14. a) Hover fingertip tracking error distribution. b) Contact finger classification accuracy distribution and c) Contact finger classification confusion matrix. d) Hand posture classification accuracy distribution and e) Hand posture classification confusion matrix. There are no samples in the white cells.

89, 90, and 90% for large tablet, small tablet and smartphone, respectively.

Discussion

The overall tracking error was approximately 20 mm, and more than half of the errors were smaller than this (Figure 13b). The near-right side of the camera showed relatively higher error values (Figure 14a), and they belong to the thumb. When the hand was at the right side of the camera, sometimes the thumb was aligned with the other fingers, and then, occasionally edges of the thumb's fingertip were unclear in the image. A sample image illustrating the error case is presented in Figure 15a.

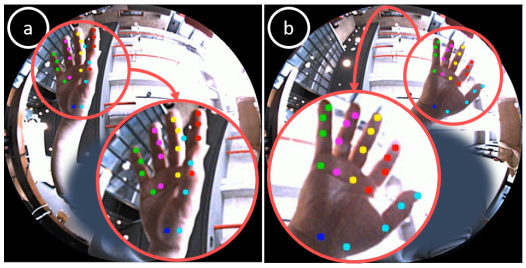

Figure 15. When the thumb was aligned with the other fingers, sometimes a) joints were incorrectly estimated owing to unclear edges of the fingertip. Cyan points represent the estimated thumb joints. b) Example of correctly estimated joints.

We compared our tracking performance to one of the state-of-the-art results. Mueller et al. [48] used flat color and flat depth images for hand pose estimation. Their model showed errors with approximated value of 20 mm across the fingertips. This comparison may be inappropriate, because the datasets are different and they used real depth images. However, this comparison shows that DeepFisheye's fingertip tracking performance is not overly inferior than that of existing solutions that work with flat images.

Finger classification accuracies across the device sizes were approximately 83%. The thumb showed the best result because it was far from the other fingers. The graph in Figure 14b shows that the performance was not good at the left and right sides of the touchscreen. When the hand moved to the sides, the fingers were occluded and that could have led to an error. Additionally, error values were also high near the camera. One of the possible reason could be that the images were dark when the hand was at a close distance, because a large part of the camera lens was covered by the hand. Another reason could be that the hand images were highly distorted when the hand was at a close distance.

It is difficult to directly compare the performance of our finger classification with that of previous studies because the sizes of the devices and target fingers were different. Nonetheless, we compared our results with state-of-the-art finger classification results to gain insight into our results. HandSee [75] could classify 3 classes with 98 and 89.7% of accuracies for 5 classes. Its differences with DeepFisheye are that it was only tested with a smartphone and it grouped the middle, ring, and pinky fingers as a single class. InfiniTouch [32] could classify five fingers at an accuracy of 96%, however, most of the fingers had to be contacted with the capacitive sensor.

The hand posture accuracy was approximately 90% across the different device sizes. The pinch and point postures were the most confusing ones. Errors in them mainly occurred at the bottom left of the touchscreen. In this area, thumbs can be easily occluded by the hand. Furthermore, similar

to finger classification, DeepFisheye's results became less accurate when the hand was very close. We compared our hand posture classification accuracy with that of other related work. Zheng et al. [79] classified three postures, including our the "open" and the "point" postures near the keyboard. Their classification result (89.1%) was comparable to ours. In contrast to DeepFisheye, they used a green background to easily segment a hand and used a camera at a distance from the keyboard.

EXAMPLE SCENARIOS WITH INTERACTION ELEMENTS

We implemented a working prototype with FisheyedSurface to demonstrate how DeepFisheye can be used for various interaction scenarios. We created a painting application that utilizes the locations of fingertips, contact finger information, and hand postures. The stream of fisheye images could be processed in real-time (18 FPS) on a GeForce GTX 1060 GPU.

Finger-aware functions are possible by using the results of contact finger estimation. A user can paint with her index finger and erase with her middle finger. Furthermore, a function preview was implemented for the multi–functional button on the top left to help inexperienced users who are unaware of the function–finger mappings (Figure 16a). When a finger approaches the function button at sufficiently close distance, the corresponding function preview appears on the button. This preview functionality was made feasible by the finger tracking. The multi–functional button reduces the need for multiple buttons on the touchscreen so that a user can use a larger canvas. Additionally, the interaction vocabulary was expanded by utilizing hand postures. For instance, dragging mode can vary according to the hand posture. The user can move a selected object with a pinch posture and copy the object with a point posture as shown in Figures 16b and c.

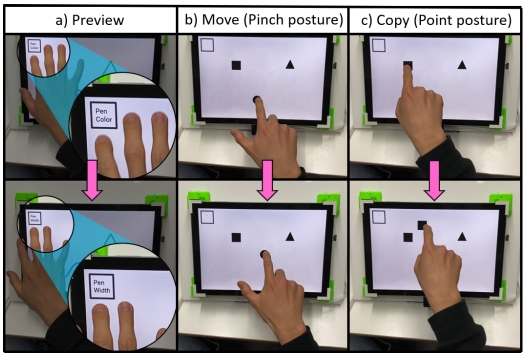

Figure 16. a) Mapped functions appear when a finger approaches the button. The user can b) move or c) copy shapes through different postures. Additional scenarios are presented in our supplementary video.

LIMITATIONS AND FUTURE WORK

We observed that the accuracies of the contact finger classification reduced at the sides of the large tablet area. In those regions, parts of the hand in the images were small and the fingers were easily occluded. Minute details of the hands are important in this case, and using images with higher resolution can improve the performance. Similarly, Xu et al. [71] used a fisheye camera to estimate the body pose: they used two images of a downsampled image capturing the whole body and a cropped image showing details of the body parts far from the camera. Our future work would be to use higher resolution images to increase tracking accuracy, and it can be accomplished without sacrificing processing speed because there is scope for optimizing the network.

The contact finger and hand posture classifiers showed relatively low accuracy near the camera. This problem may be related to the datasets. We collected both synthetic and real datasets by locating the hands evenly in the Cartesian space. However, hand images distort a lot when the hand is near the camera than far. Therefore, more diverse hand images can exist when the hand is near the camera. In our future work, we plan to create a dataset comprising more near-camera data to solve this problem.

In this study, we used the devices in a specific orientation (e.g., portrait or landscape). If the user changes the device orientation, the current DeepFisheye may fail to track the fingers because it was not trained with the hand images in the new orientation. A straightforward solution to this could be to use a separate camera for each orientation, however, it may not be cost-effective. A more desirable solution may be to augment the network with datasets covering both device orientations. The modified DeepFisheyeNet should be able to track the fingers in both device orientations. This is a new challenge deserving future research, especially because hand images in diverse orientations are expected to contain more self-occlusion issues. We expect that such occlusion issues can be overcome based on the fact that Mueller et al. [46] was able to handle similar occlusion issues by intentionally including occluded parts of a hand in their dataset.

We used Leap Motion data as ground truth data in the hover tracking test. Therefore, the test results may have been affected by the errors in the Leap Motion data. For a more accurate evaluation, we require more accurate ground truth data. The possible options that we will consider in our future work include using a multi-camera system [82] and a manual annotation method [48].

CONCLUSION

We proposed DeepFisheye, a NMFT system that expands the input space of a touchscreen. In addition to finger tracking, DeepFisheye can classify contact fingers, and recognize hand postures. DeepFisheye has a practical form factor for mobile devices because it comprises a smart device and a fisheye camera attached to that device. We evaluated DeepFisheye's performance for three different device sizes. DeepFisheye was able to track 5 fingertips of an open hand with errors of approximate value of 20 mm in various device sizes. Furthermore, it could classify contact fingers and hand postures with approximate accuracy of 83 and 90%, respectively, across the various touchscreen sizes. Despite the remaining challenges, DeepFisheye demonstrated a novel way of tracking fingers near a touchscreen within a practical form factor.

ACKNOWLEDGMENTS

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT)(No.2020-0-00537, Development of

5G based low latency device – edge cloud interaction technology)

REFERENCES

[1] 2016. Spatio-Temporal Hough Forest for Efficient Detection-Localisation-Recognition of Fingerwriting in Egocentric Camera. *Comput. Vis. Image Underst.* 148, C (July 2016), 87–96.

[2] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. 2018. Augmented Skeleton Space Transfer for Depth-Based Hand Pose Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. 2019. Pushing the Envelope for RGB-Based Dense 3D Hand Pose Estimation via Neural Rendering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. 2012. ShoeSense: A New Perspective on Gestural Interaction and Wearable Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1239–1248. DOI: http://dx.doi.org/10.1145/2207676.2208576

[5] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. 2015. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In *The IEEE International Conference on Computer Vision (ICCV)*.

[6] Hrvoje Benko, T. Scott Saponas, Dan Morris, and Desney Tan. 2009. Enhancing Input on and above the Interactive Surface with Muscle Sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*. Association for Computing Machinery, New York, NY, USA, 93–100. DOI:http://dx.doi.org/10.1145/1731903.1731924

[7] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. 2018. Weakly-supervised 3D Hand Pose Estimation from Monocular RGB Images. In *The European Conference on Computer Vision (ECCV)*.

[8] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 2015. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology (UIST '15)*. Association for Computing Machinery, New York, NY, USA, 549–556. DOI: http://dx.doi.org/10.1145/2807442.2807450

[9] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013a. UTrack: 3D Input Using Two Magnetic Sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. Association for Computing Machinery, New York, NY, USA, 237–244. DOI: http://dx.doi.org/10.1145/2501988.2502035

[10] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013b. UTrack: 3D Input Using Two Magnetic Sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. Association for Computing Machinery, New York, NY, USA, 237–244. DOI: http://dx.doi.org/10.1145/2501988.2502035

[11] Ke-Yu Chen, Shwetak N. Patel, and Sean Keller. 2016. Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1504–1514. DOI: http://dx.doi.org/10.1145/2858036.2858125

[12] X. Chen, G. Wang, C. Zhang, T. Kim, and X. Ji. 2018. SHPR-Net: Deep Semantic Hand Pose Regression From Point Clouds. *IEEE Access* 6 (2018), 43425–43439.

[13] Xiang "Anthony" Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. 2014. Air+touch: Interweaving Touch in-Air Gestures. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, NY, USA, 519–525. DOI: http://dx.doi.org/10.1145/2642918.2647392

[14] Ashley Colley and Jonna Häkkilä. 2014. Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. Association for Computing Machinery, New York, NY, USA, 539–548. DOI: http://dx.doi.org/10.1145/2686612.2686699

[15] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. 2018. SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images. In *The European Conference on Computer Vision (ECCV)*.

[16] 5DT DataGlove. 2020. 5DT Data Glove. Web. (1 May 2020). Retrieved May 1, 2020 from https://5dt.com/5dt-data-glove-ultra/.

[17] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying Finger Touches on Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3879–3890. DOI: http://dx.doi.org/10.1145/3025453.3025561

[19] Oliver Glauser, Shihao Wu, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. 2019. Interactive Hand Pose Estimation Using a Stretch-Sensing Soft Glove. *ACM Trans. Graph.* 38, 4, Article Article 41 (July 2019), 15 pages. DOI: http://dx.doi.org/10.1145/3306346.3322957

[20] Alix Goguey, Géry Casiez, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak, and Nicolas Roussel. 2014. A Three-Step Interaction Pattern for Improving Discoverability in Finger Identification Techniques. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 33–34. DOI:http://dx.doi.org/10.1145/2658779.2659100

[21] Alix Goguey, Mathieu Nancel, Géry Casiez, and Daniel Vogel. 2016. The Performance and Preference of Different Fingers and Chords for Pointing, Dragging, and Object Transformation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4250–4261. DOI: http://dx.doi.org/10.1145/2858036.2858194

[22] Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016. Porous Interfaces for Small Screen Multitasking Using Finger Identification. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 145–156. DOI:http://dx.doi.org/10.1145/2984511.2984557

[23] Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 59–70. DOI:http://dx.doi.org/10.1145/2858036.2858052

[24] Chris Harrison and Scott E. Hudson. 2009. Abracadabra: Wireless, High-Precision, and Unpowered Finger Input for Very Small Mobile Devices. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. Association for Computing Machinery, New York, NY, USA, 121–124. DOI:http://dx.doi.org/10.1145/1622176.1622199

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. 2016. Pre-Touch Sensing for Mobile Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 2869–2881. DOI: http://dx.doi.org/10.1145/2858036.2858095

[27] Christian Holz and Patrick Baudisch. 2013. Fiberio: A Touchscreen That Senses Fingerprints. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. Association for Computing Machinery, New York, NY, USA, 41–50. DOI:http://dx.doi.org/10.1145/2501988.2502021

[28] Y. Jang, I. Jeon, T. Kim, and W. Woo. 2017. Metaphoric Hand Gestures for Orientation-Aware VR Object Manipulation With an Egocentric Viewpoint. *IEEE Transactions on Human-Machine Systems* 47, 1 (2017), 113–127.

[29] Insu Kim, Keunwoo Park, Youngwoo Yoon, and Geehyuk Lee. 2018. Touch180: Finger Identification on Mobile Touchscreen Using Fisheye Camera and Convolutional Neural Network. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings (UIST '18 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 29–32. DOI:http://dx.doi.org/10.1145/3266037.3266091

[30] J. Kim, N. D. Thang, and T. Kim. 2009. 3-D hand motion tracking and gesture recognition using a data glove. In *2009 IEEE International Symposium on Industrial Electronics*. 1013–1018.

[31] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[32] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 779–792. DOI: http://dx.doi.org/10.1145/3242587.3242605

[33] Mathieu Le Goc, Stuart Taylor, Shahram Izadi, and Cem Keskin. 2014. A Low-Cost Transparent Electric Field Sensor for 3d Interaction on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 3167–3170. DOI: http://dx.doi.org/10.1145/2556288.2557331

[34] Kuo-Wei Lee, Shih-Hung Liu, Hwann-Tzong Chen, and Koichi Ito. 2019. Silhouette-Net: 3D Hand Pose Estimation from Silhouettes. (2019).

[35] Y. Lee, J. Jeong, J. Yun, W. Cho, and K. Yoon. 2019. SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360° Images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9173–9181.

[36] S. Li. 2006. Monitoring Around a Vehicle by a Spherical Image Sensor. *IEEE Transactions on Intelligent Transportation Systems* 7, 4 (2006), 541–550.

[37] Xuan Li, Chun-Ho Chen, and Yi-Pai Huang. 2016. 3D interactive system based on vision computing of direct-flective cameras. *Journal of the Society for Information Display* 24, 8 (2016), 521–528. DOI: http://dx.doi.org/10.1002/jsid.457

[38] Jaime Lien, Nicholas Gillian, M. Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous Gesture Sensing with Millimeter Wave Radar. *ACM Trans. Graph.* 35, 4, Article Article 142 (July 2016), 19 pages. DOI:http://dx.doi.org/10.1145/2897824.2925953

[39] Hyunchul Lim, Jungmin Chung, Changhoon Oh, SoHyun Park, Joonhwan Lee, and Bongwon Suh. 2018. Touch+Finger: Extending Touch-Based User Interface Capabilities with "Idle" Finger Gestures in the Air. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. Association for Computing Machinery, New York, NY, USA, 335–346. DOI: http://dx.doi.org/10.1145/3242587.3242651

[40] W. Lu, Z. Tong, and J. Chu. 2016. Dynamic Hand Gesture Recognition With Leap Motion Controller. *IEEE Signal Processing Letters* 23, 9 (2016), 1188–1192.

[41] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker. 2018. DeepHPS: End-to-end Estimation of 3D Hand Pose and Shape by Learning from Synthetic Depth. In *2018 International Conference on 3D Vision (3DV)*. 110–119.

[42] ManusVR. 2020. ManusVR Prime One. Web. (1 May 2020). Retrieved May 1, 2020 from https://manus-vr.com/prime-one-gloves/.

[43] Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. 2017. WhichFingers: Identifying Fingers on Touch Surfaces and Keyboards Using Vibration Sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. Association for Computing Machinery, New York, NY, USA, 41–48. DOI: http://dx.doi.org/10.1145/3126594.3126619

[44] Jess McIntosh, Paul Strohmeier, Jarrod Knibbe, Sebastian Boring, and Kasper Hornbæk. 2019. Magnetips: Combining Fingertip Tracking and Haptic Feedback for Around-Device Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article Paper 408, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300638

[45] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. 2018. V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation From a Single Depth Map. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[46] 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. https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/

[47] 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).

[48] 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 *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

[49] Sundar Murugappan, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. Association for Computing Machinery, New York, NY, USA, 487–496. DOI: http://dx.doi.org/10.1145/2380116.2380177

[50] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1515–1525. DOI: http://dx.doi.org/10.1145/2858036.2858580

[51] Vassilis C. Nicodemou, Iason Oikonomidis, Georgios Tzimiropoulos, and Antonis Argyros. 2018. Learning to Infer the Depth Map of a Hand from its Color Image. (2018).

[52] M. Oberweger and V. Lepetit. 2017. DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 585–594.

[53] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. 2015. Hands Deep in Deep Learning for Hand Pose Estimation. *CoRR* abs/1502.06807 (2015). http://arxiv.org/abs/1502.06807

[54] Paschalis Panteleris and Antonis Argyros. 2017. Back to RGB: 3D Tracking of Hands and Hand-Object Interactions Based on Short-Baseline Stereo. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

[55] P. Panteleris, I. Oikonomidis, and A. Argyros. 2018. Using a Single RGB Frame for Real Time 3D Hand Pose Estimation in the Wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 436–445.

[56] Keunwoo Park, Daehwa Kim, Seongkook Heo, and Geehyuk Lee. 2020. MagTouch: Robust Finger Identification for a Smartwatch Using a Magnet Ring and a Built-in Magnetometer. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. DOI: http://dx.doi.org/10.1145/3313831.3376234

[57] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. 2013. The Leap Motion Controller: A View on Sign Language. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration (OzCHI '13)*. Association for Computing Machinery, New York, NY, USA, 175–178. DOI:http://dx.doi.org/10.1145/2541016.2541072

[58] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and Robust Hand Tracking from Depth. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[59] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. 2011. Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distance with a Commodity Depth Camera. In *Proceedings of the 19th ACM International Conference on Multimedia (MM '11)*. Association for Computing Machinery, New York, NY, USA, 1093–1096. DOI: http://dx.doi.org/10.1145/2072298.2071946

[60] D. Scaramuzza, A. Martinelli, and R. Siegwart. 2006. A Toolbox for Easily Calibrating Omnidirectional Cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5695–5701.

[61] Srinath Sridhar, Anders Markussen, Antti Oulasvirta, Christian Theobalt, and Sebastian Boring. 2017. WatchSense: On- and Above-Skin Input Sensing through a Wearable Depth Sensor. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3891–3902. DOI: http://dx.doi.org/10.1145/3025453.3026005

[62] Yu-Chuan Su and Kristen Grauman. 2019. Kernel Transformer Networks for Compact Spherical Convolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

[64] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. 2014. Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks. *ACM Trans. Graph.* 33, 5, Article Article 169 (Sept. 2014), 10 pages. DOI:http://dx.doi.org/10.1145/2629500

[65] James Y Tung, Tea Lulic, Dave A Gonzalez, Johnathan Tran, Clark R Dickerson, and Eric A Roy. 2015. Evaluation of a portable markerless finger position capture device: accuracy of the Leap Motion controller in healthy adults. *Physiological Measurement* 36, 5 (apr 2015), 1025–1035. DOI: http://dx.doi.org/10.1088/0967-3334/36/5/1025

[66] Pier Paolo Valentini and Eugenio Pezzuti. 2017. Accuracy in fingertip tracking using Leap Motion Controller for interactive virtual applications. *International Journal on Interactive Design and Manufacturing (IJIDeM)* 11, 3 (2017), 641–650.

[67] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. 2019. Self-Supervised 3D Hand Pose Estimation Through Training by Fitting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[68] Jingtao Wang and John Canny. 2004. FingerSense: Augmenting Expressiveness to Physical Pushing Button by Fingertip Identification. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. Association for Computing Machinery, New York, NY, USA, 1267–1270. DOI: http://dx.doi.org/10.1145/985921.986040

[69] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. 2016. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 851–860. DOI: http://dx.doi.org/10.1145/2984511.2984565

[70] Wenbin Wu, Chenyang Li, Zhuo Cheng, Xin Zhang, and Lianwen Jin. 2017. YOLSE: Egocentric Fingertip Detection From Single RGB Images. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

[71] W. Xu, A. Chatterjee, M. Zollhöfer, H. Rhodin, P. Fua, H. Seidel, and C. Theobalt. 2019. Mo2Cap2: Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera. *IEEE Transactions on Visualization and Computer Graphics* 25, 5 (2019), 2093–2101.

[72] Linlin Yang and Angela Yao. 2019. Disentangling Latent Hands for Image Synthesis and Pose Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[73] Xing-Dong Yang, Khalad Hasan, Neil Bruce, and Pourang Irani. 2013. Surround-See: Enabling Peripheral Vision on Smartphones during Active Use. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. Association for Computing Machinery, New York, NY, USA, 291–300. DOI: http://dx.doi.org/10.1145/2501988.2502049

[74] Qi Ye and Tae-Kyun Kim. 2018. Occlusion-Aware Hand Pose Estimation Using Hierarchical Mixture Density Network. In *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer International Publishing, Cham, 817–834.

[75] Chun Yu, Xiaoying Wei, Shubh Vachher, Yue Qin, Chen Liang, Yueting Weng, Yizheng Gu, and Yuanchun Shi. 2019. HandSee: Enabling Full Hand Interaction on Smartphone with Front Camera-Based Stereo Vision. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article Paper 705, 13 pages. DOI: http://dx.doi.org/10.1145/3290605.3300935

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

[77] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-Grained Acoustic-Based Device-Free Tracking. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. Association for Computing Machinery, New York, NY, USA, 15–28. DOI: http://dx.doi.org/10.1145/3081333.3081356

[78] X. Zhang, Q. Li, H. Mo, W. Zhang, and W. Zheng. 2019. End-to-End Hand Mesh Recovery From a Monocular RGB Image. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2354–2364.

[79] Jingjie Zheng and Daniel Vogel. 2016. Finger-Aware Shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4274–4285. DOI: http://dx.doi.org/10.1145/2858036.2858355

[80] Yidan Zhou, Jian Lu, Kuo Du, Xiangbo Lin, Yi Sun, and Xiaohong Ma. 2018. HBE: Hand Branch Ensemble Network for Real-time 3D Hand Pose Estimation. In *The European Conference on Computer Vision (ECCV)*.

[81] Christian Zimmermann and Thomas Brox. 2017. Learning to Estimate 3D Hand Pose From Single RGB Images. In *The IEEE International Conference on Computer Vision (ICCV)*.

[82] 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. In *The IEEE International Conference on Computer Vision (ICCV)*.