Evaluation of Machine Learning Techniques for Hand Pose Estimation on Handheld Device with Proximity Sensor

Kazuyuki Arimatsu
Sony Interactive Entertainment Inc.
Tokyo, Japan
Kazuyuki.Arimatsu@sony.com

Hideki Mori
Sony Interactive Entertainment Inc.
Tokyo, Japan
Hideki.00.Mori@sony.com

ABSTRACT

Tracking finger movement for natural interaction using hand is commonly studied. For vision-based implementations of finger tracking in virtual reality (VR) application, finger movement is occluded by a handheld device which is necessary for auxiliary input, thus tracking finger movement using cameras is still challenging. Finger tracking controllers using capacitive proximity sensors on the surface are starting to appear. However, research on estimating articulated hand pose from curved capacitance sensing electrodes is still immature. Therefore, we built a prototype with 62 electrodes and recorded training datasets using an optical tracking system. We have introduced 2.5D representation to apply convolutional neural network methods on a capacitive image of the curved surface, and two types of network architectures based on recent achievements in the computer vision field were evaluated with our dataset. We also implemented real-time interactive applications using the prototype and demonstrated the possibility of intuitive interaction using fingers in VR applications.

Author Keywords

Hand pose estimation; finger tracking controller; capacitive image; human computer interactiton; virtual reality

CCS Concepts

•**Human-centered computing** → **Interaction devices;**
•**Computing methodologies** → **Neural networks;**

INTRODUCTION

Hand tracking plays an essential role in an immersive experience in virtual reality (VR). Some VR dedicated controllers such as Facebook's Oculus Touch [23] successfully accomplished natural interaction using hands in VR. Although the controller can precisely track hand position, their implementation for sensing finger movement is currently limited, since they provide triggers only for index and middle fingers, and the controller does not reflect all finger movement in the virtual world. Reconstructing natural five finger movements in a precise manner gains the presence of our hand in VR space, which enables intuitive interaction in VR.

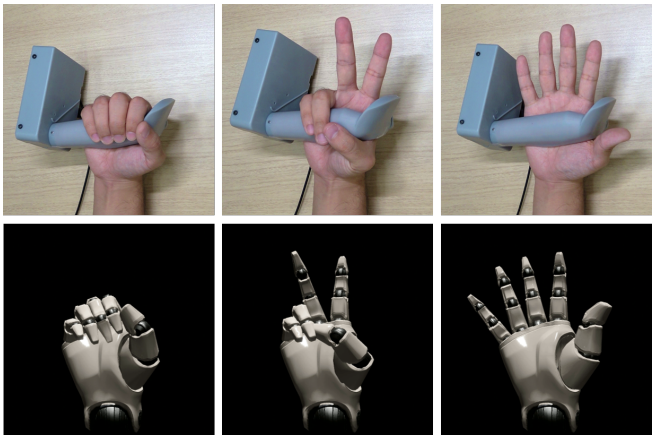

Figure 1. Hand pose estimation using capacitive proximity sensors on a handheld device. The top row displays actual hand poses holding the device, while the bottom row displays estimated poses of the corresponding hand. All estimation is performed only using sensors on the device in real-time, and no external vision sensor is used for estimation.

Five finger tracking controllers are at the consumer's fingertip as commercial products such as Valve's Index Controller [11]. However, its finger tracking is also limited in the dexterity of the estimated hand pose. Tactual Labs demonstrates a novel controller that tracks in-air finger movement with near-range radio-frequency sensing technology [31]. However, their available materials do not exhibit its capability of tracking fully articulated thumb movements, including bending inward or outward. Zhang *et al.* proposed a method to track finger movements while holding a controller using a camera mounted in front of it [58], but its application is limited because of the weight of the over-hanging camera.

Although researchers suggest interactions using bare hand for usage in augmented reality (AR) applications, VR applications still require auxiliary input such as thumbsticks, buttons, and triggers. In contrast to AR, experiences in VR tend to be accomplished in a smaller space compared to the scale of the virtual world. Thus users must use thumbsticks or another auxiliary input method for moving around the virtual world. In addition, game developers still strongly demand physical input buttons and triggers for concrete game design, which requires severe timing constraints on input and character locomotion implementation. Also, regarding haptic feedback,

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
DOI: https://dx.doi.org/10.1145/3313831.3376712

the dominant implementation of current consumer controllers for haptic feedback is vibrotactile actuators installed in a controller, and much research is reported recently to seek the eventual haptic feedback method for handheld controllers [7, 33, 5]. Therefore, precise finger tracking while a controller is in hand is desired for enabling intuitive interaction with complementary physical input and tactile feedback.

On the other hand, recent great achievements in computer vision (CV) based algorithm bring precise visual finger tracking to consumer-level products such as Leap Motion [25] and Oculus Quest [24]. However, while a user has a controller in hand, the controller interferes with line-of-sight from cameras to fingers and prevents detecting finger movement using cameras. According to the evaluation of 10 state-of-the-art (SOTA) CV based methods in *Hands In the Million Challenge* (HIM2017) [56], estimation error increased depending on the decrease of the number of visible joints, and the report remarked that estimation of occluded joints is still challenging for most methods. They also studied the relationship between tracking accuracy and the angle of palm and camera direction. According to their analysis, while mean error remained below 10 mm in 70 to 120 angle range, the error increased to 10–20 mm in other angle range, and the error in 0 to 10 angle range is more significant than 20 mm due to severe self-occlusion. Thus, tracking finger movement using the camera under occlusion or tracking from an undesirable viewpoint still has some difficulties.

Occlusion-free precise finger tracking is made possible using gloves [43, 14] or other wearables [55, 34, 30, 4, 6, 28, 29], which have made a great contribution to the research of natural interaction techniques in HCI. Glauser *et al.* proposed a method of fabrication of stretchable capacitive sensor arrays, a software pipeline to capture deformation [13], and also another novel method to capture hand poses from stretchable glove using the sensor [14]. However, glove vendors must take care of hand size variation, since oversized or undersized gloves cause problematic estimation results, and this causes an increase in production cost for multiple sizes of gloves. Yeo *et al.*, Lin *et al.*, and Kuno *et al.* reported a method to detect finger movement by observing skin deformation of backhand using a camera [55], strain gauge [34], and optical sensor [30]. However, their results had a large estimation error of thumb movement and difficulties of generalization to unseen hands. Chen *et al.* put coils on each fingertip for tracking finger movement using a magnetic field [6], Iravantchi *et al.* performed ultrasonic beamforming with a device on the wrist for gesture recognition [28], and Kim *et al.* tracks fingers using IR camera mounted on the wrist [29]. Their methods estimate finger movements in mobile setup without requiring any external sensors. However, these types of sensing technology require users to put additional sensors on fingers or wrists, which is undesirable from the perspective of ease of use. Wrist-worn wearables also have difficulties in modeling wrist bend and rotations, and could not track fully flat or over-arching hands.

Consequently, we propose a pipeline that estimates positions of each finger joint, only using data from a matrix of capacitive proximity sensors on the controller. We aim to push the boundary of finger tracking capability for more intuitive interaction using hands in virtual space. Our approach not only detects touching of fingers on a specific electrode but predicts comprehensive finger movement in 3D space utilizing values from all sensors. To achieve that, we evaluated two types of convolutional neural network (CNN) architectures studied in the CV field for pose estimation, and illustrated the suitable architecture for the sensors on the controller.

Contribution of this paper consists of the following items:
1. Built handheld finger tracking prototype with a matrix of capacitive proximity sensors installed;
2. Recorded training dataset of 12 subjects using an optical tracking system;
3. Adopted two types of CV based hand pose estimation network architectures into the dataset;
4. Evaluated the performance of them and revealed suitable architecture for finger tracking using capacitive proximity sensors.

RELATED WORKS

Capacitive sensing technology is one of the most commonly used methods for sensing finger movement since touchscreens became a widespread input device [17]. Almost all touch screen devices currently only support the detection of touching points on the surface, but recent research report estimating the orientation of a finger for further user interaction. Xiao *et al.* estimated yaw and pitch angles of touching finger, using extracted manually designed feature [54], while Mayer *et al.* leveraged CNN to estimate yaw and pitch angles [35]. For estimating finger orientation, they utilized **capacitive image**, which represents the difference of electrical capacitance of each electrode between a currently measured value and the baseline value, that is captured while a hand is not touching on the screen. The dominant value that appears in the capacitive image is a change of capacitance when fingers touch on the screen, but the capacitive image also includes a change of capacitance related to finger movement away from the screen surface that can be used for proximity sensing.

In addition, we can wrap the controller surface with sensing electrodes on a flexible printed circuit board (PCB) for sensing finger movement. PrintSense [16] proposed printed flexible capacitance sensing electrodes wrapped on the curved surface. They wrapped an object with the printed electrodes, and performed grasp detection using proximity sensing from electrodes on the surface.

However, research on estimating articulated hand pose from curved electrodes on a handheld device is still immature. Le *et al.* estimated the 3D position of five fingers using the capacitive image [32], which is captured from sensing electrodes on the whole device surface. Although they recorded 3D positions of all joints of a hand using the OptiTrack motion capturing system [26], their implementation estimated only fingertip position touching on the screen.

The difficulty is in the value of a capacitive image. Each pixel of the capacitive image is correlated to the distance between a

finger and electrode, but we cannot merely convert measured value to distance if there is more than one finger near the electrodes. Because the measured capacitance value from the electrode is defined by the sum of the capacitance of all finger tissue around the electrode, finger movement away from electrodes affects multiple electrodes. We need an algorithm for estimating comprehensive hand pose employing values from all electrodes.

On the other hand, recent great achievements on CV based algorithm using CNNs reported several approaches of holistic hand pose estimation from images.

Hand Pose Estimation from Image

Pose estimation networks from 2D images are categorized into direct regression of keypoints in 2D images [52, 3, 44] and heatmap detection from 2D images [50, 53, 37]. While direct regression models output coordinates of the keypoints directly, heatmap detection models output 2D probability maps of each keypoint. Heatmap detection models perform pixel-wise estimation resulting in better accuracy compared to direct regression approaches in 2D images, and are dominant in human pose estimation [12]. However, while applying heatmap detection in 3D position estimation, 3D volumetric heatmap requires a too high computational cost to perform inference in real-time. So, the direct regression approach was standard in real-time 3D pose estimation, even though heatmap detection models [36, 20] outperformed other models in the HIM2017 competition [56].

For achieving the generalized 3D pose estimation from wild images beyond the limitation of the current 3D pose dataset, which is recorded only in a laboratory environment with expensive tracking systems, recent approaches utilize the wild 2D pose image dataset for training 3D pose estimation [59, 44]. They separate 3D pose estimation problem into two parts; 2D joint position (x, y) estimation and depth (z) estimation, which is called 2.5D representation. Iqbal *et al.* combined this 2.5D representation and integral regression method for achieving efficient 3D hand pose estimation from a single RGB image, which they called latent 2.5D heatmap regression [27]. The proposed network has latent heatmaps and depth maps for each keypoint. The heatmaps represent 2D joint positions in sub-pixel accuracy, and the depth maps represent its 2.5D distances. The idea of 2.5D representation enables using heatmap detection for 3D pose estimation in real-time, and it performs better than direct regression for hand pose estimation from an image.

In this paper, we addressed the issues of real-time hand pose estimation using capacitive proximity sensors, utilizing these two types of pose estimation network architectures; **direct regression** and **heatmap detection**. We introduced a method for applying the 2.5D representation to the measured capacitance from electrodes on the curved surface of the controller, where their positions are defined in non-orthogonal coordinate, and evaluated the two types of networks. Due to the small number of electrodes, our results of applying these architectures to capacitive images were different from the previous comparison results using images. We evaluated them from multiple aspects and illustrated the suitable network architecture for capacitive images.

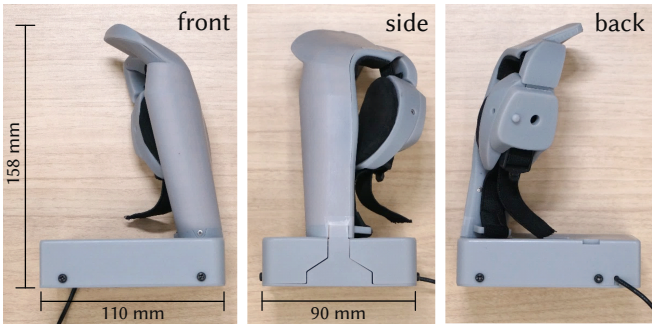

Figure 2. The dimension of the prototype. The appearance of our prototype from its front (*Left*), side (*Middle*), and back (*Right*). Users insert their hand from the side and tighten the belt for fixing the position.

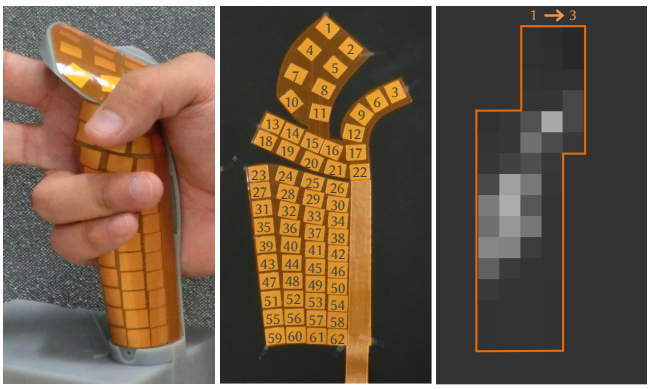

Figure 3. Capacitive proximity sensor electrode sheet inside the prototype. The prototype has 62 electrodes implemented on a flexible printed circuit board (PCB) underneath a cover (*Left*). Each square part is a sensing electrode on the unrolled flexible PCB sheet (*Middle*). Measured values are packed into a 2D capacitive image (*Right*) and fed into a neural network.

THE HARDWARE SETUP AND CAPACITANCE SENSING

We designed the shape of the prototype so that users can grip it naturally while minimizing the interference of the shape with finger movement. The size of the prototype is 158 mm $\times$ 110 mm $\times$ 90 mm, and its weight is 360 g. The grip part of the prototype is 30 mm to 40 mm in diameter and 110 mm in length (Figure 2; left). We decided these dimensions based on the measurement published by Tilley *et al.* [48] aiming for supporting 1 to 99 percentile of adults in the United States. We also keep a flat surface on the top of the device so that auxiliary input can be installed, and prepared a screw hole and a mounting pin on the backside fixture for an external tracker that estimates the global position of the hand. All of the gray parts of the prototype were 3D printed with 3D Systems Projet 6000 printer using VisiJet SL Tough material [9]. An analog front-end ASIC and USB interface IC on the sensor circuit board is installed in the bottom box part. The acquired sensor data is quantized to 16 bit values and transmitted to a PC in 60 Hz via USB serial protocol.

Underneath the 1.2 mm thickness cover, the prototype has a flexible printed circuit board (PCB) sheet wrapping the

curved surface of the grip part and the top surface (Figure 3; left). We implemented 62 capacitance sensing electrodes, which covers both areas of the controller surface as wide as possible, on the flexible PCB sheet (Figure 3; middle), while the size of all electrodes is 10 mm x 8 mm, and driven at 87.5 kHz. Referring to Grosse-Puppendahl's taxonomy [17], our sensing method is *active* capacitive sensing in *loading mode*, which is known as self-capacitance sensing. The sheet is a two-layer copper flexible PCB of 25 µm thickness polyimide, which is fabricated in a standard process. Sensing electrode patterns are on the front side, while shield patterns and wiring patterns are on the backside.

We utilized capacitive image representation [54, 35] for latter processing. Measured electrical capacitance c_i from the i^{th} electrode is subtracted by c_i^{base} and clipped from 0.0 to 1.0. The scaled value of each electrode is ordered to pixel p_{uv} in 18×10 resolution image (Figure 3; right) for feeding it into the network, where white pixels of the image represent electrodes that detect high electrical capacitance. Each value from electrode c_i is tiled in row-major order based on electrode index i (shown in Figure 3; middle) into the area marked by the line (Figure 3; right) as follows:

$$p_{uv} = \frac{(c_i - c_i^{base}) - scale_{min}}{scale_{max} - scale_{min}}, \quad (1)$$

where $scale_{min}$ and $scale_{max}$ parameters are pre-defined values for scaling measured values, and the same scale parameters were used for all datasets and demonstrations. These scale parameters depend on hardware implementation, and the values are selected appropriately for avoiding saturation. We took baseline value c_i^{base} while a hand is opened before each recording of the dataset. We asked subjects to keep their hand opened for a while before the real-time demonstration and measured this baseline value before each demonstration. The values of pixels in the capacitive image that have no corresponding electrodes were also set to c_i^{base}.

INTRODUCING 2.5D REPRESENTATION AND JOINT POSITION ESTIMATION IN UV SPACE

Thinking about applying recent CV achievements into joint position estimation, we can assume an analogy in pose estimation from depth images and from capacitive images. In 3D pose estimation from a depth image, each pixel in the depth image represents the distance between the camera and each joint. Similarly, in 3D pose estimation from a capacitive image, each pixel in the capacitive image represents the measured capacitance related to the distance from the controller surface to each joint.

Former research in the CV field introduced 2.5D representation [59, 44] that estimates joint position in the 2D image and projects the position back to the 3D position using depth value of the pixel. We also introduced 2.5D representation into our problem so that we can utilize novel CV approaches. The estimated 2D joint position in the capacitive image is converted to a 3D position using the distance from the controller surface to the joint, which is separately estimated from the capacitance value.

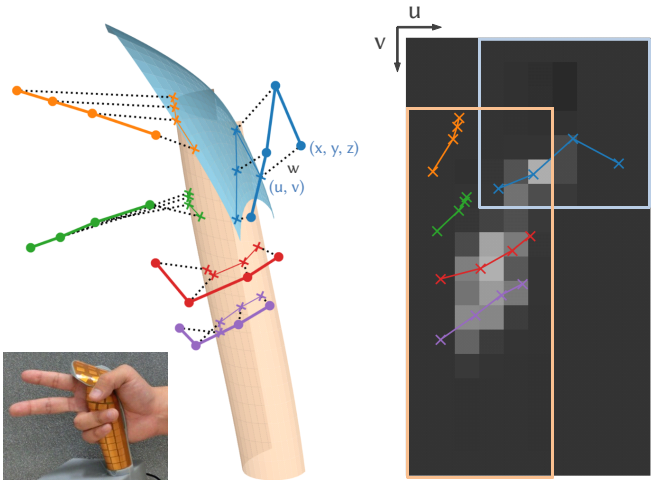

Figure 4. Joint position projection on the curved surface of the sensor sheet. Blue, orange, green, red, and purple polylines (counter-clockwise from the top right) represent thumb, index, middle, third, and little fingers, while blue (top flat) and orange (bottom cylindrical) surfaces are the approximation curve of the sensor sheet (*Left*). We projected each joint position (x, y, z) vertically (dotted line) onto the curved surface (u, v, w), which is indicated by cross markers and thin lines. Blue (top right) and orange (bottom left) areas represent UV mapped regions of each surface, and colored lines represent corresponding projected fingers on UV space (*Right*).

However, the conversion between 3D position and 2.5D position in the capacitive image on the curved surface of the electrode sheet is not obvious, while the conversion of the depth image is represented with a camera projection matrix. Therefore, we modeled the curved surface of the electrode sheet based on the device shape, and performed a projection of each 3D joint position onto UV space defined along with the curved surface of the electrode sheet (Figure 4), and employed joint position estimation in UV space.

For projecting 3D position of each joint $J_k^{3D} = (x_k, y_k, z_k)$ on the capacitive image $\mathbf{I} = \{p_{uv}\}$, we designed two polynomial surface functions $\mathrm{S_{thumb}}, \mathrm{S_{finger}}$ that define (x, y, z) position of (u, v) point on the surface, where $k \in 1, ...20$ represents the number of finger joints in one hand and $u \in 1, ...10, v \in 1, ...18$ represent horizontal and vertical coordinate of capacitive image. Both functions denote the surface of the electrode sheet inside the device, which shape is defined by the design data of the device. $\mathrm{S_{thumb}}$ represents the flat part on top near thumb (Figure 4; blue surface), while $\mathrm{S_{finger}}$ represents the cylinder part of grip (Figure 4; orange surface). Corresponding UV coordinate of these surfaces is aligned with the position of each sensor electrode on the surface, and surfaces are mapped in UV space of capacitive image (Figure 4; right). We use $\mathrm{S_{thumb}}$ for projecting the 3D position of the thumb, and $\mathrm{S_{finger}}$ for projecting other fingers. The overlapped region of two surfaces shares their vertex positions, and projected positions using two surface functions are consistent in the region. We also defined two functions $\mathrm{N_{thumb}}, \mathrm{N_{finger}}$ that represent normal vector at a specific point (u, v) on the surface using partial derivatives of $\mathrm{S_{thumb}}, \mathrm{S_{finger}}$.

Assuming 2D keypoint estimation in the capacitive image, k^{th} keypoint J_k^{2D} in the capacitive image can be defined as

$J_k^{2D} = (u_k, v_k)$ using UV coordinate on the image. For the 2.5D representation of joint position, we added vertical distance w_k from electrode surface along the normal vector at the point on the surface (Figure 4; black dotted lines), and defined $J_k^{2.5D} = (u_k, v_k, w_k)$.

We can restore the 3D position of joint J_k^{3D} from $J_k^{2.5D}$ using surface function S and normal function N as follows:

$$J_k^{3D} = \mathrm{S}(u_k, v_k) + w_k \mathrm{N}(u_k, v_k), \tag{2}$$

$$\mathrm{S},\ \mathrm{N} = \begin{cases} \mathrm{S}_{\mathrm{thumb}},\ \mathrm{N}_{\mathrm{thumb}} & k \in \textit{thumb joints} \\ \mathrm{S}_{\mathrm{finger}},\ \mathrm{N}_{\mathrm{finger}} & k \in \textit{finger joints} \end{cases} \tag{3}$$

where S represents a surface function that defines (x, y, z) position of (u, v) point on the surface and N represents a normal function that defines a normal vector at (u, v) point on the surface.

While preparing the training dataset, we converted each 3D joint position J_k^{3D} back to the corresponding position $J_k^{2.5D}$ in 2.5D representation. Conversion is solved as a non-linear optimization problem of minimizing the distance between recorded 3D joint position J_k^{3D} and projected joint position:

$$|J_k^{3D} - \mathrm{S}(u_k, v_k) - w_k \mathrm{N}(u_k, v_k)|. \tag{4}$$

We use Levenberg-Marquardt solver with the initial UV coordinate value of the nearest electrode from each joint position.

APPLYING TWO TYPES OF NETWORK ARCHITECTURES

We employed two types of hand pose estimation network architectures commonly studied in the CV field. While the direct regression architecture learns the direct transformation from the intensity values of 2D capacitive image $\{p_{uv}\}$ to the 2.5D position of each keypoint $J_k^{2.5D} = (u_k, v_k, w_k)$, which are defined by 2.5D representation in the previous section, the heatmap detection architecture learns the latent representation of keypoints as heatmaps [27].

The estimated 2.5D positions $J_k^{2.5D}$ from both networks are converted to 3D position J_k^{3D} using Equation 2 outside of the network architecture for use in applications or evaluations.

Training Dataset

For the training dataset, we selected 12 subjects (8 males and 4 females) who have a wide variety of hand shapes. Figure 5 shows profiled characteristics of hands in the dataset, plotted based on middle finger length (between the tip of middle finger and crotch) and palm width (between the edge of index finger root and the edge of little finger root). The dotted and dash-dotted lines are based on the values of previous somatometry [48].

We leveraged the OptiTrack motion capture system [26] with 12 cameras surrounding the recording hand. The system tracks the position of 3 mm retroreflective markers attached on each finger, along with the position and orientation of the handheld device using four 6 mm markers attached to it (Figure 6). We put 20 markers on the top of each joint (MCP, PIP, DIP) and the tip of each finger, referring hand model structure of commonly used public datasets such as MSRA14 [39],

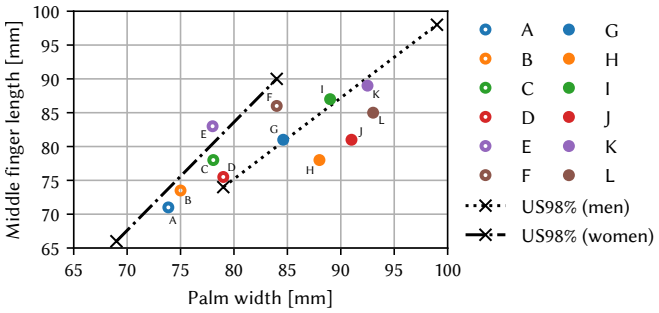

Figure 5. Hand shape profile of recorded dataset. Plots labeled A to L represent the shape of each hand in the recordings. The dotted and dash-dotted lines represent 1 to 99 percentile ranges of measured hand sizes of men and women in the United States.

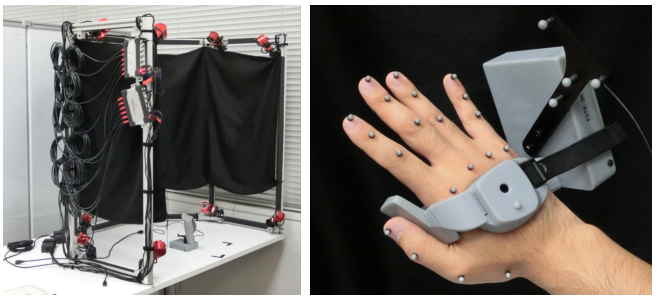

Figure 6. Marker setup for dataset recording. We employed an optical tracking system for capturing label data for training (*Left*), putting 3 mm retroreflective markers on top of all joints and tips of right-hand fingers (*Right*). 6 mm markers are attached to the controller to track its position and orientation.

MSRA15 [45], and BigHand2.2M [57]. The captured position of each marker is converted to the local position from the origin defined on the handheld device.

We recorded capacitive image and joint position simultaneously at 60 frames per second while the subject performs 11 different sets of specific movements, which are selected aiming to cover articulated natural hand movement in the usage of interaction. The 11 movements consist of opening and closing hand (*open-close*), nailing on the device (*trumpet*), bending and stretching each finger successively (*bend*), bending each finger successively and stretching them successively (*1234*), spreading fingers (*spread*), rotating around thumb (*thumb-rotate*), bending thumb inside and outside (*thumb-bend*), and repeating each specific pose (*peace, three, telephone, rock-n-roll*). The recording was performed four times per each movement, and we used the first three recordings for training, while the last one recording was used for validation. We settled the device at the proper position in the subject's hand each time before recording sessions. Prepared training dataset contains 344015 frames, and the validation dataset contains 111165 frames. For cross-validation in the latter evaluation section, we prepared 12 different datasets consisting of 11 subjects, and evaluated trained models with recordings of the excluded subject.

Also, every five frames of the recorded dataset were temporally combined before feeding them to the network, for increasing temporal stability of estimation. We extracted every

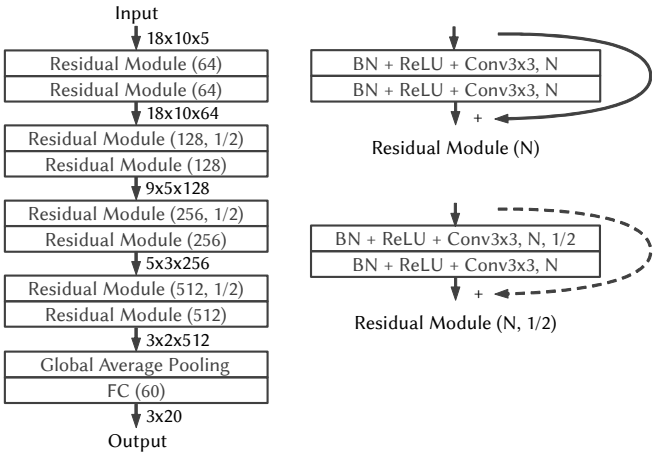

Figure 7. Structure of the direct regression network. The overall structure of the network (*Left*). The *Residual Module* block represents the structure illustrated on *Right*, while *N* represents the channel size of the convolution layer. The solid curved line on *Right* represents a skip connection that adds an image of the previous layer, while the dotted line represents a skip connection with 1x1 convolution for aligning channel size. *BN* stands for batch normalization, *Conv* for convolution, *ReLU* for rectified linear unit activation function, and *FC* for fully connected.

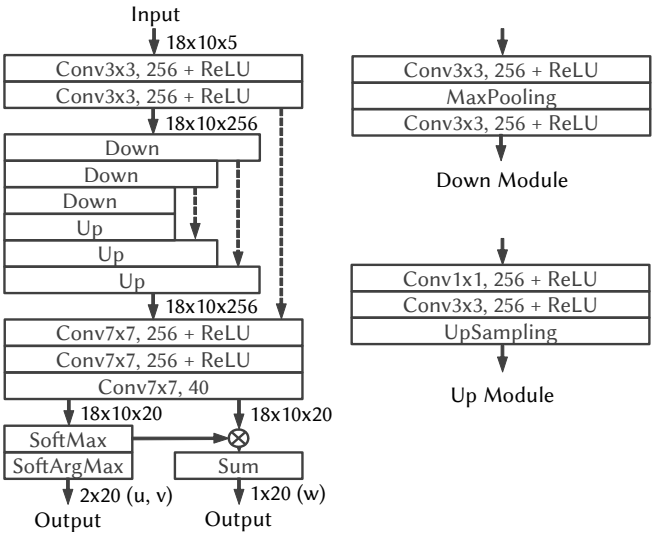

Figure 8. Structure of the heatmap detection network. The overall structure of the network (*Left*). The *Down* and *Up* block represents the structures illustrated on *Right*. The dotted line represents a skip connection that concatenates an image pass-through from the previous layer. *Conv* stands for convolution, and *ReLU* for rectified linear unit activation function.

other frame from 10 successive frames alternately and made a 165 ms sequence of frames with 33 ms intervals. The extracted sequences are fed to the network as $18 \times 10 \times 5$ resolution images. We determined the number of input frames based on the sensitivity visualization by SmoothGrad [42].

Direct Regression Network

We followed ResNet-18 architecture [18] for building a direct regression network, and reordered *BN*, *ReLU*, and *Conv* layers according to [19]. We removed the first convolutional layer and pooling layer from the original network because input image resolution is limited. Figure 7 shows the network architecture we use.

The input of the network is $18 \times 10 \times 5$ capacitive image, and the output is 3×20 values that represent 2.5D coordinate $J_k^{2.5D}$ of 20 joints in one hand, which is directly estimated from the network. We employed L_1 norm (the sum of absolute difference of the components of the vectors) for loss function.

Heatmap Detection Network

We followed latent 2.5D heatmap regression architecture [27] for building a heatmap detection network. Figure 8 shows the network architecture we use, employing hourglass architecture [37, 40] with a fixed number of channels. The difference from Iqbal *et al.*'s implementation is that we removed pooling layers from the input block and reduced the number of stacked size of hourglass network architecture due to the small resolution input image.

The input of the network is $18 \times 10 \times 5$ capacitive image and the outputs are 2×20 (u, v) and 1×20 (w) values that represent 2.5D representation $J_k^{2.5D}$ of 20 joints in one hand, which is estimated via latent heatmap representation in the network. The output of the last convolutional layer contains latent heatmap $\mathbf{H}^*$ and latent depth map $\mathbf{D}^*$ for each joint. Latent heatmap of k^{th} joint $\mathbf{H}_\mathbf{k}^*$ was converted to $J_k^{2D} = (u_k, v_k)$ coordinates of each joint through spatial softmax and soft-argmax functions below:

$$\mathbf{H_k}(u,v) = \frac{\exp(\beta_k \mathbf{H}_\mathbf{k}^*(u,v))}{\sum_{u',v' \in \Omega} \exp(\beta_k \mathbf{H}_\mathbf{k}^*(u',v'))}, \tag{5}$$

$$\hat{J_k^{2D}} = \sum_{u,v \in \Omega} \mathbf{H_k}(u,v) \cdot (u,v), \tag{6}$$

where $\hat{J_k^{2D}}$ represents the estimated values of the network, β_k represents temperature parameter for k^{th} joint, and Ω represents all pixel coordinates in the image. Latent depth image $\mathbf{D}_\mathbf{k}^*$ was multiplied with heatmap $\mathbf{H_k}$ for extracting depth of each joint and converted to $\hat{w}_k$ for each joint using the following equation:

$$\hat{w_k} = \sum_{u,v \in \Omega} \mathbf{H_k}(u,v) \circ \mathbf{D}_\mathbf{k}^*(u,v), \tag{7}$$

where circle operator means element-wise multiplication.

We used normalized value [-1, 1] to represent (u, v) coordinate on the capacitive image, while using w as vertical distance in meter, and defined loss function as follows:

$$\mathrm{L}(J_k^{2.5D}) = \mathrm{L}_1(\hat{J_k^{2D}}, J_k^{2D}) + \alpha \mathrm{L}_1(\hat{w_k}, w_k). \tag{8}$$

We employed L_1 norm for loss function, and selected 25 for scaling parameter α for equating the scale of two losses.

Training Parameters and Additional Details

We use a stochastic gradient descent optimizer with a momentum of 0.9 for training both networks. The training was done with a batch size of 32 for 70 epochs, and the model with the lowest validation error is stored. The learning rate was

decayed by a factor of 10 every 30 epochs, which is started from the initial learning rate of 0.03 for the direct regression network and 0.001 for the heatmap detection network. We defined these learning parameters based on the learning curve of each network architecture. All trainable values of each network are initialized with Glorot uniform function [15] except temperature parameter β_k in Equation 5, and β_k is initialized with a constant value of 200. We implemented each network using Keras [8] with TensorFlow [1] backend.

Dataset recording, training, and evaluation were done only with the right hand. For real-time demonstration, we flipped the input capacitive image and output position of the right-hand model for applying the result to left-hand estimation.

EVALUATION

We evaluated the performance of two types of models using the mean of absolute error (MAE), which is Euclidean distance between ground truth 3D joint position and estimated 3D joint position, which is converted from a 2.5D representation (u, v, w) to a 3D position (x, y, z) before evaluation, using Equation 2.

We performed leave-one-out cross-validation using a one-subject-out dataset to assess the performance of our networks against unseen hands, which is the most important metric for delivering our method to users worldwide. We trained 12 different models with one-subject-out datasets first, and tested each model with the dataset of the excluded subject. Table 1 shows the MAEs of models trained with one-subject-out datasets. Estimation error of subjects at the edge of the distribution (*A, E, J, L*) in Figure 5 are relatively larger than that of other subjects in both network architectures.

We also performed an evaluation using the percentage of correct keypoints (PCK) metrics [2] and the percentage of correct frame (Frame PCK) metrics [47]. Figure 9 shows the result of each model and a result of the VGG19 [41] network, which we chose for the baseline of an unmodified simpler network. Although the comparison of the direct regression model and the heatmap detection model in previous research reported that the heatmap detection model performs better than direct regression for pose estimation from a camera image [27, 56], our result of adopting these network architectures to the capacitive image shows the opposite result.

Comparison to Other SOTA Hand Pose Estimation Method

Performing comparison to another method is not simple, because there is no available dataset that contains the simultaneous recording of capacitive image and other input data for them, however, comparing our performance with other competitive methods would be informative.

Comparing to another estimation method using a capacitive image, Le *et al.* [32] evaluated their estimation error of fingertip position of five fingers using a dataset of 2 subjects, while the model is trained with 14 subjects. Their result of Euclidean distance in 3D space is 15.2 mm, which is larger than our mean MAEs. We also calculate the mean MAEs of only fingertip positions from our result; 16.60 mm for direct regression and 17.81 mm for heatmap detection. The estimation results of fingertips are worse than theirs; however, their recording only includes movements near the sensing electrodes while holding a smartphone, whereas our recording includes movements away from the controller surface, which is a more challenging task.

Comparing to a glove based method, Glauser *et al.* [14] evaluated their accuracy with one-subject-out datasets of 10 subjects. Their result of mean MAEs of joint angles of unseen hands is 7.6 deg when one-minute calibration is available and 8.3 deg without calibration. We calculated MAEs in the same metrics of Tkach *et al.*'s representation of hand pose [49], and the mean of the converted MAEs are 11.33 deg for the direct regression model and 13.67 deg for the heatmap detection model. This suggests the glove based method is suitable for precise finger tracking applications, such as motion capturing. However, our methods have an advantage in the ease of use since it can track finger movement when just grabbing a controller without wearing anything.

Comparing to CV based methods, the best MAE results for unseen hands in HIM2017 competition [56] is 10.6 mm of RCN-3D [21] and mean MAEs of Top 5 network architecture is 11.9 mm, while all networks are trained with five subjects and evaluated other five subjects of BigHand2.2M dataset [57]. If joints are occluded from cameras, the results decrease to 14.6 mm of V2V-PoseNet [36] and 15.2 mm. Although the MAE value is dependent on the evaluation dataset, the result in Table 1 suggests that our model has comparable performance to SOTA CV based algorithms.

Comparing to a commercial product, we evaluated the performance of the Valve Index controller. We recorded ground truth joint angle of *open-close* movement of 11 subjects (A to L; except H) and estimation result acquired from the Skeletal Input API of SteamVR 1.8.21 with *VRSkeletalMotionRange_WithController* option, which provides the most accurate pose compared to the actual hand pose. The mean estimation errors of DIP and PIP joint angles of thumb, and DIP, PIP, and MCP angles of the other four fingers are 20.09 deg and 12.98 deg, while ours are 8.88 deg and 7.12 deg for the same movement of the direct regression model. We conducted a further experiment in terms of user perception using our arbitrary hand pose demonstration (Figure 11; *BottomLeft*). We recruited 20 new subjects (17 males and 3 females; 19 right-handed and 1 both-handed) and asked them to answer five questions while playing the demonstration. Half of them were asked to hold our prototype in the right hand and Index controller in the left hand, while the other half were asked to hold them in the opposite hand. The questions consist of two questions regarding thumb (Q1, Q2), two questions regarding fingers (Q3, Q4), and one question regarding overall impression (Q5).

Q1 "How accurate do you feel the bending thumb (angle of each joint) on the screen is? Score 1 (completely wrong) to 7 (completely accurate)"

Q2 "How broad is the range of the reflected motion (variation of the movement)? Score 1 (reflecting only one pattern of movement) to 7 (reflecting every movement of entire range)"

Q3 "How accurate do you feel the bending four fingers (angle of each joint) on the screen is? Score 1 (completely wrong) to 7 (completely accurate)"

Model	*- A*	*- B*	*- C*	*- D*	*- E*	*- F*	*- G*	*- H*	*- I*	*- J*	*- K*	*- L*	*Mean*
Direct Regression	15.08	9.444	8.858	7.592	13.61	10.28	10.98	10.28	10.69	12.49	9.935	12.94	11.02
Heatmap Detection	16.24	10.16	9.559	8.400	13.67	11.29	11.01	10.90	12.11	12.67	11.47	13.97	11.79

Table 1. Cross-validation using one-subject-out training datasets. Each model is trained with one-subject-out datasets and evaluated with the excluded subject. *- A* to *- L* column represents the excluded subject, and *Mean* column are the mean of values of each row. All values in the table are MAE of estimated joint position in millimeter. MAEs of direct regression model are smaller than that of heatmap detection model with all subjects.

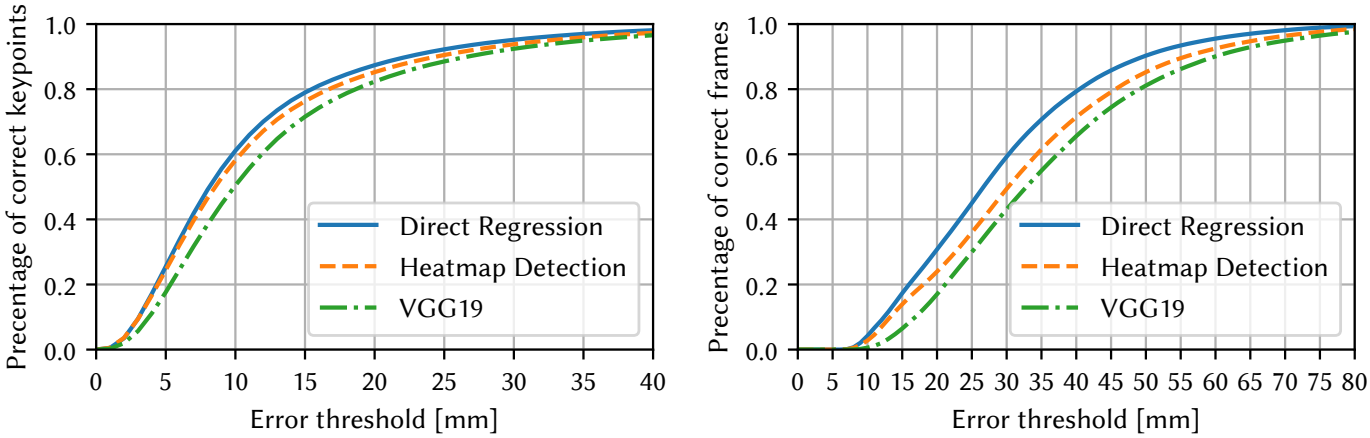

Figure 9. Percentage of correct keypoints/frames of each model. Each curve in the *left* plot represents the percentage of keypoints estimated within a specific error threshold, while curves in the *right* plot represent the percentage of correctly estimated frames. Direct regression model outperforms heatmap detection model in both metrics.

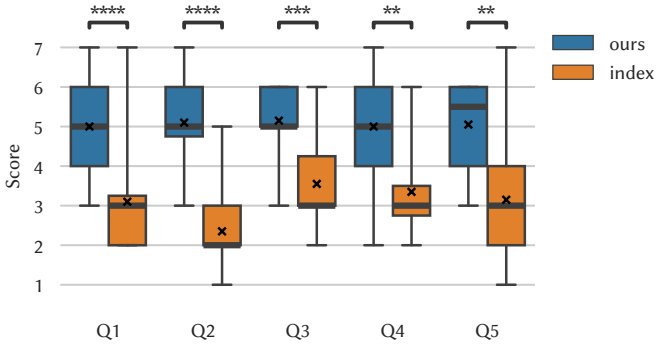

Figure 10. Perceptional difference between two controllers in the application. Boxes and vertically extending lines represent the minimum, lower quartile, upper quartile, and the maximum score of each question. The thick line and the cross marker in the box represent the median and mean of the scores. Our prototype earned significantly higher scores with p < 1e-4 (****), p < 1e-3 (***), and p < 1e-2 (**) for Wilcoxon signed-rank test.

Q4 "How far is the maximum distance of reflected motion away from the controller surface? Score 1 (reflecting only when finger touches) to 7 (reflecting every movement of entire range)"

Q5 "How much do you feel the virtual hand matches your hand movements? Score 1 (not at all) to 7 (completely matching)"

The skeletal movement of the Index controller was obtained with *VRSkeletalMotionRange_WithoutController* option, which provides a retargeted natural motion as if the user does not hold a controller. Our scores of each question (Figure 10) were significantly better than those of Index controller, in particular, regarding thumb. Before both recording and experiment using the Index controller, we asked subjects to keep drumming their fingers at least one minute so that SteamVR driver can calibrate for their hand correctly.

Estimation Accuracy and Size of the Networks

For confirming that our network comparison result is universal and independent from small changes in the network architecture, we conducted the same comparison while varying the size of networks. We modified the width of each layer of the networks described in Figure 7 and Figure 8 to half-size (*Half*) and twice-size (*Twice*). The numbers of parameters of half-, normal-, and twice-size networks are 2.8 M, 11 M, and 44 M for the direct regression network, and 4 M, 16 M, and 64 M for the heatmap detection network. Table 2 shows the change of the mean of MAEs depending on the network size. Estimation error decreases as the size of the network increases; however, the direct regression network performs better in all network sizes.

Estimation Accuracy and Number of Electrodes

The smaller number of the electrode is preferred considering cost since the number of input channels relates to the manufacturing cost, and small input resolution also can keep the computational cost low. Therefore, we conducted a further investigation of the relationship between the number of the electrodes and the estimation accuracy. We generated pseudo datasets of other numbers of electrodes using a recorded dataset with the current prototype, without paying additional effort to create another prototype.

We replaced each value of two successive electrodes with the mean value of them in the original image for simulating the capacitive image from a smaller number of electrodes. Compared to the original image that has 62 different values, vertical drop image has 31, horizontal drop image has 34, and both drop image has 17 different values in the image. We refer to these images as *V 1/2*, *H 1/2*, and *1/4* in Table 3. Estimation error of both models increases when the number of electrodes

Model	*Half*	*Normal*	*Twice*
Direct Regression	11.26	11.02	10.84
Heatmap Detection	11.91	11.79	11.73

Table 2. Estimation error and size of networks. *Half* and *Twice* represents the size of the network relative to the *Normal* network. All values are MAE of estimated joint position in millimeter. The direct regression model performs better, regardless of the network size.

Model	*Orig*	*V 1/2*	*H 1/2*	*1/4*
Direct Regression	11.02	13.51	13.58	16.35
Heatmap Detection	11.79	13.25	13.90	15.63

Table 3. Estimation error and number of electrodes. *Orig* stands for the original resolution (64ch), *V 1/2* for half vertical resolution (31ch), *H 1/2* for half horizontal resolution (34ch), and *1/4* for quarter resolution (17ch). All values are MAE of estimated joint position in millimeter. MAEs increase along with the decrease of input number of electrodes.

decreases. However, a smaller number of electrodes can keep computational and manufacturing costs low. Thus, for an application in which cost or computational resource is limited, we can choose the input number of electrodes depending on the required estimation accuracy.

Note that we just equalized adjacent pixels in the input image for simulating a lower number of electrodes, and the size of the input image is not changed. The same network architecture (Figure 7, Figure 8) is used for this evaluation to solely assess the effect of the change of input electrode number.

Estimation Accuracy and Pose Variation in the Dataset

A smaller dataset is preferred for recording, but limited hand pose variation could diminish the performance of the trained network. Thus, we performed leave-one-out cross-validation using a one-pose-out dataset to assess the performance of unseen poses, so that we can understand whether the performance is independent of a specific pose in the training dataset, which is essential in the real world usage.

Table 4 shows the MAEs of models trained with one-pose-out datasets, and the mean value of them. Estimation error of finger movement away from the device surface such as *trumpet* and *spread* is relatively large among all poses in both network architectures. This is because the change of capacitance relative to finger movement becomes smaller while being away from the sensor electrode, and sensing these poses is more difficult than other poses. On the other hand, *open-close*, *thumb-rotate*, and *thumb-bend* have relatively small estimation errors even if excluded from the training dataset, so these movements can be removed from the dataset for minimizing the effort of data recording.

APPLICATIONS

We implemented our hand pose estimation algorithm into demonstrations working in real-time at 90 fps on Intel Core i7-6950X 3.0GHz CPU and NVIDIA GTX1080 GPU. The demonstration contains scenes of interacting with objects in VR, including dexterous manipulation of small objects as well as non-verbal communication using five finger motions (Figure 11). Joint position estimation is performed by the direct regression network running on GPU.

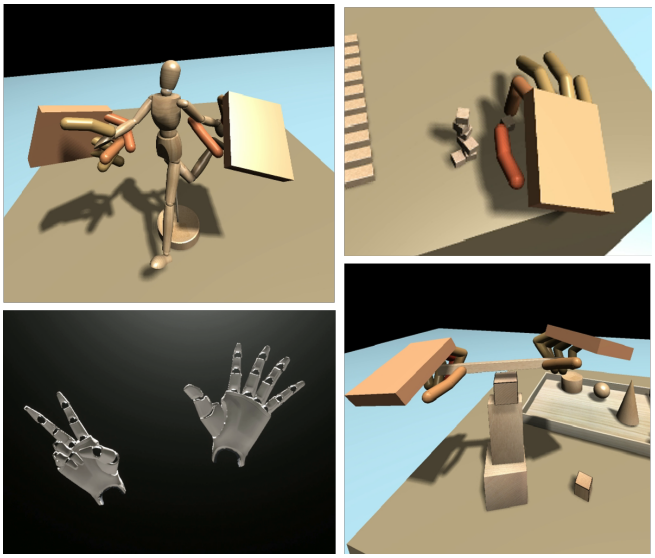

Figure 11. Example applications using hand pose estimation. Object interaction using hands in VR, including dexterous manipulation of small objects (*TopLeft, TopRight, BottomRight*). Non-verbal communication using five finger motions (*BottomLeft*).

In the interaction demonstration scene (Figure 11; *TopLeft, TopRight, BottomRight*), all interactions between fingers and objects are fully simulated with our algorithm. After transforming estimated local finger joint positions to world space referring to VIVE Tracker [10] attached to the controller, we performed physics simulation using a collision model of the virtual hand, and calculate the final pose the virtual hand reflecting the interaction between objects in the virtual world. When a fingertip touches on the object, our algorithm makes a friction constraint which allows only horizontal translation and rotation relative to the surface of the finger body at the contact point. This constraint is made only one for each finger, and behaves as the representative constraints between the finger and an object [46]. Once a constraint is made, we define the friction strength based on how much a finger intersects into an object. We also visualize the strength of friction with the color of each finger. The red finger represents it has a constraint with an object with a strong friction.

Finally, the physics engine calculates the final force applied to each object and finger, and decide the final pose of the hand in a frame. Five finger tracking and our implementation enable users to precisely handle objects in an arbitrary way, not depending on how users hold an object. For instance, users can softly pinch small objects with index finger and thumb, grabbing the same object with both hands for holding firmly, and interact with multiple objects collided with each other.

Also, we developed another demonstration using Unreal Engine [22] to illustrate the highly flexible movement of the estimated five fingers extending the range of expression of non-verbal communication using hands (Figure 11; *BottomLeft*). The demonstration shows that the potential to reflect the user's arbitrary hand pose into virtual hands enables carrying user's emotions more efficiently compared to using predefined hand pose emotion.

Model	*- OC*	*- trumpet*	*- bend*	*- 1234*	*- spread*	*- TR*	*- TB*	*- peace*	*- three*	*- TEL*	*- RnR*	*Mean*
Direct Regression	5.833	11.51	6.879	7.641	10.64	5.187	5.116	8.611	9.939	7.482	9.178	8.001
Heatmap Detection	7.046	12.63	8.961	9.853	11.59	7.103	7.142	10.31	11.61	9.828	11.84	9.811

Table 4. Cross-validation using one-pose-out training datasets. Each model is trained with one-pose-out datasets and evaluated with the excluded pose. Each column represents a pose excluded from the training dataset, and *OC* stands for open-close, *TR* for thumb-rotate, *TB* for thumb-bend, *TEL* for telephone, and *RnR* for rock-n-roll. Values in *Mean* column are the mean of values of each row. All values are MAE of estimated joint position in millimeter. The *trumpet* and *spread* poses have relatively large MAEs with both network architectures, while *OC*, *TR*, and *TB* poses have small MAEs.

CONCLUSION AND DISCUSSION

We built a handheld finger tracking prototype which has 62 proximity sensing electrodes on the curved surface, and recorded datasets consisting of 11 poses from 12 subjects who have a variety of hand size. We evaluated the performance of two types of hand pose estimation architectures based on previous research in CV, while 2.5D representation and joint position projection technique are introduced for adopting the methods on the curved capacitive image.

Suitable Network Architecture for Capacitive Images

Although heatmap detection architecture is dominant in pose estimation in the field of CV, direct regression architecture performs better with our dataset. Commonly used image datasets for hand pose estimation have 320×240 resolution [39, 45] or 640×480 [51, 57] resolution. However, our capacitive image dataset has only 18×10 resolution. This limited resolution is considered to hinder the performance of heatmap detection architecture since the architecture represents joint positions as a heatmap of image resolution. Increasing the number of sensing electrodes causes an increase in manufacturing cost. In addition, making electrode sizes smaller for placing more in the same space diminishes the signal level of each electrode. Thus, direct regression architecture would be more suitable for capacitive images.

Re-Training the Model According to Hardware Change

The trained model is durable to small signal changes caused by sweat on the palm or foreign objects, such as rings and watches, as expressed in the video figure. However, the change of the grounding condition (wireless setup) or grounded conductive material installed nearby electrodes affect the measured capacitance value [17], and cause problematic estimation error. One solution for this problem is that designing hardware so that grounded conductive materials, such as metallic contacts of physical buttons, are installed behind the shielding layer on the PCB for avoiding interference. Small ungrounded metallic parts like screws have less effect on measured signals; however, the trained model does not work with largely-changed hardware and is required to be trained with new datasets recorded with the new hardware. Transfer learning latent knowledge from the old model trained by the previous datasets can reduce the training cost comparing to training the new model from scratch [38]. The multi-task learning technique could reduce the number of recording datasets with the new hardware while avoiding overfitting.

Difficulties in Supporting Various Hand Shapes

Current estimation accuracy of unseen hands achieved comparable performance to SOTA CV algorithms [56], and is sufficient for natural interaction as described in the video material. However, performance with the hands at the edge of the distribution in the dataset is relatively low (Table 1). This could be problematic if we deliver our method to users worldwide. Per-user fine-tuning training like [14] cannot be conducted with our current training method realistically since we require an optical tracking system for recording the training dataset. For better adaptation for various hand sizes, a method for augmenting or synthesizing the capacitive image needs to be studied.

Hand Shape Mismatch causes Fingertip Misalignment

Our current algorithm cannot estimate the shape of users' hands, such as the thickness of the finger, and our physics hand model in current implementation also does not reflect the shape of users' hands into the simulation. These limitations cause a gap between the actual hand pose and virtual hand pose, and users feel difficulty in aligning two fingertips, particularly when they perform pinching action. Users can still pinch a small object if users tweak their hand pose, referring to the virtual hand they can see in the virtual world. However, this difficulty would limit the dexterous manipulation of untrained users. Reflecting the estimated finger bone length to the model of the virtual hand could make the gap smaller between the shape of the virtual hand and the actual hand, and would provide easier manipulation to untrained users.

Five Finger Tracking along with Auxiliary Input

The controller held in hand would hinder the natural movement of fingers, such as picking real objects or hand-hand interactions, and tracking bare-hand should be the right way for the eventual natural interaction. However, physical input buttons and sticks are still in demand for inputs with severe timing constraints and locomotion implementations. The advantage of our method is its capability of integrating easily to existing controllers for detecting continuous five finger movements away from the button, which is crucial for a better feeling of the presence of our hand in VR. Hand pose estimation from a capacitive image enables intuitive interaction using hand along with auxiliary physical inputs and haptic feedback, which is the finger tracking technology that the current industry needs.

ACKNOWLEDGMENT

We would like to thank Hiroshi Matsuike and Yoshitaka Mitsui for supporting demonstration implementation, Mika Fujiwara for helping dataset recording, Kunihito Sawai and Toru Kuronuma for designing the prototype, Hirofumi Kaneko for authoring the video material, and Mark Magee for proofreading the paper.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2014. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 3686–3693.

[3] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. 2016. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4733–4742.

[4] 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)*. ACM, New York, NY, USA, 549–556. DOI: http://dx.doi.org/10.1145/2807442.2807450

[5] Daniel K.Y. Chen, Jean-Baptiste Chossat, and Peter B. Shull. 2019. HaptiVec: Presenting Haptic Feedback Vectors in Handheld Controllers Using Embedded Tactile Pin Arrays. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 171, 11 pages. DOI:http://dx.doi.org/10.1145/3290605.3300401

[6] 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)*. ACM, New York, NY, USA, 1504–1514. DOI: http://dx.doi.org/10.1145/2858036.2858125

[7] Inrak Choi, Eyal Ofek, Hrvoje Benko, Mike Sinclair, and Christian Holz. 2018. Claw: A multifunctional handheld haptic controller for grasping, touching, and triggering in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 654.

[8] François Chollet and others. 2015. Keras. https://keras.io. (2015).

[9] 3D Systems Corporation. 2019a. 3D Printers. (2019). https://www.3dsystems.com/3d-printers/plastic.

[10] HTC Corporation. 2019b. VIVE Tracker. (2019). https://www.vive.com/us/vive-tracker/.

[11] Valve Corporation. 2019c. Valve Index Controller. (2019). https://store.steampowered.com/valveindex.

[12] Max Planck Institute for Informatics. 2019. MPII Human Pose Dataset. (2019). http://human-pose.mpi-inf.mpg.de/.

[13] Oliver Glauser, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. 2019a. Deformation Capture via Soft and Stretchable Sensor Arrays. *ACM Trans. Graph.* 38, 2, Article 16 (March 2019), 16 pages. DOI: http://dx.doi.org/10.1145/3311972

[14] Oliver Glauser, Shihao Wu, Daniele Panozzo, Otmar Hilliges, and Olga Sorkine-Hornung. 2019b. Interactive hand pose estimation using a stretch-sensing soft glove. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 41.

[15] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[16] Nan-Wei Gong, Jürgen Steimle, Simon Olberding, Steve Hodges, Nicholas Edward Gillian, Yoshihiro Kawahara, and Joseph A. Paradiso. 2014. PrintSense: A Versatile Sensing Technique to Support Multimodal Flexible Surface Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1407–1410. DOI: http://dx.doi.org/10.1145/2556288.2557173

[17] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. 2017. Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3293–3315. DOI: http://dx.doi.org/10.1145/3025453.3025808

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

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.

[20] Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. 2018. Improving landmark localization with semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1546–1555.

[21] Sina Honari, Jason Yosinski, Pascal Vincent, and Christopher Pal. 2016. Recombinator networks: Learning coarse-to-fine feature aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5743–5752.

[22] Epic Games Inc. 2019a. Unreal Engine. (2019). https://www.unrealengine.com/.

[23] Facebook Inc. 2016. Oculus Touch. (2016). https://www.oculus.com/rift/accessories/.

[24] Facebook Inc. 2019b. Oculus Quest. (2019). https://www.oculus.com/quest/.

[25] Leap Motion Inc. 2012. Leap Motion. (2012). https://www.leapmotion.com/.

[26] NaturalPoint Inc. 2019c. OptiTrack. (2019). https://optitrack.com/.

[27] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. 2018. Hand Pose Estimation via Latent 2.5D Heatmap Regression. In *The European Conference on Computer Vision (ECCV)*.

[28] Yasha Iravantchi, Mayank Goel, and Chris Harrison. 2019. BeamBand: Hand Gesture Sensing with Ultrasonic Beamforming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 15, 10 pages. DOI: http://dx.doi.org/10.1145/3290605.3300245

[29] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: Freehand 3D Interactions Anywhere Using a Wrist-worn Gloveless Sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 167–176. DOI: http://dx.doi.org/10.1145/2380116.2380139

[30] Wakaba Kuno, Yuta Sugiura, Nao Asano, Wataru Kawai, and Maki Sugimoto. 2017. 3D Reconstruction of Hand Postures by Measuring Skin Deformation on Back Hand. In *ICAT-EGVE 2017 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*, Robert W. Lindeman, Gerd Bruder, and Daisuke Iwai (Eds.). The Eurographics Association. DOI: http://dx.doi.org/10.2312/egve.20171362

[31] Tactual Labs. 2019. Controller Video. (2019). https://www.tactuallabs.com/.

[32] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, 779–792.

[33] Jaeyeon Lee, Mike Sinclair, Mar Gonzalez-Franco, Eyal Ofek, and Christian Holz. 2019. TORC: A Virtual Reality Controller for In-Hand High-Dexterity Finger Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 71.

[34] Jhe-Wei Lin, Chiuan Wang, Yi Yao Huang, Kuan-Ting Chou, Hsuan-Yu Chen, Wei-Luan Tseng, and Mike Y. Chen. 2015. BackHand: Sensing Hand Gestures via Back of the Hand. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 557–564. DOI: http://dx.doi.org/10.1145/2807442.2807462

[35] Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 220–229. DOI: http://dx.doi.org/10.1145/3132272.3134130

[36] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5079–5088.

[37] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*. Springer, 483–499.

[38] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. on Knowl. and Data Eng.* 22, 10 (Oct. 2010), 1345–1359. DOI: http://dx.doi.org/10.1109/TKDE.2009.191

[39] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1106–1113.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[41] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556 (2014).

[42] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[43] David J Sturman and David Zeltzer. 1994. A survey of glove-based input. *IEEE Computer graphics and Applications* 14, 1 (1994), 30–39.

[44] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. 2017. Compositional human pose regression. In *Proceedings of the IEEE International Conference on Computer Vision*. 2602–2611.

[45] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. 2015. Cascaded hand pose regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 824–832.

[46] Anthony Talvas, Maud Marchal, Christian Duriez, and Miguel A Otaduy. 2015. Aggregate constraints for virtual manipulation with soft fingers. *IEEE transactions on visualization and computer graphics* 21, 4 (2015), 452–461.

[47] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. 2012. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 103–110.

[48] Alvin R Tilley. 2002. *The measure of man and woman: human factors in design*. John Wiley & Sons.

[49] Anastasia Tkach, Andrea Tagliasacchi, Edoardo Remelli, Mark Pauly, and Andrew Fitzgibbon. 2017. Online Generative Model Personalization for Hand Tracking. *ACM Trans. Graph.* 36, 6, Article 243 (Nov. 2017), 11 pages. DOI: http://dx.doi.org/10.1145/3130800.3130830

[50] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 648–656.

[51] 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)* 33, 5 (2014), 169.

[52] Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[53] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional Pose Machines. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[54] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 47–50. DOI: http://dx.doi.org/10.1145/2817721.2817737

[55] Hui-Shyong Yeo, Erwin Wu, Juyoung Lee, Aaron Quigley, and Hideki Koike. 2019. Opisthenar: Hand Poses and Finger Tapping Recognition by Observing Back of Hand Using Embedded Wrist Camera. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New York, NY, USA, 963–971. DOI: http://dx.doi.org/10.1145/3332165.3347867

[56] Shanxin Yuan, Guillermo Garcia-Hernando, Bjorn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. 2018. Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[57] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4866–4874.

[58] Junjian Zhang, Yaohao Chen, Satoshi Hashizume, Naoya Muramatsu, Kotaro Omomo, Riku Iwasaki, Kaji Wataru, and Yoichi Ochiai. 2018. EXController: Enhancing Interaction Capability for VR Handheld Controllers Using Real-time Vision Sensing. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology (VRST '18)*. ACM, New York, NY, USA, Article 87, 2 pages. DOI: http://dx.doi.org/10.1145/3281505.3283385

[59] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. 2017. Weakly-supervised transfer for 3d human pose estimation in the wild. In *IEEE International Conference on Computer Vision, ICCV*, Vol. 3. 7.