Enhancing Interaction Capability for VR Handheld Controllers: Exploratory Approaches using Swept Frequency Capacitive Sensing and Vision Sensing in Real-time

ZHANG JUNJIAN

Graduate School of Library, Information and Media Studies

University of Tsukuba

March 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With recent rapid advances of head mounted display (HMD) technology, virtual reality (VR) and augmented reality (AR) equipment systems have entered the consumer trend. There are mainly three kinds of control solutions in VR: camera-based optical methods, data glove, and handheld motion controller. For example, Leap Motion robustly recoveries complex hand poses with high frame rate for head-mounted scenarios. And Microsoft HoloLens, can robustly recognize the Air-tap and Bloom gesture with low latency. But both of them cannot support eyes-free interactions. Glove-like controllers for VR such as Manus VR [1] and VRgluv [2] can recovery full hand pose with high quality using passive trackers and provide impressive haptic feedback that are suitable for certain games or applications. The VR consumer HMD systems, such as HTC Vive [3], Oculus Rift [4], PlayStation (PS) VR [5] headset system, whose primary interaction input devices are unanimously handheld controllers, Vive controller, Touch controller, and Move controller. With these handheld controllers, hand interaction is mainly based on touch capacitive sensing and buttons. Touch capacitive sensing, which is used to detect touch events with strong robustness, high accuracy and fast recognition speed. But for VR controllers, there is a major shortcoming that they cannot achieve more information to perform visual movements of the user's fingers.

In order to enhance the interaction and immersion in VR environments, Oculus Touch controller takes advantage of an ingenious ergonomic design combined with touch capacitive sensing, firstly providing users an enhanced sense of "hand presence". It uses capacitive sensors to determine the discrete position of fingers, rendering a quasi-reconstructed hand model to display the hand pose. Similar to the idea of Touch controllers, the Valve Knuckles controller [6] seems to be able to distinguish that how much a user's fingers like wrapping around the capacitive handle sensors.

And most recently Valve [7] redesigned the controllers with a name of Knuckles EV2 [8], improving the performance of finger tracking using a combination of high-fidelity force sensors and capac-

[1] https://manus-vr.com/
[2] https://vrgluv.com/
[3] https://www.vive.com/us/
[4] https://www.oculus.com/
[5] https://www.playstation.com/en-us/explore/playstation-vr/
[6] https://steamcommunity.com/app/633750
[7] https://www.valvesoftware.com
[8] https://www.tomshardware.com/news/valve-knuckles-ev2-developer-kits,37346.html

itive sensors. Thus, it is expected that finger tracking for controllers can increase the range and depth of interaction possible in VR.

1.2 Research Motivation

Table 1.1 provides an overview of pros and cons for the three types of VR/AR controllers. This work's motivation is located on the finger tracking on controllers, which aims to enhance the interaction possible in VR. Although both of the Touch and Knuckles controller leverage an ingenious design and traditional capacitive sensing but not support any gesture interaction when fingers are in air or not touching the sensor. In this paper, the aim is to explore new approaches to enrich inputs for VR handheld controllers. This work explores the SFC sensing method and the vision-based method. It should be noted that this work's focus is on the finger posture classification of individual fingers but not hand reconstruction or full hand pose recovery. However, for the application scenarios, the thumb postures in air recognized from the camera sensor are expected to control the movement of objects in VR, and the discrete finger gesture input from the two sensors can be used to perform a simulation of hand pose recovery. This paper focuses on Vive and Rift HMD systems because both of them are typically desktop-based VR systems with universal scalability that is not only for game application compared to PS VR.

Table 1.1: Pros and cons of the three kinds of VR controllers

	Pros	Cons	Example
Camera	Bare-hand interaction Can recover full hand pose robustly	Suffer occlusion problems Nearly no haptic feedback Limitation of field of view	Leap Motion HoloLens
Data glove	High-resolution haptic feedback Recover full hand pose with less error	Barrier to wear gloves Robust tracking needs passive tracker	Manus VR VRgluv
Handheld controller	Traditional, reliable interaction way Recognize fast with no input error Simple vibration haptic feedback Robust tracking solution	Difficult to recover full hand pose This work's Motivation	Vive Oculus Touch Knuckles PS Move

1.3 Contribution

This work's contribution is:

- First, this work presents a new exploratory design and practical implementation that employ a single NIR camera and SFC sensing circuits with Vive controllers.
- This work's second contribution is the case that, with proposed camera sensor attached on the controller, this study provides a new finger posture dataset from 20 different subjects.
- The third of this work's contribution is that findings are reported in prototyping process and this work evaluates the accuracy and real-time performance of the proof-of-concept prototypes.
- Last, the practical feasibility with this work's approaches are demonstrated by two developed Unity demo VR applications in real-time.

Chapter 2

Related Work

This work explores two methods to increase inputs for controller devices using gesture sensing technologies. One is the SFC sensing method and the other is a CNN-based gesture classification. In this section, related work is reviewed on (i) hand gesture sensing approaches and (ii) around device interaction, (iii) gesture tracking/recognition through augmenting VR HMD systems (iv) literature of wearable cameras, (v) fingers' freedom for on-device interaction and also introduce (vi) gesture recognition for SFC sensor and (vii) vision methods for hand gesture classification

2.1 Hand Gesture Sensing Techniques

There are different sensing technologies to enable hand gesture recognition. For example, capacitive sensing, it can enable touch sensing and can also determine the shape of grasp on instrumented objects or devices [1] (see a review in [2]). Recent work [3] explored to use ultrasound imaging for hand gesture recognition. They presented an approach for detecting discrete gestures and tracking continuous fingers' angles through ultrasound images captured from a probe mounted on the forearm. There are also combined sensing approaches to implement a hand gesture classification. [4] explored the combination of Electromyography (EMG) and pressure. They found that the combination of EMG and pressure data, which are only sensed at the wrist, can support the accurate classification of gestures. Data glove can obtain robust finger movements using various sensor technologies such as magnetic tracking or inertial tracking [5]. Optical methods are based on employing cameras to determine hand position, which occupies most of the relevant literature on gesture recognition. A review of the methods can be found in [6]'s work.

2.2 Hand Gesture Interaction Through Wearable Cameras

Based on vision methods, to enable a variety of interaction scenarios, body-worn camera system has been extensively studied, camera can be mounted on different body positions, such as heads, chests, shoulders, foot, wrists and fingers [7, 8, 9, 10, 11, 12, 13]. [7] present a HMD mounted device, which includes a time-of-flight (TOF) module with a standard RGB camera, supporting single-handed gestural interaction. [8] propose a single chest-mounted fish-eye camera device, imaging the user's body posture through a self-centered user view. [9] introduce a shoulder-worn system using a wearable depth-sensing and a projection device. The system allows wearers to use their hands, arms and legs as graphical interactive surfaces. [10] developed a wearable system consisting in part of a shoe-mounted depth sensor pointing upward at the wearer. The system

provides three recognized gestures for which can be performed without visual attention. While [11] first presented a wrist-worn sensor that recovers the full 3D pose of the user's hand, [12] developed a wrist-worn gesture input device specially designed for symbolic input in VR. Their prototype is able to robustly sense thumb-to-finger taps on fingertips and minor knuckles. To increase the hand-based and context-aware interactions in the environmental contexts, [13] showed a finger-worn device. It allows users to interact with specific environment by gesture input. This work's prototype implementation with a Vive controller is close to the approach that attaches a camera worn on wrist.

2.3 Gesture Tracking/Recognition through Augmenting VR HMD systems

When specifically designed to interact with VR HMD systems, camera can also be instrumented to existing VR headsets. [6] attach a lightweight stereo pair of fish-eye cameras onto a VR headset, tracking boy movements as a motion capture system. Their approach allows full-body motion to capture general indoor and outdoor scenes, including crowded scenes with many people nearby, which allows for reconstruction on larger scales. [14] develop a new HMD that enables 3D facial performance-driven animation in real-time, by combining a head-mounted depth camera and surface strain signals. Also, [15] use a monocular camera attached to a HMD, performing various facial expressions. Like these works that aim to enrich the experience of being immersed in VR, this work first augments an existing VR controller with a prototype to provide a better model for what the hands are actually doing while interacting with VR content.

2.4 Around device interaction

This work's finger gesture control in air is similar to around device interaction, which is also known as gesture interaction with mobile devices. It is focused on increasing the interaction space for the devices rather than only relying on contact with touch screen. To enable this interaction technique, beyond camera-based methods [16], there are also viable alternatives. Early methods use infrared proximity sensors to let users interact with devices around the edge or above the display [17, 18]. Magnetic sensing can detect postures and finger positions by wearable sensors like ring devices. Recent work on capacitive proximity sensing also investigated 3D gestures input around the electrodes' plane [19] or above the surface [20].

2.5 Fingers' freedom for on-device interaction

To perform finger tracking when holding the controller, it is necessary to analyze the fingers' movements that do not require a grip change. Considering the grip gesture is similar to holing a smartphone device, which is most related to one-handed interaction, which is particularly for mobile phone devices. [21] modeled the functional area of the thumb on mobile touchscreen surfaces. [22] conducted qualitative experiments to evaluate the index finger area of movements on the back of the device (BoD). Most recent work [23] use a quantitative method to empirically study all the fingers' range and comfortable area for one-handed smartphone interaction beyond the touchscreen. They derive generalizable design implications for the placement of on-device input. Totally, pre-

vious work showed that thumb and index finger have more movements so intentional input control using the fingers in one-hand interaction is proven effective. As for Vive and Touch controllers, the grip gesture is similar to holding a smartphone, so the input control design is also expected to should focus on these fingers.

2.6 Swept Frequency Capacitive Sensing (SFCS)

Capacitive sensing has been as a significant user interface in HCI community (see [2] for a survey). In this research field, SFCS is a special sensing method which first proposed and named by Sato's work Touché [24]. Touché uses a range of frequencies as a profile for different touch gestures, while conventional capacitive touch sensing method uses only a single operating frequency. Sato's team also utilize this SFCS method to explore the differentiation from impedance of different users using Touché prototype on touch screens [25], while [26] embed Touché into plants to explore botanical interaction on entertainment and aesthetic uses. Intrigued by Sato's work, Mads Hobye [1], Nikolaj Mobius [2] first employed SFCS on Arduino platform. [27] embed a signal generator module into an Arduino Uno as a human-to-human interaction touch sensor for gaming apparatus.

However, no previous work report further performance for SFC sensor matrices. Original Touché [24] project focused results and discussion on a single electrode solution. It is reported that SFC sensor matrices would bring more unique sensing dimensions described in their paper. But the work utilized a customized sensor device and it is difficult to continue from the original Touché project. [28] provide an open source library of for Arduino micro-controller. They used it for their instrumental plant application which is called Cultivating Frequencies [3]. The contributed SweepingCapSense [4] library is used to turn each of their fifteen plants into touch sensors. But importantly, they did not provide more implementation details about the plant application. It seems that they did not construct SFC sensor matrices based on the survey knowledge from the source code of the library and case study they provided.

This study selects the SFCS technology because it can provide richer touch information than traditional capacitive sensing technology. This study shows an exploratory implementation on constructing the SFC sensor matrix. This work utilizes the sensors with a Vive VR controller to explore its practicality in actual implementation.

2.7 Vision Methods for Hand Gesture Classification

Vision-based hand gesture recognition, has been employed with researchers' efforts by early work. Optical methods are based on employing cameras to determine hand position. These approaches can analyze the hand shape and motion information from captured video sequences by one or several cameras. Traditional methods based on machine learning like Support Vector Machine (SVM), Hidden Markov Model (HMM) are also reviewed in [6]'s work.

[1] http://www.hobye.dk
[2] http://dzlsevilgeniuslair.blogspot.se
[3] https://colinhonigman.com/Cultivating-Frequencies
[4] https://github.com/chonigman/SweepingCapSense

But after CNN showed its power for visual classification task of objects in images and videos [29], it has also been employed to solve the problem of hand gesture recognition. Besides the works that recover the full hand pose using CNNs [30, 31, 32], the CNN classifier is also often used in tasks of sign language recognition [33, 34, 35]. To improve the real-time performance of the CNN classifier for mobile applications, lightweight models that decrease computational cost such as Distilling [36], ShuffleNet [37], MobileNets [38] and MobileNetV2 [39] are recently attracting attention. These network models mainly use model compression technologies to reduce the number of parameters used in deep learning to support some low-configuration equipment or mobile device conditions.

Chapter 3

SFC Sensing Approach

3.1 Constructing SFC Sensor Matrices

When considering the construction of sensor matrices, there are two methods. One is to simply add micro-controllers and circuits, and then attach each electrode into or onto objects. The other is to add only circuits to make the electrodes independent of each other in software processing. In order to achieve finger tracking when holding the controller, this work's task is to attach the sensor electrodes to the Vive controller. Because of the design of vive controller and the limitation of finger movement, this study constructs a two-dimension (2D) sensor matrix. Two dimensions correspond to thumb and middle finger. The circuit details can be referred to [28].

3.2 Embedding Electrodes Into Vive Controller

After building the circuits, the electrodes is attached to the aluminum foil and then fix them on the surface of the controller where touched by fingers. In the evaluation section of previous work [28], the authors embedded the sensing copper electrode into a small rosemary plant (The electrode is placed in soil). They report that SFCS technique is not only sensitive to gesture but to the plant's position as well. This is because that different touch positions response different capacitive profiles (the resulting 2D curve). But for this work's task, the aluminum foil is small and is with no changing capacitance. So, extra tape sheets are added onto the foil to changing the capacitance, which resulting in a changed result curve when just touching the green tape. The prototype is shown in Figure 3.1. This work's SFC sensor consists of 2 circuits (Left, Upper) which can provide 2 sensing electrodes to detect touch information. Using the tapes (b, c) and aluminum foil (a), these electrodes are attached on the surface (Right) of the Vive controller in this work's experimental implementation. It is noted again that touching the aluminum foil and touching only the green tape are different touch events, which is because the capacitive profile [24] is different.

3.3 Recognizing Two Different Touch Events from One Electrode

Using one electrode from the SFC sensor, this study enables two kinds of touch events: Touch the aluminum foil (Touch Alumi) and Touch only the green tape (Touch Tape), shown as Figure 3.2. Besides not touching the sensing parts (a, No Touch), the sensor can distinguish if one is touching only the green tape (b, Touch Tape) or is touching the aluminum foil, no matter whether the finger is touching the tape (c, Touch Alumi). Designing any SFCS solutions is considered as a sampling

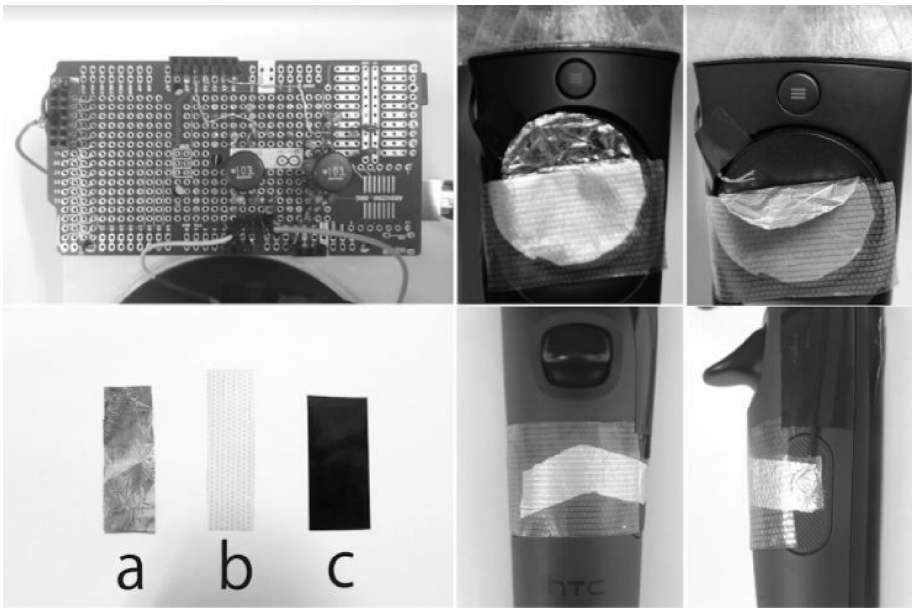


Figure 3.1: Main components of this work's SFC sensor matrix.

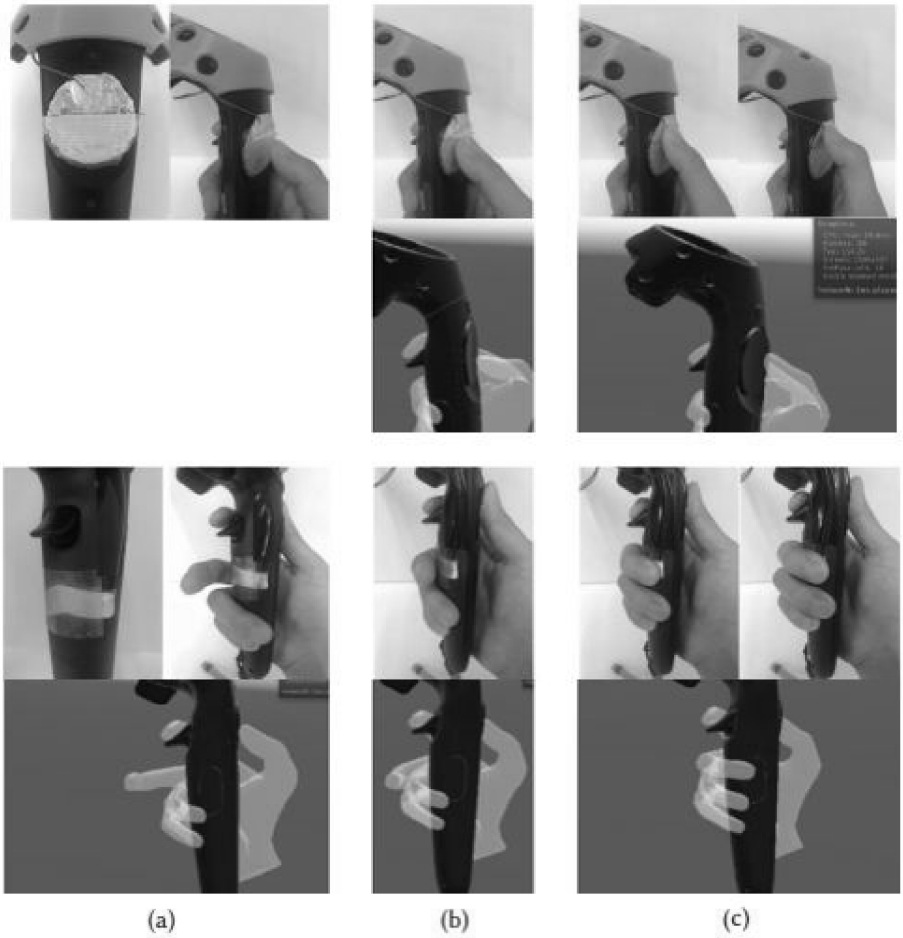


Figure 3.2: Finger gestures recognized by the SFC sensor.

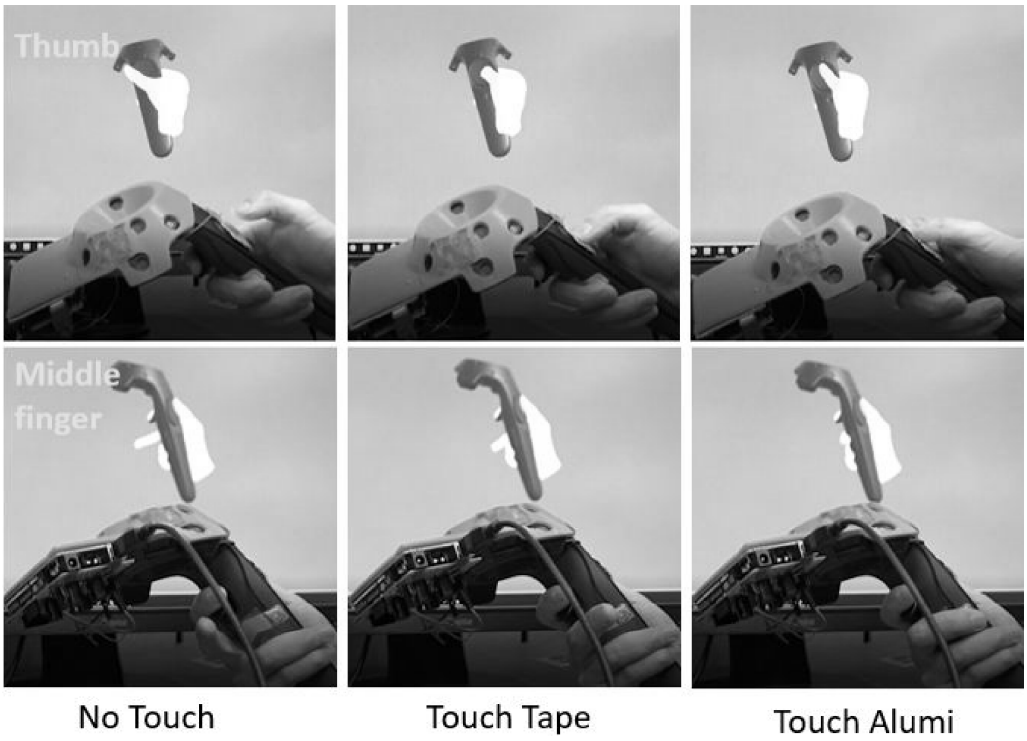


Figure 3.3: The finger tracking demo developed in Unity.

problem [24]. It depends on how many samples and what frequency bands would allow the system to accurately identify the different states. The program of sample the response signal every 160 steps. In this work's solution, for each SFCS circuit, the resulting curve are sampled between the fiftieth and eightieth step. This make us enable the recognition processing at a latency about 100ms in Unity. In actual operation, the frame rate of the system can reach 10 to 15 fps. A finger tracking demo is created in Unity using the SFC sensor (See Figure 3.3). It should be noted that even this work's prototype is now applied to a Vive controller, this study did not enable the capacitive sensing components of the Vive controller for software implementation such as the circle Touch panel.

3.4 Discussion of Limitations

For this work's current SFC sensor implementation , it needs a preprocessing of data record for a user, which is to record a 2D graph data. This study improves runtime performance by decreasing the sweep resolution, reducing the robustness of gesture recognition. For example, to achieve robust recognition it must be costumed for different users because the different capacity of human. But in controlled experimental settings, this study achieved above 90% accuracy for the touch gesture triggered with one subject from early tests. The accuracy is expected to improve by increasing the sweep range of frequencies. Machine learning method is also expected to improve the generalization among different subjects.

Another notable limitation is the unreliability for multi-touch. In early tests, the implementation of sensor matrices with 2, 3, and 4 respective circuits are independently tried. The response sig-

nals are more stable when using the 2-circuits method, which means it support multi-touch for two fingers more reliably. The reason can be referred to [24]. They discussed, *"The amount of signal change depends on a variety of factors. It is affected by how a person touches the electrode, e.g., the surface area of skin touching the electrode. It is affected by the body's connection to ground, e.g., wearing or not wearing shoes or having one or both feet on the ground.*. For the same reason, some experimental situation can also affect this work's sensing prototype. For example, the resulting graph data is unreliable when charging the battery of the Vive controller, and the capacitance is also changed when a user lift up his or her legs because of the changed capacitive path (not connected to the ground). At the same time, because SFCS is a capacitive sensing method, when it is very close to the touchpad without touching, it will cause the recognition error of the system. That is, there is no touch but the touch event is recognized. This is also a problem on the capacitance button of the Oculus Touch Controller.

Chapter 4

NIR Camera Sensing Approach

This study are motivated by impressive development on visual recognition tasks recent years. This study explores to enrich the input capabilities through the finger postures using a vision method, especially for room-scale VR environments. When first considering tracking finger movements, using depth cameras to achieve high recognition rates is also thought about. But eventually a traditional webcam is chosen to be modified into a NIR camera, which is informed by [40]'s work. While they explored how the NIR signal and controlled illumination can be used for depth sensing, this study applies similar conditions to accomplish a gesture classification task for close-range. There are three reasons why a depth camera is not chosen. First, considering that best effort should be tried to make this work's prototype compact and user-friendly, a large-sized depth camera is excluded, such as Kinect. Secondly, finger movements are needed to be analyzed at very close distances (approximately between 0.04 and 0.09 meters). So, although a smaller RealSense [1] seems to be more suitable, its depth perception limit is 0.11 meters, and it seems not to perform satisfactorily even if it is very close to 0.11 meters. Finally, it is about real-time performance. Using a depth camera increases the computational burden of the computer, so it is difficult to achieve real-time at an application level.

In addition to the benefits of using NIR reported in [40], under this work's execution conditions, selecting the use of a NIR camera also brings us three benefits. First, NIR provides less intrusive sensing than the visible spectrum when using a traditional webcam. This means that this work's prototype equipment can also be used at night or without indoor lighting. Second, under the controlled illumination, the background of the input image become not obvious even seems to be removed, which makes us be able try not to consider complex finger segmentation processing. Finally, This work's webcam costs about only 12 dollars which is not expensive. Even with the cost of modification parts needed and 3D print required, the cost of this work's prototype equipment does not exceed 32 dollars. The details of the camera sensor are give in later section.

4.1 Camera Setup

The prototype overview is shown as Figure 4.1. Main components of the camera sensor is shown as Figure 4.2. A regular webcam (a) is modified into a NIR camera by adding a bandpass filter (c), and a wide-angle lens (b) with the LEDs ring (f) with 18 LEDs. The 3D printed ceases (d,e) are used to embed the LEDs and attach to the wide-angle lens. The webcam is Logitch HD Webcam

[1] https://realsense.intel.com/depth-camera/#D415_D435

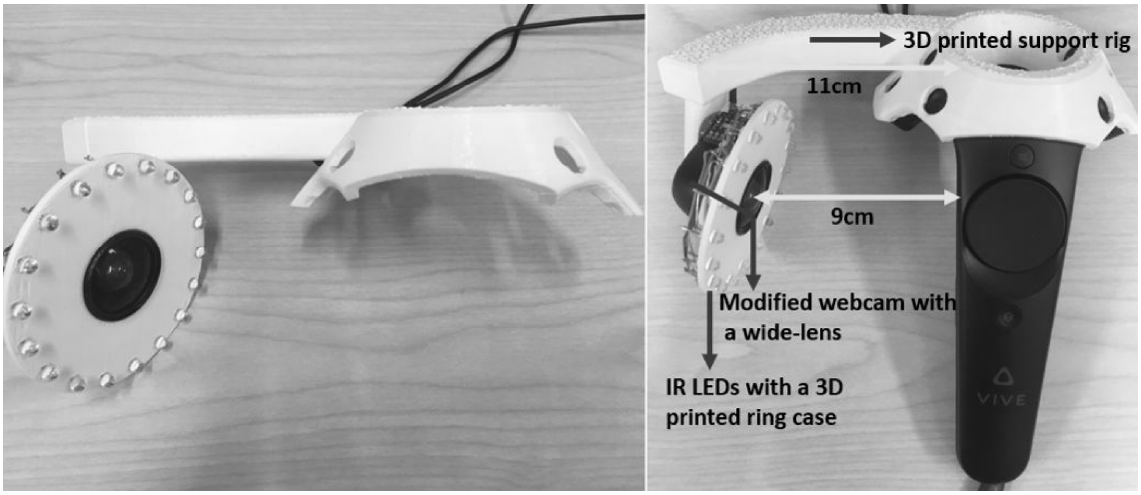


Figure 4.1: The prototype overview for the NIR camera sensing approach.

C270 [2], capturing at 30fps with a resolution of 640 × 480 pixels. The camera is mounted with a 0.4X wide-angle lens [3], so the field of view is changed from 60° to 150°, making the camera a sensor of close-range hand capture that keeps the whole hand in the camera view. The original camera does not contain an IR cut filter, so an IR bandpass filter [4] operating at 850nm is just added, and a ring of LEDs is then added to provide evenness illumination. The LEDs [5] used in this work's system have a beam angle of ±15 degree with 80% attenuation at the periphery, again operating at 850nm. The support rig is 3D printed to make the camera operate at a proper and fixed distance and angle. The angle and position is first adapt manually, then the configuration in the 3D model is changed . But currently, just some seccotineuse is used to fix the camera with the 3D prints.

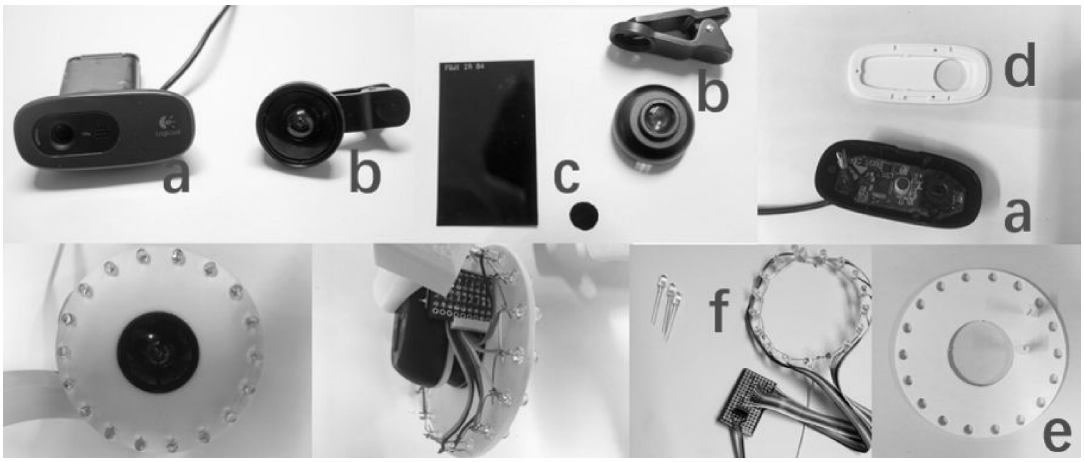


Figure 4.2: Main components of the NIR camera sensor.

[2] http://support.logitech.com/en_us/product/hd-webcam-c270
[3] https://www.taotronics.com/TT-SH014-Mobile-Phone-Camera-Lens-Kit.html
[4] http://fujifilm.jp/personal/filmandcamera/sheetfilter/ir.html
[5] http://pdf1.alldatasheet.jp/datasheet-pdf/view/635246/OPTOSUPPLY/OSI3CA5111A.html

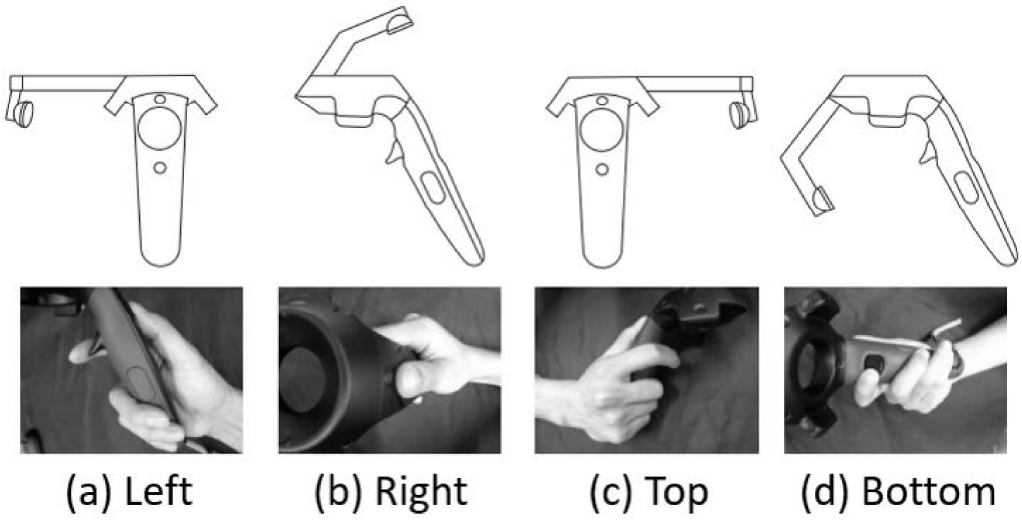


Figure 4.3: Four approaches to mount a single camera onto the controller.

4.2 Camera's Placement

This is a first exploratory work that how to determine the placement of the camera. There are mainly 4 different positions to mount a single camera onto the Vive controller, seen as Figure 4.3. However, neither approach (b) nor approach (d) can obtain a full hand (including all fingers) image. Approach (c) is a position of mirror symmetry to approach (a), which is from the back hand of view that cannot distinguish small movements of middle and ring fingers The NIR camera is selected to attach on the left for the right-hand controller, because this way seems to be able provide more finger motion information than others. Considering the design of a controller, there is an angle and position to setup the camera so that all fingers can be imaged by the modified camera. This allows us to design as many as finger postures. The insights about camera's positions are discussed in the later discussion chapter.

4.3 Design Gestures

The following observational analysis of the movement range of each finger has been made when holding the controller. When a user naturally and correctly holds the Vive controller, the individual finger freedom of his or her hands is physically constrained. The main motion range of each finger is illustrated as Figure 4.4. Besides touching the touch panel, the thumb mainly moves horizontally and vertically (a). The index finger mainly touches the trigger, moves close to or far away from the trigger (b). The middle finger mainly moves close to or far away the controller surface, including touching the controller (c). For the ring finger and small finger, both of them need to help holding the controller handle (d), but the ring finger's fingertip have movements that touch or not touch the controller. As an exploratory work, this study seeks to find whether the CNN classifier can recognize minor changes among different direction, hight, distance of individual fingers, especially when fingers are in air. So 12 different finger gestures are designed and showed as Figure 4.5

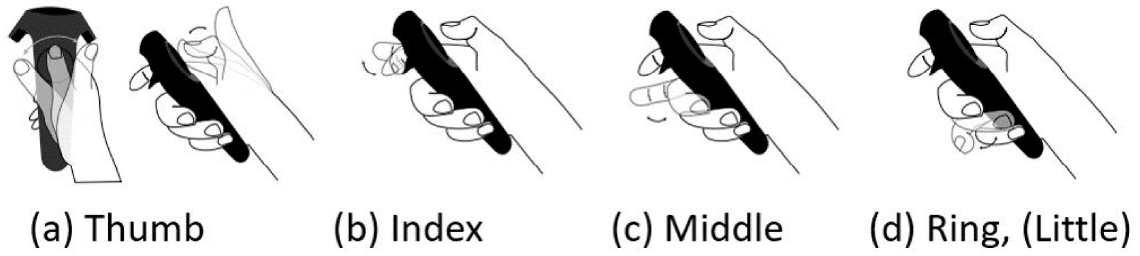


Figure 4.4: The main motion range of fingers.

4.4 Gesture Recognition Pipeline

To perform gesture recognition, users firstly have to learn to naturally and correctly hold the controller. Figure 4.6 is an example when a user is trying the controller. The size of input images captured by the NIR camera are 640×480 at a frame rate of 30 fps. Then the region of fingers are segmented, and the images are resized to 32×32 to pass to the deep learning model.

4.4.1 Finger Segmentation

In traditional gesture recognition tasks, hand tracking and hand segmentation are important parts before recognition. Under this work's conditions, since users hold the controller most of the time, hand tracking relies on the tracking devices of HTC VR system. For finger segmentation, the advantage of fixing the camera on the controller is used, that is, the overall position of the hand image is relatively unchanged. Therefore, the position relationship of the pixels are used to directly segment the image of each finger part. Figure 4.7 illustrates this work's system processing pipeline. The rectangle boxes with different colors are like bounding boxes, but actually the finger images are just segmented through the pixel position because the positional relation of camera and hand is fixed.

4.4.2 CNN classifier

This work's aim is to perform a real-time gesture recognition system in real VR applications. So this needs us to select a model that does not require high computational resources. In the developed Unity demo, TensorFlowSharp library [6] is used but that does not support a GPU-based version, so currently a LeNet-5 [41] based CNN model is implemented. Shown as Table 4.1, the CNN model is informed from the Chainer [7] library by the early test. There is a small change in C3 layer compared to the initial LeNet-5 model, though there are not obvious improvement in accuracy. About 27fps are achieved in the Unity environment on a consumer computer with this model. The comparison results with a state-of-the-art CNN classifier MobileNetV2 [39] are also reported in the later evaluation chapter.

4.4.3 Training

When training the model, the final dataset is used, which is after the data augmentation. All the dataset images are converted into gray scale images and resized to a resolution of 32×32 pixels.

[6] https://github.com/migueldeicaza/TensorFlowSharp
[7] https://docs.chainer.org/en/stable/examples/cnn.html

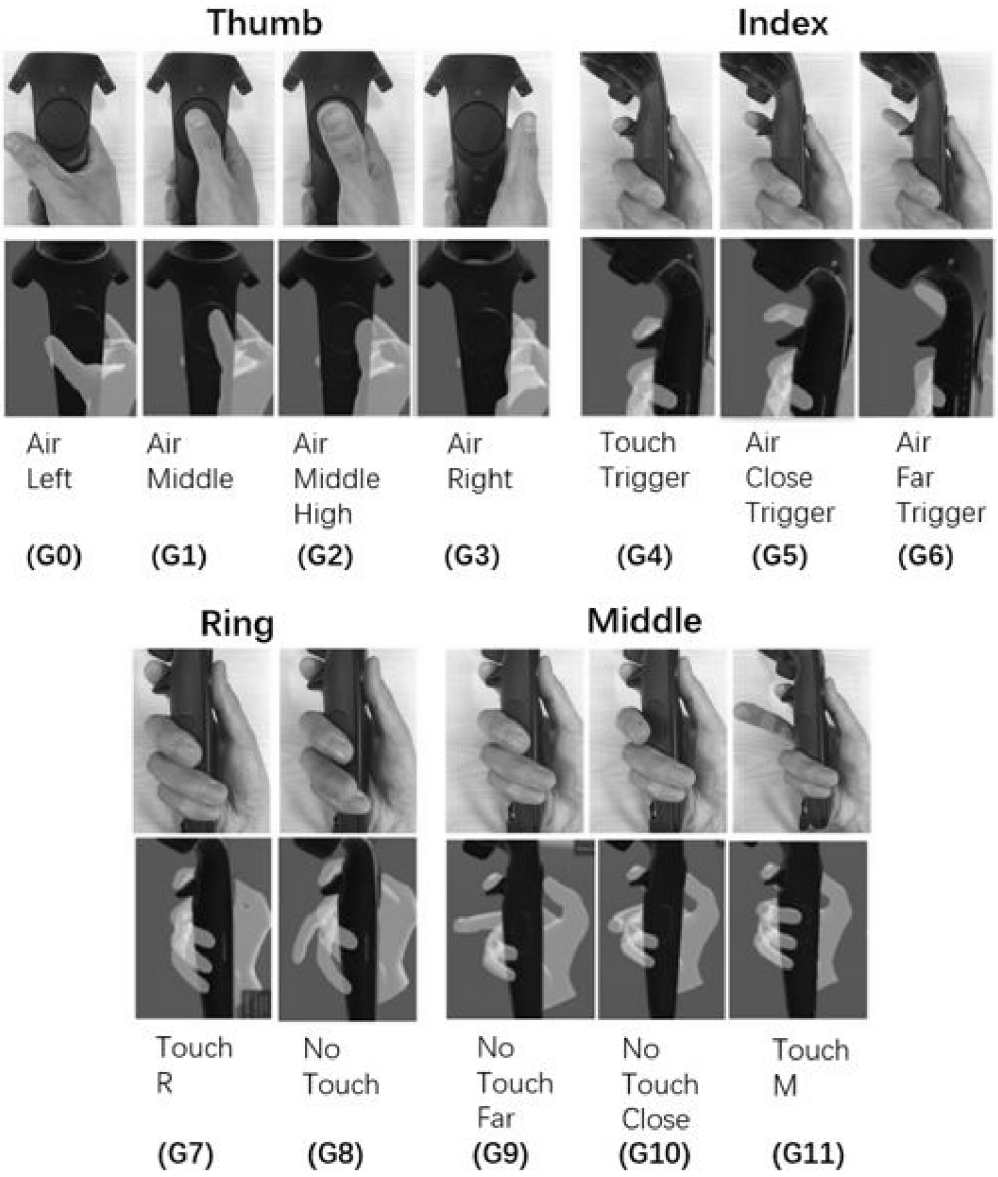


Figure 4.5: Finger postures recognized by this work's classifier applied to the NIR camera sensor

With this big dataset, a dropout rate of 0.6 and learning rate decay are used to make the model learn more robust features. The model is trained on the dataset with 20 iterations. Finally, an average 95.6% accuracy is achieved on the training set, and 93.9% on the test set with this work's model using the dataset collected by Right-hand prototype (middle) in Figure 5.1.

Figure 4.6: An example scene when a user is trying this work's prototype with the Vive controller.

Table 4.1: The architecture of the CNN model.

Layer	Layer Type	Kernel	Maps
Input	input		$1 \times (32 \times 32)$
C1	convolution	5×5	$6 \times (28 \times 28)$
M1	max pooling	2×2	$6 \times (14 \times 14)$
C2	convolution	5×5	$16 \times (10 \times 10)$
M2	max pooling	2×2	$16 \times (5 \times 5)$
C3	convolution	4×4	$120 \times (2 \times 2)$
Output	full connection		$12 \times (1 \times 1)$

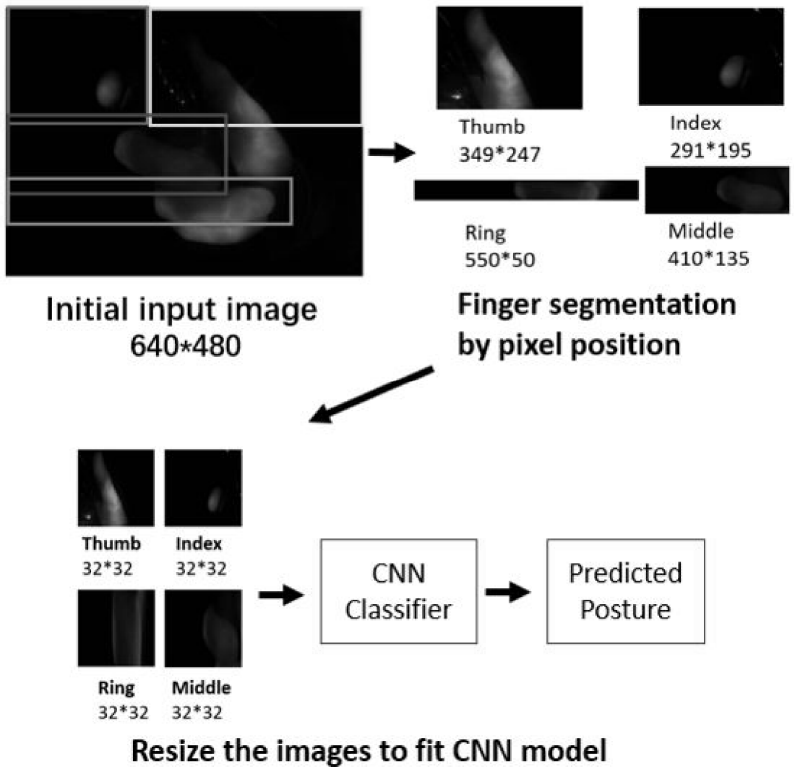


Figure 4.7: Data processing pipeline of the camera sensor.

Chapter 5

Evaluation

For the SFCS approach, this work's contribution is focused on building a sensor matrix to increase the touch input when holding the VR controller. But just as mentioned in the previous chapter about the limitations of this work's current prototypes and the limitations of the SFCS method itself, especially the problem of multi-touch limits us from making more in-depth evaluations. So in this chapter, the evaluation of NIR camera sensing approach is focused on.

5.1 Experimental Settings

All the experiments are conducted on one Windows 10 desktop with a CPU processor (Intel Core i7-7700 3.6 GHz), 48GB SDRAM and a NVIDIA GeForce GTX 1080 Ti GPU with 11GB of memory. The models are implemented using the Python libraries TensorFlow. With a dropout operation in this work's model and to achieve more objective results, all the trainings are conducted 3 times and the average statistical results are showed. Because of the long time of the training cost, tests on MobileNetV2 are all conducted with the initial images that are without the data augmentation.

5.2 Data Collection

5.2.1 Prototypes

In this work's implementation, all the datasets are collected using this work's early test prototype, the middle one in Figure 5.1. Besides the different weight, there are minor variances in fixed camera positions and angles. The evaluation results of generalization across these prototypes are also reported later.

5.2.2 Participants

The dataset is collected from 20 different subjects (13 males and 7 female) who are from this work's laboratory members and graduate students in this work's university. They are aged from 19 to 26. The subjects are with widely varying height (mean = 169.6 cm, std = 9.85 cm), increasing the variance in palm size. The length of their hands from the tip of the middle finger to the wrist were recorded (mean = 18.23 cm, std = 1.89 cm). All participants were right handed.

Figure 5.1: Different prototypes tested.

5.2.3 Procedures

When collecting the dataset, the subjects are seated all the time and they are first asked to learn how to correctly hold the controller, then they are asked to perform the 12 gestures in 12 trials. The subjects are asked to move and rotate the controller in every direction to add variance in backgrounds and finger position, which is also seen to simulate a way when one plays VR content in real. In the room-scale VR system, the dataset is collected indoor both during the day and in the evening. So the infrared light from fluorescent lamp and the sunlight through the window also increased the variance in the image backgrounds.

5.2.4 Preprocessing

The cropped images are directly captured in a Unity application, which are across the 12 finger gestures from the initial input image. Frames are extracted at about 10fps from real-time video and collect the training images and test images individually. The number of train images for each gesture of each subject is 1000, while the test image part is 300. This means the training set and test set consists of 1000×12 images and 300×12 images separately. So this work's initial dataset includes a training set of 240,000 images and a test set of 72,000 images.

5.2.5 Data augmentation

In order to achieve a robust recognition result and avoid over-fitting, each image is randomly rotated from -20^{c}irc to +20^{c}irc 5 times, and randomly adjusted the brightness 3 times, and randomly adjusted contrast 3 times; thus, the number of final train images in the database reaches $1000 \times 12 \times 20 \times 13$=3, 120, 000, and the test set reached to $300 \times 12 \times 20 \times 13$=936, 000 images.

5.3 Technical Evaluation

Results of a technical evaluation with this work's camera-based system are reported in this chapter . These include (i) The leave-one-subject-out across-subject validation result and the confusion matrix of the classifier, (ii) A comparison with MobileNetV2 in leave-one-subject validation and the real-time performance, (iii) An initial generalization test among different test prototypes that shown in Figure 5.1

Table 5.1: Leave-one-subject-out across-validation accuracy results using this work's CNN model.

	Thumb				Index			Ring		Middle			
S	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	M
1	98.93	73.77	100.0	99.58	99.95	50.60	95.82	97.94	88.81	98.66	72.42	99.98	89.71
2	99.20	83.69	87.33	91.77	99.89	69.38	97.77	97.56	78.47	78.47	79.76	87.76	87.59
3	99.31	95.87	93.99	99.99	99.75	94.91	89.02	99.13	93.27	91.91	94.91	99.98	96.00
4	98.47	66.34	99.13	85.57	94.01	41.67	96.78	93.03	94.21	83.47	90.25	99.57	86.88
5	97.74	94.77	99.63	99.07	99.37	96.91	73.18	77.75	66.92	96.03	90.36	99.99	90.98
6	99.24	32.58	92.05	99.77	97.40	01.97	97.86	57.24	68.92	72.35	69.74	69.42	71.55
7	65.87	73.81	29.88	74.35	87.81	83.70	74.13	77.74	43.52	87.62	92.79	99.96	74.26
8	92.18	71.08	85.32	99.76	93.85	93.72	86.26	86.44	90.50	85.83	56.64	93.35	86.24
9	92.44	93.71	86.33	99.98	97.47	88.22	73.17	97.90	62.69	96.56	43.68	70.71	83.57
10	99.69	91.91	84.33	94.39	92.68	90.83	94.14	50.76	94.40	99.06	73.16	95.11	88.37
11	87.87	99.48	77.97	93.02	99.49	68.13	88.67	94.90	95.78	91.05	32.32	85.29	84.50
12	99.44	99.36	94.68	95.31	99.61	90.71	85.81	83.61	85.24	94.97	93.52	66.84	90.76
13	96.50	96.89	98.94	95.08	78.33	71.41	89.72	69.84	99.08	96.17	73.94	99.31	88.77
14	96.03	96.63	97.28	99.55	99.92	99.20	54.13	88.79	99.75	97.75	92.76	95.47	93.10
15	99.92	99.41	97.75	100.0	98.05	75.35	68.37	78.43	83.76	95.42	89.81	89.89	89.68
16	97.59	93.60	92.45	94.24	97.08	94.66	92.29	86.08	96.98	96.61	93.93	84.11	93.30
17	79.98	81.34	92.36	99.62	95.18	95.19	97.00	87.49	97.29	76.05	61.29	54.36	84.76
18	96.10	97.61	90.32	68.77	97.93	83.97	87.32	43.53	89.09	80.48	51.35	96.16	81.89
19	86.99	55.95	86.66	91.13	98.98	98.26	90.48	15.80	74.26	65.07	39.35	99.43	75.19
20	98.07	89.53	72.54	63.15	99.85	92.41	99.13	95.25	54.41	82.91	90.94	97.79	86.33
M	**94.08**	84.37	87.95	**92.21**	**96.33**	79.06	86.55	78.96	82.87	88.32	74.15	**89.22**	**86.17**

*Note: "S" means Subject. "G" means Gesture. "M" means "Mean".

5.3.1 Generalization across different subjects

Shown as Table 5.1, this work's method (LeNet-5 based classifier) achieved 86.17% average accuracy in the across-subject test. The top-3 accuracy are all more than 92% which are Air Left (94.08%), Air Right (92.21%) of thumb postures, Touch Trigger (96.33%) of index finger postures. And the fourth high No Touch Far (89.22%) of middle finger postures.

5.3.2 Misclassification

The confusion matrix of this work's classifier is shown as Figure 5.2. The confusion matrix shows that minor differences between different heights in air (thumb), and different distances (index finger and middle finger) are often misclassified. There are some outliers with a lower recognition rate (< 50%) that the number text is made red in Table 5.1. In the index postures, Air Close Trigger of subject6 (1.97%) , it is supposed that this is because subject6 did not correctly perform the gesture, or he did not perform the difference between Air Close Trigger and Air Far Trigger, and the same reason as Air Close Trigger of subject4 (41.67%). After confirming the images and the confusion matrix of subject6 and subject4, a founding is that the classifier almost misclassified Air Close Trigger as Air Far Trigger. It is noted that all the subjects are asked to move and rotate the controller in every direction while keeping the finger posture immobile during the collection time, which is actually hard for them. Because they were moving and rotating the controller when performing the postures, so sometimes they had to make mistakes in performing different heights (high or low), and distances (far or close).

In the thumb postures, For Air Middle of subject6 (32.58%) and Air Middle High of subject7 (29.88%), the difference between the two thumb postures is only the height (Air Middle High is

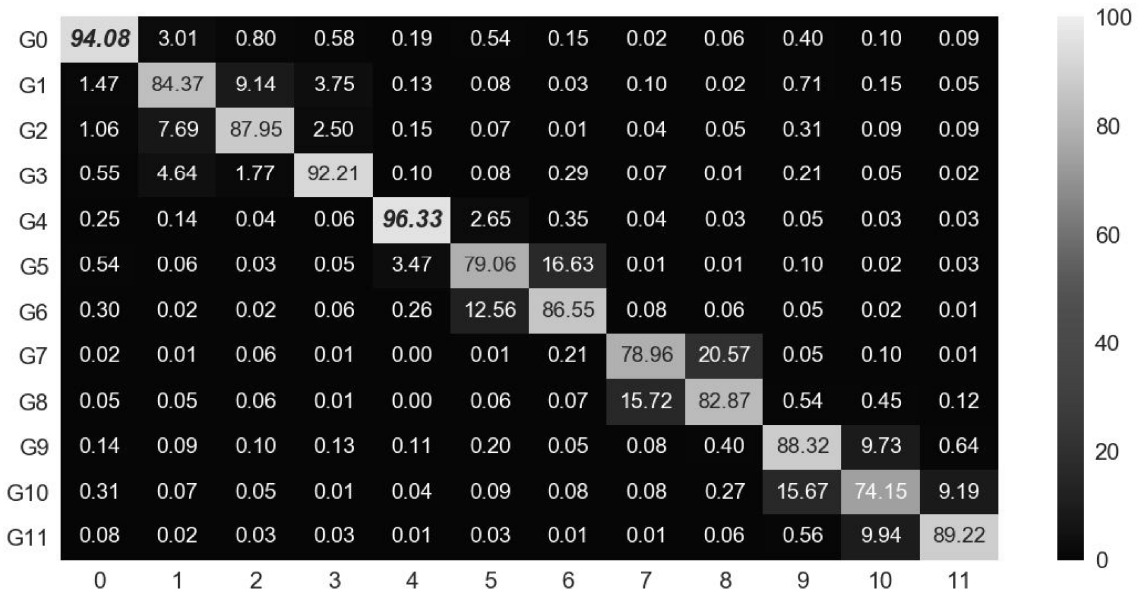


Figure 5.2: Confusion matrix that corresponds to the result of the leave-one-subject-out test. X-axis: Prediction, Y-axis: Actual.

more like a "Good job" posture). Some subjects reported that it was hard for them to perform the difference.

However, the outliers are also related to the hand size and the grip strength of subjects. In the ring finger and middle finger postures, some subjects whose hands are too small even were not able to perform the same posture (Touch R of ring finger) as others such as subject19 (152cm of height and 15cm of hand size). And it was still difficult to perform the small difference between the two postures, for example, some subjects can leave their ring fingers very far when performing No Touch, while another some subjects can not. Outliers in postures of middle finger are thought to be the same reason.

As a summary, the results show that subjects whose thumb fingers are straight (shape of thumb of a few subjects is not very straight) resulted in a better recognition in thumb postures. Results of subjects that have strong grip strength or subjects with a big hand size seem to achieve higher recognition rates.

5.3.3 Comparison with MobileNetV2

MobileNetV2 is a state-of-the-art CNN network specifically tailored for mobile and resource constrained environments. In this work's test, the specific model of Python code is informed from github open source community [1]. The real-time performance is tested in TensorFLowSharp libraries in Unity. The accuracy results are shown in Table 5.2, it shows that with the data augmentation, accuracy is improved by about 2.29% (from 83.88% to 86.17%). The MobileNetV2 can classified the postures with higher accuracy (89.38% than 86.17%), and the top-4 accuracy results are also improved obviously (more than 93%). For a more detailed comparison of the results,

[1] https://github.com/neuleaf/MobileNetV2

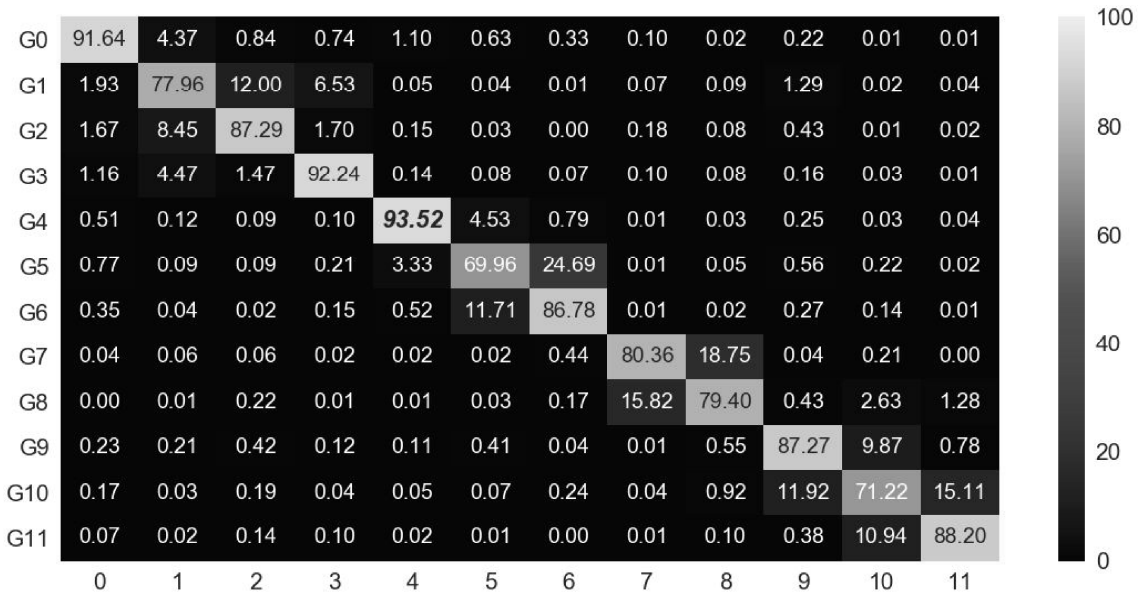


Figure 5.3: Confusion matrix for this work's model without data augmentation.

reader can refer to the confusion matrices in Figure 5.3 and Figure 5.4. Remember all the results are tested with the leave-one-subject-out technique, and the X-axis is for prediction label while Y-axis is for actual label.

The real-time performance and recognition test results are shown as Table 5.3. Because the current TensorFLowSharp liabraries only support the CPU version of TensorFlow, so the time test are mainly all from CPU processing. But early test in Python libraries on GPU did not show any improvements in recognition time (about 4ms of this work's model and 836ms of MobileNetV2 model). The recognition cost does not occupy many computer resources, but it is the procedure of transforming from inputs to tensors of TensorFlow that takes much time to process (OpenCVSharp and Unity libraries are used to solve this). The Prediction function is operated 4 times on each frame, which is because the input images are just cropped to 4 input images to classifier for the 4 fingers separately. So the 27fps is mainly resulted from ($6ms + 1.4ms + 2ms$ + other processing) $\times 4$ which is equal to a latency about 37ms. Future optimization may improve the real-time performance by implementing the recognition pipeline using C/C++ then call dynamic link library (DDL) files in real-time. Maybe the GPU support of the future version of TensorFlowSharp can also allow us to use MobilNets [38] to improve the recognition rate through GPU.

Table 5.2: Results compared to MobileNetV2.

	Thumb				Index			Ring		Middle			
	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	Mean
Ours no DA	91.64	77.96	87.29	92.24	93.52	69.96	86.78	81.12	79.40	87.27	71.22	88.20	83.88
Ours DA*	**94.08**	84.37	87.95	**92.24**	**96.33**	79.06	86.55	78.96	82.87	88.32	74.15	**89.22**	**86.17**
MobileNetV2 no DA	98.64	89.34	89.09	95.38	97.07	83.61	92.46	81.68	91.36	89.37	70.95	93.66	89.38

* "DA" means the training is implemented with Data Augmentation.

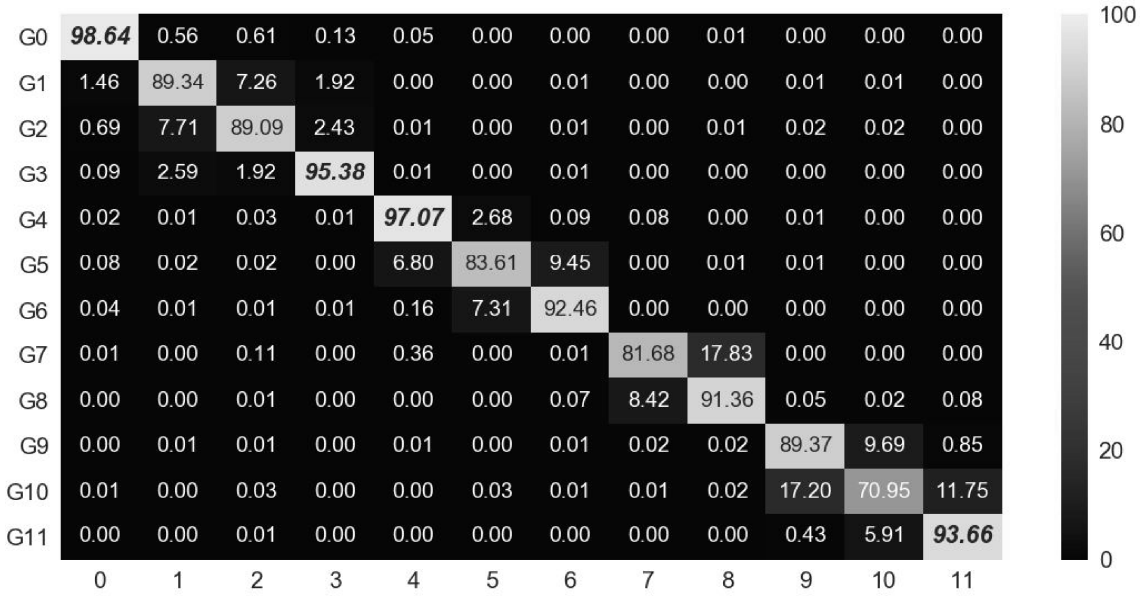


Figure 5.4: Confusion matrix for MobileNetV2 without data augmentation.

Table 5.3: Real-time test between this work's model and MobileNetV2 model.

	Frame rate	Recognition time	Time to transform input to tensorr
Ours	27 ~ 30fps	1.4ms	5 ~ 6ms
MobileNetV2	1fps	251.2ms	5 ~ 6ms

5.3.4 Generalization across different prototypes

The camera currently is just attached using seccotine, so there are variances in distance or angle of the mounted camera. And for a left hand controller, there are also variances between the fingers of one's left hand and right hand. The grip strength of different users is also a variance. So to present a general understanding for readers who want to reproduce this work, the evaluation across the three prototypes is shown in Figure 5.1. The New Right-hand version is different mainly from the weight compared to the Right-hand version, which is a new designed prototype for right hand and is more lightweight than another two prototypes.(An Arduino micro-controller is ever used as a switch of the LEDs.) Without the Vive controller (weighs 206g), the weight of the Right-hand prototype is about 284g, which is as same as the Left-hand one, while the New right-hand on is about 185g. The Left-hand version is just like a mirror flip version to the Right-hand one.

All the tests are implemented with the data collected from subject1, subject4, and subject15, and all the tests are the leave-one-subject out test, i.e.training on 19 subjects using the dataset collected by the Right-hand prototype (the dataset used in implementation), test on dataset collected by another two prototypes. This means the data is also collected using the Left-hand prototype and the New Right-hand prototype from the three subjects who are subject1, subject4, subject15. The results on this work's model and MobileNetV2 are showed in Figure 5.5 and Figure 5.6. It shows a lower accuracy on another two prototypes. Beyond the Air Left posture, top-3 postures are relatively robust across different prototypes. Results on a stronger classifier shows reliable generalization

across different prototypes. Results on the Left-hand prototype is the lowest one, the main reason is thought that because the different grip gesture when one uses left hand. The grip strength of the left hand is usually weak than one's right hand, so it is hard to perform just the same posture when using right hand.

Accuracy results on the New Right-hand prototype is lower than the original right-hand one but the top-3 high accuracy is relatively not changed (Air Left, Air Right, Touch Trigger). There are also outliers in the results.

Overall, this work's approach is reliable across different prototypes in the top-3 postures averagely. And from the comparison to MobileNetV2, it showed that better classifier can be applied to increase the posture robustly recognized in the future. The decline in accuracy is thought because of the minor changes of the camera's position (distance and angle to the hand).

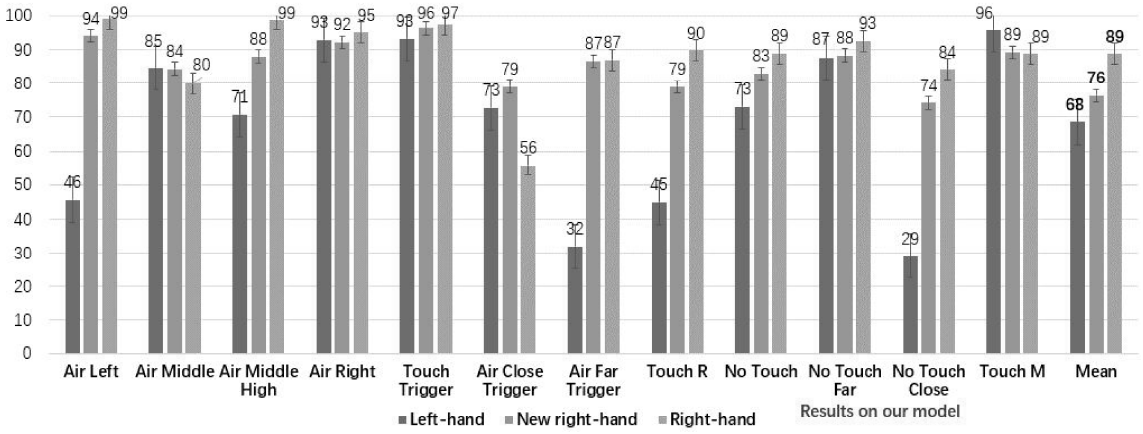


Figure 5.5: Results on generalization across different prototypes using this work's model.

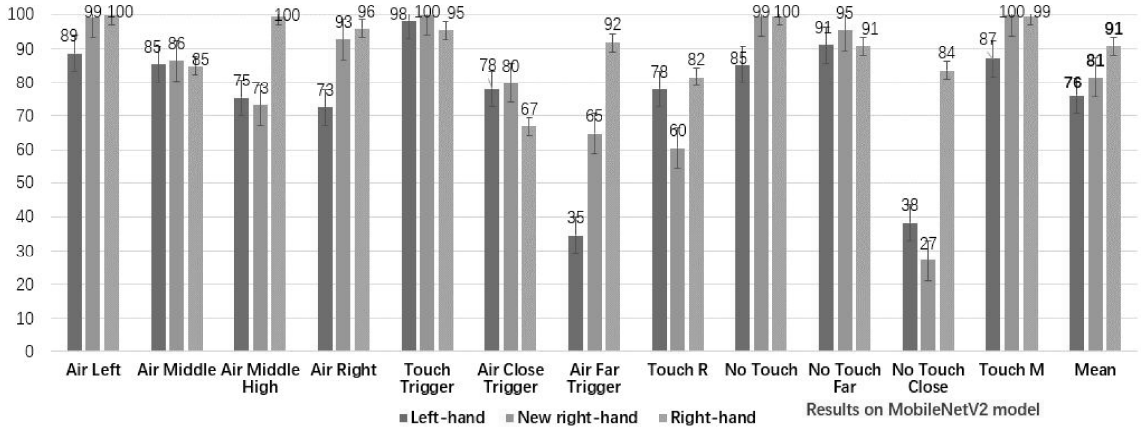


Figure 5.6: Results on generalization across different prototypes using MobileNetV2.

Chapter 6

Example Applications

With the top-3 high accuracy postures, inputs from thumb and index finger are able to be design interaction applications. However, the middle or ring finger need to help to hold the controller, so these fingers' postures are not intentional control input that thought to be not applicable. The application cases can be mainly divided into two types based on the extra finger postures inputs: (i) Control a third-person virtual character, vehicle, or other objects and extend the input pattern. (ii) Enhance the hand presence leveraging each finger posture to map matched finger animations or to render a hand model.

6.1 Extend third-person controls of VR contents

The thumb postures in air can extend the inputs beyond the thumb stick or touch panel of Vive controller. In this work's implementation with the Vive controller, the inputs can be designed to control third-person vehicles. Also, the Touch Trigger posture can extend the trigger input of Vive controller. For example, a transformable robot game is developed . Figure 6.1 illustrates this work's gesture control design for the transform robot game. In the illustration, Touch Trigger posture extends the attack pattern (A). Touch Trigger triggers a move pattern while Press Trigger triggers an another move pattern. When fingers do not touch the controller, a transform input is triggered to transform the robot to a tank robot. Leveraging the touch pad input to control a move

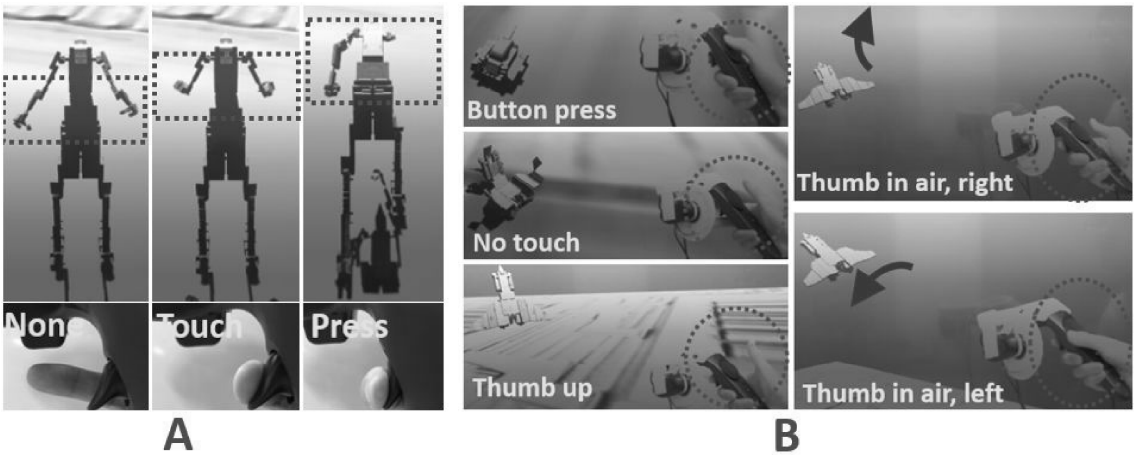


Figure 6.1: The transformable robot game developed.

operation of the tank robot on the ground. Gesture of free the fingers enable a transform input (B, Left). While performing the Air Middle High posture, it enables the rise of the air plane and using the Air Left/Right posture to control the direction of flight. Move the tank robot on the ground by the touch panel while control the plane through thumb postures in air. (B, Right)

6.2 Improve the sense of hand in VR

With further improvements on top-4 high accuracy, touch events from the touch panel of Vive and Touch Trigger, Touch M posture, can perform a grip gesture can be controlled to grasp a virtual object or throw the object. For example, showed in Figure 6.2 (A), leveraging Touch M and Touch R posture with the touch pad of Vive controller can enable a grab gesture to control VR objects. When posture a grab gesture by thumb, index finger and middle fingers, a virtual object is attracted to the hand just like some kind of gravity. Just like the solution of Oculus Touch, using the posture inputs can also render discrete hand poses to enhance the hand presence in VR. The demo showed in Figure 6.2 (B) is a finger tracking example developed in Unity. It controls a state machine of each fingers to map a corresponding finger animation, which provides a model that shows what the user's finger are doing when holding the controller. This enables first-person input control interaction. It is noted that the trigger component and the press button for middle finger on Vive controller do not support touch event input.

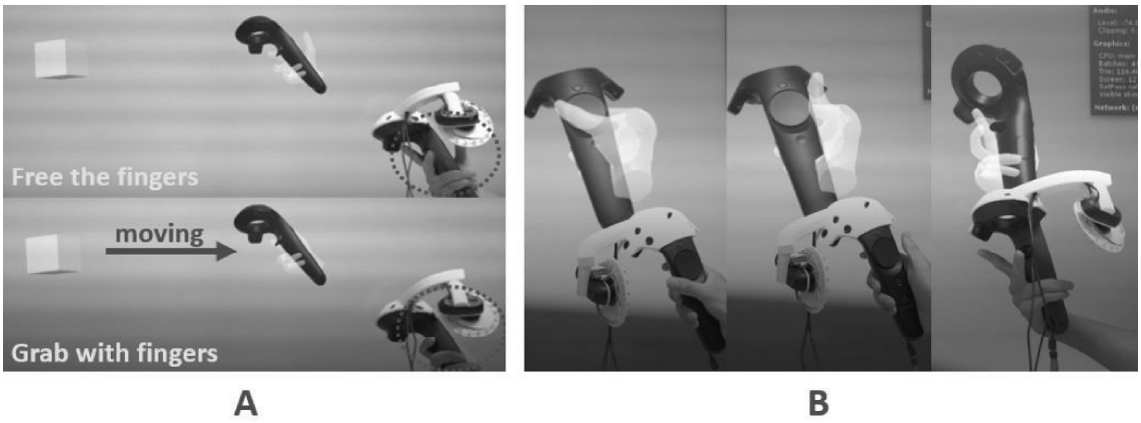


Figure 6.2: The finger tracking demo developed.

Chapter 7

Preliminary Experiment for Combination of the Approaches

In the chapters above, two methods to enhance the interaction ability for VR controllers are introduced, the SFC sensing method and the CNN-based vision sensing method. SFC sensing can reliably recognize the user's holding or touch gestures than traditional capacitive sensing, but can not perceive the fingers' gesture in the air. On the contrary, visual sensing can analyze the movement of fingers in the air, so it is expected that the combination of these two methods is a promising approach to achieve hand recovery for VR controllers. In this chapter, an initial exploration of the combination of these two methods is presented . Furthermore, based on this work's attempts, design insights learned from this work are reported.

7.1 Prototype Overview

Figure 7.1 shows the main hardware components of this work's prototype for the combination of the two approaches. The hardware parts have not changed much compared to the two sensor parts described in the previous chapters. For this hardware device, a finger tracking demo is also implemented. The execution mechanism of the demo system is shown in Figure 7.2. It is noted that the input of touched fingers represents the touch input from SFC sensor, while finger position classification label represents the input of finger postures from camera sensor. The software operating conditions of the system are the same as the experimental settings mentioned in the previous chapter.

7.2 State Machine Method

A state machine method is used to combine the output of two sensors. Considering that the SFC sensor can detect touch events more reliably, the middle finger gestures from the camera sensor is not used to input the gesture of the middle finger, but the middle finger gestures from the SFC sensor. The corresponding 14 gestures (states) are shown in Table 7.1. A state machine is also called a state transition graph. The transition mechanism of this work's state machine is shown in Figure 7.3. In total, 14 discrete gestures are classified for the combination system.

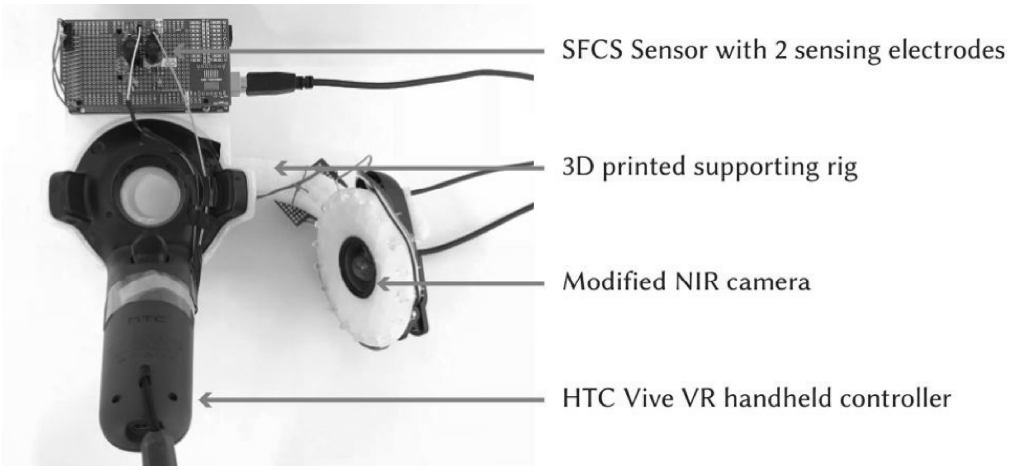


Figure 7.1: Main hardware components of this work's prototype that attached to a Vive controller.

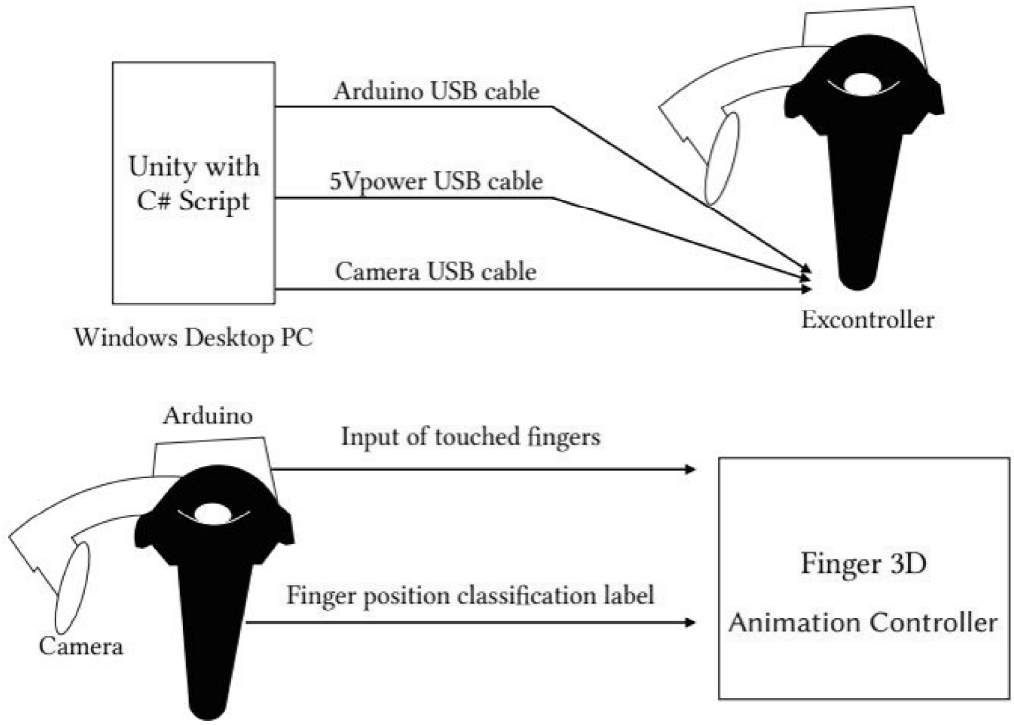


Figure 7.2: Overview of the combination system diagram.

Sensor	Thumb	Index	Middle	Ring	Num. of gestures
SFC	Touch Almui	/	Touch Almui	/	5
	Touch Tape		Touch Tape		
			No Touch		
Camera	Air Left	Touch Trigger	/	Touch	9
	Air Middle				
	Air Right	Air Close Trigger		No Touch	
	Air MiddleHigh	Air Far Trigger			

Table 7.1: States created for the state machine.

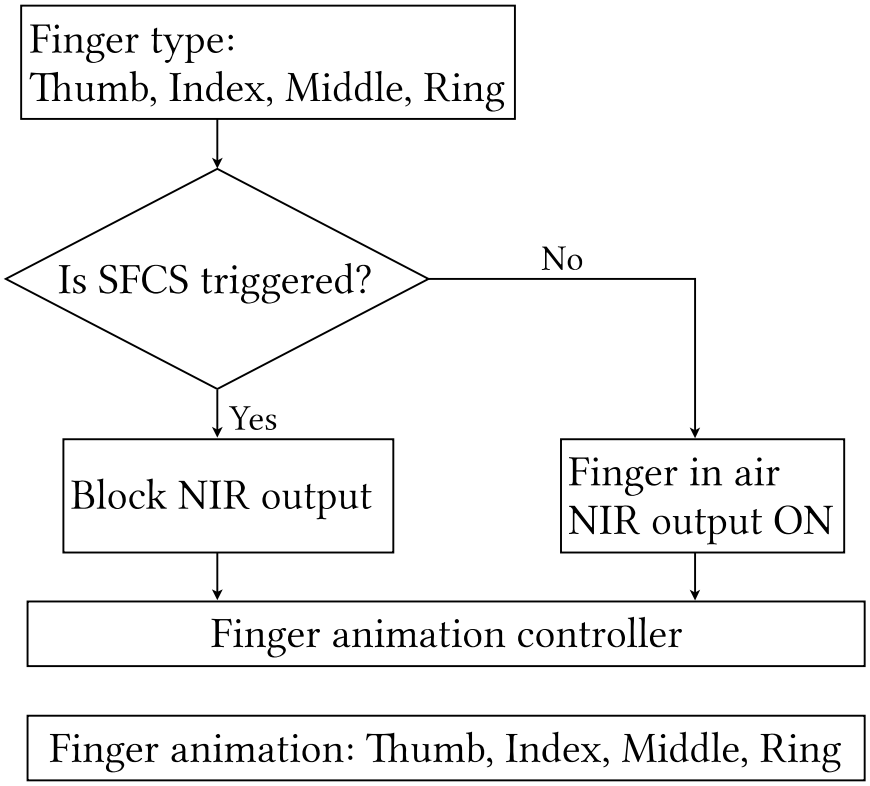


Figure 7.3: Diagram of system logic processing.

7.3 Limitations

In this work's test, although when the two sensors work separately, the real-time frame rates can reach 15 fps and 30 fps respectively. But when combining the input parts of the two sensors together, the frame rate of the system decreases to about 1 fps. Since the whole system almost relies on the CPU for calculation and processing, it is an option to upgrade the hardware. But for the combination of the two methods, as it is still a preliminary attempt, it is thought to be entirely feasible to optimize the software processing to improve the running frame rate. Similarly, the weight of the whole controller is increased by simply putting the two sensor parts together. This is a burden on users. These limitations will be discussed in detail in later chapters.

Chapter 8

Discussion

Because the traditional camera-based method and data glove method still cannot meet the user's interaction needs, such as the occlusion problem and the uncomfortable wearing sense of data glove. Oculus Touch controller and Knuckles controller provide users with a more real sense of hand ownership by finger tracking while holding the controller. These controllers inspired this work's exploration of the possibilities of finger tracking for handheld controller-based interaction. In this paper, two approaches of adding different sensors are explored in order to improve the possibility of interaction using VR controllers. One is to use the sensor matrix method based on SFCS. Compared with the traditional capacitive sensing method, SFCS can detect two-dimensional data based on time stamp and frequency, thus providing more detectable touch information. The other is gesture classification method based on deep learning, which is applied to a modified webcam. These two methods are used to implement the prototypes of the conceptual device. As for the finger tracking demo, frame rates of the two software systems can reach 15 fps and 30 fps respectively in real-time.

8.1 Limitations

Based on the principle of SFCS, the method of constructing sensor metrics can increase the resolution, but its recognition results still depend on machine learning method. Not only that, but it still relies on specific design tasks when try to construct a sensor matrix. For example, in this work's experiment, for the middle finger, when the Touch Tape gesture is triggered, the result curve of Touch Tape is one pattern, while when Touch Alumi is triggered at the same time, the result curve of Touch Tape of middle finger gesture is another pattern. This problem is called multi-touch problem by this work. Although the sensor matrix has been constructed, and each circuit has been made independent in software processing. However, when two fingers (in this work's implementation, the middle finger and the thumb) touch the sensing electrode at the same time and when two fingers touch the sensing electrode at different times, the capacitive profiles of Touch Tape in these two cases are different. Because the electrical circuit composed by the body and the sensor itself has changed, the capacitance of the whole circuit has also changed. This problem may be solved by increased the number of samples, but meanwhile this way could add extra latency of the recognition program. In total, decreasing the sweep resolution improved real-time performance of this work's prototype, but also reduced the robustness of gesture recognition in this work's application. Another limitation is that this work's current software processing only uses the consistency of curve results to distinguish different touch events. Machine learning method can increase the

reliability of the results, but it still needs to face the problem of generalization in different subjects. Team of Touché [24] project used a customized device, so using the same device may also improve the accuracy of recognition.

In the camera method, this work's finger gesture recognition accuracy results show that the classifier can be reliably recognized in the top-4 high accuracy gestures. But misclassification shows that the classifier can't recognize finger gestures in the air very well. In addition, in the actual test of finger tracking demo, it seems that the matching effect of gesture animation with accuracy over 95% makes subjects feel more reliable visually. Three different prototype devices are tested. Because of the slight difference between the position and angle of the camera, the accuracy of prototype devices that not used for data collection is lower. Especially, the left-hand device in Figure 5.1 decreased much. By making the camera fixed into the 3D model when designing the 3D print from the beginning, it is thought that this problem can be solved to some extent. It should be emphasized that the accuracy test results provided by us are generalization tests among different 20 experimental objects, so this work's equipment has a certain practical reliability.

From previous work on fingers' freedom for on-device interaction, it is known that the thumb has a larger range of motion, so the thumb-based gesture interaction may be better designed. Especially based on intentional interaction, such as left and right position and up and down position that are able to control the movement of virtual objects or interactive interfaces. In the previous chapter, four different methods to attach the camera are given. It is expected that if the camera is positioned directly above the controller in Figure 4.3, the recognition rate of the Air Left, Air Middle and Air Right gestures of the thumb could be recognized with higher rates, but the prediction of the Air Middle and Air Middle High gestures may be misclassified. This is because it is difficult to distinguish depth information from two-dimensional images. For example, in this work's current installation, similar misclassification also occurred in the Air Middle posture and the Air Right posture. But furthermore, it is also considered to add two or even three cameras to increase the accuracy of posture recognition. In this case, choosing a camera with a larger angle of view and smaller size could make the prototype more compact. At the same time, using multiple cameras can also support to analyze the depth information of finger position to recovery the 3D gesture of the whole hand.

And for both of the sensors, a common limitation is that, for both of the approaches, it is necessary that users must naturally and correctly hold the controller to enable the SFC sensor or camera sensor correctly. For the SFC sensor, especially for the middle finger, because the size of each person's hand and the length of the finger are different, when the user holds the controller with the most natural grip, the place where each person's middle finger falls correspondingly differs. Then, because of the different grip strength of subjects, the middle finger and ring finger may locate on different positions of the controller surface, resulting in an unreliable recognition.

The combined testing of the two initial methods shows that there is still space for improvement in the software and hardware parts. With regard to improving the frame rate of finger tracking demo, the input and recognition programs of SFC sensor and camera sensor can be rewrited into a dynamic link library (DDL) file based on C++, and then call the library functions in real time when the Unity program runs. In this way, the latency are thought to be decreased. In terms of

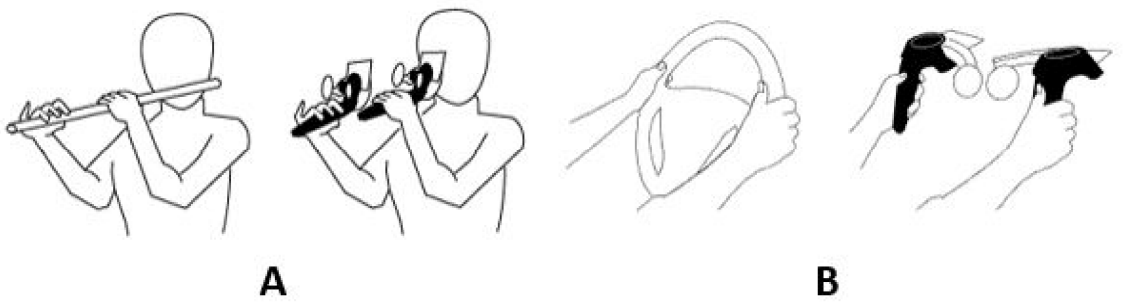


Figure 8.1: Conceptual design illustration of feasible VR example applications.

hardware equipment, choosing smaller micro-controller device should reduce the weight of the whole prototype equipment.

8.2 Design Insights

In this paper, on-controller based gesture recovery are explored. However, due to the difficulty of obtaining depth information from very close range images, the issue of continuous full hand recovery is still challenging. Even so, this work is thought to create a meaningful step forward in this design space. In this section, design insights about other application tasks based on this work's device prototype are discussed.

This paper provides a discrete real-time gesture recovery performance for Vive VR controller held by users. Based on the design of Vive controller, proposed prototype is suitable for VR applications that require a highly visual finger performance. It is also applicable to VR tools that need to be held by users. Specially, ideally it is thought that this prototype is suitable for a simulation of VR woodwind string instruments, which can be used in education field of instruments teaching. Illustrated as Figure 8.1 (A), For example, when designing a VR flute application, both of the controllers can be together used to simulation the real finger gesture when play a flute. The flute generally has six holes for users to blows across, by which users can play a melody. The six holes can be mapped to six different touch gestures from a pair of Vive controllers. One 2D SFC sensor matrix prototype can provides at least 4 different touch gestures, so it entirely satisfies the design requirements. The movement of finger in the air can be visually seen for users to give them a realistic experience of playing the flute. As for a generalization, it can be applicable to most woodwind musical instruments.

Another application design is shown as Figure 8.1, when a user is driving a car, this work's prototype can provide the user his or her finger movement visually so that the immerse experience is increased. Also, the touch information can be used to provide a interaction design, user can learn how to drive by using the steering wheel. Additionally, with the proper design of VR controller, it is believed that this work's system can be scaled well to other VR applications.

Chapter 9

Future Work

Exploratory prototypes are designed with different finger postures to perform finger tracking on Vive controllers. Because of the different hand size and grip strength of subjects, the middle finger and ring finger may locate on different positions of the controller surface, resulting in an unreliable recognition. This is found by this work's real-time demo test on middle and ring finger postures. But the ring finger postures seem not able to provide effective interaction based on the grip gesture of Vive controller, so just training the dataset without the ring finger data may improve the recognition rate. So a future work can be conducted by changing the camera distance and angle so that the camera only images the thumb, index, and middle fingers. For the combination approach, using 3D printing to customize a controller model to embed the two sensors may make the device prototype look less bulky.

Through this work, implementation like this work's NIR camera approach is thought to be feasible for other VR controllers. It is because based on the design of a controller, there is always an appropriate distance and angle to mount a camera onto it to image all fingers.For example, other VR controllers such as Oculus Touch controller, even the grip gesture is totally different from the Vive controller, this work's approach is thought to be applied like a design of Figure 9.1. The image from the camera view. Based on the design of Oculus Touch, thumb and index fingers has more movements in air. But a smaller camera, and other suitable IR LEDs should be reconsidered

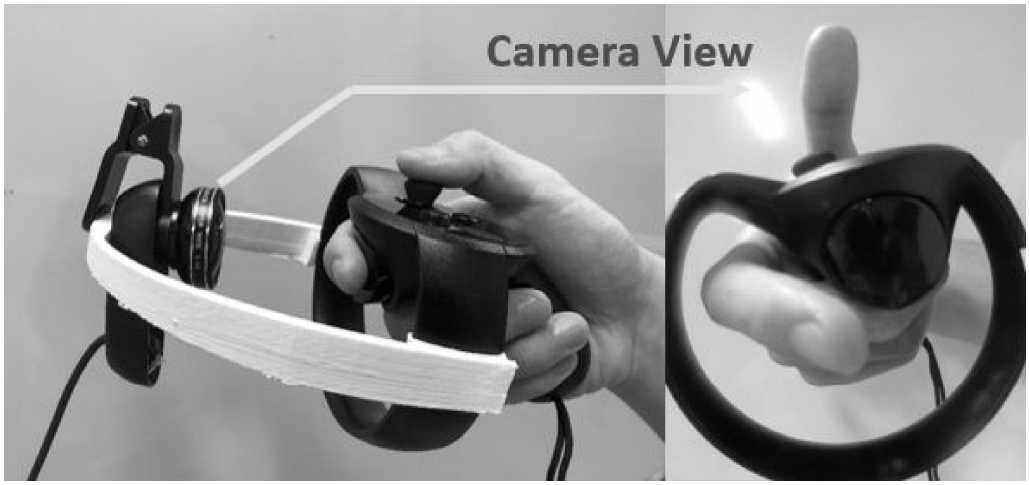


Figure 9.1: A tentative design of prototype for Touch controller.

based on the controller's design. Still, designing a customized 3D print controller should be more feasible for specific tasks.

With the development of hardware technology, if a depth camera is able to work in a close range (< 10cm). Future work are also considered to recover the full hand pose for VR controllers. But the occlusion problem from the controllers is a challenging issue. This work also can be seen a exploratory approach that recover the discrete hand pose by developed image classification approach.

Chapter 10

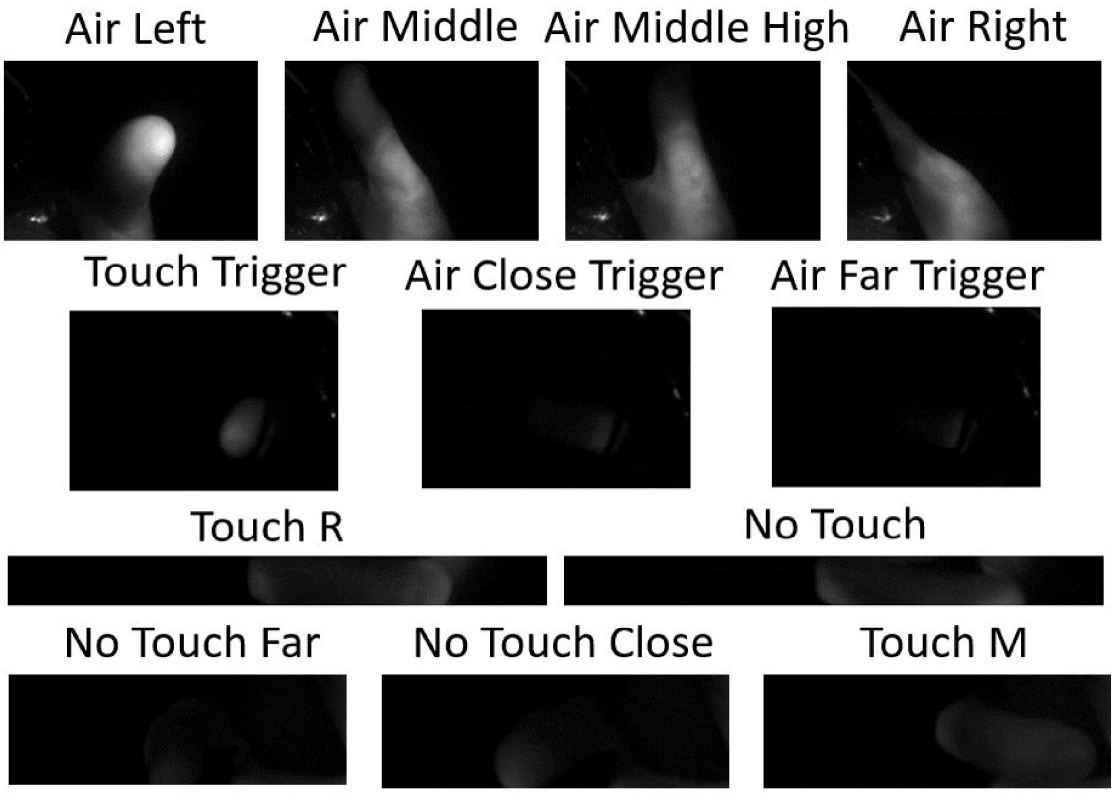
Conclusion

This paper presented two approaches that enables a real-time finger gesture recognition system for VR handheld controller. Each prototype is implemented by two different components. One is the SFC sensor that provide the system for touch information. Another is a NIR camera that applied with a trained CNN model to obtain the positions of fingers. This paper also explored an initial prototype to apply a combination of both of the proposed sensing approach to a VR handheld controller. A finger tracking demo is developed in Unity for the two sensors. Real-time performance of 15 fps and 30 fps were achieved respectively on a consumer computer.

Specially, the NIR camera approach is an enabling technology designed to provide extra inputs to extend interaction capabilities with VR controllers. The proposed prototype is currently designed for the Vive controller which is able to recognize 12 different finger gestures with relatively high accuracy. The predictive model is able to generalize on users not occurring during training with a cross-validation accuracy of 86.17%. Foundlings are that 3 postures can be robustly recognized across different subjects and across different prototypes. This work has taken the first step on finger tracking based on VR controller and believe that this study can bring new insights to other researchers in the field of human-computer interaction in VR.

Appendix

Dataset sample of finger postures captured the NIR camera.



References

[1] H. M. Elfekey and H. A. Bastawrous. Design and implementation of a new thin cost effective ac hum based touch sensing keyboard. In *2013 IEEE International Conference on Consumer Electronics (ICCE)*, pages 602–605, Jan 2013.

[2] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. 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, pages 3293–3315, New York, NY, USA, 2017. ACM.

[3] Jess McIntosh, Asier Marzo, Mike Fraser, and Carol Phillips. Echoflex: Hand gesture recognition using ultrasound imaging. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1923–1934, New York, NY, USA, 2017. ACM.

[4] Jess McIntosh, Charlie McNeill, Mike Fraser, Frederic Kerber, Markus Löchtefeld, and Antonio Krüger. Empress: Practical hand gesture classification with wrist-mounted emg and pressure sensing. In *CHI*, 2016.

[5] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, July 2008.

[6] Siddharth S. Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, Jan 2015.

[7] Andrea Colaço, Ahmed Kirmani, Hye Soo Yang, Nan-Wei Gong, Chris Schmandt, and Vivek K. Goyal. Mime: Compact, low power 3d gesture sensing for interaction with head mounted displays. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 227–236, New York, NY, USA, 2013. ACM.

[8] Liwei Chan, Chi-Hao Hsieh, Yi-Ling Chen, Shuo Yang, Da-Yuan Huang, Rong-Hao Liang, and Bing-Yu Chen. Cyclops: Wearable and single-piece full-body gesture input devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3001–3009, New York, NY, USA, 2015. ACM.

[9] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 441–450, New York, NY, USA, 2011. ACM.

[10] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. Shoesense: A new perspective on gestural interaction and wearable applications. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, CHI '12, pages 1239–1248, New York, NY, USA, 2012. ACM.

[11] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 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, pages 167–176, New York, NY, USA, 2012. ACM.

[12] Manuel Prätorius, Dimitar Valkov, Ulrich Burgbacher, and Klaus Hinrichs. Digitap: An eyes-free vr/ar symbolic input device. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, VRST '14, pages 9–18, New York, NY, USA, 2014. ACM.

[13] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 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, pages 549–556, New York, NY, USA, 2015. ACM.

[14] Hao Li, Laura Trutoiu, Kyle Olszewski, Lingyu Wei, Tristan Trutna, Pei-Lun Hsieh, Aaron Nicholls, and Chongyang Ma. Facial performance sensing head-mounted display. *ACM Trans. Graph.*, 34(4):47:1–47:9, July 2015.

[15] Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. High-fidelity facial and speech animation for vr hmds. *ACM Trans. Graph.*, 35(6):221:1–221:14, November 2016.

[16] Xing-Dong Yang, Khalad Hasan, Neil Bruce, and Pourang Irani. 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, pages 291–300, New York, NY, USA, 2013. ACM.

[17] Alex Butler, Shahram Izadi, and Steve Hodges. Sidesight: Multi-"touch" interaction around small devices. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 201–204, New York, NY, USA, 2008. ACM.

[18] Sven Kratz and Michael Rohs. Hoverflow: Expanding the design space of around-device interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 4:1–4:8, New York, NY, USA, 2009. ACM.

[19] F. Aezinia, Y. Wang, and B. Bahreyni. Touchless capacitive sensor for hand gesture detection. In *SENSORS, 2011 IEEE*, pages 546–549, Oct 2011.

[20] Mathieu Le Goc, Stuart Taylor, Shahram Izadi, and Cem Keskin. 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, pages 3167–3170, New York, NY, USA, 2014. ACM.

[21] Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. Modeling the functional area of the thumb on mobile touchscreen surfaces. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1991–2000, New York, NY, USA, 2014. ACM.

[22] Hyunjin Yoo, Jungwon Yoon, and Hyunsoo Ji. Index finger zone: Study on touchable area expandability using thumb and index finger. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '15, pages 803–810, New York, NY, USA, 2015. ACM.

[23] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. Fingers' range and comfortable area for one-handed smartphone interaction beyond the touchscreen. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 31:1–31:12, New York, NY, USA, 2018. ACM.

[24] Munehiko Sato, Ivan Poupyrev, and Chris Harrison. Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 483–492, New York, NY, USA, 2012. ACM.

[25] Chris Harrison, Munehiko Sato, and Ivan Poupyrev. Capacitive fingerprinting: Exploring user differentiation by sensing electrical properties of the human body. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 537–544, New York, NY, USA, 2012. ACM.

[26] Ivan Poupyrev, Philipp Schoessler, Jonas Loh, and Munehiko Sato. Botanicus interacticus: Interactive plants technology. In *ACM SIGGRAPH 2012 Emerging Technologies*, SIGGRAPH '12, pages 4:1–4:1, New York, NY, USA, 2012. ACM.

[27] Mert Canat, Mustafa Ozan Tezcan, Celalettin Yurdakul, Eran Tiza, Buğra Can Sefercik, Idil Bostan, Oğuz Turan Buruk, Tilbe Göksun, and Oğuzhan Özcan. Sensation: Measuring the effects of a human-to-human social touch based controller on the player experience. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3944–3955, New York, NY, USA, 2016. ACM.

[28] Colin Honigman, Jordan Hochenbaum, and Ajay Kapur. Techniques in swept frequency capacitive sensing: An open source approach. In *NIME*, pages 74–77, 2014.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[30] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Trans. Graph.*, 33(5):169:1–169:10, September 2014.

[31] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *CoRR*, abs/1606.06854, 2016.

[32] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[33] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In *Lecture Notes in Computer Science*, pages 572–578. Springer, 2015.

[34] Ao Tang, Ke Lu, Yufei Wang, Jie Huang, and Houqiang Li. A real-time hand posture recognition system using deep neural networks. *ACM Trans. Intell. Syst. Technol.*, 6(2):21:1–21:23, March 2015.

[35] Brandon Garcia and Sigberto Alarcon Viesca. Real-time american sign language recognition with convolutional neural networks. *Convolutional Neural Networks for Visual Recognition*, 2016.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[37] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

[38] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2018.

[40] Sean Ryan Fanello, Cem Keskin, Shahram Izadi, Pushmeet Kohli, David Kim, David Sweeney, Antonio Criminisi, Jamie Shotton, Sing Bing Kang, and Tim Paek. Learning to be a depth camera for close-range human capture and interaction. *ACM Trans. Graph.*, 33(4):86:1–86:11, July 2014.

[41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

Publications and Conference Presentations

[1] 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), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, Article 87, 2 pages. DOI: https://doi.org/10.1145/3281505.3283385

Acknowledgement

Firstly, I would like to express my sincere gratitude to my principal supervisor Professor Yoichi Ochiai for the continuous support of my master study. Furthermore, I am very grateful to members of my research lab who helped me, especially to Yaohao Chen, Satoshi Hashizume, Naoya Muramatsu, Kotaro Omomo, Riku Iwasaki, and Kaji Wataru, for the useful comments and cooperation, . Also, I would like to thank to all the subjects who participated in the data collection task. With their time to join this research, I complete the implementation successfully. Finally, I want to thank all the institutions for making their valuable comments on this thesis.