國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

鍾手勢：手勢辨認使用飛時相機應用於智慧型眼鏡

ChungGesture: Hand Gesture Recognition with Time-of-Flight Camera for Smart Glasses

鍾承鎧

Cheng-Kai Chung

指導教授：傅楸善 博士

Advisor: Chiou-Shann Fuh, Ph.D.

中華民國 112 年 6 月

June 2023

doi:10.6342/NTU202302797

國立臺灣大學碩士學位論文
口試委員會審定書

MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

鍾手勢：手勢辨認使用飛時相機應用於智慧型眼鏡

ChungGesture: Hand Gesture Recognition with Time-of-Flight Camera for Smart Glasses

本論文係鍾承鎧君（學號 R10922168）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 112 年 6 月 5 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Computer Science and Information Engineering on 5 June 2023 have examined a Master's thesis entitled above presented by Cheng-Kai,Chung (student ID: R10922168) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

傅楸善
（指導教授 Advisor）

[illegible]

[illegible]

系主任/所長 Director: 洪士灝

誌謝

能順利完成本篇論文首先要感謝傅楸善教授的指導，他總是不遺餘力地在研究過程中支持我並給予建議，同時也要感謝佐臻股份有限公司提供合作的機會，支持我研究這個主題。感謝公司同仁們專業的意見讓我受益良多，尤其是陳宏融同事的指導。

另外也要感謝數位相機與電腦視覺實驗室的學長姊、同學和學弟妹們在研究期間的照顧。邵育翔學長、龔柏丞學長、孫譽學長、呂英弘學長、李志洸學長、蕭延儒學長、閻楷青學姊無論在任何問題總會給予我適當的幫助，讓我在研究過程中順利度過難關。同學張季祐、許銘真、游凱任、林正偉、林佳城、林聖祐，在研究過程中互相扶持照顧，一同分享學業上的甘苦。學弟何志宏、郁霈靖、李詠億、吳柏緯、邱議禾、游惟丞，學妹方郁婷、張婷淇，實驗室因為你們的加入帶來不少歡樂，讓我忘卻課業上的煩悶。數位相機與電腦視覺實驗室的所有成員們對於幫助我完成這篇論文實在功不可沒。

最後要感謝我的家人和好友，總是毫無保留地支持我在學業上的任何決定，使我能專心投入課業，無後顧之憂。在此僅將這段時間的研究成果匯集成本論文，獻給所有曾經關心、照顧與幫助我的所有人。

doi:10.6342/NTU202302797

中文摘要

本論文提出一個名為「鍾手勢」的演算法，希望藉由搭載著 8x8 像素的低解析度飛時深度相機的智慧型眼鏡，來進行手勢辨識，其中共包含了六個精準、快速、舒適的動作，旨在提高使用者的操作體驗。

我們首先回顧了手勢辨認技術的相關知識和現有相關研究，並探索三種主要的深度預測技術：立體視覺、結構光和 ToF 相機。接著，我們針對智慧型眼鏡的特性和使用情境，提出了適合的手勢辨認應用場景和手勢操作設計，並開發了相對應的軟體系統。最後，我們進行了實驗評估，包括手勢辨認的精準度、穩定度和使用者的滿意度。實驗結果顯示，本研究所提出的手勢辨認系統具有良好的準確度和穩定度，且能有效提升智慧型眼鏡的使用體驗。

我們的演算法開發及應用皆是在佐臻的 J7EF Plus 智慧型眼鏡上，詳細的方法和流程會在論文中加以說明。

關鍵字：鍾手勢、飛時相機、手勢辨識、擴增實境眼鏡

doi:10.6342/NTU202302797

ABSTRACT

In this thesis, we propose ChungGesture, which utilizes a low-resolution Time-of-Flight (ToF) camera with 8x8 pixels to perform gesture recognition on smart glasses. The algorithm consists of six precise, fast, and comfortable gestures designed to enhance the user experience.

We first review the relevant knowledge and existing research on gesture recognition technology and explored three main depth prediction techniques: stereo vision, structured light, and ToF camera. Next, we propose suitable application scenarios and gesture designs based on the characteristics and usage context of smart glasses, and develop corresponding software systems. Finally, we conduct experimental evaluations, including gesture recognition accuracy, stability, and user satisfaction. The results show that our gesture recognition system has good accuracy and stability and can effectively improve the user experience of smart glasses.

Our algorithm development and application are performed on Jorjin J7EF Plus smart glasses, and the detailed methods and processes will be explained in this thesis.

Key words: ChungGesture, Time-Of-Flight Camera, Gesture Recognition, Augmented Reality (AR) Glasses

doi:10.6342/NTU202302797

CONTENTS

誌謝 i

中文摘要 ii

ABSTRACT iii

CONTENTS iv

LIST OF FIGURES vii

LIST OF TABLES x

Chapter 1 Introduction 1

1.1 The Concept of Smart Glasses 1

1.1.1 Hardware 2

1.1.2 Virtual Reality (VR) and Augmented Reality (AR) 3

1.1.3 Interacting with Smart Glasses 7

1.2 Hand Gestures 9

1.3 Thesis Organization 11

Chapter 2 Related Works 12

2.1 History of Smart Glasses 12

2.2 The Method of Depth Estimation 16

doi:10.6342/NTU202302797

2.2.1 Stereo Vision 17

2.2.2 Structured Light 18

2.2.3 Time-of-Flight Camera 20

2.3 Hand Gestures Recognition 23

Chapter 3 Background 26

3.1 Our Device Configuration 26

3.2 Sensor Decision 29

3.3 Hardware Limitation 30

3.3.1 The Low-Resolution Depth Sensor 30

3.3.2 Limited Field of View (FoV) 31

3.3.3 Sensor Mobility Constraint 31

3.4 Existing Techniques and Limitations 31

3.4.1 LuGesture Method 31

3.4.2 Limitations of LuGesture Method 37

Chapter 4 Methodology 40

4.1 Overview 40

4.2 Data Collection 41

4.3 Noise Reduction 42

doi:10.6342/NTU202302797

4.3.1 **Remove Background** 42

4.3.2 **Remove Twinkle** 44

4.4 Hand Area Detection 45

4.4.1 **Left-Right Hand Discrimination** 45

4.4.2 **Cropping for Left-Right Hand Differentiation** 46

4.4.3 **Calculating Hand Palm Centroid** 48

4.4.4 **Smoothing** 52

4.5 Gesture Recognition 53

4.5.1 **Swipe** 54

4.5.2 **Push and Pull** 56

4.6 Our System 57

Chapter 5 Experimental Results 59

5.1 Experimental Setting 59

5.2 Precision Result 59

5.3 Stability Result 61

5.4 Gesture Recognition Accuracy Result 64

Chapter 6 Conclusion and Future Works 66

References 67

doi:10.6342/NTU202302797

LIST OF FIGURES

Figure 1-1: Simplified representation of a “virtuality continuum” [1]............................5

Figure 1-2: Classification of interaction approaches for smart glasses [2]. 8

Figure 1-3: The classification Aigner et al. used to communicate information about objects or entities, such as specific sizes, shapes, and motion paths [3]. 10

Figure 2-1: Evolution of media devices [7]..12

Figure 2-2: Stereoscopic vision diagram [20]. 18

Figure 2-3: Structured light camera diagram [23]. 19

Figure 2-4: Overview of pulsed and modulated D-ToF and I-ToF measuring techniques [4]...21

Figure 2-5: Principle of dToF sensor [5]. 22

Figure 2-6: The components of a video- or depth-based hand 24

Figure 3-1: Jorjin J-Reality J7EF Plus [18]. 26

Figure 3-2: Sony Xperia 1 II [24]. 27

Figure 3-3: Jorjin J7EF Plus display mode. (a) 2D mode. (b) 3D mode (Jorjin, 2023) [18]. 29

Figure 3-4: Examples of difference images for each direction. (a) Swipe left. (b) Swipe

doi:10.6342/NTU202302797

right. (c) Swipe up. (d) Swipe down. The motion direction can be observed by the general direction from red (disappearing: negative values) to blue (appearing: positive values) [17]. 33

Figure 3-5: The whole process to detect swipe gestures. For Step 3, we show the graph of the 4th row. We can see that the score peaks at index 3 and 4, which is where our middle point is. The same process applies to every row for the horizontal score as shown, and every column for the vertical score [17]. 34

Figure 3-6: Illustration of the two stages of a push gesture [17]. 35

Figure 3-7: Illustration of the three stages for hand-speed detection [17].......................36

Figure 3-8: Processing to reduce jitter. (a) Unprocessed binary image. (b) Reweighted image. (c) Upscaled and blurred image after reweighting. Note that these are different frames [17]. 37

Figure 3-9: The result image of LuGesture. Hand in the (a) top-left corner, (b) upper-center corner, (c) top-right corner. 38

Figure 4-1: Main flowchart of ChungGesture. 41

Figure 4-2: The data flow chart of smart glasses [18]. 42

Figure 4-3: Sensing range of hand pixels. (a) Direct view. (b) Side view. 43

Figure 4-4: Comparison chart of twinkle processing. (a) Twinkle image. (b) After twinkle

doi:10.6342/NTU202302797

removal. 44

Figure 4-5: Determine left-right hand. 46

Figure 4-6: Hand palm image. 48

Figure 4-7: Precision problem. 49

Figure 4-8: Gaussian weighting image. (a) Averaged point. (b) Fingertip. 51

Figure 4-9: Weight table. 52

Figure 4-10: Right-hand swipe left image. 55

Figure 4-11: Illustration of a push gesture. 57

Figure 5-1: Screenshot from the Color Weakness game used in our experiments. 60

doi:10.6342/NTU202302797

LIST OF TABLES

Table 2-1. Specification of Google Glass Enterprise Edition 2 [9], Microsoft HoloLens 2 [11], and Epson Moverio BT-350 [12]. 15

Table 2-2. Comparison of 3D imaging technologies [21]. 17

Table 2-3. Comparison between dToF and iToF. 23

Table 5-1. Comparison results of precision. 61

Table 5-2. Comparison results of stability. 63

Table 5-3. Comparison results of gesture recognition. 65

doi:10.6342/NTU202302797

Chapter 1 Introduction

1.1 The Concept of Smart Glasses

Smart glasses are a type of Optical Head-Mounted Display (OHMD) device that mainly provides the wearer with real-time reference and judgment of various information by projecting computer screens onto the lenses of the glasses. With the advent of the 5G era, network transmission is getting faster. Smart glasses, combined with cloud technology, are further expanded into various fields, such as simulating real-life scenarios for assistive and medical personnel. With the assistance of smart glasses, they can handle real-time information more efficiently and give the wearer a more immersive experience. Currently, smart glasses have become a fierce competition project for major companies such as Microsoft, Amazon, Google, Apple, Sony, and Samsung.

This thesis aims to illustrate how to combine medical and entertainment activities through the use of self-developed smart glasses and gesture recognition algorithm in collaboration with Jorjin Technologies. Therefore, smart glasses play a very important role in this thesis, responsible for receiving information, data processing, triggering

doi:10.6342/NTU202302797

events, and finally displaying them on the screen, allowing users to overlay virtual world information onto the physical world and achieve a more immersive interactive experience. In this section, we will introduce the hardware of smart glasses, as well as what Virtual Reality (VR) and Augmented Reality (AR) are, and how users interact with the environment through smart glasses in both virtual and real worlds.

1.1.1 Hardware

Currently, many companies engage in the development of smart glasses. The most common hardware components include:

- Display: A miniature display screen located within the glasses that projects images and videos onto the wearer's field of view. Some glasses use micro-Organic Light-Emitting Diode (OLED) displays to provide high resolution and sharpness.
- Camera: A built-in camera that captures images and videos of the user's surroundings. This feature is essential for AR experiences as it allows the device to recognize and overlay digital information onto physical objects.
- Sensors: Smart glasses may also come equipped with a range of sensors such as

doi:10.6342/NTU202302797

accelerometers, gyroscopes, and magnetometers. These sensors track the user's head movements, allowing the device to adjust the display accordingly.

- Processor and Memory: A Central Processing Unit (CPU) and memory are essential components that enable the device to run complex algorithms and render high-quality graphics. Some smart glasses use mobile processors and operating systems similar to those found in smartphones.
- Connectivity: Smart glasses often feature Wireless Fidelity (Wi-Fi), Bluetooth, or cellular connectivity, allowing them to connect to Internet and other devices.

1.1.2 Virtual Reality (VR) and Augmented Reality (AR)

Recently, the development of Augmented Reality (AR) and Virtual Reality (VR) technologies has brought new possibilities to the field of smart glasses. Therefore, we need to first define what Virtual Reality and Augmented Reality are.

Virtual Reality (VR) and Augmented Reality (AR) are both technologies that combine computer-generated graphics with the real world. In VR, users are immersed in a completely synthetic world and can interact with it. Users can experience a sense of

doi:10.6342/NTU202302797

presence. The physical laws of the real world may not apply in this synthetic world. VR is commonly used in applications such as gaming and education. On the other hand, AR overlays virtual images onto the real-world environment, allowing users to see both the real world and virtual graphics simultaneously. For example, AR filters on social media platforms add virtual animal ears or masks onto users' faces in real time. As the Jorjin J7EF Plus smart glasses used in this thesis are an AR application, in this section we will focus on AR.

In Figure 1-1, AR is a subset of Mixed Reality (MR). MR visual displays are a specific category of Virtual Reality (VR) technologies that involve the integration of real and virtual worlds at various points along the "virtuality continuum", which ranges from entirely real environments to fully virtual ones (Milgram & Kishino, 1994) [1]. The concept of a "virtuality continuum" refers to the blending of object classes presented in a particular display situation. This continuum spans from completely real environments at one end to completely virtual ones at the other. At the left end, real environments consist solely of real objects and can be observed through a conventional video display or directly without any electronic display. At the opposite end, virtual environments consist solely of virtual objects, such as those in a computer-generated simulation.

doi:10.6342/NTU202302797

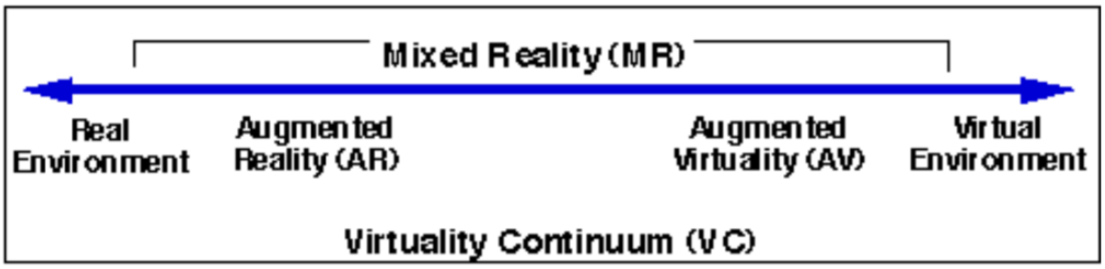


Figure 1-1: Simplified representation of a “virtuality continuum” [1].

While the term "Mixed Reality" may not be widely used, the term "Augmented Reality" (AR) has become increasingly common in the literature. Thus, we define Augmented Reality as any situation where a real-world environment is enhanced or "augmented" by virtual computer-generated objects. Clear classification helps us to have a better understanding of AR. In the following, we will introduce the six categories defined by Milgram and Kishino [1].

1. Monitor-based video displays, also known as "window-on-the-World" (WoW) displays, where computer-generated images are overlaid electronically or digitally. An example is a 3D movie or game displayed on a computer screen.
2. Video displays similar to those in Class 1, but with immersive Head-Mounted Displays (HMD’s) rather than WoW monitors.
3. HMDs equipped with a see-through capability, allowing computer-generated

doi:10.6342/NTU202302797

graphics to be optically superimposed onto real-world scenes viewed directly. An instance of this would be the Google Glass, which features a transparent display attached to the right side of the glasses.

4. Similar to Class 3, but with video viewing of the "outside" world instead of optical. The displayed world in Class 4 should correspond orthoscopically with the immediate outside real world, creating a "video see-through" system, similar to the optical see-through of option 3.
5. Completely graphic display environments to which video "reality" is added, either completely immersive, partially immersive, or otherwise.
6. Completely graphic but partially immersive environments where real physical objects in the user's environment play a role in the computer-generated scene, such as reaching in and grabbing something with one's own hand.

After a brief overview of the various types of MR, it becomes apparent that even if many projects and applications share the same AR tag, they may require quite different hardware configurations and software implementations. In this thesis, we will focus on Class 3 MR devices, also referred to as "smart glasses."

doi:10.6342/NTU202302797

1.1.3 Interacting with Smart Glasses

After understanding the hardware of smart glasses and the concepts of VR and AR, we will now introduce how to interact with smart glasses.

Smart devices today primarily use touchscreens as the main input method. However, smart wearable devices such as smart glasses do not incorporate touchscreens because they can block the user's line of sight and limit the flexibility of hand movements. The interaction approaches are classified into three categories based on Lee and Hui's work (Lee & Hui, 2018) [2]: handheld, touch, and touchless. First, handheld utilizes handheld controllers, such as smartphones. Second, touch refers to non-handheld touch-based interaction, including gestures and tapping on body surfaces, as well as touch-sensing wearable devices such as smart rings, smart wrist bands, and watches. Last, touchless includes non-handheld and non-touch input, such as mid-air hand gestures, head and body movements, gaze interaction, and voice recognition.

doi:10.6342/NTU202302797

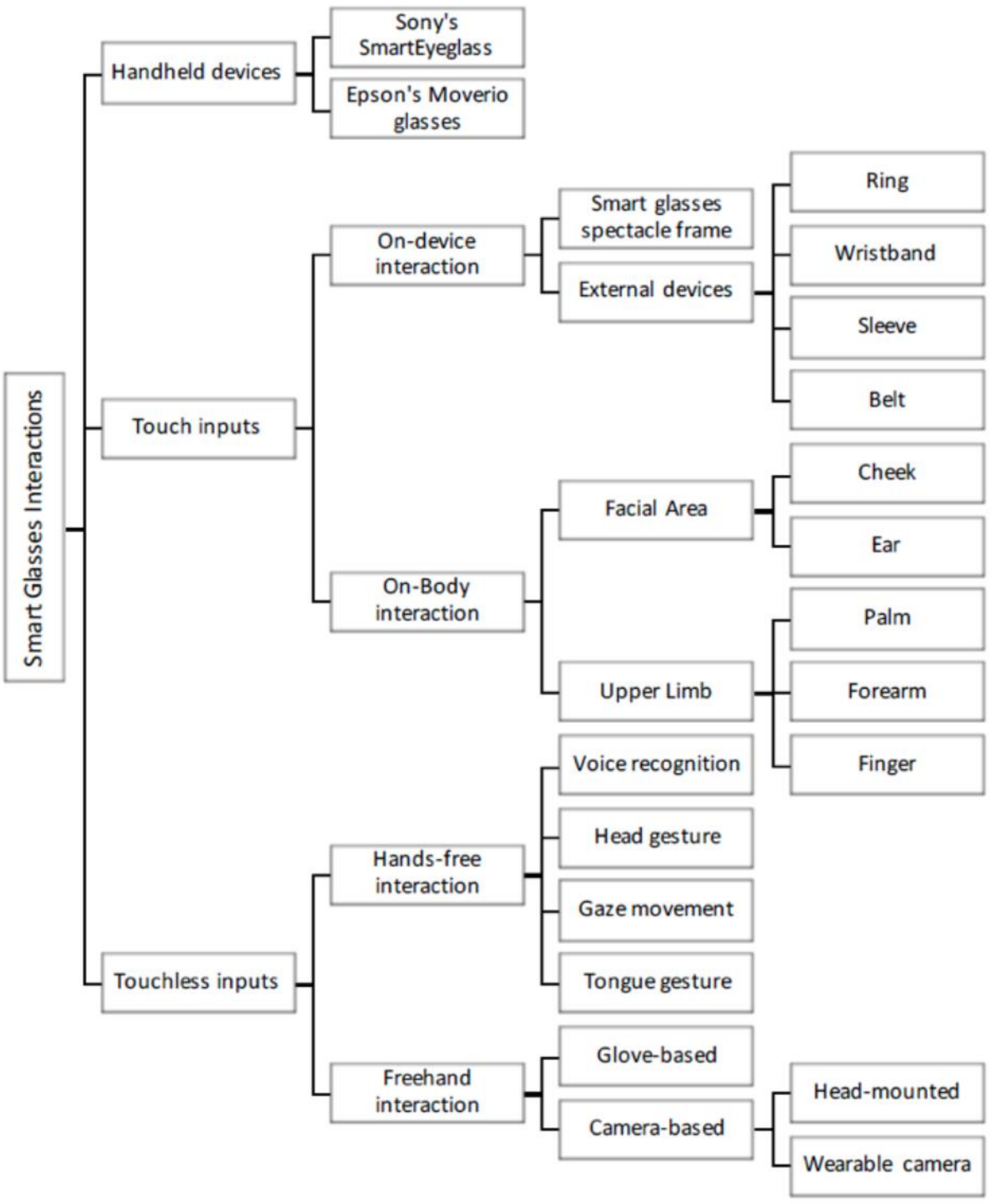


Figure 1-2: Classification of interaction approaches for smart glasses [2].

To minimize the need for additional devices, we prioritize touchless methods. As classified by Lee and Hui (2018) [2], touchless input can be divided into two types: hands-

doi:10.6342/NTU202302797

free interaction and freehand interaction. Hands-free interaction involves voice recognition, head movements, gaze movements, and tongue movements. For our project, we will concentrate on freehand interactions based on vision-based gestures.

1.2 Hand Gestures

In order to provide users with a satisfactory experience, appropriate gesture design becomes especially important. There are two common approaches to design a gesture language for user interactions: using the User Defined Interfaces (UDI) methodology to elicit gestures, or designing a language that avoids conflicts, is ergonomic, and carries high bandwidth (Aigner, et al., 2012) [3]. UDI aims to recognize gestures that users intuitively communicate with. However, although UDI improves the novice "first guess", it may produce inconsistent results since the same gesture may have a different purpose among users, or different gestures may be used to achieve the same result. Aigner et al. [3] developed a methodology similar to UDI, but with to reduce ambiguity. They defined 10 target gesture effects and categorized gesture types into 5 distinct categories.

doi:10.6342/NTU202302797

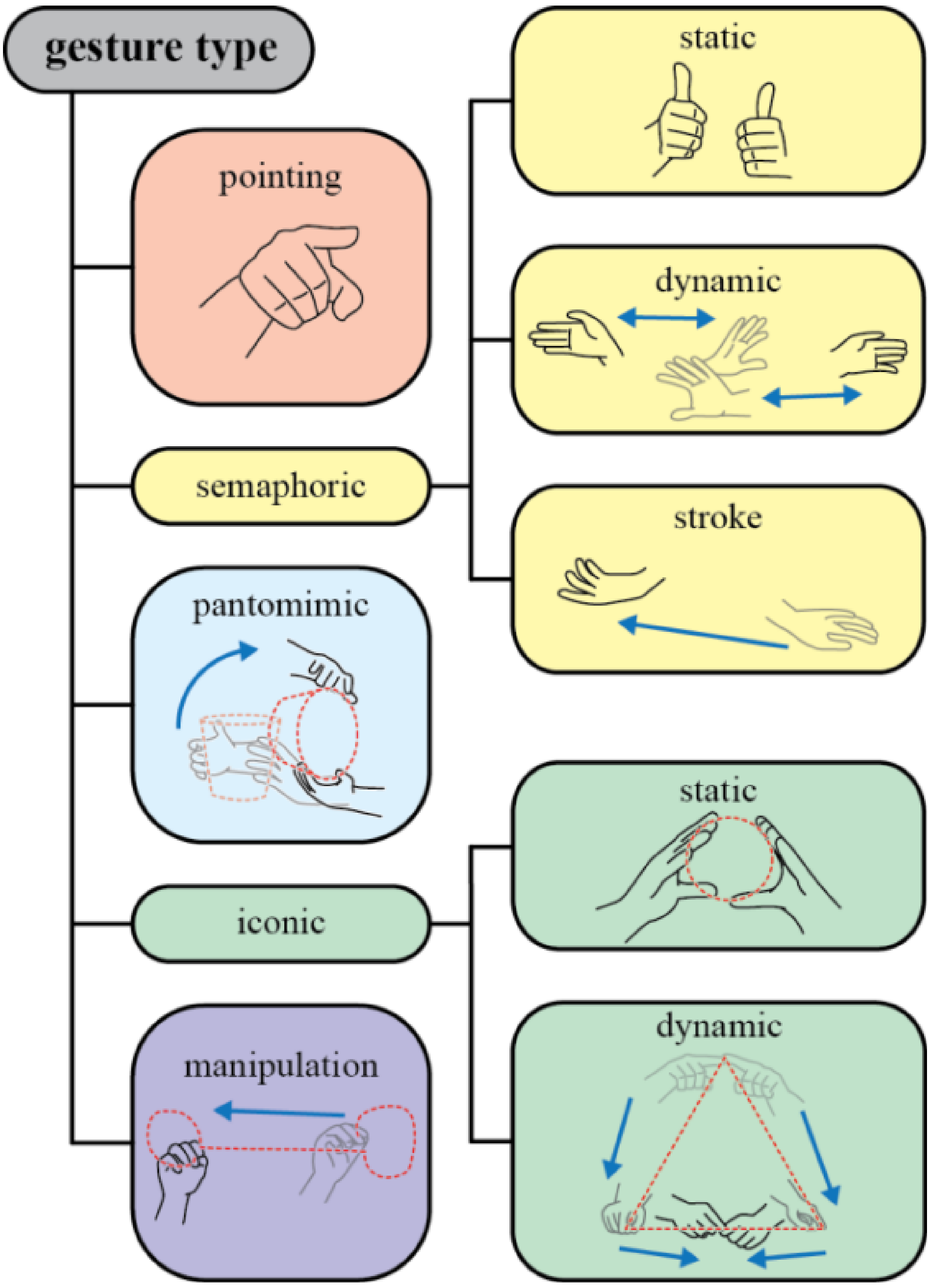


Figure 1-3: The classification Aigner et al. used to communicate information about objects or entities, such as specific sizes, shapes, and motion paths [3].

doi:10.6342/NTU202302797

The 10 target effects include *select, release, accept, refuse, remove, cancel, navigate, identify, translate,* and *rotate*. The 5 gesture types include *pointing, semaphoric, pantomimic, iconic,* and *manipulation*. Semaphoric gestures can be further classified into three types: *static, dynamic,* and *stroke*.

After a thorough understanding of the gesture design system, we can analyze the required effects and corresponding gestures from the perspective of UDI, even when there are hardware specifications limitations.

1.3 Thesis Organization

The related works about ChungGesture are briefly introduced in Chapter 2. The definition of our six gestures and methodology will be described in Chapter 3. The experimental results will be presented in Chapter 4. The last chapter contains the conclusion of this thesis and some future works.

doi:10.6342/NTU202302797

Chapter 2 Related Works

2.1 History of Smart Glasses

According to P. A. Rauschnabel, A. Brem, and Y. Ro [7], a framework for the evolution of media is proposed, consisting of five distinct generations of media. Figure 2-1 illustrates the temporal dimension along the *x*-axis and the influence of each media generation's technologies on users' lives along the *y*-axis.

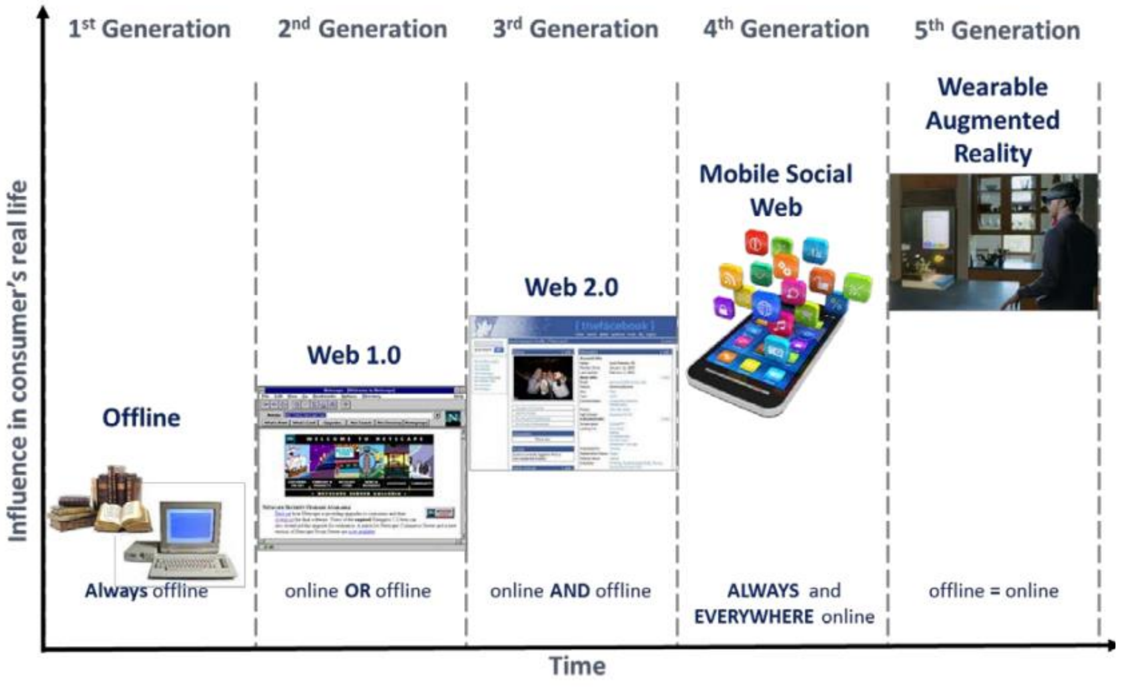


Figure 2-1: Evolution of media devices [7].

The first generation of media, referred to as offline media, encompassed stationary

doi:10.6342/NTU202302797

and uni-directional platforms such as newspapers, television, and others. These technologies relied on internal storage, cartridges (e.g., game consoles), CD-ROMs (Compact Disk – Read-Only Memory), or analog radio frequencies (e.g., TV (TeleVision) or radio) for information delivery.

The second generation, known as Web 1.0, emerged as early online technologies with static websites being prominent examples. Users in this generation predominantly consumed content produced by professional organizations, assuming a passive role. Although limited two-way communication was feasible, most Web 1.0 technologies remained uni-directional. Early websites largely served as digital brochures, primarily created by professional organizations. A few innovative individuals manually programmed HTML (Hyper-Text Markup Language) code to develop personal websites.

The third generation, which began in the early 2000s, is referred to as Web 2.0 or social media. Social media platforms enabled complex and multi-directional communication, transforming users into both consumers and producers of content (often called 'prosumers'). Factors such as faster Internet connections, user-friendly devices, and increased trust in the Internet contributed to the widespread adoption of Web 2.0 technologies. Early examples of Web 2.0 technologies include Facebook, SecondLife, and Myspace.

doi:10.6342/NTU202302797

The fourth generation extended social media from static devices to mobile devices, including laptops, tablets, and smartphones. This generation also encompasses wearable devices such as smart watches, smart clothing, and smart wristbands. Mobile technologies grant users constant access to their social media environment, resulting in social media applications such as Facebook and Instagram becoming highly popular smartphone apps.

The fifth generation of media introduces Wearable Augmented Reality Devices (WARD), which merge virtual and physical realities. These technologies integrate virtual elements into the real world, with augmented reality smart glasses being a notable example and the primary focus of this research.

P. A. Rauschnabel, A. Brem, and Y. Ro. [7] defined Augmented Reality Smart Glasses as wearable Augmented Reality (AR) devices that are worn like regular glasses and merge virtual information with physical information in a user's view field.

The first Head-Mounted Display (HMD) driven by graphics was pioneered by Ivan Sutherland in the 1960s [8]. Over the years, the term HMD has also been used in military applications to refer to helmet-mounted displays integrated with military helmets. Designing an ergonomically optimized headband to securely fasten the HMD to the user's head poses a significant challenge for HMD designers. Due to factors such as weight, weak processors, short battery life, and small screen size, HMD devices have faced

doi:10.6342/NTU202302797

challenges in achieving widespread adoption in the market.

In recent years, commercial HMD devices have been dedicated to enhancing performance and reducing weight in order to achieve greater convenience and comfort. A prime example of this trend is the Google Glass [9].

Table 2-1. Specification of Google Glass Enterprise Edition 2 [9], Microsoft HoloLens 2 [11], and Epson Moverio BT-350 [12].

Item	Google Glass Enterprise Edition 2	Microsoft HoloLens 2	Epson Moverio BT-350
Resolution (Pixels)	640×360	2048 x 1080	1280 x 720
Storage	32 GB	64 GB	48 GB
CPU	Qualcomm XR1 1.7 GHz Quad-core	Qualcomm Snapdragon 850	Intel Atom X5 (1.44 GHz Quad-Core)
RAM	DDR4 3 GB	4 GB LPDDR4x system DRAM	2 GB
Battery	800 mA·h	16,500 mA·h	2,950 mA·h
Camera	8-megapixel camera	8-megapixel camera, 1080p video recording	5-megapixel camera
Operating System	Android Open-Source Project 8.1 (Oreo)	Windows 10	Android 5.1
Weight	46 g	566 g	151 g
Sensor	Wi-Fi Bluetooth GPS 6-axis gyroscope	Azure Kinect sensor accelerometer Gyroscope Magnetometer 6-DoF (Degrees of Freedom) GPS	Wi-Fi Bluetooth GPS 3-axis gyroscope
Fluoroscopy method	Curved mirror (+Reflective waveguide): Google Light pipe	Holographic waveguide	Reflective waveguide: Epson light guide

doi:10.6342/NTU202302797

Table 2-1 presents the product specifications of three commonly found smart glasses in the market. Among them, Google Glass Enterprise Edition 2 is the most widely used, with a specific focus on utility in industrial sectors such as aviation and medicine. Microsoft HoloLens creates a Mixed Reality (MR) environment that enables interactive experiences akin to physically touching virtual objects based on a depth sensor equipped with artificial intelligence. HoloLens extends beyond industrial applications, finding use in operating rooms as well as construction and maintenance sites. In contrast, BT-350 requires a wired controller connected to the main unit for operation, which can lead to user discomfort [10].

2.2 The Method of Depth Estimation

We use ToF camera: depth sensing camera. Therefore, in this section, we will delve into the study of depth prediction techniques, which can be mainly classified into three methods: stereo vision, structured light, and ToF camera. We will provide a detailed introduction to each of these methods, with particular emphasis on ToF camera. Table 2-2 presents a comparison of these three major 3D imaging technologies.

doi:10.6342/NTU202302797

Table 2-2. Comparison of 3D imaging technologies [21].

	Stereo Vision	Structured Light	Time-of-Flight
Software Complexity	High	Medium	**Low**
Material Cost	**Low**	High	Medium
Compactness	**Low**	High	**Low**
Response Time	Medium	Slow	**Fast**
Depth Accuracy	Low	**High**	Medium
Low-Light Performance	Weak	**Good**	**Good**
Bright-Light Performance	**Good**	Weak	**Good**
Power Consumption	**Low**	Medium	**Scalable**
Range	Limited	**Scalable**	**Scalable**

2.2.1 Stereo Vision

Stereo vision, as emphasized by R. Szeliski [19], has emerged as a highly active technology in the realm of computer vision research. It pertains to the task of inferring the three-dimensional structure of a scene based on multiple digital images captured from distinct viewpoints [16]. In Figure 2-2, by employing a mathematical solution, the depth

doi:10.6342/NTU202302797

of an object can be determined by analyzing the triangle formed from the intersection point generated by projecting the object onto two distinct lenses.

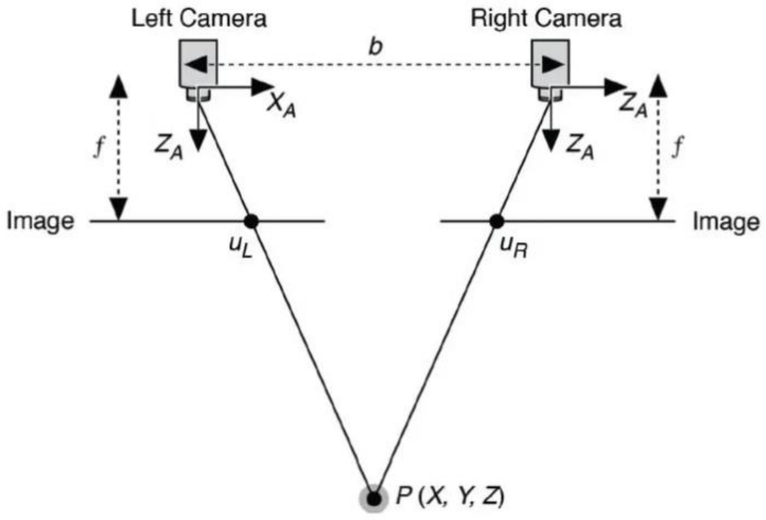


Figure 2-2: Stereoscopic vision diagram [20].

2.2.2 Structured Light

A structured light camera is a type of depth-sensing camera that uses a structured pattern of light projected onto a scene to determine depth information. The camera emits a known pattern of light, such as a grid or a series of horizontal or vertical lines, onto the objects or surfaces in its field of view [22]. By capturing the reflected or deformed pattern of light using specialized sensors, the camera can analyze the distortions in the pattern and calculate the depth of various points in the scene. In Figure 2-3, structued light relies

doi:10.6342/NTU202302797

on the principle of triangulation, where the known pattern of light is projected from a known position, and the camera observes the resulting deformation or displacement of the pattern on the objects or surfaces. By analyzing the distortions, the camera can reconstruct a depth map, providing information about the distance of each point from the camera.

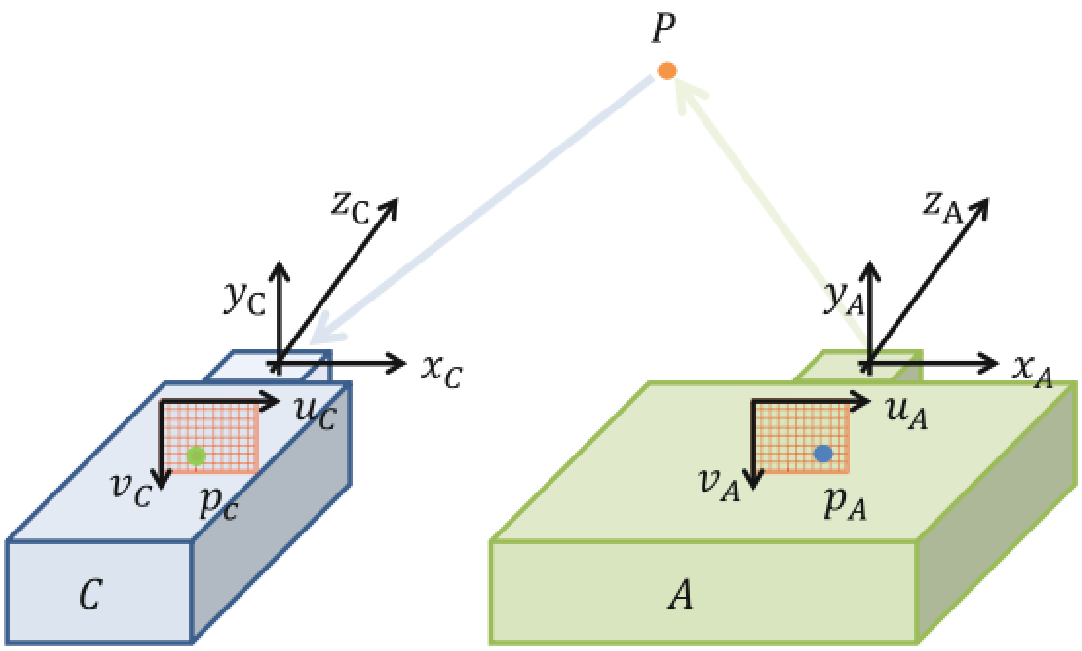


Figure 2-3: Structured light camera diagram [23].

doi:10.6342/NTU202302797

2.2.3 Time-of-Flight Camera

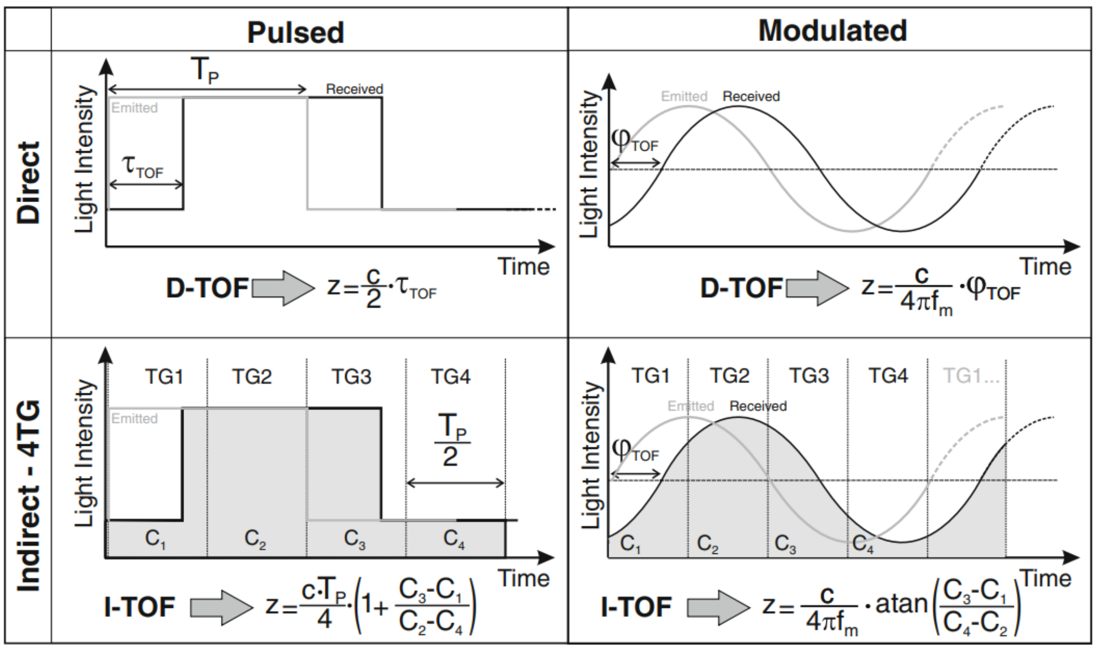
According to F. Remondino and D. Stoppa et al. [4], ToF camera can be categorized into two main types: Direct Time-of-Flight (D-ToF) and Indirect Time-of-Flight (IToF).

D-TOF is typically employed in single-point range systems, specifically in scannerless ToF systems. It is well-suited for Single Photon Avalanche Diode (SPAD)-based systems. In the case of Direct-ToF (D-ToF) measurement, the detector system initiates a highly precise stopwatch simultaneously with the emission of light pulses from the emitter. When the stopwatch is stopped, the roundtrip time τ_ToF is directly recorded. The target distance *z* can then be estimated using a simple equation:

$$Z = \frac{c}{2} \cdot \tau_{ToF}$$

An alternative solution to D-ToF is known as Indirect-ToF (I-ToF), which involves extrapolating the roundtrip time from a time-gated measurement of light intensity. In this case, a highly accurate stopwatch is not required. Instead, time-gated photon counters or charge integrators are used, which can be implemented at a pixel level with less complexity and silicon area. I-ToF is particularly suitable for ToF cameras based on electronic and photo-mixing devices. The operation principle of D-ToF and an example of a four-gates I-ToF are in Figure 2-4.

doi:10.6342/NTU202302797



Pulsed
Modulated
Direct
Indirect - 4TG
Light Intensity
Time
T_P
Emitted
Received
τ_{TOF}
φ_{TOF}
D-TOF $z=\frac{c}{2}\cdot\tau_{TOF}$
D-TOF $z=\frac{c}{4\pi f_m}\cdot\varphi_{TOF}$
TG1
TG2
TG3
TG4
TG1...
$\frac{T_P}{2}$
C_1
C_2
C_3
C_4
I-TOF $z=\frac{c\cdot T_P}{4}\cdot\left(1+\frac{C_3-C_1}{C_2-C_4}\right)$
I-TOF $z=\frac{c}{4\pi f_m}\cdot\mathrm{atan}\left(\frac{C_3-C_1}{C_4-C_2}\right)$

Figure 2-4: Overview of pulsed and modulated D-ToF and I-ToF measuring techniques [4].

As stated by P. Padmanabhan and C. Zhang et al. [5], dToF image sensors based on Time-Correlated Single-Photon Counting (TCSPC) offer high-speed performance and accurate ranging capability, and they have been extensively studied in recent years. To obtain stable depth information, the dToF approach involves detecting events across multiple laser pulses directed at the target, which are then captured by the SPAD sensor depicted in Figure 2-5. This method also mitigates the issue of multiple reflections that can impact iToF cameras.

doi:10.6342/NTU202302797

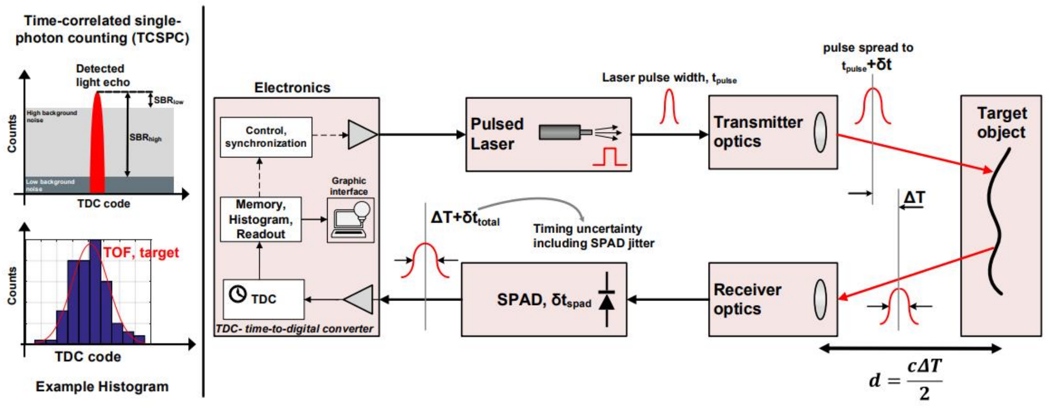


Figure 2-5: Principle of dToF sensor [5].

According to [6], dToF technology offers the advantage of more stable depth information compared with iToF. However, it also has certain drawbacks. For instance, operating a Single Photon Avalanche Diode (SPAD) in dToF requires a high bias voltage (>10 V), and shrinking the pixel size is challenging, resulting in a limited pixel count of only tens of thousands. In contrast, iToF can be implemented on a standard Complementary Metal-Oxide Semiconductor (CMOS) sensor using lower power (<3.3 V), making it easier to shrink the pixel size and achieve a larger pixel count within a smaller optical format. Additionally, SPAD technology is more expensive than iToF CMOS Image Sensor (CIS). A detailed comparison between dToF and iToF is presented in Table 2-3.

doi:10.6342/NTU202302797

Table 2-3. Comparison between dToF and iToF.

	dToF	iToF
Principle of operation	Time-dependent	Phase-dependent
Sensor	SPAD array	iTOF CIS
Difficulty of making	difficult	Easy
Power consumption	Low	High
Measurement accuracy	Stable	Depends on distance
Multiple Reflections	Easy to solve	Difficult to solve
Price	Expensive	Cheap
Application	iPhone 13	Samsung Galaxy

2.3 Hand Gestures Recognition

Hand gesture recognition plays a crucial role in facilitating natural and intuitive user interfaces for Human-Computer Interaction (HCI), offering users an effortless and user-friendly experience [13].

According to M. Oudah, A. Al-Naji, and J. Chahl [15], there are two commonly

doi:10.6342/NTU202302797

used approaches for interpreting gestures in HCI applications. The first approach utilizes data gloves, either wearable or in direct contact, while the second approach relies on computer vision without the requirement of wearing any sensors. The second approach can be further classified into various categories: color-based recognition, appearance-based recognition, motion-based recognition, skeleton-based recognition, depth-based recognition, 3D model-based recognition, and deep-learning based recognition. We will focus on depth-based recognition.

In Figure 2-6, J. Suarez and R. R. Murphy proposed a system for depth-based hand gesture recognition [14].

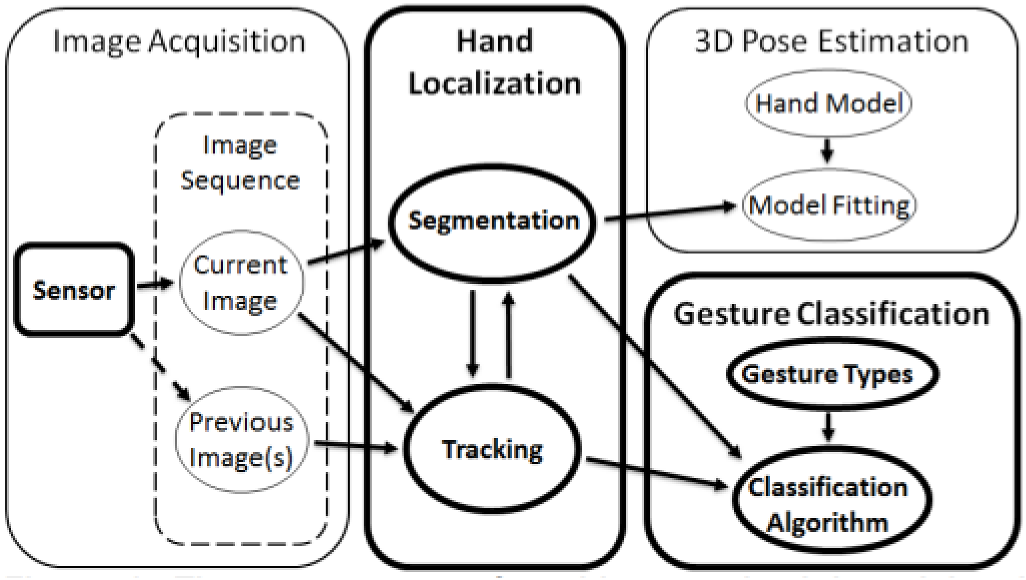


Figure 2-6: The components of a video- or depth-based hand gesture recognition and pose estimation system [14].

doi:10.6342/NTU202302797

Depth-based hand gesture recognition starts by capturing depth images, which relies on the specific sensor employed. Subsequently, hand localization is conducted on the obtained sequence of images through tracking and segmentation techniques. Finally, the segmented hand images and/or their tracked trajectories are classified into specific gestures or poses, utilizing a predefined set of gestures.

doi:10.6342/NTU202302797

Chapter 3 Background

3.1 Our Device Configuration

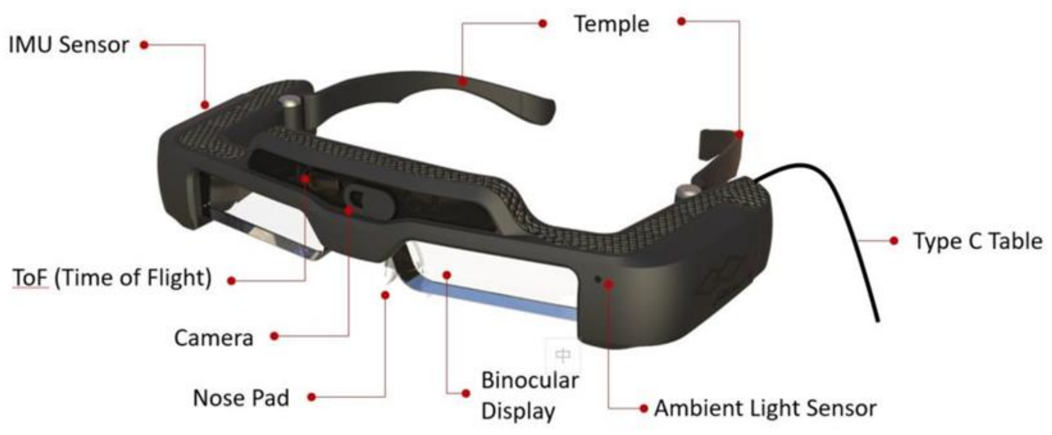


Figure 3-1: Jorjin J-Reality J7EF Plus [18].

The smart glasses used in this experiment are Jorjin J7EF Plus connected to a Sony Xperia 1 II smartphone with operating system Android 12.

doi:10.6342/NTU202302797

Figure 3-2: Sony Xperia 1 II [24].

Jorjin J7EF Plus has a 1080p (1,920x1,080 pixels) binocular display with horizontal FoV (Field of View) of 29.6 degrees. Its Inter-Pupillary Distance (IPD) is 65 millimeters. It is also equipped with 4 sensors, including an 8-megapixel RGB (Red, Green, Blue) camera, a 64 (=8x8) pixels Time-of-Flight (ToF) depth camera (ST VL53L5CX) with 15 FPS (Frames Per Second), ambient light sensor, and an IMU (Inertial Measurement Unit) consisting of an accelerometer, gyroscope, and magnetometer. The gesture recognition algorithms are run on a Micro-Controller Unit (MCU: ST STM32F401CE).

Jorjin J7EF Plus relies on a host device to provide both content and power. To ensure proper functionality, certain factors must be considered when selecting the host device:

- DP Alt (DisplayPort Alternate) Mode: a special mode defined by USB-IF

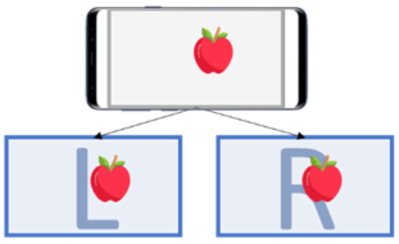
doi:10.6342/NTU202302797

(Universal Serial Bus - Implementers Forum) to transmit high-definition video over USB-C ports, is the main protocol used for graphic transmission to the glasses. The host device must support DP Alt Mode to display content on Jorjin J7EF Plus glasses.

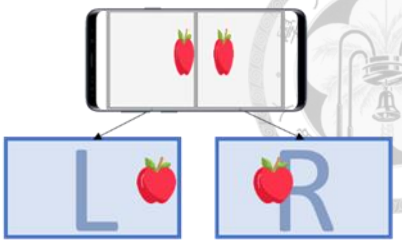
- Power output: another important factor for the glasses. A standard USB 2.0 port can only supply a maximum of 500mA current to a client device. A Power-Delivery enabled USB port can output a higher current (1.5A or more) to the connected device. If the host device cannot provide sufficient power to the glasses, unstable connections and abnormal functionalities may occur during operation.

Jorjin J7EF Plus also has 2D and 3D display modes. In 2D mode, users will see the same content on the left and the right displays on the glasses, and the aspect ratio on the glasses' displays will be the same as the original one on the host display. In 3D mode, users will see different content on the left and the right displays, and the aspect ratio on the glasses displays will be twice as the original one on the host display since the original content will be split in half then be stretched to the normal size before displaying on the glasses. In our experiment, we only consider 2D mode.

doi:10.6342/NTU202302797



(a)



(b)

Figure 3-3: Jorjin J7EF Plus display mode. (a) 2D mode. (b) 3D mode (Jorjin, 2023) [18].

3.2 Sensor Decision

In the selection of sensors, we opted for a ToF camera instead of an RGB camera. This choice was made due to the ToF camera's ability to easily eliminate background noise. In traditional RGB camera-based gesture recognition, foreground objects need to be separated from the background. This can be easily achieved when the background remains static. However, with smart glasses worn on the user's head, the background is constantly changing, making this task more challenging. By using a ToF camera, we only need to retain the pixels within the range where we anticipate the user will perform gestures, while ignoring all pixels outside this range. Another advantage is the accuracy

doi:10.6342/NTU202302797

in detecting depth variations. RGB cameras can only estimate distance variations by observing minor shape and feature differences, leading to relatively larger errors. In contrast, ToF cameras inherently possess depth information, providing a distinct advantage when designing gestures that are perpendicular to the sensor plane.

3.3 Hardware Limitation

3.3.1 The Low-Resolution Depth Sensor

Due to the limited computational power of the MCU on smart glasses, it is not as powerful as a typical CPU installed in desktop computers. Furthermore, our research focuses on low cost and low power consumption. Therefore, we choose a ToF camera with a resolution of only 8x8 as our sensor. We aim to achieve precise, fast, and comfortable gestures. However, due to the low resolution, it is not possible to accurately recognize the shape of a palm, let alone individual fingers. As a result, we exclude static gestures such as finger counting and instead choose motion-based gestures with larger displacement amplitudes, such as swiping.

doi:10.6342/NTU202302797

3.3.2 Limited Field of View (FoV)

Due to the placement of our sensor on the smart glasses, specifically on the user's forehead with a sensing area directly in front of the eyes, users need to raise their hands slightly higher to keep them within the sensing range. Besides, to ensure gesture comfort, we design our gestures to minimize upper arm movements and instead focus on movements of the forearm.

3.3.3 Sensor Mobility Constraint

Due to the placement of the sensor on the user's head in smart glasses, the sensing area is influenced by head movements and rotations. Even slight shaking can result in static objects being detected as moving objects. Furthermore, if the head rotates too much, it may cause the hand to move out of the sensing area, interrupting an incomplete gesture. Therefore, when designing gestures, we took into account the importance of interactivity.

3.4 Existing Techniques and Limitations

3.4.1 LuGesture Method

The previous development of Jorjin J-Reality J7EF Plus Smart Glasses was mainly

doi:10.6342/NTU202302797

based on Y. H. Lu's work [17]. LuGesture focuses on hand gesture recognition using ToF technology integrated into the smart glasses. LuGesture enables the recognition of three gesture sets, including:

1. Swipe + Push/Pull: The four swipe directions include up, down, left, and right. These specific gestures have been chosen to emulate the arrow keys on a keyboard. The complete process involves the following steps: First, binarize the pixels based on whether they fall within the sensing range. Then, calculate the difference between the current and previous frames, and apply image blur to distribute the differences more evenly. By analyzing the blurred difference image and identifying the gradient directions, we can determine the swipe direction for each difference frame. When the hand is no longer present or stops moving, we can determine the gesture direction by examining the most frequently occurring direction in the sequence. Alternatively, we can sum the scores of all frames in the sequence to determine the gesture direction.

   The push and pull gestures are designed to be triggered when a hand moves towards or away from the sensor, covering a certain distance. The algorithm consists of two stages: ready stage and trigger stage. In ready stage, we detect the presence of a hand. We calculate the mean distance of the pixels that detect

doi:10.6342/NTU202302797

an object within the sensing range, which serves as the start distance. The algorithm transitions from ready stage to trigger stage when the difference between the current mean distance and the start distance exceeds a specified distance threshold. This transition triggers the push gesture if the current distance is larger, while it triggers the pull gesture if the current distance is smaller. Once the gesture is triggered, the algorithm is reset, allowing for the detection of subsequent gestures.

Due to the possibility of simultaneous triggering of swipe gestures and push/pull gestures, LuGesture assigns higher priority to push/pull gestures.

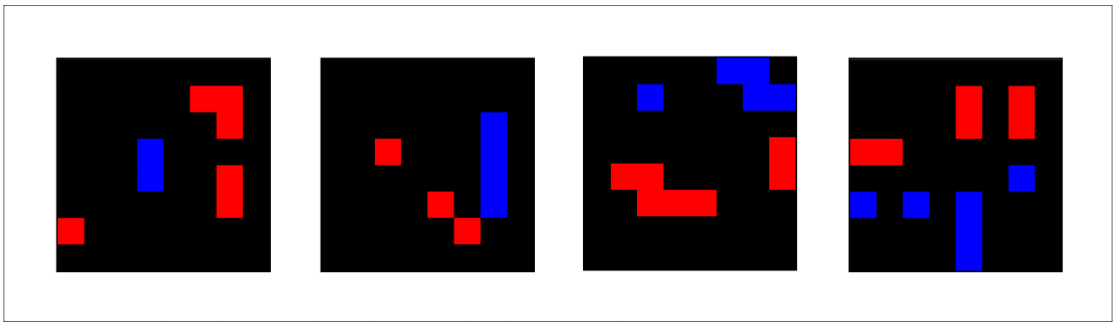


Figure 3-4: Examples of difference images for each direction. (a) Swipe left. (b) Swipe right. (c) Swipe up. (d) Swipe down. The motion direction can be observed by the general direction from red (disappearing: negative values) to blue (appearing: positive values) [17].

doi:10.6342/NTU202302797

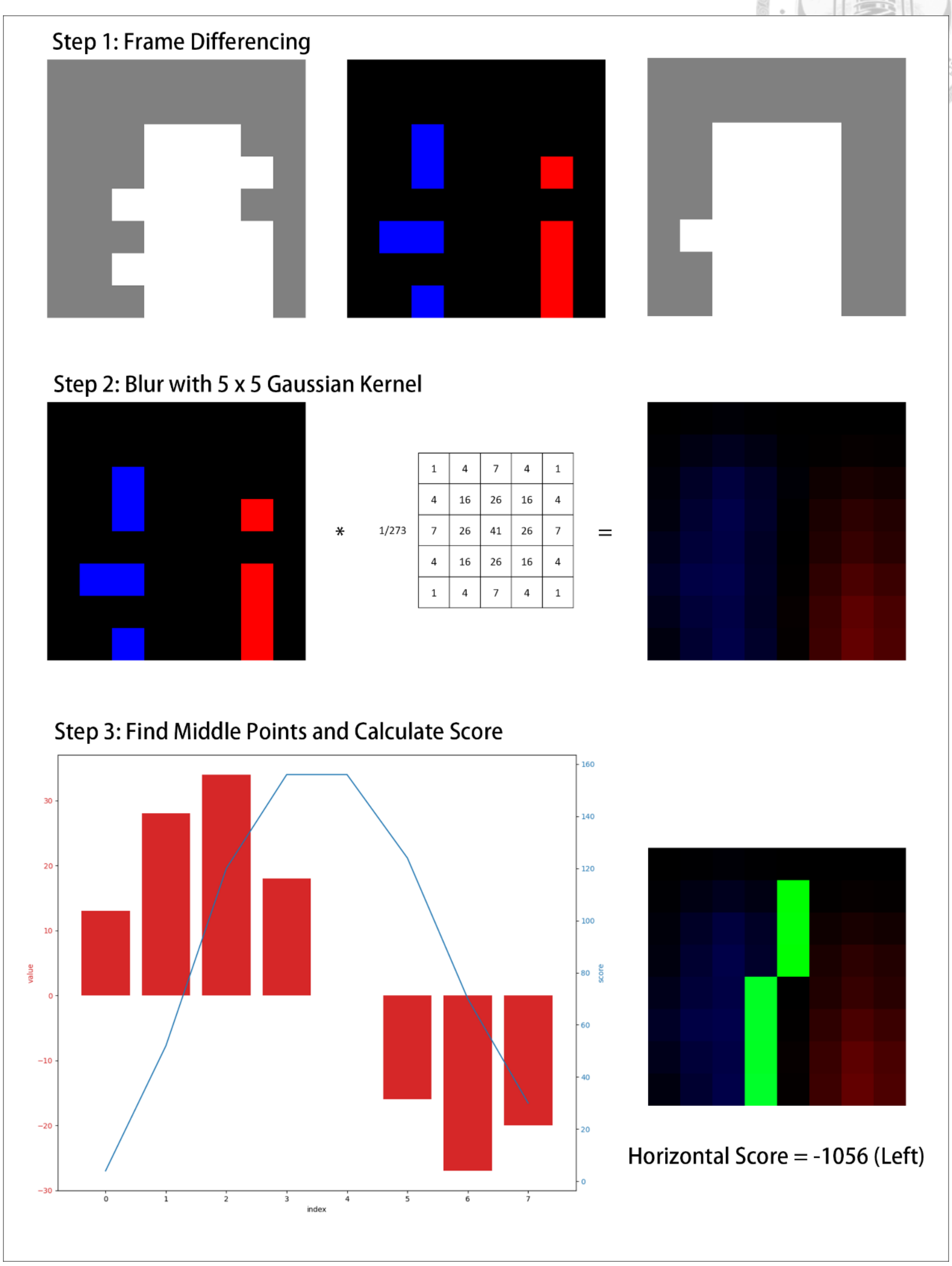

Step 1: Frame Differencing
Step 2: Blur with 5 x 5 Gaussian Kernel
*
1/273
1 4 7 4 1
4 16 26 16 4
7 26 41 26 7
4 16 26 16 4
1 4 7 4 1
=
Step 3: Find Middle Points and Calculate Score
value
score
index
Horizontal Score = -1056 (Left)

Figure 3-5: The whole process to detect swipe gestures. For Step 3, we show the

doi:10.6342/NTU202302797

graph of the 4th row. We can see that the score peaks at index 3 and 4, which is where our middle point is. The same process applies to every row for the horizontal score as shown, and every column for the vertical score [17].

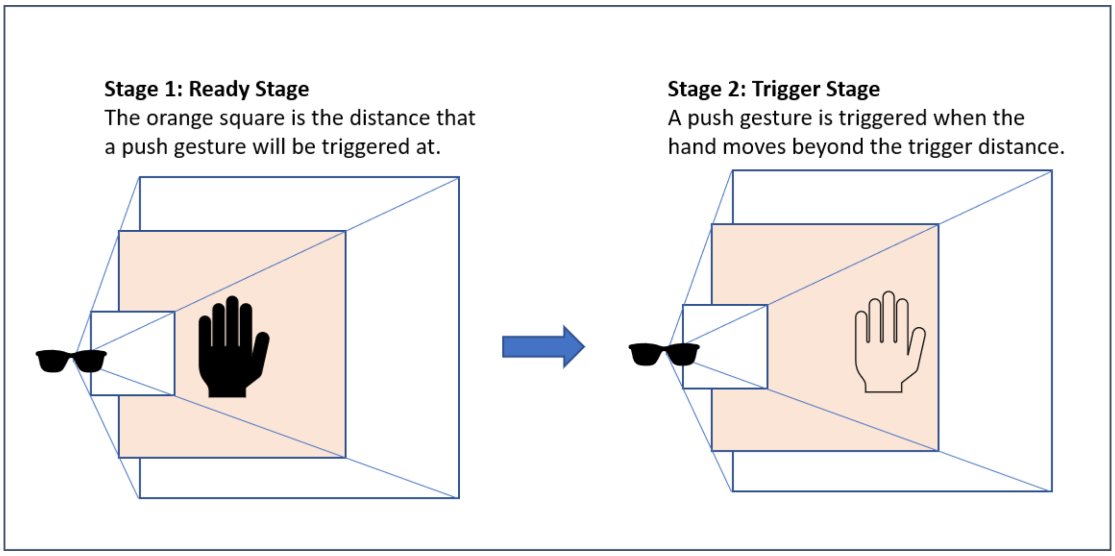

Stage 1: Ready Stage
The orange square is the distance that a push gesture will be triggered at.
Stage 2: Trigger Stage
A push gesture is triggered when the hand moves beyond the trigger distance.

Figure 3-6: Illustration of the two stages of a push gesture [17].

2. Hand Speed: This gesture is primarily designed for throwing games and records the speed at which the hand is pushed forward. Imagine an invisible wall in front of us, with the closer side referred to as the close side and the farther side as the far side. The algorithm can be divided into three stages: ready stage, trigger stage, and sleep stage. In the ready stage, the algorithm detects the hand but waits for the hand to cross the invisible wall, as the number of pixels on the far side is less

doi:10.6342/NTU202302797

than the required threshold for triggering the speed registration. Once the hand crosses the wall, it enters the trigger stage. In the trigger stage, LuGesture calculates the hand speed by subtracting the mean distance between hand pixels in the current and previous frames, capturing the maximum speed since entering the ready stage. During the sleep stage, LuGesture pauses its operation to avoid triggering unnecessary gestures. After the sleep time expires, it can re-enter the ready stage for further detection.

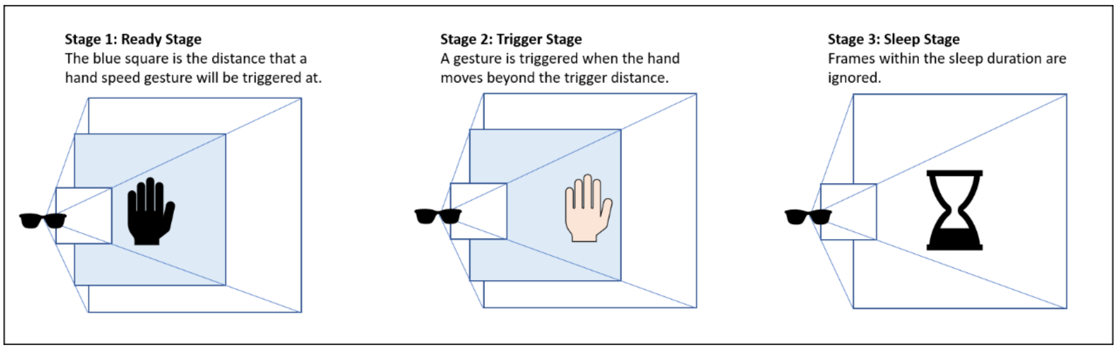
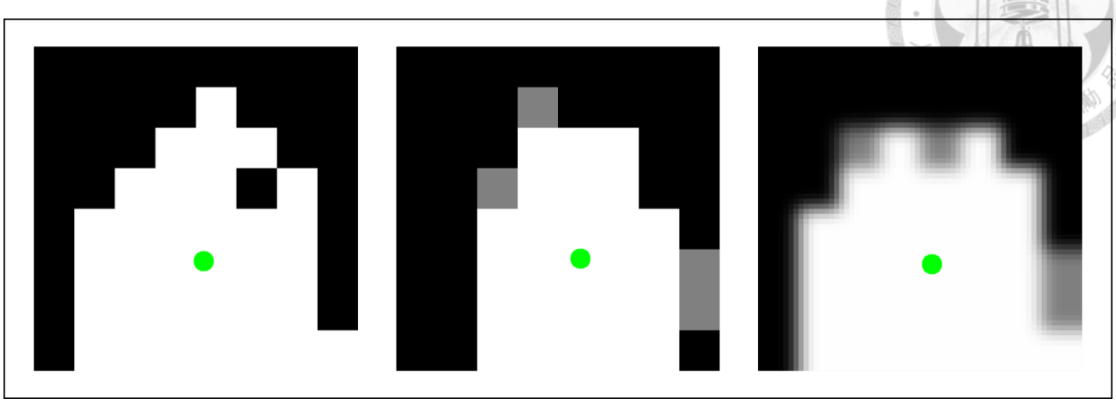

Stage 1: Ready Stage
The blue square is the distance that a hand speed gesture will be triggered at.
Stage 2: Trigger Stage
A gesture is triggered when the hand moves beyond the trigger distance.
Stage 3: Sleep Stage
Frames within the sleep duration are ignored.

Figure 3-7: Illustration of the three stages for hand-speed detection [17].

3. Hand Tracking: This gesture is designed to allow users to use their hands as a cursor even in low-resolution conditions. The complete process involves: binarizing the image based on the sensing range to extract hand pixels, adjusting weights to reduce the impact of twinkle pixels, upscaling the image, applying image blurring, calculating the centroid, and performing interpolation between

doi:10.6342/NTU202302797

frames to improve FPS.



(a) (b) (c)

Figure 3-8: Processing to reduce jitter. (a) Unprocessed binary image. (b) Reweighted image. (c) Upscaled and blurred image after reweighting. Note that these are different frames [17].

3.4.2 Limitations of LuGesture Method

After conducting practical experiments with LuGesture, we observe that the gesture recognition performance did not meet expectations, as shown in Figure 3-9, with frequent instances of misjudgment. Therefore, we analyze and identify potential risks associated with LuGesture:

1. Excessive non-palm region: During Hand Tracking, the hand's centroid struggles to reach the upper region. Even when the palm moves upward, the sensor may still

doi:10.6342/NTU202302797

detect the forearm, thereby affecting centroid calculation.

2. Insufficient stability: The low resolution (8x8 pixels) of the camera has a limited number of pixels. Consequently, even a single pixel can have a significant impact on the centroid calculation. Additionally, the inherent limitations of ToF technology can result in twinkle issues when the palm edges fall between two pixels, greatly impacting stability.

3. Lack of functional integration: Each component of LuGesture operates independently, lacking seamless integration into a comprehensive system.

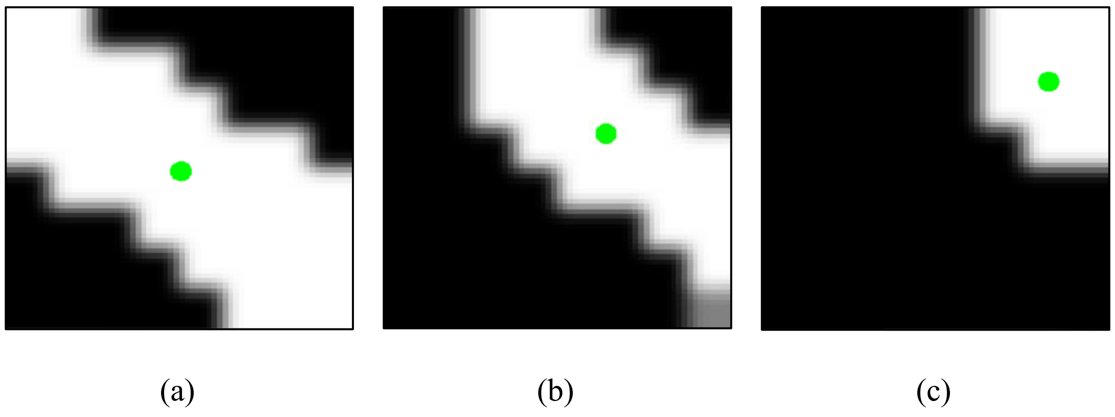


Figure 3-9: The result image of LuGesture. Hand in the (a) top-left corner, (b) upper-center corner, (c) top-right corner.

To address these identified risks, ChungGesture proposes the following solutions to achieve improved results below:

doi:10.6342/NTU202302797

1. Differentiating between left and right hands.
2. Improving precision.
3. Improving stability.
4. Enhancing gesture recognition accuracy.
5. Integrating all components into a unified system.

Detailed explanations of these solutions will be provided in Methodology.

doi:10.6342/NTU202302797

Chapter 4 Methodology

4.1 Overview

Based on the three risks outlined in Chapter 3, we have developed a system that achieves the functionalities of 4 swipe directions, push/pull, and tracking by accurately calculating the centroid of the palm. Figure 4-1 illustrates the main process flow of ChungGesture: First, we apply distance and signal strength filtering. Second, we filter out the background and partial arm by defining a reasonable range of palm movements. After that, we handle the twinkle pixels along the palm edges and determine whether it is the left or right hand. Finally, based on the left/right hand, we filter out the remaining arm regions, leaving only the palm, and calculate its centroid for accurate gesture recognition.

doi:10.6342/NTU202302797

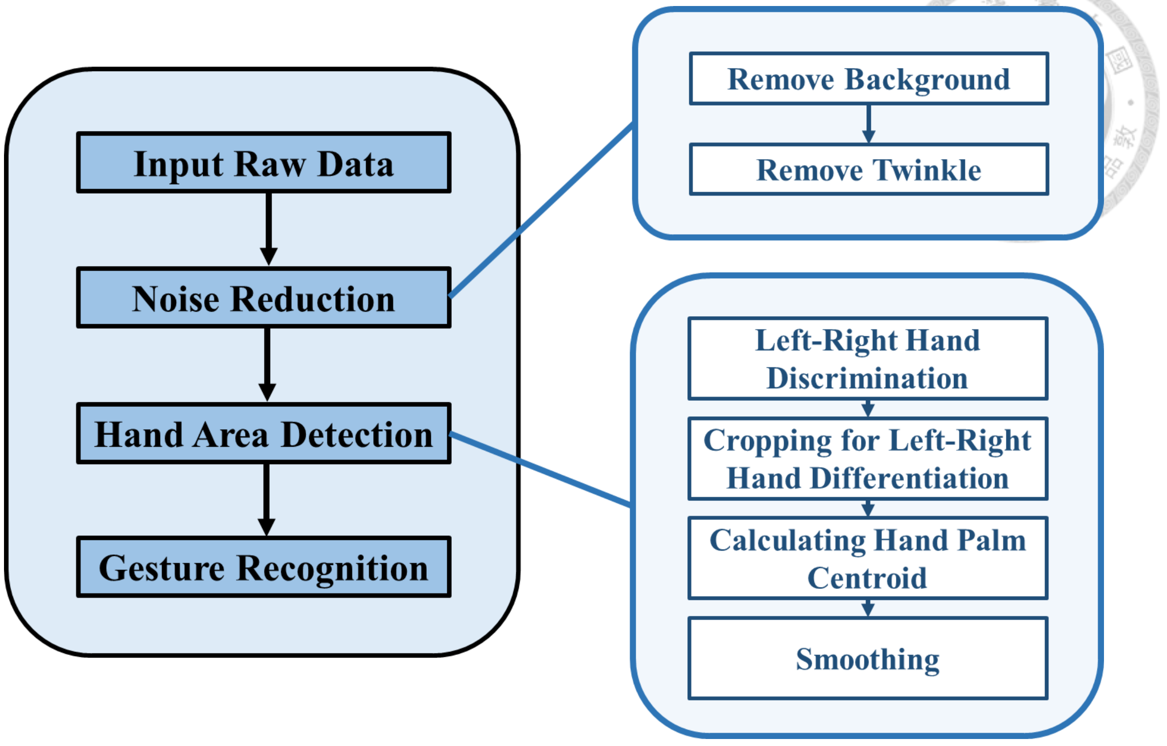


Figure 4-1: Main flowchart of ChungGesture.

4.2 Data Collection

Based on Jorjin Software Development Kit (SDK) in Figure 4-2, depth information and signal strength of the ToF Camera can be obtained by calling the API (Application Programming Interface) named "*onTofIncomingFrame*". The data are stored and transmitted in arrays of size 64 (=8X8 pixels).

doi:10.6342/NTU202302797

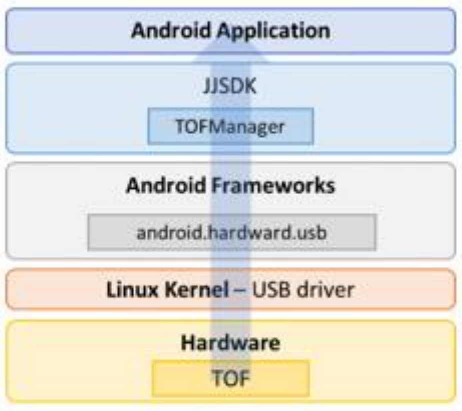


Figure 4-2: The data flow chart of smart glasses [18].

4.3 Noise Reduction

4.3.1 Remove Background

In order to extract only the palm information, the first step after obtaining the depth information and signal strength is to remove background noise. Based on our observations, hand movement range for most individuals falls within the range of 0~45 centimeters. Therefore, pixels outside this range are initially ignored. Additionally, to mitigate distance measurement errors in the ToF camera, ChungGesture also considers the signal strength returned by the ToF camera as an auxiliary parameter. Only the data with signal strength above a certain threshold (=122) is selected to ensure data accuracy. If all pixels are

doi:10.6342/NTU202302797

filtered out, the frame is deemed invalid, and no further processing or calculations are performed on that frame. Conversely, the remaining pixels are referred to as valid pixels.

Generally, when waving the hand, the palm tends to tilt slightly forward, with the fingertips being farthest away from our body in Figure 4-3. Therefore, we use the distance of the farthest valid pixel as the reference and retain the valid pixels within 10 centimeters of this distance. These retained pixels are referred to as hand pixels, while the remaining valid pixels are assumed as arm and ignored. The depth values of all hand pixels are summed and averaged to obtain the depth of the hand.

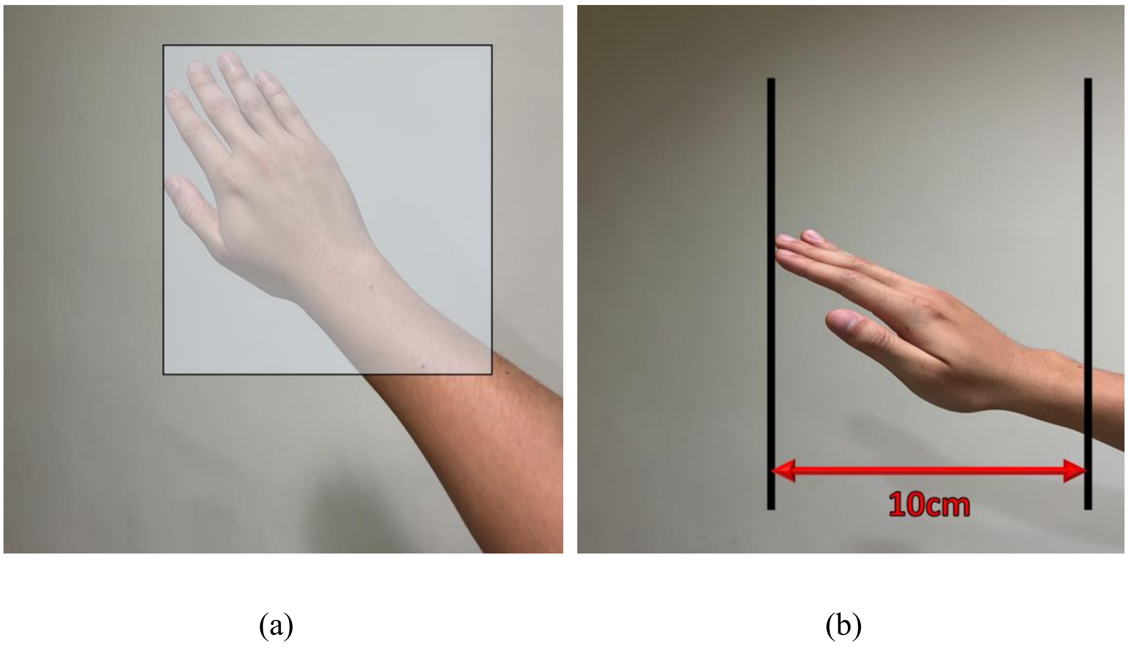
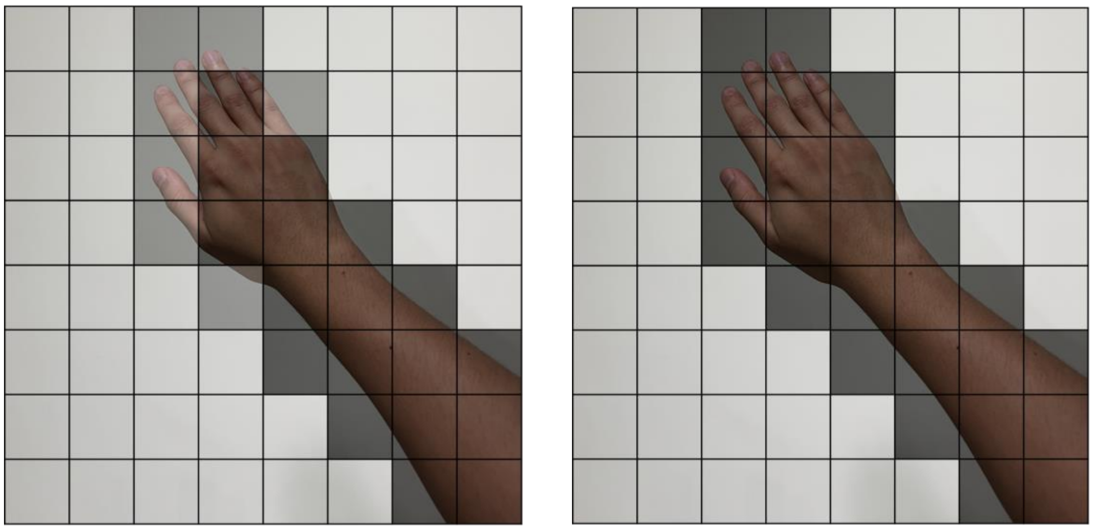


Figure 4-3: Sensing range of hand pixels. (a) Direct view. (b) Side view.

doi:10.6342/NTU202302797

4.3.2 Remove Twinkle

After performing the aforementioned processing steps, we can roughly filter out the shape of the hand. However, due to the accuracy and low resolution of the ToF camera, when the edge of the palm falls between pixels in Figure 4-4 (a), gray pixels are prone to twinkle. Therefore, we keep a record of the hand pixels from the previous 8 frames and compare them with the current frame. If a pixel was present in the past but not in the current frame, or vice versa, we consider those pixels as twinkle pixels.



(a) (b)

Figure 4-4: Comparison chart of twinkle processing. (a) Twinkle image. (b) After twinkle removal.

doi:10.6342/NTU202302797

By considering both the twinkle pixels and the hand pixels together, we classify them based on their connectivity. Connected pixels in Figure 4-4 (b), refer to those pixels that are connected to each other. This approach helps reduce the instability caused by flickering.

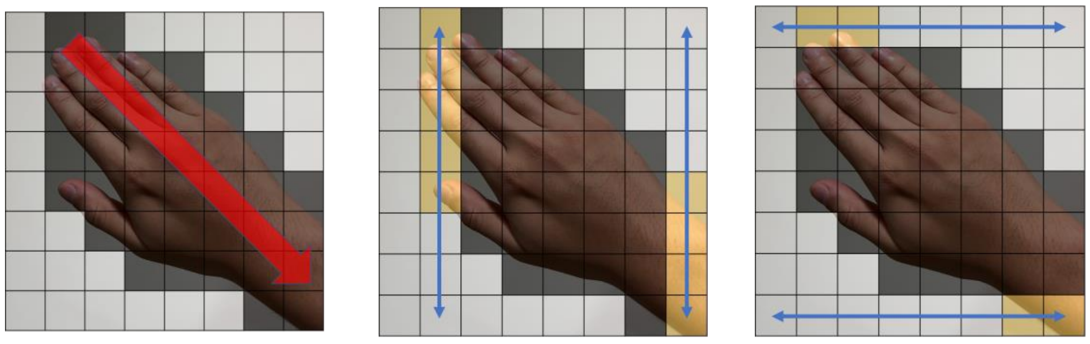
4.4 Hand Area Detection

4.4.1 Left-Right Hand Discrimination

Through observation, we have found that when most people wave their right hand, it tilts to the left, and vice versa in Figure 4-5 (a). Taking the example of the right hand, after performing noise reduction on the frame, we can observe that the connected pixels form a connection from the top left to the bottom right. Leveraging this characteristic, we identify the leftmost and rightmost columns of connected pixels. We calculate the average y-coordinate values of the connected pixels on these two columns in Figure 4-5 (b). Similarly, we identify the topmost and bottommost rows and calculate the average x-coordinate values of the connected pixels on these two rows in Figure 4-5 (c). If the average y-value of the leftmost column is higher than that of the rightmost column, and

doi:10.6342/NTU202302797

at the same time, the average *x*-value of the top row is to the left of the average *x*-value of the bottom row, and this pattern persists for three consecutive frames, then it is determined as a right hand, and vice versa.



(a) (b) (c)

Figure 4-5: Determine left-right hand.

4.4.2 Cropping for Left-Right Hand Differentiation

The size of an object varies with its distance from the eyes, and the same applies to the hand in relation to glasses. The closer the hand is, the larger it appears and the more pixels it occupies. Through repeated observations, we can utilize the average depth calculated in Section 4.3.1 to determine the expected number of pixels for the palm at that

doi:10.6342/NTU202302797

distance in Equations 1 and 2:

$$L = (d \times \tan(\theta)/l) \tag{1}$$

$$h(d) = (W \times H)/L^2 \tag{2}$$

where L represents the actual length of a pixel; d represents the average depth of the hand; θ represents the angle between the line connecting the glasses and the palm and the palm itself; l represents the length of the palm as it appears in the image; h represents the number of pixels for the hand; and W and H represent the width and height of the palm, respectively.

After obtaining the information of the left and right hands along with their corresponding pixel counts, we take the right hand as an example. In Figure 4-6 (a), based on the recorded top-left point (red pixel) mentioned in Section 4.4.1, we follow the sequence indicated in Figure 4-6 (b) to selectively retain the connected pixels while not exceeding the expected number of pixels for the hand. All remaining pixels are filtered out, resulting in a representation of the palm in the form of pixels in Figure 4-6 (c). The yellow pixels are referred to as hand palm pixels, and the blue pixel represents the final pixel we have chosen.

doi:10.6342/NTU202302797

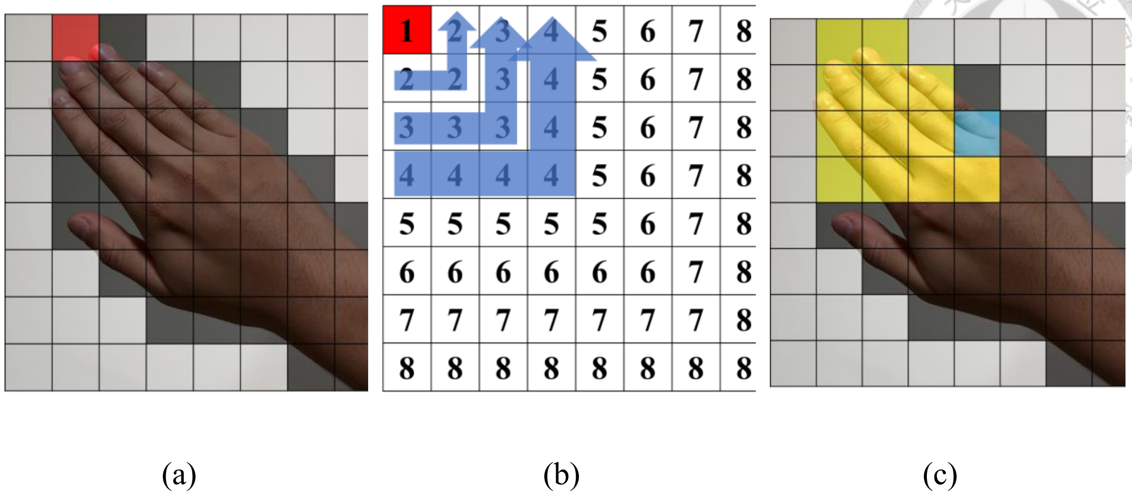


Figure 4-6: Hand palm image.

4.4.3 Calculating Hand Palm Centroid

Now we have obtained the pixels representing the shape of the hand palm, the next crucial step is calculating the centroid. This part is considered the most important because subsequent gesture recognition and tracking are based on the centroid. We evaluate the effectiveness of our methods by precsion and stability. Therefore, we have developed four methods, which will be explained in detail below, and the comparative results will be presented in Chapter 5.

In Method 1, if we simply take the average of the coordinates of hand palm pixels, it can be easily influenced by edge twinkle pixels, even after processing the twinkle pixels.

doi:10.6342/NTU202302797

Therefore, to increase stability, we apply Gaussian weighting to the averaged point using Equations 5 to 7. This helps us reduce the impact of edge twinkle pixels, where points closer to the center (hand palm center) have higher weights, while points farther from the center (hand palm edges) have lower weights. By recalculating the centroid based on these weights, we obtain our hand palm centroid.

Using the aforementioned method still poses a problem, which is the precision. Taking the right hand as an example, when our right palm reaches the top left corner of the sensor detection area, the calculated hand palm centroid is in Figure 4-7. However, if the palm continues to extend towards the top left corner beyond the sensor detection area, the calculated hand palm centroid remains at the same position. We can never reach the actual top left corner.

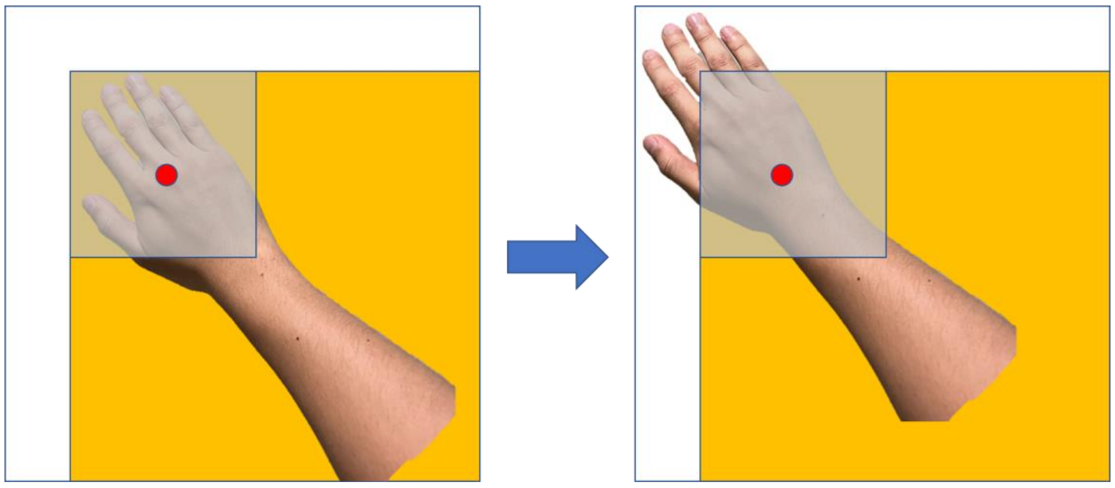


Figure 4-7: Precision problem.

doi:10.6342/NTU202302797

To address the aforementioned issue, we record the hand palm centroid and depth when the hand first reaches the top left corner. We simulate the movement of the hand palm towards the top left corner by pushing it forward. Here, we set a parameter *P* representing the longest distance of the push (5 centimeters). Assuming the hand palm continues to touch the top left corner, we apply Equation 3 to scale the hand palm centroid proportionally:

$$(x,y)_f = (x,y)_c + \frac{(d_c - d_r)}{P} \times ((x,y)_t - (x,y)_c) \tag{3}$$

where, (*x, y*) represents the coordinate values; *f* represents the final result; *c* represents the current frame; *r* represents records when first reaches the corner; *t* represents the target corner; *d* represents the depth; and *P* represents the longest distance of the push. Methods 2, 3, and 4 are also developed to address the mentioned issue.

Method 2 involves proportional scaling, specifically, the resolution is projected from 7x7 to 8x8. The path traveled by the hand is normalized to a length of 8 units. The hand palm centroid, which has undergone Gaussian weighting in Method 1, is proportionally scaled using Equation 4:

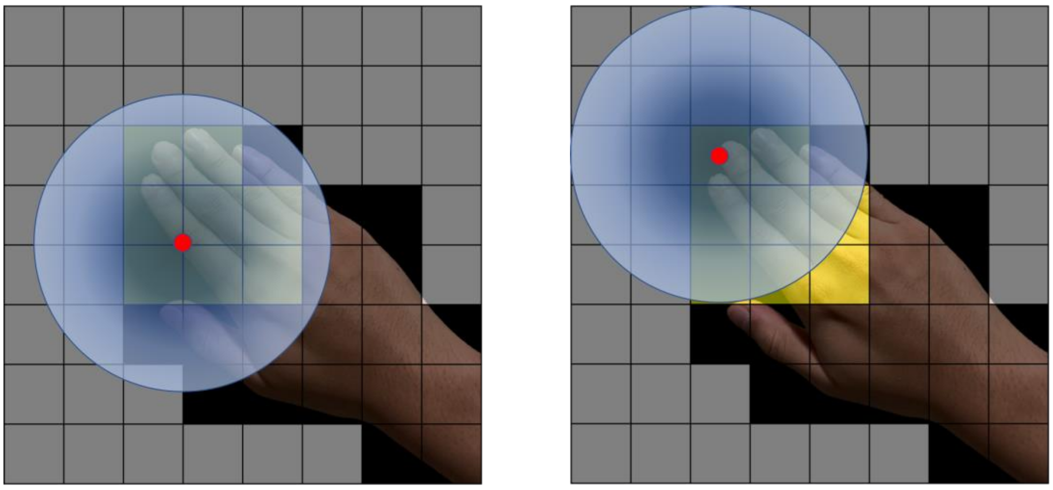
$$(x,y)_{output} = \frac{(x,y) - (x,y)_{min}}{(x,y)_{max} - (x,y)_{min}} \tag{4}$$

where $(x,y)_{max}$ represents the max coordinate of the sensor, which means (7, 7) in this

doi:10.6342/NTU202302797

paper; $(x,y)_{min}$ represents the min coordinate of the sensor, which means (0, 0) in this paper.

Method 3, similar to Method 1, employs Gaussian weighting using Equations 5 to 7. However, the key distinction lies in the fact that we now assign weights based on fingertip position in Figure 4-8.



(a) (b)

Figure 4-8: Gaussian weighting image. (a) Averaged point. (b) Fingertip.

$$D_{ij} = \sqrt{(i - c_x)^2 + (j - c_y)^2} \tag{5}$$

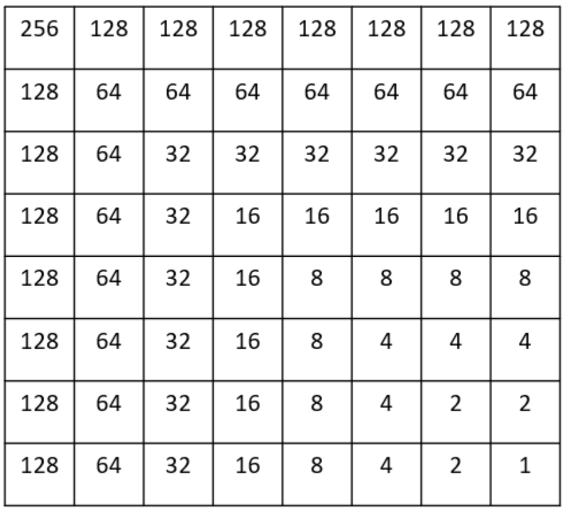
$$W_{ij} = \frac{1}{\sigma\sqrt{2\pi}} e^{-D_{ij}{}^2/(2\sigma^2)} \tag{6}$$

$$C = \left(\frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(i * W_{ij})}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} W_{ij}}, \frac{\sum_{i=0}^{M}\sum_{j=0}^{N}(j * W_{ij})}{\sum_{i=0}^{M}\sum_{j=0}^{N} W_{ij}}\right) \tag{7}$$

doi:10.6342/NTU202302797

where D_{ij} represents the distance between coordinate (i, j) and center coordinate (c_x, c_y); W_{ij} represents the weight from Gaussian; C represents the coordinate after reweighting; and M and N represent the dimension of hand palm pixels matrix.

Method 4 involves directly assigning an 8x8 weight table based on the left- or right-hand determination obtained from Section 4.4.1. The centroid is then calculated using this weight distribution, as illustrated in Figure 4-9.



256	128	128	128	128	128	128	128
128	64	64	64	64	64	64	64
128	64	32	32	32	32	32	32
128	64	32	16	16	16	16	16
128	64	32	16	8	8	8	8
128	64	32	16	8	4	4	4
128	64	32	16	8	4	2	2
128	64	32	16	8	4	2	1

Figure 4-9: Weight table.

4.4.4 Smoothing

doi:10.6342/NTU202302797

To improve stability, we consider the inherent inaccuracies of ToF sensor itself. Even when the hand remains motionless, slight fluctuations can be observed in the depth values provided by the sensor, which can affect our determination of the hand palm centroid. To address this issue, we incorporate a Gaussian weighting technique by considering the hand palm centroids of the previous three frames when calculating the current frame's hand palm centroid. This approach is based on the assumption that hand movements are coherent, and the variations between consecutive frames are not significant. After applying Gaussian weighting, we obtain the final hand palm centroid with improved stability.

4.5 Gesture Recognition

After undergoing the smoothing process, the hand palm centroid allows us to achieve accurate hand tracking. In this section, we will explain how we utilize the hand palm centroid to enable detection of four swipe directions (up, down, left, and right) as well as push and pull gestures.

doi:10.6342/NTU202302797

4.5.1 Swipe

In the case of detecting the four swipe directions, there are two aspects to consider: observation and recognition. First, we aim to achieve recognition by analyzing the user's motion vectors. However, each individual's hand length, hand palm size, motion habits, and speed differ, and there is also a distinction between left and right hands. As a result, the obtained motion vectors will vary. We have previously explained the process of obtaining the hand palm centroid to facilitate our observation and analysis in Figure 4-10. Taking the example of a right-hand swipe left, through repeated observations, we noticed that even for a simple leftward motion, the hand palm centroid enters from the upper-right position of the frame and exits from the lower-left position. We record displacement vector during this process, taking four frames as a unit. Considering the limitation of 15 Frames-Per-Second (FPS), a higher number of frames may introduce delay and misjudgment, while a lower number may overlook slower hand motions. Similarly, we record displacement vectors for swipe right, up, and down, corresponding to the other three directions.

doi:10.6342/NTU202302797

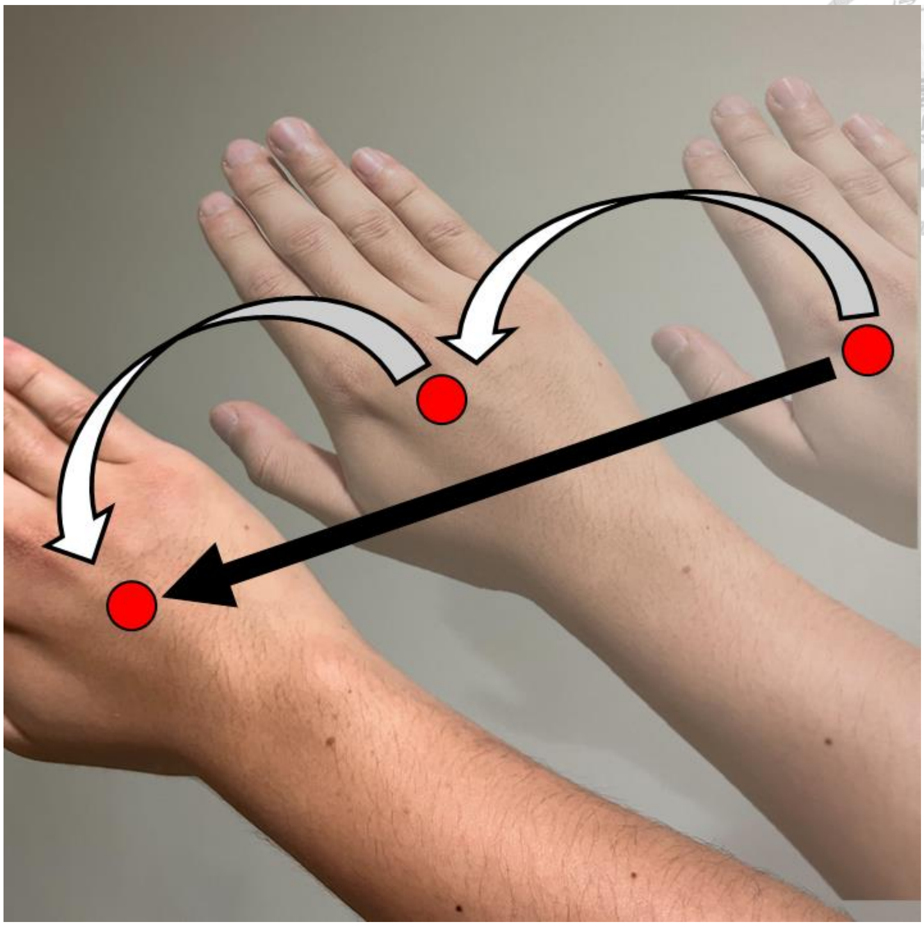


Figure 4-10: Right-hand swipe left image.

Once we have the statistical vectors for the four swipe directions, we have a basis for gesture recognition. When the user actually performs a hand swipe, it generates a displacement vector. If this displacement vector reaches a certain magnitude within four frames, we compute the dot product between this vector and the previously recorded statistical vectors for each swipe direction. Subsequently, we calculate the angle using the following equation:

doi:10.6342/NTU202302797

$$Angle = \cos^{-1}\left(\frac{V_D \cdot V_C}{|V_D| \times |V_C|}\right)$$

where V_D represents the current displacement vector, and V_C represents the previously recorded statistical displacement vectors. The recognized gesture corresponds to the smallest calculated angle. We employ this approach because we consider the statistical vectors as ground truth, and calculating the angle between the displacement vector and the statistical vector is akin to measuring the similarity to the corresponding gesture.

4.5.2 Push and Pull

Next, we discuss the push/pull gestures, which involve movements perpendicular to the plane of the ToF sensor sensing area. These gestures are closely tied to our depth information. The determination of these two gestures primarily relies on the hand's depth in Section 4.3.1. Their triggering conditions are associated with depth variation. We have set two parameters: a depth variation threshold for triggering push and pull gestures. The threshold for push is set to 12 centimeters, while the threshold for pull is set to 8 centimeters. Similar to the swipe gestures, we consider a window of four frames. We detect whether the hand's depth variation exceeds the threshold within these four frames. If the depth variation increases beyond 12 centimeters, it is recognized as a push gesture. Conversely, if the depth variation decreases beyond 8 centimeters, it is recognized as a

doi:10.6342/NTU202302797

pull gesture.

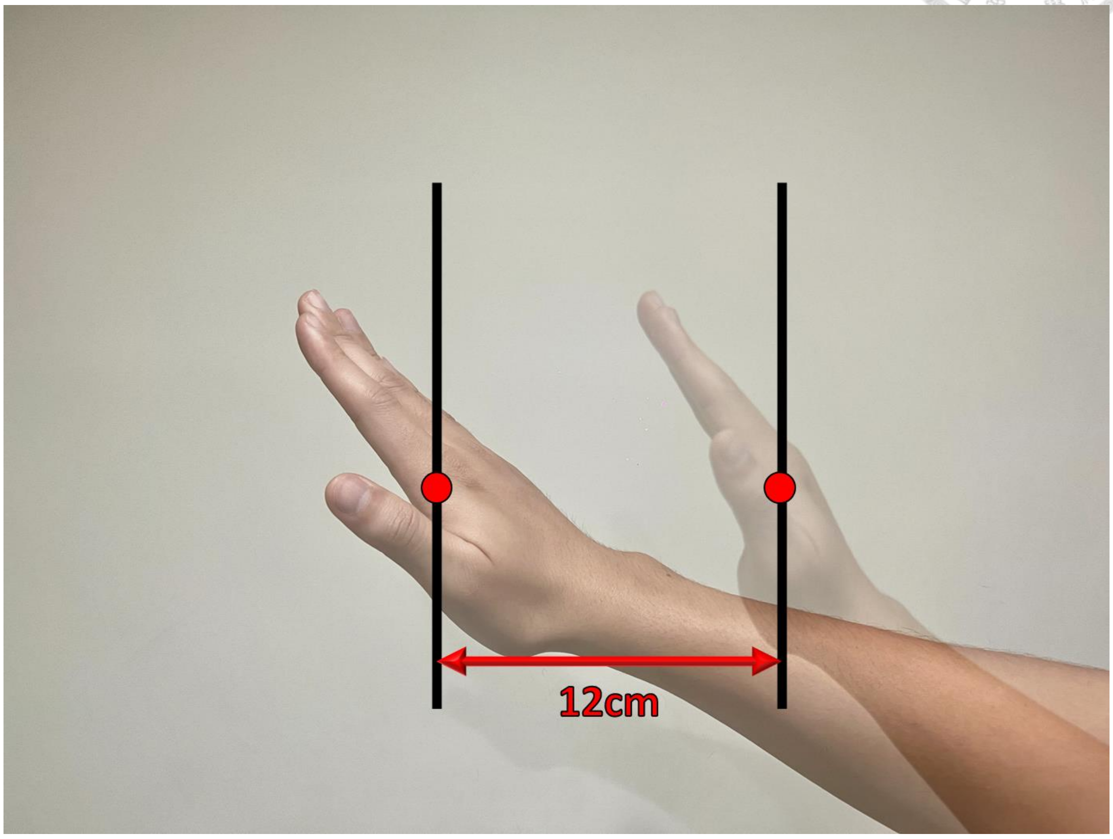


Figure 4-11: Illustration of a push gesture.

4.6 Our System

Our ChungGesture recognition system is primarily based on the hand palm centroid. Whether it is hand tracking, the four swipe directions, or push/pull gestures, required

doi:10.6342/NTU202302797

displacement and depth variation can be obtained through the methods proposed in this thesis. Therefore, we can integrate all gestures into a single system. However, during the same motion, it is possible to trigger multiple gestures simultaneously. For example, while performing a left swipe, the hand may also move closer to the body, triggering both a swipe left and a pull gesture. To address this, we have set a cooldown parameter (0.5 seconds) to prevent continuous triggering of multiple gestures. Furthermore, we assign priorities to these gestures based on their triggering difficulty and the frequency of gesture usage. Push/pull gestures are given higher priority, followed by the four swipe directions.

doi:10.6342/NTU202302797

Chapter 5 Experimental Results

5.1 Experimental Setting

In this thesis, our experimental setup consists of Jorjin's J-Reality J7EF Plus smart glasses and Sony's Xperia 1 II smartphone. Data acquisition is performed through the SDK provided by Jorjin, which imposes certain environmental limitations. Therefore, all algorithms in this experiment are implemented within Unity 2020.3.32f1 version. The Minimum API Level and Target API Level are set to Android 8.1 'Oreo' (API level 27). The entire project is built into an APK file and installed on the Xperia 1 II running the Android 12 operating system. Once the smart glasses are connected to the smartphone, data transmission begins, and the smartphone screen is displayed on the smart glasses synchronously. All our participants are students from the College of Electrical Engineering and Computer Science, National Taiwan University.

5.2 Precision Result

In our experiment, we used the color weakness game to test precision. The color

doi:10.6342/NTU202302797

weakness game involved dividing the screen into equally sized blocks. A red dot would move according to the hand palm centroid, and the objective was to keep the red dot within the only block on the screen with a different color for at least 0.5 seconds to obtaining score. As the score increased, the size of the blocks decreased, and the number of blocks displayed on the screen increased. Ultimately, we measured the precision based on the maximum achievable resolution.

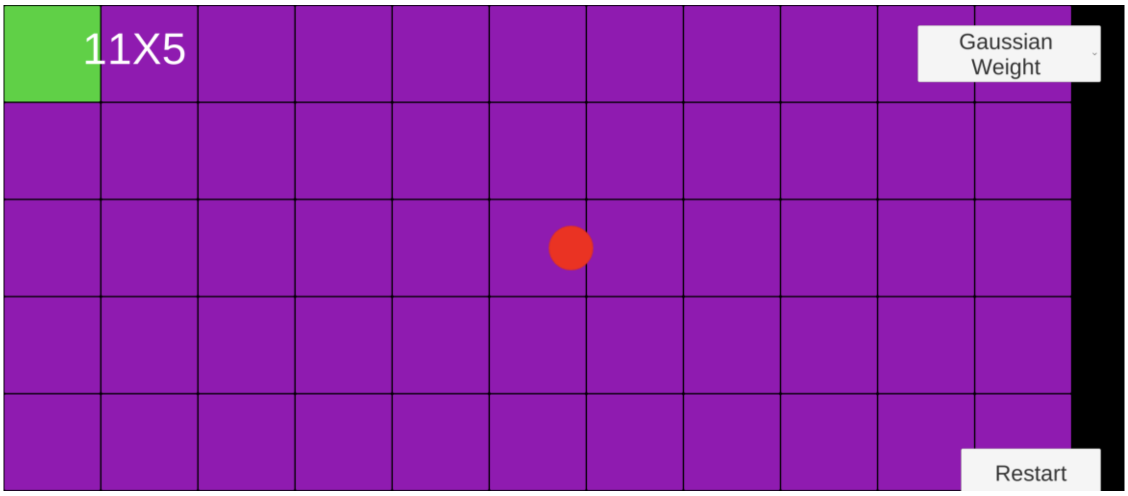


Figure 5-1: Screenshot from the Color Weakness game used in our experiments.

The test results, in Table 5-1, demonstrate that all of our methods outperform LuGesture. This is attributed to LuGesture's excessive inclusion of non-palm regions, which prevents it from reaching the upper-left corner, causing it to freeze at a resolution of 5x3. In contrast, our methods rank in the following order: Method 2, Method 3, Method 4, and Method 1.

doi:10.6342/NTU202302797

Table 5-1. Comparison results of precision.

	Resolution ↑ (pixels)	**Rank ↓ of Precision**
Method 1	12x7	4
Method 2	**17x10**	**1**
Method 3	16x9	2
Method 4	14x8	3
LuGesture	5x3	5

5.3 Stability Result

To test the stability, we employed a method where the hand was kept fixed in a consistent position. Using the initial hand palm centroid as the reference point, we measured the maximum and average displacement of the hand palm centroid over the following 2 seconds. This stability test was conducted simultaneously with the four methods mentioned in Section 4.4.3, along with LuGesture, to ensure that the input data remain consistent across all tests.

doi:10.6342/NTU202302797

We conduct tests at hand-to-glasses distances of 25 cm, 35 cm, and 45 cm, and the results are summarized in Table 5-2. Method 2 performed better only at closer distances, with a decreasing performance as the distance increased. This can be attributed to the centroid being scaled proportionally. With fewer pixels being available at farther distances, the calculated centroid is more susceptible to the influence of twinkle pixels, which amplifies the impact. On the other hand, Method 1 consistently performed well across all distances. Method 3 exhibited a deteriorating performance as the distance increased, highlighting the significance of adjusting the weight distribution between edge and center pixels as the pixel count decreases. Method 4 and LuGesture maintained a moderate performance throughout the distances. Based on these tests, we concluded that Method 1 is the most stable approach.

doi:10.6342/NTU202302797

Table 5-2. Comparison results of stability.

Distance	**Method**	**Max Displacement (↓)**	**Average Displacement (↓)**	**Rank ↓ (Max)**	**Rank ↓ (Avg.)**
25cm	Method 1	**0.30797**	**0.07797**	**1**	**1**
	Method 2	0.49090	0.13533	2	2
	Method 3	0.59835	0.22994	4	3
	Method 4	0.61624	0.26741	5	5
	LuGesture	0.53071	0.23308	3	4
35cm	Method 1	0.433898	**0.17689**	2	**1**
	Method 2	0.60592	0.22490	5	5
	Method 3	**0.37658**	0.18161	**1**	2
	Method 4	0.46323	0.20023	3	3
	LuGesture	0.48287	0.20820	4	4
45cm	Method 1	**0.37906**	**0.18205**	**1**	**1**
	Method 2	0.58799	0.25649	5	5
	Method 3	0.46471	0.21001	4	4
	Method 4	0.40675	0.18507	2	2
	LuGesture	0.41865	0.19921	3	3

doi:10.6342/NTU202302797

5.4 Gesture Recognition Accuracy Result

In this experiment, we employ Method 1 to conduct 50 tests with ten participants respectively using their right hand for swipe gestures in four directions (up, down, left, and right), as well as push/pull gestures in Table 5-3. It can be observed that ChungGesture outperformed LuGesture in all gesture directions, particularly in Swipe up and Swipe right. Participants often swing their hands diagonally from the lower-left corner to the upper-right corner of the sensor's sensing range while performing these two gestures, resulting in mutual misinterpretations. To address this issue, we pre-record vectors to achieve better performance.

doi:10.6342/NTU202302797

Table 5-3. Comparison results of gesture recognition.

Accuracy ↑	ChungGesture	LuGesture
Swipe Left	**95%**	92%
Swipe Right	**93%**	89%
Swipe Up	**91%**	87%
Swipe Down	**91%**	88%
Push	**96%**	**96%**
Pull	**93%**	**93%**

doi:10.6342/NTU202302797

Chapter 6 Conclusion and Future Works

In this thesis, we propose ChungGesture, which consists of six precise, fast, and comfortable gestures aimed at enhancing the user experience of Jorjin J-Reality J7EF Plus users. Despite the low resolution of low-cost and power-efficient ToF sensor, we encounter challenges such as excessive non-palm areas and insufficient stability based on LuGesture. However, through our efforts and the methods proposed in Chapter 4, we have been able to overcome hardware limitations and improve the mentioned issues.

In Chapter 5, we identify instances of gesture misjudgment, mainly due to variations in individuals' swinging habits. Designing a gesture recognition system that is applicable to every individual remains a goal that requires further research and effort.

Currently, we have successfully improved the gesture recognition accuracy from a minimum of 87% to 91% and integrated different functionalities into the ChungGesture system. In the future, additional gesture testing in different environments, such as multi-user settings and handling varying lighting conditions, will be necessary to achieve higher precision and stability. Once everything is refined, we hope to further develop ChungGesture towards a 3D mouse and explore more augmented reality applications through collaboration with Jorjin.

doi:10.6342/NTU202302797

References

[1] P. Milgram and F. Kishino, “A Taxonomy of Mixed Reality Visual Displays,” *IEICE Transactions on Information and Systems*, Vol. 77, No. 12, pp. 1321-1329, 1994.

[2] L. H. Lee and P. Hui, “Interaction Methods for Smart Glasses: A Survey,” *IEEE Access*, Vol. 6, pp. 28712-28732, 2018.

[3] R. Aigner, D. Wigdor, H. Benko, M. Haller, D. Lindbauer, A. Ion, S. D. Zhao, and J. T. K. V. Koh, “Understanding Mid-Air Hand Gestures: A Study of Human Preferences in Usage of Gesture Types for HCI,” *Microsoft Research TechReport* MSR-TR-2012-111, 2, 30, 2012.

[4] F. Remondino and D. Stoppa, Eds., *TOF Range-Imaging Cameras*, Vol. 68121, Springer, Heidelberg, Germany, 2013.

[5] P. Padmanabhan, C. Zhang, and E. Charbon, “Modeling and Analysis of a Direct Time-of-Flight Sensor Architecture for LiDAR Applications,” *Sensors*, vol. 19, No. 5464, pp. 1-27, 2019.

[6] M. S. Keel et al., "A VGA Indirect Time-of-Flight CMOS Image Sensor with 4-Tap **7-** μ m Global-Shutter Pixel and Fixed-Pattern Phase Noise Self-Compensation," *IEEE Journal of Solid-State Circuits*, vol. 55, pp. 889-897, 2020.

[7] P. A. Rauschnabel, A. Brem, and Y. Ro, “Augmented Reality Smart Glasses:

doi:10.6342/NTU202302797

Definition, Conceptual Insights, and Managerial Importance," Unpublished Working Paper, The University of Michigan-Dearborn, College of Business, https://www.researchgate.net/publication/279942768_Augmented_Reality_Smart_Glasses_Definition_Conceptual_Insights_and_Managerial_Importance, 2015.

[8] I. E. Sutherland, "Head-Mounted Three Dimensional Display," *Proceedings of Fall Joint Comput. Conf. AFIPS*, San Francisco, CA, 33, pp. 757–764, 1968.

[9] Google, "Glass Enterprise Edition 2," https://support.google.com/glass-enterprise/customer/answer/9220200?hl=en&ref_topic=9235678, 2023.

[10] D. Kim and Y. Choi, "Applications of Smart Glasses in Applied Sciences: A Systematic Review," *Appl. Sci.*, Vol. 11, 4956, pp. 1-21, 2021.

[11] Microsoft, "Microsoft HoloLens 2," https://www.microsoft.com/zh-tw/hololens/hardware, 2023.

[12] Epson, "Epson Moverio BT-350," https://www.epson.com.tw/c/Moverio-BT-350/p/V11H837054, 2023.

[13] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-Based Hand-Gesture Applications," *Communications of the ACM*, vol. 54, pp. 60-71, 2011.

[14] J. Suarez and R. R. Murphy, "Hand Gesture Recognition with Depth Images: A Review," *2012 IEEE RO-MAN: International Symposium on Robot and Human*

doi:10.6342/NTU202302797

Interactive Communication, Paris, France, pp. 411-417, doi: 10.1109/ROMAN.2012.6343787, 2012.

[15] M. Oudah, A. Al-Naji, and J. Chahl, "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques," *Journal of Imaging*, Vol. 6.8: 73, pp. 1-29, 2020.

[16] W. Daolei and K.B. Lim, "Obtaining Depth Map from Segment-Based Stereo Matching Using Graph Cuts," *Journal of Visual Communication and Image Representation*, Vol. 22, Issue 4, pp. 325-331, 2011.

[17] Y. H. Lu, "LuGesture: Hand Gesture Recognition with Time-of-Flight Camera for Smart Glasses," Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, 2022.

[18] Jorjin, "Jorjin J-Reality J7EF Plus," https://www.jorjin.com/products/ar-smart-glasses/j-reality/j7ef-plus/, 2023.

[19] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, London, 2021.

[20] C. Walker, "Stereo Vision Basics," http://chriswalkertechblog.blogspot.com/2014/03/stereo-vision-basics.html, 2023.

[21] L. Li, "Time-of-Flight Camera – An Introduction," Texas Instrument Technical White Paper, 2014.

[22] J. Geng, "Structured-Light 3D Surface Imaging: A Tutorial," *Advances in Optics and*

Photonics, Vol. 3, No. 2, pp. 128-160, 2011.

[23] P. Zanuttigh, G. Marin, C. D. Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, *Time-of-Flight and Structured Light Depth Cameras*: *Technology and Applications*, Springer, Berlin, 978-3, 2016.

[24] Sony, "Sony Xperia 1 II," https://www.sony.com.tw/zh/electronics/smartphones/xperia-1m2, 2023.

doi:10.6342/NTU202302797