Auckland University of Technology

Augmented Reality on Mobile Devices using Both Front and Rear Cameras

by

Tongtong Liu

supervised by

Dr. Minh Nguyen and A. Prof. Wei Qi Yan

A thesis submitted in partial fulfillment of the requirements
for the Master Degree in Computer and Information Science

School of Engineering, Computer, and Mathematical Science
Department of Computer Science

July 2020

Abstract

When people buy sunglasses in the store, it is hard for them to see themselves wearing sunglasses in the mirror. Besides, if people want to buy glasses online, they can't try on the glasses and they don't know if the glasses fit them well or not. Thus, this thesis proposes a glasses try-on system. Augmented reality has received extensive attention from researchers in recent years. This paper will discuss the development, application and challenges of augmented reality. Besides, an augmented reality application is proposed in this study, and this application is based on mobile devices. In order to improve the interaction, both front and rear cameras are required in this application. Two glasses try-on systems are built in this thesis, which are 2D and 3D glasses try-on systems. The first system is to use the glasses image to implement the try-on system. Users need to scan a glasses picture and use the system to try on the glasses. The second system is to use 3D glasses models. The user first scans the QR code, and then the corresponding 3D glasses model will be presented for users to try.

Acknowledgements

In the beginning, I really want to say that Dr. Minh Nguyen, who is my primary supervisor is a very good man. During the project, we did a lot of research together. He is a very patient and attentive person, under his very patient guidance, I did learn a lot about the Augmented Reality technology. Absolutely, I cannot finish my job without his guidance. I really want to thank Dr. Minh Nguyen. And I would like to say the same word to my secondary supervisor, Professor A. Yan WeIQi. Thanks for his help.

I declare that this article is written by me. This report is my own work. It does not contain the work of anyone else or materials of other academic articles. No unfair means are used to finish this thesis.

Signed:

Date: 02/20/2020

Contents

List of Figures

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AR	Augmented Reality
HMD	Head-mounted Display

Dedicated to myself

1

Introduction

Augmented reality (AR) is a technology which gives us a new way to see, hear, and feel our environment. As early as the 1960s, AR technology was initially realized. Nowadays, AR has been truly practical. With the development of the Internet and smartphones, augmented reality has gained a lot of attention and a huge number of augmented reality applications are being created recently.

According to [20], AR is a technology that connects the real word with objects that are created by computers, such as graphics, sounds, videos or other digital information. Besides, an AR system has three features [1]:

- Combines real objects with virtual objects in a real environment;
- Real and virtual objects are aligned with each other;
- Interactive;
- Three-dimensional.

AR system is not limited by the display devices. There are mainly three types of devices to show results of AR: Head-Mounted Displays (HMDs), mobile devices and computers. The first HMD is developed in 1968 (Figure 1.1). It is used to show three-dimensional images.

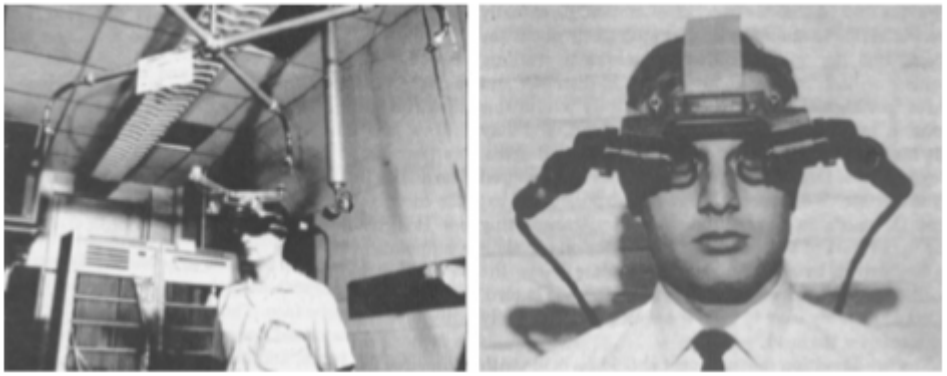


FIGURE 1.1: The first HMD [1]

The term "augmented reality" was firstly presented by Caudell and Mizell in 1992 [21]. In 1993 a GPS-based outdoor system was developed which used spatial audio to provide navigation for the visually impaired. The first mobile AR system was developed in 1997, and Feineretal created a prototype mobile system (MARS) to record three-dimensional information of a building [1]. Today, most modern mobile platforms support AR, such as tablet computers and mobile phones.

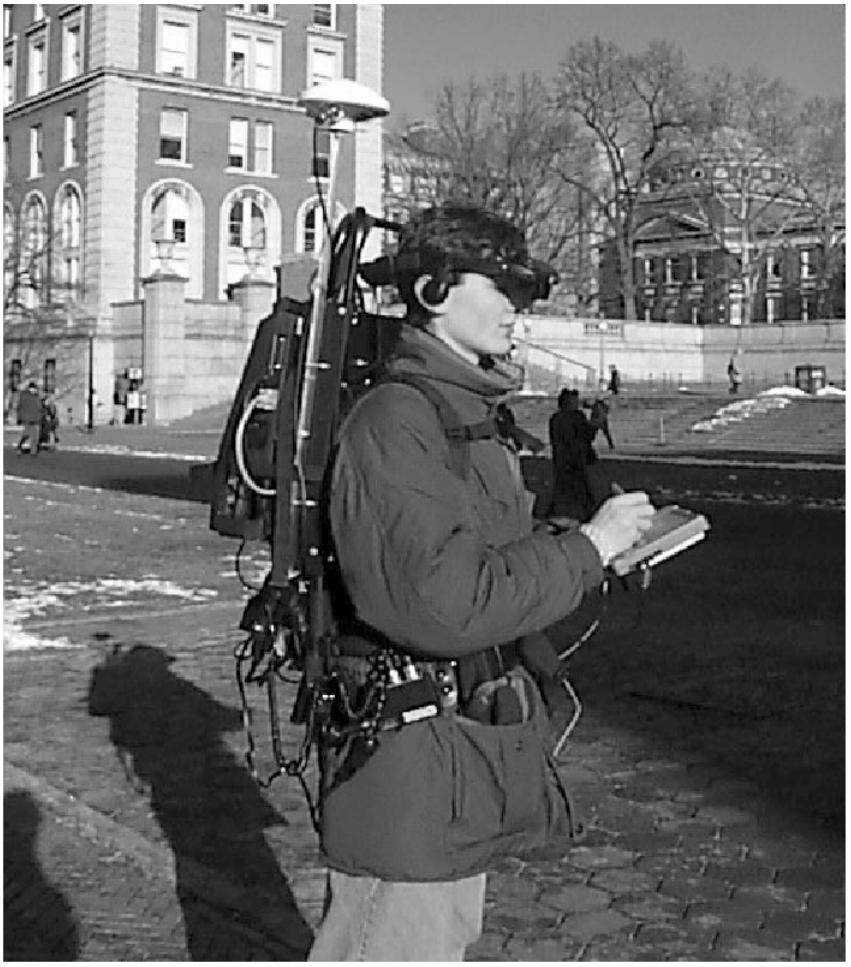


FIGURE 1.2: The first mobile AR system [2]

1.1 Application areas of Augmented Reality

Augmented reality technologies have been used in various areas, such as entertainment, education, medicine and automotive industry.

1.1.1 Applications for entertainment

In recent years, AR games have caused a lot of attention. Pokémon GO is a very popular outdoor game using AR technology and it is a location-based AR game (Figure 3). Players can go different places and catch different creatures which is one of the most attractive features of this game. Furthermore, this game encourages people to play outdoors and exercise. The paper [21] took Pokémon GO as an example and presented that the development of augmented reality (AR) is becoming more commonplace. Giving users a better experience and interaction with reality is one of the major advantages of AR. AR is popular in many industries such as e-commerce, promotion and travel and tourism.

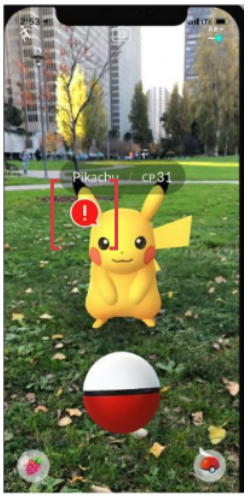


FIGURE 1.3: Pokémon GO

There are also a lot of indoor AR games. [22] built a mobile AR game based on SIFT image recognition. This research improved the SIFT algorithm and applied it to the AR game, which improved the accuracy of the AR system. The paper of [23] designed an AR robotic shooting game. The players can use body gestures to control the robot and virtual objects. [3] developed a chess game using AR technique. It uses fingers to interact with the virtual chess by an RGB-D camera sensor (Figure 1.4).

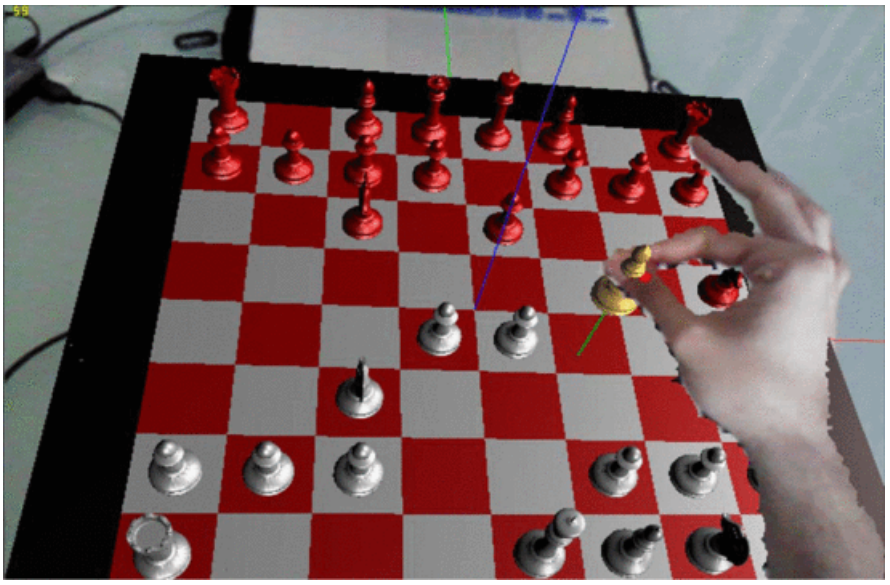


FIGURE 1.4: Chess game [3]

In addition to the AR games, some papers have built applications for AR dog. More and more people in modern society like pets, especially dogs. Some people even treat dogs as part of their own family. Dogs are the most faithful companions of human beings, and their company can add a lot of fun to people's lives. Not only can dogs help reduce the stress of daily life, but also soothe people when they feel lost. Besides, dog walking can help people increase their physical activity. The system built in the paper [24] studied the impact of human-dog interaction on humans. This system uses a head-mounted device to display the AR dog. Users can play with the AR dog, feed the AR dog and walk the AR dog. The results of this research prove that the AR dog has some positive effects on human emotions and behaviors.

The research of [25] also pointed out that the interaction between human and animals is beneficial to people's physical and mental health. Despite the benefits of AR animals, they still have some limitations. One of the biggest limitations is that human can't touch the virtual animals. Thus, further studies are needed in systems for AR animals.

1.1.2 Application for education

The study of [26] has shown that AR has the potential to stimulate students' interest in learning. As a new technology, AR has many advantages in the educational environment. Teachers can use AR to improve interaction with students in teaching, thereby improving students' learning efficiency. What is more, AR also enables users to see elements that are not easy to see in the real world, which can help students understand new knowledge [26]. Especially for preschool education, abstraction and complex concepts

make understanding difficult, which may make young children lose interest in learning. Therefore the traditional teaching methods should be changed, and AR applications are beneficial for young children to learn new knowledge.

There are already many AR applications for education. The research [27] claimed that AR has a great impact on education. The AR application created in this paper is used to learn the structure of a desktop PC. There are three learning modes in this application. First, students can learn and observe various parts of the computer. Then the students can assemble a computer according to the tutorial. If they put a component in the wrong place, the application will point out their mistakes. Finally, students can take tests to find out how well they have mastered computer assembly.

The paper [20] developed a foreign language learning application using AR. This application is able to recognize a foreign word and show its virtual object on the screen. It provides a good way for people to learn foreign languages.

The research [28] designed an AR-based gear game. This game used a motion sensing camera to capture human bone joint points and develop a gear interactive game. It is a shooting game. Users use their hands to control the gears in the game and use the gears to control the shooting direction. This game helps increase students' interest in learning. Furthermore, It makes fun and helps students learn the scientific concepts of gears during the interactive process.

The paper [4] created a digital game using AR. This digital game helps users to learn mathematics.

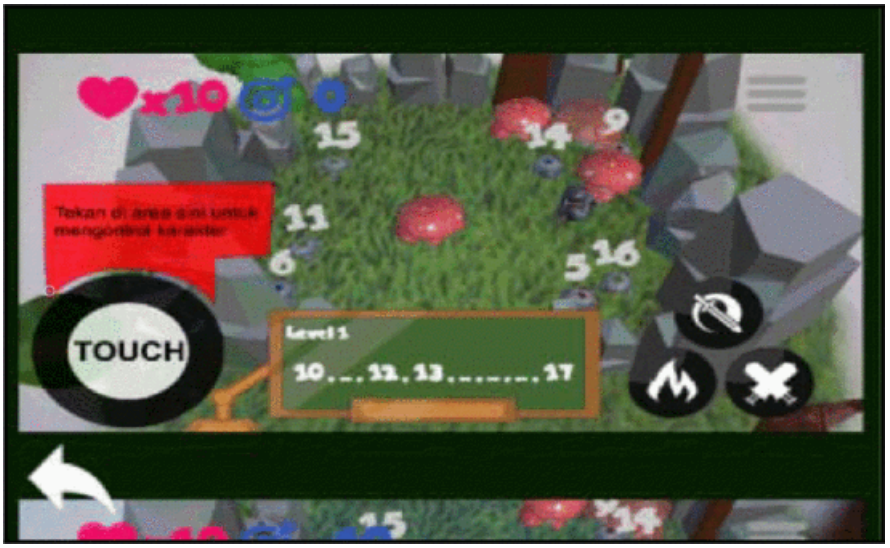


FIGURE 1.5: A digital game to learn mathematics [4]

Many students found that it is hard to learn solid geometry. We usually draw 3D shapes on a 2D plane to understand solid geometry, but this expression is very abstract and difficult to understand. Therefore, the research [29] has created a system which can show 3D objects in the real 3D world. In addition, students can use their hands to control 3D objects. This system helps students learn and understand solid geometry better.

The study [30] uses an AR application to develop students' computational thinking. The logic of computational thinking refers to the logic of computer program operation. Nowadays our lives can not be separated from computers. Computers can perform tasks efficiently through simple repetitive tasks, so it is necessary for us to learn computational thinking. The application designed in this paper is similar to programming toys. Children use physical cards to form a 3D virtual map. After the map is completed, children can use a character to test the map. Different maps represent different stories. Children have to use computational thinking to form their own stories.

1.1.3 Medical applications

Medical imaging has been an important technique in our life. Doctors have the longest growth cycle in almost all industries. Medical students usually need to learn a lot of knowledge from books or videos, but they have fewer practical opportunities. Medicine is a highly practical subject. Only by participating in a large number of clinical practices can medical students improve their operating skills.

According to the research of [5], medical students are required to take a lot of medical training, and their training needs to be supported by a lot of resources such as high definition pictures, care experience for real patients and anatomical manikins. But these resources and the interaction with students are limited. Thus, AR based medical training systems are very useful and helpful to improve medical education.

With the development of AR technologies, many research and technology companies are applying AR technology to surgery. Through AR technology, the 2D medical image information can be three-dimensionalized, making doctors' surgical treatment easier and more accurate.

The paper of [5] described a medical training system called Augmented Reality Environment for Life Support Training (ARLIST) system (Figure 1.6). This system uses regular manikins. There are some sensors that mounted inside the manikins' body, allowing students to interact with it. Another advantage of this system is that it can save the resources of corpses and experimental animals. [31] created a medical training system using AR. This system can detect hands and fingers motions, and gesture input

is the main interaction method. The system can generate three-dimensional objects in the real world. The AR system created by the paper [32] built a 3D model of the cardiovascular system. The system enables users to view the location of human organs and detailed information about the cardiovascular system.

FIGURE 1.6: ARLIST system [5]

The company EchoPixel founded in 2012 developed an interactive 3D platform that converts 2D images into 3D images. This platform helps doctors diagnose the internal organs of patients from various angles. The 3D images can be customized according to the requirements of the program. Doctors can scale the images. They can also extract the questionable parts of the images separately or print them out for further research. Thus, doctors can find and check out the lesions in the patient's internal organs more easily.

Augmedics is an Israeli company founded in 2014. This company developed an AR head-mounted display used for spinal surgery. This device provides surgeons with X-ray vision during surgery. It allows surgeons to see the anatomy of the patient's body through the skin and tissue. This device makes easier, faster and safer surgery.

Besides, there are also many AR applications designed for patients. An Alzheimer's charity in Australia has launched an app to help patients with dementia, called Dementia-Friendly Home. This application can help Alzheimer's patients make their rooms more livable. Staff at the charity said environmental challenges, visual challenges and memory challenges are the problems faced by patients with dementia. They developed this app to allow people with Alzheimer's to live independently, and help them build self-confidence.

The study of [6] designed an AR game. This game uses hand interaction to assess the patient's upper limb motor function.

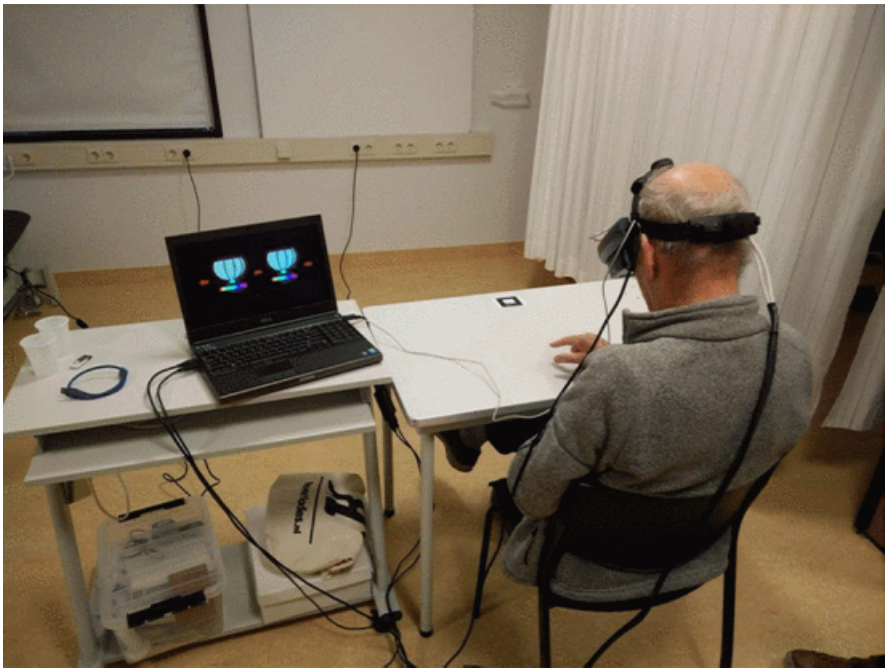


FIGURE 1.7: Application to assess patients' ability [6]

1.1.4 Applications for E-commerce

Augmented reality has been an effective tool for E-commerce in modern life. Many companies have used AR technology as a tool to present their products, such as Snap Nike and Adidas. Some cosmetic companies develop magic mirrors using AR. Users can have virtual facial makeup using AR mirrors, such as different lipsticks and eye shadows [33].

The study of [34] showed that online shopping has a significant impact on AR technology. Customers could not try on the products physically during online shopping, but they can see themselves wearing virtual products by using AR. In this way, consumers can experience the products they want to buy more realistically. Thus AR is experiencing huge popularity nowadays.

[34] introduced an AR application called Lify, which enables users to try different clothes using AR. The use of this application is beneficial to reduce the number of product returns.

[7] designed a footwear try-on system which allows users to see themselves wearing different shoes virtually. In their system, an RGB-D camera was used to take depth

images of users. It is helpful to get 3D information of the user's foot. This system can identify and track user's shoes, and display shoe models on the user's foot.

FIGURE 1.8: A try-on system [7]

AR brings many conveniences to online shopping and many brands have noticed the advantages of AR. Zara created an AR application in 2018. When users scan an AR marker, they will see a model showing them Zara's costumes. The model will pose, walk and talk, which will help users better understand Zara's products. In addition to enabling AR shopping, this app also has social media sharing capabilities, where users can take and push holographic images or videos.

Many times when we visit IKEA, we may consider whether the furniture look good in our home. IKEA developed an AR application in 2014 that can solve this problem. In the AR application, users can use their mobile phones to scan the surrounding environment and place 3D furniture models in the surrounding environment. The biggest highlight of the app is that it can automatically resize according to the size of the surrounding furniture. For example, if a chair is placed next to a table, the virtual chair will automatically adjust to the right size to help you judge.

Nike added AR technology to the release and sale of shoes. They hid the limited edition shoes on the street and used a hunting game like Pokémon GO to make customers find and buy their favorite limited edition shoes.

L'Oreal, Sephora, Estee Lauder, Shiseido and other beauty brands launched AR makeup. Besides, an E-commerce enterprise from China called JD.com has added AR makeup

to their application. When users buy lipsticks on JD.com, they can try the lipsticks virtually by using AR technology.

1.2 AR challenges

Although a lot of AR systems have been developed, there are still some limitations and challenges. Interaction and tracking between generated objects and real objects are the major challenges in augmented reality, a lot of work has been done to improve the system's performance [35].

The virtual objects in AR must be placed in the same coordinate system as the real objects [36]. When the camera moves, the 2D projection of the virtual objects on the screen must move accordingly. According to the research of [37], there are two kinds of markerless camera tracking methods: image-based and model-based methods. Image-based tracking method is faster and more stable. [38] used an image-based tracking method to build an outdoor AR tracking system by using homography. This system could also be used to measure head rotation. Recently, the tracking methods have been improved clearly by using markerless model-based tracking techniques.

Hand tracking is one of the major interaction methods. [39] developed an AR system based on hand gesture recognition. [8] proposed an interactive system based on AR which has the function of object detection and hand tracking. If the camera detects an image target, a 3D model will be shown. This system also allows users to interact with virtual objects by hand tracking.

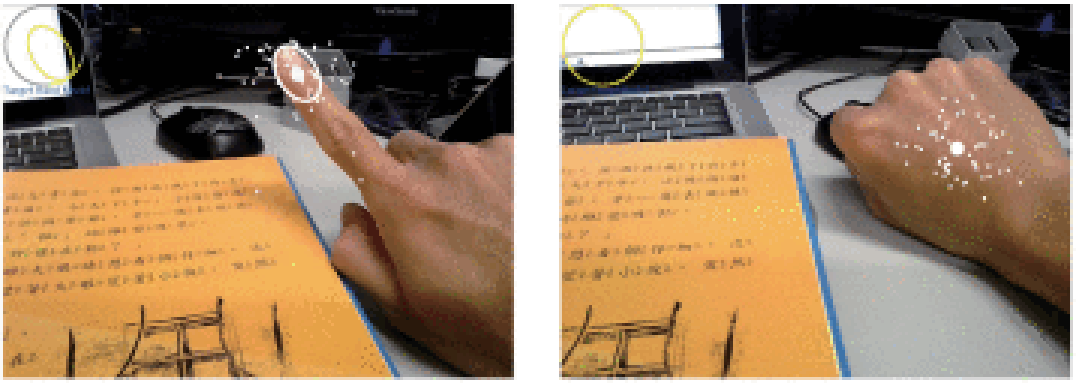


FIGURE 1.9: Finger tracking and hand tracking [8]

Besides, many sensors have been considered in order to show a better tracking performance, such as mechanical devices, inertial devices, ultrasonic devices, magnetic sensors, optical sensors, inertial devices, GPS and compass. [9] used wearable sensor networks

in their AR system (Figure 1.9). They developed an intelligent physical rehabilitation system which combined augmented reality serious games with wearable sensor networks. This system is used to help patients improve their engagement during physical rehabilitation. Moreover, only one camera is used in the AR system of the above literature.

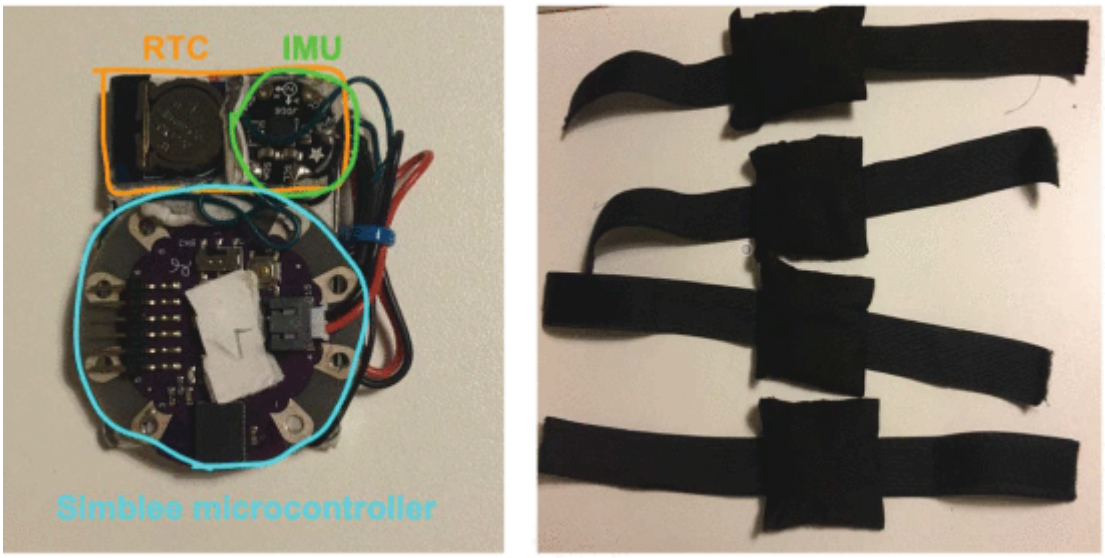


FIGURE 1.10: Smart and wearable sensors [9]

There are also some limitations of AR. In smartphones, AR can only use limited storage space, small processing power and a small amount of memory. Another one of the biggest concern about augmented reality is privacy. People may get stranger information from AR-enabled image recognition software, and see more private information from their Facebook, Instagram, Twitter or other online profiles.

1.3 Research Goals

In this research, we will build a mobile application using AR. This system will show a 3D virtual object if the rear camera detects an image target. Therefore, this application requires image recognition technology. What is more, the front camera could detect the user's face and the rotation and location of the user's head. The virtual object rotates with the rotation of the human head. Two kinds of glasses try-on systems will be built in this research. This system can run on mobile devices, and use both front and rear cameras to improve users' AR experience. Furthermore, we will compare these two systems and give improvement.

1.4 Thesis structure

There are five sections in this thesis:

- Chapter 2 introduces the description and methods of AR. The background and some main technologies of the try-on system are also introduced.
- Chapter 3 introduces the two glasses try-on system and the methods to build these two systems.
- Chapter 4 shows the try-on results of the two glasses try-on systems. The discussion analyses the results and the pros and cons of the two systems. The improvement of these two systems is proposed.
- Chapter 6 summarizes this thesis and talk about the future development of the AR try-on system.

2

Background

2.1 Augmented Reality

Augmented reality (AR) is a new technology that combines computer-generated information with the real world. It is believed that AR comes from the development of virtual reality (VR), but there are obvious differences between them. VR gives users a completely immersive effect in the virtual world, which is to create another world; while AR technology brings virtual information into the user's real world. Humans can perceive these virtual information through the sensory organs, such as eyes, nose and ear. There are some new technologies and methods included in AR, such as multimedia, 3D modelling, real-time tracking and registration. Nowadays, with the declining prices of the hardware devices, the improvement of image processing technology, and the development of the software, the number of AR applications is increasing rapidly. This section mainly focuses on the technologies and applications to build AR.

2.1.1 AR Display technology and devices

There are two main techniques to visualise AR information which are optical combination and video mixing [40]. Optical combination refers to generating a virtual object directly in the user's surroundings through optical technology. Users can see the 3D AR objects in the real world. The AR glasses of Microsoft and Google are two examples of optical combination. On the other hand, there are two steps to create a virtual object in video mixing:

1. Captures the real environment by video.

2. Virtual objects are then added to the video and when the mixing process is finished, users will be able to see these virtual objects [41].

The HTC Vive VR headset is an example of video mixing. It uses an inbuilt camera to generate AR objects on the device.

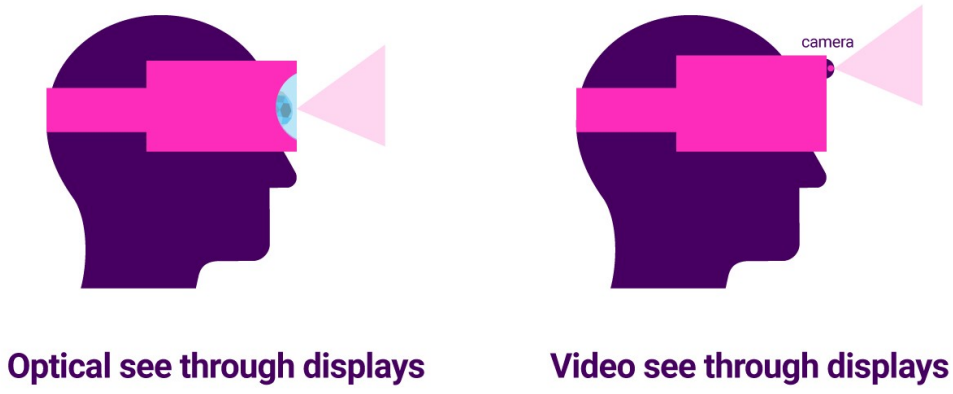


FIGURE 2.1: Optical combination and video mixing [10]

According to the way users use AR devices, AR display devices can be divided into three categories, namely hand-held displays, spatial displays, and head attached displays [40]. These AR display devices use various visualization techniques, and these display methods are suitable for different situations.

1. Hand-held display

   Hand-held displays have the features of low cost and easy access, such as mobile phones and computers. Therefore hand-held displays are more popular than the other display devices currently. These devices use a mobile camera to display AR objects. The camera records a video of the real world and then transmits them on the hand-held device [40]. After that, the video recorded by the camera is merged with the virtual objects. Finally, users can see the mixed videos in the device. These devices usually need depth cameras, motion sensors and AR technologies such as ARKit and ARCore, or WebAR for display.

   Although handheld AR is a perspective video, it deserves special mention. The rise of handheld AR is a turning point in the true popularity of the technology. ARKit, ARCore, MRKit, and other augmented reality libraries make complex

computer vision algorithms available to anyone. In hand-held AR, people only need a smartphone to access AR experiences.

FIGURE 2.2: Hand-held devices [11]

2. Head attached displays

   Head attached displays are devices that can be placed on the front of the user's head. This kind of device can bring users a more realistic experience. These devices are further divided into three classifications: retinal display, head-mounted display, and head projector display.

   Retina display can visualize a virtual object at a few centimetres of eyes. It gives the illusion that the virtual object is in front of the users. This device sends a beam through a small high-speed laser and simulates a continuous eye scan. Because the light beam is fast, the retina absorbs enough transmitted images so the brain can recognize it [40]. There are some AR glasses can achieve retina display, such as the Glyph headset.

(a) (b)

FIGURE 2.3: Glyph headset [12][13]

FIGURE 2.4: HMD [14]

Head-mounted displays (HMDs) appeared in the 1960s, and the word AR has not even been coined [42]. In these devices, two forms of visualization are available: optical see-through and see-through video. See-through video is video mixing. In optical see-through, the real world is first recorded by the camera, then the virtual image and the real image are integrated into a single composition.

3. Spatial displays

   Spatial display is different from hand-held and head attached displays. It is applied on non-mobile applications. These display devices use large spatially-aligned optical elements, such as mirror beam combiners and transparent screens [41].

2.1.2 Markers

In the AR devices, the process of the images are divided into two steps:

- Recognition phase;
- Tracking phase.

The aim of the recognition phase is to find out the location of the 3D objects. To achieve this goal, the AR devices have to recognise the objects in 2D coordinate, and then assign it a corresponding 3D object [40]. This recognition is based on artificial markers, marker-based, natural markers or markerless. At first, AR applications were created on marker-based method, and later on, the markerless approach replaced the

marker-based method. The marker-based method requires a pre-made marker. This marker could be a QR code or a template card with a certain shape. Then place the marker in a position and use a camera to detect and recognise the marker. The information in the marker will be obtained after recognition and the 3D object will be shown. Normally, marker-based methods are often easier to identify because they often contrast more sharply with reality. They are more widely used because of their fast recognition.

2.1.3 Image processing

When we use a marker-based method to implement an AR application, we need four steps to process the image [43]:

1. Pre-processing;
2. Marker detection;
3. Identification;
4. Computation of location and pose.

To detect a marker, we must identify the borders of the marker. There are two methods to do the identification. The first one is image delimitation, and the other one is to search the boundaries in a gray scale image [43]. In the method of image delimitation, there may be many other objects in the scenario which will affect the identification. So we need to discard these objects. To achieve this, we have to assign a marker to each object and then find out the effective marker. The second method has to find the contour of the objects, assign lines to these contours and extend the lines [40]. This method is too complicated, so normally we don't use it.

There are two kinds of markers, which are the template marker and the 2D barcode marker. The template marker is a symbol which has black and white patterns in the center.

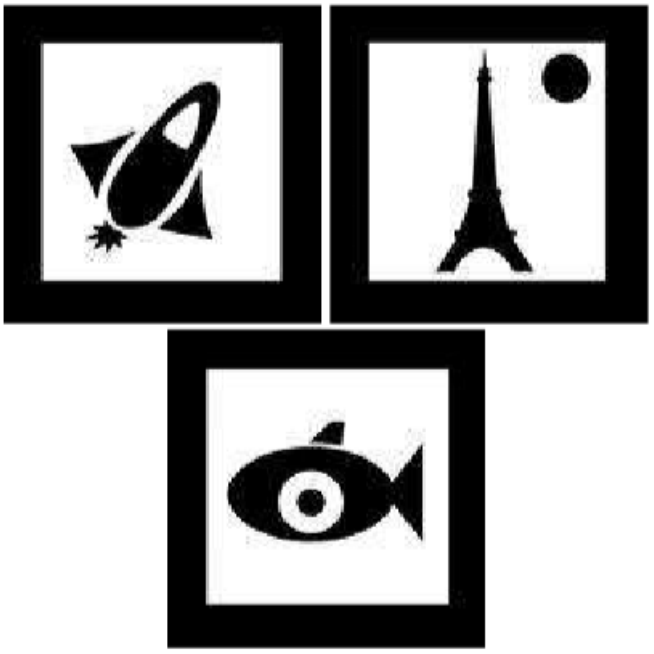


FIGURE 2.5: Template markers

The 2D barcode marker also has a black and white pattern, but it has small cells compared with the template marker. There are two types of 2D barcode marker: binary data markers and binary ID markers. Binary data markers can store more information, such as DataMatrix, QR Code and PDF417.

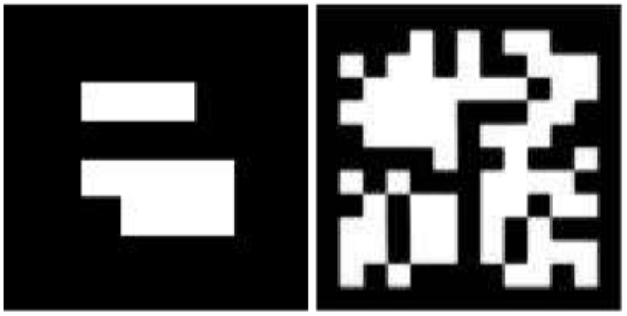


FIGURE 2.6: Barcode markers

2.1.4 Methods of AR implement

There are three methods to implement AR: marker-based, location-based and computer vision [44]. At first, some AR systems used two-dimensional markers (sometimes called trigger images) to show augmentations. In this method, a marker is defined in AR authoring environment, and the marker is unique to that augmentation.

FIGURE 2.7: Marker based augmentation

Lately, with the development of mobile AR, more AR applications focus on geographic and GPS access to display augmentation in the physical environment. The AR game Pokémon Go is an example of location-based AR.

FIGURE 2.8: Location based augmentation

Nowadays, the AR applications are more likely to identify some complex 3D objects, light and shadows.

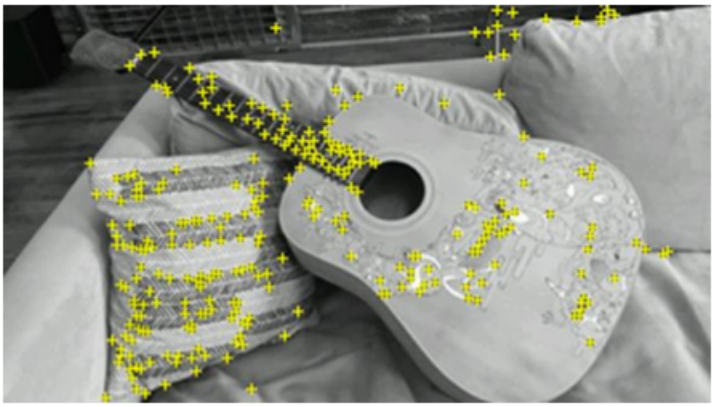


FIGURE 2.9: computer vision based augmentation

2.2 Software tools

Several companies offer tool kits to aid developers to create AR applications. This section will introduce some software tools to develop AR programs. The software tool kits consist of API, libraries of specialized functions, development systems and developer user-interfaces.

1. ARToolKit

   ARToolkit is an open source AR (augmented reality) SDK. It is a library written in the C / C ++ language, which makes it easy to write augmented reality applications. The most difficult part of augmented reality is overlaying the virtual image onto the user's viewport in real time and accurately aligning with objects in the real world. ARToolKit uses video tracking to calculate the camera position and orientation in real time. It provides fast and accurate marker tracking, allowing you to quickly develop many newer and more interesting AR programs.

2. OpenGL OpenGL is a graphical interface library. It is a set of specifications for calling GPU functions, and it mainly defines a series of functions for manipulating graphics and pictures. It can also draw from simple graphics to complex three-dimensional scenes.

3. OpenCV

   OpenCV is the Open Source Computer Vision Library. It provides a basic algorithm library for image processing and video processing, and also involves some machine learning algorithms. Such as video noise reduction, tracking of moving objects, and recognition of targets (such as faces recognition).

4. Relationship of these tool kits

   OpenCV focuses on obtaining information from the collected visual images and uses machines to understand the images; OpenGL uses machines to draw appropriate visual images for people to see. ARToolkit relies on OpenCV and OpenGL. Although large functions can also be achieved with OpenCV, but use ARToolkit is more convenient and efficient.

2.3 AR try-on systems

Virtual try-on systems has been a hot topic recently. There are already some try-on systems, especially in 3D garment try-on systems. Nowadays, online shopping is becoming more and more popular. People don't need to go out to buy clothes and food. They can stay at home and shop online. But there is one problem: the clothes they shop online may not fit them. Many people always find that the clothes don't fit them when they receive them. They have to return theses clothes to the seller, which is very annoying. According to Walker Sands Communications, thirty-five of customers prefer to shop online if they could virtually try on an item [45]. There is an urgent need for a virtual try-on system.

2.3.1 Virtual 3D garment try-on systems

A set of studies of clothes try-on system have been done. The paper [46] proposed an online 3D virtual garment try-on system. A specific 3D body model is created for each user in the system by using their body information, such as the length of head, neck, shin, hand and foot. The basic body model is downloaded from Poser [47] and then get the triangular body model by layering and triangulating the basic body. Then adjust the shape and size of the triangular model body to generate a specific body for users. The model of the garment users choose will be generated in different size and color. Finally, match the body model and the garment model to complete the virtual try-on system. This is a real-time try-on system and the motion and rotation of the model are obtained from the image or a video captured by a camera. This method is similar to the method used in online virtual fitting room [48].

There is a different virtual garment try-on system [49]. They use human body models and garment images to implement the virtual try-on system. The idea of this system is to extract garments in a picture using semantic segmentation techniques and match them to human body models. This system focuses on garment segmentation methods,

and it proposes a method of using recursive convolutional network (RNN) to solve the image segmentation problem.

The research [50] created a ties try-on system. This research doesn't build a 3D body model for the user and they use a simple image. They use the golden ratio to locate the neck after face detection. According to [51], the golden ratio means the height of the neck is 0.61 times the height of the face.

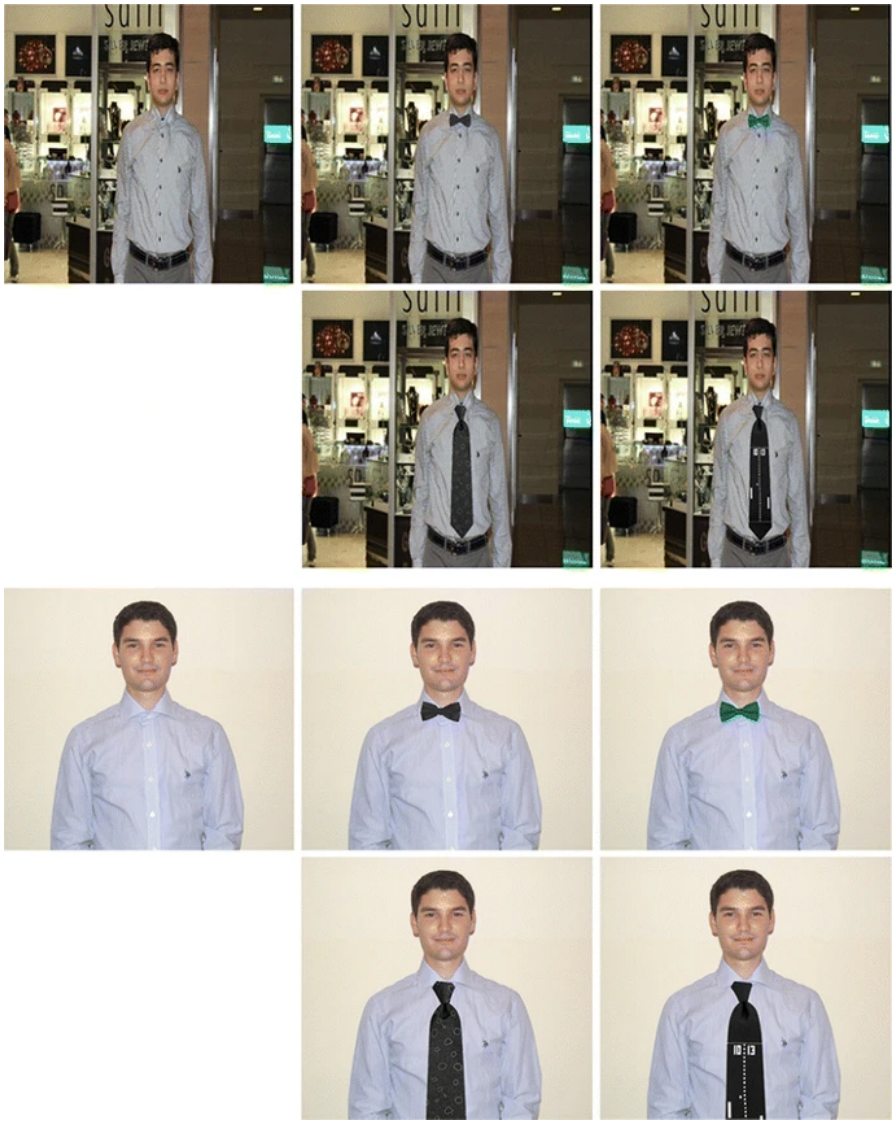


FIGURE 2.10: Ties try on picture

2.3.2 Glasses try on systems

Besides, there are glasses try-on systems also. For example, DITTO and Ray-Ban Virtual Mirror offers eyeglasses try-on service [52]. DITTO is actually an online glasses shop. Customers can try on all the glasses sold on the shop.

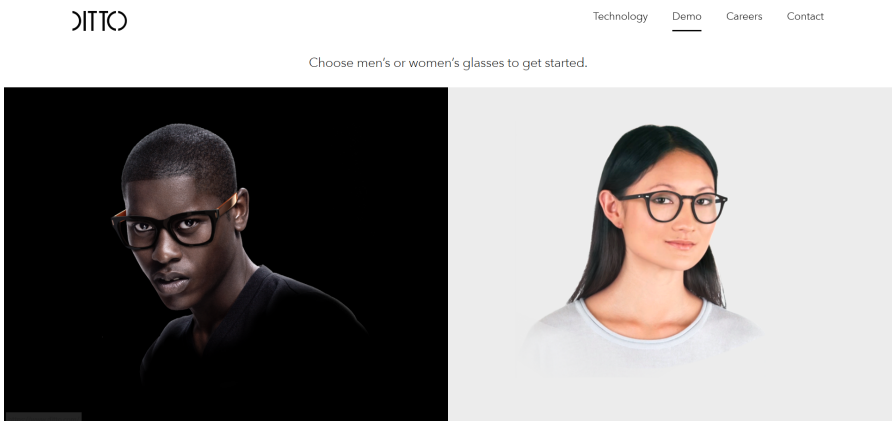


FIGURE 2.11: DITTO

In glasses try on systems, face detection is the first problem to be concerned. It aims to find the face region in an image. Many algorithms have been developed for face detection, such as LBP (local binary patterns), PCA (Principal Component Analysis) Haar, HOG (histogram of oriented gradient) and SVM (support vector machine)-based face detection. LBP is a local feature extraction method [53]. It is used to describe the local texture characteristics of an image. Moreover, it has significant advantages such as strong classification ability, high computing efficiency and gray invariance. HOG describes edge features [54]. Haar uses three kinds of features to describe a picture which are edge features, line features and center-surround features [15].

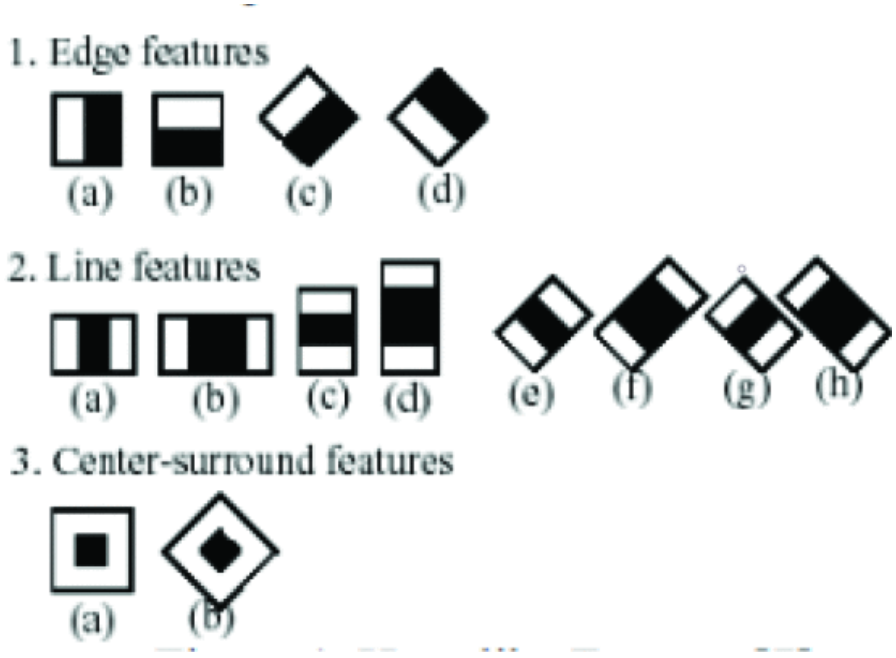


FIGURE 2.12: Haar features [15]

The Haar feature reflects the local changes in image gray to some extent. In face detection, certain features of a human face can be simply represented by rectangular features mentioned above. The picture below shows some features of the face using Haar feature extraction. In the picture, the eyes are darker than the surrounding area and the nose bridge is lighter than the sides. Similarly, other targets, such as eyes, can also be represented by some rectangular features.

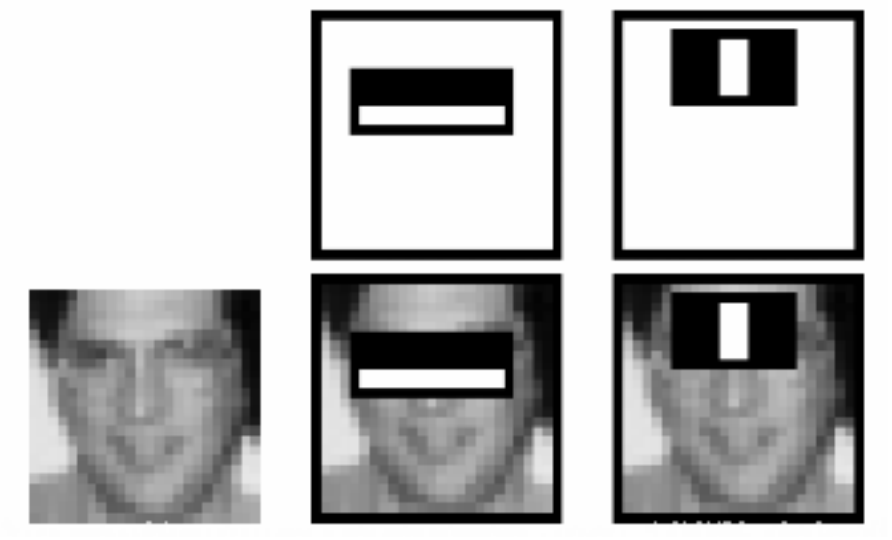


FIGURE 2.13: Face features using Haar [15]

SVM-based face detection is a method using machine learning. The picture below is the network structure of SVM [54]. SVM convert the input vector from low-dimensional input space to the high-dimensional feature space by nonlinear mapping firstly. Then SVM will find the hyperplane with the largest interval.

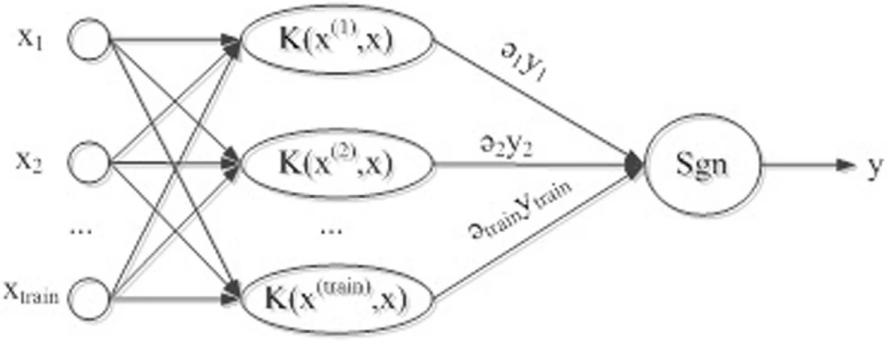


FIGURE 2.14: The network structure of SVM

Some research use 3D face reconstruction in their glasses try-on system. 3D Morphable Model (3DMM) is a popular solution of 3D face reconstruction. 3DMM was first created by Volker Blanz and Thomas Vetter in 1999 [55]. The three-dimensional deformation model is based on a three-dimensional face database and is constrained by face shape

and face texture statistics. At the same time, the influence of the pose and illumination factors of the face is considered, so the generated three-dimensional face model has high accuracy. The 3DMM model is a linear combination of 3D face data objects. Based on the 3D face representation, we build a 3D deformed face model which is composed of personal face models. Each face model contains two vectors S_i and T_i. In this way, the new 3D face model can be expressed as:

$$S_{model} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i \tag{2.1}$$

$$T_{model} = \overline{T} + \sum_{i=1}^{m-1} \beta_i t_i \tag{2.2}$$

Among them, $\overline{S}$ represents the average face shape model, s_ir epresents the PCA principal component of the shape, and α_i represents the corresponding coefficient. The texture model is the same.

Therefore, a new face model can be linearly combined from an existing face model. That is, by changing the coefficients, different human faces are generated based on the existing human faces.

Although the first version of 3DMM solved the expression of the face deformation model, it was obviously insufficient in the expression of facial expressions. In 2014, the Faceware-House paper proposed and disclosed a facial expression database, which made 3DMM more expressive [56]. Thus the linear representation of the face model can be represented as:

$$S_{model} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i + \sum_{i=1}^{m-1} \beta_i t_i \tag{2.3}$$

Based on the above principles, the face reconstruction problem is transformed into a problem of calculating α and β. Here is a face photo and we can get the 68 feature point coordinates $X_{2d}(x, y)$ of the face from there. There are corresponding 68 feature points $X_{3d}(x, y, z)$ in the BFM (Basel Face Model) model. Based on this information, you can find the coefficient α and β, and fits the face The model with the face in the photo.

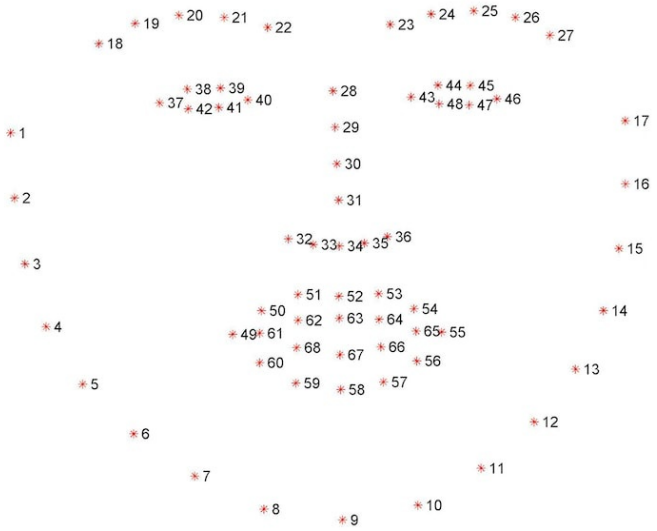


FIGURE 2.15: 68 face points

The formulas to solve this problem are as follows:

$$X_{2d} = s * P * R * S + t_{2d} \tag{2.4}$$

$$S_{model} = \overline{S} + \sum_{i=1}^{m-1} \alpha_i s_i + \sum_{i=1}^{m-1} \beta_i t_i \tag{2.5}$$

Here, X_{2d} is the point where the three-dimensional model is projected onto the two-dimensional plane, s is the scaling factor, P is an orthogonal projection matrix, R is a rotation matrix, S represents a three-dimensional face model, t_{2d} and is a displacement matrix.

There are also other 3D face reconstruction methods. PRNet was used reconstruct a 3D face model in the study [57]. This study saves 3D face information in a 2D position map and use a Convolutional Neural Network (CNN) to regress a 3D shape from a single 2D image. This method is modified in the paper [58]. Their method doesn't use the 3DMM face model in CNN algorithm.

2.3.3 Magic mirror

Magic mirror is also a kind of try-on system [16]. People can use the magic mirror to try different hair styles, makeup, and dressing. Users can find themselves a suitable appearance in the mirror.

FIGURE 2.16: Magic mirror [16]

There are two sub-systems in the magic mirror system: magic closet and beauty e-expert [16]. The magic closet helps users to find their suitable clothing. The beauty e-expert system is a system for users to try different hairstyles and makeup.

FIGURE 2.17: Hairstyle and makeup try on [16]

2.3.4 Try on systems using RGB-D sensors

Some studies uses RGB-D sensors to implement the virtual try-on systems. RGB-D sensors are sensors that can give image depth and color. There are many kinds of RGB-D sensors, such as Kinect v1, Kinect v2, Asus Xtion PRO LIVE, RealSense and Apple PrimeSense Carmine. Moreover, some of the smartphones are equipped with depth sensors [59]. An RGB-D image contains RGB information and depth information of the image. The depth information of an RGB-D image refers to the distance between the image plane and the corresponding object in the RGB image. A Kinect v2 sensor is used in the Kinect-Based try-on system to get the user's image [59]. This study downloaded 3D clothes models online and then align these models on the 2D body image.

FIGURE 2.18: Kinect v2 [17]

The Kinect sensor can detect 25 joints of tracked skeleton of the user, and these joints are used to match 3D clothes models.

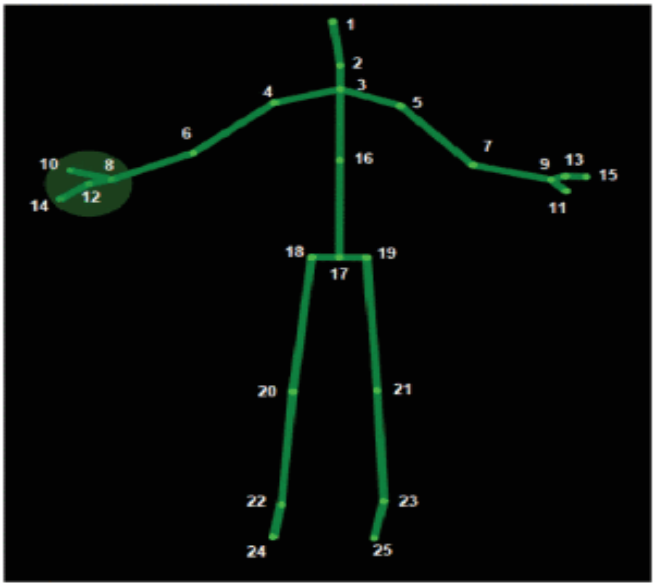


FIGURE 2.19: Joints of tracked skeleton [18]

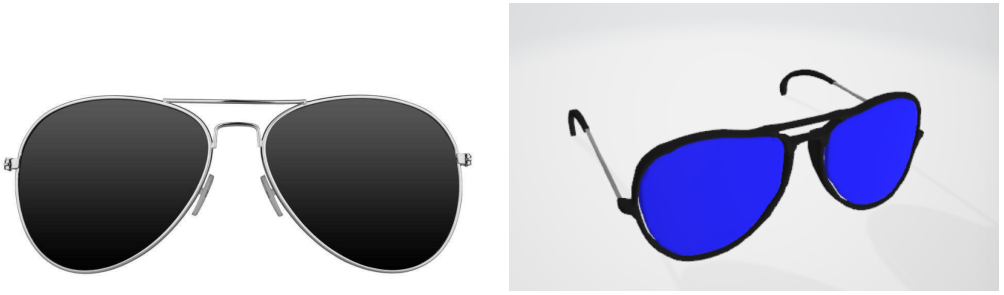
LiveScan3D is an open source library which has been used for 3D body reconstruction [18]. It has a feature of low cost. There are four Kinect v2 sensors in LiveScan3D. Each sensor is connected to a computer. Thus, we can use LiveScan3D to record dynamic scenes from multiple viewpoints at the same time. We can also clearly record the shape of objects when they are in motion.

The study [60] proposed a garment try-on system using Kinect V1 Xbox 360 sensor. The Kinect camera in this study is used to obtain the 3D information of the user's body. Besides, Kinect can also detect the user's pose and skin color.

3

The design

In this chapter, we will discuss the process to build a glasses try-on system in detail. There are two ways to build the glasses try-on system in this thesis. One method is to use 2D glasses pictures to complete the try-on system, and the other method is to use 3D glasses models to try on the glasses. This section will explain the implementation and differences between the two methods in detail.



(a) 2D glasses picture (b) 3D glasses model

FIGURE 3.1: 2D and 3D glasses

3.1 Development environment

This thesis uses OpenCV library for image processing and uses unity to develop AR projects. OpenCV is a cross-platform computer vision library which can be used for image processing, computer vision, and pattern recognition.

3.1.1 OpenCV

OpenCV(Open Source Computer Vision Library) was founded by INTEL in 1999. It can be used across platforms and can run on Windows, Linux, and Mac OS operating

systems. It has the advantages of high efficiency and lightweight. It is mainly written in C or C ++. The main interface of OpenCV is C ++, but it also provides interfaces for programming languages such as C-sharp, Python, and MATLAB. A lot of algorithms about pattern learning and machine processing have been implemented.

OpenCV has a wealth of algorithms for machine vision, from image processing to pattern recognition, from still images to dynamic video, from two-dimensional planes to three-dimensional reconstructions, covering most of the application fields Sof machine vision. And many of its algorithms have been well optimized, coupled with the fact that the code is basically written in C or C ++ and is completely open, we can transplant its algorithms, such as to ARM microprocessors, micro-controller systems.

OpenCV provides a good implementation algorithm for basic concepts and theories related to image processing, such as threshold segmentation, edge extraction, and template matching. OpenCV is open source and we can go to see and analyze its specific code, which is very helpful for us to learn the knowledge of image processing.

OpenCV has been widely used. Augmented reality, face recognition, gesture recognition, motion recognition, motion tracking, object recognition and image segmentation are the applications areas of OpenCV.

3.1.2 Unity

Unity is a game development platform. A lot of popular games have been created by Unity, such as Heroes of Warcraft, Kerbal Space Program and Wasteland 2. Unity is usually used to build projects, such as 2D or 3D games, VR or AR applications, web front-ends and 3D animation. It can run under Windows and Mac. We can use Unity to create AR applications and publish them to Windows, Mac, WebGL and Android platforms.

Due to the visualization and cross-platform development of Unity, most AR recognition SDK developers have launched plugins for Unity, such as Vuforia. Unity also launched multiple SDKs that support AR technology, including ARFoundation, ARKit Plugin For Unity and ARCore Plugin For Unity.

3.2 Face detection

Face detection is the first step for the system. In order to show glasses on people's face in the camera, we need to detect some important points in a human face, such as eyes,

mouth and nose. In this thesis, we use OpenCV and Dlib landmark detector for face detection.

There are two methods to do face detection in OpenCV. The first method is the Haar-Adaboost algorithm that is most commonly used in face detection. This algorithm is also used in other object detection. Adaboost is a set of machine learning frameworks. Based on the given positive samples and sub-samples, train a model for identifying objects such as positive samples. The essence of this model is a classifier, also known as a cascade classifier. The second method is based on the HOG + SVM algorithm in OpenCV.

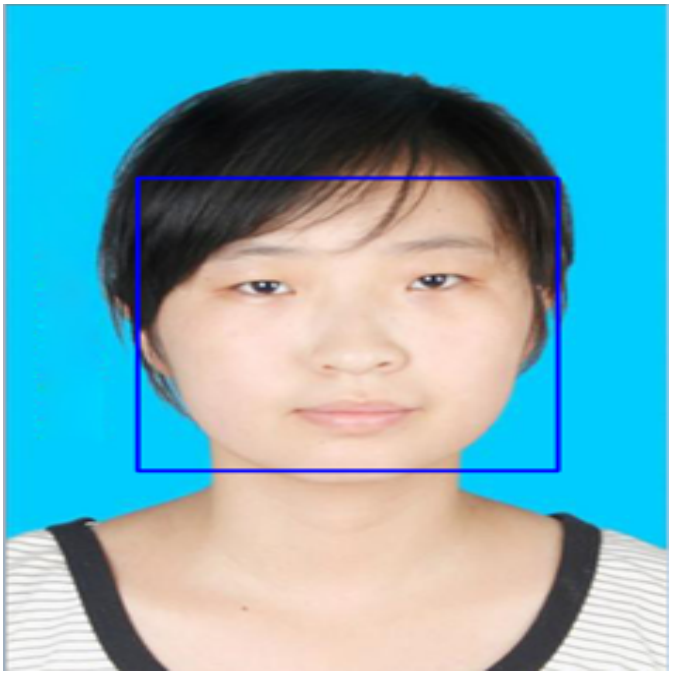


FIGURE 3.2: Face detection result

Landmark is a technology for extracting facial feature points. The Dlib can detect 68 important points in human's face which is very useful for this glasses try-on system. These points in the landmark can be used to extract the eye, mouth and nose area for fatigue detection, and the nose and other parts can be used for 3D pose estimation.

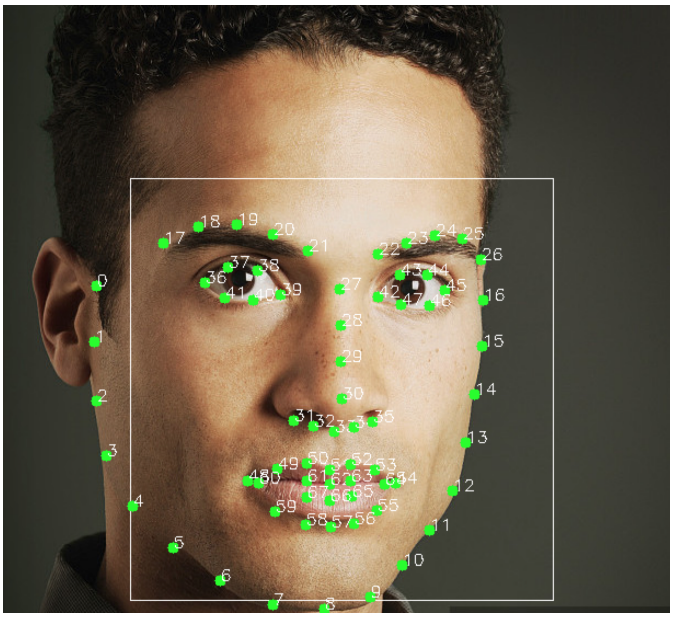


FIGURE 3.3: 68 dlib landmarks

Dlib landmark detection is proposed in the research [61]. Their method is based on the cascade of regressors. First, we need to understand what is a cascade of regressors.

Face alignment problem can be seen as learning a regression function F. The image I is the input. The output θ is the location of feature points. The face pose could be defined as follows:

$$\theta = F(I) \tag{3.1}$$

The cascade of regressors approximates the function F by learning multiple regression functions f1, ..., fn-1, fn. Then the face pose could be defined as:

$$\theta = F(I) = fn(f_{n-1}(...f_1(\theta_0, I), I), I) \tag{3.2}$$

$$\theta_i = f_i(\theta_{i-1}, I), i = 1, ..., n \tag{3.3}$$

The cascade means that the input of f_i relies on the output of f_{i-1}. The learning purpose of each f_i is to approximate the true position of the feature point.

In Dlib, the formula of the cascade of regressors is defined as follows:

$$\widehat{S}^{(t+1)} = \widehat{S}^{(t)} + r_t(I, \widehat{S}^{(t)}) \tag{3.4}$$

Among them, s^{t+1} represents the shape of the t-th level regressor, which is a vector composed of facial landmark coordinates, t represents the number of cascades, I is the image.

They use the gradient tree boosting algorithm to train each regressor. The relationship between the initial shape and the real shape is represented by a gradient tree. The features extracted from the current image during the input use pixel differences as features (features are the basis for tree splitting).

The first tree is built in gradient tree, and we need to input N pictures into this tree. Each picture will fall into one of the leaf nodes. After the whole pictures are completed (some leaf nodes may not have any pictures, some leaf nodes will have Multiple pictures), calculate the residual (the meaning of the residual is the difference between the current shape and the true shape of each picture), average all the residuals in the same leaf node, and keep it in this leaf. The second tree in gradient tree is based on the first tree, using residual + picture current shape $\geq$ current shape to update the current shape. And so on, to build a whole gradient tree.

The algorithm for the training method is as follows:

The training data is:

$$(I_{\pi_i}, \widehat{S}_i^t, \triangle S_i^{(t)})_{i=1}^N \tag{3.5}$$

The learning rate is:

$$0 < v < 1 \tag{3.6}$$

1. Initialise

$$f_0(I, \widehat{s}^{(T)}) = argmin \sum_{i=1}^{N} || \triangle S_i^{(t)} - \gamma||^2 \tag{3.7}$$

2. for k=1, ..., K:

   - Set for i =1, ..., N

$$r_{ik} = \triangle S_i^{(t)} - f_{k-1}(I_{\pi_i}, \widehat{S}_i^{(t)} \tag{3.8}$$

   - Fit a regression tree to the r_{ik} giving a weak regression function

$$g_k(I, \widehat{S}^{(t)}) \tag{3.9}$$

   - Update

$$f_k(I, \widehat{S}^{(t)}) = f_{k-1}(I, \widehat{S}^{(t)}) + v g_k(I, \widehat{S}^{(t)}) \tag{3.10}$$

3. Output

$$r_t(I, \widehat{S}^{(t)}) = f_k(I, \widehat{S}^{(t)}) \tag{3.11}$$

3.3 Glasses try-on system by using 2D glasses pictures

In this section, we will talk about the design using glasses pictures to build the glasses try-on system. Here is the flow chart for this glasses try-on system:

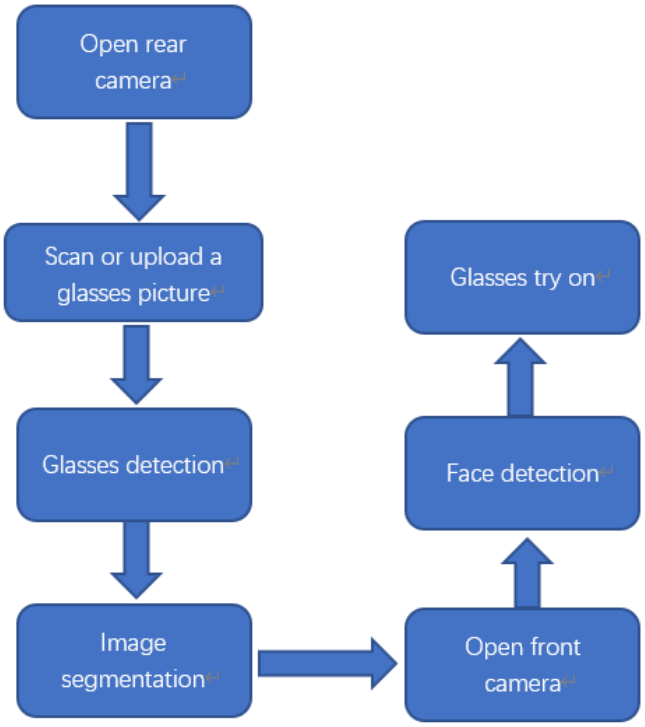
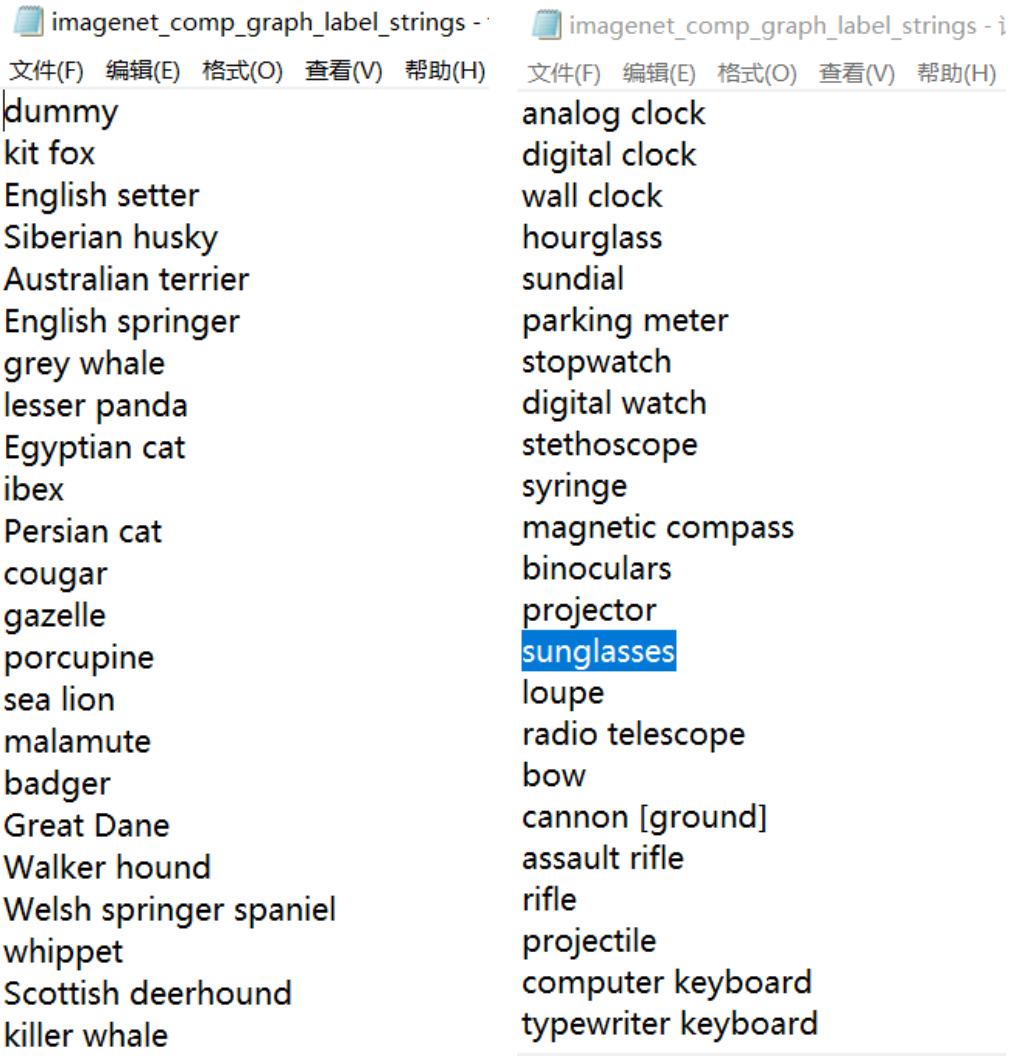


FIGURE 3.4: Flow chart of 2D glasses try-on system

3.3.1 Glasses detection and image detection

In this method, users need a glasses picture first. The picture can be taken by camera or downloaded online. We use a DNN (Deep Neural Networks) model to detect if there are glasses in the picture. OpenCV 3.3 already has the DNN module. There are three kinds of framework models in OpenCV DNN module, which are Caffe, TensorFlow and Yolo. TensorFlow model can identify around 1000 kinds of objects. More importantly, it can detect glasses. Thus, in this method, we use the TensorFlow model for glasses detection.



(a) tensorflow label 1 (b) tensorflow label 2

FIGURE 3.5: TensorFlow detection objects

3.3.2 Image segmentation

There may be many objects in the pictures that detect glasses, but we only need the information about the glasses. Thus, we need to do image segmentation to remove the unwanted information in the pictures.

Image segmentation is an effective way to understand images and it is also the first step to analyse images. As a hot topic of computer vision, a lot of research has been done on image segmentation. The aim of image segmentation is to find targets and its background in the picture and separate them. Image segmentation technology has been highly valued since the 1970s. Although researchers have proposed many methods for various problems in image segmentation, there is still no universally applicable theory

and method. However, many new ideas, new methods and improved algorithms have emerged in recent years.

Nowadays, image segmentation is frequently used in medical image analysis, machine vision, face recognition, satellite image processing and transportation systems [62]. The aim of image segmentation is to distinguish the targets and background in the image clearly. There are varieties of segmentation algorithms which are classified as follows [63]:

- Intensity based methods (such as threshold based method)
- Discontinuity based methods (such as edge detection)
- Similarity (region) based methods
- Clustering methods
- Graph based methods
- Pixon based methods
- Hybrid methods

This thesis will use Canny edge detection method for glasses image extraction. The edge detection methods are well developed in image processing. In this method, edges are detected first and then connect all the edges to form the glasses boundaries. There are mainly four steps for Canny edge detection which are filtering, edge enhancement, Non-maximum Suppression and Hysteresis Thresholding [64].

1. Filtering

   Edge refers to the mutation of gray level or structure in the image. The edge detection algorithms are mainly based on the difference operation. We usually use the first or second derivatives for differentiation. In the image, the edge can be seen as a pixel with a large first derivative or a second derivative with 0. However, the differentiation is very sensitive to noise, so the noise in the image can easily affect Canny detection result. We need to use filters to remove the noise in the image and improve the results of edge detection. This thesis will use Gaussian filter to reduce the noise.‘

2. Edge enhancement

   Edge enhancement is to detect the change in the intensity of the neighborhood of all the pixels in the image. This algorithm can find the points that have significant

changes in the neighborhood intensity values of the gray points of the image. In implementation, it can be determined by calculating the magnitude of the gradient. In this thesis, we will use to Sobel kernel to calculate the edge gradient of the image [64].

The image is filtered with the Sobel operator in both x and y directions. The Sobel kernel used in Canny is expressed as follows:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.12}$$

Then we can get two matrices as a result of Sobel filtering. The edge gradient G and direction θ of all the points in the image can be found using the following formula:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.13}$$

$$\theta = \arctan \frac{G_y}{G_x} \tag{3.14}$$

3. Non-maximum Suppression

   It's hard to say that pixels can be determined as edges only by using gradient values. The edges detected by the Sobel operator are too thick. Non-maximum suppression is a kind of edge sparse technology and we use it to refine the edges. We need to find the point that has the local maximum value. This point is the edge. Other pixels with non-maximum value are not edges and we need to remove them from the edges by setting their gray value to 0. In order to remove these unwanted points, each point in the image is examined to determine if it is the local maximum in its neighborhood in the gradient direction.

4. Hysteresis Thresholding

   The remaining pixels can more accurately represent the actual edges in the image. However, due to the noise and color changes in the image, there are still some edges is not the true edges. To improve the accuracy of the edge detection results, two thresholds of min value (minVal) and max value (maxVal) are used. All edges whose intensity gradient exceeds max value are determined as edges. The edges whose intensity gradient is less than the min value are not determined as edges and they will be removed. The judgment is whether it is based on its continuity; if it is connected to a pixel that is a "sure-edge" pixel, it is an edge; if it is connected to a pixel that is not a "sure-edge" pixel, it is not an edge (discard).

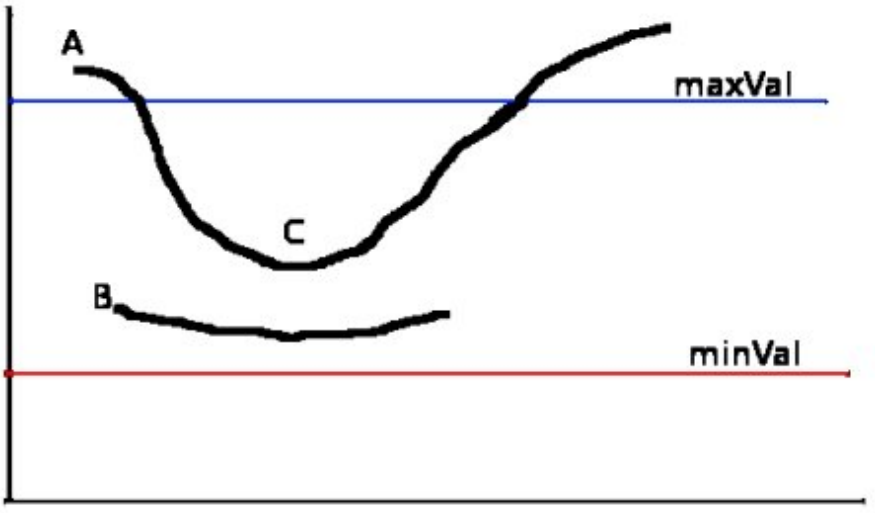


FIGURE 3.6: Hysteresis thresholding [19]

In the figure above, edge A is larger than the maxVal, which is showing us the edge A is the true edge. Edge C also is a true edge since it is connected with edge A even that edge is smaller. But edge B is different, it is not connected with any true edge, then it can be discarded.

3.4 Glasses try-on system by using 3D glasses models

The idea of this method is that the user scans a QR code and then users are able to see themselves wearing the 3D glasses in the camera. In this method, we will develop a 3D application in Unity. Here is the flow chart for this system:

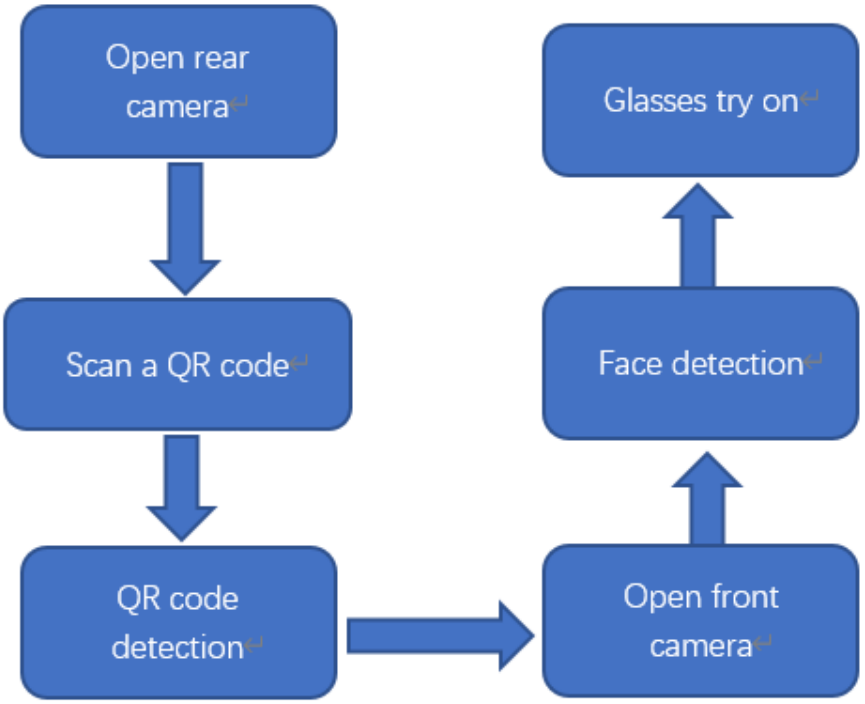


FIGURE 3.7: Flow chart of 3D glasses try-on system

First, we need to understand Unity's coordinate system. A 3D coordinate system represents a three-dimensional space. There are two kinds of 3D coordinate systems, which are left handed coordinate system and right handed coordinate system. In the left handed coordinate system, the X axis points to the right, the Y axis moves up and the Z axis points forward; while the X axis and Y axis of the right handed and the left handed coordinate system are the same, and Z axis points in opposite direction. OpenCV uses right hand coordinate. But in unity, we use left handed coordinate.

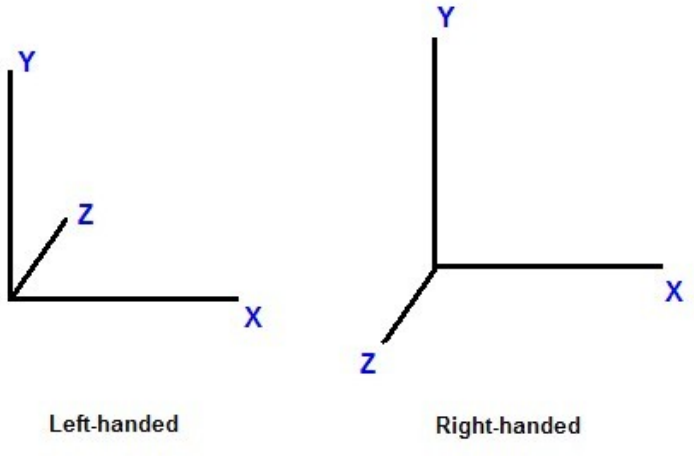


FIGURE 3.8: 3D coordinate system

There are four kinds of coordinate systems in Unity, which refer to world space coordinates, screen space coordinates, viewport space coordinates and GUI system.

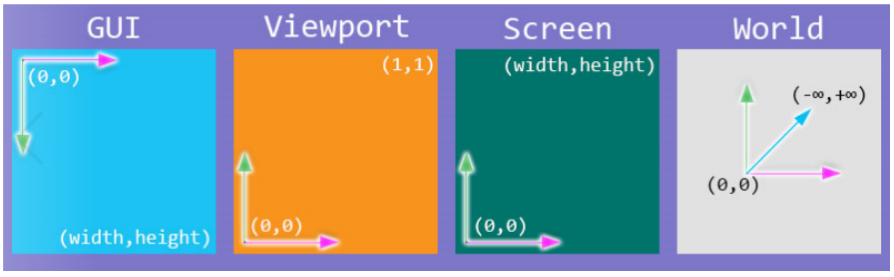


FIGURE 3.9: Unity coordinates

1. World space

   The left-handed coordinate system of the Unity engine is also called the world coordinate system. In general, the local coordinates and the origin of the world coordinate system are coincident. You cannot place all models on the origin of the world coordinate system, so you need to move the model. When the model is moved, the local coordinates of the model will be converted into world coordinates. The process of this movement is to convert the local coordinates of the model into world coordinates. The process of knowledge conversion is implemented inside the engine editor. In fact, the points of the model are multiplied by the world matrix.

2. Screen space

   The mobile coordinated mobile game developed by Unity often uses the screen coordinate system. The screen coordinates are the commonly used computer screens. It is in pixels. The origin point is in the lower-left corner of the screen, and the

upper right corner is the point (Screen.Width, Screen.Height). The position of a point in Z-axis is measured in the camera's world coordinates. Usually, click the object on the screen with the mouse, it is the screen coordinates. UI operations are also based on the screen coordinate system.

3. Viewport space

   You can see objects in the virtual world only through the camera. The camera has its own viewport coordinates, and objects must be converted to viewport coordinates to be seen. The lower-left corner of the camera's viewport is (0,0) and the upper right corner is (1,1). The position of a point in Z-axis is measured in the camera's world coordinates. The (0,0) point and the (1,1) point are calculated by scaling.

4. GUI system

   When we are doing Unity game development, we often use GUI to do some tests, such as displaying a button to control the game. The coordinate system where this Button is located in the GUI coordinate system. Its origin point (0, 0) is in the upper left corner. The screen width is Screen.width and the height is Screen.height, so the coordinates of the lower right corner of the GUI system are: (Screen.width, Screen.height), which is a two The coordinate system of the dimension, the value of a point in z-axis is 0.

Next, let's talk about the transformation relationship between the four coordinate systems.

- World coordinate to camera coordinate

  We see the world with a camera. If the camera is moved anywhere, it could be able to see anywhere in the world. If the camera is located in a location(t_x, t_y, t_z), We can say that the camera coordinates are translated with respect to the world coordinates. The camera may be looking in any direction. We can say that the camera is rotated relative to the world coordinate system [65]. So we can use a rotation matrix and a translation vector to represent the relationship between world coordinates and camera coordinates.

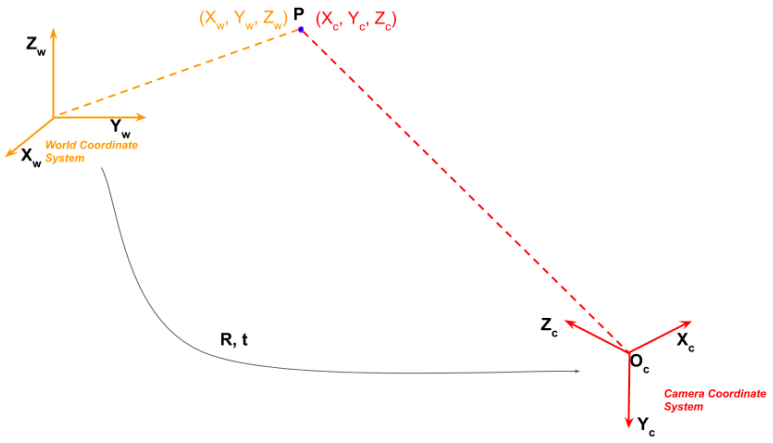


FIGURE 3.10: World coordinate to camera coordinate system

In the figure above [65], P means a point in world space. R means rotation matrix. T means translation vector.

The coordinate system transformation of a 3D object can be represented by the rotation transformation of the coordinate system and the translation transformation, and this is the same with the conversion relationship between the world coordinate system and the camera coordinate system. Rotate different angles around different axes to get different rotation matrices.

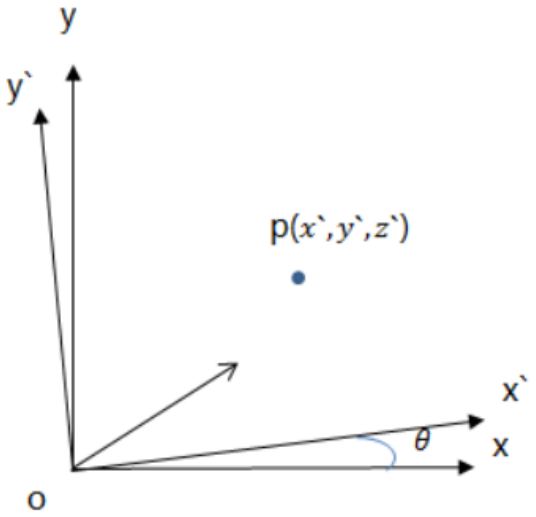


FIGURE 3.11: Rotate the coordinate system around the z axis

The transformation can be represented by the following formula:

$$x = x'cos\theta - y'sin\theta \tag{3.15}$$

$$y = x'sin\theta + y'cos\theta \tag{3.16}$$

$$z = z' \tag{3.17}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_1 \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{3.18}$$

Similarly, rotate the coordinate system around the x-axis and the y-axis, and we can get:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = R_2 \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} cos\varphi & 0 & -sin\varphi \\ 0 & 1 & 0 \\ sin\varphi & 0 & cos\varphi \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = R_3 \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{3.20}$$

Then, we can get the rotation matrix:

$$R = R_1 R_2 R_3 \tag{3.21}$$

Finally we can get the coordinate of point P in camera coordinate:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \tag{3.22}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{3.23}$$

- Camera coordinate to image coordinate

  The transformation from the camera coordinate system to the image coordinate system belongs to the perspective projection transformation, and it is transformed from 3D to 2D.

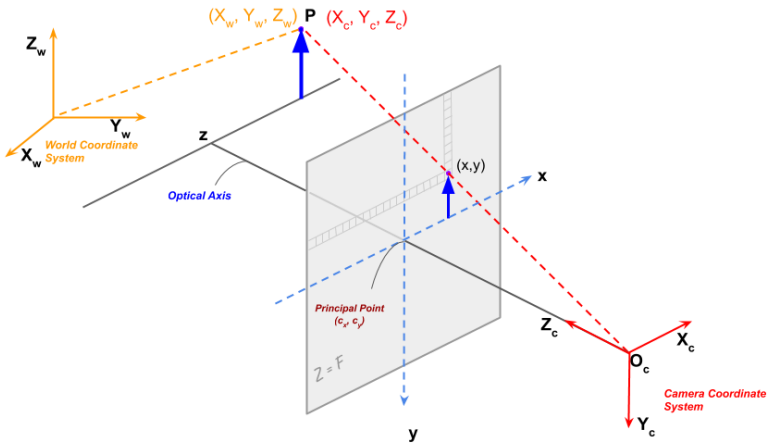


FIGURE 3.12: Camera coordinate to image coordinate

This transformation can be represented by the following:

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.24}$$

- Image coordinate to pixel coordinate. This transformation is different from the previous coordinate system transformation. At this time, there is no rotation transformation, but the coordinate origin position is not the same, and the size is not the same. Then design the telescopic transformation and translation transformation.

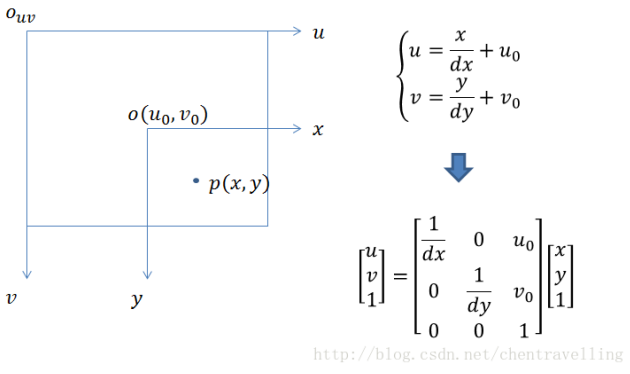


FIGURE 3.13: Image coordinate to pixel coordinate

The transformation is represented by the formula:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/(dx) & 0 & u_0 \\ 0 & 1/(dy) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{3.25}$$

3.4.1 3D head pose estimation

Face pose estimation is mainly to obtain the angle information of the face orientation and position with respect to the camera. The face orientation information obtained in this research is represented by three Euler angles (pitch, yaw, roll).

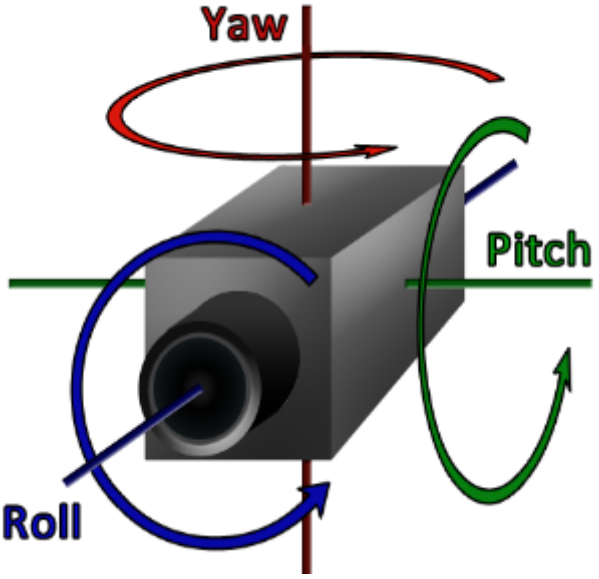


FIGURE 3.14: Euler angles

- Pitch: pitch angle, which means that the object rotates around the x-axis.
- Yaw: yaw angle, which means that the object rotates around the y-axis.
- Roll: roll angle, which means that the object rotates around the z-axis.

There are four steps to calculate the three Euler angles:

- Use face detection and Dlib landmark detection to obtain the 2D locations of n key points in a face image (n can be defined according to its tolerance for accuracy, here we select 6 key points).;

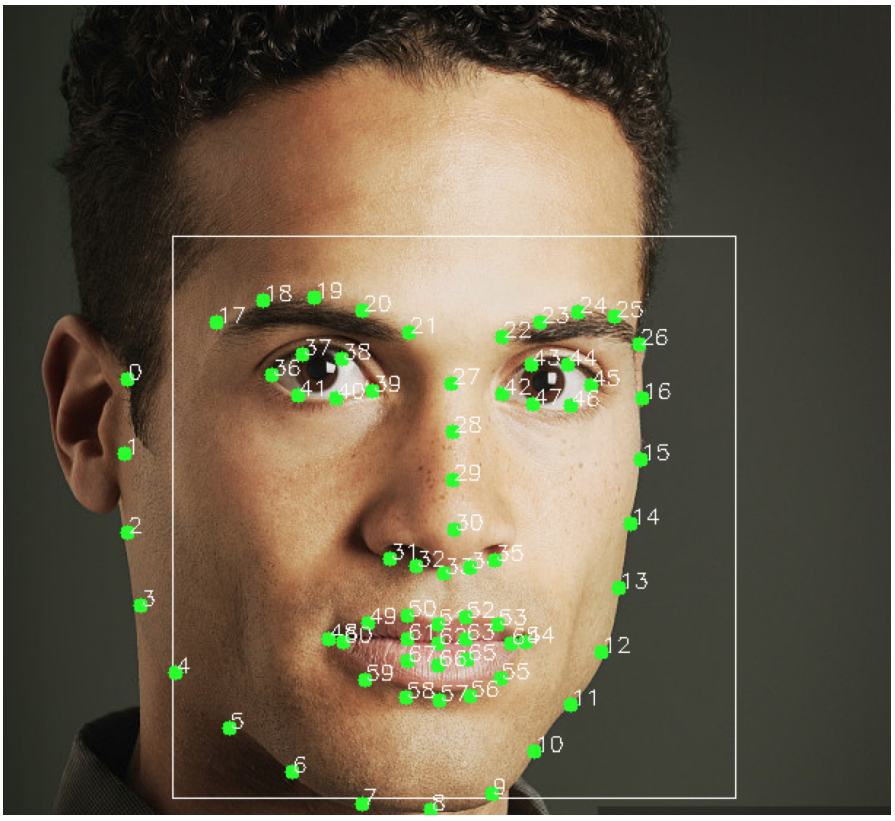


FIGURE 3.15: Dlib detection

- Select 6 key points in a 3D face model. This research defines a 3D face model with 6 key points (left eye, right eye, nose tip, Left mouth corner, right mouth corner and jaw);
- Camera calibration
- Solve the rotation vector using OpenCV's solvePnP function;
- Convert the rotation vector to Euler angle.
- Calculate the face location.

3.4.2 Dlib landmark detection

We can use Dlib landmark detector to detect 68 key points in human's face. According to the 6 3D points we select above, the 6 2D points should be:

- Nose: points(30)
- Chin: points(8)
- Left eye corner: points(36)

- Right eye corner: points(45)
- Left mouth corner: points(48)
- Right mouth corner: points(54)

3.4.3 Define 6 key points of a 3D face model

In this method, we need to find the 3D locations of the 6 key points selected in a 2D image. There are two ways to find the 3D locations of the key points. One is to build a 3D face model for the people in the image. The other way is to use a generic 3D face model. We use the following 3D points in this research [66]:

- Nose (0.0,0.0,0.0)
- Chin (0.0, -330.0, -65.0)
- Left eye corner (-225.0, 170.0, -135.0)
- Right eye corner (225.0, 170.0, -135.0)
- Left mouth corner (-150.0, -150.0, -125.0)
- Right mouth corner (150.0, -150.0, -125.0)

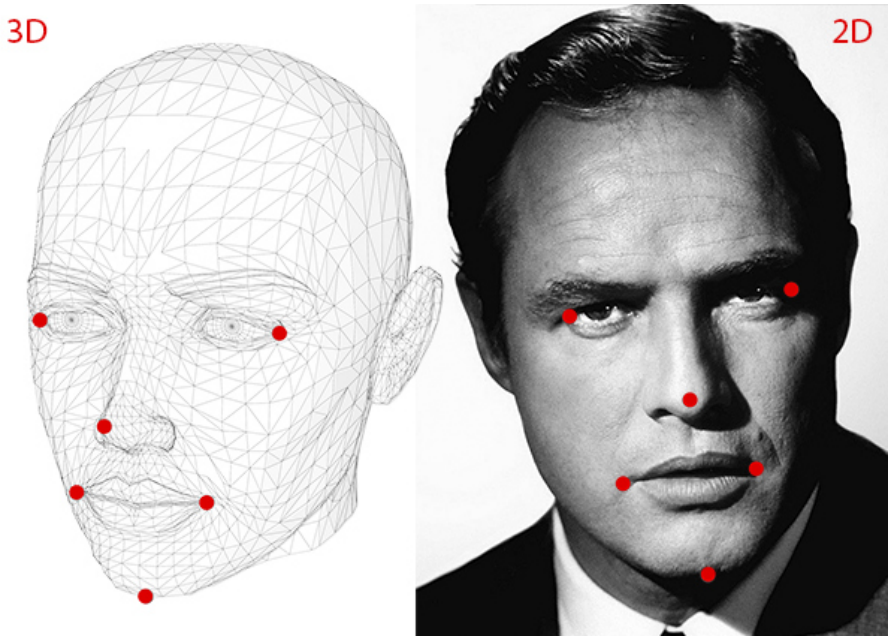


FIGURE 3.16: 6 key points in human's face

3.4.4 Camera calibration

The world we live in is three-dimensional, and the photos are two-dimensional, so we can think of the camera as a function, the input is a scene, and the output is a grayscale image. The goal of camera calibration is to find a suitable mathematical model and find the parameters of this model. Then we can use the mathematical model to approximate the process from the 3D world to the 2D image. This approximation process is called "camera calibration". We use simple mathematical models to express complex imaging processes and find the inverse process of imaging. The camera after calibration can reconstruct the three-dimensional scene, which is also called the perception of depth.

We have known the 3D locations of the 6 key points and their 2D positions in the image of the human face. Then 3D-points could be projected on the 2D image. To find the relationship between the points of a 3D face model in the real world and its corresponding 2D-points in the image, we have to build a mathematical model of the camera imaging. These parameters of the mathematical model are camera parameters. There are two kinds of parameters of the camera, which are intrinsic parameters and extrinsic parameters. Intrinsic parameters refer to focal length, optical center and radial distortion coefficients. The rotation and translation of the camera relative to the world coordinate system are the extrinsic parameters [67]. The focal-length and optical center can be estimated. In this method, the focal-length is the image width in pixels and the optical center is the image center [66]. The distortion coefficients are set to zero. Then we can get intrinsic parameters of the camera. the function below is showing the camera intrinsic matrix :

$$k = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.26}$$

Among them, f_x and f_y are the focal lengths. In general, they are equal. Here $f_x = f_y =$ image width. $x0$ and $y0$ are the main point coordinates (relative to the imaging plane). In this method, $x_0 =$ image width/2 and $y_0 =$ image height/2. s is the coordinate axis tilt parameter, and ideally 0.

The distortion coefficients in this research is defined as:

$$d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.27}$$

Extrinsic parameters can be calculated by PnP(Perspective-n-Point) algorithm. When we know n 3D space points and their projection points in a 2D image, PnP is a problem of estimating the positions of the points in the camera coordinate system. The PnP problem has several solutions, such as P3P, EPnP, Direct Linear Transform (DLT) and Levenberg-Marquardt optimization [68]. This research will use Levenberg-Marquardt optimization and EPnP to solve PnP problem. The results of these two methods will be compared.

- Levenberg-Marquardt optimization

  In the solution of Levenberg-Marquardt optimization, the rotation matrix and the translation vector are chosen randomly firstly. Then project the 6 points in the 3D world space to the 2D image using the selected rotation matrix and translation vector. Then calculate the errors between the projected points and the 2D coordinate points extracted by the feature point extraction algorithm. Finally, the rotation matrix and the translation vector could be found by using the iteration algorithm. The iteration is based on Levenberg-Marquardt optimization and the optimization is performed by minimizing the function below [69]:

$$F(x) = \sum_{i=1}^{n} f(M_i, k_1, k_2, p_1, p_2, R_i, t_i) = \sum_{i=1}^{n} (u_{di} - u_j)^2 + (v_{di} - v_i)^2 \tag{3.28}$$

  Among them, (u_{di}, v_{di}) is the projected 2D points. (u_j, v_i) is the actual coordinates of the points. k_1, k_2, p_1, p_2 are the distortion coefficients, and they are set to zero. Then we use Levenberg-Marquardt optimization to minimize the function F(x). A Jocobian matrix is defined follows in Levenberg-Marquardt optimization:

$$J(x_k) = \begin{bmatrix} \frac{\partial F}{\partial x_0} & \cdots & \frac{\partial F}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial F}{\partial x_0} & \cdots & \frac{\partial F}{\partial x_n} \end{bmatrix} \tag{3.29}$$

  The step length for Levenberg-Marquardt optimization is [69]:

$$d_k = -(J(x_k)^T J(x_k) + uI)^{-1} J(x_k) F(x_k) \tag{3.30}$$

  Among them, I is the unit matrix. u is the Levenberg-Marquardt parameter. The iteration steps are as follows [69]:

  1. Set the parameter u and accuracy ε;
  2. Calculate F(x) and J(x);
  3. Calculate d_k, and $x^{k+1} = x^k + d_k$;

4. If $F(x^{k+1}) < F(x^k)$ and $||d_k|| < \varepsilon$ stop the iteration. If $F(x^{k+1}) < F(x^k)$ and $||d_k|| \geq \varepsilon$, set u=u/10 and go to step 2;
5. If $F(x^{k+1}) \geq F(x^k)$, set $u = 10u$ and calculate d_k, go to step 4.

Finally we can get the extrinsic parameters of the camera.

- EPnP

  The EPnP solution was proposed by the paper [70]. Compared with Levenberg-Marquardt optimization, EPnP is a non-iterative algorithm. There are at least 4 key points used in the EPnP algorithm. The world coordinates of each point are expressed by a weighted sum of virtual control points. Besides, we use four virtual control points and theses points must be non-coplanar.

  The world coordinates of the key points are:

$$P_i, \quad i = 1, ..., n. \tag{3.31}$$

  The expressions of the 4 control points are as follows:

$$C_j, \quad j = 1, 2, 3, 4 \tag{3.32}$$

  The relationship between the key points in the world coordinates and the control points is as follows:

$$P_i^w = \sum_{j=1}^{4} \alpha_{ij} C_j^w, \quad with \quad \sum_{j=1}^{4} \alpha_{ij} = 1 \tag{3.33}$$

  This relationship could also be expressed as follows:

$$\begin{bmatrix} P_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} C_1^w & C_2^w & C_3^w & C_4^w \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1^w \\ \alpha_2^w \\ \alpha_3^w \\ \alpha_4^w \end{bmatrix} \tag{3.34}$$

  The points in camera coordinates could be expressed as:

$$P_i^c = \sum_{j=1}^{4} \alpha_{ij} C_j^c \tag{3.35}$$

  In the formulas above, w means that the points coordinates are expressed in the world coordinate system, c means that the points coordinates are expressed in the camera coordinate system, and α_{ij} are homogeneous barycentric coordinates.

To increase the stability of the solution, the first control point is the centroid of the key points, which is:

$$C_1^w = \frac{1}{n}\sum_{i=1}^{n} P_i^w \tag{3.36}$$

The remaining three points are aligned with the main axis direction. First, we need to calculate the matrix:

$$A = \begin{bmatrix} (P_1^w)^T - (C_1^w)^T \\ ... \\ (P_n^w)^T - (C_1^w)^T \end{bmatrix} \tag{3.37}$$

Then calculate the three eigenvalues λ_1, λ_2 and λ_3 of $A^T A$. The eigenvectors for the three eigenvalues are v_1, v_2, v_3. The remaining three points are expressed as:

$$\begin{aligned} C_2^w &= C_1^w + \sqrt{\tfrac{\lambda_1}{n}} v_1 \\ C_3^w &= C_1^w + \sqrt{\tfrac{\lambda_2}{n}} v_1 \\ C_4^w &= C_1^w + \sqrt{\tfrac{\lambda_3}{n}} v_1 \end{aligned} \tag{3.38}$$

So far, we know the locations of these 4 control points in the world space, and the homogeneous barycentric positions of each 3D point. If we can solve the coordinates of the 4 control points in the camera coordinate system, we can calculate the coordinates of these 3D points in the camera coordinate system, and then we can solve the camera external parameters.

We have known the 3D key points P_i, the projection of the 3D key points on the pixel plane $U_i(u_i, v_i)$, the camera intrinsic matrix K and the scalar projective parameters w_i. We can get:

$$w_i \begin{bmatrix} U_i \\ 1 \end{bmatrix} = K P_i = K \sum_{j=1}^{4} a_{ij} C_j^c \tag{3.39}$$

The formula above could also be expressed as:

$$w_i \begin{bmatrix} U_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} a_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{3.40}$$

Then we can get the following formula:

$$\begin{aligned} \textstyle\sum j = 1^4 (\alpha_{ij} f_x x_j^c + \alpha_{ij}(c_x - u_i) z_j^c) = 0 \\ \textstyle\sum j = 1^4 (\alpha_{ij} f_y y_j^c + \alpha_{ij}(c_y - v_i) z_j^c) = 0 \end{aligned} \tag{3.41}$$

In the formula above, the homogeneous barycnetric coordinates α, the camera intrinsic parameters f_x, f_y, c_x, c_y, the 2D points (u_i, v_i) are known variables. Unknown variables are the positions of these 4 control points in the camera space, x_i^c, y_i^c and z_i^c. There are a total of 12 unknown parameters. One point can determine 2 equations. All the points can form a linear equation:

$$Mx = 0 \tag{3.42}$$

Among the equation, x represents the 12 unknown parameters and the size of M is 2n x 12. Then, the solution of the above equation is:

$$x = \sum_{i=1}^{N} \beta_i V_i \tag{3.43}$$

V_i is the right singular vector of M and the corresponding singular value is 0. The specific solution method is to find the eigenvalues and eigenvectors of $M^T M$, and the eigenvectors with eigenvalues of 0 are V_i. Note that no matter how many point pairs, the size of $M^T M$ is always 12 x 12. The next question is how to determine the coefficient β and there are four cases for the question:

1. N=1

$$x = \beta V \tag{3.44}$$

2. N=2

$$x = \beta_1 V_1 + \beta_2 V_2 \tag{3.45}$$

3. N=3

$$\begin{gathered} L\beta = \rho \\ L = [V_1, V_2, V_3] \\ \beta = [\beta_{11}, \beta_{12}, \beta_{13}, \beta_{22}, \beta_{23}, \beta_{33}] \end{gathered} \tag{3.46}$$

4. N =4

$$\begin{gathered} L\beta = \rho \\ L = [V_1, V_2, V_3] \\ \beta = [\beta_{11}, \beta_{12}, \beta_{13}, \beta_{14}, \beta_{22}, \beta_{23}, \beta_{24}, \beta_{33}, \beta_{34}, \beta_{44}] \end{gathered} \tag{3.47}$$

At this point, the initial values of β_1, β_2, β_3 and β_4 are solved. Then we use Gauss-Newton to optimize the results. The equation for Gauss-Newton optimization:

$$Error(\beta) = \sum_{(i,j) \quad s.t. \quad i<j} (||C_i^c - C_j^c||^2 - ||C_i^w - C_j^w||^2) \tag{3.48}$$

The control point coordinates in the camera coordinate system are expressed as:

$$C_i^c = \sum_{j=1}^{4} \beta_j V_j^{[i]} \tag{3.49}$$

Among the equation, V_j^i represents a vector of 3 elements occupied by the ith control point in V.

The steps for calculating camera extrinsic parameters are:

1. Calculate centroid coordinates Centroid coordinates in the camera coordinate system:

$$P_0^c = \frac{1}{n} \sum_{i=1}^{n} P_i^c \tag{3.50}$$

Cnetroid coordinates in the world coordinate system:

$$P_0^w = \frac{1}{n} \sum_{i=1}^{n} P_i^w \tag{3.51}$$

2. Remove centroid

$$P^c = \begin{bmatrix} (P_1^c)^T - (P_c^c)^T \\ ... \\ (P_N^c)^T - (P_c^c)^T \end{bmatrix} \tag{3.52}$$

$$P^w = \begin{bmatrix} (P_1^w)^T - (P_c^w)^T \\ ... \\ (P_n^w)^T - (P_c^w)^T \end{bmatrix} \tag{3.53}$$

3. Calculate rotation matrix

$$H = (P^c)^T P^w \tag{3.54}$$

The SVD(singular value decomposition) for H is:

$$[U \Sigma V] = SVD(H) \tag{3.55}$$

The rotation matrix R is:

$$R = UV^T \tag{3.56}$$

4. Calculate translation matrix t

$$t = P_c^c - RP_c^w \tag{3.57}$$

Please note, there are four solutions of β. Therefore, the calculation here is actually performed four times, and the solution with the smallest reprojection error is finally selected.

3.4.5 Euler angle

The output of the PnP problem includes a rotation matrix and a translation vector. Here we need to convert the rotation information into Euler angles.

The rotation matrix is one of the representations of object rotation information, and it is a common representation in OpenCV. In addition to the rotation matrix, there are Euler angle, Direction Cosine Matrix, Quaternion, and Axis-Angle. Among these rotation representation methods, matrix rotation and Euler rotation are two more commonly used representation methods. Matrix rotation uses a 4X4 matrix to represent a transformation matrix rotated around an arbitrary axis. While Euler angles define the rotation order of the object, how many degrees to rotate about a certain axis. Different rotation orders will lead to different rotation results.

In this section, we will introduce the method of converting the rotation vector to Euler angle.

There are two methods to convert the rotation vector to Euler angle:

$$rotationvector \rightarrow Quaternion \rightarrow Eulerangle \tag{3.58}$$

$$rotationvector \rightarrow rotationmatrix \rightarrow Eulerangle \tag{3.59}$$

1. Convert rotation vector to Quaternion

   Any rotation in the 3D world can be expressed by rotating around an axis through a certain angle, which is the Axis-Angle representation method. We can use a 3D vector (x, y, z) to express an axis, and use θ to represent the rotation angle. Intuitively, a four-dimensional vector (theta, x, y, z) can represent any rotation in 3D space.

   Note that the 3D vector (x, y, z) is only used to indicate the direction of the axis, so a more compact representation is to use a unit vector to indicate the direction axis, and the length of the 3D vector to represent the angle value theta. In this way, a 3D vector (theta * x, theta * y, theta * z) can be used to represent any rotation in 3D space, provided that (x, y, z) is a unit vector. This is how the rotation vector is represented.

   Quaternion is also a commonly used representation of rotation. Assuming that (x, y, z) is a unit vector in the axis direction, and theta is the angle turned around

the axis, then the quaternion can be expressed as:

$$q = \begin{bmatrix} cos(\theta/2) \\ xsin(\theta/2) \\ ysin(\theta/2) \\ zsin(\theta/2) \end{bmatrix} \tag{3.60}$$

Because the conversion from quaternion to Euler angle formula is simpler, we will first convert the rotation vector to quaternion. The codes for this conversion are as follows:

```
theta = cv2.norm(rotation_vector, cv2.NORM_L2)
q.w = math.cos(theta / 2)
q.x = math.sin(theta / 2)*rotation_vector[0][0] / theta
q.y = math.sin(theta / 2)*rotation_vector[1][0] / theta
q.z = math.sin(theta / 2)*rotation_vector[2][0] / theta
```

The quaternion to Euler angle conversion formula is as follows:

$$\begin{bmatrix} \varphi \\ \theta \\ \end{bmatrix} = \begin{bmatrix} arctan\frac{2(wx+yz)}{1-2(x^2+y^2)} \\ arcsin(2(wy-zx)) \\ arctan\frac{2(wx+yz)}{1-2(x^2+z^2)} \end{bmatrix} \tag{3.61}$$

The result of arctan and arcsin is [-π/2, π/2], which cannot cover all Euler angles, so atan2 is used instead of arctan:

$$\begin{bmatrix} \varphi \\ \theta \\ \end{bmatrix} = \begin{bmatrix} atan2(2(wx+yz), 1-2(x^2+y^2)) \\ arcsin(2(wy-zx)) \\ atan2(2(wx+yz), 1-2(x^2+z^2) \end{bmatrix} \tag{3.62}$$

The codes for the conversion from Quaternion to Euler anle are:

```
void quaterniondToEulerAngle(Quaterniond& qua, double& roll,
                             double& yaw, double& pitch)
{
    double m = qua.y * qua.y;

    // pitch
    double p1 = +2.0 * (qua.w * qua.x + qua.y * qua.z);
    double p2 = +1.0 - 2.0 * (qua.x * qua.x + m);
```

```
        pitch = std::atan2(p1, p2);

        // yaw
        double y = +2.0 * (qua.w * qua.y - qua.z * qua.x);
        y = y > 1.0 ? 1.0 : y;
        y = y < -1.0 ? -1.0 : y;
        yaw = std::asin(y);

        // roll
        double r1 = +2.0 * (qua.w * qua.z + qua.x * qua.y);
        double r2 = +1.0 - 2.0 * (m + qua.z * qua.z);
        roll = std::atan2(r1, r2);
    }
```

2. Convert rotation vector to rotation matrix When calculating coordinate transformations, a more convenient representation of rotation is the Rotation Matrix. In 3D space, we use a 3x3 matrix to represent the rotation matrix. The calculation method for converting Euler angles into a rotation matrix is as follows (Assuming Euler angles yaw, pitch, and roll are φ, θ and ψ):

$$\begin{aligned} R(\varphi,\theta,\psi) &= R_z(\varphi)R_y(\theta)R_x(\psi) \\ &= \begin{bmatrix} cos\varphi cos\theta & cos\varphi sin\theta sin\psi - sin\varphi cos\psi & cos\varphi sin\theta cos\psi + sin\varphi sin\psi \\ sin\varphi cos\theta & sin\varphi sin\theta sin\psi + cos\varphi cos\psi & sin\varphi sin\theta cos\psi - cos\varphi sin\psi \\ -sin\theta & cos\theta sin\psi & cos\theta cos\psi \end{bmatrix} \end{aligned} \tag{3.63}$$

Among this,

$$R_z(\varphi) = \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.64}$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{3.65}$$

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix} \tag{3.66}$$

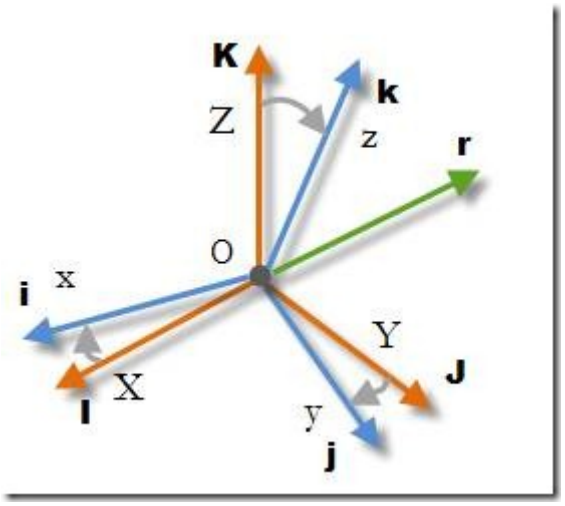


FIGURE 3.17: Rotation angle

In this method, we can use Rodrigues function in OpenCV to convert a rotation vector to rotation matrix. This can be implemented by the following codes:

```
//rotM is the rotation matrix, rvev is the rotation vector
rotM = new Mat(3,3,CvType.CV_64FC1);
Calib3d.Rodrigues(rvec, rotM);
```

Another name for the rotation matrix is the Direction Cosine Matrix (DCM), which is more commonly used in the field of gyro mechanics. The origin of the name DCM is actually another way of expressing three-dimensional rotation with three angle values in addition to Euler angles. It is assumed that the vector of the three coordinate axes is I, J, K when the rigid body starts, and The vector of the three coordinate axes when the target is facing is i, j, k. Then the rotation can be represented by the angle between the three coordinate axes and the original coordinate axis, as shown in the figure below: The rotation angle could be calculated by the following codes:

```
//rotM is the rotation matrix
double x = Math.Atan2(rotM.get(2, 1)[0],
                              rotM.get(2, 2)[0]);
double y = Math.Atan2(-rotM.get(2, 0)[0],
                Math.Sqrt(rotM.get(2, 1)[0] *
                        rotM.get(2, 1)[0] +
                        rotM.get(2, 2)[0] *
                        rotM.get(2, 2)[0]));
```

```
double z = Math.Atan2(rotM.get(1, 0)[0],
                              rotM.get(0, 0)[0]);
x = x * (180 / Math.PI)+180;
y = y * (180 / Math.PI);
z = z * (180 / Math.PI);
```

Finally, we can use Quaternion. Euler function in OpenCV to convert the rotation angles to Euler angles, which is as follows:

```
glasses.transform.rotation =
Quaternion.Euler((float)x,
                 (float)y, (float)z);
```

3.5 QR code

3.5.1 QR code structure

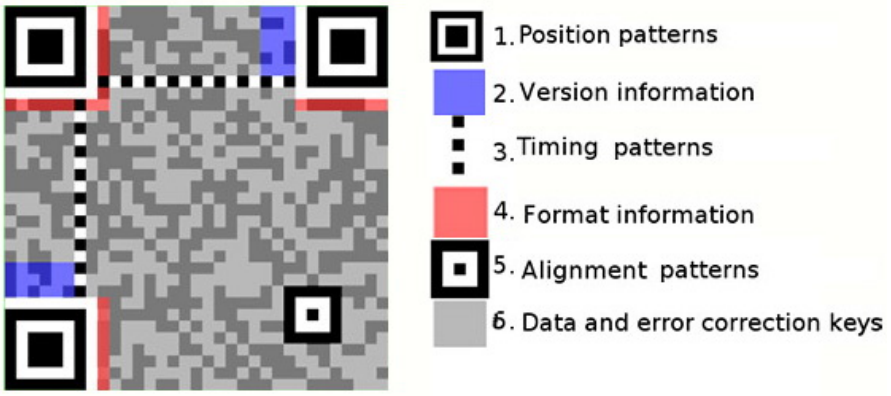


FIGURE 3.18: The structure of QR code

The picture above shows a basic structure of the QR code. From the figure, we can see that there are 5 parts in the QR code [71]:

1. Position patterns

   In the QR code image, the position patterns are placed at three corners. These patterns are used for the definition of the location, size and angle of the QR code. Thus, they can help the scanning device to recognize the code quicker.

2. Version information

   This area contains the version information of the QR code.

3. Timing patterns

   These patterns are represented by two lines. They are used to calculate the coordinate of a symbol [61].

4. Format information

   These patterns are used to define the data format encoded in QR code.

5. Alignment patterns

   The scanning device can define the perspective distortion of the QR code image using alignment patterns.

6. Data and error correction keys

   The data contains the code to be saved. They are composed of black squares and lines, which are named as modules. The number of modules can vary depending on the amount of information.

3.5.2 QR code decoding

This thesis uses OpenCV QR code detector to do the decode. The positioning in opencv is based on a fixed ratio of the black and white intervals of the position patterns. This ratio is 1: 1: 3: 1: 1, that is, five horizontal segments with horizontal or vertical scanning are regarded as part of the positioning pattern.

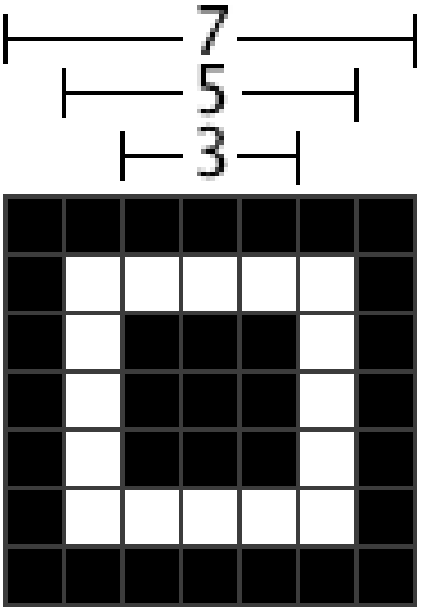


FIGURE 3.19: Ratio of the black and white intervals

The steps to positioning a QR code are as follows:

1. A set of center points of the three position patterns are obtained after horizontal and vertical scanning.
2. The k-means clustering is applied to iterate the three sets into three center points, then the center points of the position patterns are obtained.
3. The relationship between the angle and the area is used to determine the order of the center points.
4. Obtain the outer frame of the position patterns using floodfill function in OpenCV. Determines the bottom left and top right feature points based on the furthest diagonal distance, and determine the top left feature points based on the distance relationship.
5. Use the distance to determine the two points in the lower left and upper right patterns that are furthest from the upper left corner. The previous feature points intersect to form the lower right corner feature point.
6. Perform perspective transformation.
7. Decoding.

4

The implementation

4.1 The data

This section will introduce the data used in this project. There are two kinds of data used in this system, which are 2D glasses images and 3D glasses models. 2D glasses images are downloaded from google picture. There are a lot of glasses images online. We can search them in google and download.

3D glasses models can be created using a 3D modelling software, such as C4D and MAYA. Or you can download free glasses models online. There are many free 3D glasses models online. Here we will give several websites to download 3D glasses models.

- https://free3d.com/3d-models/glasses
- http://www.cadnav.com/3d/Glasses.html
- https://archive3d.net/?tag=glasses
- https://sketchfab.com/tags/glasses
- https://www.turbosquid.com/Search/3D-Models/free/glasses

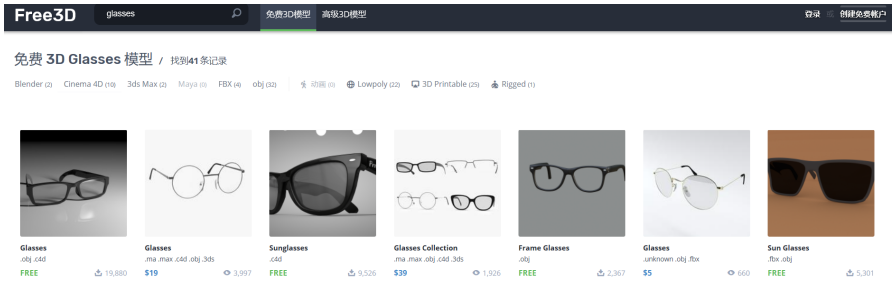


FIGURE 4.1: A website to download 3D glasses models

4.2 QR code generator

The idea for the glasses try-on system is that when people scan the QR code of a 2D glasses image or a 3D glasses model, they will see themselves wearing the glasses in the camera. Thus, we need to create QR codes for 2D glasses images and 3D glasses models firstly. There are many free QR code generators online. Here we will give an example to generate a QR code:

1. Find the URL: https://www.the-qrcode-generator.com/

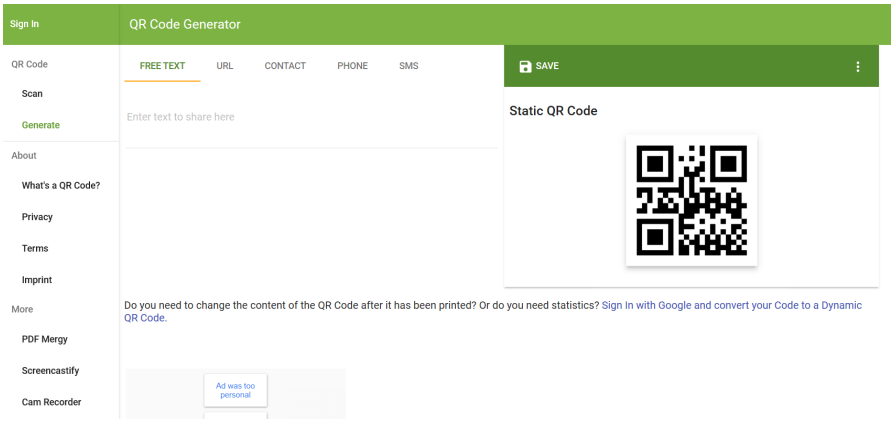


FIGURE 4.2: QR generator

2. Select a category you'd like to enter, here we use "FREE TEXT".

3. Input the text you want to save in the QR generator.

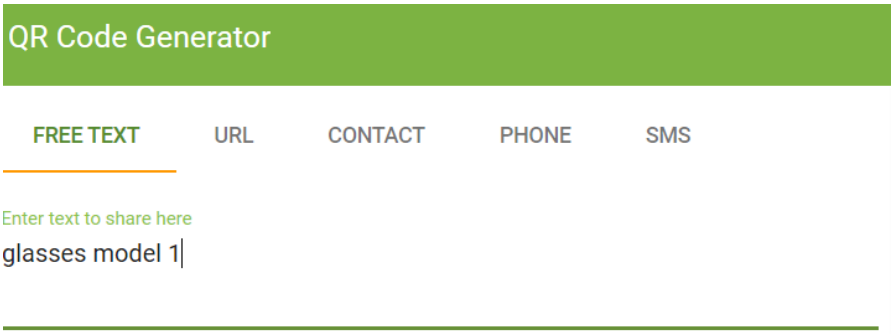


FIGURE 4.3: Selct a category and enter the text

4. Save the QR code on you device.

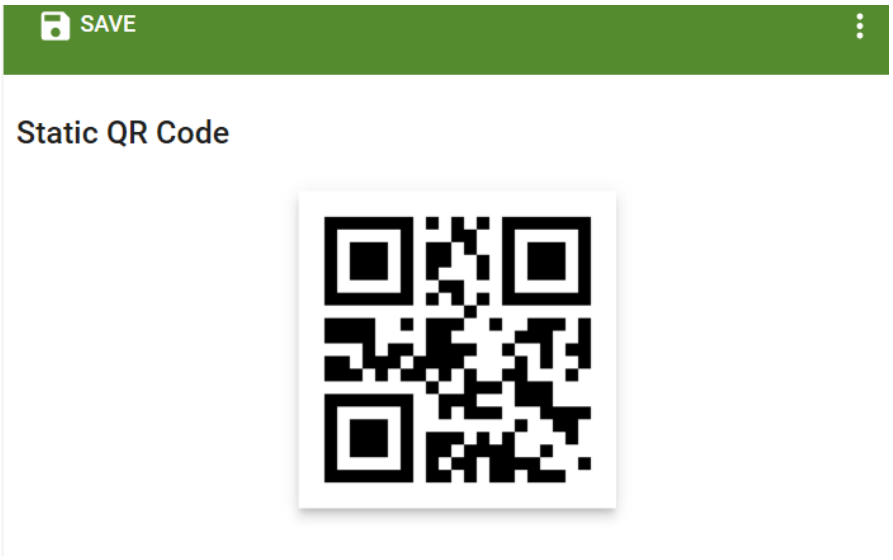


FIGURE 4.4: QR code

4.3 QR code reader

Here are the codes to read a QR code using OpenCV:

```
// load a QR code image.
Texture2D img = Resources.Load ("QR") as Texture2D;
Mat imgMat = new Mat (img.height,
                    img.width, CvType.CV_8UC4);
Utils.texture2DToMat (img, imgMat);
Mat grayMat = new Mat ();
Imgproc.cvtColor (imgMat, grayMat, Imgproc.COLOR_RGBA2GRAY);

// Detect
Mat points = new Mat ();
Mat straight_qrcode = new Mat ();
QRCodeDetector detector = new QRCodeDetector ();
bool result = detector.detect (grayMat, points);

//decode
if (result) {
    string decode_info = detector.decode (grayMat,
                                    points, straight_qrcode);
    Debug.Log (decode_info);
    }
```

The QR code detector result is as follows:

FIGURE 4.5: Decode result

4.4 2D glasses try-on system

In this section, we will talk about the details to complete the 2D try-on system. To fit the glasses image to human face, we need to do image segmentation, contour extraction and filling, head pose estimation and glasses rotation.

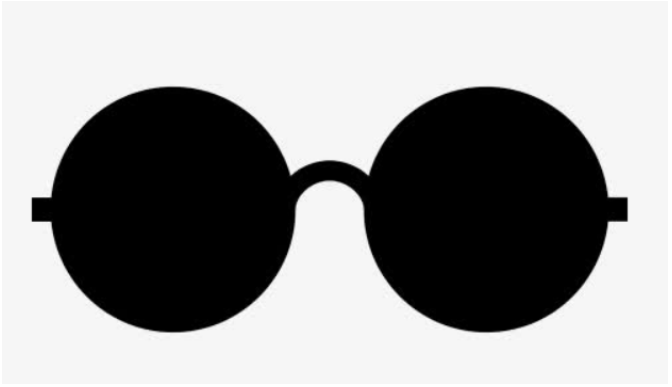
4.4.1 Image segmentation

We use canny detection to segment the glasses image, here are the codes for canny detection:

```
//canny edge detection
Mat grayMat = new Mat();
Mat edge = new Mat();

//rgbaMat is the original image
Imgproc.cvtColor(rgbaMat, grayMat, Imgproc.COLOR_RGBA2GRAY);
OpenCVForUnity.ImgprocModule.Imgproc.blur(grayMat, edge,
                                          new Size(3, 3));
OpenCVForUnity.ImgprocModule.Imgproc.Canny(grayMat, edge,
```

```
                                                    100, 300, 3);
```

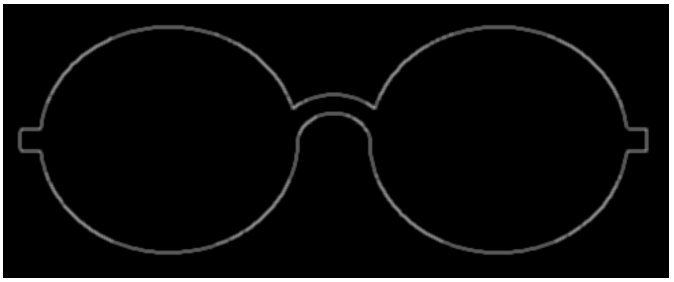
The results for image segmentation are:



(a) Original glasses image 1

(b) Edge detection result 1

(c) Original glasses image 2



(d) Edge detection result 2

FIGURE 4.6: Canny detection results

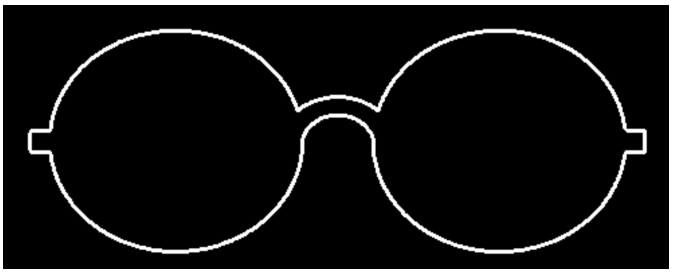
4.4.2 Contour extraction and contour filling

In the image segmentation section, we use Canny edge detection to find the glasses edges. But we haven't extracted the glasses image in the picture. In this section, we use contour extraction method to further process the picture we got from image segmentation section.

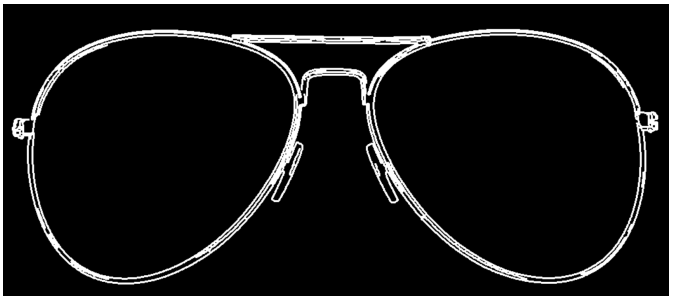
We use findContour and drawContour methods in OpenCV to get the glasses image. First, we can use findContour method to extract the glasses contour. Then, we fill the glasses contour by using drawContour.

The codes for Contour extraction and contour filling are:

```
//Gaussian filtering
Imgproc.GaussianBlur(edge, edge, new Size(3, 3), 3, 3);
Mat img = new Mat();
Imgproc.threshold(edge, img, 0, 255, Imgproc.THRESH_OTSU);

//find contours of the image
List<MatOfPoint> contours = new List<MatOfPoint>();
Mat hierarchy = new Mat();
Imgproc.findContours(img, contours, hierarchy,
                    Imgproc.RETR_EXTERNAL,
                    Imgproc.CHAIN_APPROX_NONE, new Point(0, 0));

//drawcontours
Mat drawImage = Mat.zeros(rgbaMat.size(), CvType.CV_8UC1);
Debug.Log("contoours: " + contours.Count);

//only one contour
double area = Imgproc.contourArea(contours[0]);
Imgproc.drawContours(drawImage, contours, 0,
                    new Scalar(255, 255, 255), -1);
```

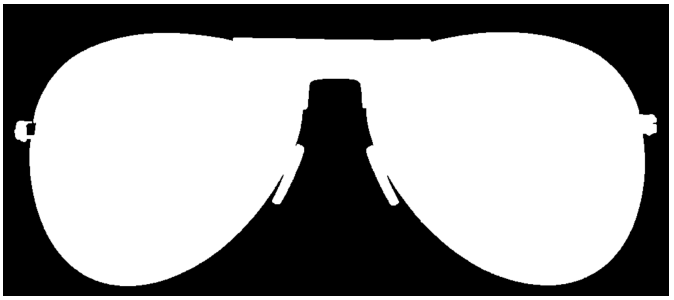


(a) contour result 1



(b) Contour result 2

FIGURE 4.7: Contour extraction results



(a) Contour filling result 1

(b) Contour filling result 2

FIGURE 4.8: Contour filling results

4.4.3 Head pose estimation and glasses rotation

This section will introduce the method to estimate the user's head pose information. There are three issues that need to be considered: face rotation, glasses rotation and glasses image scaling.

4.4.3.1 Face rotation

As people's face rotation may be flexible, we need to figure out face rotation in a 2D coordinate system. The rotation can be expressed by the angle between the line where the two eyes are located and the horizontal direction, which are:

- Left eye coordinates: (x1,y1)
- Right eye coordinates: (x2,y2)

$$Angle = atan(\frac{x2 - x1}{y2 - y1})\frac{180}{\pi} \tag{4.1}$$

4.4.3.2 Glasses image rotation

Glasses rotation refers to rotating the glasses image around a certain point in a 2D coordinate system. As the glasses may be rotated in the original image, we need to figure the original glasses rotation firstly. We have figure out the contour of the glasses in the image in section 3.3.3. In this section, we use the minimum bounding rectangle to calculate the original glasses rotation.

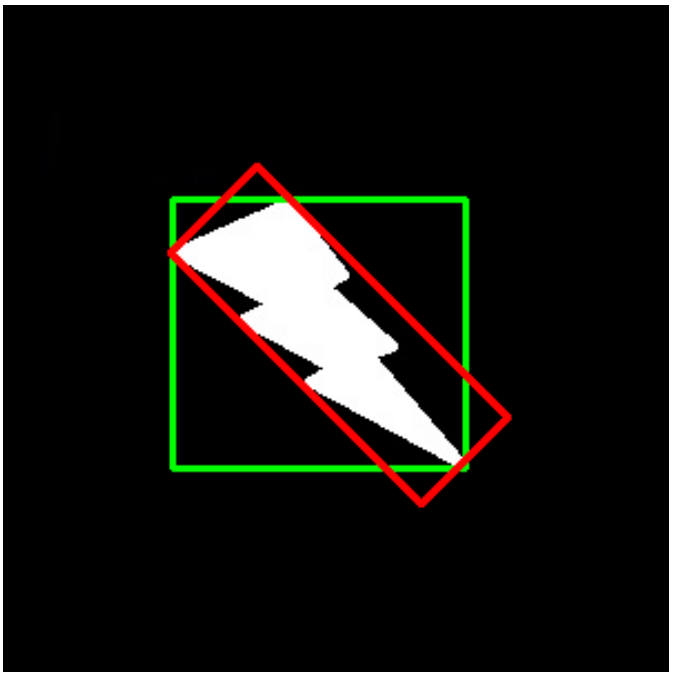


FIGURE 4.9: Bounding Rectangle

In the picture above, the green rectangle is a bounding rectangle of the image. The red rectangle is the minimum bounding rectangle of the image. In this research, we will use the red rectangle to find the glasses rotation in the original glasses image.

Here we use minAreaRect function in OpenCV. From this function, we can get the center point, height, width and angle of the minimum bounding rectangle. Note that, in the minAreaRect function, the width and height of the rectangle are not defined by length. When the x-axis rotates counterclockwise, the first side of the rectangle that the x-axis encounters is the width of the rectangle. The other side is the height of the rectangle. The rotation angle θ is the angle between the x-axis and width of the rectangle. Besides, The rotation angle is obtained by rotating counterclockwise around the x-axis. So the angle θ ranges from -90 degrees to 0. Here we need to figure out the correct rotation angle. We assume that the width of the glasses image should be longer than the height of the glasses image. In the result of the minAreaRect function, if the width is longer than the height, the image is rotated counterclockwise and the rotation angle is right. If the width is shorter than the height, the image is rotated clockwise and the rotation angle = 90 + rotation angle (rotation angle > 0).

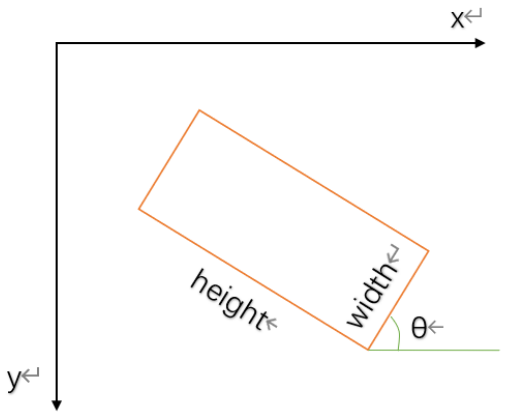


FIGURE 4.10: The rotation angle of minAreaRect function

The codes to find the original glasses rotation angle are as follows:

```
//draw the minmum rectangle
MatOfPoint2f contourPoints = new
                            MatOfPoint2f(contours[0].toArray());
RotatedRect minRect = Imgproc.minAreaRect(contourPoints);
minRectWidth = minRect.size.width;
minRectHeight = minRect.size.height;
//the width of sunglasses > the height of sunglasses
double glassangle=minRect.angle;
if (minRectHeight > minRectWidth)
{
    double a = minRectHeight;
    minRectHeight = minRectWidth;
    minRectWidth = a;
    if (glassangle < 0)
        glassangle = 90 + glassangle;
}
Debug.Log("glassangle: " + glassangle);
glassCenter = minRect.center;
Debug.Log("min width: " + minRectWidth +
            "  height: " + minRectHeight);
```

Then we need to rotate the glasses image to fit the user's face. The simplest example of two-dimensional rotation is the rotation around the coordinate origin, as shown in the following figure:

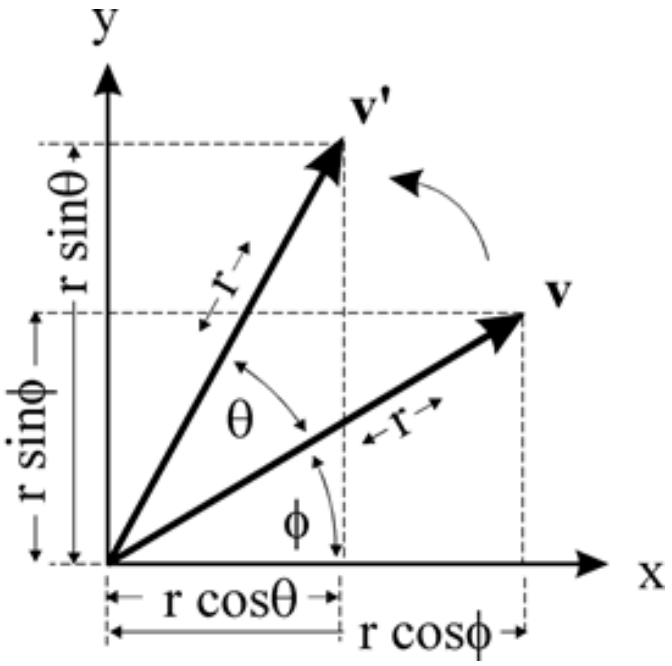


FIGURE 4.11: Image rotation around the coordinate origin

In the picture above, the point v(x, y) is the point before rotation, and v'(x', y') is the point after rotation. The rotation angle is θ. The angle ϕ is the angle between the vector from point v to the origin point and the x axis. r represents the distance between the origin point and point v. The expression of the coordinate of point v can be defined as follows:

$$x = rcos\phi \tag{4.2}$$

$$y = rsin\phi \tag{4.3}$$

The coordinate of point v' is expressed as:

$$x' = rcos(\theta + \phi) \tag{4.4}$$

$$y' = rsin(\theta + \phi) \tag{4.5}$$

Then the relationship between point v and point v' can be represented by the following formula:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.6}$$

When the image is rotated around an arbitrary point, we need to move the origin point from the upper left corner to the rotation center (x_0, y_0) firstly. Then rotate the picture

and the rotation center is the origin point. After the rotation is completed, transform the coordinate origin to the upper left corner of the rotated image.

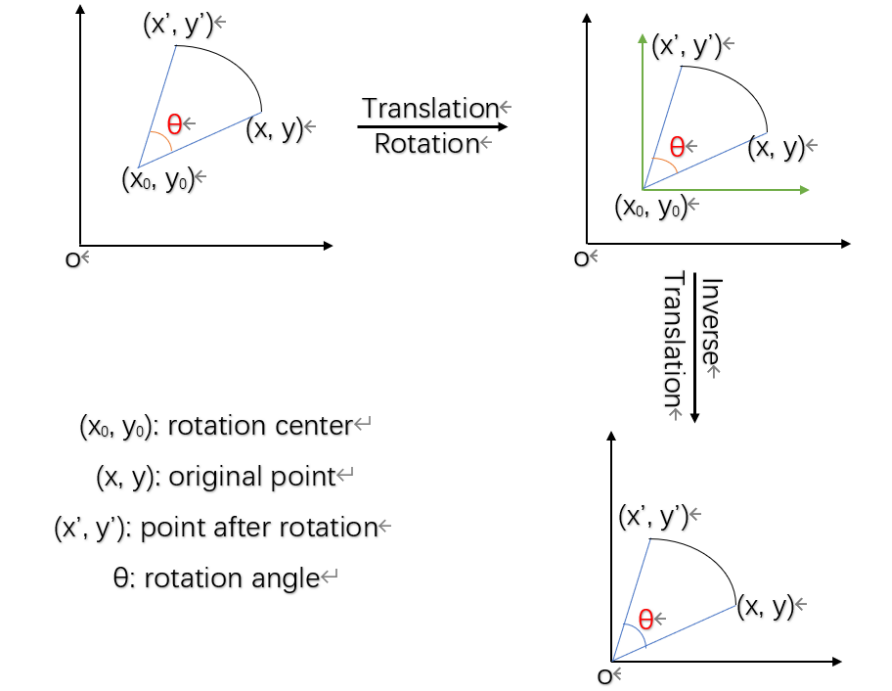


FIGURE 4.12: Image rotation around an arbitrary point

The formula used to rotate point v (x, y) to point v'(x', y') is as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & x_0 - x_0 cos\theta + y_0 sin\theta \\ sin\theta & cos\theta & y_0 - x_0 sin\theta + y_0 cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4.7}$$

We can define the rotation matrix M as:

$$M = \begin{bmatrix} cos\theta & -sin\theta & x_0(1 - cos\theta) + y_0 sin\theta \\ sin\theta & cos\theta & y_0(1 - cos\theta) + x_0 sin\theta \end{bmatrix} \tag{4.8}$$

In OpenCV, we use the getRotationMatrix2D function to find the rotation matrix M and use the warpAffine function to get the rotated image. The codes for image rotation are as follows:

```
//Calculate the image size after rotation
//The picture is rotated around the center of the glasses image
Rect newRect = new RotatedRect(glassCenter,
                new Size(glassesImage.width(),
                glassesImage.height()), -angle).boundingRect();
```

```
Mat affine_matrix = Imgproc.getRotationMatrix2D(glassCenter,
                    -angle, 1.0);

//Adjust rotation center to the center of the rotated image,
//otherwise you will get only part of the result
double a = affine_matrix.get(1, 2)[0] +
            (newRect.width / 2 - glassCenter.x);
double b = affine_matrix.get(0, 2)[0] +
            (newRect.height / 2 - glassCenter.y);
affine_matrix.put(0, 2, b);
affine_matrix.put(1, 2, a);

//Get the rotated image
Imgproc.warpAffine(glassesImage, rotatedImage,
                affine_matrix, newRect.size());
```

4.4.3.3 Image scaling

Image scaling is the last step in the design of the system. After image rotation, we need to resize the glasses image. Here we use the resize function in OpenCV to scale the image. (x, y) is the position of the pixel in the original image. The width and height of the image before rotation are w_1 and h_1. (x', y') is the point after rotation. w_2 and h_2 are the width and height of the rotated image. Then the coordinate transformation formulas in the image are as follows:

$$x' = x\frac{w_2}{w_1} \tag{4.9}$$

$$y' = y\frac{h_2}{h_1} \tag{4.10}$$

4.5 3D glasses try-on system

The implementation of the 3D glasses try-on system is different from the 2D glasses try-on system. We need to convert the user's head location from a 2D world to a 3D world. This section will introduce the methods to calculate the user's 3D head location.

4.5.1 3D head location

When a person's head rotates and moves in the camera, we need to find its new position in the 3D coordinate. This involves the coordinate transformation. If a person's head moves, We can get the pixel coordinates of the 6 key points by Dlib detector and we have defined the world coordinates of these points. But we do not know their camera coordinates. In this section, we need to calculate the world coordinates of the 6 points. As we have talked about the conversion of the points from world space to camera space before. The formula of this conversion is as follows:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.11}$$

In this formula, (X_c, Y_c, Z_c) is the camera coordinate of the key point, and (X_w, Y_w, Z_w) is the world coordinate of the key point. R is the rotation matrix and T is the translation matrix. The codes to convert the world coordinate of a key point to camera coordinate are as follows:

```
//camera position
Mat cameraPosition = new Mat(3, 1, CvType.CV_64FC1);
//rt is rotation and translation matrix
Mat rt = new Mat(3, 4, CvType.CV_64FC1);

List<Mat> ad = new List<Mat>();
//rotM is rotation matrix
ad.Add(rotM);
//tvec is translation matrix
ad.Add(tvec);
Core.hconcat(ad, rt);
//point is the world position of a key point
cameraPosition = rt * point;

//M is the affine transformation
M.SetRow(0, new Vector4((float)rotM.get(0, 0)[0],
                    (float)rotM.get(0, 1)[0],
                    (float)rotM.get(0, 2)[0],
                    (float)cameraPosition.get(0, 0)[0]));
```

```
M.SetRow(1, new Vector4((float)rotM.get(1, 0)[0],
                    (float)rotM.get(1, 1)[0],
                    (float)rotM.get(1, 2)[0],
                    (float)cameraPosition.get(1, 0)[0]));
M.SetRow(2, new Vector4((float)rotM.get(2, 0)[0],
                    (float)rotM.get(2, 1)[0],
                    (float)rotM.get(2, 2)[0],
                    (float)cameraPosition.get(2, 0)[0]));
M.SetRow(3, new Vector4(0, 0, 0, 1));

//set the glasses rotation and position on user's face
//convert right-handed to left-handed system coordinate
ARM = ARcamera.transform.localToWorldMatrix *
        invertYM * transformationM * invertZM;

if (sunglasses != null)
{
    ARUtils.SetTransformFromMatrix(sunglasses.transform, ref ARM);
    sunglasses.SetActive(true);
}
```

5

Results and Discussion

5.1 Initial Results

In the next sections, we will show the results we got both from the 2D and 3D glasses try-on system. Furthermore we will compare these results and discuss their advantages and disadvantages.

5.1.1 Results of 2D glasses try-on system

This section will show the results of the glasses try-on system using glasses images.

5.1.1.1 Results of glasses image segmentation

Glasses image segmentation is the first and essential part for the 2D glasses try-on system. Here shows several different glasses pictures segmentation results.

Figure 5.1: Glasses 1

FIGURE 5.2: Glasses 2

FIGURE 5.3: Glasses 3

FIGURE 5.4: Glasses 4

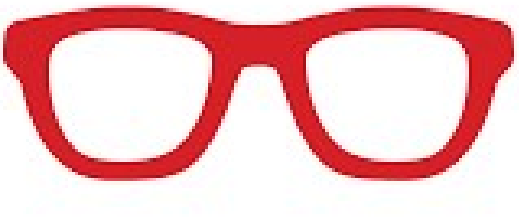


FIGURE 5.5: Glasses 5

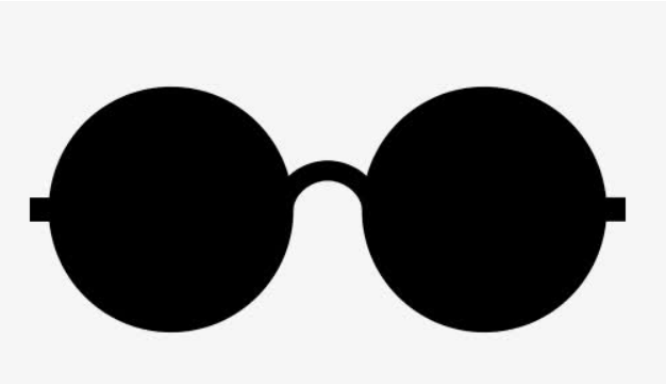


FIGURE 5.6: Glasses 6

The segmentation for these glasses image are as follows:

FIGURE 5.7: Segmentation result 1

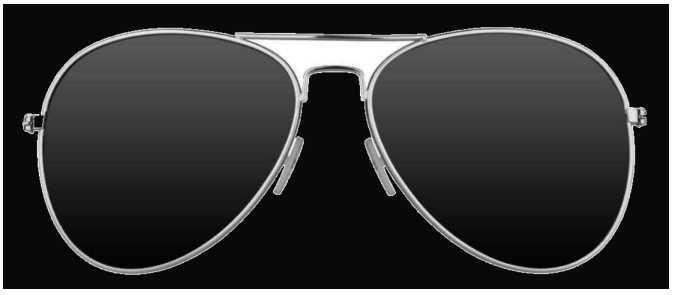


FIGURE 5.8: Segmentation result 2

FIGURE 5.9: Segmentation result 3

FIGURE 5.10: Segmentation result 4

FIGURE 5.11: Segmentation result 5

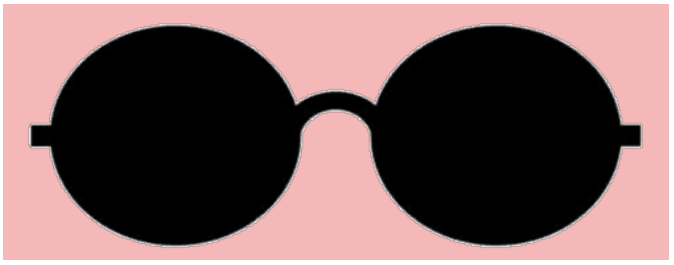


FIGURE 5.12: Segmentation result 6

As we can see from these images, the segmentation results are not perfect. In glasses2 and glasses5, a small part of the background could not be removed. What's more, there are some glasses images that can't be processed by using this segmentation method, such as the following pictures:

(a) Glasses 7

(b) Glasses 8

FIGURE 5.13: Glasses images can't be processed

The segmentation results of those pictures are:

(a) Glasses 7

(b) Glasses 8

FIGURE 5.14: False results

What these pictures have in common is that the color of the glasses is similar to the background color.

5.1.1.2 Glasses try-on results

FIGURE 5.15: 2D glasses try-on result 1

FIGURE 5.16: 2D glasses try-on result 2

FIGURE 5.17: 2D glasses try-on result 3

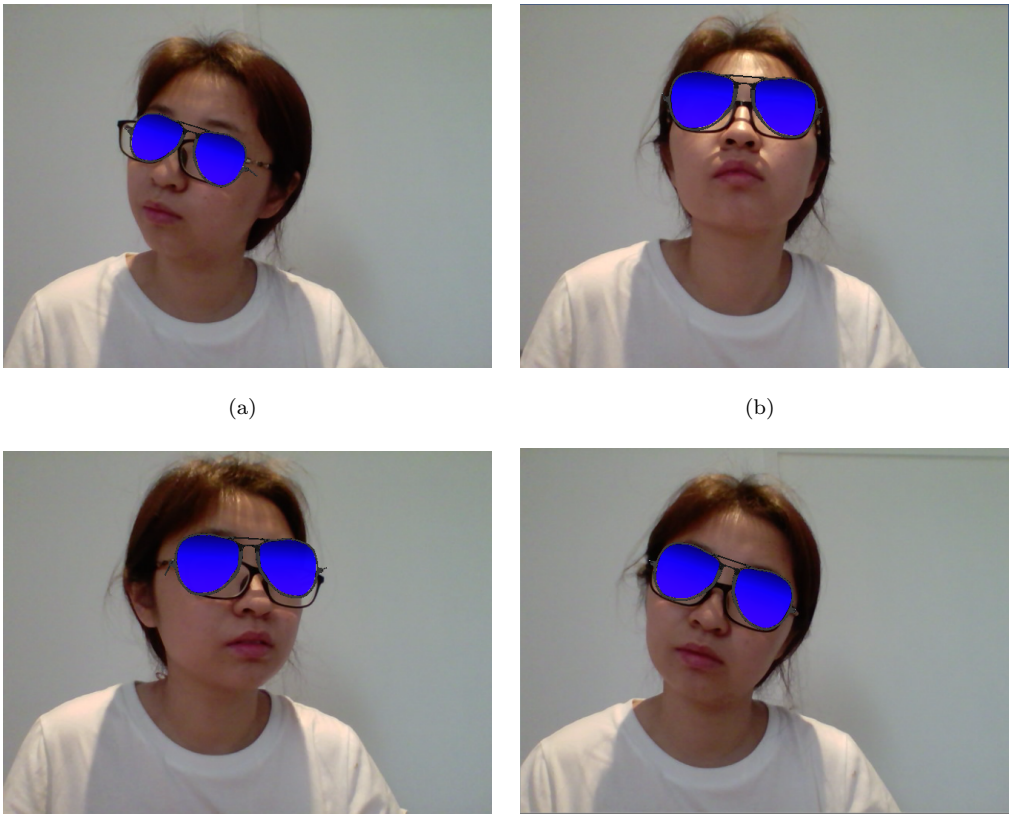
FIGURE 5.18: 2D glasses try-on result 4

FIGURE 5.19: 2D glasses try-on result 5

FIGURE 5.20: 2D glasses try-on result 6

5.1.2 Results of 3D glasses try-on system

1. 3D glasses try-on results using Levenberg-Marquardt optimization

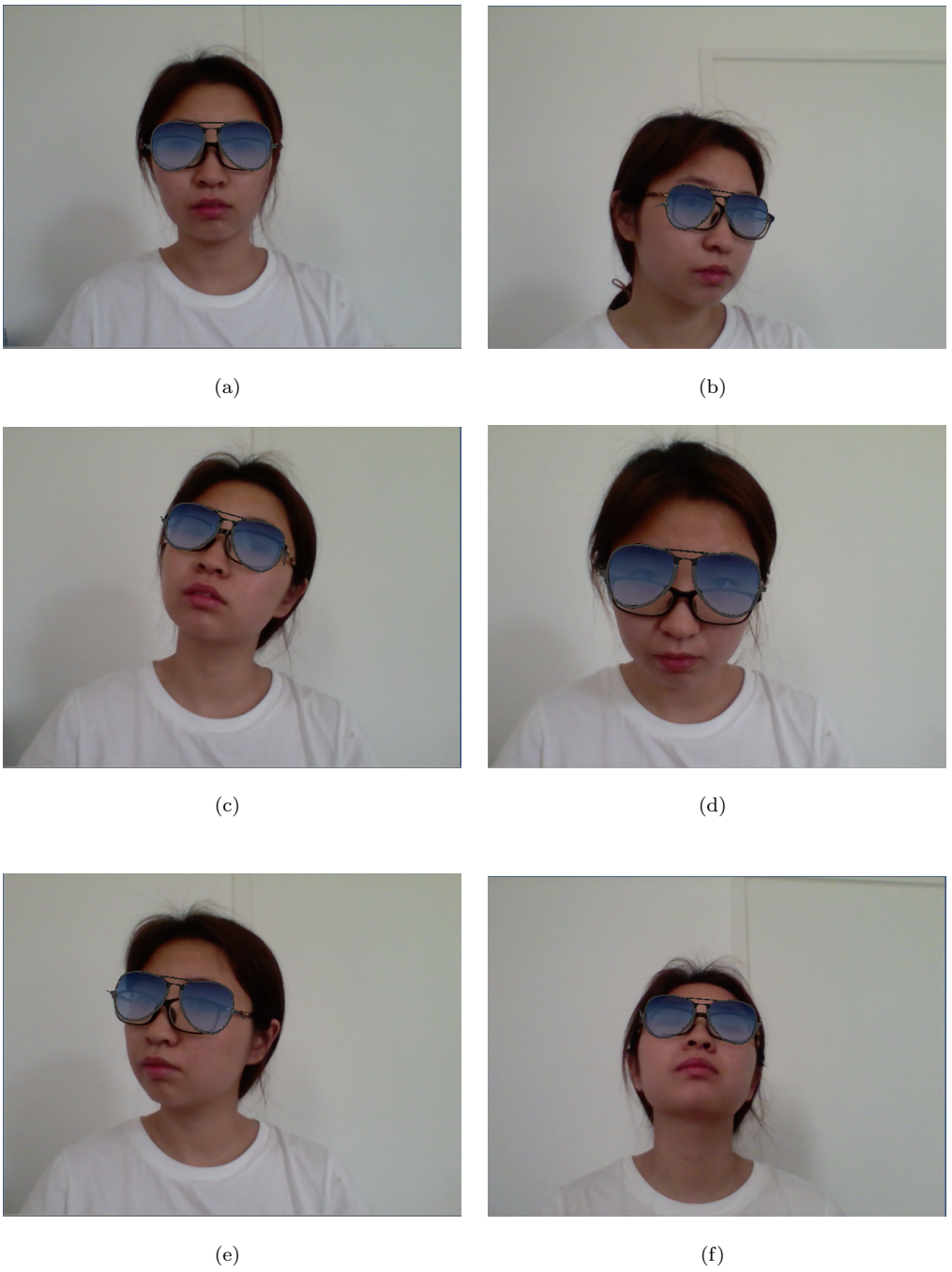


(a) (b)

(c) (d)

FIGURE 5.21: 3d glasses try-on results using Levenberg-Marquardt optimization 1

The transparency of the glasses could also be changed in Unity and that allows users to feel more real.



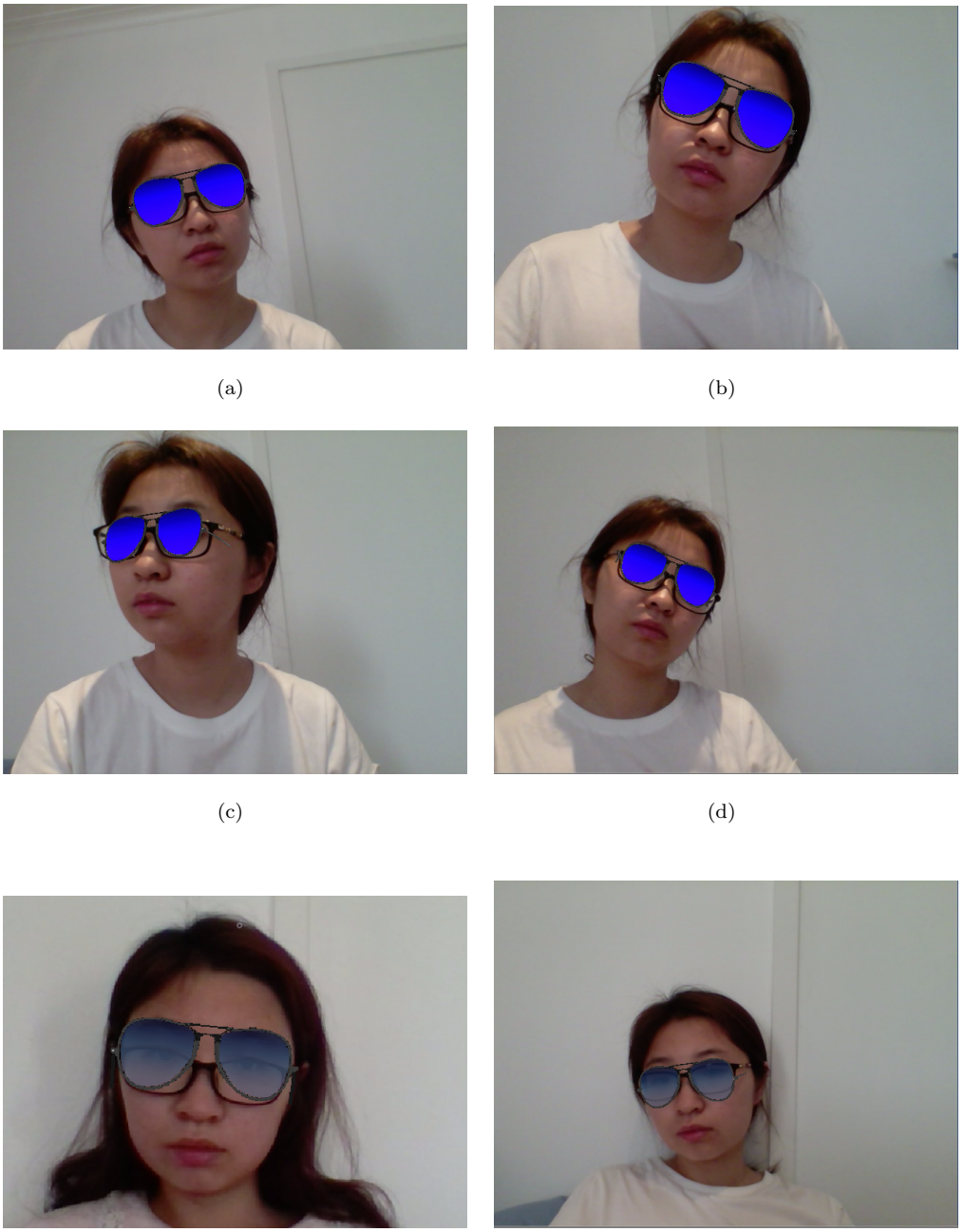
(a) (b)

(c) (d)

(e) (f)

FIGURE 5.22: 3d glasses try-on results using Levenberg-Marquardt optimization 2

2. 3D glasses try-on results using EPnP

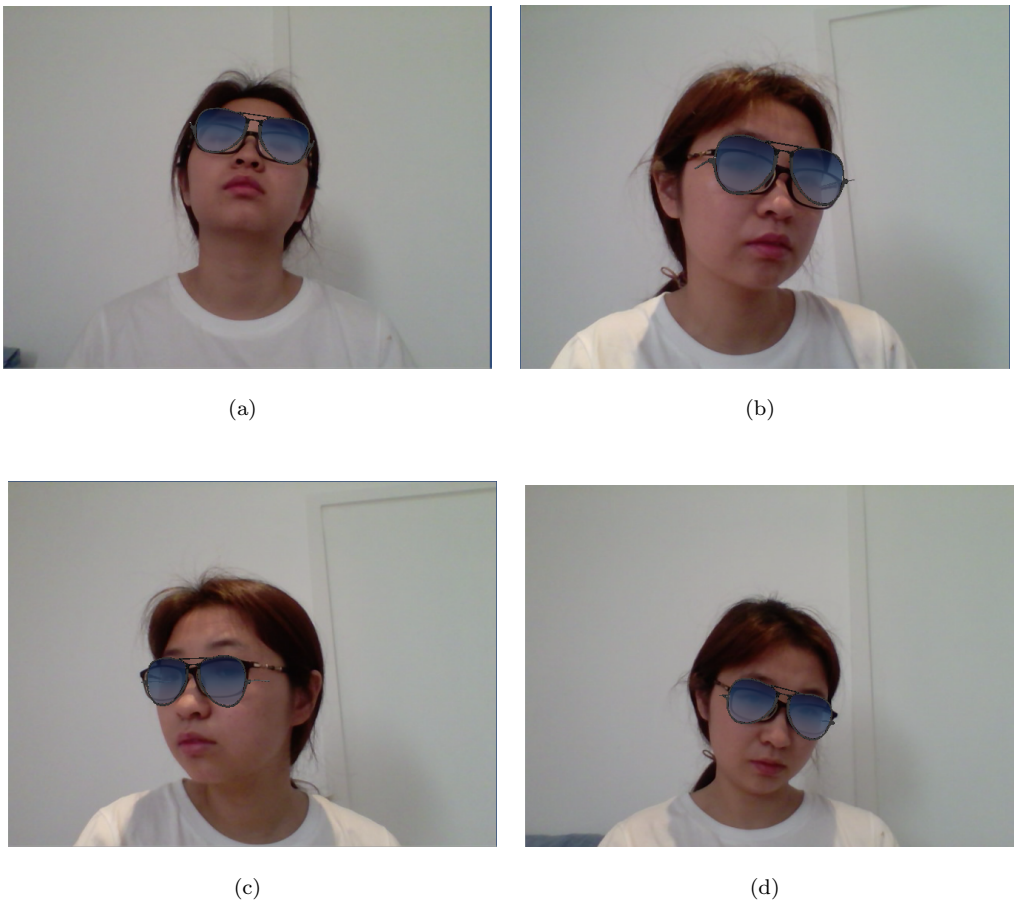


(a) (b)

(c) (d)

(e) (f)

FIGURE 5.23: 3d glasses try-on results using EPnP 1



(a) (b)

(c) (d)

FIGURE 5.24: 3d glasses try-on results using EPnP 2

5.2 Discussion

This thesis uses two methods to complete the glasses try-on system. This system uses the front and rear cameras of the mobile phone. Users use the rear camera to detect a QR code and the system can load the glasses model according to the QR code. Then users can see themselves wearing glasses in the front camera. These QR codes can be shown on magazines. When people see glasses on magazines, they can scan the QR codes and try on these glasses. This is the advantage of using front and rear cameras in this system. Both the two glasses try-on methods have advantages and disadvantages. There are mainly two disadvantages of the 2D glasses try-on system.

1. Selection of the glasses pictures

   Users can download glasses pictures online or take a picture of glasses by themselves, but there are many limits. Because the selection of the glasses images is important, and this will directly affect the glasses try on results. Firstly, the

glasses image must be high resolution. Secondly, the background in the picture and the glasses image should not be too similar. Besides, if users use their own camera to take glasses photos, the camera must be HD camera.

2. Image segmentation

   In this thesis, the image segmentation result for the glasses. In some cases, the segmentation method used in this thesis can't remove all the background in the glasses picture. It is hard to do the segmentation perfectly.

However, the 2D glasses try-on system also has advantages:

1. People don't need to build 3D glasses models for this system.
2. There isn't any database to store the 3D glasses models.

Compared to the 2D glasses try-on system, the 3D glasses try-on system need to build a database to store these 3D glasses models. And users can't try the glasses that are not in the system database. But the advantages of the 3D glasses try-on system are more obvious. Users can see themselves wearing glasses more realistically, and more people will be happy to use 3D glasses try-on system. The results also show that using EPnP method in camera calibration gives better and more accurate results.

5.3 Aspects that need improvement

1. The 2D glasses segmentation result need to improve. As we can see from the glasses segmentation results, there are many problems. In some of the pictures, the glasses were removed. And a part of the background in the pictures couldn't be removed. These problems will give users a bad experience. In order to improve these results, we can use other segmentation methods. Deep learning is famous in solving computer vision problems and it can improve the accuracy of image segmentation effectively. Fully Convolutional Networks (FCN) is an example of the deep learning method for image segmentation and it is said that FCN is the first deep learning method used for image semantic segmentation. Semantic image segmentation enables pixel-level classification. Other image segmentation methods using deep learning include SegNet, Dilated Convolutions, DeepLab, RefineNet, PSPNet and Large Kernel Matters.
2. Users must use the existing 3D models in the system.

In the 3D glasses try-on system, users must use the existing 3D models in the system. They can't use their own 3D glasses models to try on glasses. And the 3D models in the database are also limited.

6

Conclusion and Future Works

6.1 Conclusion

When people shop glasses online, they can't try on the glasses and they don't know if the glasses fit them or not. When people buy sunglasses in a store, they find that it is hard to see themselves wearing sunglasses in the mirror. When people read a fashion magazine, they see the sunglasses and they may want to try on the sunglasses. In order to solve these problems, this thesis proposed a glasses try-on system using AR technique. People can put a QR code of the sunglasses on the magazine. The readers can scan the QR code and try on the sunglasses on their phone. So this thesis develops two glasses try-on systems: 2D and 3D glasses try-on systems.

In the first system, users can scan a glasses picture or download a picture online, then use the system to try on the glasses. The system also has functions of glasses detection and glasses image segmentation. In order to get a good result, the glasses picture must be in high resolution. What is more, the color of the glasses image and the background color must be different. It is hard for the system to do the segmentation if their colors are too similar.

In the second system, users can scan a QR code, can then they will see themselves wearing glasses in the camera. But the glasses model must be stored in the system. People can't use or upload their own 3D glasses models. This system has an advantage is that people can see themselves in 3D space, which will make users feel more real and give users a good try on experience.

6.2 Future Work

There is two main work to do in the future: improve image segmentation results and improve the functions of the 3D glasses try-on system.

As we talked in section 4, we need to improve the image segmentation method for the glasses picture. Nowadays, there are so many advanced learning algorithms that already applied in image segmentation and they work very well. Some articles have shown that CNN has a good performance on image segmentation [72][73][74]. In the future, the research can focus on how to improve the image segmentation result by using deep learning.

For the 3D glasses try-on system, whether it can be combined with web. This means that we store 3D glasses models online. Users can upload new 3D models and use them in the try-on system.

By the way, Augmented Reality Technology is not only suitable for online shopping for glasses. My project is just a simple AR application for glasses try on. For the future work, there are so many things we can do, such as we can make a try-on system for the clothes, accessories and cosmetic. Girls can immediately see what they look like when they get dressed. For that case, we can design an application for the online shopping platform, it can make more people apply our products so that they can buy what they want.

Bibliography

[1] Rick Van Krevelen. Augmented reality: Technologies, applications, and limitations. 04 2007.

[2] Steven Feiner, Blair Macintyre, Tobias Höllerer, and Anthony Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. volume 1, pages 74–81, 12 1997. doi: 10.1007/BF01682023.

[3] Marios Bikos, Yuta Itoh, Gudrun Klinker, and Konstantinos Moustakas. An interactive augmented reality chess game using bare-hand pinch gestures. 2015. doi: 10.1109/cw.2015.15.

[4] Julio Cristian Young, Marcel Bonar Kristanda, and Seng Hansun. Armatika: 3d game for arithmetic learning with augmented reality technology. 2016. doi: 10.1109/iac.2016.7905744.

[5] Fabricio Pretto, Isabel Harb Manssour, Maria H. Itaqui Lopes, and Marcio S. Pinho. Experiences using augmented reality environment for training and evaluating medical students. 2013. doi: 10.1109/icmew.2013.6618311.

[6] Jeffrey Goderie, Rustam Alashrafov, Pieter Jockin, Lu Liu, Xin Liu, Marina A. Cidota, and Stephan G. Lukosch. [poster] chirochroma: An augmented reality game for the assessment of hand motor functionality. 2017.

[7] Yu-I Yang, Chih-Kai Yang, and Chih-Hsing Chu. A virtual try-on system in augmented reality using rgb-d cameras for footwear personalization. *Journal of Manufacturing Systems*, 33(4):690–698, 2014.

[8] Pei-Hsuan Chiu, Po-Hsuan Tseng, and Kai-Ten Feng. Interactive mobile augmented reality system for image and hand motion tracking. *IEEE Transactions on Vehicular Technology*, 67:9995–10009, 2018.

[9] Joao Monge and Octavian Postolache. Augmented reality and smart sensors for physical rehabilitation. 2018. doi: 10.1109/icepe.2018.8559935.

[10] Etonam Amelessodji, Giulio Di Gravio, Patrick Kuloba, and Jackson G. Njiri. Augmented reality (ar) application in manufacturing encompassing quality control and maintenance. *International Journal of Engineering and Advanced Technology Regular Issue*, 9(1):197–204, 2019. doi: 10.35940/ijeat.a1120.109119.

[11] Inc Prism Visual Software. Mobile devices. URL http://www.prismvs.com/handheld-devices.php.

[12] Dp. URL https://www.amazon.com/dp/B01MQWK336?tag=picclick0f-20&linkCode=osi&th=1&psc=1.

[13] Jual promo ready glyph avegant..personal theather. URL https://www.tokopedia.com/avegant/promo-ready-glyph-avegant-personal-theather.

[14] Inspired vr headsets, Nov 2016. URL http://charlesayats.fr/best-vr-headset/.

[15] Sunitha M. R, Fathima Khan, Gowtham Ghatge R, and Hemaya S. Object detection and human identification using raspberry pi. *2019 1st International Conference on Advances in Information Technology (ICAIT)*, 2019. doi: 10.1109/icait47043.2019.8987398.

[16] Si Liu, Luoqi Liu, and Shuicheng Yan. Magic mirror: An intelligent fashion recommendation system. 2013. doi: 10.1109/acpr.2013.212.

[17] Xbox one kinect sensor, . URL https://www.gamesmen.com.au/xbox-one-kinect-sensor.

[18] Andrea Vitali and Caterina Rizzi. Acquisition of customer's tailor measurements for 3d clothing design using virtual reality devices. *Virtual and Physical Prototyping*, 13(3):131–145, 2018. doi: 10.1080/17452759.2018.1474082.

[19] Canny edge detection. URL https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html.

[20] Florentin Alexandru Dita. A foreign language learning application using mobile augmented reality. *Informatica Economica*, 20(4/2016):76–87, 2016.

[21] Samrat Nath. Understanding the rise of augmented reality–based apps post-pokémon go. *Interactions: Studies in Communication Culture*, 9:319–334, 11 2018. doi: 10.1386/iscc.9.3.319_1.

[22] Boping Zhang. Design of mobile augmented reality game based on image recognition. *Eurasip Journal on Image Video Processing*, 2017(1):90, 2017.

[23] Giovanni Piumatti, Andrea Sanna, Marco Gaspardone, and Fabrizio Lamberti. Spatial augmented reality meets robots: Human-machine interaction in cloud-based projected gaming environments. 2017. doi: 10.1109/icce.2017.7889276.

[24] Nahal Norouzi, Kangsoo Kim, Myungho Lee, Ryan Schubert, Austin Erickson, Jeremy Bailenson, Gerd Bruder, and Greg Welch. Walking your virtual dog: Analysis of awareness and proxemics with simulated support animals in augmented reality. *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2019. doi: 10.1109/ismar.2019.000-8.

[25] Nahal Norouzi, Gerd Bruder, Jeremy Bailenson, and Greg Welch. Investigating augmented reality animals as companions. *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2019.

[26] Bimo Sunarfri Hantono, Lukito Edi Nugroho, and P. Insap Santosa. Meta-review of augmented reality in education. 2018. doi: 10.1109/iciteed.2018.8534888.

[27] Muhammad Zahid Iqbal, Eleni Mangina, and Abraham G. Campbell. Exploring the use of augmented reality in a kinesthetic learning application integrated with an intelligent virtual embodied agent. *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2019.

[28] Jen-Yang Chen, Chuan-Hsi Liu, Chaur-Heh Hsieh, Shih-Yu Huang, Wen-Kai Wang, and Bo-Hong Nien. Kinect augmented reality gear game design. 2017. doi: 10.1109/icasi.2017.7988429.

[29] Rui Cao and Yue Liu. Hand controlar: An augmented reality application for learning 3d geometry. *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2019.

[30] Adson Marques Da Silva Esteves, Andre Luiz Maciel Santana, and Rodrigo Lyra. Use of augmented reality for computational thinking stimulation through virtual. *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, 2019. doi: 10.1109/svr.2019.00031.

[31] Ryosuke Umeda, Mohamed Atef Seif, Hiroki Higa, and Yukio Kuniyoshi. A medical training system using augmented reality. 2017. doi: 10.1109/iciibms.2017.8279706.

[32] Nurul Shuhadah Rosni, Zahidah Abd Kadir, Megat Norulazmi Megat Mohamed Noor, Zaidatul Husna Abdul Rahman, and Nurulain Abu Bakar. Development of mobile markerless augmented reality for cardiovascular system in anatomy and physiology courses in physiotherapy education. *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020. doi: 10.1109/imcom48794.2020.9001692.

[33] Mark Yi-Cheon Yim, Shu-Chuan Chu, and Paul L. Sauer. Is augmented reality technology an effective tool for e-commerce? an interactivity and vividness perspective. *Journal of Interactive Marketing*, 39:89–103, 2017.

[34] Mustafa Atalar and Mahmut Ozcan. New augmented reality application in e-commerce and m-commerce. 2017. doi: 10.1109/ubmk.2017.8093403.

[35] Ihsan Rabbi and Sehat Ullah. A survey on augmented reality challenges and tracking. *Acta Graphica Journal for Printing Science Graphic Communications*, 24 (1-2):29–46, 2013.

[36] Piotr Siekański, Jakub Michoński, Eryk Bunsch, and Robert Sitnik. Catcha: Real-time camera tracking method for augmented reality applications in cultural heritage interiors. *ISPRS International Journal of Geo-Information*, 7(12), 2018. ISSN 2220-9964. doi: 10.3390/ijgi7120479. URL http://www.mdpi.com/2220-9964/7/12/479.

[37] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: A hands-on survey. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2633–2651, Jan 2016. doi: 10.1109/tvcg.2015.2513408.

[38] Simon Prince, Ke Xu, and Adrian David Cheok. Augmented reality camera tracking with homographies. *IEEE Computer Graphics and Applications*, 22:39–45, 2002.

[39] Dragoş Datcu, Stephan Lukosch, and Frances Brazier. On the usability and effectiveness of different interaction types in augmented reality. *International Journal of Humanâcomputer Interaction*, 31(3):193–209, 2015.

[40] Anacleto Correia and Victor Conceica. Survey on augmented reality technologies for naval training. 2019. doi: 10.23919/cisti.2019.8760962.

[41] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. A K Peters, 2005.

[42] Woodrow Barfield. *Fundamentals of wearable computers and augmented reality*. CRC Press, Taylor Francis Group, 2016.

[43] Sanni Siltanen. *Theory and applications of marker-based augmented reality*. VTT, 2012.

[44] International communication association. In *Emerging Mobile Methods: Understanding Augmented Reality Technologies as a Methodological Intervention, Stimulus, and Object of Study*, pages 1–36, 2017.

[45] The connected consumer and the changing face of commerce. URL https://www.walkersands.com/resources/the-future-of-retail-2017/.

[46] Zheng Shou, Binqiang Yu, Gang Chen, Hengjin Cai, and Qiaochu Liu. Key designs in implementing online 3d virtual garment try-on system. 2013. doi: 10.1109/iscid.2013.46.

[47] The premier 3d rendering animation software. URL https://www.posersoftware.com/.

[48] Congfeng Jiang and Yinghui Zhao. Govfir: Grid computing based online virtual fitting room. *2008 International Conference on Computer Science and Software Engineering*, 2008. doi: 10.1109/csse.2008.864.

[49] Dongjoe Shin and Yu Chen. Deep garment image matting for a virtual try-on system. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019. doi: 10.1109/iccvw.2019.00384.

[50] Mert Kaya and Devrim Ünay. Dressboard: An embedded virtual try-on system for ties and bowties. *Journal of Signal Processing Systems*, 73(2):143–152, Oct 2013. doi: 10.1007/s11265-013-0738-2.

[51] Marcus Frings. The golden section in architectural theory. *Nexus Network Journal*, 4(1):9–32, 2002.

[52] Pedro Azevedo, Thiago Oliveira Dos Santos, and Edilson De Aguiar. An augmented reality virtual glasses try-on system. 2016. doi: 10.1109/svr.2016.12.

[53] Zheng Jun, Hua Jizhao, Tang Zhenglan, and Wang Feng. Face detection based on lbp. *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*, 2017. doi: 10.1109/icemi.2017.8265841.

[54] Boping Zhang. Augmented reality virtual glasses try-on technology based on ios platform. *EURASIP Journal on Image and Video Processing*, 2018(1), 2018.

[55] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH 99*, 1999.

[56] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014. doi: 10.1109/tvcg.2013.249.

[57] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, page 557–574, 2018. doi: 10.1007/978-3-030-01264-9_33.

[58] Davide Marelli, Simone Bianco, and Gianluigi Ciocca. A web application for glasses virtual try-on in 3d space. *2019 IEEE 23rd International Symposium on Consumer Technologies (ISCT)*, 2019.

[59] Khalil M. Ahmad Yousef, Bassam J. Mohd, and Malak Al-Omari. Kinect-based virtual try-on system: A case study. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019. doi: 10.1109/jeeit.2019.8717498.

[60] Miaolong Yuan, Ishtiaq Rasool Khan, Farzam Farbiz, Susu Yao, Arthur Niswar, and Min-Hui Foo. A mixed reality virtual clothes try-on system. *IEEE Transactions on Multimedia*, 15(8):1958–1968, 2013. doi: 10.1109/tmm.2013.2280560.

[61] Ammar Mohammed Ali and Alaa Kadhim Farhan. Enhancement of qr code capacity by encrypted lossless compression technology for verification of secure e-document. *IEEE Access*, 8:27448–27458, 2020. doi: 10.1109/access.2020.2971779.

[62] Weihang Zhang, Xue Wang, Wei You, Junfeng Chen, Peng Dai, and Pengbo Zhang. Resls: Region and edge synergetic level set framework for image segmentation. *IEEE Transactions on Image Processing*, 29:57–71, 2020. doi: 10.1109/tip.2019.2928134.

[63] A. M. Khan and Ravi S. Image segmentation methods: A comparative study.

[64] Yong Woon Kim, Ah Reum Oh, Innila Rose J, and Addapalli V N Krishna. Analyzing the performance of canny edge detection on interpolated images. 2019. doi: 10.1109/ictc46691.2019.8939595.

[65] Satya Mallick. Geometry of image formation, Feb 2020. URL https://www.learnopencv.com/geometry-of-image-formation/.

[66] Satya Mallick. Head pose estimation using opencv and dlib, Sep 2016. URL https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/.

[67] Sadekar Kaustubh and Mallick Satya. Camera calibration using opencv, Feb 2020. URL https://www.learnopencv.com/camera-calibration-using-opencv/.

[68] Xiao Lu. A review of solutions for perspective-n-point problem in camera pose estimation. *Journal of Physics: Conference Series*, 1087:052009, 09 2018. doi: 10.1088/1742-6596/1087/5/052009.

[69] Tian Shao-Xiong, Lu Shan, and Liu Zong-Ming. Levenberg-marquardt algorithm based nonlinear optimization of camera calibration for relative measurement. *2015 34th Chinese Control Conference (CCC)*, 2015. doi: 10.1109/chicc.2015.7260394.

[70] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81 (2):155–166, 2008.

[71] . URL http://qr.biz/articles/the_structure_of_qr_code/.

[72] Hu Tao, Weihua Li, Xianxiang Qin, and Dan Jia. Image semantic segmentation based on convolutional neural network and conditional random field. 2018. doi: 10.1109/icaci.2018.8377522.

[73] Wenxiu Wang, Yutian Fu, Feng Dong, and Feng Li. Semantic segmentation of remote sensing ship image via a convolutional neural networks model. *IET Image Processing*, 13(6):1016–1022, Oct 2019. doi: 10.1049/iet-ipr.2018.5914.

[74] Loretta Ichim and Dan Popescu. Road detection and segmentation from aerial images using a cnn based system. 2018. doi: 10.1109/tsp.2018.8441366.