Virtual Conference with Augmented Technology

Weizhe Wang and Mingxiao Zhao

Advisor:
Professor Xinming Huang

A Major Qualifying Project
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for
the Degree of Bachelor of Science in
Electrical and Computer Engineering

Abstract

This report introduces a software-based system that aims to improve the overall quality of the user experience (QoE) on existing online conference meeting platforms, such as Zoom or Teams. The objective of the system is to explore methods to increase immersion and comfort in virtual meetings while mitigating the negative effects of prolonged use, commonly referred to as "Zoom fatigue." The proposed solution consists of five functional modules: Multiple Target Windows Merging and Alignment, Background Removal, Vision Angle Tilting by Head Orientation, Auto Display Resizing, and Boundary Warning. Importantly, the system does not require additional hardware or virtual reality tools, as a standard built-in or USB webcam is sufficient. The software aims to balance the cost and convenience for users while enhancing their overall experience. This report provides technical details of the proposed approach, which offers a cost-effective, user-friendly, and accessible solution for online video conferencing and meetings. Additionally, a software interface is designed for users to evaluate this new online meeting experience and provide feedback.

Acknowledgements

We would like to express our deepest gratitude to:

WPI for this opportunity

Professor Xinming Huang for valuable guidance and support

WPI Funds for financial support

William Appleyard for supplies

Friends and peer students for fantastic ideas and suggestions

Coffee and Tea for energy

Table of Contents

List of Figures

Listings

1 Introduction

1.1 Online Meetings

The emergence of the Covid-19 pandemic in late 2019 prompted a significant shift in the working and learning models of organizations and schools worldwide. As a result, virtual meetings have become an essential tool to maintain societal stability. This is evident from the exponential increase in the number of participants on online meeting platforms such as Zoom, which reported over 300 million daily meeting participants by April 2020 [4]. Tencent Meeting, one of China's biggest online meeting platforms, reported over 200 million active users and held 4 billion meetings in 2020 [5]. The widespread adoption of virtual meetings has made theses platforms essential for many people, with other online meetings applications such as Google Meet and Microsoft Teams also experiencing a significant surge in active users. Notably, the continued use of virtual meetings is projected to remain high until 2024, even after the pandemic [6]. Given the growing importance of virtual meetings, it is crucial to explore and improve the user experience to ensure that the potential benefits of this mode of communication are fully realized.

1.2 Shortcomings and Limitations in Online Meetings

Online meetings have become indispensable tools for people to communicate, share files, and make presentations from remote locations. However, the increased reliance on virtual meetings has presented challenges for many users, some of whom report feeling mentally and/or physically exhausted from long hours of video conferencing. This phenomenon, commonly referred to as "Zoom Fatigue," can be attributed to several factors, including prolonged periods of direct eye gaze. Unlike in-person group meetings where the direct mutual gaze is limited, video conferences require all participants to stare at the users on the screen, leading to increased attentional demands and energy depletion [7]. Moreover, the varying sizes of framed heads and image jumping can be disconcerting, as participants struggle to focus on multiple faces in small boxes that shift positions during discussion. Early research suggests that larger head images activate the sympathetic nervous system associated with the fight-or-flight response, adding to the cognitive load experienced by online meeting participants [8]. Additionally, from a User Interaction perspective, many users find online meetings to be user-friendly but not immersive or engaging.

1.3 Proposed Solutions

We designed 5 main functions that should ease the adverse effects mentioned in Section 1.2 with the intention of creating a better experience for online meeting users.

- The **first** function is Multiple Target Windows Merging and Alignment. Windows Merging and Alignment (WMA) is designed to merge and align all visual streams of other participants into a single virtual meeting space with the aim of providing users with a more immersive and engaging experience. Displaying all participants in a single virtual meeting space has many advantages. For example, this approach enables all participants to be visually co-present in a shared digital space, replicating a face-

to-face meeting environment. The integration of WMA also provides users with a more natural and psychologically closer experience that is intended to lessen the cognitive burden on users. By rendering a unified visual interface with less clutter and reduced image jumping, this feature optimizes the user experience and creates a more efficient and engaging virtual environment.

- The **second** function of our proposed system is Background Removal. This function allows users to customize their virtual background, something which has been implemented in many existing online meeting platforms. However, we aim to introduce this function in a different way. Our system will provide a proper virtual background to the merged window, giving the impression that all participants are sitting in the same meeting room. Users can choose a background that is more natural, such as a conference room from their organization. Alternatively, a blurry background is set as the default option. By providing a realistic and natural setting, the function of Background Removal aims to increase the immersive experience of online meetings, allowing users to better focus on the content and reducing the cognitive load.

- The **third** function is Vision Angle Tilting by Head Orientation (VTHO). Given the aforementioned limitation, presenting all participants' faces simultaneously in a single window is not a feasible approach when the number of participants exceeds four, as it will result in reduced face sizes. Consequently, only two to three participants can be exhibited at a time. The user can maneuver between participants by slightly turning their head towards the left or right. To simulate this functionality, directional sounds can be implemented to create the impression of sounds emanating from either side. Following the sound cues, the user can naturally turn their head in the corresponding direction to observe the participant. To ensure optimal user experience, it is advisable to maintain proportional alignment between the head tilting angles and the display adjustments, instead of perfect equivalence.

- The **fourth** function is Auto Display Resizing (ADR). As explicated in Section 1.2, the presentation of images with different sizes may induce disorientation and additional cognitive exertion for the users, leading to fatigue. The ADR feature is designed to address this issue by dynamically resizing the images of individual participants on the screen, ensuring that all individuals are presented at a comparable size.

- The **fifth** function is Boundary Warning. While this feature may not directly influence user experience, it is instrumental to the processing of the user's image from the perspective of other participants. The Boundary Warning feature functions by generating an alert signal when the user's head movements cause them to deviate or move outside the general field of view. The signal is represented by the appearance of red lines at the edge(s) of the display window, thereby enabling the user to discern the direction of their deviation. The purpose of this function is to promote and ensure an enhanced user experience for all meeting participants.

2 Background

This chapter provides background information on the working principle of online meetings, methods of head & body tracking, and methods of background removal. These concepts are referenced and built upon in our designs.

2.1 Current Online Meetings

Online meetings, virtual meetings, and video conferences are just different names of the same thing that are used for people to communicate from different locations. The basic working principle of online meetings is simple. From each user, the camera captures the image (real-time video is just a sequence of images) and sends this information to other ends. In this way, each user end will have images of all the meeting participants, and each image will be displayed in a small box. The user will see all the boxes arranged together including the one showing him/herself. The final display output would look like a grid frame.

Figure 1: The Display Interface of Zoom Conference

Other common functions include file upload and download, screen share, auto-displaying box switch, and so on. Usually, in a lecture or information session, people will see one big box display the one who is currently speaking and many small boxes display other participants. The image shown in the big box will switch as different people speak. In a group discussion or regular meeting, people prefer to use the grid diagram mode as shown in Figure 1.

Some online meeting platforms provide services of background blur, virtual background, special effects, and cartoon avatars. These functions conduct image processing and generate new output images to display. To succeed in these functions, it will require more advanced technologies including object recognition and head/body tracking.

2.2 Head/Face Tracking

Head (face) tracking is a widely recognized and valuable technique with numerous applications. Its utility is evident in various domains, including identity verification, people enumeration, video conferencing, and specialized simulations such as the Microsoft Flight Simulator. Several distinct methodologies of head tracking exist, each tailored to optimize the performance of the critical features associated with their respective applications.

2.2.1 Infrared Projection & Capture

Facial recognition represents the predominant means by which individuals engage with face tracking in their everyday lives. This technology finds widespread application in smartphones for functions such as unlocking screens, effecting payment transactions, and populating passwords automatically. Head tracking for these purposes centers on the acquisition and analysis of detailed information relating to facial characteristics. **Infrared dot projection** and analysis typically exhibit strong performance in this regard, exemplified by Apple Inc.'s Face ID. The hardware of the Face ID system consists of two main modules: a dot projector to project a grid of infrared dots onto users' faces, and an infrared camera to read the pattern [9].

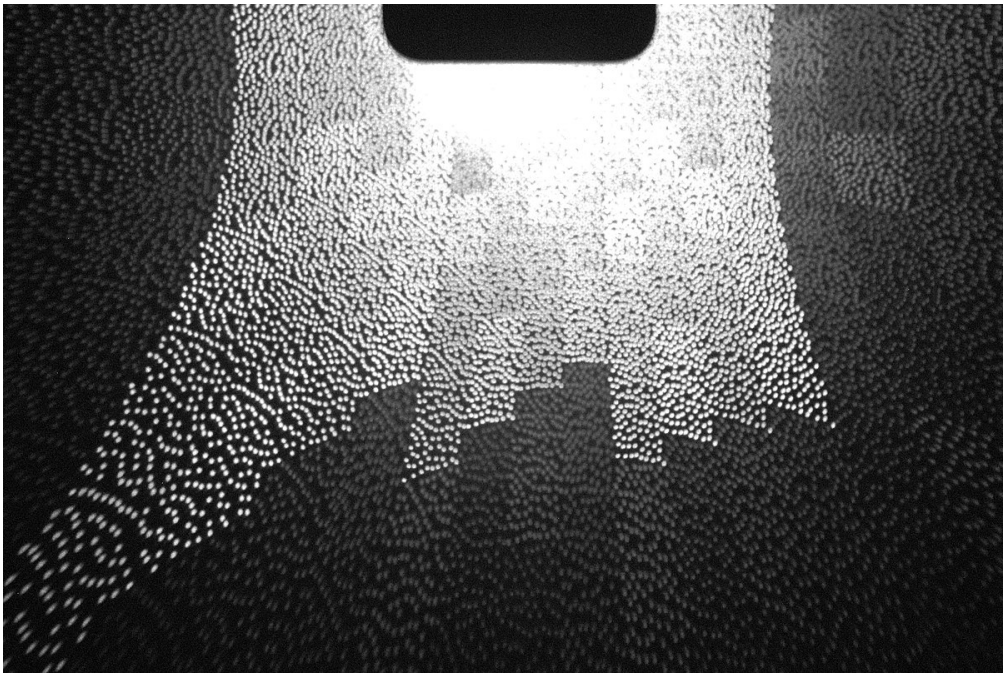

Figure 2: Face ID Infrared Dot Projector

Following the projection phase, the camera will capture an infrared image to generate a 3D facial map, which is then compared with the user's registered model for authentication purposes. Successful completion of the comparison process results in user authentication. The grid used in the projection process incorporates over 30,000 infrared dots, thereby ensuring the requisite level of accuracy for Face ID authentication, even for users wearing glasses. According to Apple Inc., the accuracy of Face ID is 1,000,000:1 [10]. Similar

technologies are used for many other purposes such as attendance records and gate/door access for buildings.

2.2.2 Shape Matching

Head tracking can be applied to people counting. The people-counting system is usually used for video surveillance, statistical analysis of people accessing an area, and security tasks. It performs count distinction between the input and output numbers of people moving through the region of interest (ROI). The head tracking in the people counting system does not detect heads based on facial features. It sets a camera at a high-altitude position and captures the image information from the top view. The detection is based on finding people's heads through pre-processed image correlation with circular patterns while tracking is based on a Kalman filter to determine the trajectory of the head candidates [11]. The updated people counters in the target ROI are then calculated based on the direction of the trajectories.

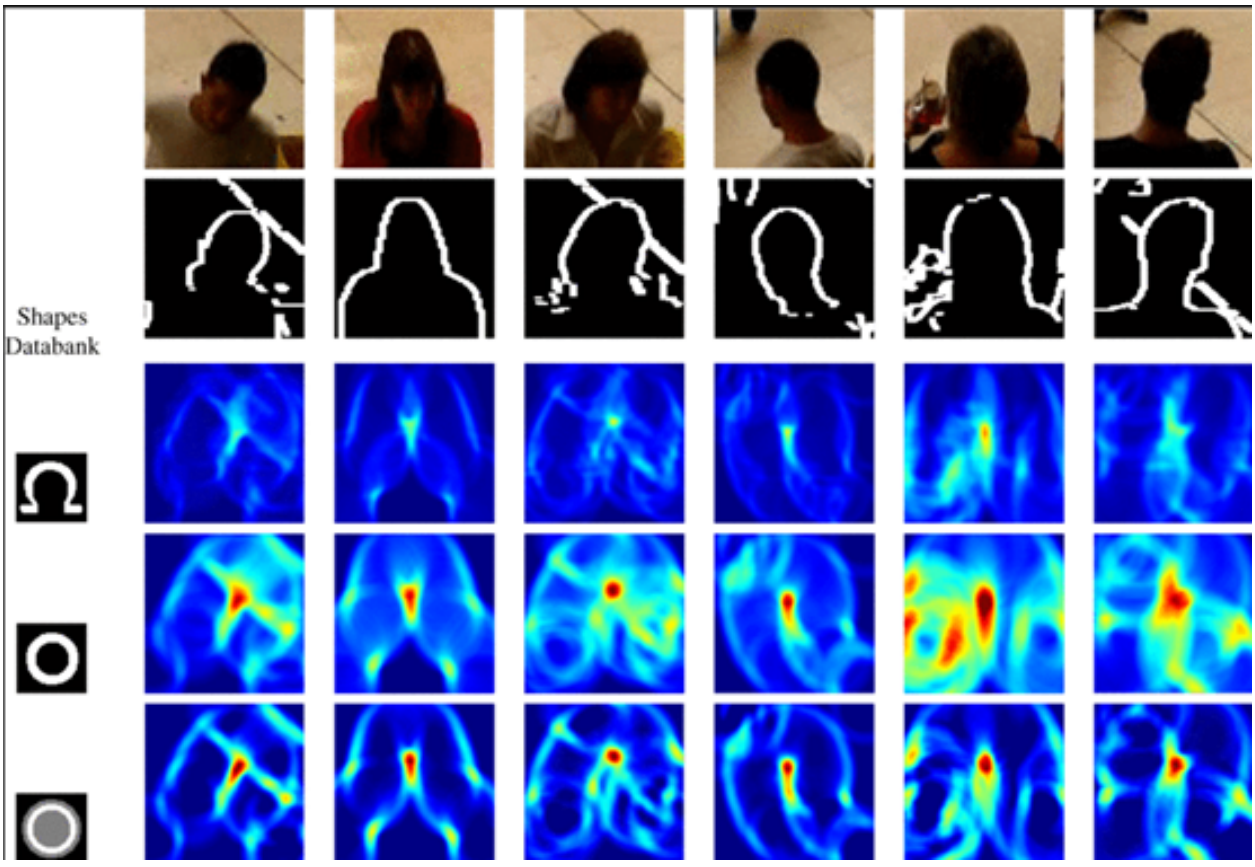

Figure 3: Head Detection for People Counter

As depicted in Figure 3, the technique utilized here involves the identification of heads through **shape matching**. While this approach performs admirably in the context of approximate people enumeration, with accuracies ranging between 87% and 98%, it does not capture any additional information related to head size or facial characteristics. The exclusive emphasis on head shapes and movement patterns ensures that the system remains both stable and cost-effective.

2.2.3 Face Mesh

Real-time image processing applications, such as video conferences, rely on face tracking to enable functions like beautification and cartoon avatar creation. These features require information regarding the size, shape, and facial characteristics of the user's face to generate the corresponding processed images. A widely recognized approach entails the identification and extraction of facial landmarks using machine learning (ML) algorithms trained on datasets, resulting in the creation of a face mesh. This technique offers real-time performance without the need for a specialized depth sensor [8].

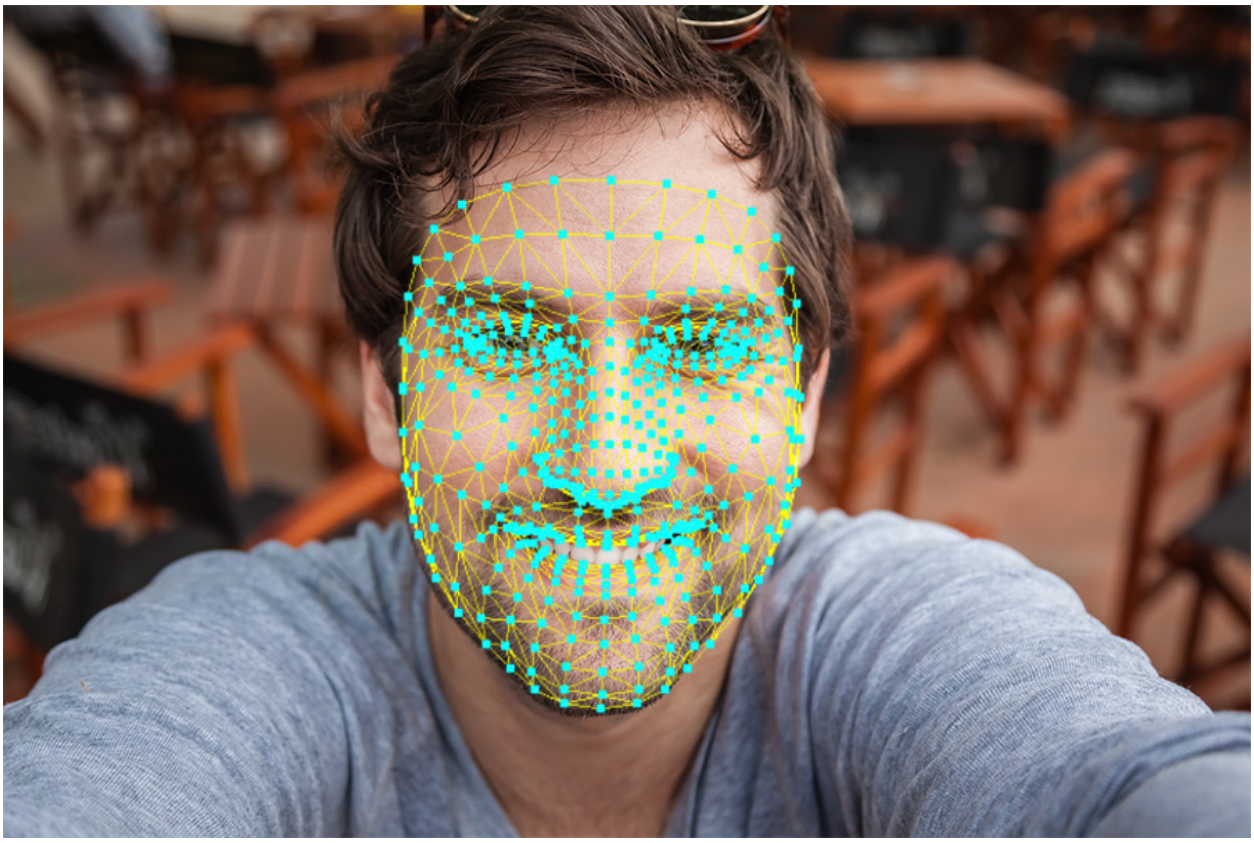

Figure 4: MediaPipe Face Mesh Solution Rendering

Face Mesh approach leverages facial features to achieve real-time face recognition and tracking. This technique employs a well-established machine learning pipeline consisting of two real-time deep neural network models, which work in tandem to accurately regress 3D face surfaces.

Unlike other methods, the face mesh solution is solely based on computer vision and a pre-trained model, thus eliminating the need for additional equipment like infrared projectors or depth cameras. As well as providing face position tracking, the face mesh also supplies information on facial shape, facing angle, facial expression, and other parameters.

2.3 Body Tracking

2.3.1 MediaPipe Pose

The utilization of body tracking is ubiquitous across various domains such as sports, health monitoring, video surveillance, video conferencing, and interactive projects. In the realm of computer vision, body tracking methods are akin to that of Face Mesh, as exemplified by **MediaPipe Pose** - a prominent machine learning-based approach that offers high-precision tracking of body poses [1]. This method involves the inference of 33 distinct 3D landmarks across the entire body, as depicted in the figure below.

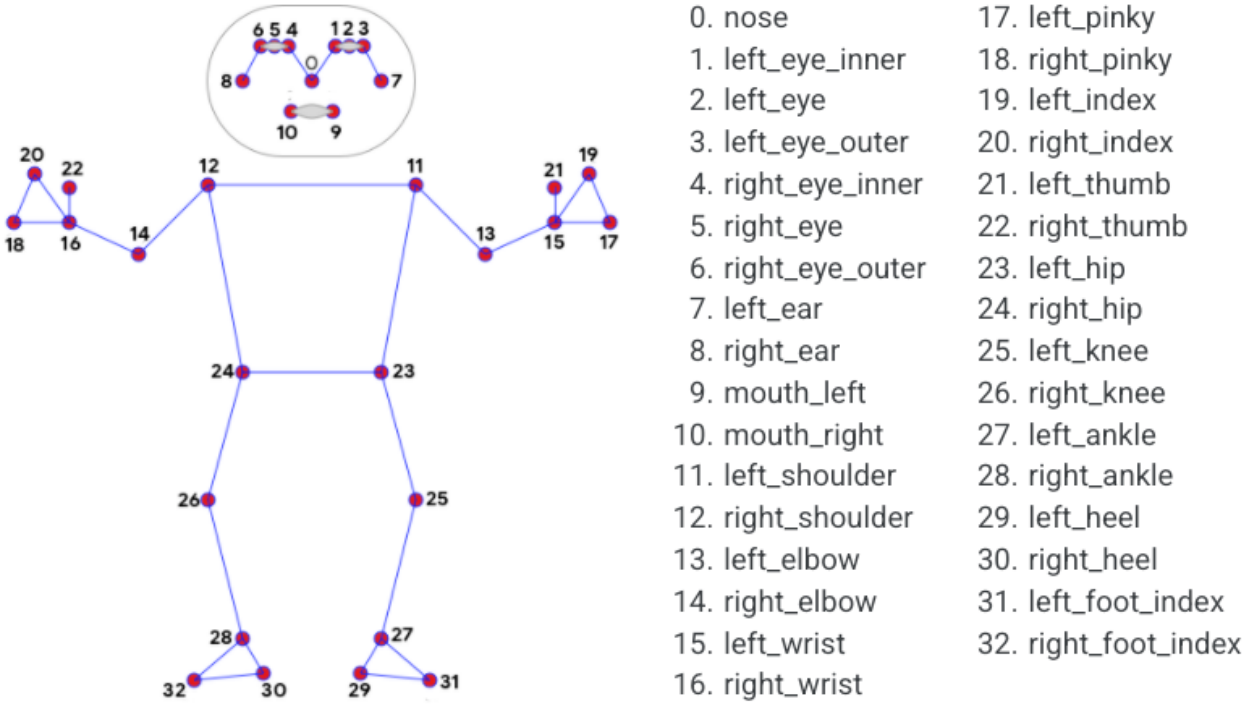

Figure 5: Pose Landmarks Sample [1]

From a theoretical perspective, the method based on landmark detection can be applied to face detection, as it encompasses 11 face landmarks within the landmark group. Nonetheless, this approach fails to provide the level of precision required for most computer vision projects involving faces.

Advanced techniques typically rely on depth or LiDAR cameras to attain high accuracy and good real-time performance, especially in cases where the body is in non-frontal poses or partial visual occlusions.

2.3.2 Depth Camera with Inertial Sensor

A tracking method utilizing a depth camera and inertial sensors worn by the user has been introduced, as it can offer several benefits over other tracking methods [12]. Drawing on prior tracking techniques, it combines a generative tracker and a discriminative tracker to obtain the closest poses in a database. The additional sensors enable the development of a new inertial-based pose retrieval and an adapted late fusion step to calculate the final body pose.

One of the advantages of this method is its ability to maintain tracking in challenging body positions or when the body is partially obstructed. This is particularly useful in scenarios such as sports training, physical therapy, or rehabilitation exercises, where the user's body is often in non-frontal or complex poses. With a database of 50,000 poses, the accuracy, and overall performance are satisfactory [12]. Nonetheless, there are some limitations to this method. The requirement for a depth camera and inertial sensors restricts its use to specific environments and applications, and the cost of these sensors may also be a concern. Additionally, the accuracy of the method heavily depends on the quality of the database used for pose retrieval, which may require a considerable amount of data collection and preprocessing efforts.

2.4 Background Removal/Subtraction

The removal of background elements is a fundamental technique in video analysis, particularly in video surveillance. The concept of background removal/subtraction refers to a range of video processing methods that aim to differentiate between the foreground and background in a video sequence through the application of a background model [13]. The conventional approach involves evaluating the images based on three attributes, namely, foreground detection, background maintenance, and post-processing.

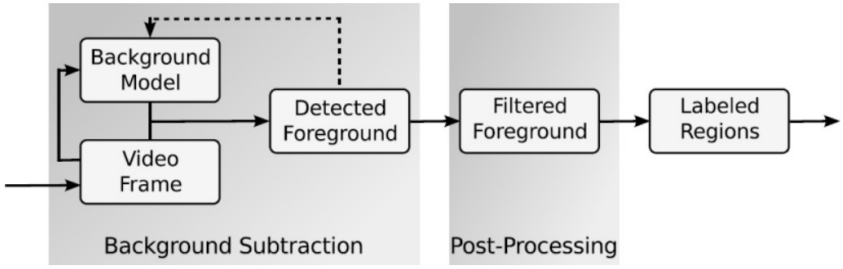

Figure 6: Typical Background Subtraction Process

The technique involves setting a threshold to check foreground pixels, followed by the separation of such pixels from the image. This is typically followed by a series of image processing steps to interpolate and predict the missing approximate pixels via regression [14]. While this method delivers high accuracy in detecting moving objects when the camera remains in a fixed location, it encounters some challenges in distinguishing the stationary foreground from the background.

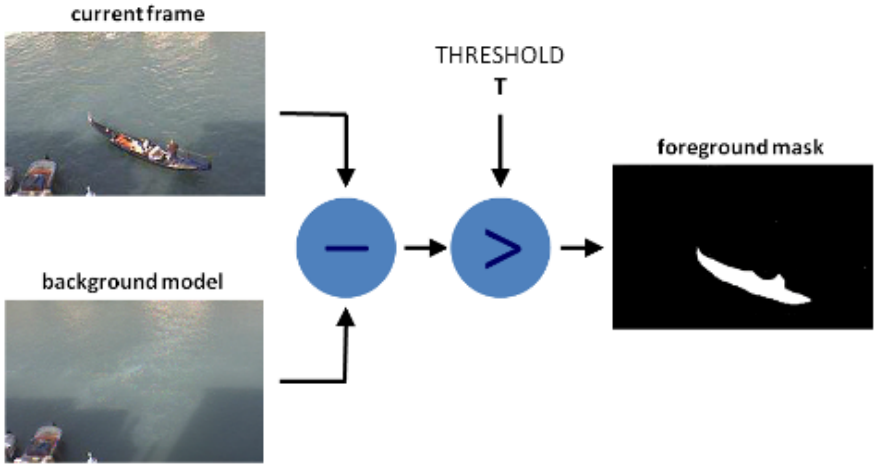

Figure 7: OpenCV Background Subtraction Example [2]

2.5 Networking

Online video conferencing has become an essential tool in today's interconnected world. Network protocols such as **Transmission Control Protocol** (TCP) and **User Datagram Protocol** (UDP) emerged to transfer packets of audio and video data across the internet. Both TCP and UDP protocols are widely used transport layer protocols that deliver data over an IP network, but the difference between TCP and UDP data transporting mechanisms characterizes them with advantages and disadvantages under different scenarios.

2.5.1 Transmission Control Protocol

TCP was developed in the early 1970s as a reliable protocol for transmitting data over the network. The protocol of TCP focuses on preventing potential data loss or errors during transmission by establishing a bidirectional connection between the sender and receiver. As shown in Figure 8 [3], each time the sender sends a packet to the receiver, the receiver returns a message back to the sender acknowledging the reception of the packet.

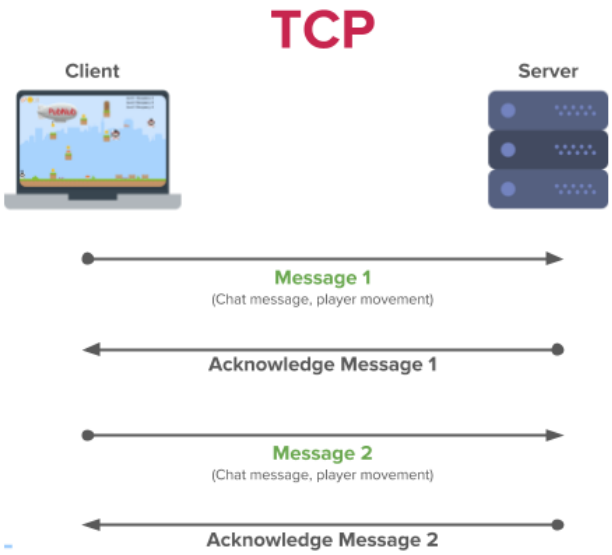

Figure 8: TCP Handshake

The TCP protocol encapsulates a header at the beginning of each TCP segment(packet). The TCP header carries important information about the data being transmitted. This information is stored followed by the structure shown in Figure 9 [15]. Specifically, the **sequence number** identifies the first byte of the TCP segment, and **acknowledgment number** records the number of the next data byte the receiver is prepared to accept [16].

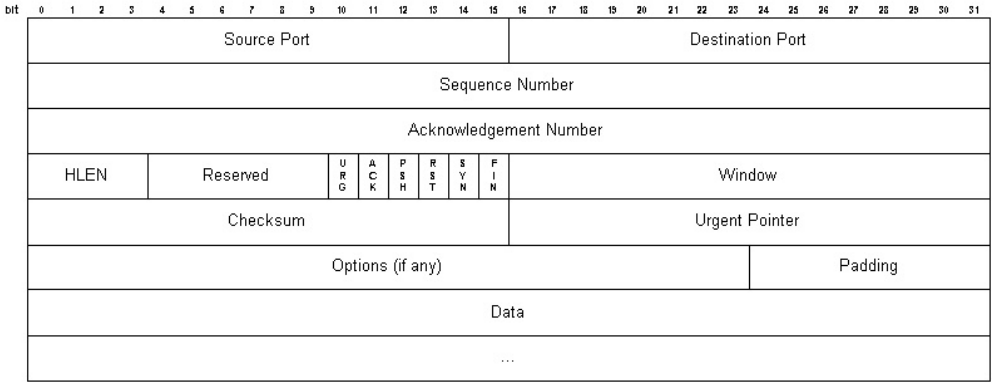

Figure 9: TCP Header

The nature of the TCP protocol determines its advantage in reliability and transmission quality. On the other hand, the TCP handshake exposes its limitation of time efficiency when processing high-volume data under a strict time constraint.

2.5.2 User Datagram Protocol

The UDP protocol was developed in the 1980s for low latency and fast communication. As an alternative to TCP, UDP protocol does not require a handshake for every transmission between sender and receiver. This feature reduced the header size and the delivery time of each packet. As a trade-off, the receiver sacrificed the ability to check the data integrity, and the sender will never know if the packet has ever been delivered (see Figure 10). Without modification, the UDP protocol lacks reliability when handling sensitive data.

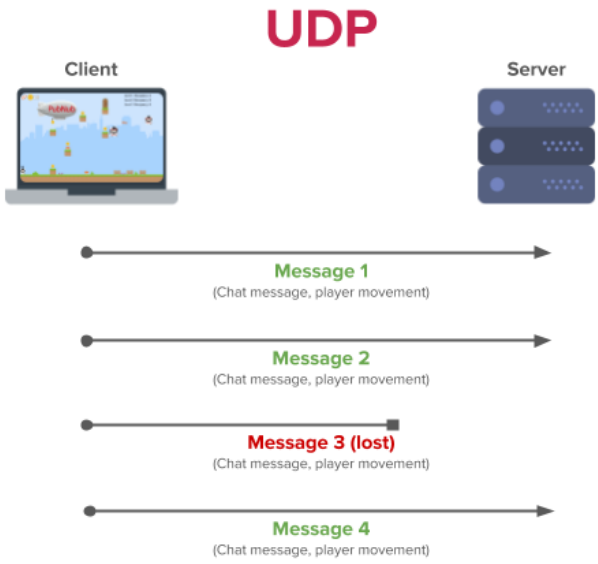

Figure 10: UDP Protocol [3]

2.6 Image Augmentation

Image Augmentation is a significant key factor in image processing. With **Image Kernel**, the grey-scale images can be sharpened, blurred, outlined, embossed, and so on [17]. An image can be read as a matrix where each element represents a pixel of the image. The value of each element is between 0 and 255, referring to the brightness. The image kernel itself is a small matrix to apply these processing effects. Take a 3 x 3 kernel as an example. It influences a group of pixels in the range of 3 x 3 and repeats this step for all the pixels. The entries of the kernel matrix will be the corresponding coefficients of the pixels' brightness values. The sum of nine products indicates the new brightness value for a pixel after processing. For instance, if an image needs to be blurred, the brightness of pixels should be lowered at a different scale. The center pixel should be a bit brighter than the border ones. Thus, a **Blur Kernel** should look like:

$$\begin{Bmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{Bmatrix}.$$

If an image needs to be sharpened, we will want the contrast of the adjacent pixels to be clearer. The center pixel's brightness should be enhanced while the border faded. A **Sharpen Kernel** should look like:

$$\begin{Bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{Bmatrix}.$$

The following figures show a simple comparison.

0	0	0
0	1	0
0	0	0

(a) Original Kernel

(b) Original Image

Figure 11: Original Image without Processing

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

(a) Blur Kernel

(b) Blurred Image

Figure 12: Image after Blurring Process

0	-1	0
-1	5	-1
0	-1	0

(a) Sharpen Kernel

(b) Sharpened Image

Figure 13: Image after Sharpening Process

There are more effects that could be applied with the proper arrangement of the image kernel. Images after processing could have clearer features for analysis. Image processing works efficiently in many areas including image enhancement, pictorial pattern recognition, and the efficient coding of pictures for transmission or storage [18].

2.7 Line Detection

Line Detection is an algorithm that takes multiple edge points and finds all the lines on where these edge points lie [19]. It is widely used in image processing. When images are to be used for different analyses such as object recognition or object positioning, it is important to reduce the amount of unrelated data in images while preserving the target data and structural information. Currently, there are three main solutions for line detection: the Hough Transform, the Convolution-based Techniques, and the Canny Edge Detector.

2.7.1 Hough Transform

Hough Transform is a straight line detector and the output is a parametric description of the lines in an image such as

$$r = xcos(\theta) + ysin(\theta).$$

That means for Hough Transform, the lines in images are described in **Polar Coordinate**. In general, we can define a set of lines that goes through point (x_0, y_0) as

$$r_0 = x_0 cos(\theta) + y_0 sin(\theta),$$

so each pair (r_0, θ) represents each line that passes the point. For a given (x_0, y_0), we can plot a set of lines that goes through it in a plane $\theta - r$, and we will get a sinusoid curve. Repeating this operation for all the points in an image will generate multiple curves. If two curves of different points intersect, then both points belong to the same straight line [20][21]. For instance, drawing the plot for (x, y) points (8, 6), (4, 9), and (12, 3), we will have:

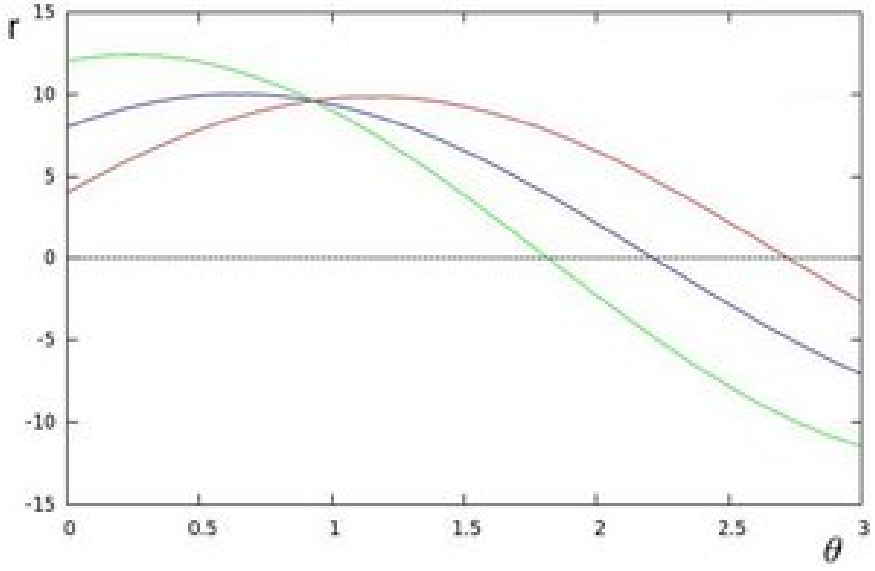

Figure 14: Example: Finding line at (0.925, 9.6) by Hough Transform

As the three curves intersect at (0.925, 9.6), they are in the same line. Thus, in general, the Hough Transform method detects lines by finding the intersection between curves. With more intersections, the output line has more points on it. We can set a threshold for the minimum number of intersections to narrow down the valid candidates of lines. The Hough Transform method is a classic plane computer vision algorithm that grants high flexibility.

2.7.2 Convolution-Based Method

In a **convolution-based** method, the line detector consists of a convolution mask tuned to detect the presence of lines of a particular width and an orientation (n, θ) [22]. The following are four example kernels:

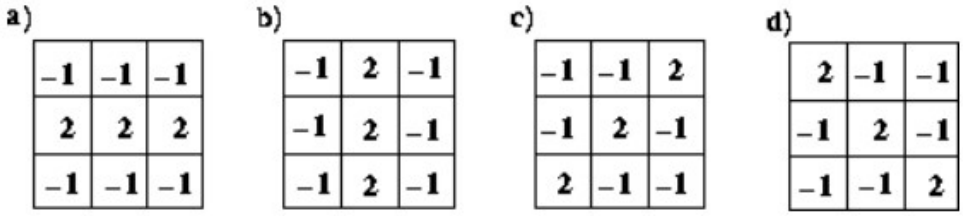

Figure 15: a) Horizontal b) Vertical c) Oblique + 45 degree d) Oblique - 45 degree

The masks will run over the image, and if a mask is overlaid on the image, multiply the coincident values and sum all the results. The output will then be the convolved image. For example, [23]

						Horizontal line				convolved image		
				0	0	0	0		-	-	-	-
				1	1	1	1	=	-	6	6	-
	Mask		*	0	0	0	0		-	-	-	-
−1	−1	−1										
2	2	2										
−1	−1	−1										
			*			Vertical line				convolved image		
				0	0	1	0		-	-	-	-
				0	0	1	0	=	-	0	0	-
				0	0	1	0		-	-	-	-

Figure 16: Example: Convolution-based Detector Finding Horizontal Line

Convolution-based line detector is a feasible method but tends to have negative performance when finding dark lines against a light background.

2.7.3 Canny Edge Detector

There are more methods that also can perform line detection. One is the **Canny Edge Detector** with assist of the Gaussian Filter method.

Canny Edge Detector is usually used to take a normal grey-scaled image and output a lined black-and-white image. The basic working principle can be summarized as (1) finding the intensity gradients of the image, (2) applying gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection, (3) applying a double threshold to determine potential edges, and (4) track edge by hysteresis [24]. Some research groups realized the enhancement of Canny detection by scale multiplication [25]. The Canny Edge Detection is a multi-stage method as shown in the figure below. [26]

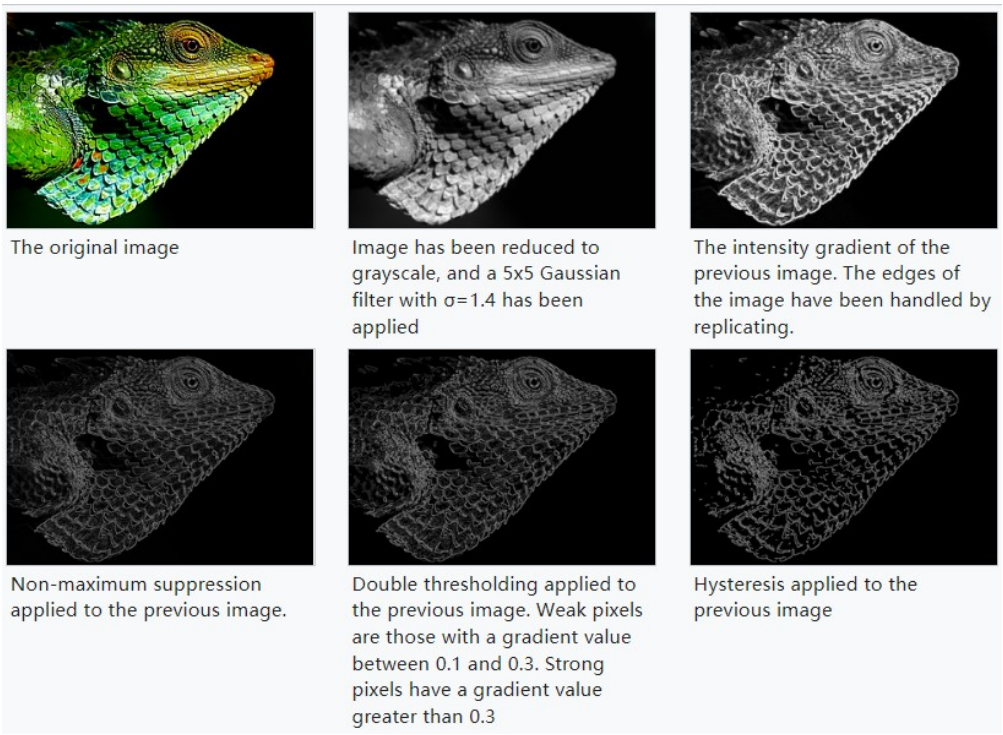

Figure 17: Canny Edge Detection Processing Stages

As mentioned in Figure 17 above, the image was processed with a Gaussian filter in one of the stages. The **Gaussian filter** is used to filter out noises of the images so that the Line Detection will not be affected. To smooth an image, the Gaussian kernel is convolved with the image. Thus, basically, the Gaussian Filter method is also a kind of convolution-based method. The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}); 1 \leq I, j \leq (2k+1).$$

The Gaussian Filter can also be customized for the horizontal and vertical scales to better match the processing purposes. It is widely used for image processing and augmentation. Although Gaussian Filter itself does not work as a line detector, it could assist and enhance most of the line/edge detectors.

2.8 Limitations & Potential

Current models of online meeting platforms have several limitations and untapped potential. For instance, head (face) tracking solutions are currently only used for specific visual effects such as beautifying or avatars. However, this approach provides additional information about the detected face, including the facing angle, absolute position, and size of the face, which could be used for a wide range of applications. In this study, we leverage this information on the facing angle and size of the face to develop the VTHO function and ADR function and use the information on absolute position to establish the boundary warning section.

Moreover, the existing background removal functions available in most online meeting platforms are often insufficient. Typically, users opt for background removal and replace it with a virtual background, but the recognition of the human body at the contour is frequently unstable. Although the background subtraction method is highly accurate in analyzing a single image, its performance deteriorates in real-time conditions. In real-time conditions, both the environment and the foreground pixels are constantly changing, and the background subtraction method takes a relatively long time to process. Additionally, it performs poorly in adapting to artificial lighting environments. Indoor lighting environments primarily depend on artificial light sources, which change at specific frequencies. Therefore, it is challenging for the background subtraction method to recognize background pixels with rapid and repetitive changes.

Our proposed designs aim to address these issues and unlock the potential for the further development of online meeting platforms.

3 Technical Design

This chapter elucidates the operational principles, functional details, and essential insights regarding the five technical design sections. Our objective is to develop software solutions that are accessible to all online meeting users, without requiring additional equipment such as AR/VR headsets or depth/LiDAR cameras. The majority of the designs are based on established or currently utilized technologies or concepts, with pertinent background information being presented in the preceding chapter. Further details can be found in Section 2.

3.1 Multiple Target Window Merging and Alignment

The conventional display mode of online meetings involves rendering participants in small, separate boxes that are then arranged in a grid format. However, the Multiple Target Window Merging & Alignment (WMA) function seeks to merge and align these individual display windows to create a more lifelike and authentic meeting environment. This function mitigates the cognitive demands placed on users and reduces the likelihood of "Zoom Fatigue."

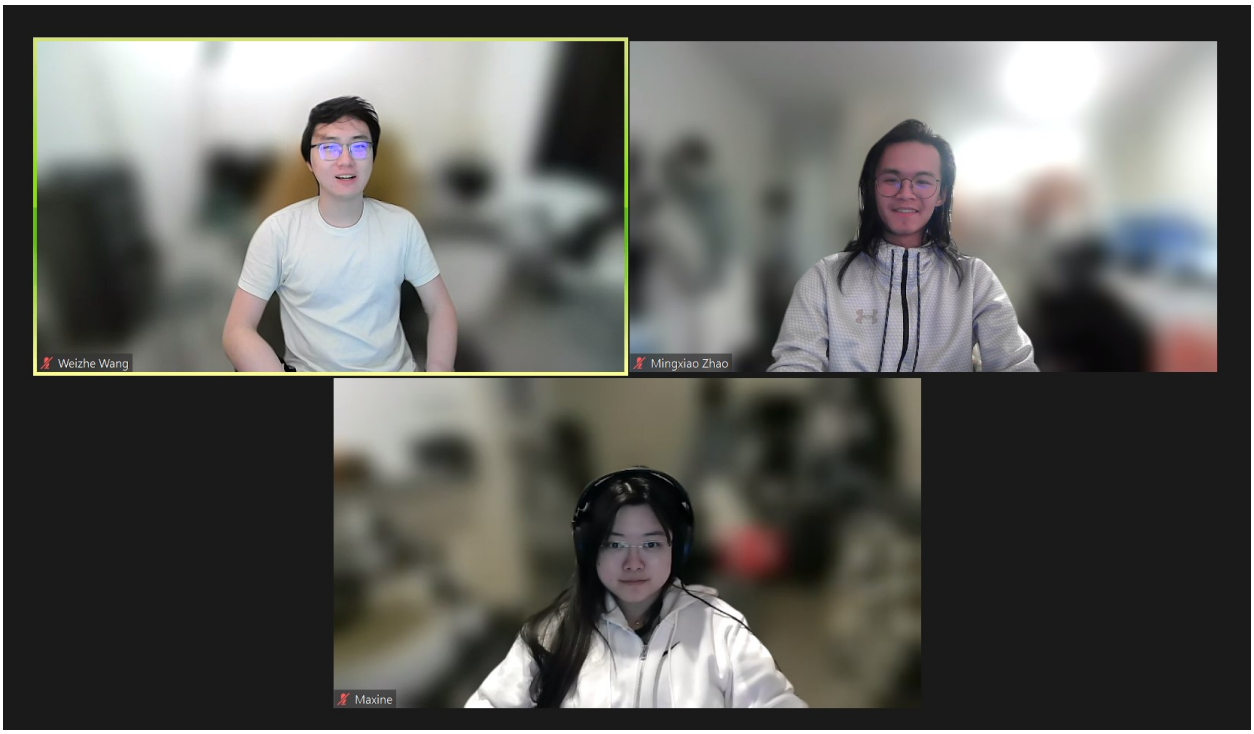

Figure 18: Zoom Display Mode

Figure 19: WMA Display Mode

Figure 19 depicts the outcomes of the merging and basic alignment function, whereby the challenge of disparate table colors is addressed by segmenting the table pixels and assigning a uniform RGB value. Further details pertaining to this approach are expounded upon in Section 3.2. Compared with Figure 18, WMA only displays the target participants but not the main user, which aims to intimate the reality.

To accomplish the merging function, the raw images undergo a series of processing steps, which involve determining the absolute position of the lower edge of the human body for each input image, stitching the images together, and aligning the y-axis position of the human figures.

If applicable, the absolute position of the lower edge of the human body is critical in identifying the table. Since our design lacks depth or LiDAR cameras, it solely relies on 2D computer vision. The approach involves selecting several appropriate straight lines in the image and determining which line should be designated as the table edge. As illustrated in the block diagram, the process involves four distinct stages of image processing.

To capture the details of the table, we set the camera vertical, which can also simulate the conference situation with a smartphone camera. These four processing stages depend on pure 2D computer vision algorithms. The step-by-step processed effect after each stage is shown in Figure 20.

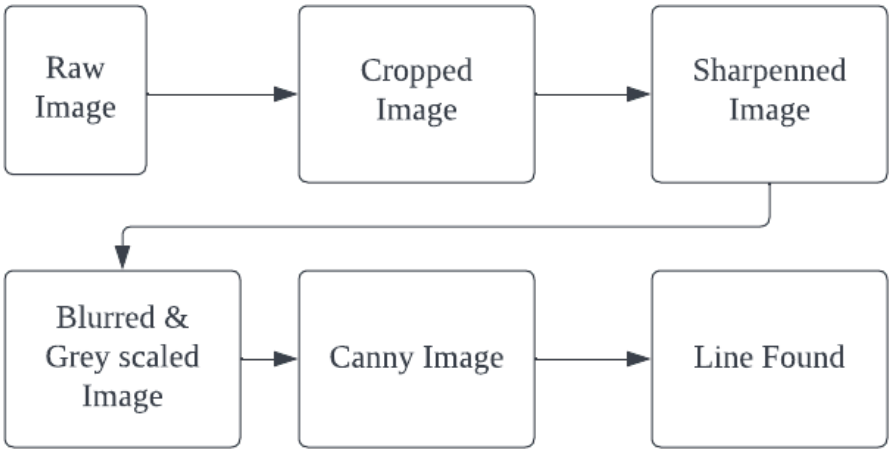

Figure 20: Edge Detection Block Diagram

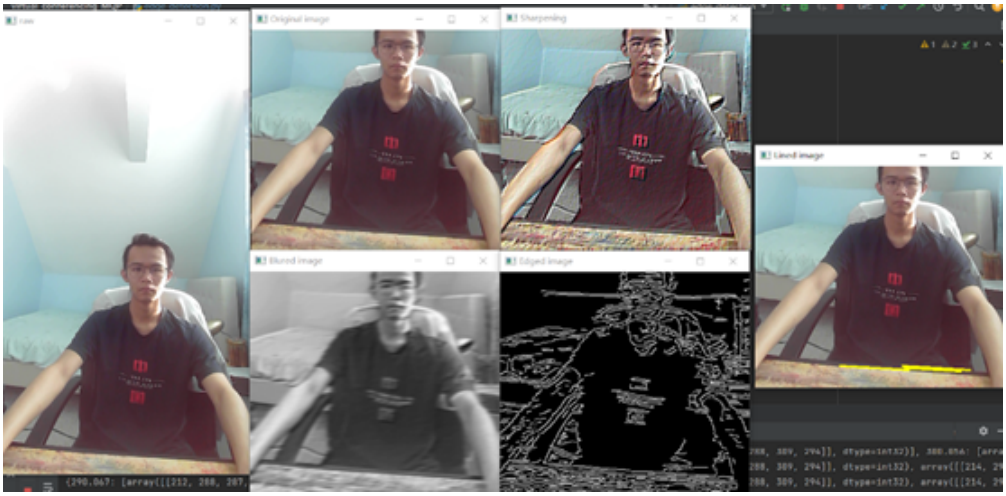

Figure 21: Effect on Each Processing stage

The purpose of the **cropping** stage is to reduce the scope of the processing area, as the table edge is expected to appear at the bottom of the image. This step is intended to increase processing efficiency by reducing the amount of data that needs to be analyzed. Specifically, the Region of Interest is defined as the lower half of the image.

The **sharpening** stage serves to accentuate and approximate line-like features in the image. In this project, the programming language employed is Python, and the filter2D() method is utilized for image sharpening. This method, also referred to as Image Augmentation by Image Kernel, is introduced in Section 2.6. The filter2D() method accepts several parameters, three of which are particularly significant. The "source" parameter corresponds to the unprocessed input image, "ddepth" designates the expected integer value of the output image's depth (default/non-change = -1), and "kernel" denotes the matrix employed to implement the desired process. [22].

```
import cv2
import numpy as np
#Read an image here
#Crop the image here
kernel = np.array([[-2, -1, 0], [-1, 1, 1], [0, 1, 2]])
emboss = cv2.filter2D(src = cropped_image, ddepth = -1, kernel)
cv2.imshow("Sharpening", emboss)

```

Listing 1: Image Sharpening Example

The above is an example of a sharpening process. We input the pre-cropped image, set a kernel as

$$\begin{Bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{Bmatrix},$$

and display the output image of this stage. A series of images with effect after each stage can be found in Figure 21.

The **Blur & Grey Scale** stage is a further processing step after sharpening. The Gaussian Blur method is employed to blur the image, with more emphasis on the horizontal direction, as the table edge is expected to be closer to a horizontal line. The horizontal lines should be more distinguishable after this step, simplifying the line checking in the subsequent stage. As the stage after this step only accepts grey-scaled images as input, the image is converted into greyscale in this stage. The objective of this series of processing is to identify the table edge or lower human body edge, and hence, the color of the image does not affect the result.

The subsequent stage in the processing pipeline is the **Canny** stage, which takes as input the grey-scaled image output from the previous stage. The purpose of this stage is to apply the Canny Edge Detection method in order to identify suitable edge candidates. This method is a multi-stage algorithm that involves performing Noise Reduction, Intensity Gradient Finding, Non-maximum Suppression, and Hysteresis Thresholding [27].

The figure depicting the outcome of the Canny Edge Detection method is presented in the sub-figure located at the middle right bottom of Figure 21. The application of appropriate thresholds in this stage results in the image containing only straight lines (edges) rather than color blocks. This image serves as the input for the final stage of edge detection.

The detection method is the Hough Line Transform method. The variables of this method include but are not limited to the minimum and maximum lengths of target lines, the minimum gap between lines, and an array to store the information of detected lines.

```
        import cv2
as        import numpy as np
        # An cropped input image being blurred here

        edged = cv2.Canny(blurred, threshold1=10, threshold2=50) #Canny Process
        linesP = cv2.HoughLinesP(edged, 1, np.pi / 180,
                                 threshold=15, lines=np.array([]),
                                 minLineLength=30, maxLineGap=3) #Hough Transform
    Detect

        if linesP is not None:          # Draw and Display the lines
        for i in range(0, len(linesP)):
            l = linesP[i][0]
            cv.line(cdstP, (l[0], l[1]), (l[2], l[3]), (0,0,255), 3, cv.LINE_AA)
        cv.imshow("Detected Lines", cdstP)

```

Listing 2: Lines Detection & Drawing Example

However, the basic application of edge detection, as described above, may not be sufficient. Despite the horizontal Gaussian blur applied in the previous stage, some vertical lines may still be present in the resulting image. To address this issue, certain conditions are applied to filter out undesired lines.

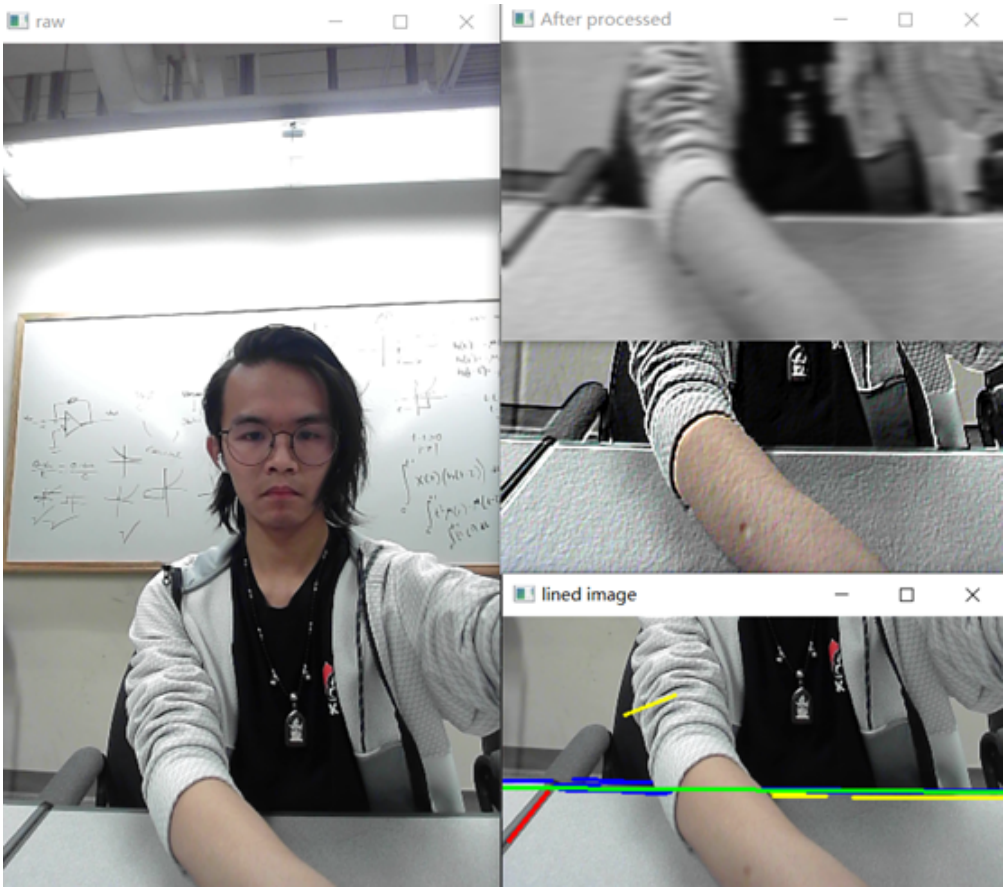

Figure 22: Detected Lines with Validity Check

As illustrated in Figure 22, a color-coded scheme is employed to differentiate between different types of lines detected by the Hough Line Transform method. The Red lines are eliminated as they fail to satisfy the slope check criteria. All Yellow lines passing the slope check are considered valid candidates. However, a subset of these lines may appear in an inappropriate location. Therefore, we group the valid candidates with similar Y-axis values and slope values. Once a certain group of elements reaches a reasonable number, they are labeled as Blue lines, which indicates that they are selected candidates for approximating the table edge. Based on the Blue lines, we derive an approximation of the table edge and mark it with a Green line. This approach enables us to detect the table edge despite obstructions such as the arm shown in Figure 22.

With the aforementioned processing steps, we obtain the absolute position of the human body, which allows us to conduct window merging and alignment.

3.2 Background Removal and Foreground Alignment

The core algorithm underlying the WMA in Figure 19 consists of two primary components: background removal in the designated area and foreground alignment based on the given edge location.

For background removal, we evaluated multiple approaches to identify an optimal real-time solution, leveraging OpenCV and Image Segmentation methods. While exploring options, we first considered the background subtraction algorithm from the standard subtraction to MOG2. Although this method exhibited good accuracy in distinguishing foreground and background objects and boasted outstanding real-time performance, it demonstrated low stability in real-time conditions, which could easily be influenced by camera exposure or unstable lighting environments. Next, we explored foreground extraction using the stereo/depth camera. However, the accuracy of depth output from stereo vision cameras, such as the ZED Camera, suffered from the light condition in the user's room, resulting in low stability when separating the user from the background. Overall, its performance in real-time conditions was even worse than background subtraction. Therefore, we instead opted for the image segmentation solution provided by MediaPipe, which offered fairly stable and real-time capability.

The primary challenge in optimizing the user's visual experience lies in combining each individual camera feed into one virtual environment since the most critical criterion to measure the user experience is how well each user is aligned with the other. The virtual environment consists of two primary components: the background and the conference table. Ideally, the real-world table of each conference participant should align with the virtual conference table. To achieve this, the program required the relative edge location for each meeting participant and the ratio to scale it to the coordinate of the background canvas. To facilitate access to this data while processing each client's camera feed, we developed a data structure that attaches the table edge location and the alignment information with each video frame. When the video stitching program ran, it could easily fetch the alignment and scale parameters from this data structure. After obtaining the alignment parameters for each user's camera feed, the program could offset each user's foreground vertically to match the alignment line, which was the virtual table edge in the background shown in Figure 23.

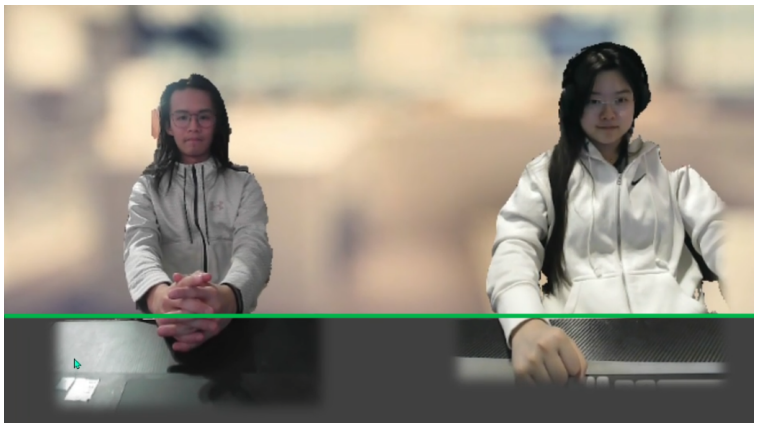

Figure 23: The virtual table edge (green line)

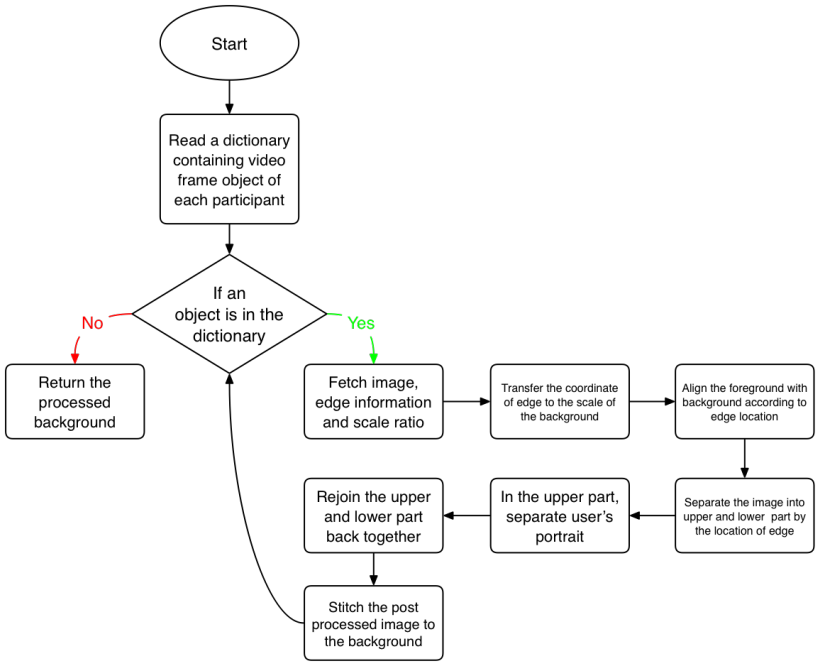

Figure 24: Flowchart of image processing and alignment

In the program, we implemented both foreground alignment and background removal in the stackIMG function. This function accepts five input parameters: a dictionary data structure to hold a collection of video frames from each user's camera, a background image for the virtual background, the required shape to fit the camera feed into the background canvas, and the horizontal spacing between each user, the distance from the user camera's boundary to the margin of the background canvas. The function can utilize these parameters to allocate each user's camera to their designated space on the background canvas. A flowchart in Figure 24 illustrates the image alignment and background removal procedures behind this function; at the beginning of the flowchart, the program fetches each meeting participant's frame object from the input

dictionary. This object is a data structure that includes the current frame of the participant's camera feed and the edge location, information that is critical for the program to alter the foreground's vertical location. After this process, the program will separate the foreground image by the edge into two pieces. As shown in Figure 25, The upper piece(blue) is above the table edge, and the lower part(green) is below the edge. The background removal program only processes the upper part, removing the participant's background. On the other hand, the lower part remains unchanged, retaining the portrait. After the background removal is completed, these two parts are stitched together and placed on the designated area on the background canvas.

Figure 25: Parsing the user's camera feed by the edge of table

3.3 Vision Angle Tilting by Head Orientation

The Vision Tilting by Head Orientation (VTHO) method is based on the face mesh algorithm provided by MediaPipe. The face mesh function can detect face landmarks and calculate the head pose angle. By taking the nose as the center, the head pose angle (x, y) can be utilized to adjust the display window, similar to the action of a scrollbar in the event of multiple meeting participants. During the calibration stage, the user can set the initial head pose position to zero. During the meeting, the user can turn their head slightly to the left or right to shift their focus towards a specific participant. To create a more natural user experience, vertical vision tilting is enabled for a slight change in the angle. The angle of the head tilting and the corresponding vision-changing angle are in an appropriate ratio since the screen's width and length are limited, and excessive head movement can be awkward. The ratio can be easily adjusted within the program code.

The following three figures show the effects of VTHO. As the user looks left, forward, and right, he will see the left part, middle part, and right part of the target display.

Figure 26: VTHO Applied to Sample Target Window: Look Left

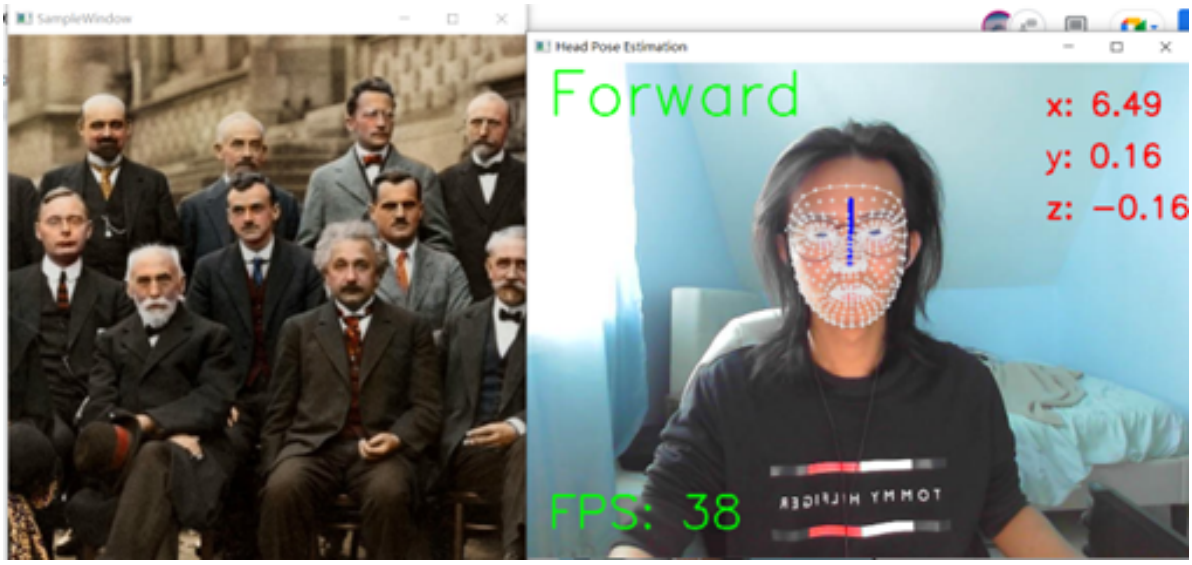

Figure 27: VTHO Applied to Sample Target Window: Look Forward

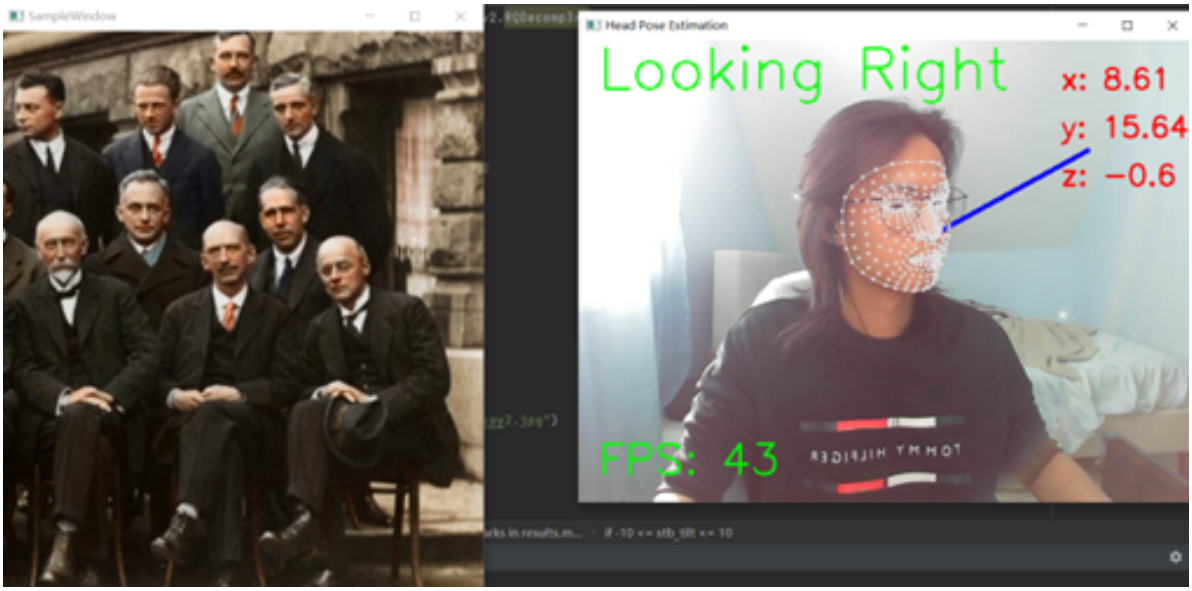

Figure 28: VTHO Applied to Sample Target Window: Look Right

The proposed method for achieving VTHO involves the allocation of pixels for the head posing angle, as well as the display of the selected area. However, a direct linkage between the head pose angle and the display area proportionally may result in "jumping" problems where the displayed image changes too rapidly when the speed of turning the head is high. In turn, this can cause the display image to appear unstable and bounce continuously.

To address this issue, the proposed method incorporates a debouncing algorithm to render the field of view more natural. Specifically, the method dynamically records a list of head-tilting angles (with a standard of 10 elements) and returns the average value for allocation. The list continually discards the earliest data when recording a new one to maintain the ten-element list and ensure that the average change is not excessively large. As a result, the proposed method can enhance the fluency of display during vision changes.

```
            import cv2
            import mediapipe as mp
            import numpy as np

            if len(self.tilt_buffer) >= hist:
                # pop the first element of the buffer when buffer > 10
                del self.tilt_buffer[0]
            self.tilt_buffer.append(head_dir)

            loc_tilt_buffer = self.tilt_buffer
            stb_tilt = np.mean(loc_tilt_buffer, axis=0)
            stb_x, stb_y = stb_tilt
            # calculate the ratio of head tilt to image rolling
            tilt_y_step = (y_center - halfFOV_w) / headbound_R
            # calculate the ratio of head tilt to image rolling
            tilt_x_step = (x_center - halfFOV_h) / headbound_U

```

Listing 3: VTHO Debouncing Algorithm

As illustrated in the provided sample code, in the event that the number of recorded elements is less than the pre-determined size, the list continues to record data. Once the list size meets the pre-set size, the earliest element is discarded while a new one is recorded. This debouncing method functions by recording head-tilting data and computing an average value that governs display changes. The resulting change will be more natural and smooth before applying the debouncing algorithm.

3.4 Auto Display Resizing

The previous study indicates that the different sizes of face images in an online meeting can increase the user's brain activity costs and thus make "Zoom Fatigue" more severe [7].

The Auto Display Resizing function aims to reduce the brain activity requirement by decreasing the differences in sizes between each displayed face of other meeting participants. The basic logic is to detect the size of each face, set a proper range of size as standard, compare the detected size and the standard range, and adjust the displayed image based on the result of the comparison.

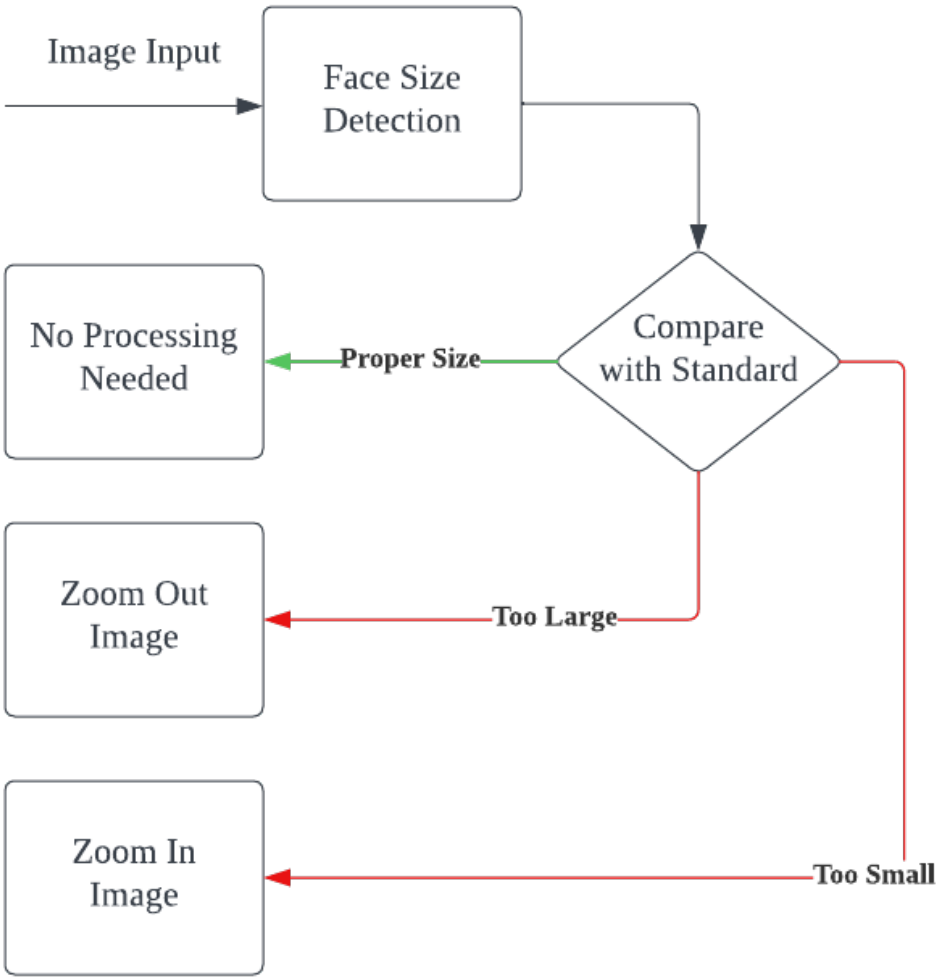

Figure 29: ADR Block Diagram

Upon detection of the face, the Auto Display Resizing function exclusively requires size information of the face, thereby obviating the need for the Face Mesh method. The ADR function relies on a lightweight model featuring merely six facial landmarks, thereby ensuring high-speed and cost-effective processing conditions [28].

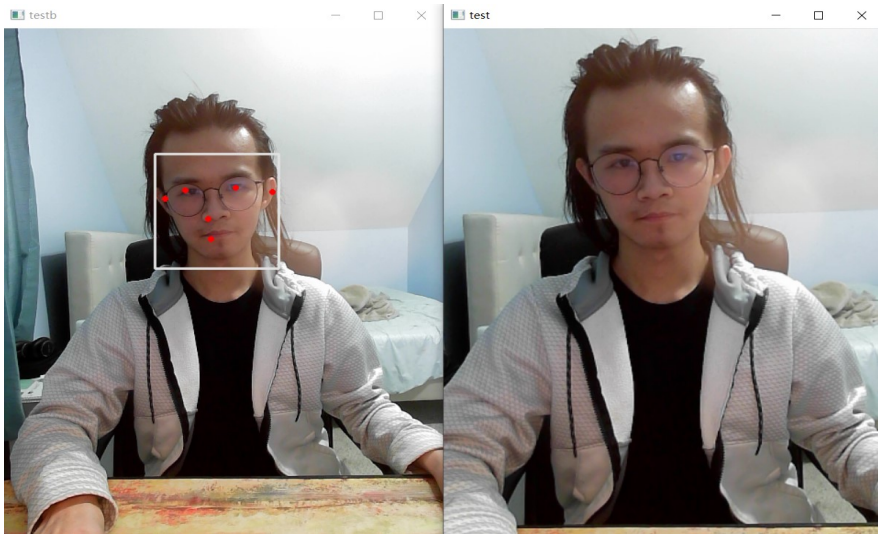

Figure 30: ADR Face Detection and Adjustment: Zoom-In Example

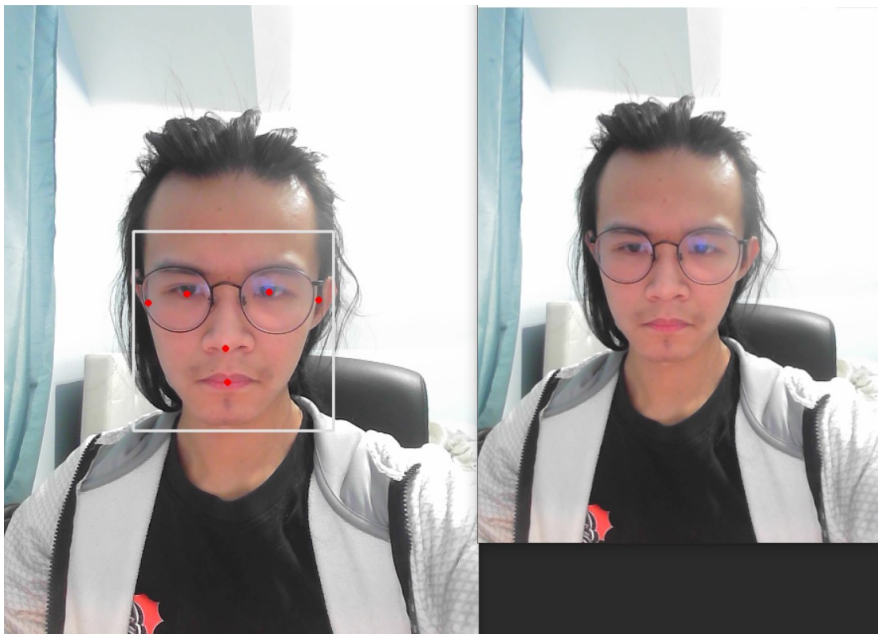

Figure 31: ADR Face Detection and Adjustment: Zoom-out Example

As shown in Figure 30 and Figure 31, the face frame strictly captures the Face but not the head. We do not capture the whole head so the size will not be influenced by the length or width of the head. A previous study published the average face width is to be 128.4 ± 10.1 millimeters [29]. This data is considered to be more constant than the head width value.

After the face size detection and the comparison, the ADR will adjust the image by zooming out or in at an appropriate scale. The ADR function can be used in real-time conditions, automatically and dynamically adjusting the images.

```
        import cv2
        import mediapipe as mp
        import numpy as np

        auto_rsz = AutoResize() #Resize function defined somewhere else
        h, w, c = image.shape
            ratio, bound_warni = auto_rsz.bound_n_resize(image, 170)
            adjust_w = round(w * ratio)
            adjust_h = round(h * ratio)
            new_shape = (adjust_w, adjust_h)
            if ratio > 1:
                rsz_image = cv2.resize(image, new_shape,
                interpolation=cv2.INTER_LINEAR)
                ah, aw = rsz_image.shape[:2]
                dn = round(ah * 0.5 + h * 0.5)
                up = round(ah * 0.5 - h * 0.5)
                lt = round(aw * 0.5 - w * 0.5)
                rt = round(aw * 0.5 + w * 0.5)
                rsz_image = rsz_image[up:dn, lt:rt]
            else:
                rsz_image = cv2.resize(image, new_shape,
                interpolation=cv2.INTER_AREA)


```

Listing 4: ADR Adjustment Example

Above is an example of a basic working model under certain scaling conditions. The conditions can be simply modified from the user's end.

Activating this function might result in the continuous changing of the target figure(s) if the target participant(s) keep moving during the meeting, which could cause an unnatural experience. Therefore, the ADR should be turned on and off easily by users during the meeting so that it will only be used to calibrate the image when needed.

3.5 Boundary Warning

Boundary Warning is not a function that directly improves the Quality of User Experience of online meetings but contributes to guaranteeing the validity of lasting image processing performance. When the user is in the middle of an online meeting using our applications, he/she will have a window displaying all other participants sitting side by side. However, not like the common display mode that most online meeting platforms are currently using, the users of our application will not see himself/herself in the window. We designed the display mode this way to enhance the realistic level since people in in-person can only see others as well. The problem is the user does not know if he/she is always in a proper position where the camera can capture the human figure or the required information since the user does not have a self-display window. The boundary warning function is designed to solve this problem while not influencing the QoE.

Face/head tracking is used for the development of this function too. The face-tracking method we used gathers information including the absolute position of the face/head. This information will be used to determine if the head is in a proper position so that the image can still be processed correctly.

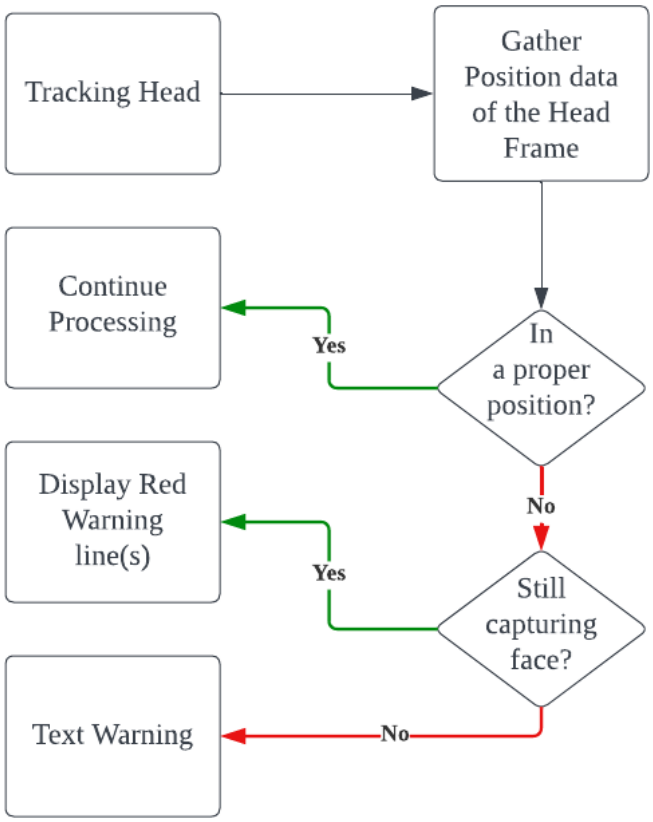

Figure 32: Boundary Warning Block Diagram

The determinative conditions for the position check are the distances between the head frame and the edges of the processing area. If the distance between the head frame and a certain edge is too close, the boundary warning function will be activated and displays a red line on the corresponding edge as shown in Figure 33. This function can also work when the head frame is too close to multiple edges, for example in the corner of the camera capturing area.

When the head frame crosses the edge(s) or when the head tracking method can no longer detect the head frame, a text warning will be displayed on the screen as shown in Figure 34.

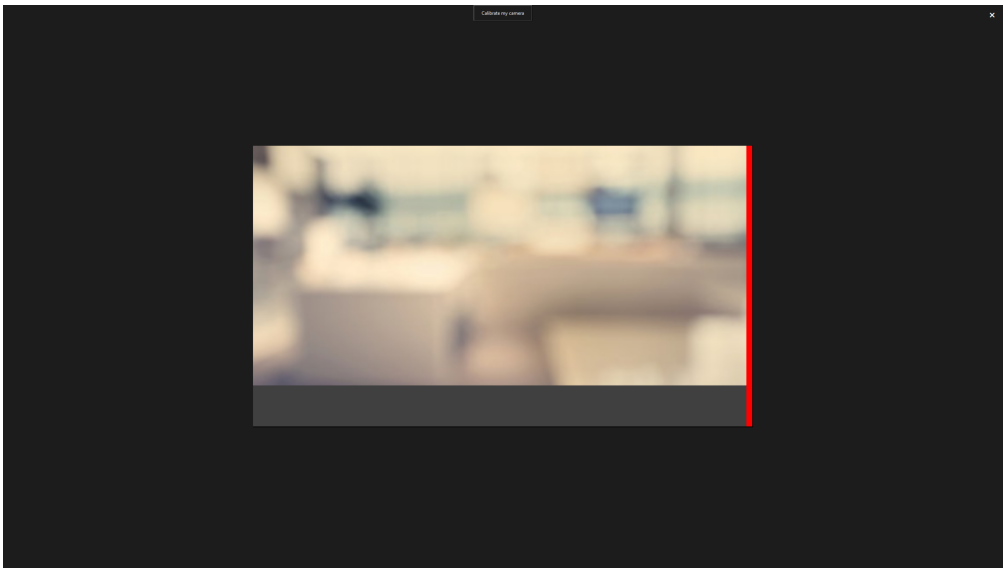

Figure 33: Boundary Warning When Head is Close to the Right Edge

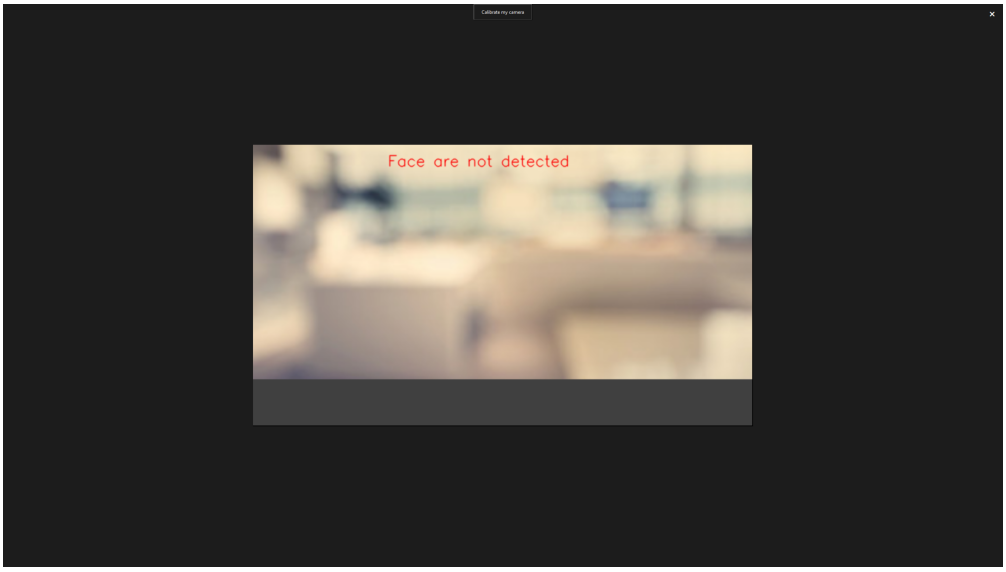

Figure 34: Boundary Warning When Head is Outside camera's FOV

4 Demo & Presentation

This chapter provides introductions to the user-end stages and some User Interface designs. To better convey our design ideas, we conducted some front-end designs which will be explained here. Upon the completion of reading this chapter, users should understand how to use the application we designed.

4.1 Working Stages

Before entering the meeting room, there is a calibration stage. In this stage, the project collects information to calculate the absolute position of the user. This result will be stored and fixedly activated during the meeting stage. It regulates the "table finding" function and guarantees the stability of the experience during the meeting stage. If the camera position or user position needs to be changed during the meeting stage, the user can come back to the calibration stage for re-adjustment. Also, the initial position and initial head-posing angle are detected and fixed during the calibration stage. The later calculation during the meeting stage will consider this initial position as zero.

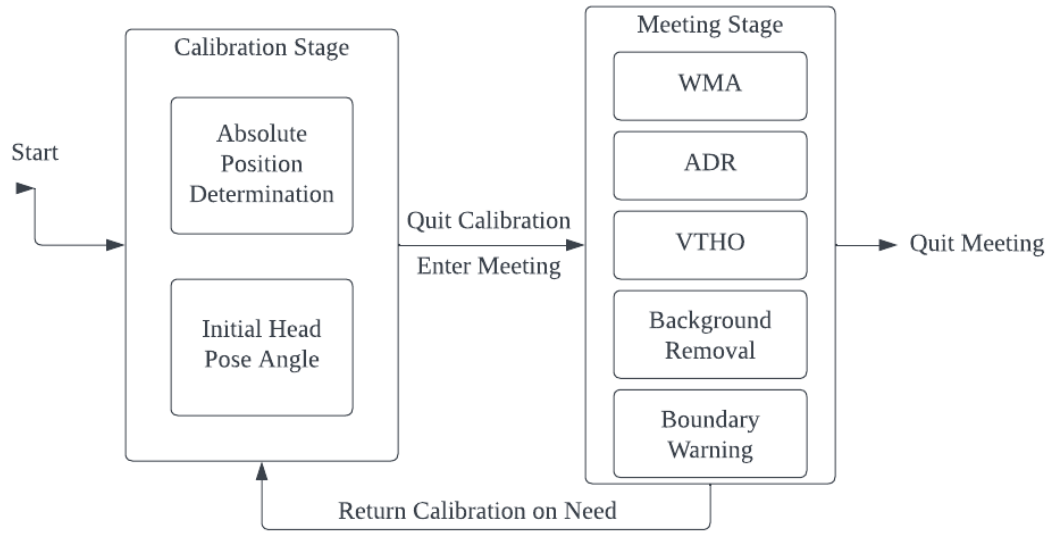

Figure 35: Working Stages

When the calibration is finished, users can quit this stage and automatically enter the meeting stage. During the meeting stage, all the designed functions will start and keep working until the end of the stage.

Basically, our demo application requires the least user participation to conduct the prior calibration. The users can return to the calibration at any time if re-adjustment is needed.

4.2 Graphical User Interface design

The user graphic user interface (GUI) is an essential feature for augmenting user experience and demonstrating the concept of our project. The GUI was developed using the Tkinter library, a widely-used Python module that offers a straightforward and flexible solution for constructing GUI applications. The principle of our GUI design is to enhance the ease of use for users, so we prioritized the minimalist design approach while retaining a user-friendly interface.

4.2.1 Main Window

The architecture of the user interface involves two layers: the main window and the pop-up calibration window. The main window serves as a primary window for displaying the virtual conference room. When the user decides to re-calibrate the application, they can access the calibration window by pressing a button located on the main window as shown in Figure 36.

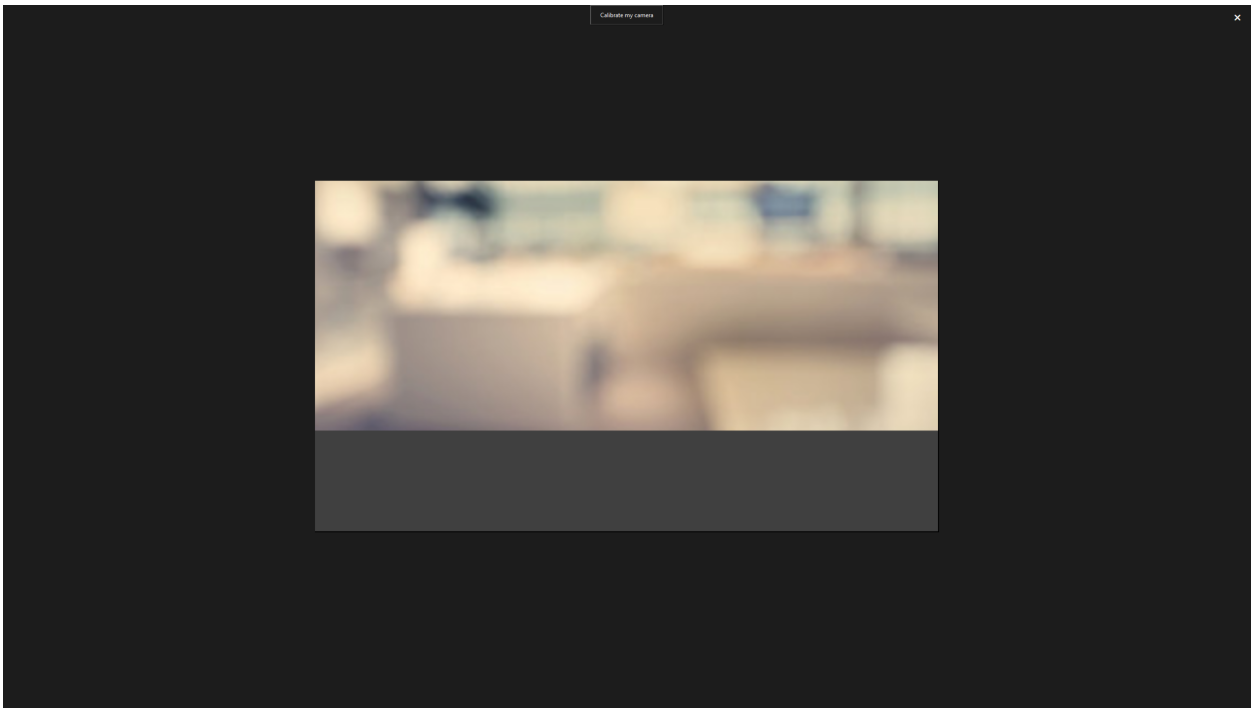

Figure 36: Working Stages

4.2.2 Calibration Window

The calibration window shown in Figure 37 enables the user to adjust the camera angle and recenter head tracking program. In this mode, the table edge is highlighted with different colors to indicate the quality of the table position. Meanwhile, the window prompts users with directions to adjust their webcam. After the completion of camera calibration, the user can exit this window by clicking on a confirm button or simply closing the window.

Figure 37: Calibration Window

5 Conclusion & Future Development

The goal this project focuses on achieving is to create one or multiple software solutions to solve the "Zoom Fatigue" problem and to improve the overall quality of user experience. Based on the background research, we further investigated the potential causes of "Zoom Fatigue" and explored solutions. The potential causes include the multiple small windows display mode, the auto jumping speaker view, and the varying head frame sizes of other participants. These factors are drawing huge attention and requiring a lot of brain processing, thus causing or exacerbating the "Zoom Fatigue."

To solve the specific problems, we proposed developing a window merging & alignment function to integrate all the target windows into one, a head-oriented vision-changing function to replace the dynamic jumping view, and an auto-resizing function to dynamically scale the head frames of the target participants to proper sizes. Two other assisting functions are designed and applied to enhance or guarantee performance. These functions aim to reduce brain activities and provide a more comfortable mental environment for users.

The proposals and designs for these functions are well-evaluated based on several factors. The main purpose of this project is to make the online meeting more realistic and natural since reality requires the least user brain activity. Virtual Reality (VR) technologies usually are an inevitable potential solution, but they have their own problems such as physical burden on the neck or cybersickness. Considering the inconvenience and the high cost of VR headsets, we decided to develop pure software solutions without VR or any other extra devices. Our project works with only one normal built-in or web camera. The program is mostly based on light-level trained models and 2-Dimensional computer vision, so it is expected to run on any common computer easily. To better convey our ideas, we integrate this project as a demonstration consisting of a simple user interface for free exploration.

While this project has made several useful contributions to the display mode and interactions of the online meeting, there is always more room for improvement. Regarding the WMA function and the background removal function, we recommend training a maturer model to boost the accuracies of absolute position determination and front ground segmentation. Currently, the position determination in the WMA function is based on 2-Dimensional image augmentation and detection. It works with low cost and low requirements but sometimes fails to narrow down the valid candidates. With time and budget limitations, we did not train a model for it. A well-trained model which could recognize table or desk surfaces through 2D cameras will increase the accuracy and speed of position determination. Similarly, a well-trained model will be able to improve the performance of background removal. The current background removal function does not have excellent performance when processing detailed parts such as the fingers and the hair. This problem exists in most of the current online platforms. A mature trained model should make some improvements but it is expected to take a long time for training.

Also, we recommend bringing the processing to a more powerful server. Currently, most of the processing is executed on each user's end and the data is transmitted through the network. This method distributes the calculation requirements to all the users. If all the calculation and processing can be done on a powerful server and then only send the corresponding output data to each user's end, the device requirement will be lowered significantly. When this step is achieved, the program should also be able to be run on mobile devices, further expanding the applications. We also recommend the future group develop mobile-end applications. With the development of mobile-end streaming and video platforms, the public is now showing more interest in mobile-end applications because of their convenience, so we believe a strong and stable mobile application will be a trending pattern of video conferences.

In addition to the enhancement potential, we also have some new ideas and concepts. At the beginning of the project, we had an idea to place 3-Dimensional figures of all the participants including the user him/herself into a virtual room. Participants are sitting around a round table, and each member can see others by turning his/her head. The visual output should be similar to VR games but all the figures are not cartoon avatars but real human figures, and the display will be on the screen but not through a headset. This mode best restores the reality scene and requires the least extra devices. The conceptual graph of this display mode can be found in Figure 38 under Appendix A. However, this method still needs multiple cameras for each user to capture the 3D details and create the human figure in the virtual room. Also, it would need a lot more time and cost. These are the reasons we discarded the plan. However, we believe such a development path has potential and will be valuable if more investments are made. To succeed in this purpose, we recommend students with strong Computer Science or Interactive Media & Game Development backgrounds to take over and continue exploring.

References

[1] "MediaPipe pose."

[2] "OpenCV: How to Use Background Subtraction Methods."

[3] "UDP vs TCP: Why to Run Gaming Servers Separate from Chat."

[4] B. Evans, "The zoom revolution: 10 eye-popping stats from tech's new superstar."

[5] E. Gao, "Tencent meeting held 4 bn meetings in 2020, hit 200 mn users."

[6] W. Standaert, S. Muylle, and A. Basu, "How shall we meet? understanding the importance of meeting mode capabilities for different meeting objectives," vol. 58, no. 1, p. 103393.

[7] J. Bailenson, "Opinion | why zoom meetings can exhaust us."

[8] "MediaPipe face mesh."

[9] Y. Kubota, "Apple iPhone X Production Woe Sparked by Juliet and Her Romeo."

[10] T. Warren, "Apple's face id struggles detailed in new iphone x report," Oct 2017.

[11] J. García, A. Gardel, I. Bravo, J. L. Lázaro, M. Martínez, and D. Rodríguez, "Directional people counter based on head tracking," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 9, pp. 3991–4000, 2013.

[12] T. Helten, M. Muller, H.-P. Seidel, and C. Theobalt, "Real-time body tracking with one depth camera and inertial sensors," in *2013 IEEE International Conference on Computer Vision*, (Sydney, Australia), p. 1105–1112, IEEE, Dec 2013.

[13] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *CVPR 2011*, pp. 1937–1944, 2011.

[14] B. Garcia-Garcia, T. Bouwmans, and A. J. Rosales Silva, "Background subtraction in real applications: Challenges, current models and future directions," vol. 35, p. 100204.

[15] J. Kristoff, "The transmission control protocol."

[16] L. Dostalek, *Understanding TCP/IP a clear and comprehensive guide to TCP/IP protocols*. From technologies to solutions, Birmingham, U.K: Packt Pub., 1st edition ed., 2006.

[17] V. Powell, "Image kernels explained visually."

[18] T. Huang, W. Schreiber, and O. Tretiak, "Image processing," *Proceedings of the IEEE*, vol. 59, no. 11, pp. 1586–1609, 1971.

[19] P. H. King, "Digital image processing and analysis: Human and computer applications with cviptools, 2nd edition (umbaugh, s.; 2011) [book reviews]," *IEEE Pulse*, vol. 3, no. 4, pp. 84–85, 2012.

[20] "OpenCV: Hough Line Transform."

[21] "Document: Line Detection."

[22] "Python OpenCV - filter2d() function."

[23] "Wikipedia: Line detection," Jan. 2022. Page Version ID: 1064863210.

[24] B. Wang and S. Fan, "An improved canny edge detection algorithm," *2009 Second International Workshop on Computer Science and Engineering*, vol. 1, pp. 497–500, 2009.

[25] P. Bao, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1485–1490, 2005.

[26] "Canny edge detector," Jan. 2023. Page Version ID: 1136103505.

[27] "OpenCV: Canny edge detection."

[28] "Mediapipe face Detection."

[29] V. Holinko, I. Cheberiachko, H. Symanovych, and J. Kicki, "Designing the half-masks of filter respirators for workers of mining enterprises," *E3S Web of Conferences*, vol. 123, p. 01001, 01 2019.

6 Appendix. A: Conceptual Graph of 3D Display Mode

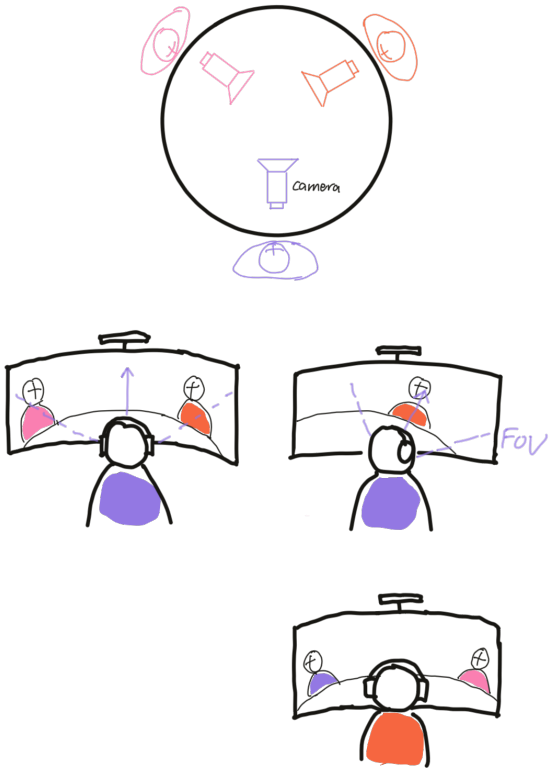

Figure 38: 3D Display Mode Conceptual Graph

7 Appendix. B: Github Repo

More details about the code and demo video can be found through this link:
https://github.com/tommywwz/virtual_conferencing_MQP.git