ACH1

Final Report of

Image Analyzer for People with Low Vision

by

Tong-yan CHAN Ka-ho LAU Joel Arvin BERAGO

Advised by

Prof. Albert C. S. CHUNG

Submitted in partial fulfillment

of the requirements for COMP 4991

in the

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

2015– 2016

Date of submission: 19th April, 2016

Abstract

The proposed image analyzer is a prototype wearable device system that aims to provide visually impaired people a system with navigational and social enhancement functions. The system has two parts, an image processing system and an output system. The image processing system utilizes a RGB-D camera to detect for path, stair, sign and face. Path direction is outputted using three vibration motors with 6 combinations while the detected stair, sign and face are outputted using audio speaker. The user can add their friends' face into the system so that the system will notify user any nearby friends.

Table of Contents

1 Introduction

1.1 Overview

A survey [1] reveals there are 174,800 visually impaired people in Hong Kong and 285 million people worldwide [2]. Visually impaired people require traveling aids like traditional white canes or guide dogs for navigation. However, these tools are labels of visually impaired people. A survey [3] and an interview [4] found that people who suffer from low vision often prefer not to use these tools as they feel embarrassed or vulnerable. There is a need for better travelling aids that can enhance their social life quality and keep guarantee on safety.

Various research projects, such as NAVI [5], and stereo camera systems [6], have attempted to use object recognition and computer vision to develop navigational aid systems for path finding and obstacle detection for the visually impaired. Although these systems can detect obstacles, their limitation is that their systems cannot recognize specific obstacles that require extra caution like stairs, or provide useful information like signs and known person.

This project extends previous research projects by presenting solutions to their limitations through implementation of additional functions to provide more navigational information to users. In addition to basic obstacle detection, stairs, sign and facial recognition will be applied so that supplementary navigational information can be conveyed to users by means of audio or haptic feedback.

This project aims to select and combine different algorithms with a reliable and efficient performance. Since one of the goals is to protect people from danger, it is not necessary to outline or recognize every single object. Rather, it is more important to detect, recognize and indicate obstacles and their locations correctly.

1.2 Objective

The main goal of this project is to assist the visually impaired in navigation. This is achieved by building a real-time and reliable system that will utilize the Microsoft Kinect, an RGB-D camera, and computer vision & image processing algorithms for detection and recognition of path, stairs, sign, and faces.

In order to achieve this goal, the following objectives are focused on:

1. Create a system interface to the Kinect camera.
2. Implement and integrate detection and recognition algorithms for path, stairs, signs and faces.
3. Build a feedback system using audio and haptic feedbacks.

Objective 1 is achieved by creating a portable battery-powered Kinect camera and the driver library and system development kit is used to retrieve color and depth frames for analysis and image processing.

Objective 2 is achieved by integrating the implemented detection modules into dedicated asynchronous threads.

- Path direction advice is generated from Ground Normal, and Histogram-shaped based Segmentation.
- Stair Detection is done by Canny Edge Detection, and Hough Line Transform.
- Sign Recognition is done by Conic Fitting, and OCR.
- Face Recognition is done by Haar-like Feature Cascade Classifiers, and Local Binary Pattern.

Input color and depth frames are preprocessed to provide significant data for analysis and speed up processing time.

Objective 3 is achieved by having a having a dedicated asynchronous thread for processing and outputting notifications to the user. Output notifications are made by text-to-speech of detected stairs, people, and signs, while vibration is encoded for path advice.

1.3 Literature Survey

1.3.1 Finding Ground and Obstacle

For detecting obstacle, the process needs to remove noise and background first and do segmentation. After removing the floor segment, other segments are defined as obstacle until further classification. This process needs fast but only fair quality segmentation since speed of detecting obstacle is more important. The segmentation method used in [7] is suitable for this requirement. Its performance in [7] gave 73.1+/13% correct segmentation rate and used 0.8s of processing time. Assume user walking slowly and carefully, its performance can handle usual situation to notify user to avoid collide with obstacles. On the other hand, segmentation can save the time for classification by processing each segmented region with prior knowledge whether than scanning the whole image.

The proposed segmentation method in [7] involved two parts.

Part 1: Delaunay triangulation is applied on the coordinates on the horizontal plane. Calculating cross product of the vertices of the output triangle, can find the normal vector. For each Normal vector of floor is segmented if the vertical-component of the normal vector exceeds a threshold which is found in experiment.

Part 2: A depth histogram is generated for the remaining image pixels. Image pixels are segmented by using multiple distance thresholds which are the local minima of the histogram.

1.3.2 Stair Detection

Stair detection can be done by identifying the characteristics of stairs then applying corresponding algorithms to detect the characteristics. Stairs are comprised of a sequence of steps, which can be considered as a continuous set of parallel edges. Edge detection, one of the fundamental approaches to object recognition, can applied to recognize stairs.

Wang [8] describes an algorithm which then utilizes Hough transform to detect straight lines from edges points found by an edge detection. Lines can be represented in the standard polar coordinate form

$$r = ysin\theta + xcos\theta$$

A set of stairs or a set of parallel lines, can be easily detected as the angle θ should be similar. The orientation of the detected stairs should be perpendicular to the parallel lines, with a feature line indicating the middle of the stairs found by taking the middle point of the longest line of the parallel lines.

The algorithm then utilizes depth information to differentiate stairs from objects with straight parallel lines, such as stairs. A two tier hierarchical SVM-based classifier is first fused to classify pedestrian crosswalks from stairs, then another is used to classify upstairs from downstairs. However, the classifier will not be used in this project as it is unnecessarily complex. Instead, a simpler algorithm based on observations of the differences of upstairs and downstairs will be used. Based on the graph results of plotting the depth of points along a line across the stairs, the graph exhibits an ascending zigzag pattern which can be calculated and interpreted as stairs.

The overall approach of the proposed implementation will be satisfactory for the needs of this project. Although the proposed method is restricted to the quality of the depth and RGB images and that it fails to detect stairs with less than 3 steps, this can be overcome by reconfiguring parameters to support detecting smaller stairs.

1.3.3 Conic fitting in Sign Recognition

Conic fitting is an algorithm to find a best-fit conical curve of a given set of points, an ellipse or a circle. This algorithm was proposed by Fitzgibbon [9]. He described the algorithm as follows.

For a given set of 2D data points,

$$P = \{\boldsymbol{x}_i\}_{i=1}^{n}, \text{ where } \boldsymbol{x}_i = (x_i, y_i),$$

the general equation for conical sections, $F(\boldsymbol{a}; \boldsymbol{x})$, is,

$$F(\boldsymbol{a}; \boldsymbol{x}) = A_{xx}x^2 + A_{xy}xy + A_{yy}y^2 + A_x x + A_y y + A_0,$$

which can be written as the dot product of $\boldsymbol{x}_i$ and the vector of parameters $A_\alpha, \boldsymbol{a}$,

$$F(\boldsymbol{a}; \boldsymbol{x_i}) = [x_i{}^2 \quad x_i y_i \quad y_i{}^2 \quad x_i \quad y_i \quad 1] \cdot [A_{xx} \quad A_{xy} \quad A_{yy} \quad A_x \quad A_y \quad A_0]$$

$$F(\boldsymbol{a}; \boldsymbol{x_i}) = \boldsymbol{x_i} \cdot \boldsymbol{a}$$

A conical curve, $C(\boldsymbol{a})$, is then represented implicitly as,

$$C(\boldsymbol{a}) = \{\boldsymbol{x} | F(\boldsymbol{a}; \boldsymbol{x}) = 0\},$$

and a distance metric of the distance from a point, $\boldsymbol{x}_i$, to the curve $C(\boldsymbol{a})$, is defined as

$$\delta(C(\boldsymbol{a}), \boldsymbol{x}_i) \ ,$$

such that the error function,

$$\epsilon^2 = \textstyle\sum_{i=1}^{n} \delta(C(\boldsymbol{a}), \boldsymbol{x}_i) \ ,$$

attains its global minimum at $\boldsymbol{a}_{min}$ and the best-fit conical curve is represented by $C(\boldsymbol{a}_{min})$.

1.3.4 Local Binary Patterns in Face Recognition

Local Binary Patterns (LBP) is one of a common algorithm used in face recognition. It is a means to extract LBP histogram for description of texture features in greyscale. According to [10], the pixels of an image are labelled by thresholding the neighborhoods (P, R) of each pixel, $p_{(x_i,\ y_k)}$, and the operator considers the result as a binary number, where P is the number of sampling points on a circle with radius R and centered at $p_{(x_i,\ y_k)}$. An example with neighborhood (8, 1) is illustrated in Figure 1 & Figure 2.

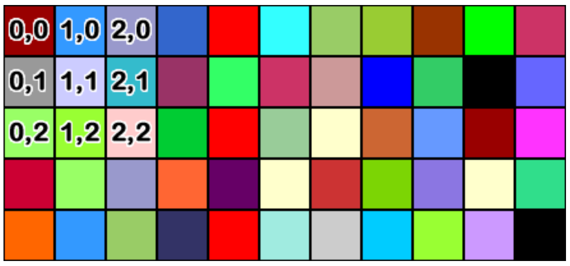


Figure 1: A sample image

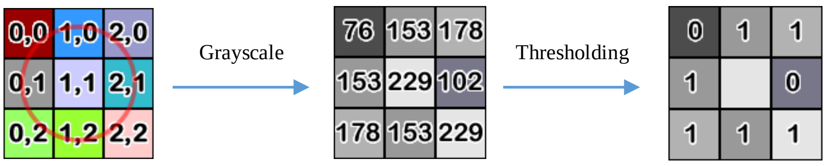


Figure 2: Flow of LBP generation

LBP is defined as,

$$\mathrm{LBP} = \sum_{i=0}^{7} d_i \times 2^i$$

In this case, LBP for pixel $[1,1] = 01101111_2 = 111$ and this pixel is labelled by 111. If a sampling point is not at the center of a pixel, its pixel value is bilinearly interpolated. To label the whole image of the texture, the same steps should be repeated for all pixels. By storing the count of each LBP, says 111 in the example, a LBP histogram of the labeled image is generated. Hence, a texture can be represent by a k-dimensional vector, where k is the number of different features or labels of the texture computed, and it can be regarded as a model of the texture, i.e. a model of the face in this case.

With a set of models of faces from the same person, a texture class is formed so that each person has his/her own texture class in database. To recognize a face, the model of the face is compared to the database to find out the nearest models. Within these models, the class that the majority falls is consider as the result with a reference of distance, which is a measure of confidence belonging to the reference class.

2 Design

There are three main parts in our system: the system interface, integrated detection and recognition algorithms, and the feedback system.

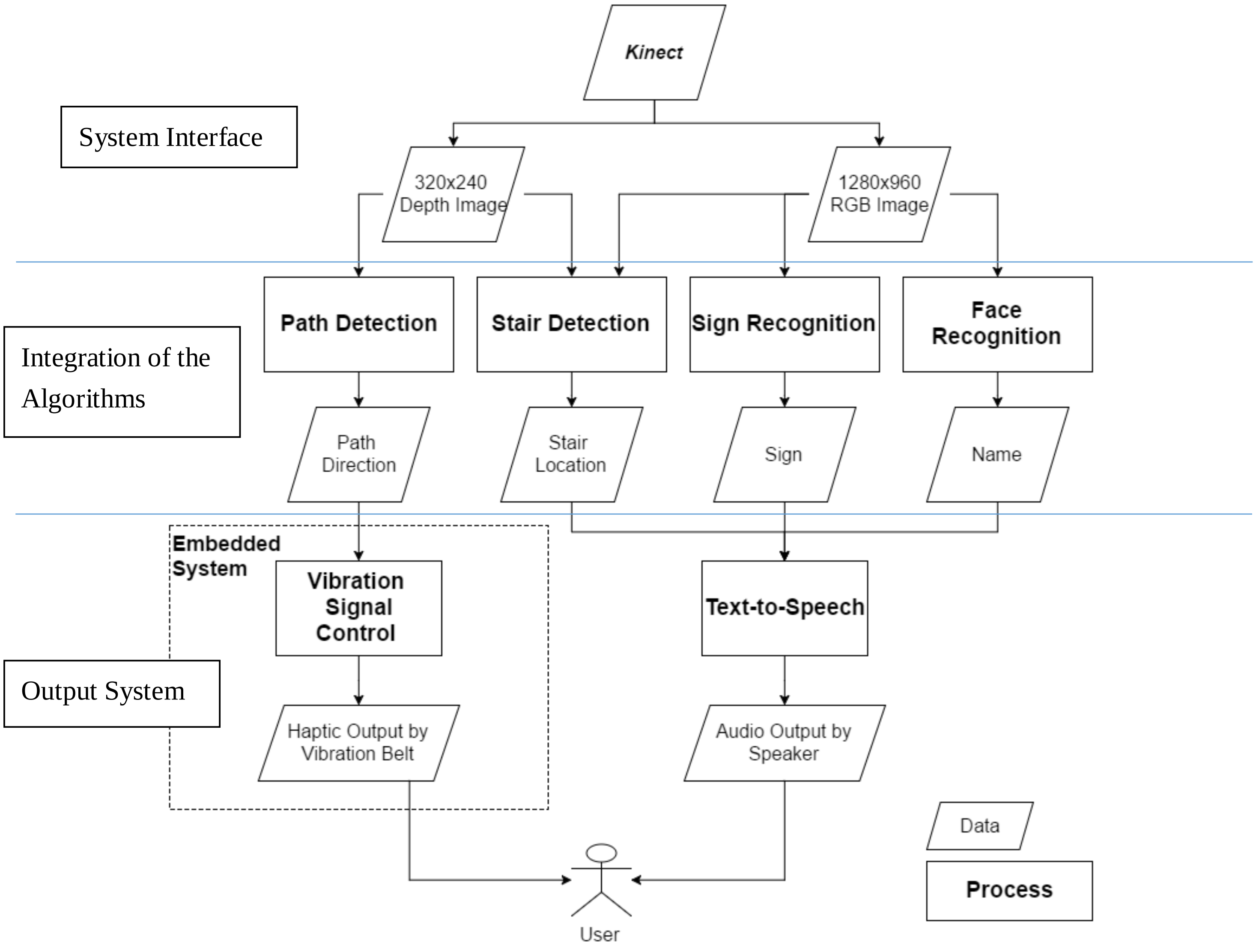


Figure 3. System flow chart using Input-Process-Output model

2.1 System Design

2.1.1 System interface

The system uses the OpenNI driver library to obtain the Depth and RGB image from the Kinect for use in the detection and recognition algorithms.

2.1.2 Integration of the algorithms

As the system needs to run in real-time, the system modules are integrated together by multithreading. The individual modules run in their own asynchronous threads to maximize processor usage, and greatly improve total system runtime by preventing individual system modules from slowing down the entire system.

2.1.3 Output system

The hardware system includes the Microsoft Kinect for Xbox, a laptop, vibration belt and battery-powered power supply with voltage regulator and voltage meter in circuit. There will be embedded system for handling the control signal from laptop for controlling the on/off of vibration motors on the belt. The Kinect and vibration belt will be fastened around the waist on the body of user in order to achieved best view in both sign, human face and more ground in the same time.

2.2 Path Detection

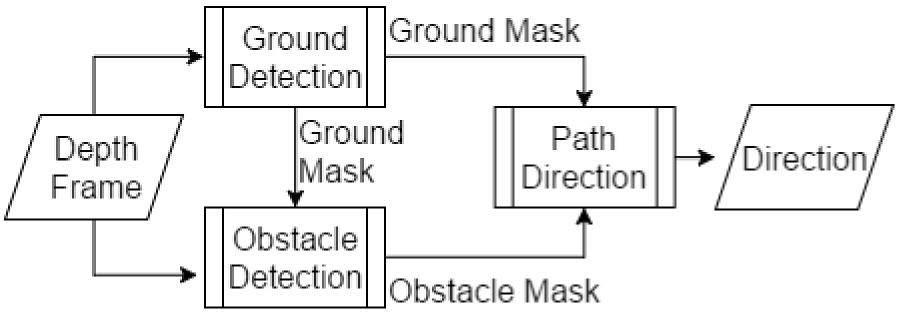


Figure 4. Data flow in Path Detection

2.2.1 Ground Detection

As mentioned in [7], the direction of normal could be used to identify the ground region. The depth image was partitioned into many regular square planes. Two vectors were picked inside the plane and did cross product to find the normal vector. Assuming the pixels in the plane belong to the same plane if the plane area was small enough practically. False positive of the plane normal would be eliminated by calculating the height of the plane. 16-bit depth image was used for calculation in order to preserve the distance information. To save computation cost, only the lower part of image was processed. A Boolean matrix was used to mark down the position of square plane which belong to ground. Morphological operations were carried on to remove false negative. A ground mask would be draw according to the boundary and values in Boolean Matrix.

2.2.2 Obstacle Detection

8-bit image was used for segmentation for faster processing speed and less noisy. The found Ground region will be removed in the depth image before calculating pixels' histogram for obstacle detection. It was assumed that the pixel values within two local minima in the histogram could be regarded as one segment. Then the pixels were labelled in the depth image according to the intervals, the pixels with the same label were regarded as one segment. Stripes masks were used to divide the depth into two depth images to avoid too much bounding areas in the later part. Each segment will be analyzed one by one to find the obstacle using find contour. Bounding areas of the obstacles were calculated by using convex hull. The obstacle mask was constructed by drawing all the filled convex hull into one image.

2.2.3 Path Direction

The refined Ground region was obtained by subtracting obstacle mask with ground mask and it should be safe enough to estimate the path for walking. By assumption, ground region with longer extension should able to keep the user walking further and less likely to have collision. In order not to disturbed by the noise in the edge, it is proposed to use the center of mass of the ground region as the indicator of the path direction. However, since the proposed system cannot control the body movement of the user, only rough direction (Left, Left-to-Center, Center, Right-to-Center, Right and No Path) of the path direction will be output to user.

2.3 Stair Detection

Stairs Detection is done in two main parts: extracting the parallel edge lines, then detecting the stairs by depth features.

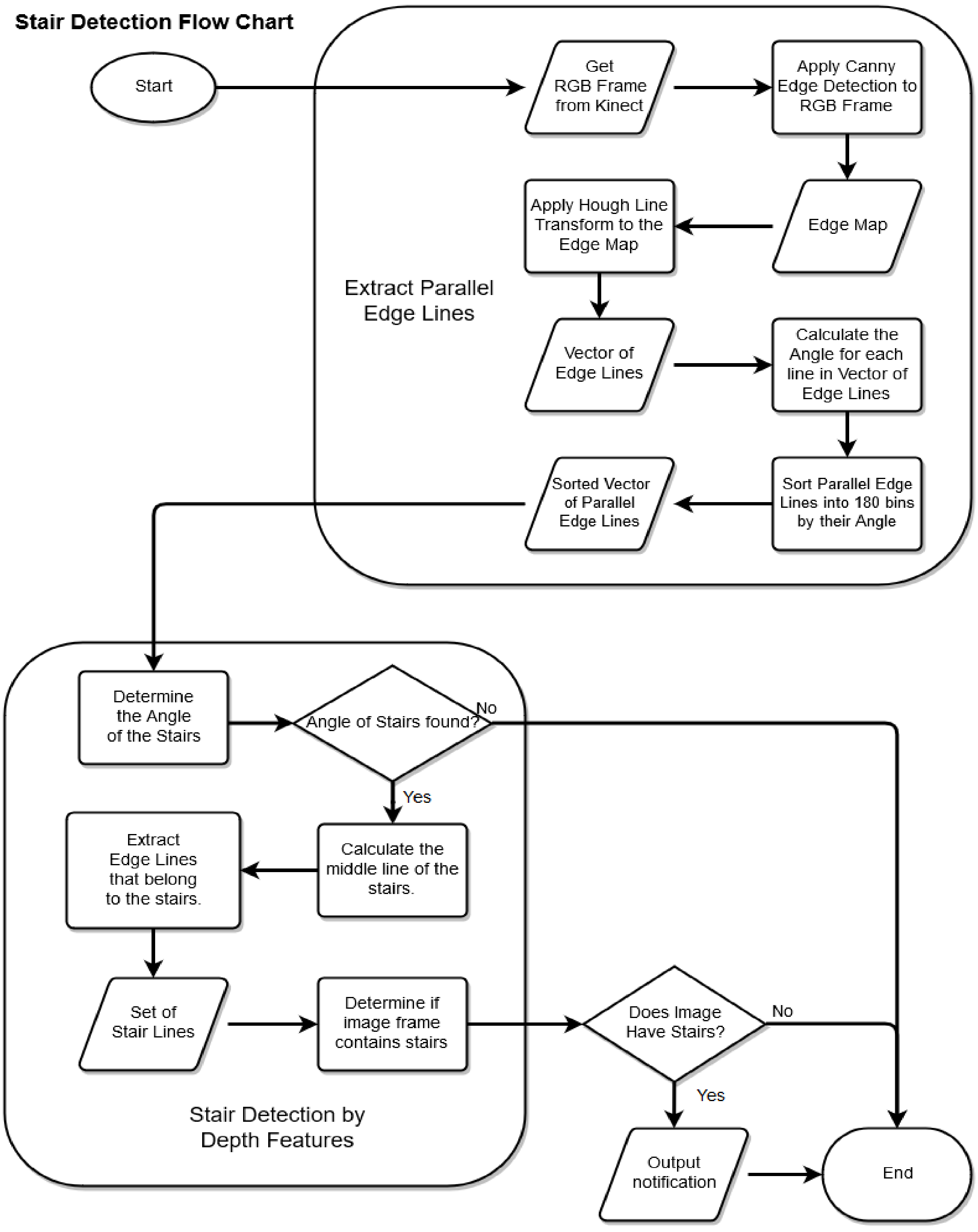

Stair Detection Flow Chart
Start
Get RGB Frame from Kinect
Apply Canny Edge Detection to RGB Frame
Edge Map
Apply Hough Line Transform to the Edge Map
Extract Parallel Edge Lines
Vector of Edge Lines
Calculate the Angle for each line in Vector of Edge Lines
Sort Parallel Edge Lines into 180 bins by their Angle
Sorted Vector of Parallel Edge Lines
Determine the Angle of the Stairs
Angle of Stairs found?
No
Yes
Calculate the middle line of the stairs.
Extract Edge Lines that belong to the stairs.
Set of Stair Lines
Determine if image frame contains stairs
Does Image Have Stairs?
No
Yes
Stair Detection by Depth Features
Output notification
End

Figure 5 Stair Detection Flow Chart

2.3.1.1 Extract Parallel Edge Lines

A common approach to detecting stairs is by detecting the edge feature of stairs. Stairs are comprised of many steps, which can be regarded as many parallel edges. Edge detection algorithms are applied to a color image to obtain the stair edges points. Straight line feature extraction is then applied to extract which edge points are straight lines. This system will employ Canny Edge Detection and Hough Line Transform to extract the edge points and straight lines respectively.

2.3.1.2 Detect Stairs by Depth Features

After the parallel edge lines have been extracted, there is sufficient information to imply that there could be stairs in front of the user. Information about the stairs can be obtained, such as:

- Stair angle
- Stair direction
- Which lines belong to the set of stair lines.

However, detecting edge features is insufficient to accurately detect stairs as there are other objects, such as zebra crossings, which are also made up of many parallel edges. Another unique feature of stairs is its structure which is a zigzag pattern made from the step treads and step risers. This feature can be prominently seen from a depth image of the stairs, as seen in , and the algorithm will be based off the features of the .

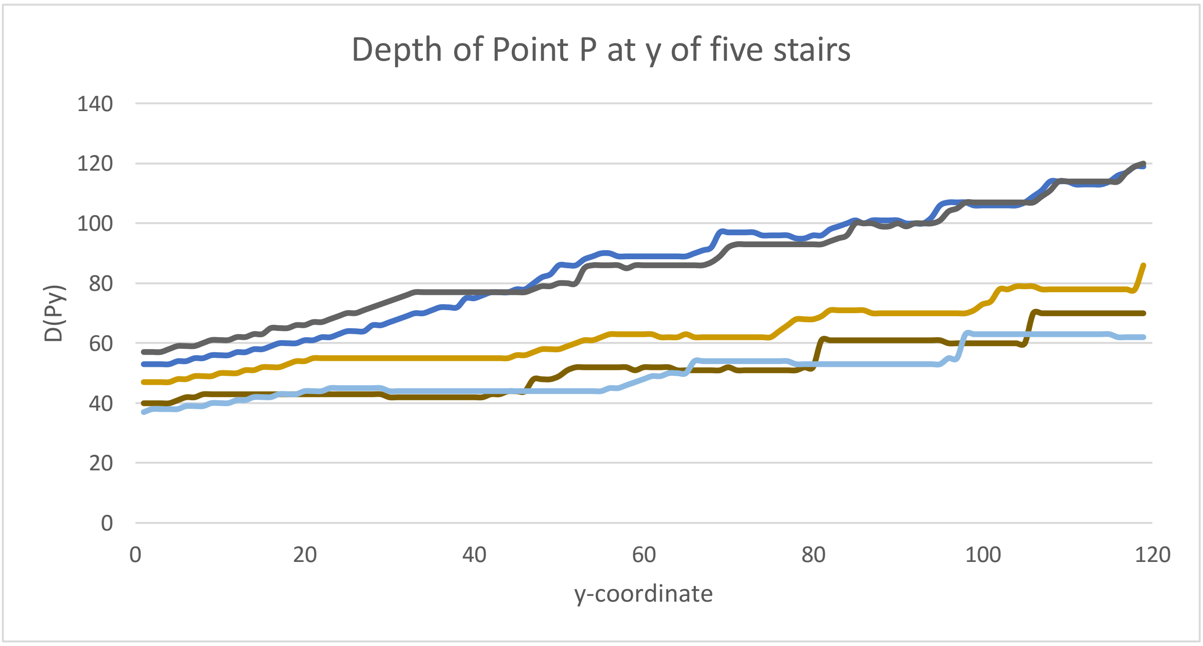


Figure 6 plots the depth values across 5 stairs. The chart shows the ascending trend and zigzag pattern caused by the stair steps and risers.

2.4 Sign Recognition

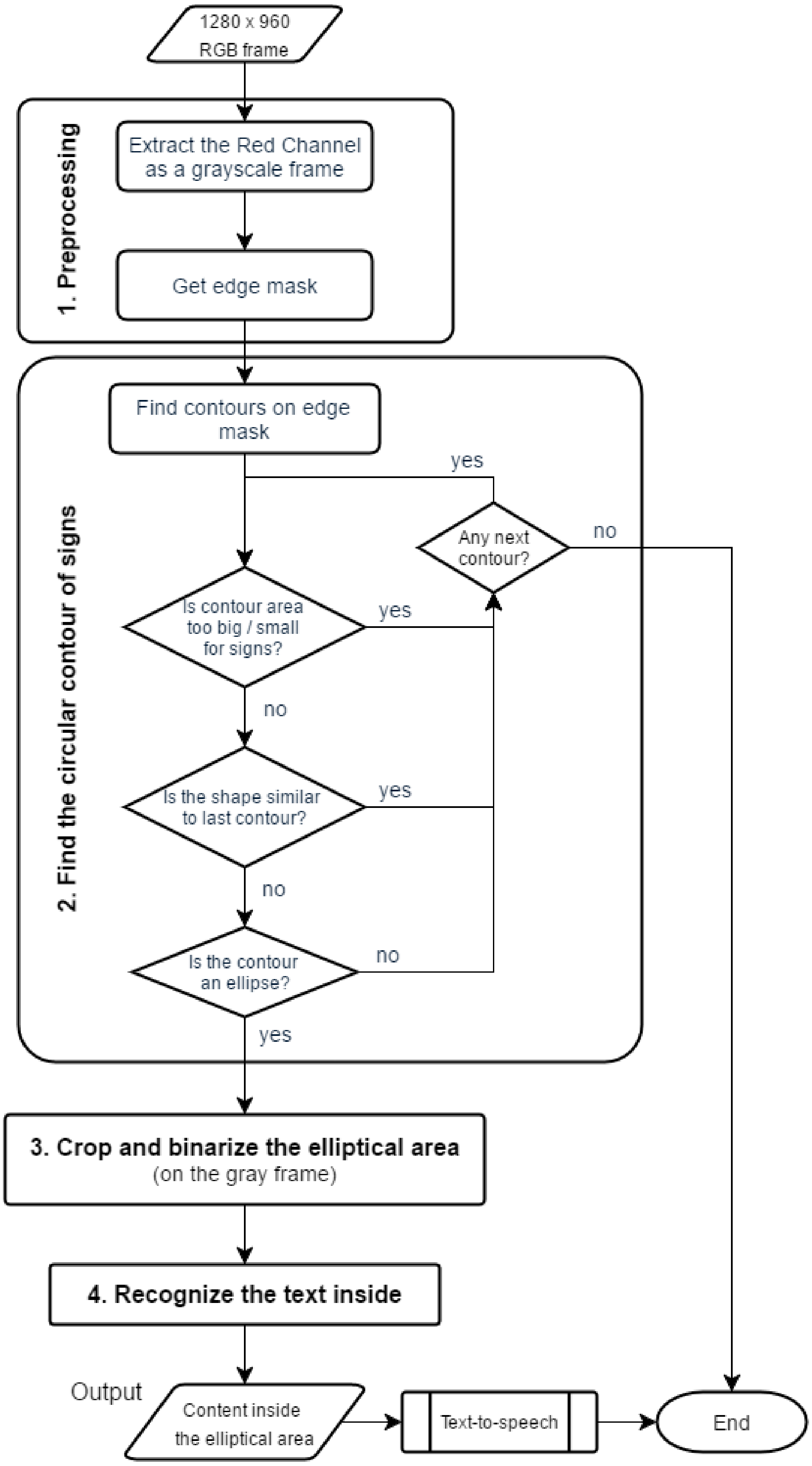

1280 x 960
RGB frame
1. Preprocessing
Extract the Red Channel as a grayscale frame
Get edge mask
2. Find the circular contour of signs
Find contours on edge mask
yes
Any next contour?
no
Is contour area too big / small for signs?
yes
no
Is the shape similar to last contour?
yes
no
Is the contour an ellipse?
no
yes
3. Crop and binarize the elliptical area (on the gray frame)
4. Recognize the text inside
Output
Content inside the elliptical area
Text-to-speech
End

Figure 7: flowchart of sign recognition

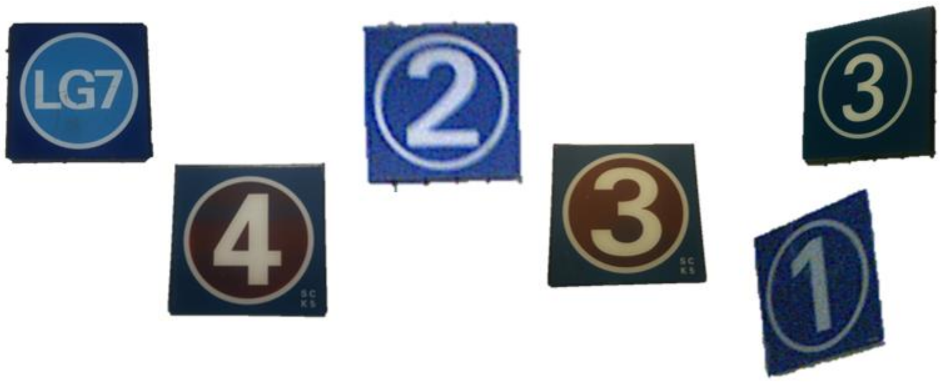


Figure 8: typical signs in HKUST

Merely colour frames were processed during the recognition process, as signs do not contain spacial features but are only flat plates or boards. As shown in Figure 8, it was observed that typical signs in HKUST were of the same structure and contained a common feature which was consisting of some characters inside a circle. Therefore, structural analysis was chosen to recognize signs and the system achieved it by recognizing the text inside a circle after obtaining the position of a sign in a frame. The algorithm involved the following 4 main steps, while Figure 7 was a more detailed flow chart, and illustrated in Figure 9.

1. Colour frame pre-processing
2. Find the elliptical contour of a sign
3. Crop and binarize the elliptical area if there is any ellipse found
4. Recognize the text inside the cropped area

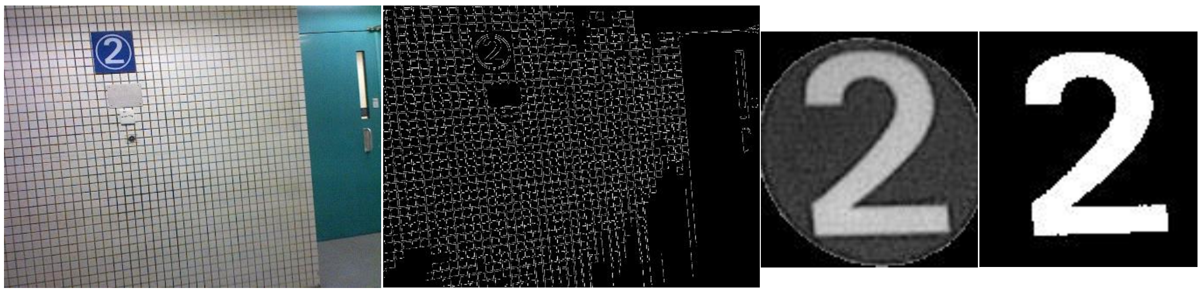


Figure 9: illustration of the sign recognition process

2.5 Face Detection and Recognition

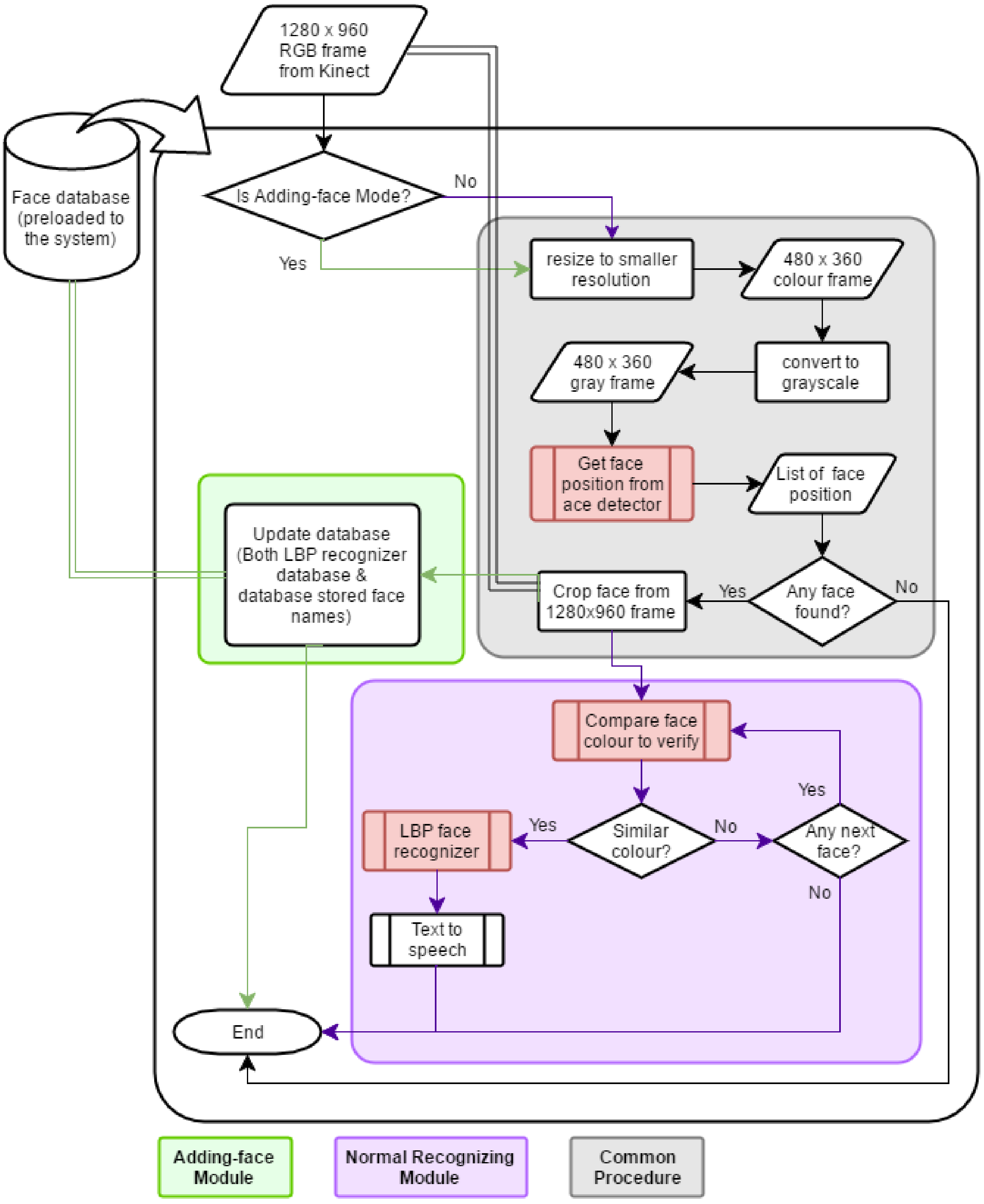

1280 x 960 RGB frame from Kinect
Face database (preloaded to the system)
Is Adding-face Mode?
No
Yes
resize to smaller resolution
480 x 360 colour frame
convert to grayscale
480 x 360 gray frame
Get face position from ace detector
List of face position
Any face found?
Yes
No
Crop face from 1280x960 frame
Update database (Both LBP recognizer database & database stored face names)
Compare face colour to verify
Similar colour?
Yes
No
Any next face?
Yes
No
LBP face recognizer
Text to speech
End
Adding-face Module
Normal Recognizing Module
Common Procedure

Figure 10: Flowchart of face recognition

Same as sign recognition, merely colour frames are processed during the recognition process. To recognize faces, it is better to locate where the faces are first, instead of passing and comparing the entire image because computers might consider the whole image as a face without removing the background, reducing the accuracy and efficiency of face recognition.

Therefore, the system needs to find out if there is any face on the frame and where it is so face detector is applied first, followed by face recognizer. This can enhance the overall accuracy and efficiency. A complete flow chart is shown in Figure 10.

2.5.1 Preload Database

There were 3 files needed to be preloaded – the file stored human face features, the file stored face features of recognized people, and the file stored the corresponding name of each recognized person. One reason is that human face features and face features of recognized people were needed before face detection or face recognition carried out. In addition, as recognized people were represented by numbers called 'label' in the file stored their face features, another file need to store their names so as to give feedback to users. In this file, each line recorded the label and the name of 1 person, with the format designed as:

label,string of name

The file was named as "***names.csv***".

2.5.2 Adding-face Mode

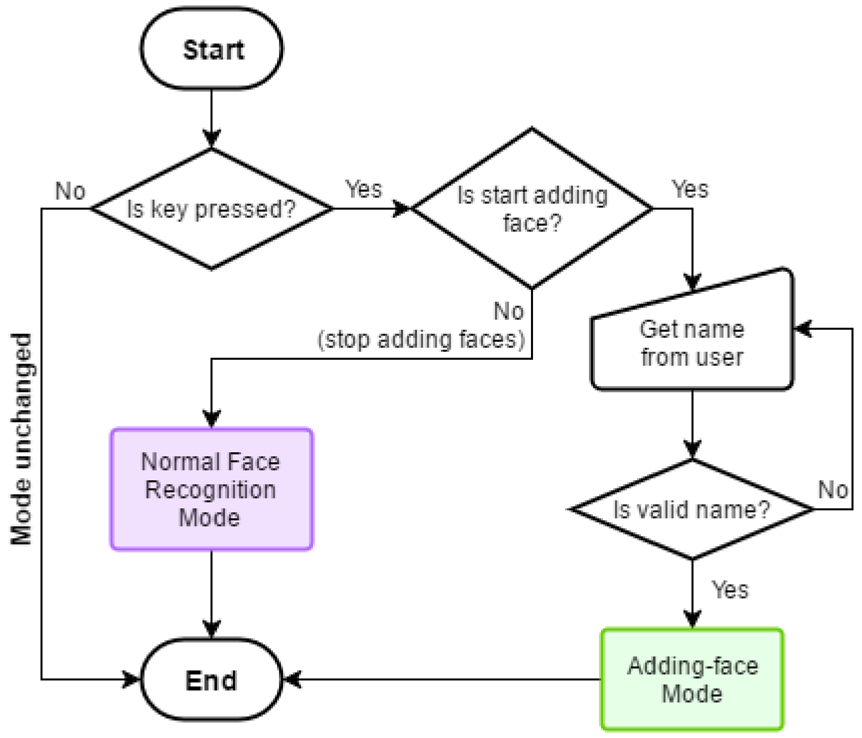


Figure 11: Flowchart of Adding-face Mode

This mode allowed users to add faces of new friends to the system and enhance the recognition prediction of their friends. This mode required face detector to determine face positions, followed by cropping the face and passing the cropped face to update the LBP face recognizer and the database stored face names. Hence, this mode only re-used the implemented face detector and the implemented face recognizer, together with some logic to allow users to switch between normal recognizing mode and adding-face mode. The logic flow is shown in Figure 11.

2.5.3 Face Detection

Faces could be detected by apply a classifier with a database containing the features of human faces directly. However, there might be false positive while detecting face so verification of the detected face-like objects is need. It is observed that the colour of false positive is usually different from human face colour so average face colour was chosen to be a parameter for verification, which eliminates false positive by comparing the colour of detected face-like objects to the pre-calculated average face colour. If they are in similar colour, the detected face-like object is regarded as a positive sample of face and proceed to the next step. Otherwise, it is dropped.

2.5.4 Face Recognition

Faces of difference people contain different sets of features, says LBP. By differentiating these sets of features, faces could be recognized.

After training the LBP recognizer as mentioned previously, the result could be extracted and pre-loaded with the person name of corresponding label stored in "***names.csv***" to the real-time system when it is turned on. The detected faces cropped could then be passed to the recognizer and matched the nearest faces after face detection.

3 Implementation

3.1 System Implementation

3.1.1 Integration of the algorithms

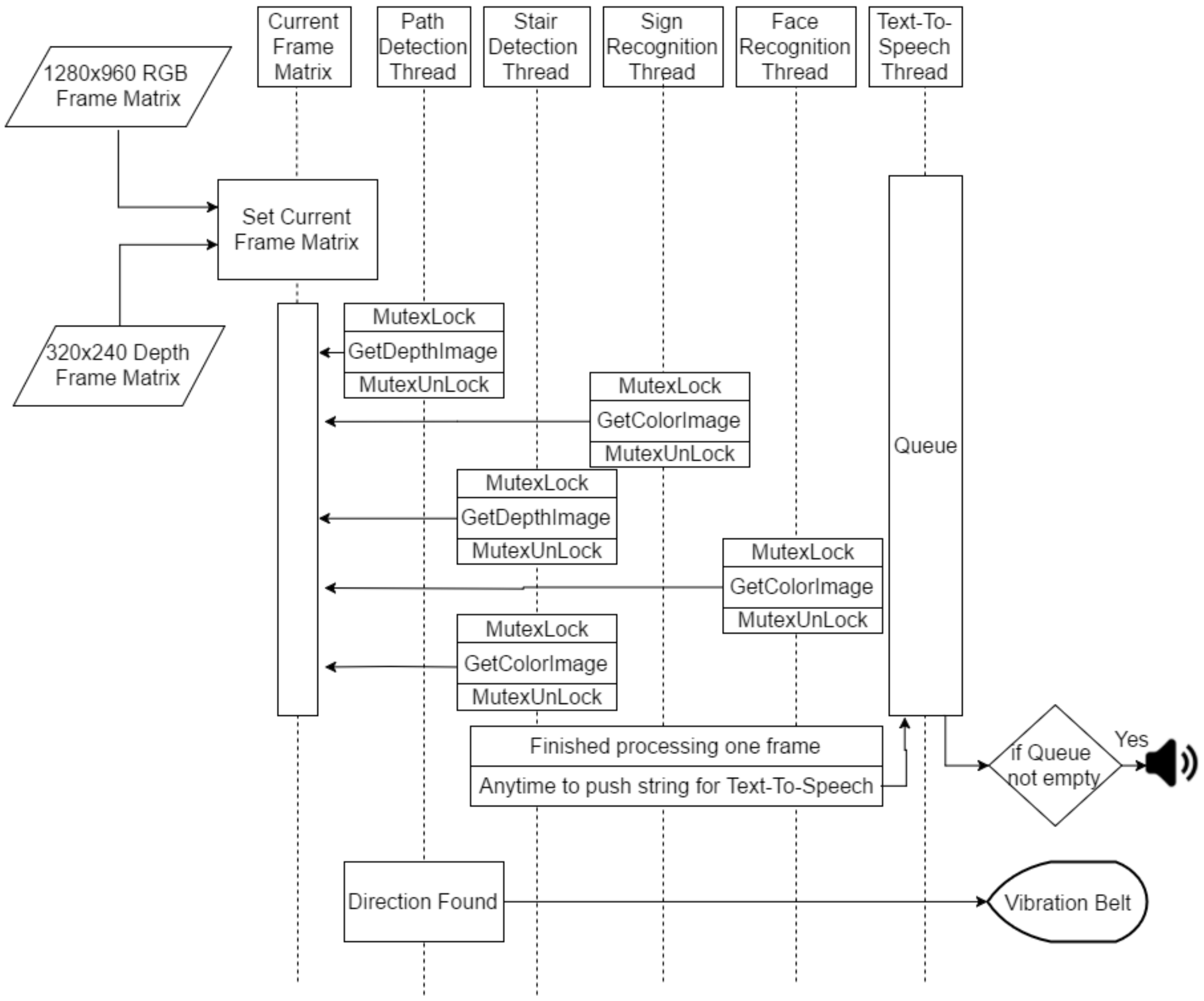


Figure 12. Multithreading process flow chart of a single frame

The Kinect, detection modules, and output modules ran in their own threads by using the C++11 asynchronous multithreading library. Each thread contained an infinite while loop that executed their required functions. Mutexes from multithread library were used when the colour and depth frames between the Kinect thread and detection modules were required to access.

3.1.2 Hardware System

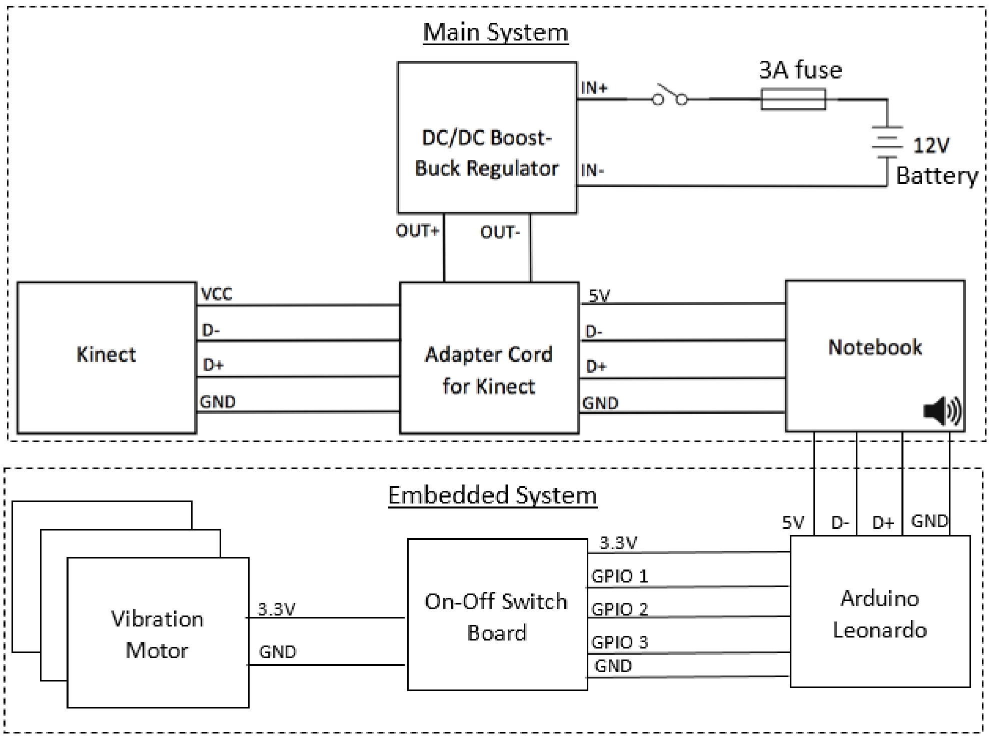


Figure 13. Schematic of the main system and embedded system

For main system, the image processing algorithm was running on the notebook. The Kinect communicated with the Notebook via USB and powered by the 12V battery. The voltage regulator circuit regulated and ensured the input voltage from 3x 3.7V cells Lithium Polymer batteries to an output voltage of 12V. A voltage meter was added in order for developer to know the remaining charge in the battery. The speaker was embedded in notebook.

Arduino board was the control unit of the embedded system and communicated with Notebook via USB. The C program use polling to passively receive the signal (1 to 5) from the notebook and then set/reset the GPIO1~3 according to the switch case. The GPIOs were connected to the base of the transistor switches on the on/off switch board. Setting the GPIO will drive the transistor into saturation and then close the motor circuit. The circuit schematic and real circuit board was put in Appendix E: On-Off switch Circuit Board

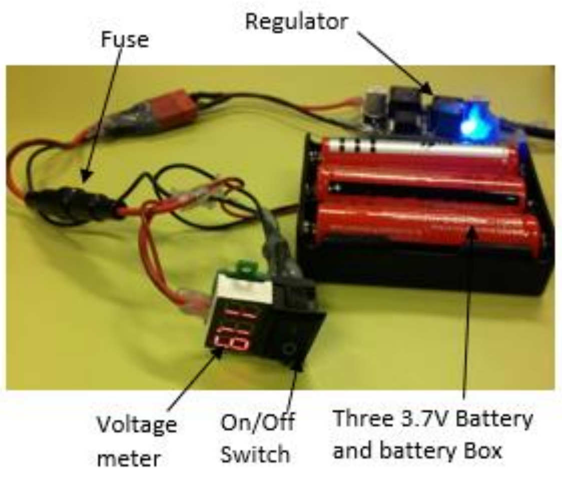


Figure 14. Portable power supply for Kinect

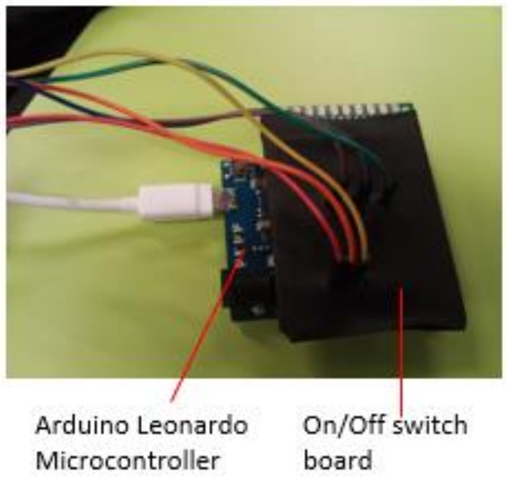


Figure 15. The embedded system hardware for vibration motor control

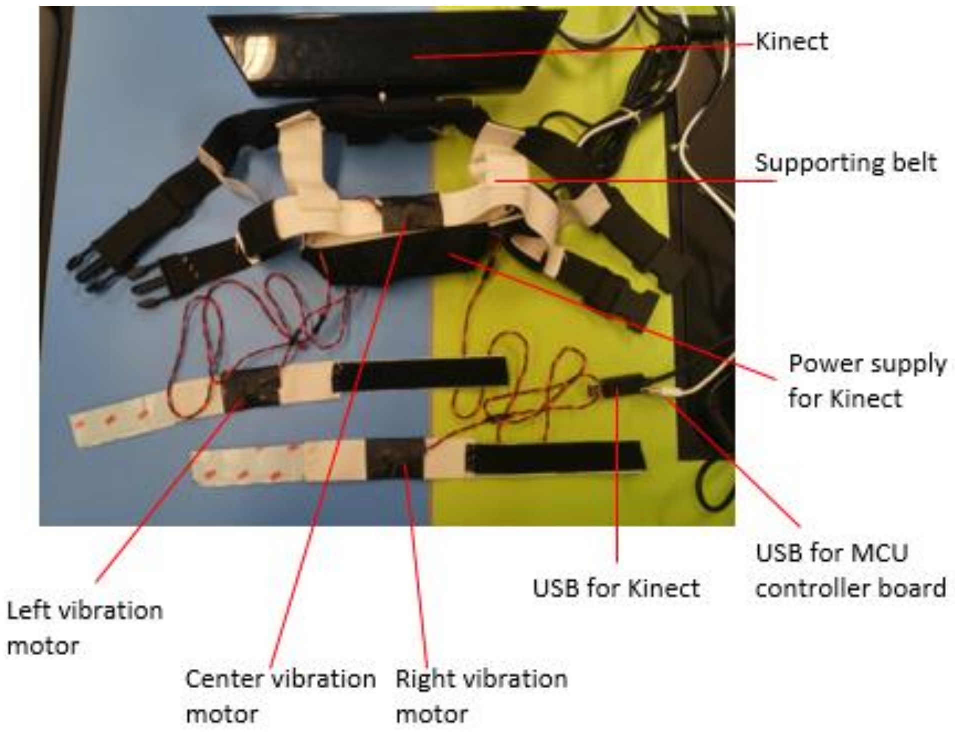


Figure 16. All hardware setting on the supporting belt except notebook

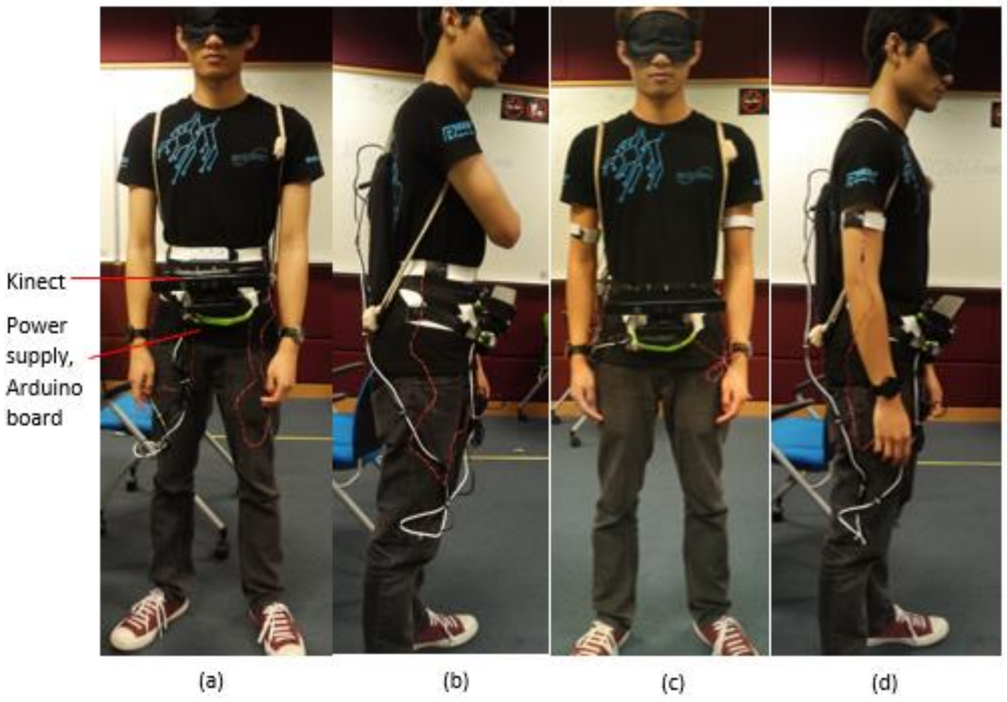


Figure 17. The actual installation of proposed system

The left and right vibration motor could be fasten around waist(Figure 17a&b)or arm(Figure 17c&d) which was depending on the user’s preference since some people could not differentiate which motor was vibrating. The notebook was put in the back pack.

3.2 Path Detection

Only 320 x 240 depth image was used in this section. 5 videos were collected for testing the program. The route of path of Video 1 and Video 2 were shown on Appendix H: Video 1 Sample path in HKUST for testing. and Appendix I: Video 2 Sample path in HKUST for testing.. Other 3 videos were ground only video and used for finding experimental threshold.

3.2.1 Ground Detection

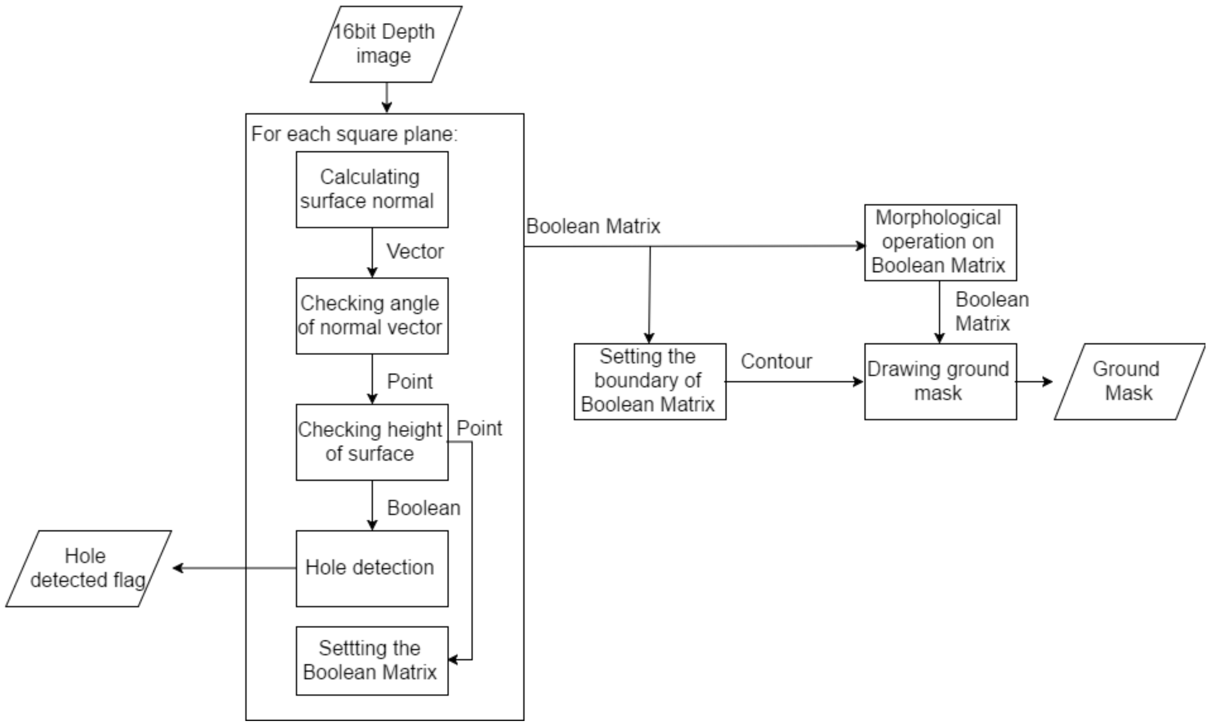

16bit Depth image
For each square plane:
Calculating surface normal
Vector
Checking angle of normal vector
Point
Checking height of surface
Point
Boolean
Hole detection
Hole detected flag
Settting the Boolean Matrix
Boolean Matrix
Morphological operation on Boolean Matrix
Boolean Matrix
Setting the boundary of Boolean Matrix
Contour
Drawing ground mask
Ground Mask

Figure 18. Flow chart of Ground detection

Calculating surface normal

Since image was 2D and the pixels contained the distance value generated by Kinect, each pixel could be represented in a 3D space as Figure 19: Space model of the image. to be point $\mathcal{P}(x, y, z)$. At least three pixels were required to construct two vectors by equation (1) and as shown in Figure 21: vectors in a square plane. The resulting vector of the cross product of the vectors selected within the same square was the normal of the square plane.

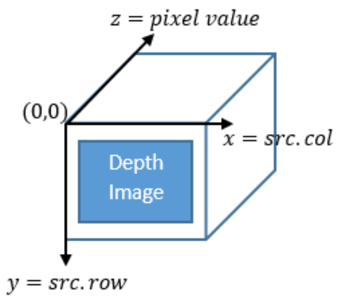


Figure 19: Space model of the image.

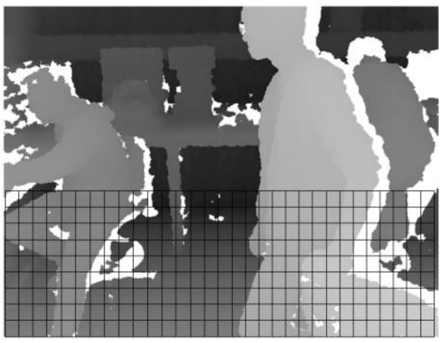


Figure 20: Portioning the lower half of the image.

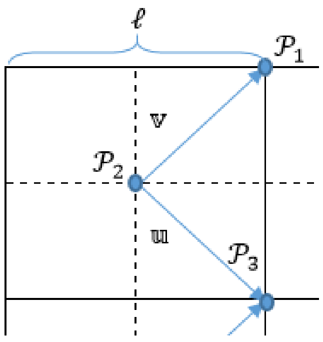


Figure 21: vectors in a square plane

$$\mathbb{v} = \mathcal{P}_1 - \mathcal{P}_2 = (v_x, v_y, v_z) \quad and \quad \mathbb{u} = \mathcal{P}_2 - \mathcal{P}_3 = (u_x, u_y, u_z) \tag{1}$$

$$\mathbb{w} = \mathbb{u} \times \mathbb{v} = \begin{vmatrix} u_y & u_z \\ v_y & v_z \end{vmatrix} i + \begin{vmatrix} u_x & u_z \\ v_x & v_z \end{vmatrix} j + \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} k = (n_x, n_y, n_z) \tag{2}$$

Checking angle of normal vector

The direction of the vector was calculated by using

$$\theta = \tan^{-1}\left(\frac{\sqrt{{n_x}^2 + {n_z}^2}}{|n_y|}\right) \tag{3}$$

$\mathbb{w}$ has $n_y > 0$ was filtered before calculating the θ to save computation effort. Three ground only videos were used to find the threshold of θ. Statistics of all θ in radian were collected as shown in Appendix C.I. Since variant increased after smoothing with Gaussian Blur, no smoothing was applied before this algorithm. The decided range of θ is 0.01 to 0.5.

It was possible to have both non-ground pixel and ground pixel in the same square plane area. However, it was an acceptable way to calculate the surface normal since only the accuracy of boundary of the ground region would be lost sometime without loss of the major ground region. The calculation was fast and in constant time. The result was acceptable as shown in Figure 22. The color image and the result of the found normal vectors for ground region

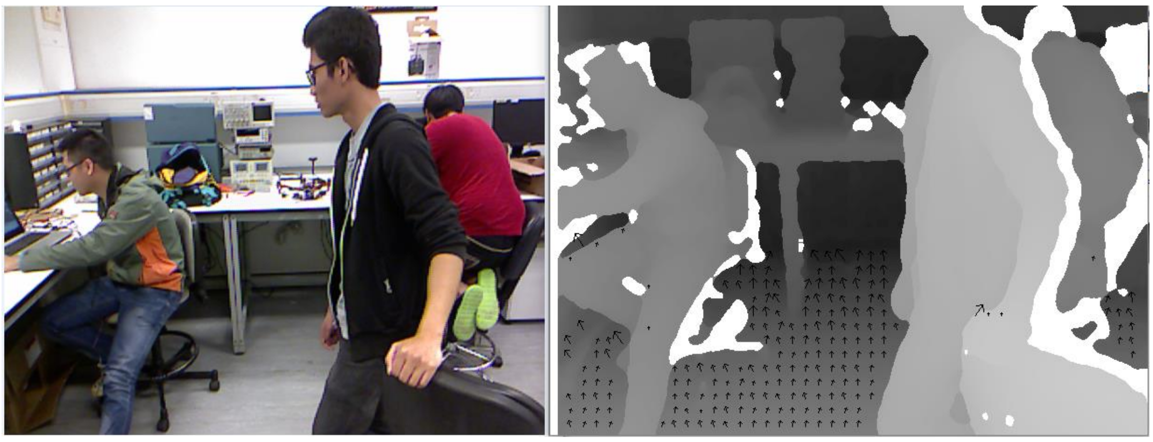
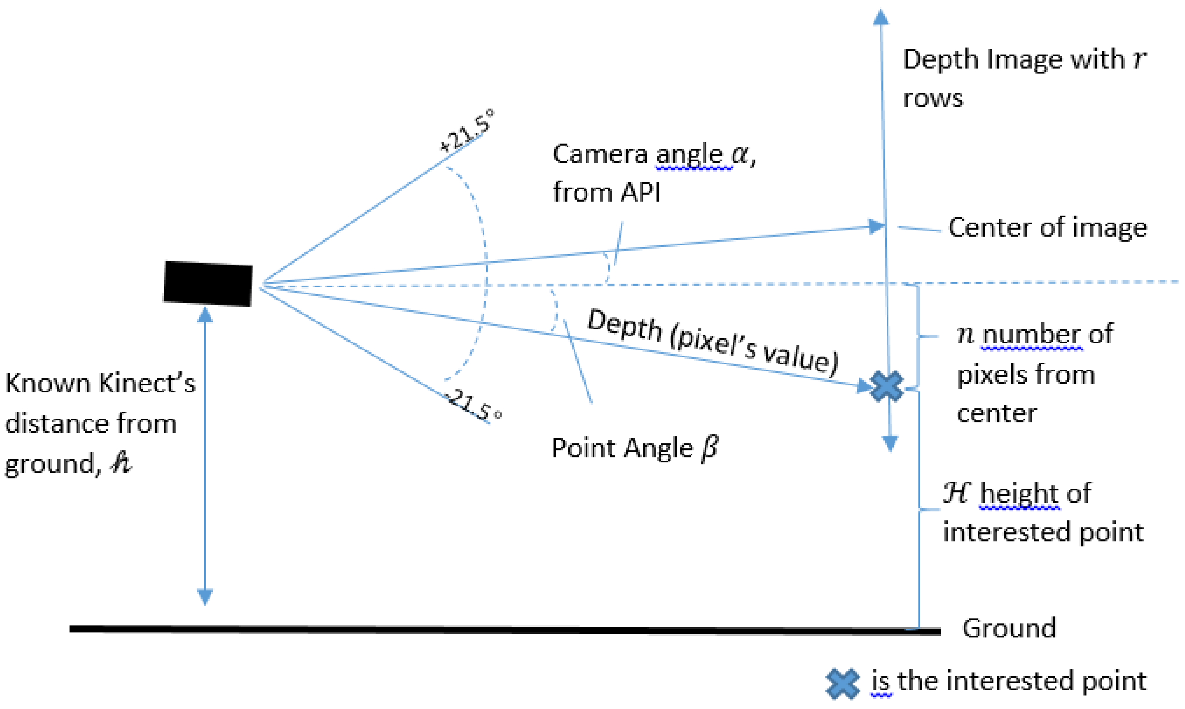


Figure 22. The color image and the result of the found normal vectors for ground region

Checking height of surface

By comparing the height $\mathcal{H}$ of the pixels from the ground to a threshold, which was found by experiment, the false positive of square planes found in previous section could be filtered.



Depth Image with r rows
+21.5°
Camera angle α, from API
Center of image
Depth (pixel's value)
n number of pixels from center
Known Kinect's distance from ground, $\hbar$
-21.5°
Point Angle β
$\mathcal{H}$ height of interested point
Ground
is the interested point

Figure 23. The geometry of calculating pixel height

$$\beta = \alpha + \frac{n}{r} * 43°$$

$$\mathcal{H} = \hbar + depth * \sin\beta$$

β was calculated by assuming the β was directly proportional to n. However, the assumption was not true that β for the point far away from center was not a constant ratio relationship with n. The calculated height was getting a static error for non-ground surface normal. Three ground only videos were used to find the threshold of height. Statistics of all height in mm were collected as shown in Appendix F: Statistics about ground pixel. The decided threshold of the height was 260mm.

The ground detection algorithm was tested under Video 2 and result was shown in . The frame was count as success only if only ground was detected and the detected area was the major ground area.

Plane edge(ℓ)	Accuracy

8	48.8%
12	64.4%

Table 1

Hole detection

Height checking was also used for detecting hole. If the square plane had height smaller than a threshold h, it was count as part of hole. For each depth image, if number of parts of hole greater than g, it was suspected hole in single frame. If more than k frames are suspected to have hole, it claims that there is a hole in front of user.

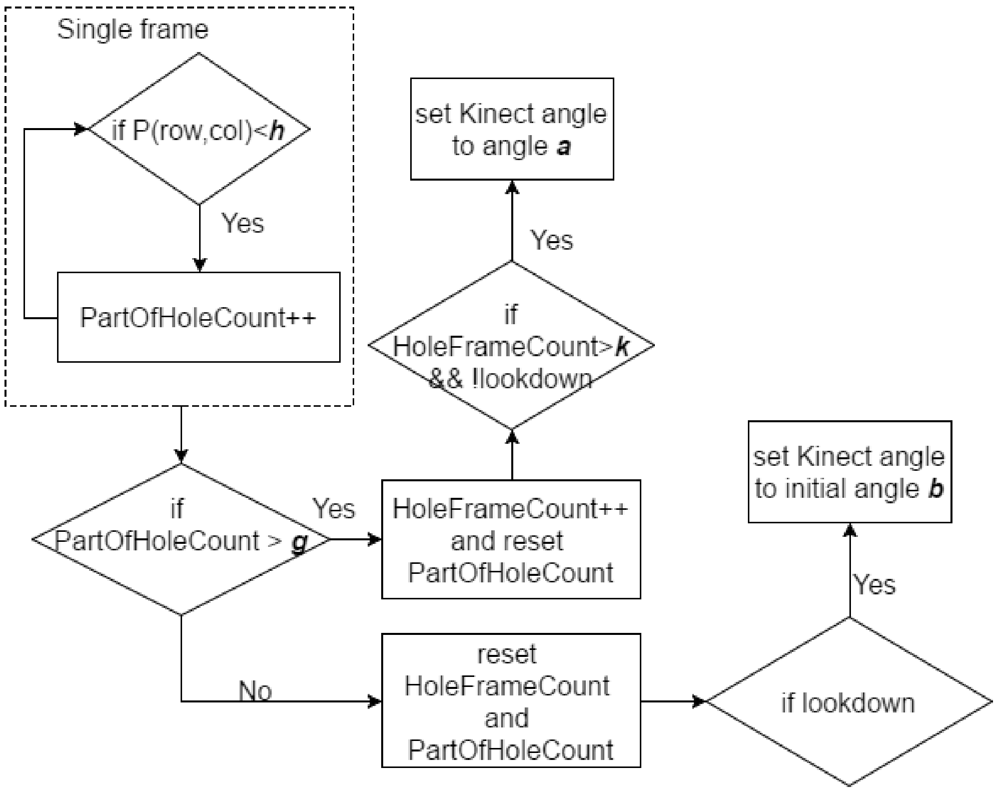

Single frame
if P(row,col)<*h*
Yes
PartOfHoleCount++
set Kinect angle to angle ***a***
Yes
if HoleFrameCount>***k*** && !lookdown
if PartOfHoleCount > ***g***
Yes
HoleFrameCount++ and reset PartOfHoleCount
set Kinect angle to initial angle ***b***
Yes
No
reset HoleFrameCount and PartOfHoleCount
if lookdown

Figure 24. Flow chart of hole detection decision making

Base on experiment, the adopted threshold h was -260mm, g was 10 and k was 5.

Setting the Boolean Matrix

There was some false negative within the ground region. In order to solve this problem, A Boolean Matrix was created. If the square plane in was ground area, the matrix will be marked as white (pixel value 255). Using Boolean Matrix ($\frac{320}{\ell} \times \frac{240}{\ell}$) required less computation cost than using the Depth matrix (320×240).

Morphological operation on Boolean Matrix

Then the Boolean Matrix was applied with the morphology operation, dilation, in order to repair the ground region.

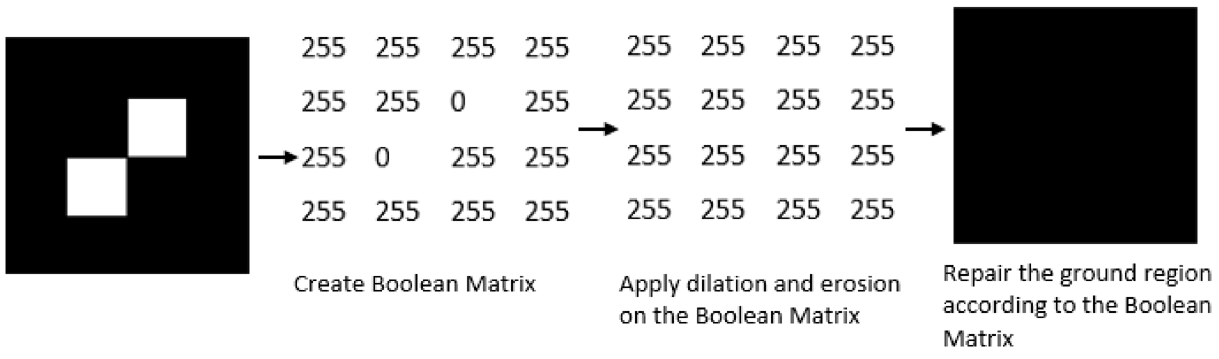


Figure 25. The flow of repairing the ground region

Test again with the same method as :

Plane edge (ℓ)	Dilation size	Accuracy
12	3x3	74.9%

Table 2

Setting the boundary of the Boolean Matrix

The morphological operation generated false positive of ground region in the boundary. Therefore, it was proposed to find the boundary of the Boolean Matrix, before morphological operation, using find contour algorithm. After morphological operation, even-odd rule algorithm would be used to test whether the matrix element is inside the boundary before the Boolean element is used to draw the ground region.

Drawing ground mask

Ground mask (320x240) was drawn according to the elements' value in Boolean matrix with position inside the boundary. The square planes were drawn using the following equation:

Point at upright corner of square plane,

$$P_{UpRight} = (x_{BooleanMatrix} \times \ell, y_{BooleanMatrix} \times \ell + (\ell - 1))$$

Point at upright corner of square plane,

$$P_{DownLeft} = (x_{BooleanMatrix} \times \ell + \ell, y_{BooleanMatrix} \times \ell + (2\ell - 1))$$

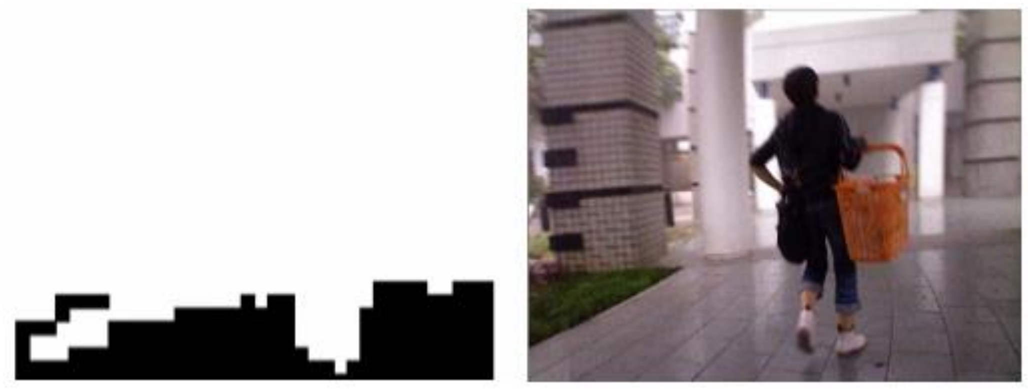


Figure 26. Example of ground mask and its corresponding color image

3.2.2 Obstacle Detection

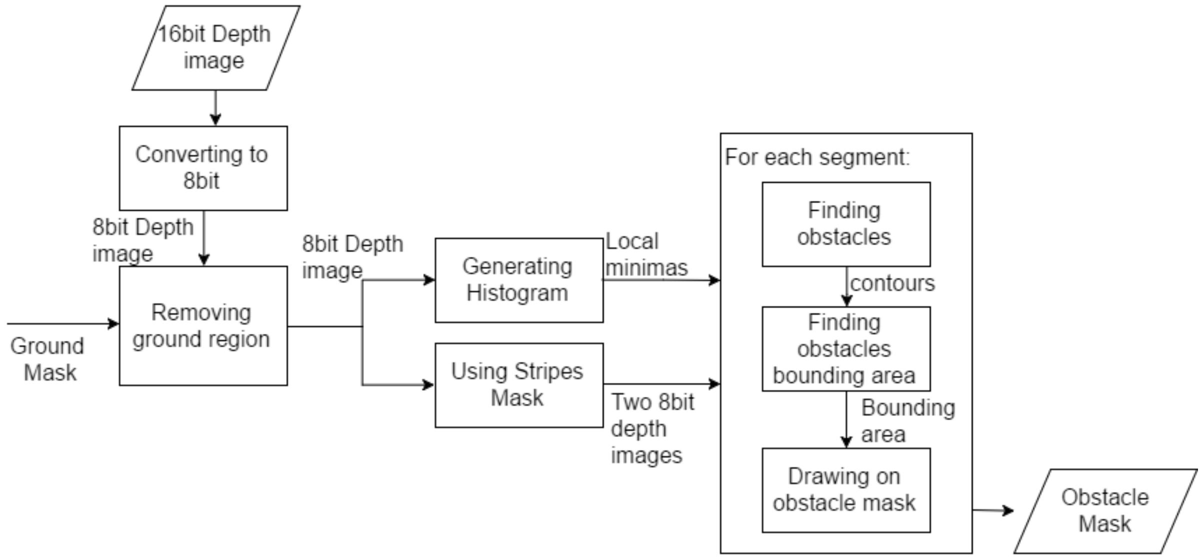


Figure 27. Flow chart of Obstacle Detection

Converting to 8bit

The compressed depth image is obtained by:

$$dst(i,j) = 255 \times \frac{src(i,j)}{2^{16} - 1}$$

Removing ground region

Depth image was bitwise and with the ground mask.

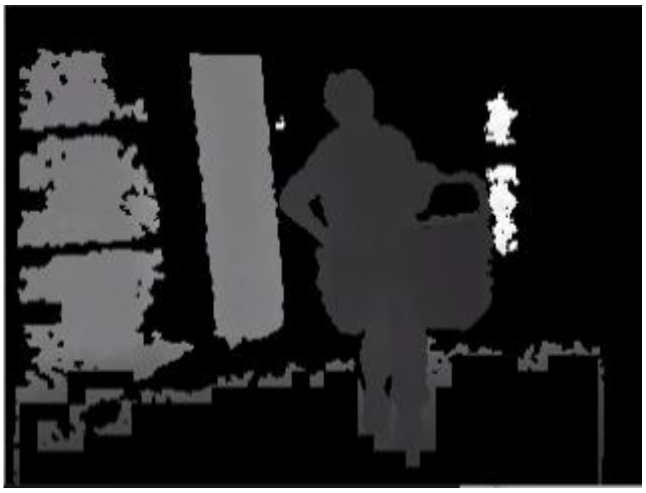


Figure 28. Ground removed in depth from Figure 18

Generating histogram

The calculated histogram is not smooth. Therefore, 1D Gaussian blur is used to smooth the histogram as shown in Figure 31 with kernel length 1x17 which is found by testing.

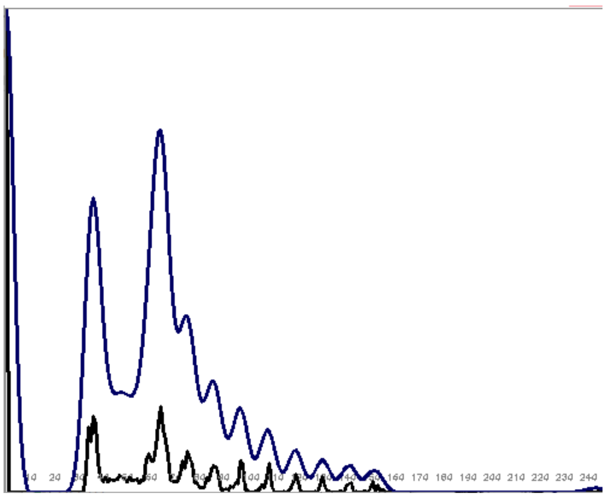


Figure 29. Histogram without smoothing is black while the blue line is the result of smoothing

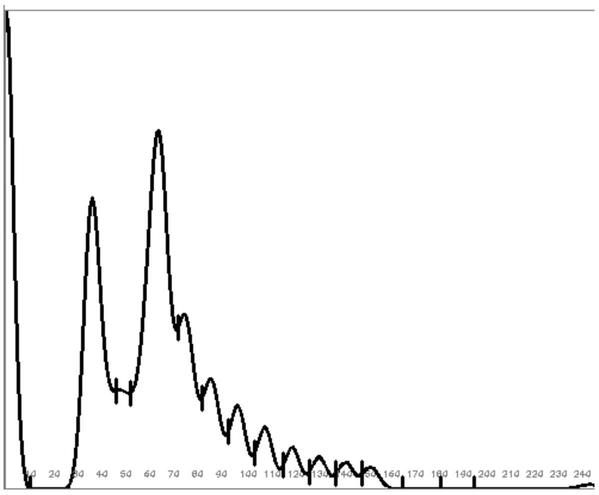


Figure 30. The result of finding local minima

During searching for the local minima, the slopes on each point was calculated by taking first derivative. The local minima were found when the change of slope sign was detected as Figure 30. The result of finding local minima. Since the depth value from far distance was not useful for detecting obstacles, searching local minima was stopped after a threshold in order to save processing time and number of segments in the later process. The threshold is 195 which was almost 3 meter away from the user.

Using Stripes Mask

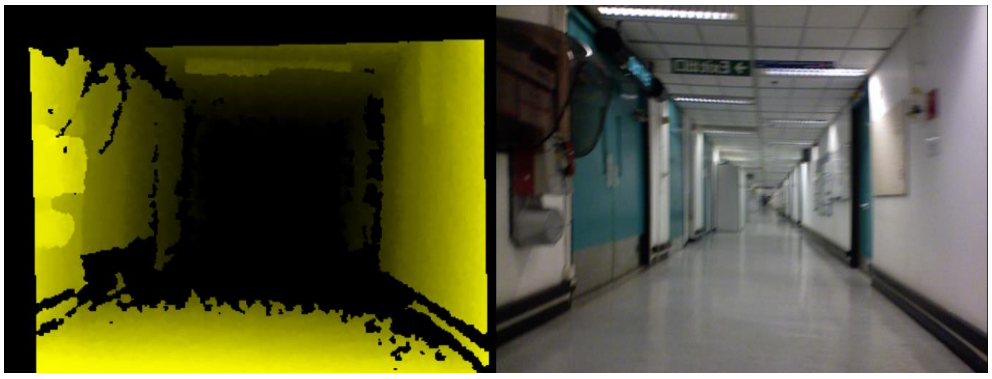


Figure 31. A corridor outside laboratory in HKUST

As shown in Figure 32 , the histogram-shape base segmentation of depth gave many rectangle segments along the corridor. Convex hull of the obstacle would cover the whole rectangle as shown in Figure 33. In order to prevent such defect, each segment was partitioned before carrying out find contour algorithm and draw convex hull algorithm using vertical stripes as mask. Finally, all found contours were draw into the same image with convex hull to construct an obstacle mask. This method would not increase the computation cost too much since the mask and partition could be obtained using bitwise operator. This was requiring less computation power than finding the concave hull.

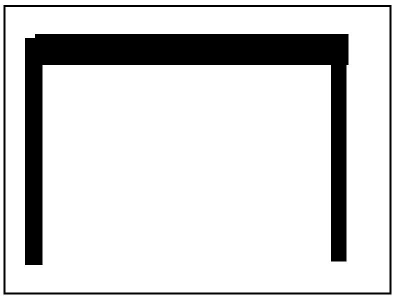


Figure 32. One of the segment of corridor

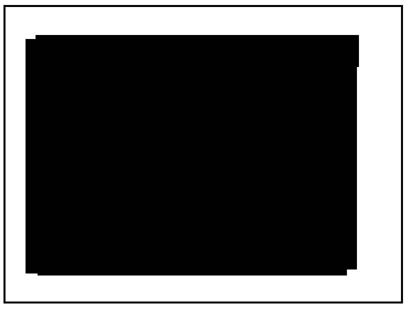


Figure 33. Convex hull of the Figure 32

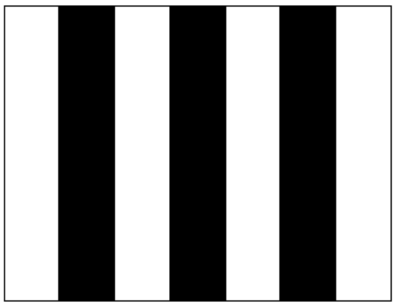


Figure 34. Mask for First partition

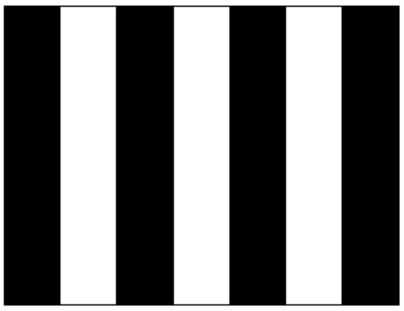


Figure 35. Mask for the Second partition

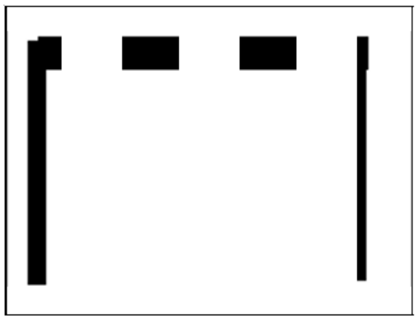


Figure 36. First partition of the corridor segment

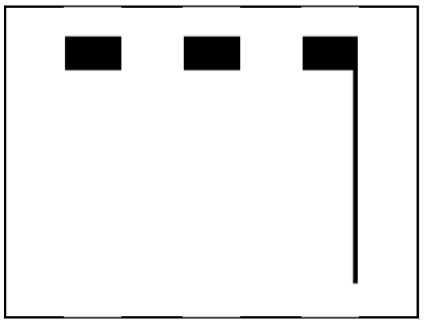


Figure 37. Second partition of corridor segment

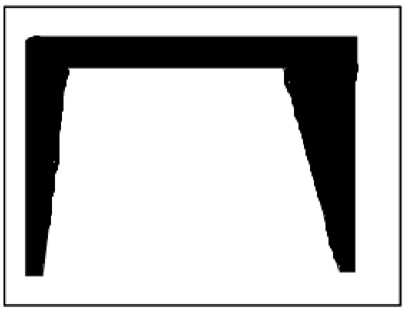


Figure 38. Result of convex hull of Figure 36 and Figure 37and combine them

Finding Obstacle

**Remark: The following figures were shown without using stripe mask for better understanding.

After finding the local minima, pixels belonged to the same interval would have the same index. Each index corresponded to one greyscale value as so that find contours algorithm could work. However, find contours algorithm generated contours which merged different labeled segments since the image was too complex as .

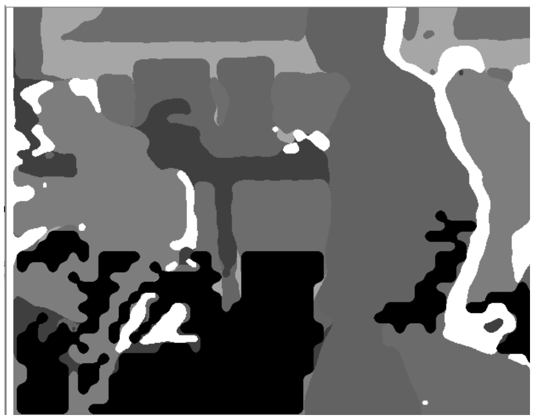


Figure 39: This is a visualization of different segments

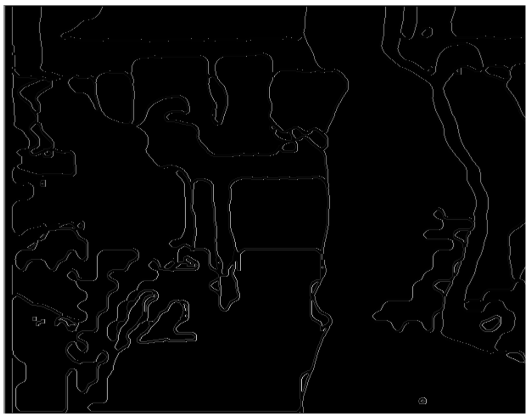


Figure 40: Poor result of finding contours of all segment

The problem was solved by drawing each labeled segment separately into different Matrices as Figure 41 in order to reduce the complexity of the image for finding contours. Although multiple Matrix were created to store the segment, the source depth image would be scant one time when determining the label of pixels. Therefore, this solution would not slow down the processing speed. The only drawback was that it required more memory space.

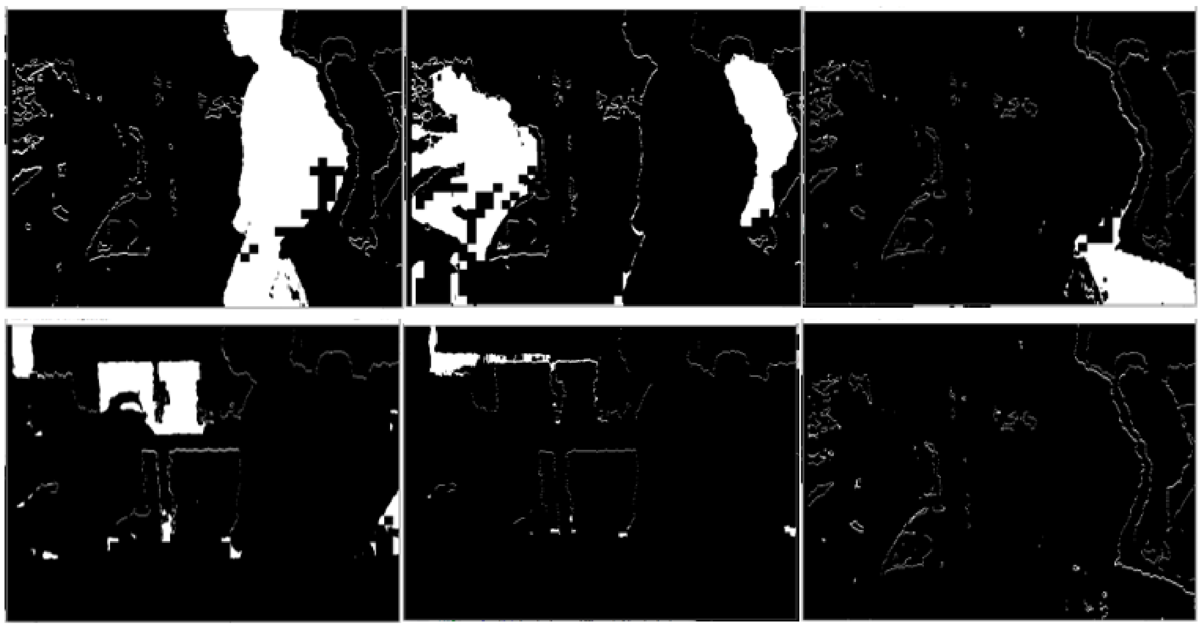


Figure 41: Image of each labelled segments

Before identifying the segments as obstacles, the small contours were filtered. Contours is eliminated if its size smaller than a plane's perimeter($\ell \times 4$), and if the area divided by arc length was close to 1 which meant it was a line. Finally, the remained contours would be defined as obstacles.

Finding obstacle bounding area

The resulted of segmentation was under-segmentation since many of the contours were eliminated. This result is acceptable to locate the rough position of the obstacle. However, for safety reason, convex hull of each contour was calculated as Figure 42 & Figure 43 in order to generate an area which included the whole body of the obstacle.

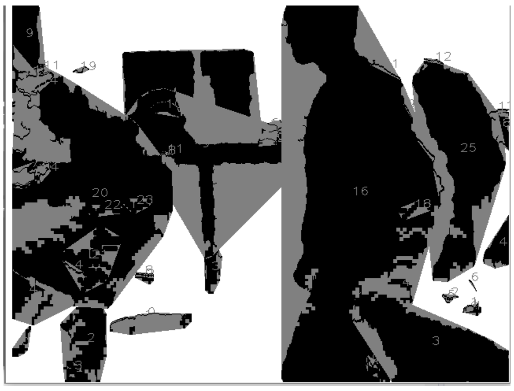


Figure 42: Visualization of contours within its convex hull

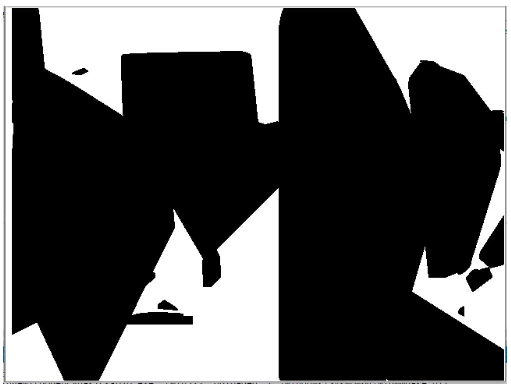


Figure 43: Obstacle mask

Drawing on obstacle mask

The convex hull will be drawn in the same matrix in order to generate an obstacle mask as shown in Figure 43.

3.2.3 Path Direction

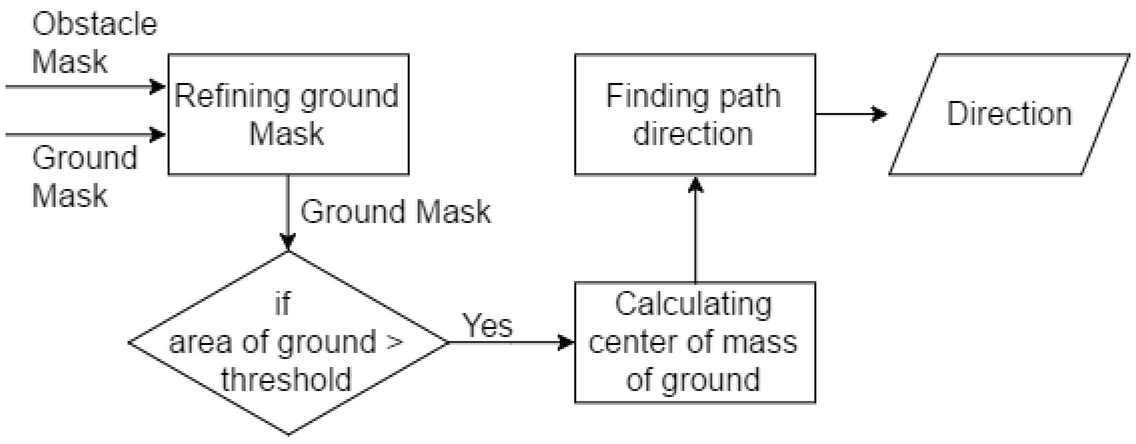


Figure 44. Flow chart of finding Path Direction

Refining the ground region obtained in Drawing ground mask in 2.2.1 Ground Detection. and the ground region was defined as path.

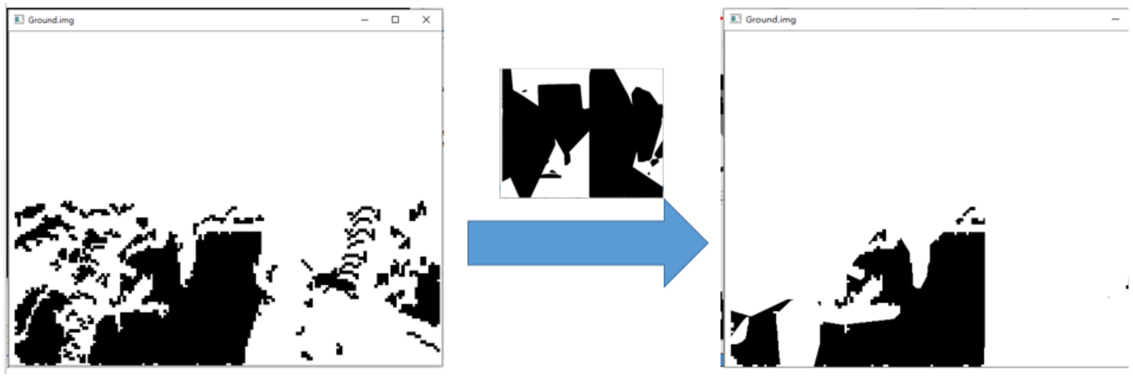


Figure 45: The procedure of refining the ground region

Same test as used in Table 1 is repeated :

Plane edge (ℓ)	Dilation size	1D Gaussian Blur kernel size	Accuracy
12	3x3	1x17	84.1%
12	5x5	1x17	86.0%

Table 3: The result of using ground detection with obstacle detection

Calculating center of mass

The proposed method to find the direction of the path is using the center of mass of the ground region. First, find the contour of the obtained ground region in the previous step. Then, the contour with the maximum area will be selected. Finally, the moment and center of mass of the selected ground region will be computed by using Green Theorem.

$$m_{ji} = \sum_{(x,y)\in O} (Intensity(x,y)x^j y^i)$$

O was the region enclosed by the contour, (α, β) was the center of mass, $\alpha = \frac{m_{10}}{m_{00}}, \beta = \frac{m_{01}}{m_{00}}$

Finding path direction

The column position α, will be treated as the direction of the path. r was number of row of image. $\alpha < r/4$ was Left, $\frac{3}{12} \times r < \alpha < r \times \frac{5}{12}$ was Left-Center, $\frac{5}{12} \times r < \alpha < r \times \frac{7}{12}$ was Center. $\frac{7}{12} \times r < \alpha < r \times \frac{9}{12}$ was Right-Center and $\frac{9}{12} \times r < \alpha < r$ was Right.

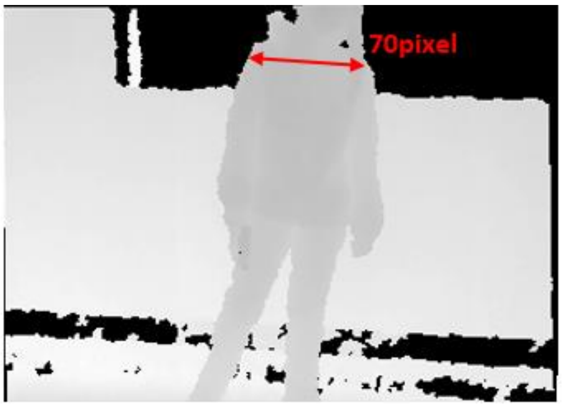


Figure 46: The width of shoulder when the depth image can just see the feet

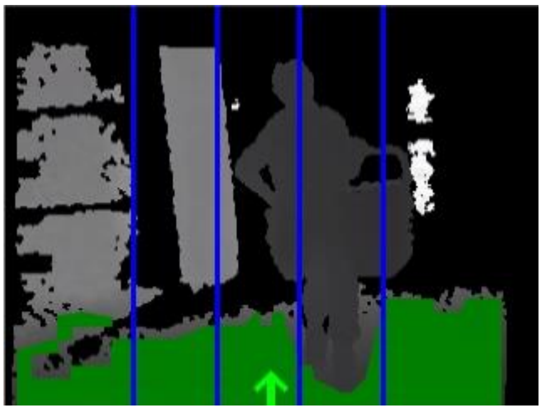


Figure 47: Image is partitioned into five direction

Arrow was draw in the frame as to judge the correctness of the path advice if the direction of the path was obstacle free.

	Plane edge (ℓ)	Dilation size	Number of bin in 1st partition	Number of bin in 2nd partition	Accuracy
Video 1	12	3x3	21	7	70.0%
Video 2	12	3x3	21	7	85.5%

Table 4: The result of testing path finding

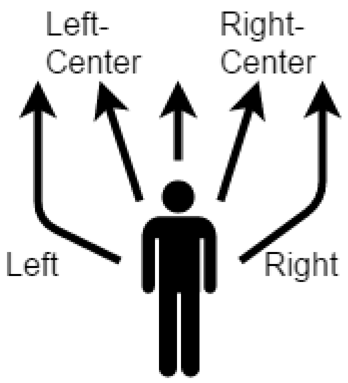


Figure 48 The actual body movement according to the Direction

Direction	**Vibration motor enable**
Left	Left
Left-Center	Left, Center
Center	Center
Right-Center	Right, Center
Right	Right
No Path	No vibration

Table 5 The Encoding table of vibration motor enable correspond to Direction

3.3 Stair Detection

3.3.1.1 Extract Parallel Edge Lines

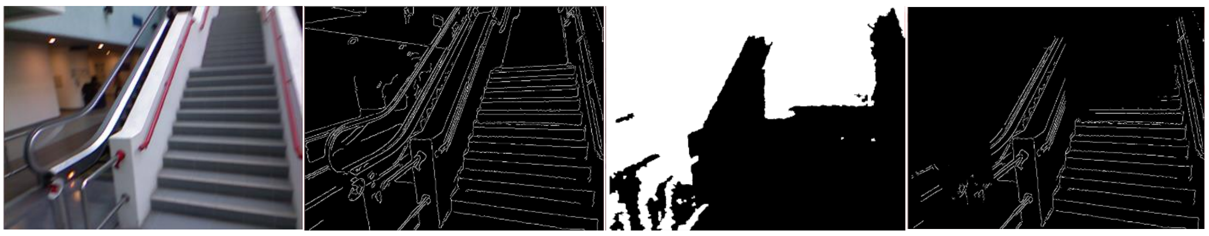
Get RGB Frame from Kinect

First, the colour image received from the Kinect camera is blurred to remove excess noise. The normalized box filter is used to blur the colour image, using the kernel K with box size 3x3:

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ & & \cdots\cdots\cdots & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

Apply Canny Edge Detection to RGB Frame

Canny Edge algorithm is then applied to the colour image to obtain an edge point map. In our experiments, the parameters used are: upper threshold = 120, lower threshold = 60, aperture size = 3.



(A) (B) (C) (D)

Figure 49 (A) Original color image; (B) Edge map from Canny Edge Detection algorithm

Apply Hough Line Transform to the Edge Map

Then straight edge lines are extracted by applying the Probabilistic Hough Line Transform the remaining edge points. The probabilistic version of Hough Line Transform is used instead to directly obtain the end points $(x_0, y_0), (x_1, y_1)$, instead of the polar coordinate point form (ρ, θ). In our experiments, the parameters used are: ρ = 2, θ = $\frac{\pi}{720}$, accumulator threshold = 100, minimum length = 60, minimum gap = 5.

Calculate the Angle for each line in Vector of Edge Lines

After the straight edge lines are found, the angle of the lines must be calculated to determine whether two lines are parallel. A line can be expressed as two end points, $(x_0, y_0), (x_1, y_1)$, with the angle θ. Then,

$$\theta = \tan^{-1} \frac{\mathrm{y}_1 - y_0}{\mathrm{x}_1 - X_0}$$

Sort Parallel Edge Lines into 180 bins by their Angles

After obtaining the angle of the lines, discard lines that have angle $59° < \theta < 122°$ as lines within this angle range would be almost parallel to the user. Hence, stairs are very unlikely to be in front of the user.

Next, lines of similar angle are considered parallel and are grouped together into group G_θ. The grouping is done by offsetting the angle θ to the middle of the group of size n, where the middle of the group is calculated by $f(\theta)$:

$$f(\theta) = \begin{cases} \theta - (\theta \ \% \ n) & if \ \theta \ mod \ n \leq \frac{n}{2} \\ \theta + (-\theta \ \% \ n) & otherwise \end{cases}$$

$$for \ \ 0° \leq \theta < 180°, \quad 180 \ \% \ n = 0$$

In our experiments, we take $n = 10$.

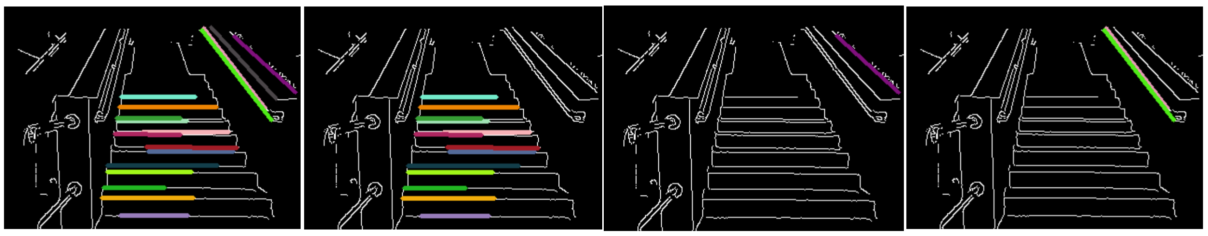


Figure 50(A) $G_{\theta=0°,40°,50°}$ *(B)* $G_{\theta=0°}$ *(C)* $G_{\theta=40°}$ *(D)* $G_{\theta=50°}$

3.3.1.2 Detect Stairs by Depth Features

Determine the Angle of the Stairs

From the groups of parallel edge lines, find group G_θ with the largest size and treat angle θ as the general angle of the stairs. However, the group must have at least 3 parallel lines to be considered as stairs. If no such group exists, then the image frame does not contain any stairs and the program starts over.

Calculate the middle line of the stairs.

Using the end points of every edge line, even non parallel lines, apply Least Squares Method $\rho(r) = r^2/2$ to obtain a best fit line, then shift this line by $\pm 90°$ to obtain a line that represents the direction and the midline of the stairs.

Extract Edge Lines that belong to the stairs.

For all edge lines, if an edge line intersects with the stairs midline, then this line must belong to the set of edge lines representing the stairs.

Determine if image frame contains stairs

Using an intersection point P_y from the line l from the set of stairs and the stairs midline, where y is the y-coordinate of the line l, find the corresponding point $D(P_y)$ from the depth image. Then line l is the stairs step edge line if it satisfies:

$$\begin{cases} if\ \ D(P_y) > \ D(P_{y-1}),\ then\ \ l\ \ is\ \ stairs \\ else\ \ if\ \ D(P_y) > (\ D(P_{y-1}) - \sigma\ ), then\ \ l\ \ is\ \ stairs \\ else, l\ \ is\ \ not\ \ stairs \end{cases}$$

This is means that the depth must have an ascending trend, with at most tolerance σ. If all lines in the set of stairs are step edge lines, then the frame contains stairs and send output to Text to Speech, else the program starts over.

3.4 Sign Recognition

3.4.1 Colour Frame Preprocessing

In this phase, 1280x720 colour frames were passed to this sign recognizer but preprocessing was needed to remove noise and be prepared for later processing use - find the elliptical contour of signs in the next phase - before conducting the recognition. Hence, the edge mask of a frame was necessary because it was useful to find contour. There were 4 steps to prepare an edge mask, which were as follows.

Firstly, colour space of the colour frame was converted from 32-bit RGB to 8-bit grayscale by extracting the intensity of each pixel because edge detector, which was applied next, only cared about the contrast among pixels. The gray matrix Y in OpenCV is defined by,

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B,$$

where R, G and B are the matrix of the R, G and B channel of the source

However, there might be light reflection, as shown in Figure 51 and Figure 52 affecting the result. To address this problem, it was observed that the Red Channel was less affected by light reflection and the contrast between background colour and the text (white area) within a sign was the highest among all channels, while Blue channel contained more noise, so Red Channel was chosen to be the grayscale intensity, instead of the OpenCV default one, such that,

$$Y = 1 \cdot R + 0 \cdot G + 0 \cdot B$$

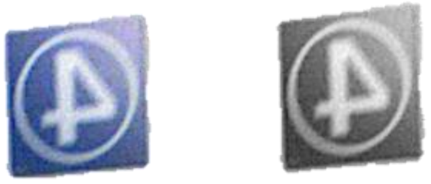


Figure 51: Original (Left); Grayscale defined in OpenCV (Right)

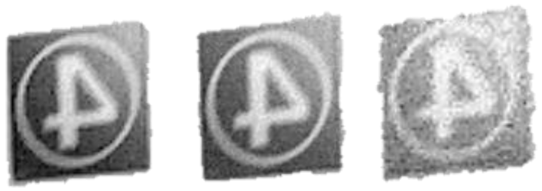


Figure 52: Red channel (Left); Green channel (Middle); Blue channel (Right)

Secondly, the gray frame was smoothed by applying Gaussian Filter so that noise could be reduced in the blurry gray frame. This step was essential to prevent false positive of edge detection, which was the next step, because edge detector always regards contrasts caused by noise as edges. The filter therefore helped reduce the effects caused by obvious noise. Gaussian Filter with kernel size (k, k), Gaussian kernel standard deviation in x direction, σ_x, and Gaussian kernel standard deviation in y direction, σ_y was applied. Gaussian Filter in two dimensions is defined by,

$$G_0(x, y) = Ae^{-\left[\frac{(x-\mu_x)^2}{2\sigma_x}+\frac{(y-\mu_y)^2}{2\sigma_y}\right]},$$

where μ_x, μ_y are the mean (the peak) and σ_x, σ_y represent the variances of x and y.

By experiment, k = 7, σ_x = 2 and σ_y = 2 were used.

Thirdly, Canny Edge Detector was with 45 as the lower threshold, 108 as the upper threshold and 3 as kernel size was applied to get the edge mask. It was assumed that the ratio of lower threshold to upper threshold was 1:2.3.

Fourthly, unobvious edges may disconnect during last process so edge masks were dilated such that edges were able to connect each other nearby and became a joined line. Dilation with kernel size 2 was applied on the edge mask. Samples of final edge masks are shown in Figure 53.

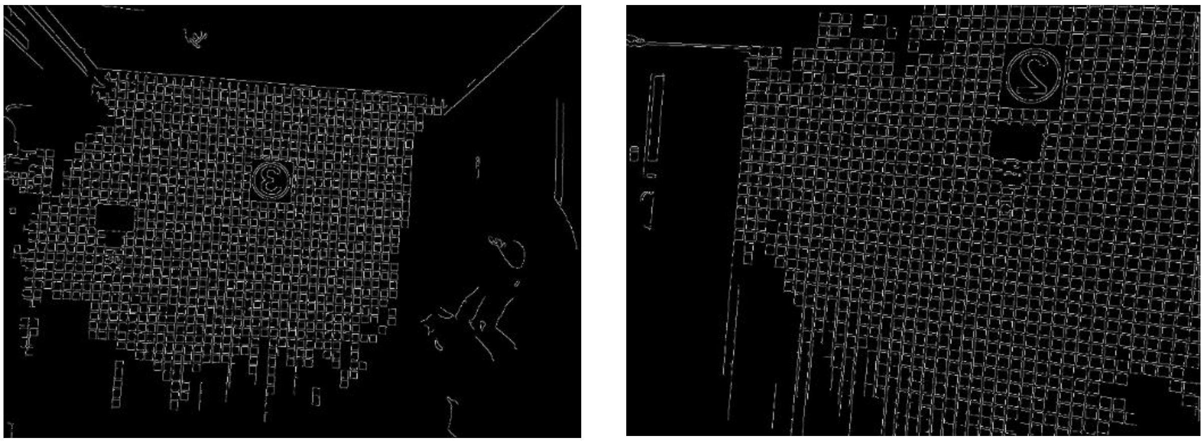


Figure 53: Samples of edge masks

3.4.2 Find the Elliptical Contour of a Sign with Conic Fitting

The goal could be achieved by finding contours in a frame by border following on the binary edge mask on the binary edge mask prepared first, followed by checking whether a contour was an ellipse. If there was any elliptical contour, it proceeded to the next step. Otherwise, the system processed next contour or next frame.

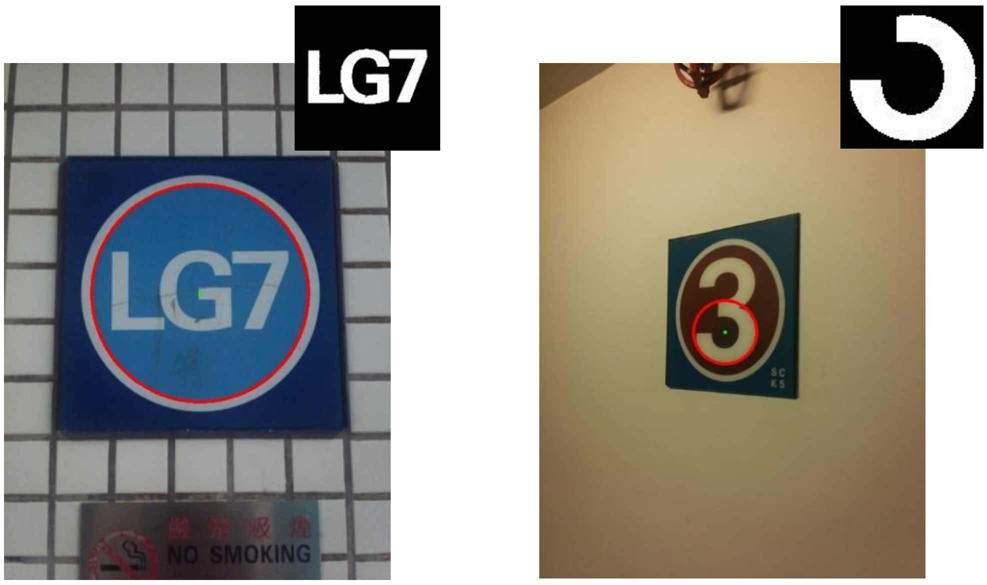


Figure 54: circle detection results by Hough Transform

It was designed to determine whether there were circles. However, after some test, it was observed that signs could not be detected well at all viewing angles, which is shown in Figure 54. Therefore, the proposed algorithm was modified to detect ellipses, instead of finding circles.

Before checking if a contour an ellipse, the following two decisions were made to ignore some cases and speed up the entire process.

1. Is the contour area beyond a reasonable range of size of sign (too big or too small)?
2. Is the contour shape and size similar to the last contour examined?

If the answers of above questions were negative, the contour was then examined to check whether it was ellipse by conic fitting to find the best-fit ellipse of the contour. The center, the radii, and the rotation angle, θ, of the best-fit ellipse could then be obtained. After that, the best-fit ellipse was compared with the contour mask to check whether they were of similar sharp and size and at the same position. This was achieved by drawing the best-fit ellipse on another m x n matrix with the same size of the frame. The similarity between the contour mask matrix, C, and the best-fit ellipse matrix, E, could be calculated by dot product:

$$cos(x) = \frac{C \cdot E}{\sqrt{C \cdot C}\sqrt{E \cdot E}}$$

If $cos(x)$ is closed to 1, the contour was considered as an ellipse, as well as the contour of a sign. Figure 55 and Figure 56 show the detection results of two samples of the modified algorithm.

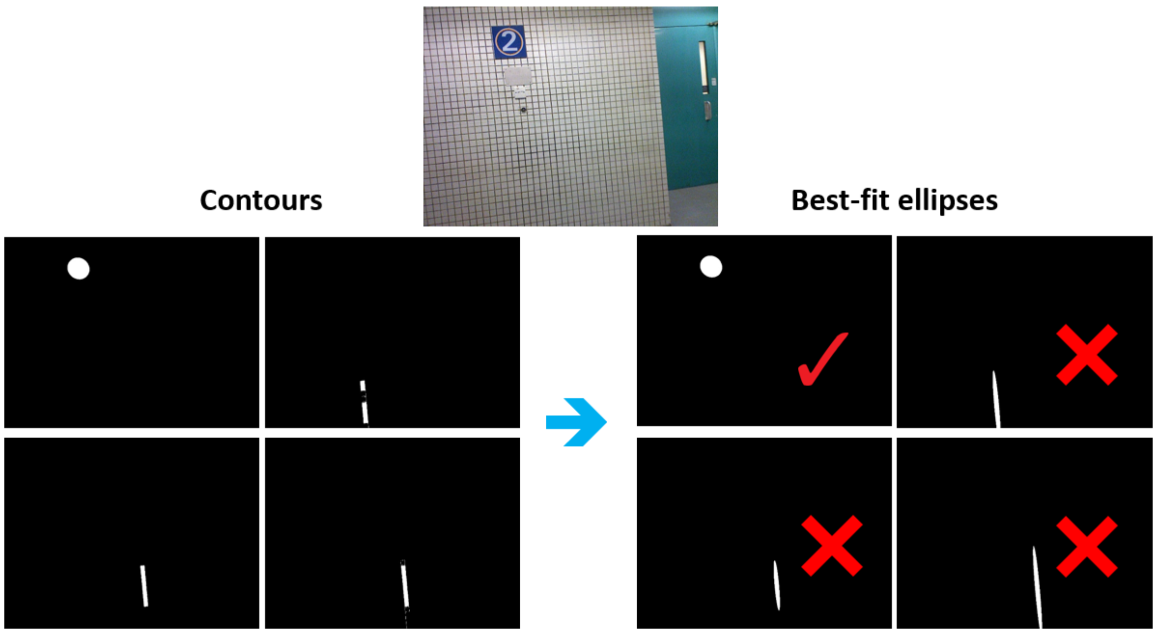


Figure 55: sample result of finding elliptical contour

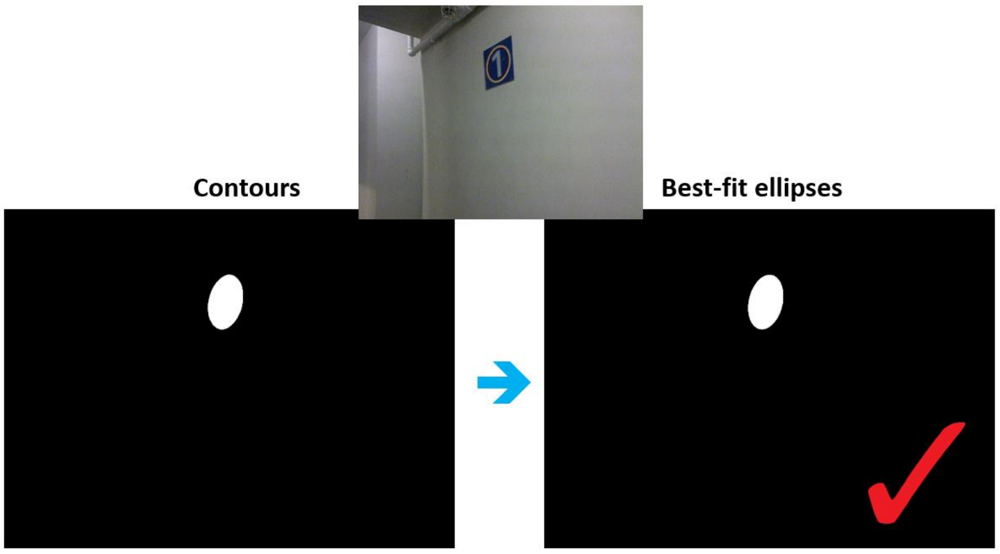


Figure 56: sample result of finding elliptical contour. the image of sign is captured from side view.

3.4.3 Crop and Binarize the elliptical area

The gray image was then cropped by logically AND the elliptical contour mask so only content inside the circle was extracted. Then, the cropped area was binarized by calculating the average intensity, m, of the area. The best case of the binary image is that the text is in white while the background is in black. Therefore, the gray area was applied thresholding with threshold, t = m and it gave a binary image. Thresholding with maximum value and threshold, t, are defined by:

$$\mathrm{dst}(x,y) = \begin{cases} maxVal, & if \;\; src(x,y) > t \\ 0, & if \;\; src(x,y) \leq t' \end{cases}$$

where maxVal is the maximum value of intensity of a pixel (255 for 8-bit)

$$Threashold, t = \;\; average \;\; intensity \times 1.35$$

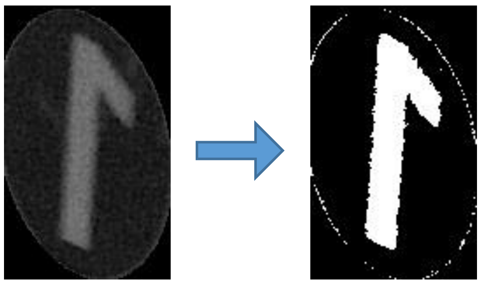


Figure 57: a white circle appears in the binary image

However, a white circle or ellipse occasionally appeared on the cropped binary image, as shown in Figure 57. It needed to be filled with black in order to be removed. As the position of the white circle was not fixed, the system could not fill it directly. The solution was to make the background of the sign (the area outside the sign) in white by logically OR the binary image and the inversion (negative image) of the elliptical contour mask, followed by removing the white area from the 4 corners by flood fill. This procedure is illustrated by Figure 58.

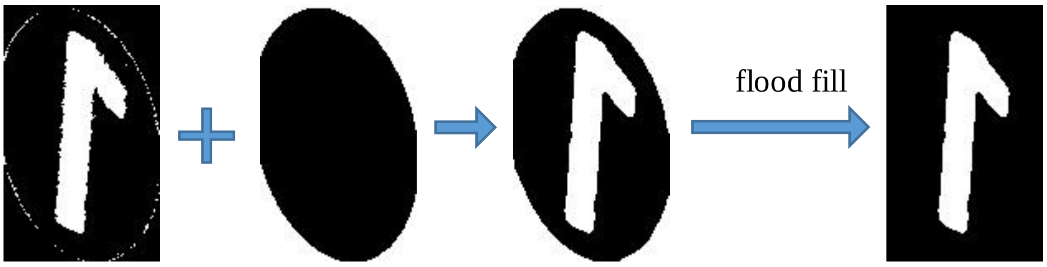


Figure 58: an illustration of removing a white circle

3.4.4 Recognize the Text inside the cropped area with OCR

Clear binary images consisted of white text only in this stage and some samples are shown in Figure 59. Hence, it could be recognized by character recognizer directly. However, since input frames were laterally inverted, the binary images of text must be inverted before recognition. It was achieved by re-mapping. Two maps, $map_x(x,y)$ and $map_y(x,y)$, which had the same dimension as the binary images were first obtained by:

$\forall\ i \in [0, width\ \ of\ \ image),\ j \in [0, height\ \ of\ \ image)$:

$$map_x(i,j) = width\ \ of\ \ image - i$$
$$map_y(i,j) = j$$

Then, the following mapping was employed to invert images:

$$dst(x,y) = src\left(map_x(x,y), map_y(x,y)\right)$$

Lastly, in order to identify characters, OCR was applied and get the text results.

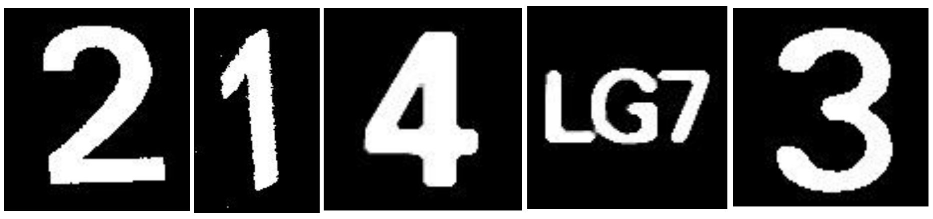


Figure 59: samples of text images

3.5 Face Detection and Recognition

3.5.1 Preparation

3.5.1.1 Obtain Database for Classifier of Faces

Since human faces contain common features, faces could be detected by those features with a classifier. Haar-like feature was chosen as the method to extract human face features and Haar feature-based cascade classifier was applied to detect if there is any face on a frame. ‘`haarcascade_frontalface_alt.xml`’ from OpenCV was used. It was loaded into the system first before it runs the face detector.

3.5.1.2 Pre-calculate average Face Colour

Besides from face features, face colour was also another parameter to verify whether the colour of a face-like object detected by the face detector was similar to the colour of human faces, so as to avoid false positive. Therefore, preparation of the average face colour of people around the world was needed so that the comparison could be done during running time.

Three steps were done to ensure that the calculation result was fair.

1. Since the database contain a relatively large area of background, faces were first cropped by face detector. Hence, the result obtained was less affected by the background.
2. The faces were resized into the same size so that weighting of the colour of each face was equal to each other.
3. Intensities of faces were equalized because their contrasts and brightness were different. This was achieved by converting the colour space from RGB to YUV first, followed by equalizing the Y-channel. The colour space was then changed back to RGB after equalization.

After that, the mean of intensity of each colour channel of human faces could be calculated from the 3 channels, R, G and B, of faces in LFW database. With the mean, μ, the variance and the standard derivation of each channel could also be calculated by,

$$Var(X) = E[(X-\mu)^2]$$

$$\sigma = \sqrt{E[(X-\mu)^2]} = \sqrt{Var(X)}$$

where X is a random variable.

The calculation results are shown in Table 6.

	R-channel	G-channel	B-channel
Mean	153.315	120.392	100.175
Standard derivation	76.6897	72.5205	68.3169

Table 6: Mean and S.D. of human face colour from the LFW database

3.5.1.3 Train Recognizer

Before passing detected faces to the face recognizer, the face recognizer had to be trained to learn the set of features of each person. Similar to detection, the database of sets of features of different people was required to be loaded to the system before running the recognizer.

For face recognition, LBP was chosen to as the method to extract features of faces and LBP recognizer from OpenCV was used. To begin with, face images of each person to be recognized (in this case, our group members) were needed. They were invited to stand in front of Kinect for a while to capture their faces by the real-time face detector (more details could be found in next section), which had been implemented in the earlier stage of this project and had been using for detecting faces in this system.

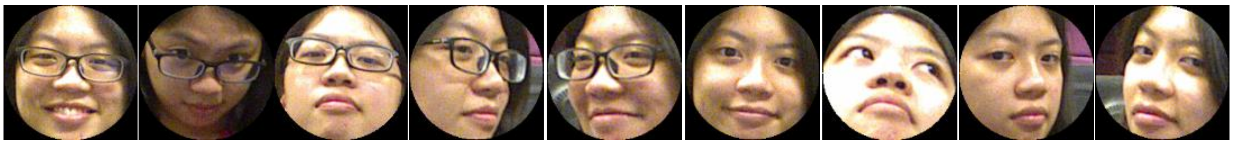


Figure 60: samples of faces at different angles and with / withour glasses

These images were then separated into two sets, the training set and the testing set. In order to obtain a better recognition, as shown in Figure 60, the training set has to be clear and includes different angles of faces of the same person, and also the faces without glasses if they wear glasses. All other images were allocated to the testing set. 70 face images of each teammate were chosen to be the training set for recognizer. All these training images are shown in Appendix F. There were also 160 face images randomly chosen from LFW face database as a training set of 'Guest'. Also, since all of the faces passed to the recognizer has to be in the same size, all of them were resized to a fixed size 120x120.

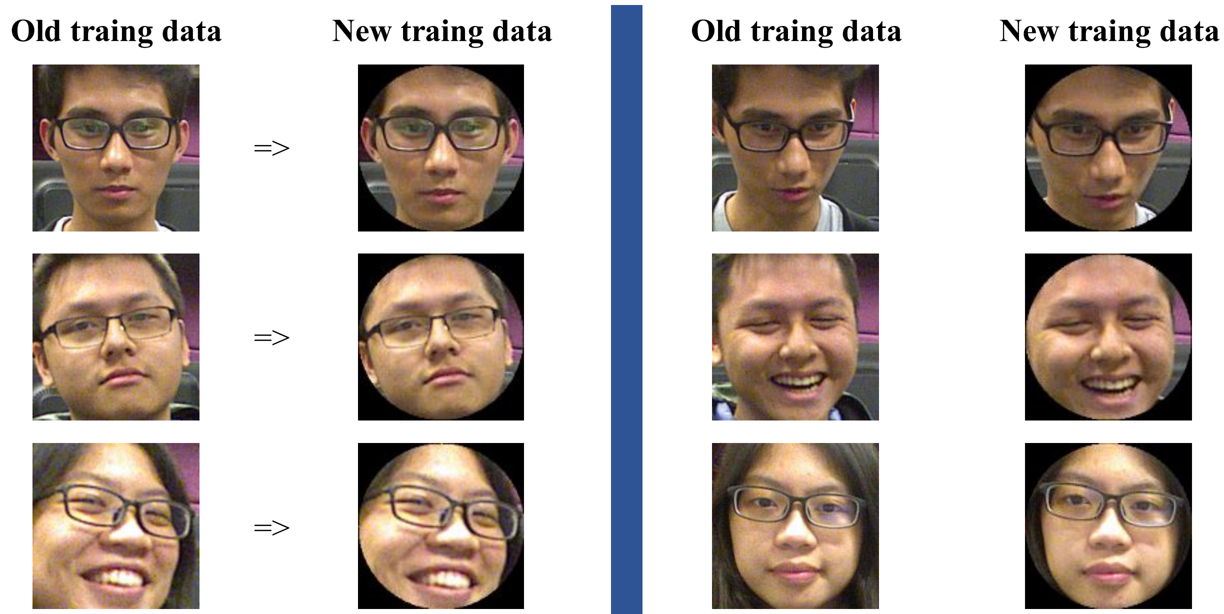


Figure 61: samples of faces w[illegible] and without background

During the development process, face images with background were used. After some test, it was found that removing the background, as well as hair style, could improve the confidence of recognition result. Hence, a new database, in Appendix F, was trained with only the circular area of the face detection result cropped. Some samples are shown in Figure 61. Average confidence of face recognitions in a 90s video by the old database and the new database is shown in Table 7, the higher the confidence, the lower the certainty of recognition prediction.

Old database	New database	Improvement (%)
53.6003	48.413	9.678%

Table 7: Average confidence obtained by using the old and new database respectively

3.5.2 Face Detection

3.5.2.1 Detect Faces

Figure 62: positive samples of face detection

As mentioned, Haar cascade classifier, `haarcascade_frontalface_alt.xml`' from OpenCV, with minNeighbors = 6 to avoid false positive, was used as the face detector to detect if there is any face on a frame. Before passing frames to face detector, colour frames were converted into grayscale matrix, Y, with the OpenCV default conversion defined by,

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B,$$

where R, G and B are the matrix of the R, G and B channel of the source

and it was equalized afterwards. Some positive samples are shown in Figure 62. Moreover, in order to speed up the Haar cascade classifier detection result, the minimum and maximum face sizes were defined as,

$$Minimum\ \ face\ \ size = Frame\ \ Width\ \ \div 20$$
$$Maximum\ \ face\ \ size = Frame\ \ Width\ \ \div 4$$

Average face width of our team members were measured and was 14.5cm. With the above limitations and the angle of view of Kinect (43° in hright, referring to Figure 23), the furthest and nearest distance that faces could be detected by the face detector was calculated,

$$View\ \ of\ \ Kinect\ \ in\ \ width = 43° \times \frac{1280}{960} = 57.33°,$$

$$furthest\ \ distance = \frac{14.5cm\ \ x\ \ 20 \div 2}{tan 28.67°} \approx 265cm,$$

$$nearest\ \ distance = \frac{14.5cm\ \ x\ \ 4 \div 2}{tan 28.67°} \approx 53cm$$

During implementation process, original 1280 x 960 colour frames had been passed into the classifier for detection. This had caused the detection speed extremely slow. However, it was known that detection does not require high resolution, so, in late development process, the colour frames were first resized to 480x360 with a scale factor of 0.375, without affecting face detection result. Table 8 shows the average duration of face recognition before and after resizing frames.

Input: 270 1280x960 image frames extracted from a video containing faces **Size after resizing: 480x360 (Scale Factor = 0.375)**			
Trial	**Avg before resize (s)**	**Avg after resize (s)**	**Improvement (%)**
1	0.367047	0.256529	30.11%
2	0.383246	0.23135	39.63%
3	0.328429	0.199471	39.27%
Average	**0.3596**	**0.2291**	**36.29%**

Table 8: Average duration of face recognition before and after resizing frames

3.5.2.2 Verify detection result

After getting face detection result from the classifier, there might be false positive so verification of the detected face-like objects was need. It was observed that the colour of false positive was usually different from human face colour. Therefore, average face colour was chosen as a parameter for verification, which eliminates false positive by comparing the colour of detected face-like objects to the pre-calculated average face colour. If they are in similar colour, the detected face-like object is regarded as a positive sample of face and proceed to the next step. Otherwise, it is dropped.

The intensity of detected face-like object was equalized before comparing the face colour because of the variation of contrast and brightness among frames. This was achieved by converting the colour space of the face-like object detected from RGB to YUV first, followed by equalizing the Y-channel. The colour space was then changed back to RGB after equalization.

The mean and standard derivation of face colour of each colour channel, μ_i and σ_i, had already been calculated during preparation. From the result, if the intensity of a colour channel of a pixel fell between $(\mu_i - k\sigma_i)$ and $(\mu_i + k\sigma_i)$, where k is a constant, it was considered as

a similar face colour. Pixels of similar colour in each channel were marked as ‘1’ while other pixels were marked as ‘0’. As a result, a similarity mask was generated.
Without loss of generality, similarity mask of each channel, i, and k = 1.15, was represented by,

$$dstMask_i(x,y) = \begin{cases} 1, & if \;\; k\sigma_i \leq \;\; src_i(x,y) - \mu_i \leq k\sigma_i \\ 0, & otherwise \end{cases}.$$

The mask was then converted to grayscale for comparison. If 40% - 80% of pixels of a face are of similar colour, the face was verified as a correct detection. Otherwise, it will be dropped. Examples of false positive is shown in Figure 63, and it could be eliminated after comparing face colour.

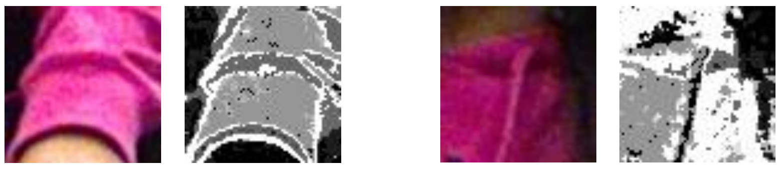


Figure 63: examples of false positive and its similarity mask

3.5.3 Face Recognition

Before passing to the face recognizer, it is known that recognition requires higher resolution than that of detection, so detected faces were cropped from the original 1280x960 frames. Therefore, a mapping between the face positions and sizes from 480x360 frames to 1280x960 frames were needed. The corresponding positions and sizes in 1280x960 frames which were defined by,

Corresponding Position:

$$(x_{1280}, y_{1280}) = \left(\frac{x_{480}}{Scale\;\; Factor}, \frac{y_{480}}{Scale\;\; Factor}\right)$$

Corresponding Size:

$$Width_{1280} = \frac{Width_{480}}{Scale\;\; Factor}\;, \qquad Height_{1280} = \frac{Height_{480}}{Scale\;\; Factor}$$

where scale factor = 0.375 in our system

To recognize faces via LBP recognizer, texture classes were prepared by sets of training images as mentioned previously. Each texture class contained a set of the texture models, i.e. models for face images of the same person, with certain features. After comparing the distances of the testing LBP model to each model in the texture classes, the nearest class, which contains the shortest distance to the testing model, is consider as the recognition prediction result. As the result, the distance of the nearest class is an important parameter of how precise or how confident the recognition prediction is. In order words, if the distance to the nearest model is too large, the recognizer may give a wrong result or the detected face may not exist in the database, resulting in incorrect recognition. Hence, the faces with large distance will be regarded as Guest in the system. Currently, the threshold of confidence is set to 100. Figure 64 shows an example of face recognition prediction result with confidence (the higher the confidence, the lower the certainty of recognition).

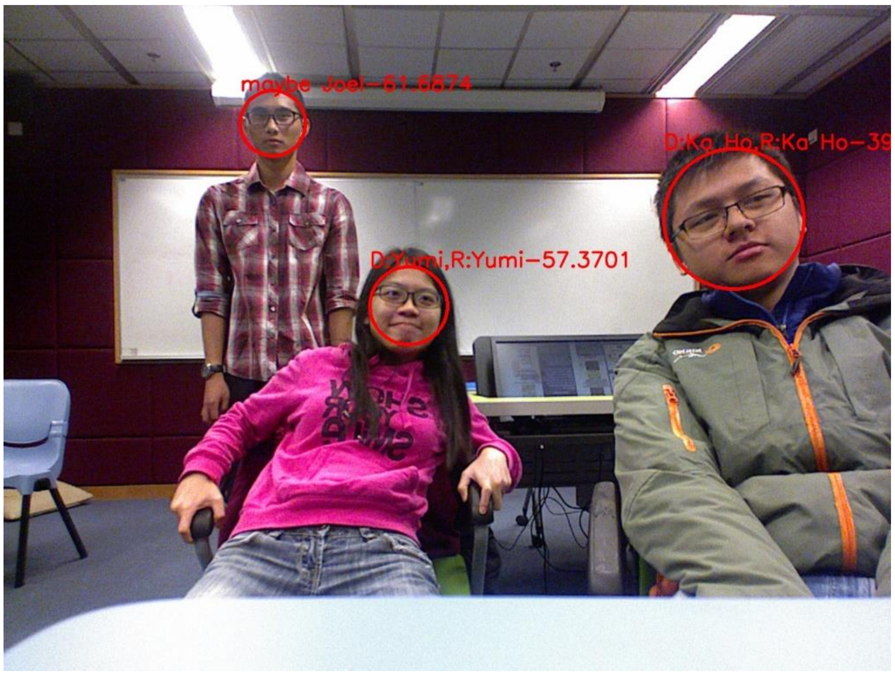


Figure 64: Face recognition predictions and their confidences

During implementation process, it was observed that it took longer time to recognition a larger face. To speed up the process, faces larger than 120x120 was resized to 120x120. A test was carried to understand the improvement in Table 9 before and after this modification.

Input: 270 1280x960 image frames extracted from a video containing faces Resized to 120x120 when recognizing face			
Average Duration of Entire Face Detection and Recognition Process on 1 frame			
Trial	**Before resize (s)**	**Avg after resize (s)**	**Improvement (%)**
1	0.256529	0.231685	9.68%
2	0.254496	0.229999	9.63%
3	0.255393	0.218818	14.3%
Average	**0.25547**	**0.22683**	**11.2%**
Average Duration of predicting 1 face with LBP Face Recognizer			
Trial	**Before resize (s)**	**After resize (s)**	**Improvement (%)**
1	0.0966527	0.0712845	26.2%
2	0.0955271	0.0727303	23.9%
3	0.096126	0.0719619	25.1%
Average	**0.096102**	**0.071992**	**25.1%**

Table 9: Average processing duration of before and after resizing

3.5.4 Further Improvement and Simple Tracking System

As there might be incorrect prediction because of more noise or blurred frames in a real-time system, it might not be reliable to determine a person with only one capture and one prediction. The solution was to keep a counter of different people for each detected face and to detect several faces at the same time so that previous detections and the respective recognition prediction results were stored. Also, a person needs to be recognized for k times such that he / she is considered to be recognized to make the recognition result more reliable, and k = 3.

In order to achieve this, a tracking system was needed. Another advantage of the tracking system was that, after recognizing a person, the system did not need to recognize the face again at the same position and not to keep notifying the prediction result of a person and did not annoy users.

Since the face detection algorithm used the most computation power, compared to other algorithm, a fast, simple and light tracking system was needed. It was, therefore, done by comparing face centers of current frame and last frame. If the center moved within an offset (in the system, offset = 50), the detected face was considered as the same face object as last frame around that position. Suppose there are m faces detected last frame and n faces detected in current frame,

$$\forall\ i \in [0, m], j \in [0, n],$$
$$\begin{cases} abs\left(lastFaceCenter_{i_x} - currentFaceCenter_{j_x}\right) < offset \\ abs\left(lastFaceCenter_{i_y} - currentFaceCenter_{j_y}\right) < offset \end{cases} \Rightarrow same\ \ face\ \ object$$

On one hand, if a tracked face object is being predicted as the same person for k = 3 times, the recognition result would output to the text-to-speech queue. On the other hand, if a tracked face object at certain position was not detected for certain frames, f = 6, it will be erased from the system memory. Hence, if faces appear at that position again, it will be recognized again.

In addition, if there are more than 3 times wrong prediction in the beginning, the prediction label memorized by the tracking system could be updated when another prediction label (of the same face object) is recognized by more than doubled times of the old prediction, so that it did not keep providing an incorrect prediction.

$$\frac{\boldsymbol{Times\ \ of\ \ new\ \ prediction}}{\boldsymbol{Times\ \ of\ \ old\ \ prediction}} > \mathbf{2} \Rightarrow \boldsymbol{Update\ \ label}$$

4 Testing

4.1 Path Detection

The participants are normal students wearing sleep mask, safety goggles for simulating Glaucoma and Cataract. A classroom and residential hall were picked up to form a route that can capture most of the walking environment in HKUST for testing the performance of the system. Then, the route was analyzed to plan a way for visually impaired people to follow in order to reach the destination. Finally, the duration and speed of participants used in the test will be recorded and analysis.

4.1.1 Glaucoma and Cataract simulator

A simple Glaucoma and Cataract simulator were provided to the participant The goggles as shown in could be switch to use different filter for experiencing Glaucoma and Cataract. The filter Cataract was made with low transparent plastic paper while the filter for Glaucoma was made with black paper punched with a 1mm hole. There is not standard quantitative measurement for Glaucoma and Cataract, therefore the sight of view is reduced to only allow to see the blur boundary of objects as shown in Figure 66. Comparing the sight of view from different filters and no filterin order to provide very limited information from vision.

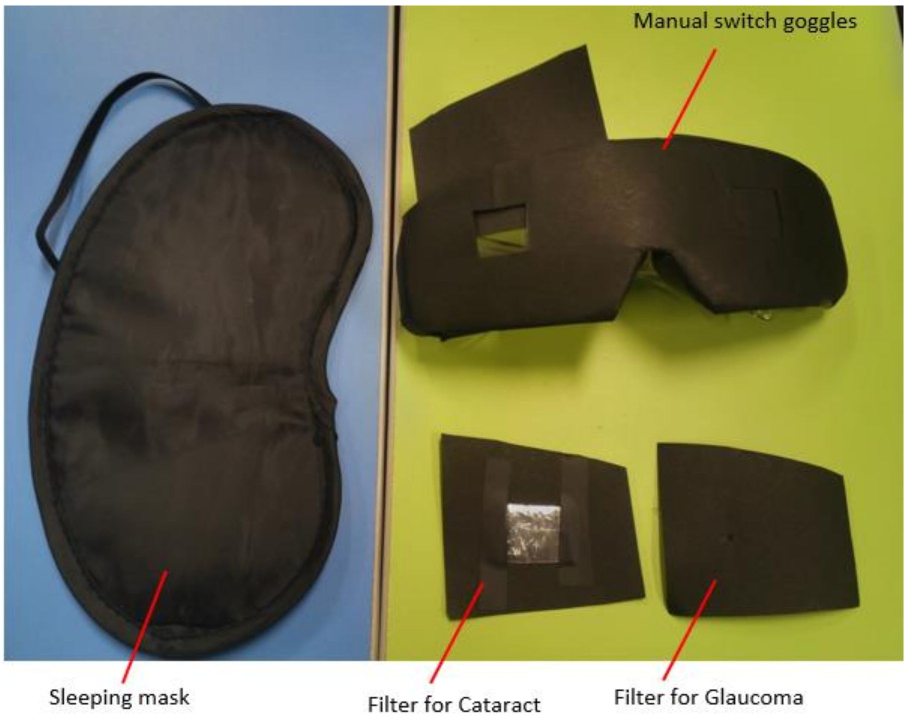


Figure 65. The weak Glaucoma and Cataract simulator while the sleeping mask is for blind

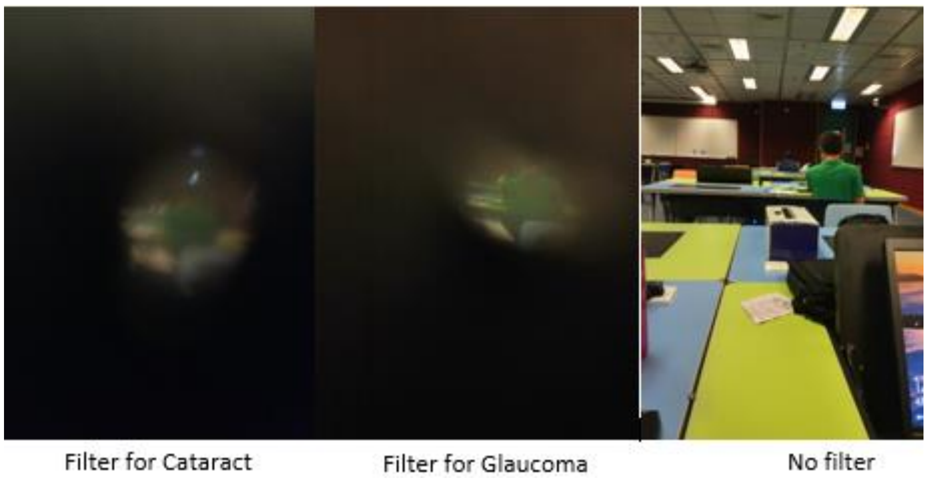


Figure 66. Comparing the sight of view from different filters and no filter

4.1.2 Testing with the routes

The proposed system requires the user know the route to the destination. These analysis are examples of how blind people planning a route. The locations marked with checkpoint in Figure 67. Route from Room 2127B to Hall 1 and Figure 75 are useful for user to aware their location. The checkpoints will be described in Table 10 and Table 11.

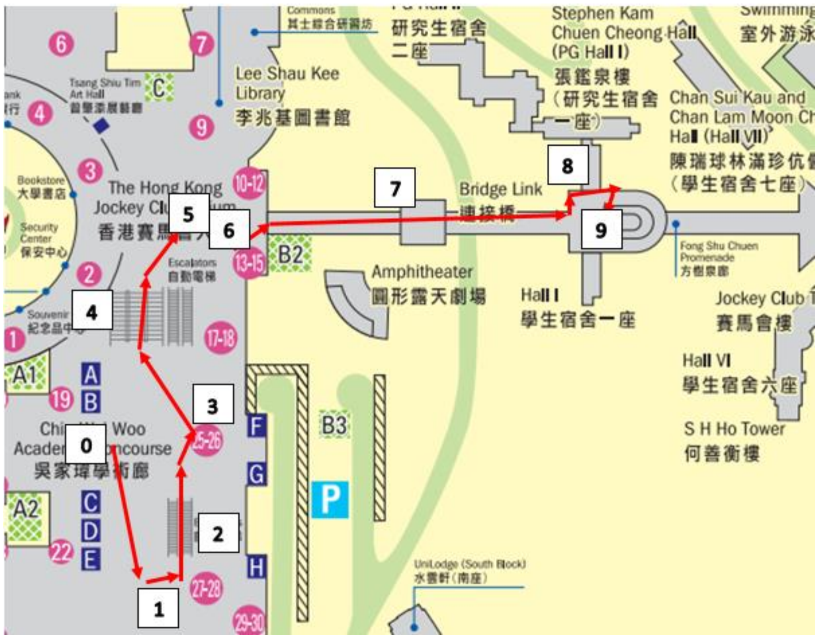


Figure 67. Route from Room 2127B to Hall 1

4.1.2.1 Table 10. Analysis for Route 1

No.	Checkpoint	Description
1	0	This is the starting point outside room2127B in HKUST.
2	0 → 1	Walk straight
3	1	Find out the turn by searching the path on right hand side
4	1 → 2	Turn 10 o'clock then walk forward until hole is detected.
5	2	Walk down the stairs using the handrails.
6	2 → 3	Walk forward until no path is found, then the Café is on the right side.
7	3	Turn right and walk a few steps towards the Café, then turn left and continue walking forward to the Academic Concourse.
8	3 → 4	Walk forward through the Academic Concourse. When there is no path detected, turn right and then find out the turn by searching the path on left hand side. Walk forward until hole is detected.
9	4	Walk up the stairs using the handrails.
10	4 → 5	Turn to 2 o'clock direction and walk through the Atrium until hole is detected.
11	5	Walk down the stairs using the handrails.
12	5 → 6	Turn left and walk toward Learning Common. Turn back if no path is found. Walk until hole is detected.
13	6	Using the elevator to go down. This is all the floors from LG1 to LG5. Handrail will be useful to approach the next elevator. Floor signs of LG3, LG4 and LG5 will be recognized.
14	6 → 7	After finishing the last elevator in LG5, and then turn left and walk forward to until the entrance of bridge. Walk forward until hole is detected.
15	7	Walk down the stairs.
16	7 → 8	Keep walking forward until stop in front of the wall
17	8	Either turn left or right to take the lift. Since the system cannot recognize Lift, user has to either ask for help, use hand or ear to locate and take the lift.
18	8 → 9	Leave the lift, then walk forward to the entry of Hall 1.
19	9	This is the ending point outside Hall 1 entry.

The examples of collision avoidance in Route 1. Arrow is the output of path detection and green area is the detected path.

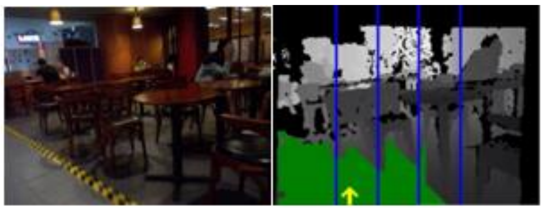


Figure 68. Avoid collision by walking left-center in Checkpoint No.6

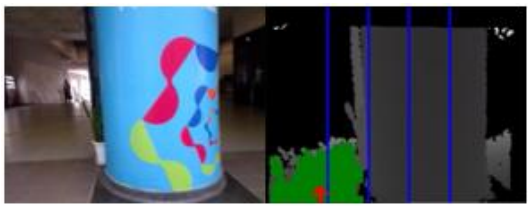


Figure 69. Avoid collision by walking left in Checkpoint No.8

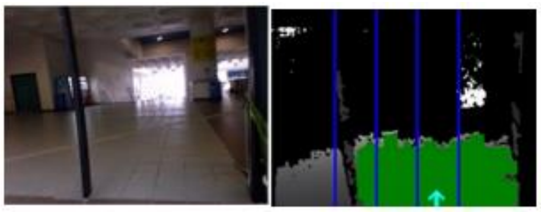


Figure 70. Avoid collision by walking right-center in Checkpoint No.8

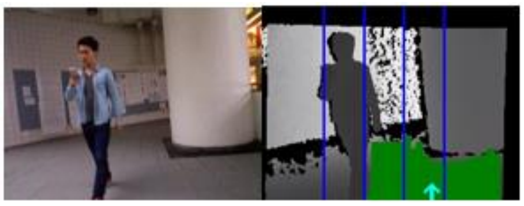


Figure 71. Avoid collision by walking right-center in Checkpoint No.12

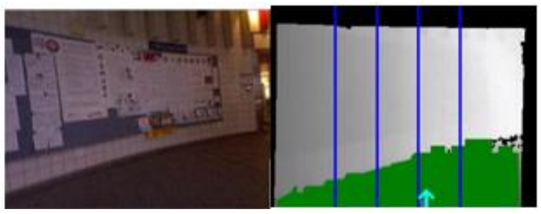


Figure 72. Avoid collision by walking right-center in Checkpoint No.12

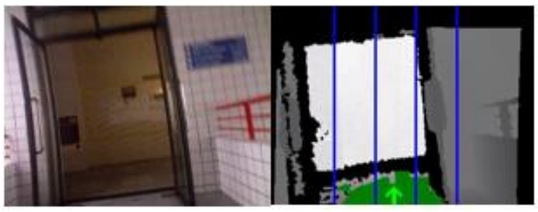


Figure 73. Avoid collision by walking center in Checkpoint No.17

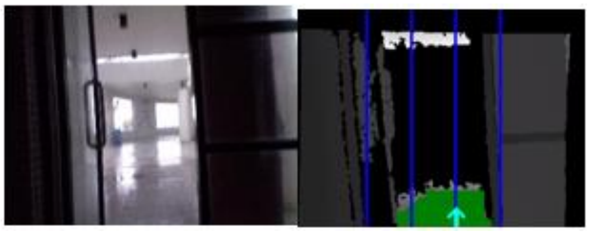


Figure 74. Avoid collision by walking center in Checkpoint No.19

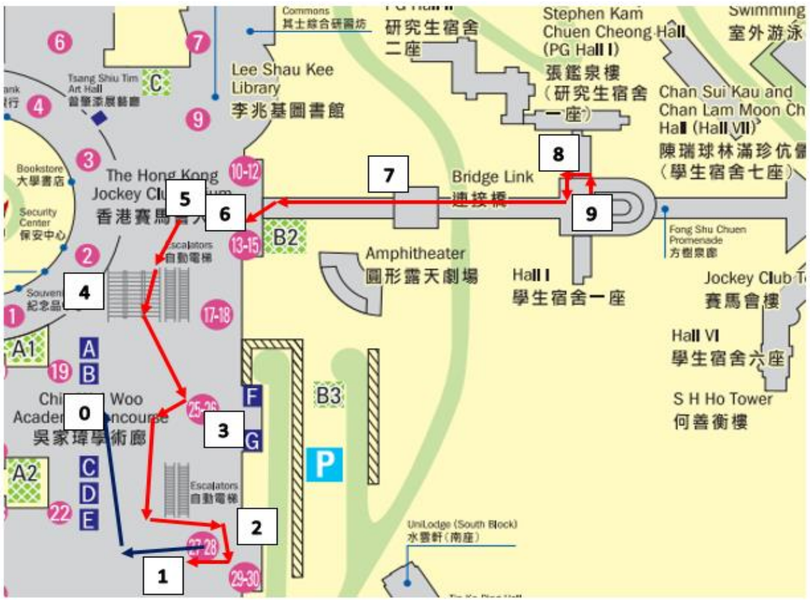


Figure 75 Path from Hall 1 to Room 2127B

4.1.2.2 Table 11. Analysis for Route 2

No.	Checkpoint	Description
1	9	This is the starting point outside Hall 1 entry.
2	9 → 8	Turn right or left and walk forward.
3	8	Since the system cannot recognize Lift, user has to either ask for help, use hand or ear to locate and take the lift. Leave the lift, then turn right or left. Turn left or right when facing a wall. Then exit to the bridge entrance.
4	8 → 7	Walk forward until system alerts user that stairs are found, then walk up the stairs.
5	7	Walking up the stairs.
6	7 → 6	Walk forward until the entrance of the LG5 bridge. Then turn to 2 o'clock until wall is found, then turn left to find the escalator.
7	6	Using the elevator to go down. This is all the floors from LG1 to LG5. Handrail will be useful to approach the next elevator. Floor signs of LG3, LG4 and LG5 will be recognized.
8	6 → 5	Upon reaching LG1, walk slightly forward then turn around to 5 o'clock direction. Walk along the right wall, until stairs are detected on the left side, opposite the right wall.
9	5	Walk up the stairs using the handrails.
10	5 → 4	Turn 10 o'clock direction then walk through the atrium until stairs are found.

11	4	Walk up the stairs using the handrails.
12	4 → 3	Walk forward to Academic Concourse. At the entrance of the concourse, turn to the 11 o'clock direction, then walk forward until the Café, when there is no path detected.
13	3	At the Café, walk right until there is a wall, then turn left and walk forward along the right wall.
14	3 → 2	Keep walking forward along the right wall until there is a corridor on the right side, then turn to 10 o'clock and walk forward. Walk forward until there is a path to the right side, then walk forward. Continue walking forward until there is another path on the right side.
15	2	Walk forward and search for the lift button on the right wall and take the lift going up to second floor.
16	2 → 1	Exit lift 27/28 then turn right and walk forward outside the engineering commons exhibition.
17	1	Turn to 10 o'clock and walk forward until the wall is reached.
18	1 → 0	Walk forward until room 2127B is reached.
19	0	This is the ending point outside room2127B.

The examples of collision avoidance of Route 2. Arrow is the output of path detection and green area is the detected path.

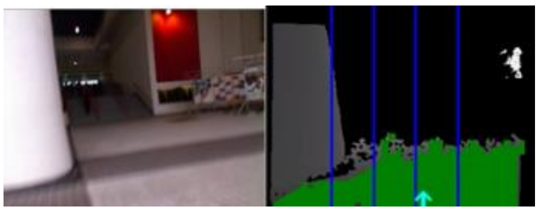


Figure 76. Avoid collision by walking left-center in Checkpoint No.10

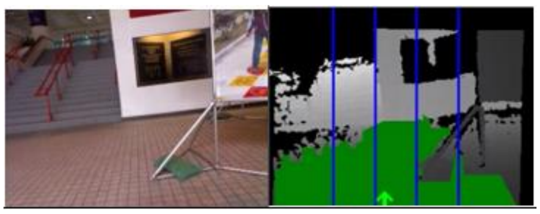


Figure 77. Avoid collision by walking center in Checkpoint No.10

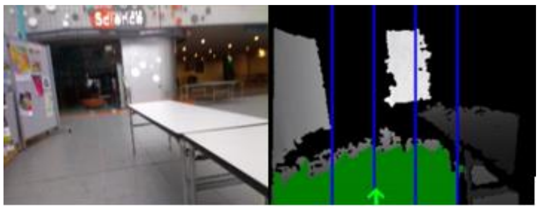


Figure 78. Avoid collision by walking center in Checkpoint No.12

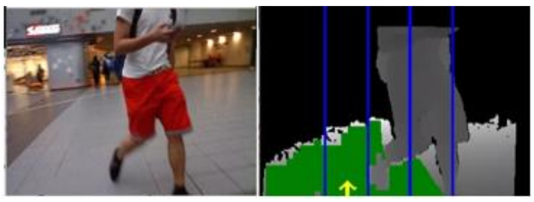


Figure 79. Avoid collision by walking left-center in Checkpoint No.12

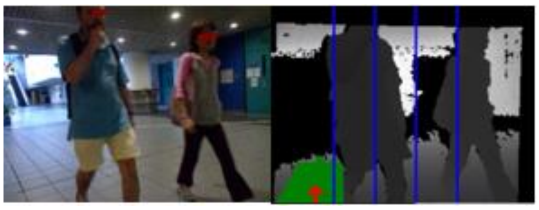


Figure 80. Avoid collision by turning left in Checkpoint No.13

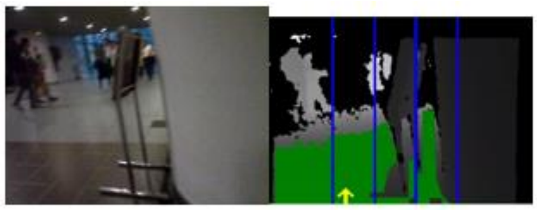


Figure 81. Avoid collision by walking left-center in Checkpoint No.14

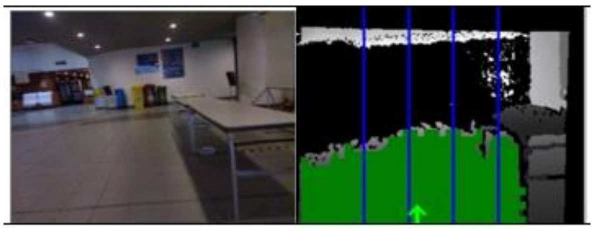


Figure 82. Avoid collision by walking center in Checkpoint No.14

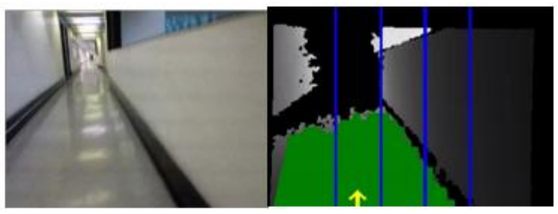


Figure 83. Avoid collision by walking left-center in Checkpoint No.18

4.2 Stair Detection

A frame is considered as a false detection if it wrongly detects stairs when the image does not have stairs in it or when the image has a stair but does not detect it properly. Otherwise, the frame is considered as correct. Hence, accuracy is defined as

$$Accuracy = \frac{total\ \ number\ \ of\ \ corrections}{total\ \ number\ \ of\ \ frames} \times\ 100\%$$

In a sample 20 seconds video with 50 frames:

Recording of Upstairs:

Total Number of detected stairs	50
Number of that are correct	50
Accuracy:	100%

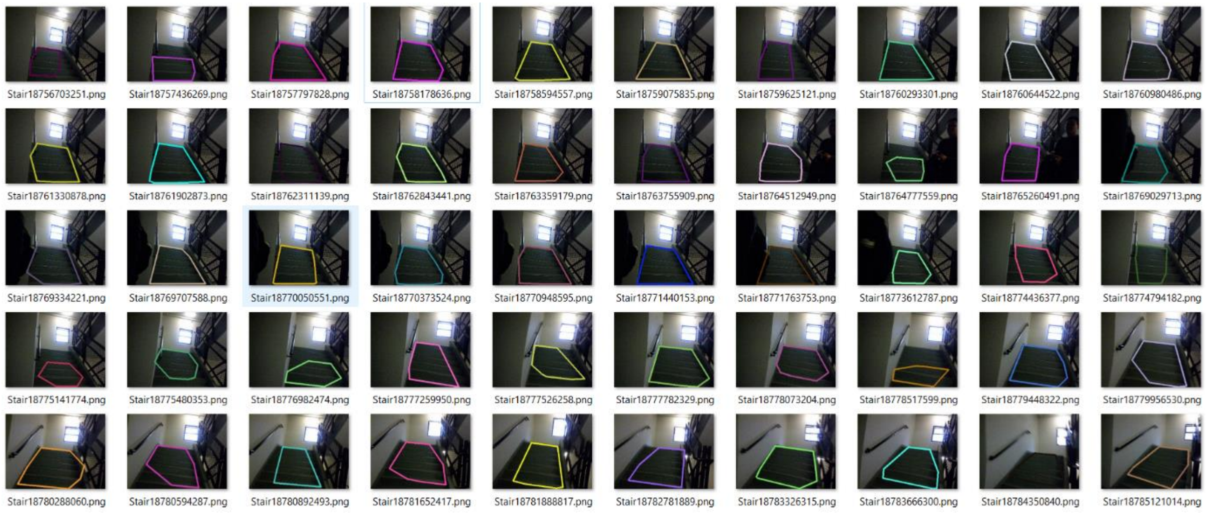


Figure 84: 50 detected upstairs images

Recording of Downstairs

Total Number of detected stairs	50
Number of that are correct	45
Number of undetected stairs	5
Accuracy:	90%

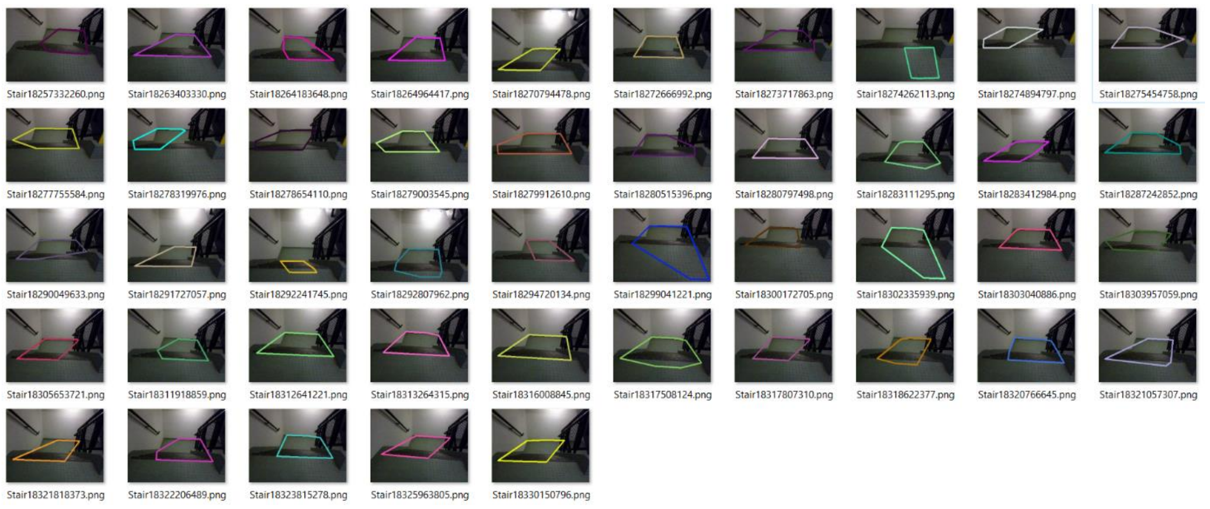


Figure 85: 45 detected downstairs images

However, on a recording using Path Detection's Route 2:

Total Number of Detected Frames	759
Total Number of that are correctly contains stairs	518
Accuracy:	68.24%

The accuracy has dropped down to 68.24% during live testing.

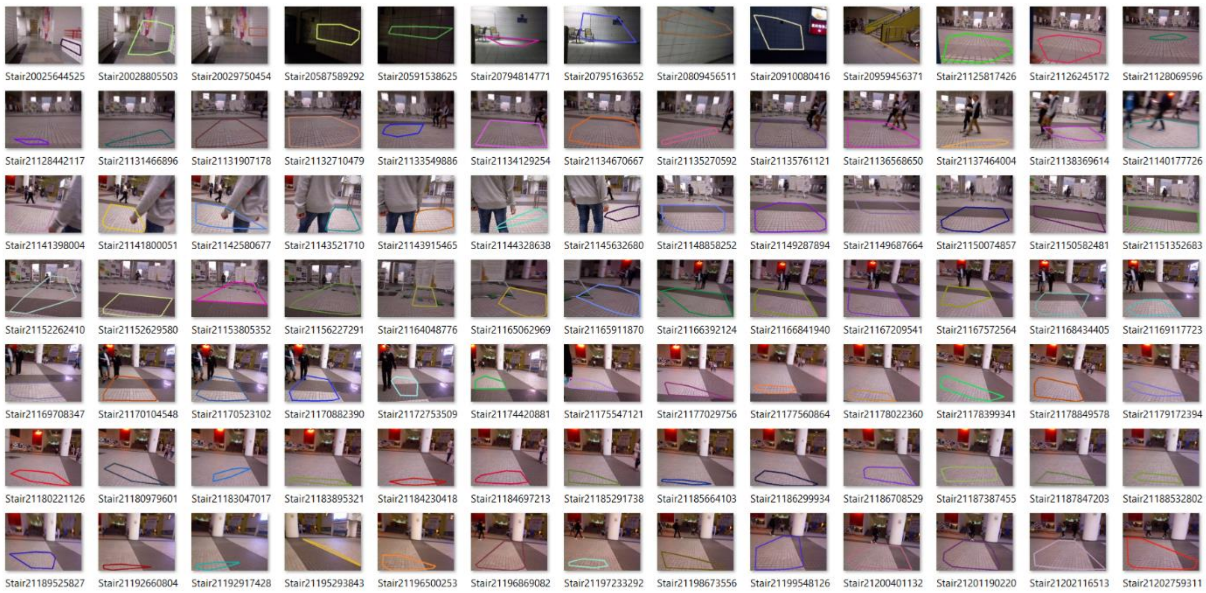


Figure 86: Incorrect detection of stairs

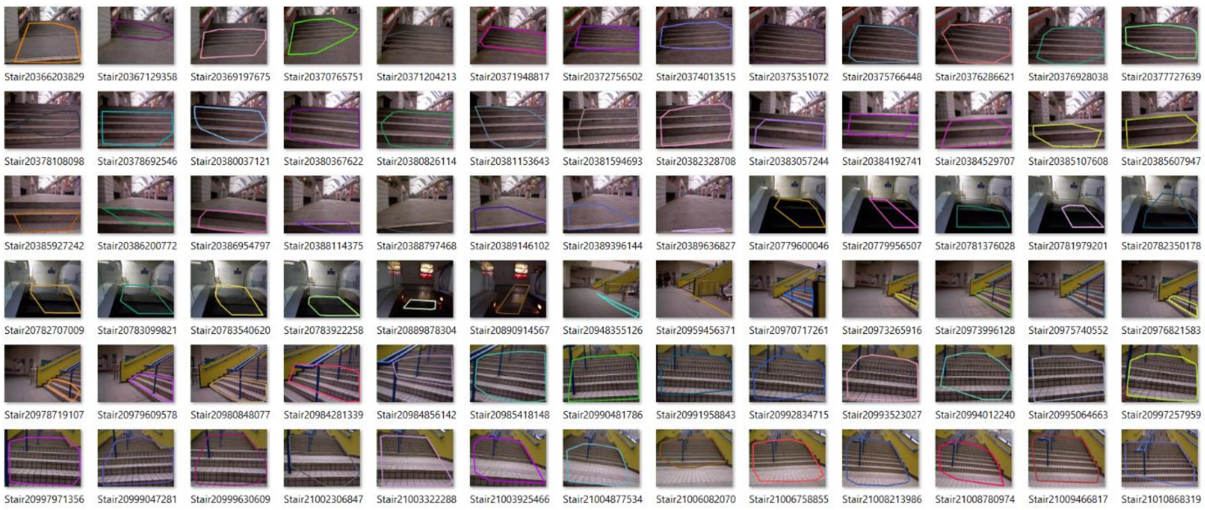


Figure 87 Correctly detected stairs

4.3 Sign Recognition

$$Sign\ \ Detection\ \ Accuracy = \frac{Correct\ \ Detection}{Total\ \ Detection} \times 100\%$$

$$Sign\ \ Recognition\ \ Accuracy = \frac{Correct\ \ Recognition}{Total\ \ Recognition} \times 100\%$$

Testing Set:	Test_Sign_1
Input:	81 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	LG3
Camera Distance:	50cm - 250cm
Brightness:	Normal (stairs)
Sign Angle:	Frontal
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
81	0	0	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
81		0		100%

81 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
80	1	80 / 81 x 100% = 98.8%

Table 12

In short, for Test_Sign_1, the overall accuracy of sign recognition was 100% x 98.8% = 98.8%.

Testing Set:	Test_Sign_2
Input:	73 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	LG3
Camera Distance:	50cm - 350cm
Brightness:	Normal (stairs)
Sign Angle:	Left hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
71	0	2	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
71		2		71/73 x 100% = 97.3%

71 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
69	2	= 97.2%

Table 13

In short, for Test_Sign_2, the overall accuracy of sign recognition was 97.3% x 97.2% = 94.6%.

Testing Set:	Test_Sign_3
Input:	82 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	G Floor
Camera Distance:	50cm - 350cm
Brightness:	Normal (stairs)
Sign Angle:	Left hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
81	0	1	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
81		1		81/82 x 100% = 98.8%

81 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
81	0	100%

Table 14

In short, for Test_Sign_3, the overall accuracy of sign recognition was 98.8% x 100% = 98.8%.

Testing Set:	Test_Sign_4
Input:	82 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	1st Floor
Camera Distance:	50cm - 350cm (From lift to the sign)
Brightness:	Bright (Area near lifts)
Sign Angle:	Frontal
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
82	0	0	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
82		0		100%

82 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
82	0	100%

Table 15

In short, for Test_Sign_4, the overall accuracy of sign recognition was 100% x 100% = 100%.

Testing Set:	Test_Sign_5
Input:	72 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	1st Floor
Camera Distance:	50cm - 450cm (From lift to the sign)
Brightness:	Bright (Area near lifts)
Sign Angle:	Right hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
66	0	6	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
66		6		66/72 x 100% = 91.7%

66 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
66	0	100%

Table 16

In short, for Test_Sign_5, the overall accuracy of sign recognition was 91.7% x 100% = 91.7%.

Testing Set:	Test_Sign_6
Input:	87 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	2nd Floor
Camera Distance:	30cm - 350cm (From lift to the sign)
Brightness:	Bright (Area near lifts)
Sign Angle:	Left hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
85	0	2	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
85		2		85/87 x 100% = 97.7%

85 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
85	85	100%

Table 17

In short, for Test_Sign_6, the overall accuracy of sign recognition was 97.7% x 100% = 97.7%.

Testing Set:	Test_Sign_7
Input:	87 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	2nd Floor
Camera Distance:	50cm - 400cm
Brightness:	Normal (stairs)
Sign Angle:	Left hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
71	0	16	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
71		16		71/87 x 100% = 81.6%

71 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
71	0	100%

Table 18

In short, for Test_Sign_7, the overall accuracy of sign recognition was 81.6% x 100% = 81.6%.

Testing Set:	Test_Sign_8
Input:	68 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	3rd Floor
Camera Distance:	50cm - 400cm
Brightness:	Normal (stairs)
Sign Angle:	bird-eye view (Go downstairs)
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
56	0	12	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
56		12		56/68 x 100% = 82.4%

56 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
51	5	51/56 x 100% = 94.6%

Table 19

In short, for Test_Sign_8, the overall accuracy of sign recognition was 82.4% x 94.6% = 80.0%.

Testing Set:	Test_Sign_9
Input:	99 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	4th Floor
Camera Distance:	50cm - 500cm
Brightness:	Normal (stairs)
Sign Angle:	Elevation (Go upstairs)
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
65	0	34	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
65		34		65/99 x 100% = 65.7%

65 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
54	11	54/65 x 100% = 83.1%

Table 20

In short, for Test_Sign_9, the overall accuracy of sign recognition was 65.7% x 83.1% = 54.6%.

Testing Set:	Test_Sign_10
Input:	75 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	LG5
Camera Distance:	50cm -300cm
Brightness:	Dim
Sign Angle:	Frontal
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
42	0	33	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
42		33		42/75 x 100% = 56%

42 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
40	2	40/42 x 100% = 95.2%

Table 21

In short, for Test_Sign_10, the overall accuracy of sign recognition was 56% x 95.2% = 53.3%.

Testing Set:	Test_Sign_11
Input:	88 frames extracted from a 10s video containing 1 sign/frame
Sign Appear:	LG5
Camera Distance:	50cm - 400cm
Brightness:	Dim
Sign Angle:	Right hand side
Sign Position:	Various on frame

Sign Detection (detecting ellipses)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
30	0	58	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
30		58		30/88 x 100% = 34%

30 True Positives passed to OCR

Sign Recognition (recognizing text)		
Correct Text	***Incorrect Text***	***Accuracy***
29	1	29 / 30 x 100% = 96.6%

In short, for Test_Sign_11, the overall accuracy of sign recognition was 34% x 96.6% = 32.6%.

4.4 Face Detection and Recognition

A list of videos with different condition were recorded in order to test this module. Frames of these videos were first extracted and analyzed one-by-one. Performance of face detection and face recognition were evaluated and the accuracies were calculated by,

$$Face\ \ Detection\ \ Accuracy = \frac{Correct\ \ Detection}{Total\ \ Detection} \times 100\%$$

$$Face\ \ Recognition\ \ Accuracy = \frac{Correct\ \ Recognition}{Total\ \ Recognition} \times 100\%$$

In the following session, the common conditions of testing sets would be stated first, following by a detailed table showing the testing result of one sub-set of the corresponding testing set, as an example. For the other subsets, their summaries would be listed

Table 22 shows the common condition for Testing Set A.

[Test_Face_A] Common Conditions	
Wearing Glasses:	Yes
Camera Distance:	53cm – 265cm (Referring to limitations stated in 3.5.2.1) [Starting from 265cm to 53cm and move back to 265cm far]
Brightness:	Bright
Face Position:	Various on frames

Table 22: Common Condition for Testing Set A

Detailed Result:

Testing Set: *Test_Face_A1*
Input: 100 frames extracted from a 20s video containing 1 face/frame
Faces Appear: Joel (Label = 1)

Frame No	**No of face detected on a frame**	**Face detected samples** (Special cases only)	**Is similar face colour?**	**Is face detection correct?** (after comparing face colour)	**Face Prediction Label**	**Is face recognition correct?**	**Is recognition correct?** (after applying tracking system)
0	1		1	Yes	3	**No**	**No**
1	1		1	Yes	2	**No**	**No**
2	1		1	Yes	2	**No**	**No**
3	2		1	Yes	0	**No**	**No**

			0	Yes	**N/A**	**N/A**	**N/A**
4	1		1	Yes	1	1	1
5	1		1	Yes	1	1	1
6	1		1	Yes	1	1	1
7	1		1	Yes	1	1	1
8	1		1	Yes	1	1	1
9	1		1	Yes	1	1	1
10	1		1	Yes	1	1	1
11	1		1	Yes	1	1	1
12	1		1	Yes	1	1	1
13	1		1	Yes	1	1	1
14	1		1	Yes	1	1	1
15	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
16	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
17	1		1	Yes	1	1	1
18	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
19	1		1	Yes	1	1	1
20	1		1	Yes	2	**No**	**1**
21	1		1	Yes	1	1	1
22	1		1	Yes	1	1	1
23	1		1	Yes	1	1	1
24	1		1	Yes	1	1	1
25	1		1	Yes	1	1	1
26	1		1	Yes	1	1	1
27	1		1	Yes	1	1	1
28	1		1	Yes	1	1	1
29	1		1	Yes	1	1	1
30	1		1	Yes	1	1	1
31	1		1	Yes	1	1	1
32	1		1	Yes	1	1	1
33	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
34	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
35	1		1	Yes	1	1	1
36	1		1	Yes	1	1	1

37	1		1	Yes	1	1	1
38	1		1	Yes	1	1	1
39	1		1	Yes	1	1	1
40	1		1	Yes	1	1	1
41	1		1	Yes	1	1	1
42	1		1	Yes	1	1	1
43	1		1	Yes	1	1	1
44	1		1	Yes	2	***No***	1
45	1		1	Yes	1	1	1
46	1		1	Yes	3	***No***	1
47	1		1	Yes	1	1	1
48	1		1	Yes	1	1	1
49	1		1	Yes	1	1	1
50	1		1	Yes	1	1	1
51	1		1	Yes	1	1	1
52	1		1	Yes	2	***No***	1
53	1		1	Yes	1	1	1
54	1		1	Yes	1	1	1
55	1		1	Yes	2	***No***	1
56	1		1	Yes	2	***No***	1
57	1		1	Yes	1	1	1
58	1		1	Yes	1	1	1
59	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
60	1		1	Yes	2	***No***	1
61	1		1	Yes	2	***No***	1
62	1		1	Yes	1	1	1
63	1		1	Yes	1	1	1
64	1		1	Yes	3	***No***	1
65	1		1	Yes	1	1	1
66	1		1	Yes	2	***No***	1
67	1		1	Yes	1	1	1
68	1		1	Yes	1	1	1
69	1		1	Yes	2	***No***	1
70	1		1	Yes	1	1	1
71	1		1	Yes	1	1	1
72	1		1	Yes	2	***No***	1
73	1		1	Yes	1	1	1

74	1		1	Yes	1	1	1
75	1		1	Yes	1	1	1
76	1		1	Yes	1	1	1
77	1		1	Yes	1	1	1
78	1		1	Yes	1	1	1
79	1		1	Yes	1	1	1
80	1		1	Yes	1	1	1
81	1		1	Yes	3	***No***	1
82	1		1	Yes	1	1	1
83	2		0	Yes	**N/A**	**N/A**	**N/A**
			1	Yes	1	1	1
84	1		1	Yes	2	***No***	1
85	1		1	Yes	1	1	1
86	1		1	Yes	1	1	1
87	1		1	Yes	1	1	1
88	1		1	Yes	2	***No***	1
89	1		1	Yes	2	***No***	1
90	1		1	Yes	2	***No***	1
91	1		1	Yes	2	***No***	1
92	1		1	Yes	2	***No***	1
93	1		1	Yes	1	1	1
94	1		1	Yes	1	1	1
95	1		1	Yes	1	1	1
96	1		1	Yes	2	***No***	1
97	1		1	Yes	2	***No***	1
98	1		1	Yes	1	1	1
99	1		1	Yes	1	1	1

Table 23: A detailed table showing the testing result of Testing subset A1

Here is a summary of the above table for Test_Face_A1 (face testing subset A1).

Testing Set: Test_Face_A1				
Input: 100 frames extracted from a 20s video containing 1 face/frame				
Faces Appear: Joel (Label = 1)				
Face Detection (After comparing face colour)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
94	2	6	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
96		6		96/102 x 100% = 94.1%
Only 94 True Positives passed to LBP face recognizer				
Face Recognition				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
69	0	23	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
69		25		69 / 94 x 100% = 73.4%
Face Recognition (After applying the simple tracking system)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
90	0	4	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
90		4		90 / 94 x 100% = 95.7%

Testing Set: Test_Face_A2
Input: 143 frames extracted from a 20s video containing 1 face/frame
Faces Appear: Lau Ka Ho (Label = 2)

Face Detection (After comparing face colour)

True Positives	*False Negatives*	*True Negatives*	*False Positives*
132	4	11	0

Total no. of correct detection	*Total no. of incorrect detection*	*Accuracy*
136	11	136/147 x 100% = 92.5%

Only 132 True Positives passed to LBP face recognizer

Face Recognition

True Positives	*False Negatives*	*True Negatives*	*False Positives*
108	0	24	0

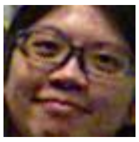
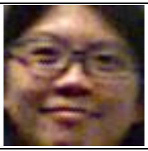
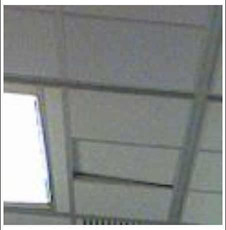
Total no. of correct detection	*Total no. of incorrect detection*	*Accuracy*
108	24	108/132 x 100% = 81.8%

Face Recognition (After applying the simple tracking system)

True Positives	*False Negatives*	*True Negatives*	*False Positives*
130	0	2	0

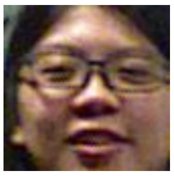
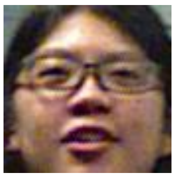
Total no. of correct detection	*Total no. of incorrect detection*	*Accuracy*
130	2	130 / 132 x 100% = 98.5%

Testing Set: Test_Face_A3
Input: 138 frames extracted from a 20s video containing 1 face/frame
Faces Appear: Chan Tong Yan, Yumi (Label = 3)

Frame No	**No of face detected on a frame**	**Face detected samples** (Special cases only)	**Is similar face colour?**	**Is face detection correct?** (after comparing face colour)	**Face Prediction Label**	**Is face recognition correct?**	**Is recognition correct?** (after applying tracking system)
0	1		1	Yes	3	1	1
1	1		1	Yes	3	1	1
2	1		1	Yes	3	1	1
3	1		1	Yes	3	1	1
4	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
5	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
6	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
7	1		1	Yes	3	1	1
8	1		1	Yes	3	1	1
9	1		1	Yes	0	***No***	1
10	1		1	Yes	3	1	1
11	1		1	Yes	3	1	1
12	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
13	1		1	Yes	3	1	1
14	2		1	Yes	3	1	1
			0	Yes	**N/A**	**N/A**	**N/A**
15	1		1	Yes	3	1	1
16	1		1	Yes	3	1	1
17	1		1	Yes	3	1	1
18	1		1	Yes	3	1	1
19	1		1	Yes	3	1	1
20	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
21	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**

22	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
23	1		1	Yes	2	***No***	***No***
24	1		1	Yes	3	1	1
25	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
26	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
27	1		1	Yes	3	1	1
28	1		1	Yes	3	1	1
29	1		1	Yes	3	1	1
30	1		1	Yes	3	1	1
31	1		1	Yes	3	1	1
32	1		1	Yes	3	1	1
33	1		1	Yes	3	1	1
34	1		1	Yes	3	1	1
35	1		1	Yes	3	1	1
36	1		1	Yes	3	1	1
37	1		1	Yes	3	1	1
38	1		1	Yes	3	1	1
39	1		1	Yes	3	1	1
40	1		1	Yes	3	1	1
41	1		1	Yes	3	1	1
42	1		1	Yes	3	1	1
43	1		1	Yes	3	1	1
44	1		1	Yes	3	1	1
45	1		1	Yes	3	1	1
46	1		1	Yes	3	1	1
47	1		1	Yes	3	1	1
48	1		1	Yes	3	1	1
49	1		1	Yes	3	1	1
50	1		1	Yes	3	1	1
51	1		1	Yes	3	1	1
52	1		1	Yes	2	***No***	1
53	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
54	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**

55	1		1	Yes	2	***No***	1
56	1		1	Yes	2	***No***	1
57	1		1	Yes	2	***No***	1
58	1		1	Yes	2	***No***	1
59	1		1	Yes	2	***No***	1
60	1		1	Yes	2	***No***	1
61	1		1	Yes	2	***No***	1
62	1		1	Yes	3	1	1
63	1		1	Yes	2	***No***	1
64	1		1	Yes	3	1	1
65	1		1	Yes	3	1	1
66	1		1	Yes	3	1	1
67	1		1	Yes	3	1	1
68	1		1	Yes	3	1	1
69	1		1	Yes	3	1	1
70	1		1	Yes	3	1	1
71	1		1	Yes	3	1	1
72	1		1	Yes	3	1	1
73	1		1	Yes	3	1	1
74	1		1	Yes	3	1	1
75	1		1	Yes	3	1	1
76	1		1	Yes	3	1	1
77	1		1	Yes	3	1	1
78	1		1	Yes	3	1	1
79	1		1	Yes	3	1	1
80	1		1	Yes	3	1	1
81	1		1	Yes	3	1	1
82	1		1	Yes	3	1	1
83	1		1	Yes	3	1	1
84	1		1	Yes	3	1	1
85	1		1	Yes	3	1	1
86	1		1	Yes	3	1	1
87	1		1	Yes	3	1	1
88	1		1	Yes	3	1	1
89	1		1	Yes	3	1	1
90	1		1	Yes	3	1	1
91	1		1	Yes	3	1	1

92	1		1	Yes	3	1	1
93	1		1	Yes	3	1	1
94	1		1	Yes	3	1	1
95	1		1	Yes	3	1	1
96	1		1	Yes	3	1	1
97	1		1	Yes	3	1	1
98	1		1	Yes	3	1	1
99	1		1	Yes	3	1	1
100	1		1	Yes	3	1	1
101	1		1	Yes	3	1	1
102	1		1	Yes	3	1	1
103	1		1	Yes	3	1	1
104	1		1	Yes	0	***No***	1
105	1		1	Yes	3	1	1
106	1		1	Yes	3	1	1
107	1		1	Yes	3	1	1
108	1		1	Yes	3	1	1
109	1		1	Yes	3	1	1
110	1		1	Yes	2	***No***	1
111	1		1	Yes	3	1	1
112	1		1	Yes	3	1	1
113	1		1	Yes	3	1	1
114	1		1	Yes	3	1	1
115	1		1	Yes	3	1	1
116	1		1	Yes	3	1	1
117	1		1	Yes	3	1	1
118	1		1	Yes	3	1	1
119	1		1	Yes	3	1	1
120	1		1	Yes	3	1	1
121	1		1	Yes	3	1	1
122	1		1	Yes	3	1	1

123	1		1	Yes	3	1	1
124	1		1	Yes	3	1	1
125	1		1	Yes	3	1	1
126	1		1	Yes	3	1	1
127	1		1	Yes	3	1	1
128	1		1	Yes	3	1	1
129	1		1	Yes	3	1	1
130	1		1	Yes	3	1	1
131	1		1	Yes	3	1	1
132	1		1	Yes	3	1	1
133	1		1	Yes	3	1	1
134	1		1	Yes	3	1	1
135	1		1	Yes	3	1	1
136	0	**N/A**	**N/A**	***No***	**N/A**	**N/A**	**N/A**
137	1		1	Yes	3	1	1

Testing Set:	Test_Face_A3
Input:	138 frames extracted from a 20s video containing 1 face/frame
Faces Appear:	Chan Tong Yan, Yumi (Label = 3)

Face Detection (After comparing face colour)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
126	1	12	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
127		12		127/139 x 100% = 87.5%

Only 126 True Positives passed to LBP face recognizer

Face Recognition				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
113	0	13	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
113		13		113/126 x 100% = 89.7%

Face Recognition (After applying the simple tracking system)				
True Positives	***False Negatives***	***True Negatives***	***False Positives***	
125	0	1	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
125		1		125/126 x 100% = 99.2%

[Test_Face_B] Common Condition	
Wearing Glasses:	No
Camera Distance:	53cm – 265cm (Referring to limitations stated in 3.5.2.1) [Starting from 265cm to 53cm and move back to 265cm far]
Brightness:	Bright
Face Angle:	Frontal
Face Position:	Various

Testing Set:	Test_Face_B1
Input:	128 frames extracted from a 20s video containing 1 face/frame
Faces Appear:	Joel (Label = 1)

Face Detection (After comparing face colour)

True Positives	*False Negatives*	*True Negatives*	*False Positives*	
122	1	6	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
123		6		123/129 x 100% = 95.3%

Only 122 True Positives passed to LBP face recognizer

Face Recognition

True Positives	*False Negatives*	*True Negatives*	*False Positives*	
91	0	31	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
91		31		91/122 x 100% = 74.6%

Face Recognition (After applying the simple tracking system)

True Positives	*False Negatives*	*True Negatives*	*False Positives*	
120	0	2	0	
Total no. of correct detection		***Total no. of incorrect detection***		***Accuracy***
120		2		120 / 122 x 100% = 98.4%

Testing Set:	Test_Face_B3
Input:	144 frames extracted from a 20s video containing 1 face/frame
Faces Appear:	Chan Tong Yan, Yumi (Label = 3)

Face Detection (After comparing face colour)

True Positives	***False Negatives***	***True Negatives***	***False Positives***
135	1	9	0

Total no. of correct detection	***Total no. of incorrect detection***	***Accuracy***
136	9	136/145 x 100% = 93.8%

Only 135 True Positives passed to LBP face recognizer

Face Recognition

True Positives	***False Negatives***	***True Negatives***	***False Positives***
95	0	40	0

Total no. of correct detection	***Total no. of incorrect detection***	***Accuracy***
95	40	95/135 x 100% = 70.4%

Face Recognition (After applying the simple tracking system)

True Positives	***False Negatives***	***True Negatives***	***False Positives***
132	0	3	0

Total no. of correct detection	***Total no. of incorrect detection***	***Accuracy***
132	3	132/135 x 100% = 97.8%

4.5 Integrated system

Appendix G: Video 1&2 were used to get the running times of the algorithms after integrating into multithreading structure, as shown in Table 24.

Algorithm	Sign	Path	Stair	Face
Average (s)	0.1243	0.0659	0.0704	1.1355
Min (s)	0.0830	0.0254	0.0121	0.3242
Max (s)	0.5462	0.1785	0.5777	1.3323
S.D.	0.0474	0.0220	0.0511	0.1550
Sample Size (frame)	1888	9706	10485	781
Total Seconds used from Route 2 recording	234.6792	639.5991	737.7052	886.7892

Table 24 shows the evaluated running times of the algorithms.

5 Evaluation

5.1 Path Detection

Please find the Appendix J: Statistic for Route 1 and Route 2. detail data record for time and distance. The speed was calculated for estimate the performance of the proposed system in each checkpoint.

The average walking speed was calculated by $\frac{Total\ distance\ of\ Route\ 1\ and\ Route\ 2}{total\ time}$

Participant with emulator	Average walking speed(meter/minute)
Blind	18.00
Cataract	23.57
Glaucoma	27.22

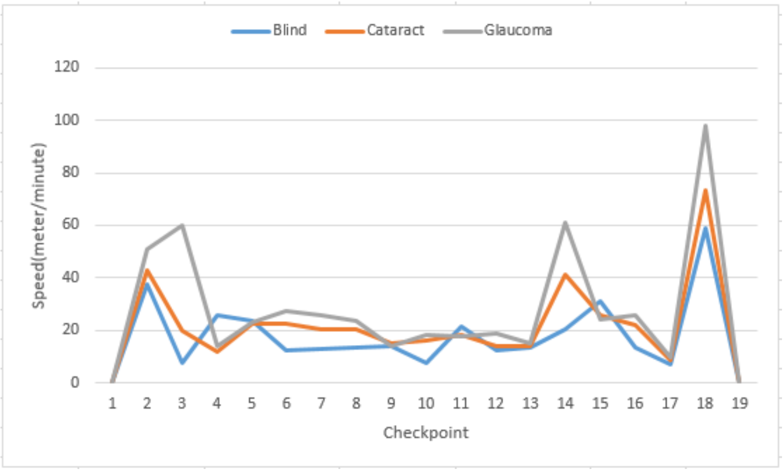


Figure 88. Graph of participant's walking speed in each checkpoint on route 1

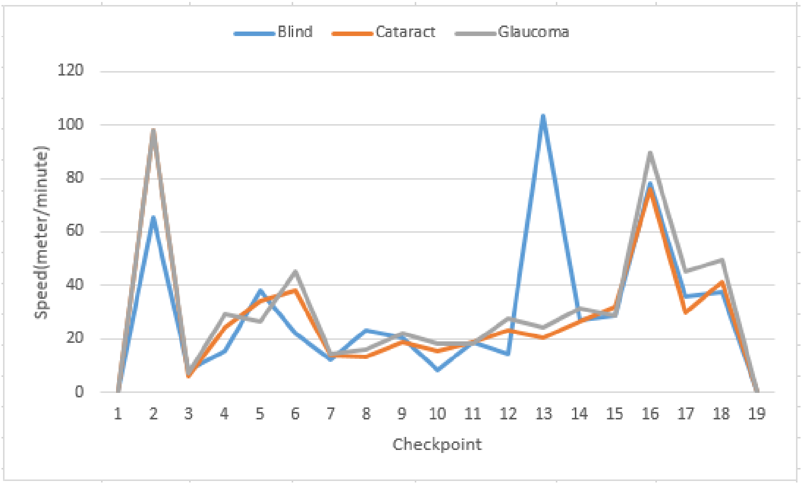


Figure 89. Graph of participant's walking speed in each checkpoint on route 2

Testing result

The output of Path Detection could help the serious visually impaired people to complete the path without other people's help. However, completely blind people needed some help when they were walking in No.6, 8 and 10 checkpoints on Route 1, No.10,12 and 14 checkpoints on Route 2. The area of these checkpoints were very board so blind people could get lost direction sometime since the proposed system could not provide guarantee path information to the destination. However, it could provide good direction to avoid collision during the test.

Users can search wall or big static obstacle by using the output of Path if there is no path.

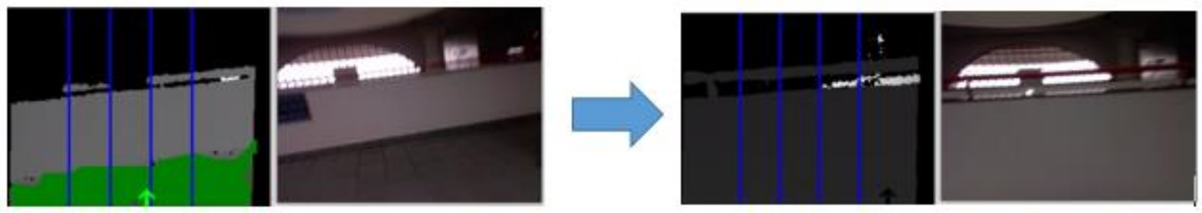


Figure 90 Avoid collision by stop walk in Checkpoint No.12

When users are walking stairs, there will no vibration at all since there surface area on stair is higher or smaller than threshold for ground's height. Before finishing the stair, the device will start detecting path which will a good information for user to be careful of body balance.

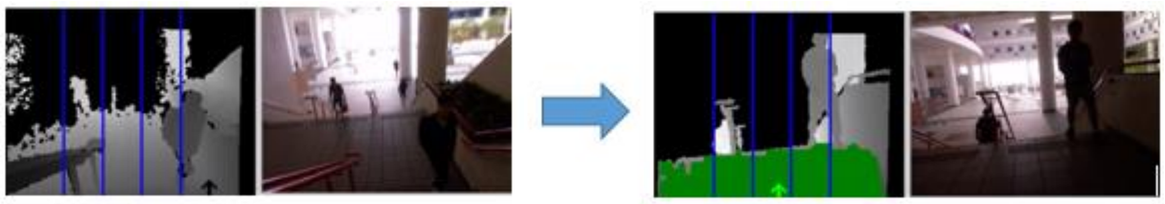


Figure 91. Path detected before finishing the down stair

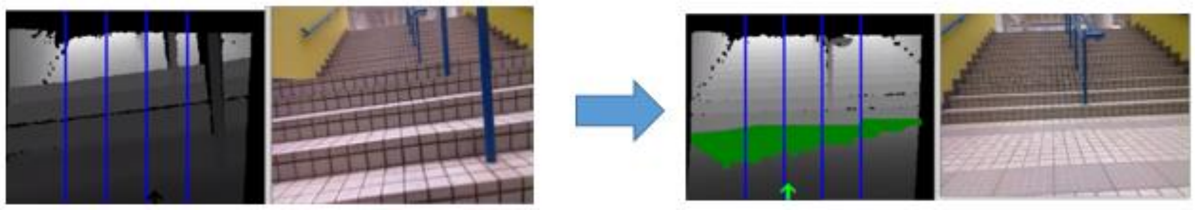


Figure 92. Path detected before finishing the up stair

5.2 Stair Detection

Stairs detection is quite reliable with average 95% detection rate when stairs are actually there. However, the detection algorithm is very much affected by tiled floors. The accuracy detection rate drops down to 68.24%.

In practice, the 68.24% accuracy is quite acceptable because the actual output to user is not continuous when the user moves or shifts the walking direction after being alerted. Hence, false detections would only have one or two output alerts to the user, while correctly detected stairs would have a continuous rate of output alerts to the user.

5.3 Sign Recognition

For the performance of sign recognition, it identified signs well under normal or bright light condition, with mostly over 90% accuracy at all viewing angles.

However, when the device moved downward or upward, the recognition accuracy decreased to 55% - 80%. It might be because, when users walking downstairs or upstairs, the degree of camera vibration was larger than that of walking on flat floor, leading to larger degree of frame distortion. Figure 93 shows an example of distorted frame

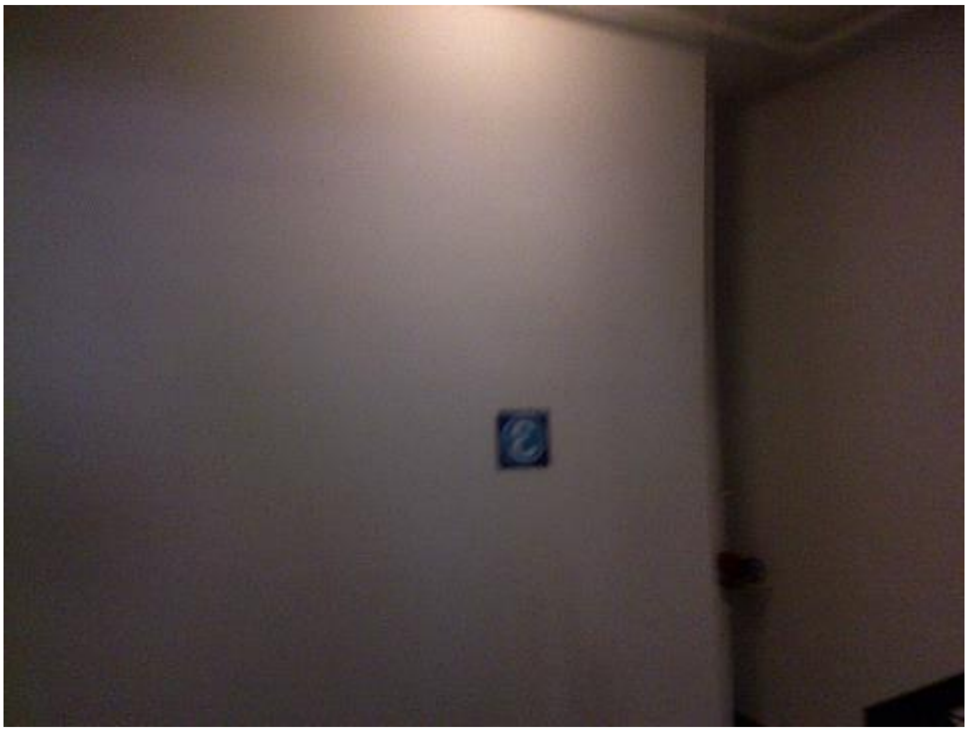


Figure 93: Example of distorted frame

On the other hand, under low light condition, frames received from Kinect contained much more noise than that in normal or bright light condition. This led to the low accuracy and performance of detecting circular contour of signs.

In short, sign recognition could identify signs normally, but when under low light condition, users may need to move steadly and stand closer in order to obtain a reliable recognition result.

5.4 Face Recognition

In our testing, the performance and accuracy of face recognition was good and not many wrong predictions appeared with the simple tracking system. Since the tracking system only notify users when 3 times of the same predictions appear, it avoided incorrect prediction appeared occasionally. For the testing conducted, the tracking system improved the accuracy from (73.4% + 81.8% + 89.7% + 74.6%+ 68.5% +70.4%) / 6 = 76.4% to (95.7% + 98.5% + 99.2% + 98.4% + 83.7% + 97.8%)/6 = 95.6%.

Moreover, both people wearing glasses or without wearing glasses could also be recognized.

In short, users friend faces could be accurately identified by the system.

6 Discussion

6.1.1 The defect of using partitioning in finding path direction and solution

In the progress report, it was proposed to use partitioning method to find the longest path for advising the path direction which could guarantee the path direction was safe. However, the path advice would be unstable when there was no obstacle as a result of confusing the user. Therefore, center of mass of the ground was used to find the path direction as descripted in 2.2.3 Path Direction.

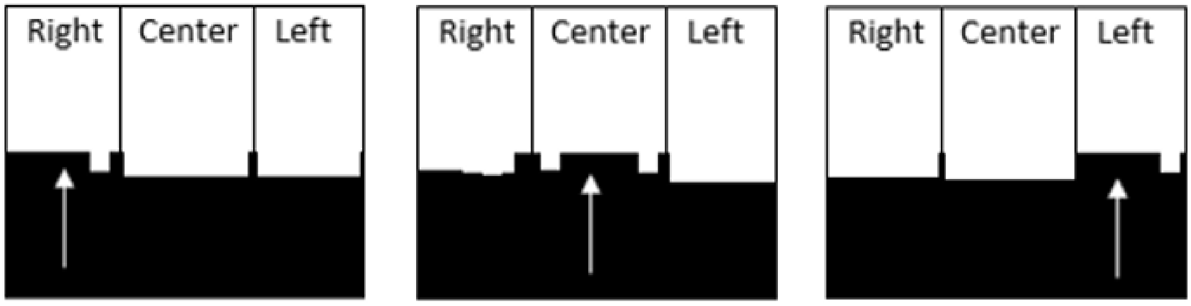


Figure 94: The unstable path advice

6.1.2 Defect of using camera angle in calculating height and suggestion

The retrieved angle using Kinect's SDK is the Kinect angle in the center of Kinect device, however, shaking during movement causes the image rotate θ in roll as result the height of pixel calculated in the depth image is not correct. It is suggested to use a inertial measurement unit to obtain θ and calculate the compensation of angle of view in the image before calculating the height.

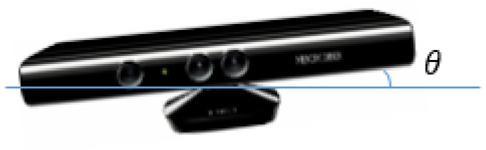


Figure 95 Roll rotation of device during shaking

6.1.3 Limitation of Kinect's view for shortest distance

Every depth frame provide valid depth value if the actual distance greater than 1.4 meter, therefore, obstacles within this range are unseen. To use Kinect in the proposed system, requiring user to start the device within an obstacle free circle with radius 1.4 meter.

7 Conclusion

The wearable system provides visually impaired people with navigational and social enhancement services. The overall performance of the system is acceptable as the accuracies of single image frames are important for detecting and recognizing at that particular instance, while in practice, it generally produces understandable and correct input to users that can react to. Nevertheless, there are plenty of improvements that can be made to achieve better results and a better wearable system.

Recommendation

1. Port to embedded system

Since the prototype wearable system is bulky and conspicuous, visually impaired people may not want to use it yet. The software detection system can be exported to a more portable hardware system, such as using a Raspberry Pi, smartphones and smartwatches, which can replace the laptop, vibration belt and audio speaker system.

8 References

[1] Census and Statistics Department, Hong Kong Special Administrative Region, “Social data Collected via the General Household Survey - Special Topics Report No.62: Persons with disabilities and chronic diseases,” Dec. 2014 [Online]. Available: http://www.statistics.gov.hk/pub/B11301622014XXXXB0100.pdf

[2] World Health Organization, “Global data on visual impairment 2010,” [Online]. Available: http://www.who.int/blindness/GLOBALDATAFINALforweb.pdf?ua=1

[3] C. H. Wong, *低視能人士調查報告* [Survey Report on People with Low Vision], Retina Hong Kong, Aug. 2002 (in Chinese).

[4] L. K. Fok, *人物素描: 突破宿命、自力更生的原玲* [An interview with a visually impaired masseur], Retina Hong Kong, Sep. 2003 (in Chinese).

[5] M. Zöllner, S. Huber, H. C. Jetter and H. Reiterer, “NAVI - A Proof-of-Concept of a Mobile Navigational Aid for Visually Impaired Based on the Microsoft Kinect,” University of Konstanz, Germany, 2011.

[6] G. Balakrishnan, G. Sainarayanan, R. Nagarajan and S. Yaacob, “A Stereo Image Processing System for Visually Impaired,” *World Academy of Science, Engineering and Technology*, vol. 2, no. 8, pp. 2794-2803, 2008.

[7] X. Wei, S. L. Phung and A. Bouserdoum, "Scene segmentation and pedestrian classification from 3-D range and intensity images," in 2012 IEEE International Conference on Multimedia and Expo (ICME), Melbourne, VIC, 2012.

[8] S. Wang and Y. Tian, “Detecting Stairs and pedestrian crosswalks for the blind by RGBD camera,” *Bioinformatics and Biomedicine Workshops (BIBMW), 2012 IEEE International Conference*, pp. 732-739, 4 Oct 2012.

[9] A. W. Fitzgibbon and R. B. Fisher, "A Buyer’s Guide to Conic Fitting," Department of Aritificial Intelligence, Edinburgh University, 1995.

[10] T. Ahonen, A. Hadid and M. Pietikainen, “Face Recognition with Local Binary Patterns,” 2004.

Appendix A: Minutes

A.1 Minutes of the 1st FYP meeting

Date: 19/08/2015

Time: 1300

Place: HKUST library LC10

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

1. **Approval of minutes**

   None

2. **Report on progress**

   2.1. Java is not well supported by OpenCV and OpenGL. So we decided to use C++, Visual C++ 2013 as IDE and Microsoft Kinect SDK 1.7. It seems that we can use JNI / NDK to reuse our coding for android version.

   2.2. So far we brought about HKD200 electronic staff for modify the power supply of Kinect to DC.

3. **Discussion items**

   3.1. Proposal Introduction draft

      3.1.1. Overview draft

         3.1.1.1. Statistics about visual impair people

         3.1.1.2. General problem of visual impair people

         3.1.1.3. Proposal aim to help visual impair people

      3.1.2. Objective draft

         3.1.2.1. Suggesting improvement according to the past research

         3.1.2.2. Challenges plan to target

      3.1.3. Literature Survey

         3.1.3.1. Binsy N Rashad , Nishadha S.G, "Artificial Vision for the Blind Using Motion Vector Estimation Technique", *International Journal of Innovative Research in Science, Engineering and Technology* , Volume 3, Special Issue 5, July 2014,India

   3.2. We discussed and concluded brief direction of the system flow

      3.2.1. Using video camera and depth camera

      3.2.2. To set up a safety distance. Detection process start if object enter the safety distance.

      3.2.3. Separate the input frame into two parts by a horizon line.

      3.2.4. Initially, all objects are defined as obstacle.

        3.2.5. Recognition will start for up or down stairs first.
        3.2.6. If obstacles pass the alert distance, text-to-speech will notify the user
        3.2.7. Alternative object recognition will be added if possible
    3.3.

4. Goals for the coming week

4.1. Each of us continue to study more research paper and share in the next meeting

5. Meeting adjournment and next meeting

The meeting was adjourned at [1900].

The next meeting will be on 19/08/2015. The place is to be decided later

A.2 Minutes of the 2nd FYP meeting

Date: 24/08/2015

Time: 1500

Place: Learning Common LC-10

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: none

Recorder: Tong-yan Chan

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   - **1.1.** Kinect power supply modification is done
   - 1.2. Kinect data can be recorded by Kinect Studio
   - **1.3.** A project included Kinect library is created with Visual Studio
   - **1.4.** Papers and references to be included:
     - **1.4.1.** NAVI
     - **1.4.2.** Obstacle-Free Pathway Detection by Means of Depth Maps
2. **Discussion items**
   - 2.1. Convert depth data obtained from Kinect into grey-scale images for later use
   - 2.2. Things to add:
     - 2.2.1. Face detection
     - 2.2.2. Hand gesture
     - 2.2.3. Change the vibration wristbands to a belt
     - 2.2.4. Path advisor, with reference to a past FYP using accelerometer (not confirmed yet)
     - 2.2.5. Social media update (not confirmed y

3. **Goals for the coming week**
   - 3.1. Email the communication tutors questions about the proposal
     - 3.1.1. Should we include things that are not confirmed to implement in the proposal?
     - 3.1.2. Can we implement things that are not mentioned in the proposal?
     - 3.1.3. Should we put the system flow in Design?
     - 3.1.4. What is the format of minutes?
   - 3.2. Write a draft of the proposal

4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1900].

   The next meeting will be on 07/09/2015. The place is to be decided later

A.3 Minutes of the 3rd FYP meeting

Minutes of the 2nd Project Meeting

Date: 07/09/2015

Time: 1430

Place: Rm3142

Attending: Prof. Albert Chung, Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Joel Berago

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

   2.1. Kaho explained that there exist two similar projects to our idea.

   2.2. Prof. Chung told us to set our own goals first.

   2.3. Kaho explained our goals.

      2.3.1. Want to help visually impaired people.

      2.3.2. Inform users where obstacles are located relative to the user.

      2.3.3. Notify user by sound and vibrations.

      2.3.4. Enhance users' social life by facial recognition.

   2.4. Prof. Chung responded that it is reasonable to have two focuses, safety and social aspects.

   2.5. Prof. Chung advised us that obstacle recognition does not need to be a very general system. Only define some scenarios.

   2.6. Prof. Chung asked our division of work.

      2.6.1. Kaho answered by explaining the flow of our system.

      2.6.2. First, we will set a safety distance and every object greater than the safety distance is ignored. Then do segmentation to find the floor and remaining obstacles.

      2.6.3. Classify everything as obstacle, then try to recognize if it is a specific type of obstacle to be detected.

      2.6.4. Then we notify user about the obstacle.

   2.7. Prof. Chung asked how our system is worn

      2.7.1. Joel answered that it is either mounted on a helmet or a belt.

   2.8. Prof. Chung suggested to try and test the systems first. He wants to have something that works.

   2.9. Prof. Chung explained that he does not expect us to build every single part. Using existing solutions but we need to acknowledge and understand the properties of the solution and why it is an acceptable solution.

2.10. Kaho also mentioned that we want to have hand gesture and text-to-speech for UI as input. Kinect can already support these.

3. **Discussion items**

   None

4. **Goals for the coming week**

   4.1. Prof. Chung advised us to find out more about the needs of visually impaired people. Find interviews or news articles.

   4.2. Prof. Chung encouraged us to work on the details of the algorithms used and focus on the basic goals first.

   4.3. Prof Chung advised us to look at CNN for object recognition.

5. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1515].

   The next meeting will be on 04/10/2015. The place is to be decided later

A.4 Minutes of the 4th FYP meeting

Date: 04/10/2015

Time: 1500

Place: Learning Common LC-1

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Tong-yan Chan

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   1.1. KaHo has test the histogram-based segmentation
   1.2. Yumi has tried to use some OCR library
   1.3. Joel has tried using Hough line transform to detect straight line
2. **Discussion items**
   2.1. Microsoft Kinect SDK provides a good sample project, KinectBrigdeOpenCV project, for us to as a base project start our coding for our early development.
   2.2. Gaussian Blur should be used instead of mean filter in segmentation
   2.3. Floor signs in HKUST need to be recognized, for example, LG1, LG5, 4, etc.

3. **Goals for the coming week**
   3.1. To collect color images and depth images for testing the code in next week

4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1900].

   The next meeting will be on 12/11/2015. The place is to be decided later

A.5 Minutes of the 5th FYP meeting

Date: 12/11/2015

Time: 1900

Place: Comp Lab RM4213

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   - 1.1. Kaho has studied Delaunay triangulation.
   - 1.2. Yumi has tested detecting circles in the sign using Hough circle transform.
2. **Discussion items**
   - 2.1. Sample data needs to be recorded live from the sample project. A recording function needs to be implemented into the sample project by ourselves.
3. **Goals for the coming week**
   - 3.1. Joel will add the image capturing function in the project for collecting images for testing purposes.
   - 3.2. Yumi will find another solution to detect circles.
4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [time].

   The next meeting will be on 05/12/2015. The place is to be decided later

A.6 Minutes of the 6th FYP meeting

Date: 05/12/2015

Time: 1400

Place: Comp Lab RM4213

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   - 1.1. KaHo improved the histogram segmentation algorithm by adding a 1D Gaussian Blur to smooth the depth Histogram.
   - 1.2. Joel studied the Microsoft Kinect bridge OpenCV sample project and added an image capturing function for saving the color and depth image captured by Kinect.
   - 1.3. Yumi changed the algorithm to detect ellipse instead of circles. It is detected by using conic fitting.
2. **Discussion items**
   - 2.1. Floor signs and obstacles and ground images need to be indoors.
3. **Goals for the coming week**
   - 3.1. Improve the sign recognition, stairs detection and ground detection base on the collected images.
4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1600].

   The next meeting will be on 25/12/2015. The place is to be decided later

A.7 Minutes of the 7th FYP meeting

Date: 27/12/2015

Time: 1500

Place: Comp Lab RM4213

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   1.1. Joel found that the ground region generated from obstacle detection can be used to remove unnecessary lines in the stair image after applying canny edge detector.
   1.2. KaHo can find the ground normal in the image by using cross product.
2. **Discussion items**
   2.1. Yumi suggested to change the color image from 640x320 to 1280x960 since more accurate results can be achieved for sign and face recognition.
   2.2. Yumi collected hundreds of color face image of Yumi, KaHo and Joel for face recognition training.
   2.3. Joel suggest to add OpenNI library in the project for easier control and access of the Kinect.
3. **Goals for the coming week**
   3.1. KaHo will create an obstacle mask to reduce the false positive results of ground normal.
   3.2. Joel will study OpenNI library and study how to get the height of pixel from the depth image.
4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1700].

   The next meeting will be on 09/01/2015. The place is to be decided later

A.8 Minutes of the 8th FYP meeting

Date: 09/01/2015

Time: 1000

Place: Comp Lab RM4213

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   1.1. Joel has created a new project which can use OpenNI, Microsoft Kinect SDK, OpenCV, Tesseract OCR and Microsoft Speech SDK.
   1.2. KaHo has started to tune the parameters for creating an obstacle mask
   1.3. Yumi has trained the face recognition algorithm with the collected face images.
   1.4. Yumi has finished the text-to-speech implementation.
2. **Discussion items**
   2.1. All the algorithms need to be moved and integrated into the new project
   2.2. Depth data structure in OpenNI SDK needs to be tested and the existing algorithms need to have modifications to use that depth data format.
   2.3. More face image need to be collected for training the face recognition algorithm
   2.4. The calculation of pixel height in depth image needs to be further researched.
3. **Goals for the coming week**
   3.1. Joel will code the pixel height calculation
   3.2. KaHo will design the strategy for path finding
   3.3. Yumi will improve the accuracy of sign and face recognition
4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1200].

   The next meeting will be on 20/01/2015. The place is to be decided later

A.9 Minutes of the 9th FYP meeting

Date: 20/01/2015

Time: 1300

Place: Comp Lab RM4213

Attending: Tong-yan Chan, Ka-ho Lau, Joel Berago

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   1.1. Joel and KaHo added the stair detection and obstacle detection algorithm in the new project and test to execute them together.
   1.2. Joel added the Multithreading structure and text-to-speech function into the new project.
   1.3. Yumi improved the sign recognition
2. **Discussion items**
   2.1. The Multi-threading should add timestamp checking for frame synchronization.
   2.2. Stair detection may be separated out from the obstacle detection thread to improve the processing speed of obstacle detection thread
3. **Goals for the coming week**
   3.1. Joel will add timestamp checking function for the frame synchronize purpose
   3.2. Joel will improve height calculation function
   3.3. Yumi and KaHo will improve the accuracy of the obstacle detection, sign and face recognition.
4. **Meeting adjournment and next meeting**

The meeting was adjourned at [1430].

The next meeting will be on 02/02/2015. The place is at Rm3142 with Prof. Albert Chung.

A.10 Minutes of the 10th FYP meeting

Date: 02/02/2015

Time: 1600

Place: Rm3142

Attending: Prof. Albert Chung, Tong-yan Chan, Ka-ho Lau, Joel Berago,

Absentees: None

Recorder: Ka-ho Lau

Approval of minutes

The minutes of the last meeting were approved without amendment.

1. **Report on progress**
   - 1.1. Joel presented the design and implementation of the stair detection to Prof. Albert Chung
   - 1.2. KaHo presented the design and implementation of path finding to Prof. Albert Chung
   - 1.3. Yumi presented the design and implementation of sign recognition and face recognition to Prof. Albert Chung
2. **Discussion items**
   - 2.1. The noise problem in depth image may be solved by getting helpful information from color image
   - 2.2. Our group should start to plan the venue to do the demonstration in FYP presentation
3. **Goals for the coming week**
   - 3.1. Start to integrate the current document to the progress report
   - 3.2. Plan to integrate
4. **Meeting adjournment and next meeting**

   The meeting was adjourned at [1700].

   The next meeting will be after the progress report has been reviewed.

Appendix B: Project Planing

B.1 Division of Work

	Ka-ho Lau	Tong-yan Chan	Joel Berago
Requirement Analysis and Information Research	X	X	X
Needs Analysis	X	X	
Kinect power supply modification	X		
Kinect-system interface			X
Obstacle detection	X		X
Stairs detection			X
Sign recognition		X	
Face recognition		X	
System flow design	X	X	X
Text-to-speech		X	
Vibration belt	X		
User Input		X	X
System Integration	X	X	X
Proposal Report	X	X	X
Progress Report	X	X	X
Regular Reports	X	X	X
System Testing and Debugging	X	X	X
Evaluation	X	X	X
Final Report and Project Poster	X	X	X
Presentation and Demonstration	X	X	X

Table 25:Division of Work Table

B.2 Gantt Chart

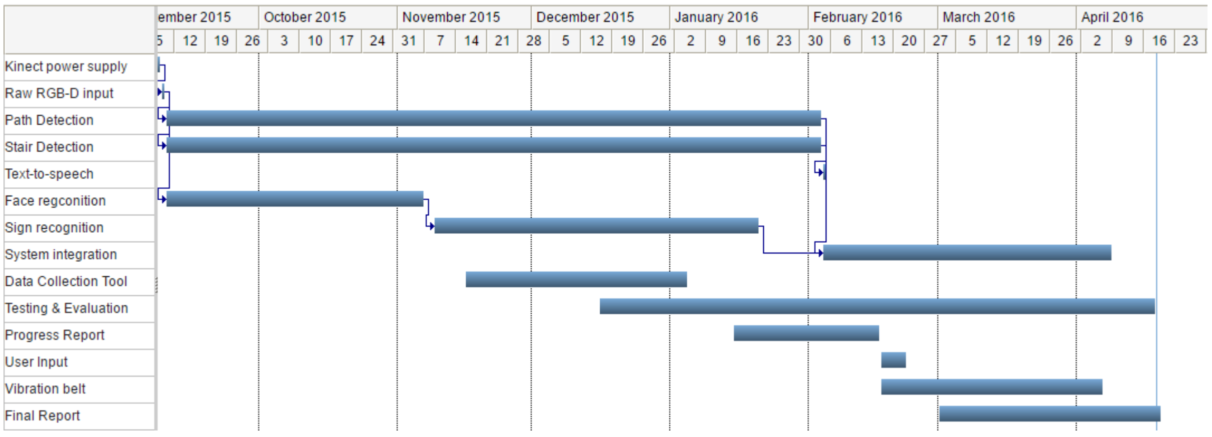


Figure 96 Gantt Chart

Appendix C: Required Hardware and Software

Hardware:

- Microsoft Kinect Sensor
    - It is a device built with RGB camera, depth camera, multi-array microphone, accelerometer and the elevation of the camera is adjustable.
- DC battery and adapter
    - 3x 3.7V cells Lithium Polymer battery provides power for Kinect operation.
- Arduino Leonardo microcontroller
- Vibration motor x 3 and on-off switch board
- Computer: PC / Notebook
- CPU: Dual-core, 2.66GHz or above
- Memory: 2GB RAM
- Graphic Card: support DirectX 9.0c

Software:

- Operating System: Microsoft Windows
- Visual C++ Redistributable Packages for Visual Studio 2013
- OpenCV 2.4.12 in C/C++
- OpenNI SDK 2.2
    - It allows an application to initialize Kinect sensor and receive depth, RGB, and IR video streams from the Kinect. It provides an interactive interface to sensors and .ONI recordings created with Kinect.
- Windows Kinect SDK 1.7
    - It is released by Microsoft for developing Kinect applications.
- Tesseract OCR 3.0
- Microsoft Speech SDK 5.1

Appendix D: Needs Analysis

Other than studying articles, interviews and news related to visually impairment, a simulation of being visually impaired was done to have better understanding of them. The simulation procedure and how it was carried out is as follows:

1. Make special glasses, which has partially blocked and blurred sight, to simulate the low vision visually impaired people
2. Invite a groupmate to act as a travelling aid which notifies the tester whenever there is obstacles or danger in front
3. Put a phone with camera in a pocket near the tester breast
4. Pick two routes with indoor and outdoor environment
5. Record the sight in front of the tester
6. Review the videos to understand the situation and what was overlooked during the simulation

Our summery is that tester can barely aware something in front of him but not able to identify the object. Tester lost their distance perception after wearing the distinctive glass. Therefore, tester often feels stressful in making decision to avoid obstacles since he cannot judge when he will collide on them.

Appendix E: On-Off switch Circuit Board

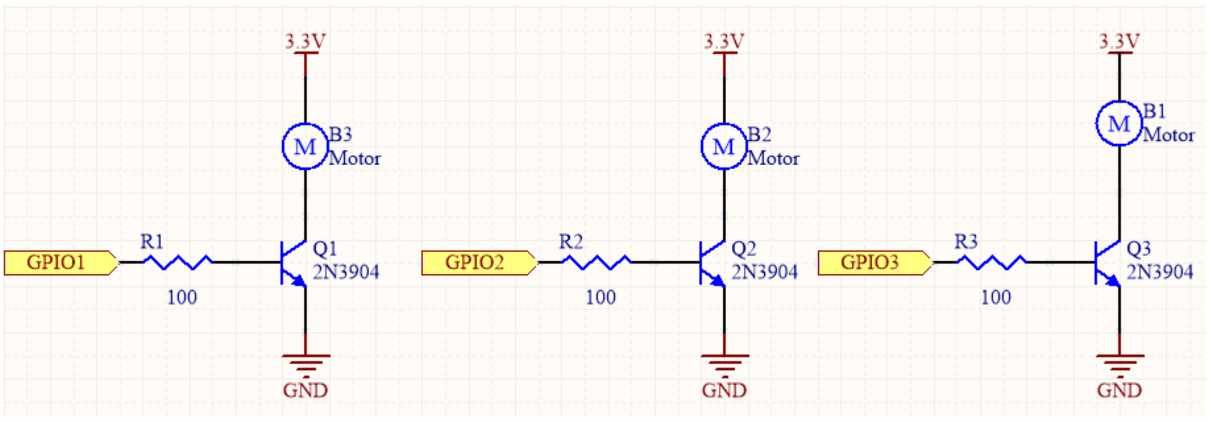
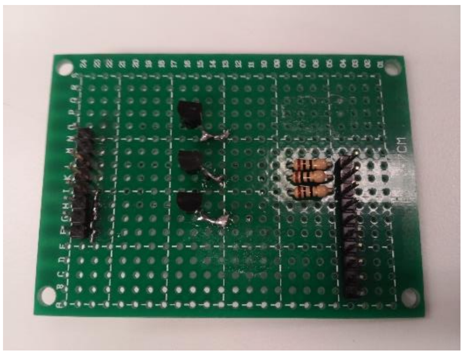
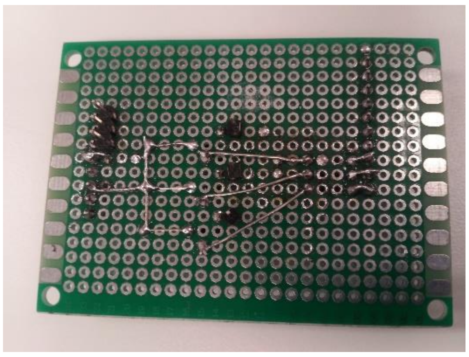


Figure 97 Schematic of On-Off switch



Top Layer



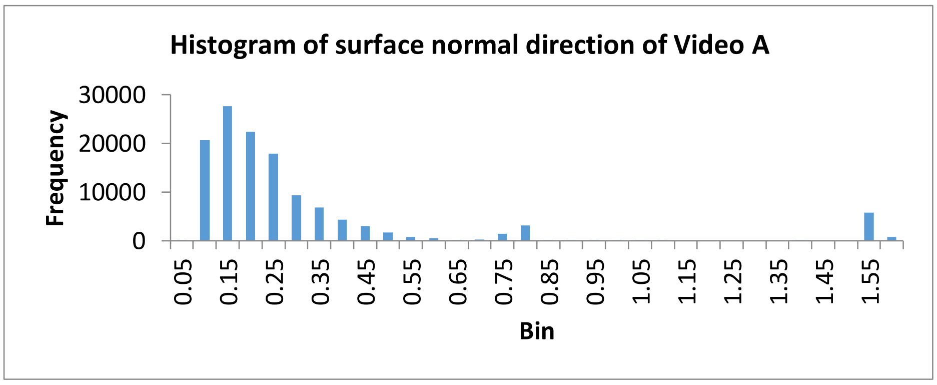
Bottom Layer

Figure 98 On-Off switch Circuit Board

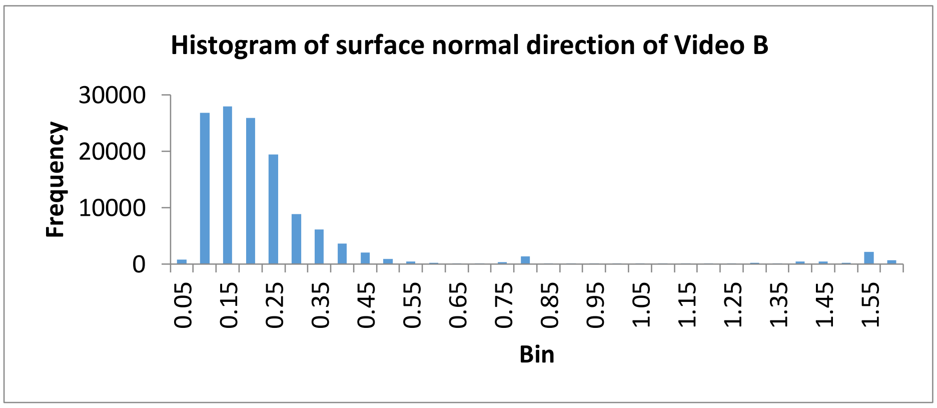
Appendix F: Statistics about ground pixel

The video contain ground only image with resolution 320x240 and depth only image. Statistics are calculated by using Microsoft Excel 2013. "Total" is the number of pixel collected in the whole video.

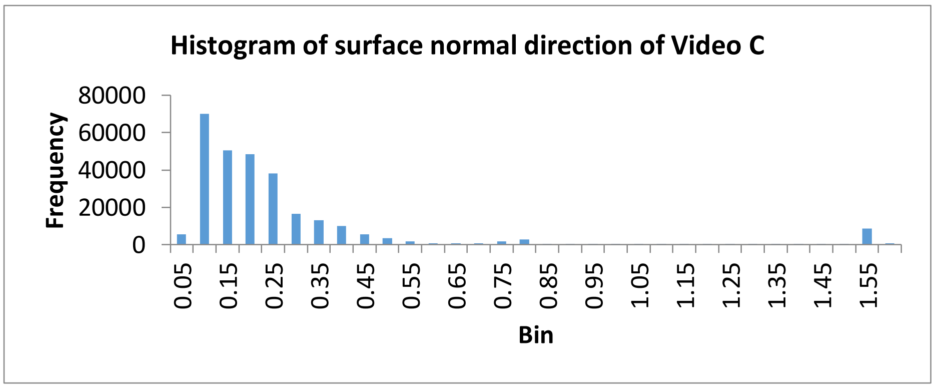
I. Surface normal direction



Total	127464
min	0.029276
max	1.56005
SD	0.328763
mean	0.285282

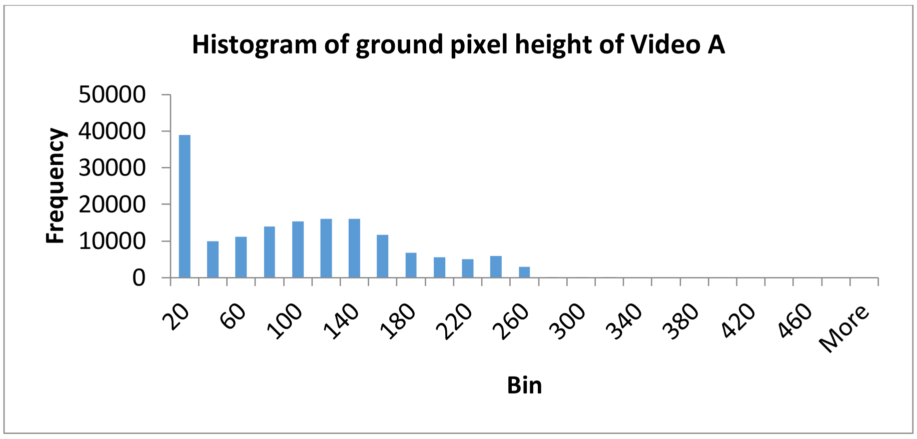


total	130191
min	0.013024
max	1.56628
SD	0.267235
mean	0.233628

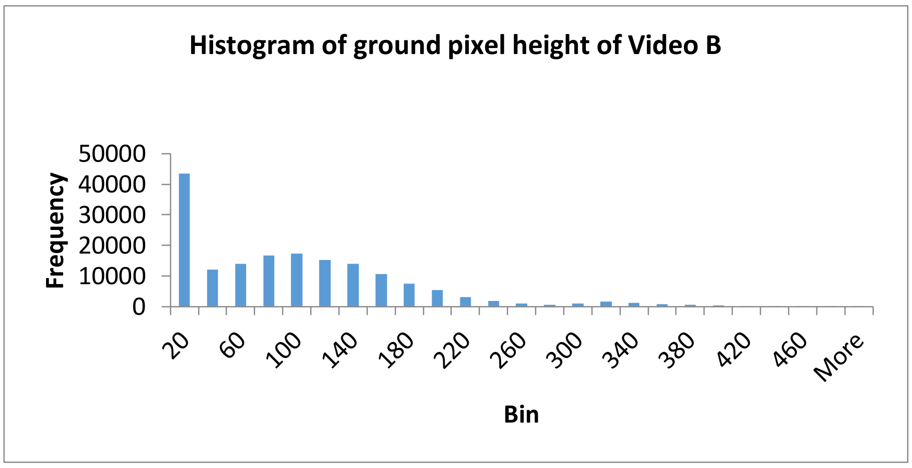


total	279125
min	0.011916
max	1.56082
SD	0.27848
mean	0.234966

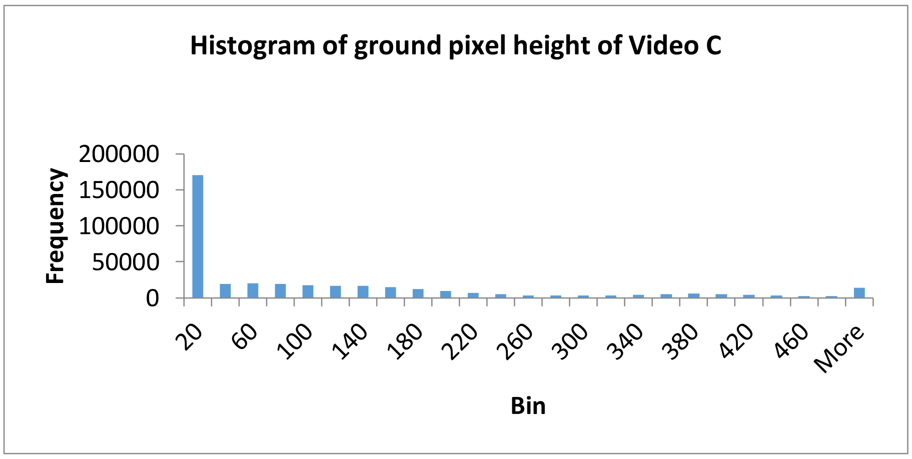
II. Ground pixel height



total	159674
min	-445
max	305
SD	90.12319
mean	79.05785



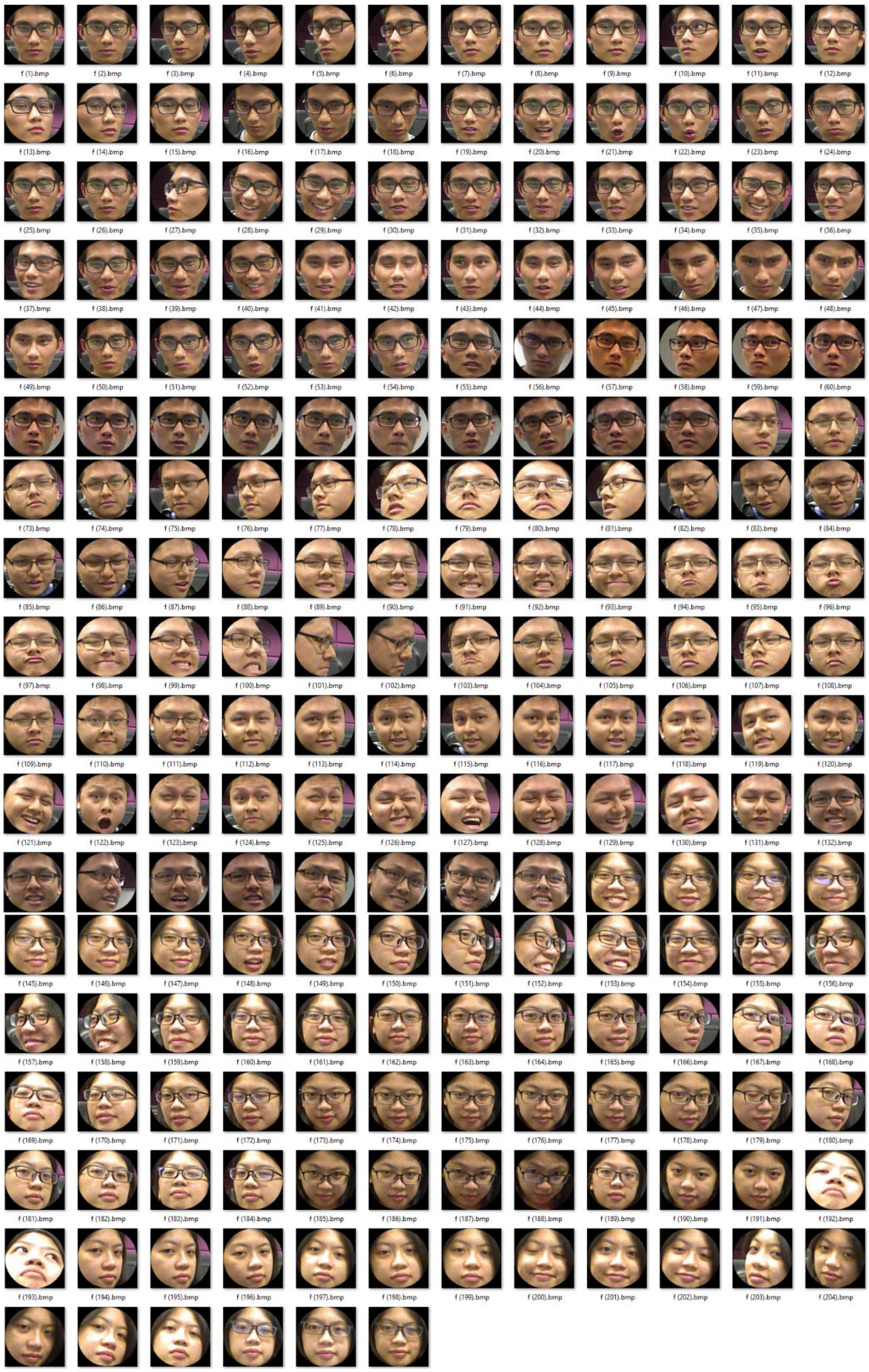
total	169274
min	-750
max	484
SD	104.0687
mean	74.63628



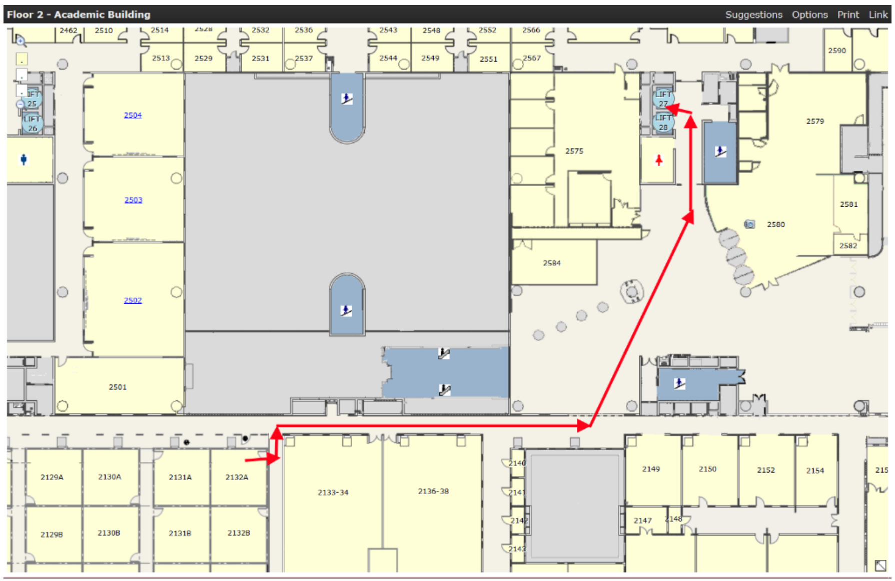
total	389992
min	-1372
max	1252
SD	218.7812
mean	54.16491

Appendix G: Training Set of Face Recognizer

Only our teammate faces before equalization shown. (Guest images are not shown here)



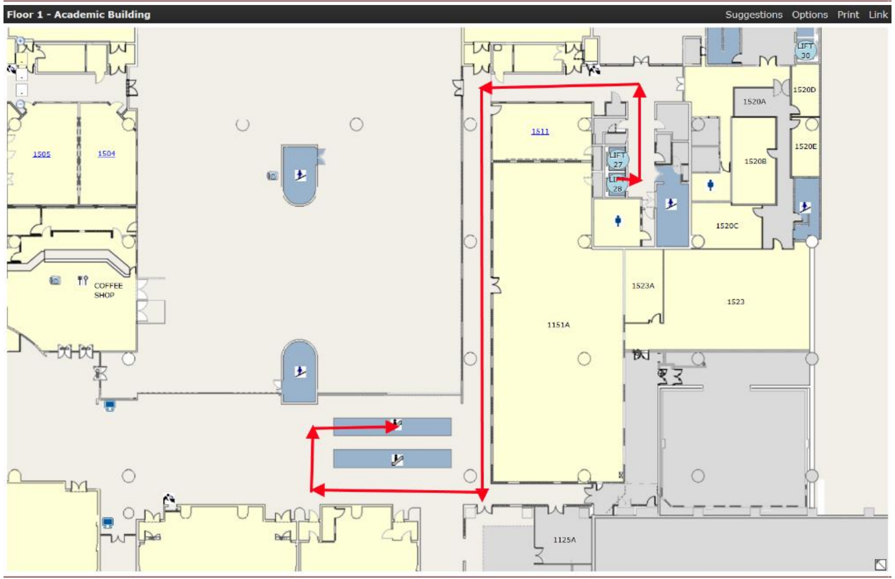
Appendix H: Video 1 Sample path in HKUST for testing.



From Floor 2,

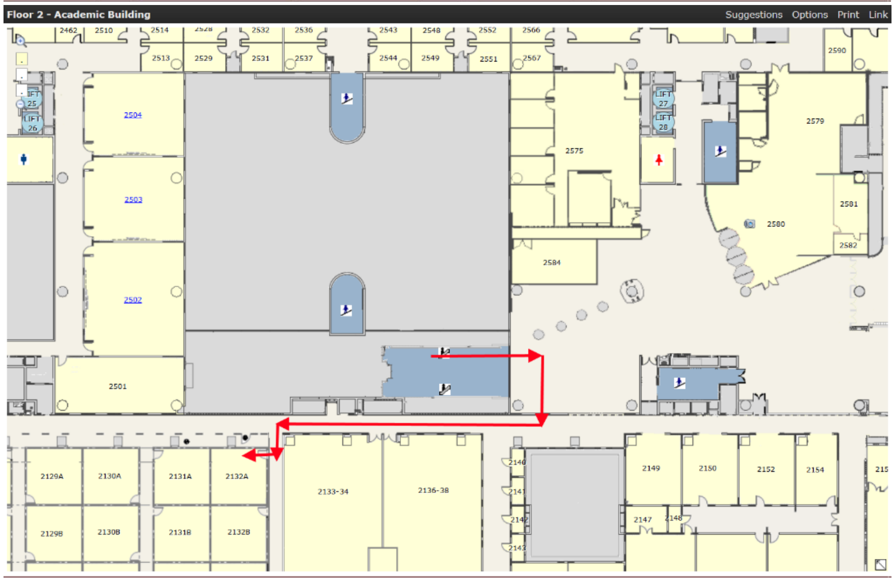
Room 2132A

To Floor 2, Lift 27/28 down to Floor 1



From Floor 1, Lift 27/28

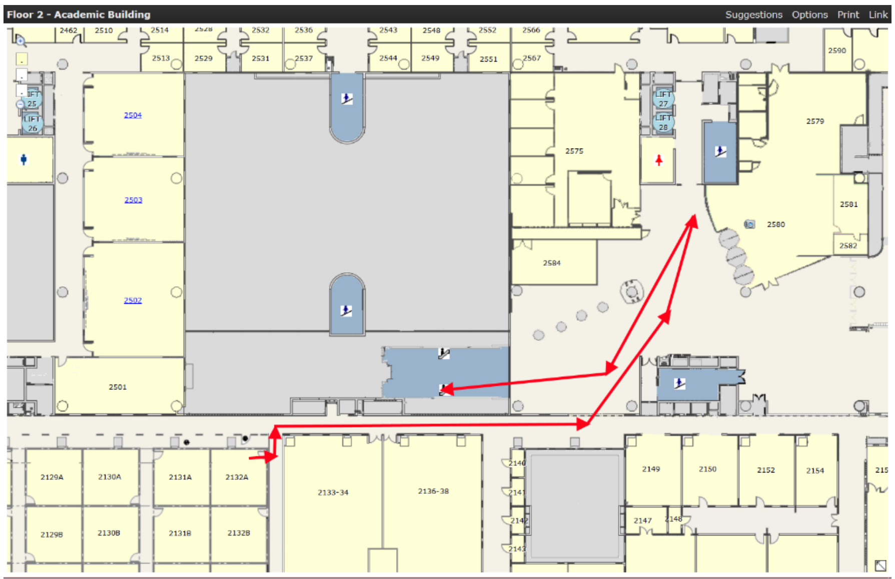
To Floor 1, Stairs outside LT-D going up



From Floor 2, Stairs outside Engineering Commons

To Floor 2, Room 2132A

Appendix I: Video 2 Sample path in HKUST for testing.

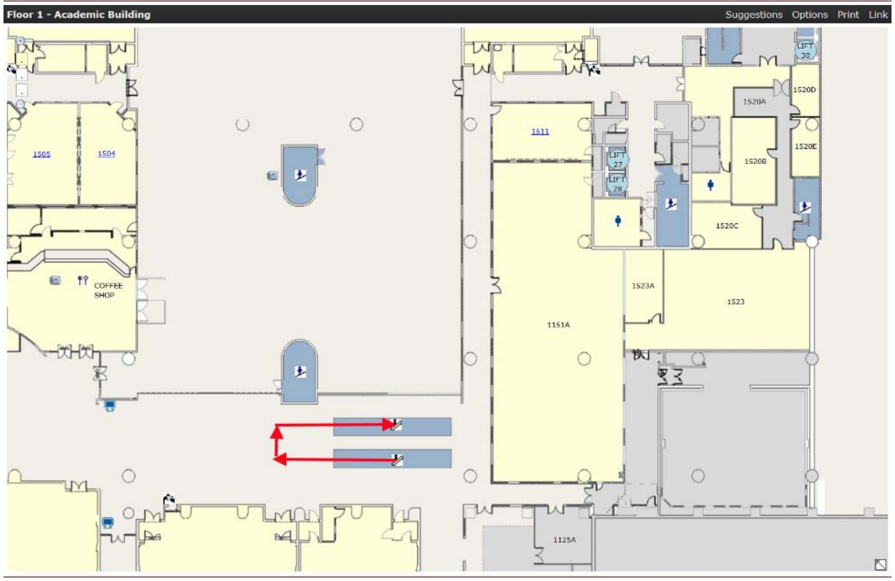


From Floor 2, Room 2132A

Pass by Floor 2, Lift 27/28

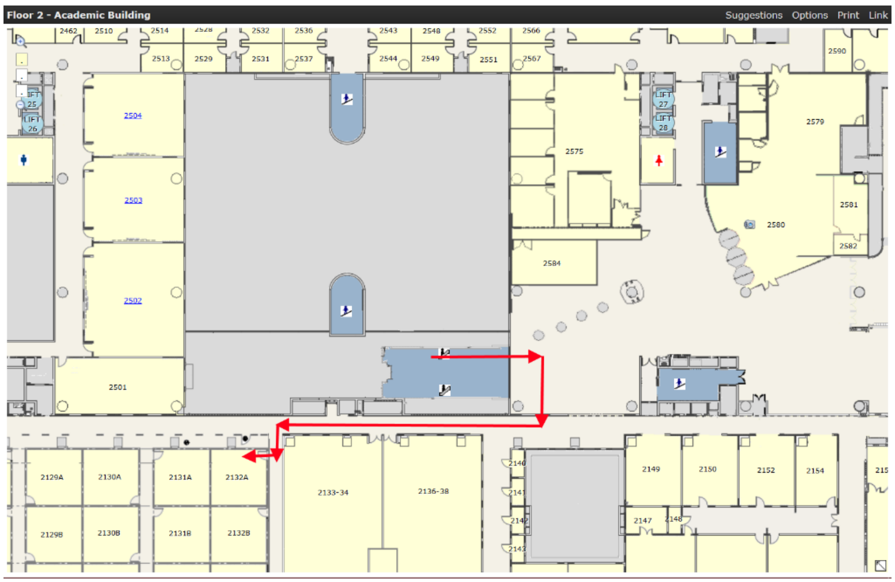
To Floor 2,

Stairs outside Engineering Commons going down



From Floor 1, Stairs outside LT-D

To Floor 1, Stairs going up to Engineering Commons

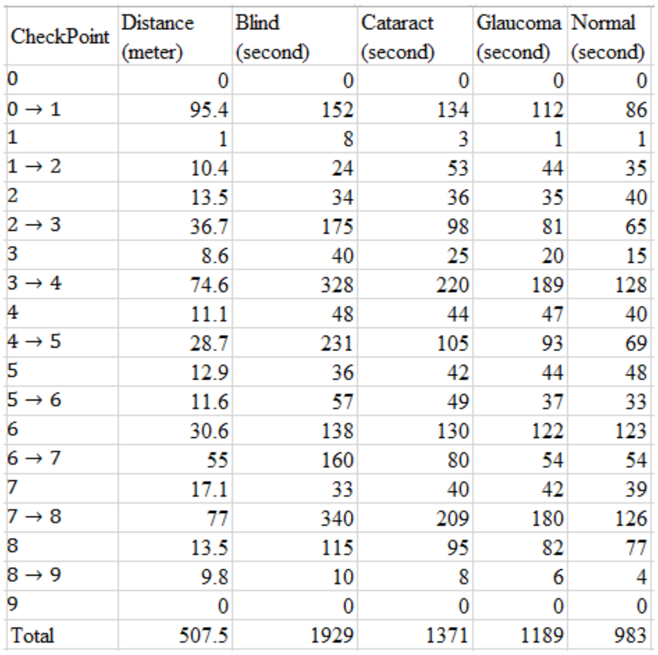


From Floor 2

Stairs outside Engineering Commons

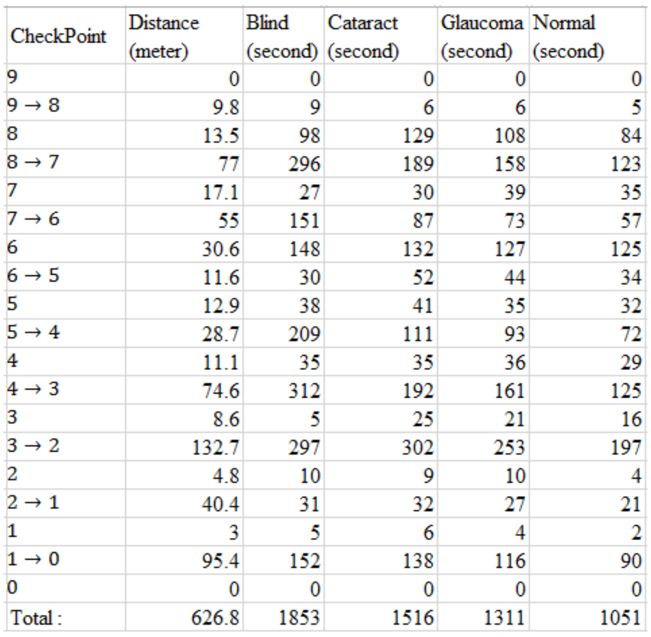
To Floor 2, Room 2132A

Appendix J: Statistic for Route 1 and Route 2.



CheckPoint	Distance (meter)	Blind (second)	Cataract (second)	Glaucoma (second)	Normal (second)
0	0	0	0	0	0
0 → 1	95.4	152	134	112	86
1	1	8	3	1	1
1 → 2	10.4	24	53	44	35
2	13.5	34	36	35	40
2 → 3	36.7	175	98	81	65
3	8.6	40	25	20	15
3 → 4	74.6	328	220	189	128
4	11.1	48	44	47	40
4 → 5	28.7	231	105	93	69
5	12.9	36	42	44	48
5 → 6	11.6	57	49	37	33
6	30.6	138	130	122	123
6 → 7	55	160	80	54	54
7	17.1	33	40	42	39
7 → 8	77	340	209	180	126
8	13.5	115	95	82	77
8 → 9	9.8	10	8	6	4
9	0	0	0	0	0
Total	507.5	1929	1371	1189	983

Table 26. Time and distance record for testing route 1



CheckPoint	Distance (meter)	Blind (second)	Cataract (second)	Glaucoma (second)	Normal (second)
9	0	0	0	0	0
9 → 8	9.8	9	6	6	5
8	13.5	98	129	108	84
8 → 7	77	296	189	158	123
7	17.1	27	30	39	35
7 → 6	55	151	87	73	57
6	30.6	148	132	127	125
6 → 5	11.6	30	52	44	34
5	12.9	38	41	35	32
5 → 4	28.7	209	111	93	72
4	11.1	35	35	36	29
4 → 3	74.6	312	192	161	125
3	8.6	5	25	21	16
3 → 2	132.7	297	302	253	197
2	4.8	10	9	10	4
2 → 1	40.4	31	32	27	21
1	3	5	6	4	2
1 → 0	95.4	152	138	116	90
0	0	0	0	0	0
Total :	626.8	1853	1516	1311	1051

Table 27. Time and distance record for testing route 2

Appendix K: Data collection Tool for test during coding

The images are captured and loaded using the imwrite and imread functions provided by OpenCV 2.4 library. Color frames are captured at 1280x720 pixel resolution, and the depth frames are captured at 320x240 pixel resolution.

The videos are recorded and played back by the OpenNI Recorder class provided by OpenNI 2.2 library. The color stream is configured to record at 1280x720 pixel resolution at 12 fps, while the depth stream is configured to record at 320x240 pixel resolution at 30fps. Meanwhile, the view angle of Kinect of each frame is save into a file using C++ iostream library.