

Design & Innovation Project (DIP)

Project Report

<<Multi-view Recognition System>>

Project Group: <<E022>>

School of Electrical and Electronic Engineering
Academic Year 2019/20
Semester 1

Table of Contents

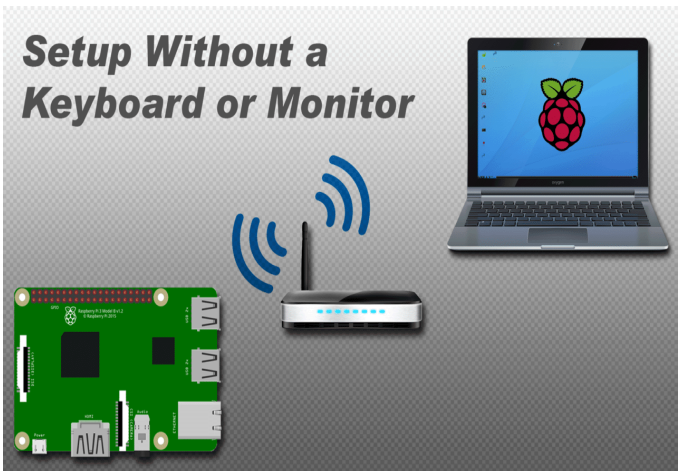
1. Purpose / Project Objectives
2. Project Summary
3. Scope
4. Schedule
5. Cost / Benefit
6. Outcome / Expected Outcomes
7. Reflection
8. References

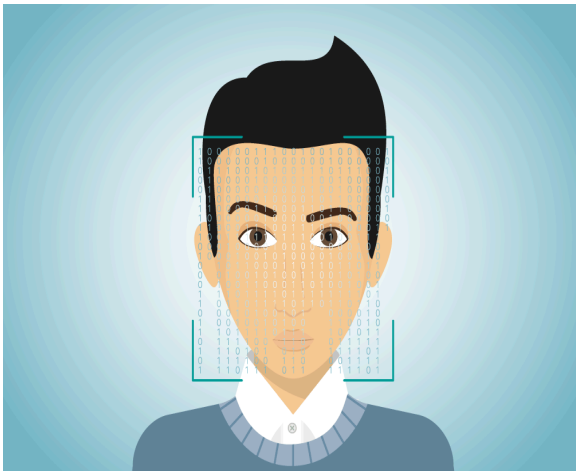
Appendix

1. Purpose/ Project Objectives(He weiyang)

Project Objectives

The general goal of this project is to set up an intelligent system for real-time, multi-camera recognition using Raspberry Pi and personal computer. The *Computer Vision* module of Machine Learning will be researched to provide the necessary knowledge for performing facial recognition. Specifically, existing *Convolutional Neural Network* (CNN) will be utilised to as the model after modification and optimization. All the group members will be able to experience and get inspired from the application of neural networks in real life scenario.



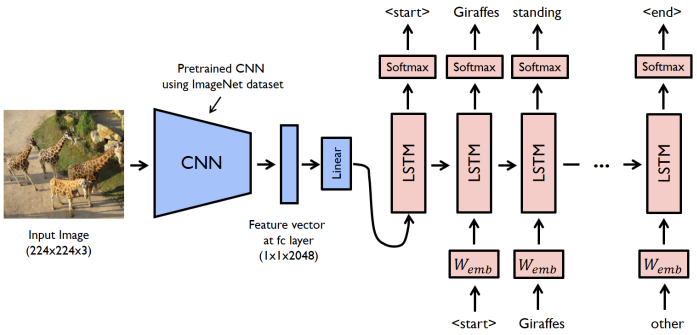


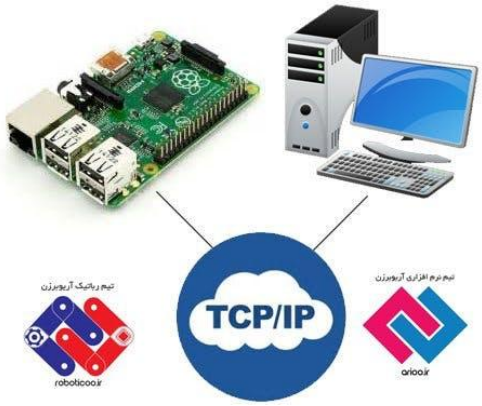
Project Overview

The expected components of the system for final use should be one Raspberry Pi, one Raspberry Pi camera connected through cable to the Pi, suitable ethernet/wireless connection from Pi to the personal computer, and one high performance personal computer. When the system is executing, Raspberry Pi will continuously capture video frames from the connected camera. At the same time, Raspberry Pi will analyse each received frames to calculate the face information in the format of vector. Then, the captured video frames and face information vector will be sent to the personal computer. The personal computer is responsible for providing user interface and matching received

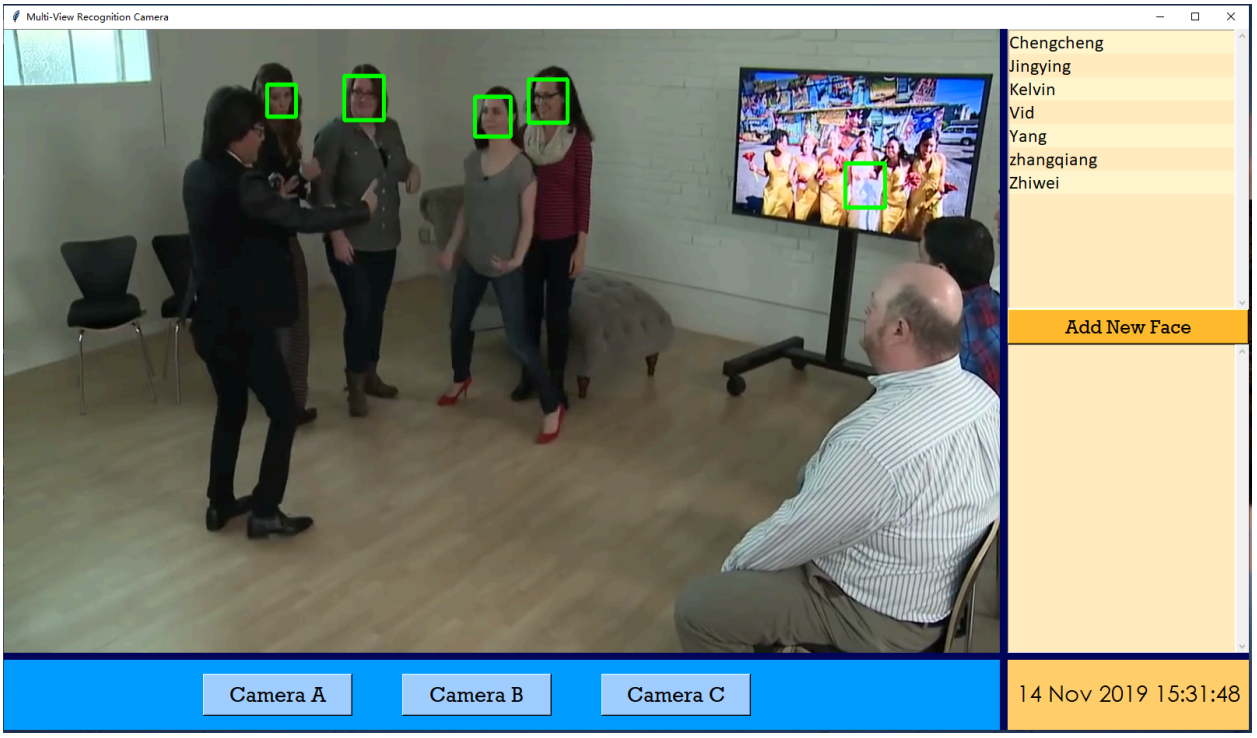
face data with stored face data. User is able to watch the real-time frames with people's faces encircled by a rectangle, read the matching and identification results from a notification panel and extract the past video frames for display according to face name or time range.











To achieve the goal, seven members of our group are divided into three subgroups. The number of members and respective task for each subgroup is listed below:

Group 1 (**3 persons**): Test and select boards that works best for the purpose of the project. (Raspberry Pi 4B, NVIDIA Jetson Nano, Google Coral Edge TPU Board etc.)

Group 2 (**2 persons**): Code in Python to implement desired GUI for the product

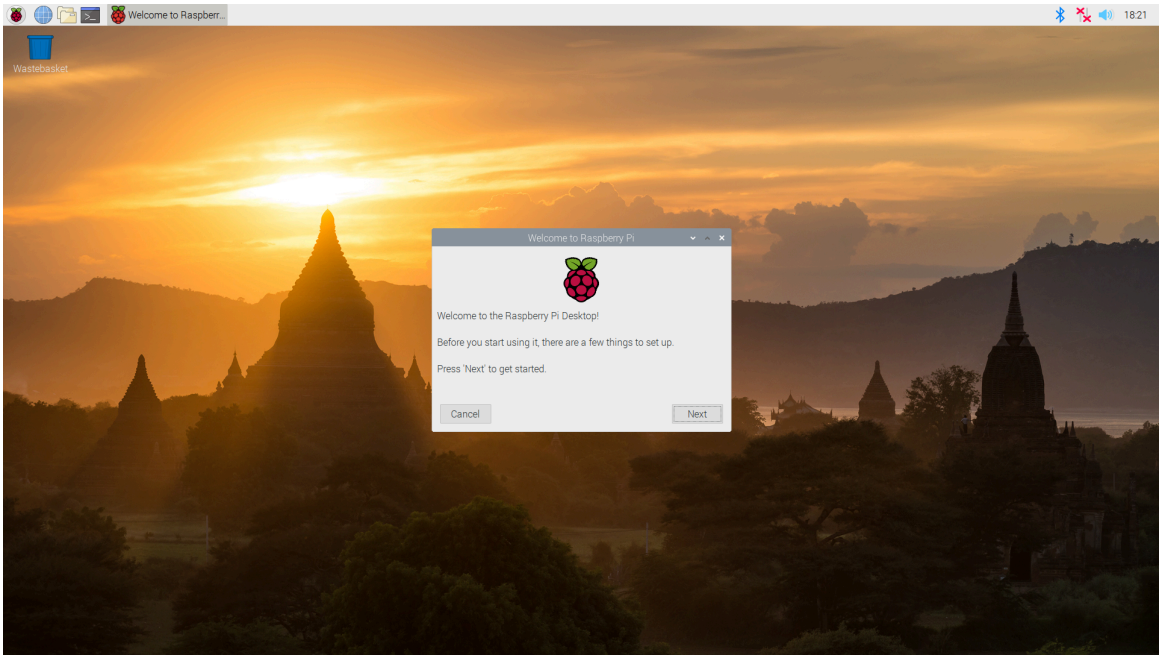
Group 3 (**2 persons**): Build up a backend server/database using Pytorch (as well as Tensorflow)

Group1 will firstly study the operation of Raspberry Pi and facial recognition machine learning code in Python. Then, the group1 members will test these codes on Raspberry Pi to make sure all the functions can be realized. Group2 will use the **Tkinter** module from Python to build a GUI. The members of Group2 are required to learn the Tkinter functions as well as communication methods between personal computer and Raspberry Pi such as TCP/UDP network. Group3 will build the database for the received video frames through GUI, and provide an API for GUI to extract video frames. All three subgroups are to work in a parallel manner, and combine the final work together to perform the operation.

Specific Requirement

1）For Raspberry Pi

- A Raspberry Pi 4 is needed to set up the system.
- Necessary peripherals are Raspberry Pi camera, intel neural compute stick, microSD card, mouse, keyboard and HDMI cable.
- Raspberry Pi is powered up after connecting the power cable.
- Raspbian operating system is used to perform configuration.



- Raspbian OS needs to be updated to get access to the camera.
- Python3 and relevant packages for running the camera and facial recognition function should be installed.



apt

Some Python packages can be found in the Raspbian archives and can be installed using apt. For example:

```
sudo apt update
sudo apt install python3-picamera
```

This is the preferred method of installing software, as it means that the modules you install can be kept up to date easily with the usual `sudo apt update` and `sudo apt full-upgrade` commands.

Python packages in Raspbian which are compatible with Python 2.x will always have a `python-` prefix. So, the `picamera` package for Python 2.x is named `python-picamera` (as shown in the example above). Python 3 packages always have a `python3-` prefix. So, to install `picamera` for Python 3 you would use:

```
sudo apt install python3-picamera
```

Uninstalling packages installed via APT can be accomplished as follows:

```
sudo apt remove python3-picamera
```

They can be completely removed with `purge`:

```
sudo apt purge python3-picamera
```

pip

Not all Python packages are available in the Raspbian archives, and those that are can sometimes be out-of-date. If you can't find a suitable version in the Raspbian archives, you can install packages from the Python Package Index (PyPI). To do so, use the `pip` tool.

`pip` is installed by default in Raspbian Desktop images (but not Raspbian Lite). You can install it with `apt`:

```
sudo apt install python3-pip
```

To get the Python 2 version:

```
sudo apt install python-pip
```

`pip3` installs modules for Python 3, and `pip` installs modules for Python 2.

For example, the following command installs the Unicorn HAT library for Python 3:

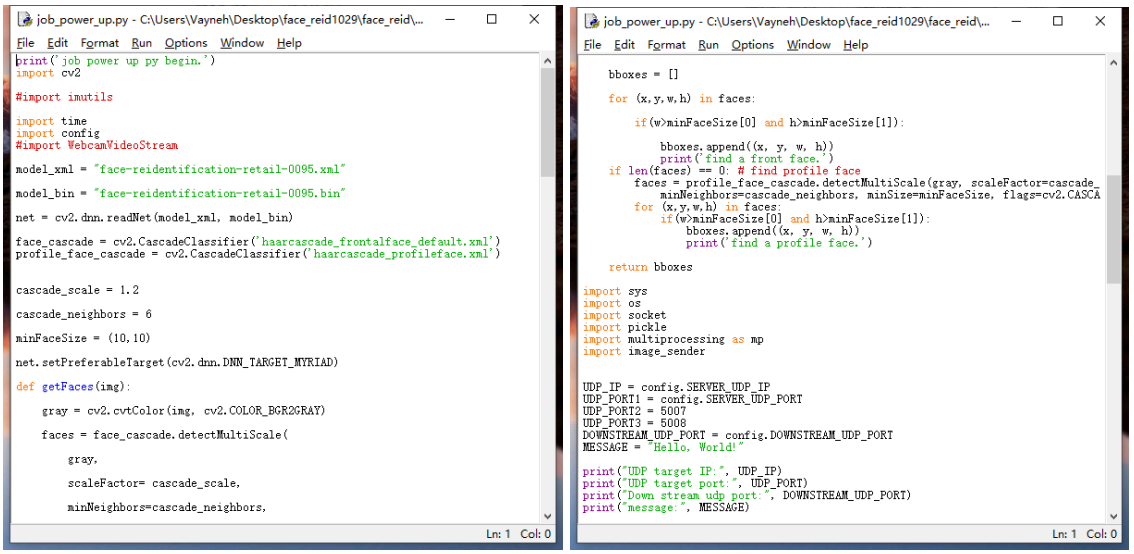
```
sudo pip3 install unicornhat
```

The following command installs the Unicorn HAT library for Python 2:

```
sudo pip install unicornhat
```

Uninstall Python modules with `sudo pip3 uninstall` or `sudo pip uninstall`.

- The python program running on the Raspberry Pi is required to open the camera connected to it and send these video frames to the PC for real-time display using Socket module. Meanwhile, the Raspberry Pi use functions from OpenCV module to do face feature extraction. Face feature is represented by multiple sets of numbers produced by facial recognition algorithm. Face feature also includes the relatively accurate position of the face which is to be sent to the PC for UI displaying purpose.



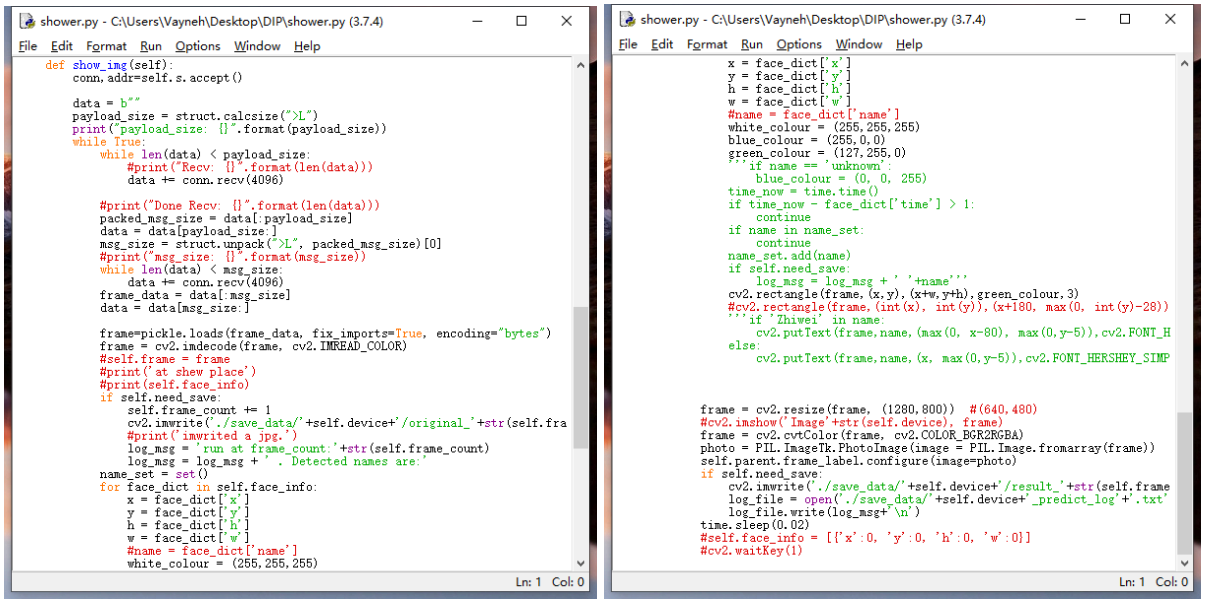
- The Raspberry Pi connects with PC under the local area network, which means they have to connect to the same wireless network or using ethernet direct connection. Moreover, the agreed TCP-PORT and UDP-PORT should be cleared before use.
- Python program “job_power_up.py” on the Raspberry Pi needs to be executed before running the UI. Raspberry Pi can be controlled by PC using VNC viewer for convenience.

2) For Personal Computer

- Python3 and required packages such as opencv, pillow, tkinter, socket and numpy should be installed.
- PC displays real-time video frames with encircling rectangles around identified faces while performing matching and fusion to the received face features from the Raspberry Pi. This means the program will compare newly identified face data with the stored face data to determine the identity of the person. The process result will be shown at the notification panel of the UI.
- After displaying on the UI, the received video frames will be tagged with identified face name (if it has match result) and recording time. Then, they will be saved under a folder for

extraction.

- User needs to click “add face” button to store newly found face data for future matching. For extraction of past video, time range or face name can be chosen in order to find the desired sets of frames.



```
shower.py - C:\Users\Vayneh\Desktop\DIP\shower.py (3.7.4)
File Edit Format Run Options Window Help
    def show_img(self):
        conn, addr=self.s.accept()

        data = b""
        payload_size = struct.calcsize(">L")
        print("payload_size: {}".format(payload_size))
        while True:
            while len(data) < payload_size:
                #print("Recv: {}".format(len(data)))
                data += conn.recv(4096)

            #print("Done Recv: {}".format(len(data)))
            packed_msg_size = data[:payload_size]
            data = data[payload_size:]
            msg_size = struct.unpack(">L", packed_msg_size)[0]
            #print("msg_size: {}".format(msg_size))
            while len(data) < msg_size:
                data += conn.recv(4096)
            frame_data = data[:msg_size]
            data = data[msg_size:]

            frame=pickle.loads(frame_data, fix_imports=True, encoding="bytes")
            frame = cv2.imdecode(frame, cv2.IMREAD_COLOR)
            #self.frame = frame
            #print('at shew place')
            #print(self.face_info)
            if self.need_save:
                self.frame_count += 1
                cv2.imwrite('./save_data/'+self.device+'/original_'+str(self.fra
                #print('imwrited a jpg.')
                log_msg = 'run at frame_count:'+str(self.frame_count)
                log_msg = log_msg + ' . Detected names are:'
            name_set = set()
            for face_dict in self.face_info:
                x = face_dict['x']
                y = face_dict['y']
                h = face_dict['h']
                w = face_dict['w']
                #name = face_dict['name']
                white_colour = (255, 255, 255)
Ln: 1  Col: 0
```

```
shower.py - C:\Users\Vayneh\Desktop\DIP\shower.py (3.7.4)
File Edit Format Run Options Window Help
                x = face_dict['x']
                y = face_dict['y']
                h = face_dict['h']
                w = face_dict['w']
                #name = face_dict['name']
                white_colour = (255, 255, 255)
                blue_colour = (255, 0, 0)
                green_colour = (127, 255, 0)
                '''if name == 'unknown':
                    blue_colour = (0, 0, 255)
                time_now = time.time()
                if time_now - face_dict['time'] > 1:
                    continue
                if name in name_set:
                    continue
                name_set.add(name)
                if self.need_save:
                    log_msg = log_msg + ' '+name'''
                cv2.rectangle(frame, (x, y), (x+w, y+h), green_colour, 3)
                #cv2.rectangle(frame, (int(x), int(y)), (x+180, max(0, int(y)-28))
                '''if 'Zhiwei' in name:
                    cv2.putText(frame, name, (max(0, x-80), max(0, y-5)), cv2.FONT_H
                else:
                    cv2.putText(frame, name, (x, max(0, y-5)), cv2.FONT_HERSHEY_SIMP



            frame = cv2.resize(frame, (1280, 800))  #(640, 480)
            #cv2.imshow('Image'+str(self.device), frame)
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
            photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))
            self.parent.frame_label.configure(image=photo)
            if self.need_save:
                cv2.imwrite('./save_data/'+self.device+'/result_'+str(self.frame
                log_file = open('./save_data/'+self.device+'_predict_log'+'.txt'
                log_file.write(log_msg+'\n')
            time.sleep(0.02)
            #self.face_info = [{'x': 0, 'y': 0, 'h': 0, 'w': 0}]
            #cv2.waitKey(1)
Ln: 1  Col: 0
```

2. Project Summary

As showcased above, this project is led by a team of 7, which can be further divided into three sub-groups, namely, group1, group2 and group3.

The general working relationship between the three groups can be well illustrated in the Workflow Chart below.

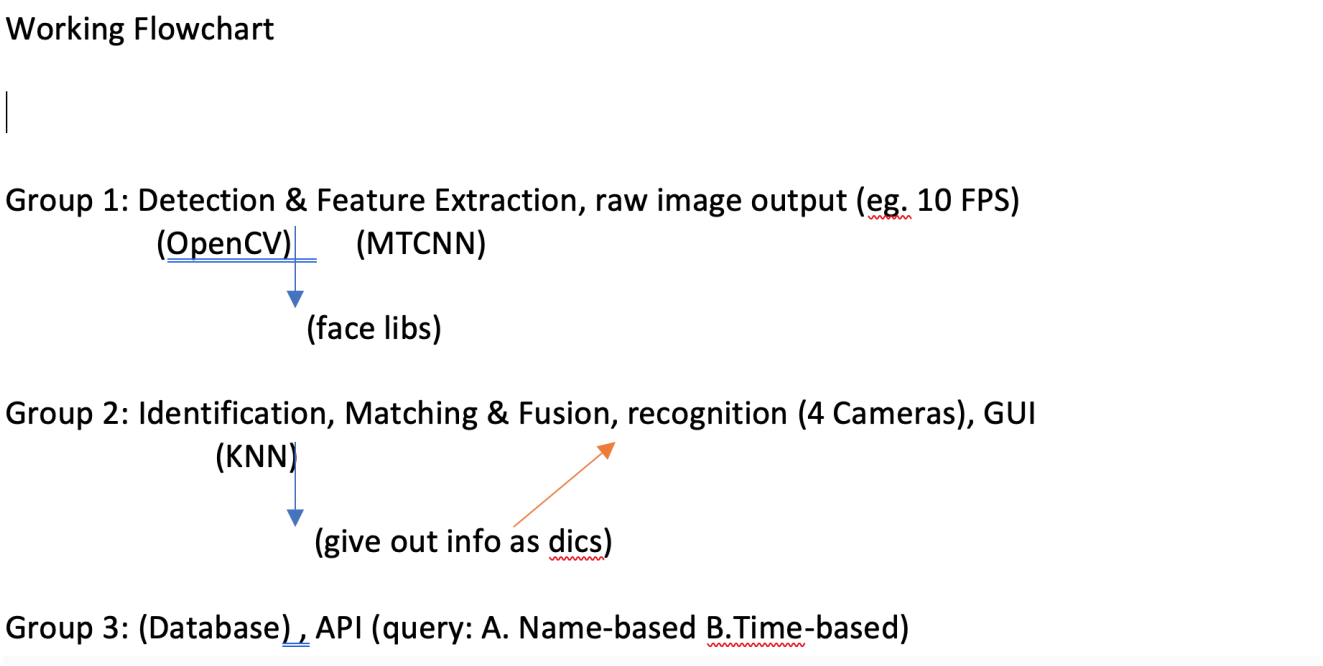


Figure. 1.a

The main objective of each group and its completion status can be described as follows:

Group 1 (**3 persons**):

- Objective: 1. Test and select boards that works best for the purpose of the project.
  2. Algorithm for Face Detection as well as Face Recognition
  (along with Feature Extraction)

  Boards Tested:

  (Raspberry Pi 4B, NVIDIA Jetson Nano, Google Coral Edge TPU Board etc.)

- Completion status: ***Satisfactory***

*Disclaimer: Most of the algorithms(Coding Part) are developed in this project is finished with the assistance from the FACIAL RECOGNITION phd Team in Delta-NTU lab.

The related libraries (major ones only) used in this project includes:

1. Face Detection: OpenCV, OpenCL
2. Feature Extraction: MTCNN
3. Identification, Matching & Fusion and Face Recognition: KNN

Sample Codes *(Fusion)*:

```
import socket
import pickle
import load_face_vector_train_knn
import numpy as np
from sklearn import preprocessing
import time
import cv2
import process_fusion

neigh, name2id, id2name, neigh_distance = load_face_vector_train_knn.load_knn()

UDP_IP = '0.0.0.0' # "127.0.0.1"
UDP_PORT = 5005

downstream_sock_port = 5006
sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

downstream_sock = socket.socket(socket.AF_INET, # Internet
                     socket.SOCK_DGRAM) # UDP

fusion_processor = process_fusion.FusionProcessor()

count = 0
def fuse(self):
    while True:
        data, addr = sock.recvfrom(1024*200) # buffer size is 1024 bytes . if oversized, increase buffe
        msg_dict = pickle.loads(data)

        # for key in msg_dict:
        #       print(key, msg_dict[key])
        face_vector= msg_dict['face_vector']
        face_vector2 = []
        for tmp in face_vector[0]:
            face_vector2.append(tmp[0][0])
        face_vector = face_vector2
        #print('Received a face vector')
        count += 1
        #print('count:', count)
        face_vector = preprocessing.normalize([face_vector], norm='l2')
        #face_vector = face_vector[0]
        preded = neigh.predict(face_vector)
        preded_proba = neigh.predict_proba(face_vector)
        distants, idxs = neigh_distance.kneighbors(face_vector)
        #print('Rearest person: ', id2name[str(preded[0])],  ' Probability: ', preded_proba[0])
        print('Find a face, nearest distant: ', distants[0])
        self.notification.insert('insert','Find a face, nearest distant: '+distants[0]+'\n')
        person_name = 'unknown'
        if distants[0] < 0.7:
            print('Recogize a person: ',id2name[str(preded[0])])
            person_name = id2name[str(preded[0])]
            self.notification.insert('insert','Recogize a person: '+id2name[str(preded[0])]+'\n')
        else:
            print('saved an unknown person')
            self.notification.insert('insert','Saved an unknown person\n')
            # ticks = time.time()
            # face_img = msg_dict['face_img']
            # cv2.imwrite("./unknow_face_data/unknown_face_img_"+str(ticks)+'.jpg', face_img)
            # save_dict = {'person_name': 'unknown', 'ticks': str(ticks), 'face_vector':face_vector}
            # f = open("./unknow_face_data/face_vector_of_unknown_"+str(ticks), "wb")
            # pickle.dump(save_dict, f)
```

Out of the 4 testing boards suggested, 3 are tested.

Among them are Raspberry Pi 4B (single), Raspberry Pi 4B (with Intel® Computing Stick 2) and finally NVIDIA Jetson Nano.

* Google Coral Edge TPU Board is not available for testing due to transportation issues.

After comparing the stats, NVIDIA Jetson Nano is initially selected as the go-to choice for our project. However, since its existing CV2 library turns out to be not compatible with our coding, this motion is unfortunately aborted. Thus, the second best performing sample, Raspberry Pi 4B(with Intel® Computing Stick 2) is chosen eventually.

For more details and testing data, please refer to # Appendix B

Group 2 (**2 persons**):

- Objective: Code in Python to implement desired GUI for the product
- Completion status: *Mostly Satisfactory*

Testing results indicates that most of the desired functionalities can be properly achieved by the GUI designed by group two.

Examples:

1. CAMERA SWITCHING

2. NAME-BASED EXTRACTION

3. TIME-BASED EXTRACTION

However, the Graphical User Interface itself is not user-friendly enough for complete new customers to use, considering the fact that prompts and reminders are not provided sufficiently to guide through the users in the application.

Group 3 (**2 persons**):

- Objective: Build up a backend server/database using Pytorch (as well as Tensorflow)
- Completion Status: *Mildly Satisfactory*

Based on the three main functionalities provided by group 1 and 2, group three is able to come up with an appropriate API (Application Programming Interface) that access the features or data of the operating system as planned.

However, during the real demo session, several bugs incurred malfunctioning of the entire application, which indicates potential risks and possibilities of failing in real life functioning.

We intend to reduce error rate and fix these problems as the project continues to be developed in the coming future.

The overall project of EE022 (**Multi-view Recognition System**) successfully build a system that achieves the following functionalities:

a. Face Detection

b. Face Recognition (Including Identification, Matching & Fusion done by three individual cameras placed at three different respective angles)

c. Database(*Name,Time and Camera exposure duration*) set up

To sum up, all of the above gives us the desirable application we want, which is ***a high-functioning real-time facial recognition system*** **that provides user enquiry capability** (person-based or time-based) **with a relatively low and practical cost.**

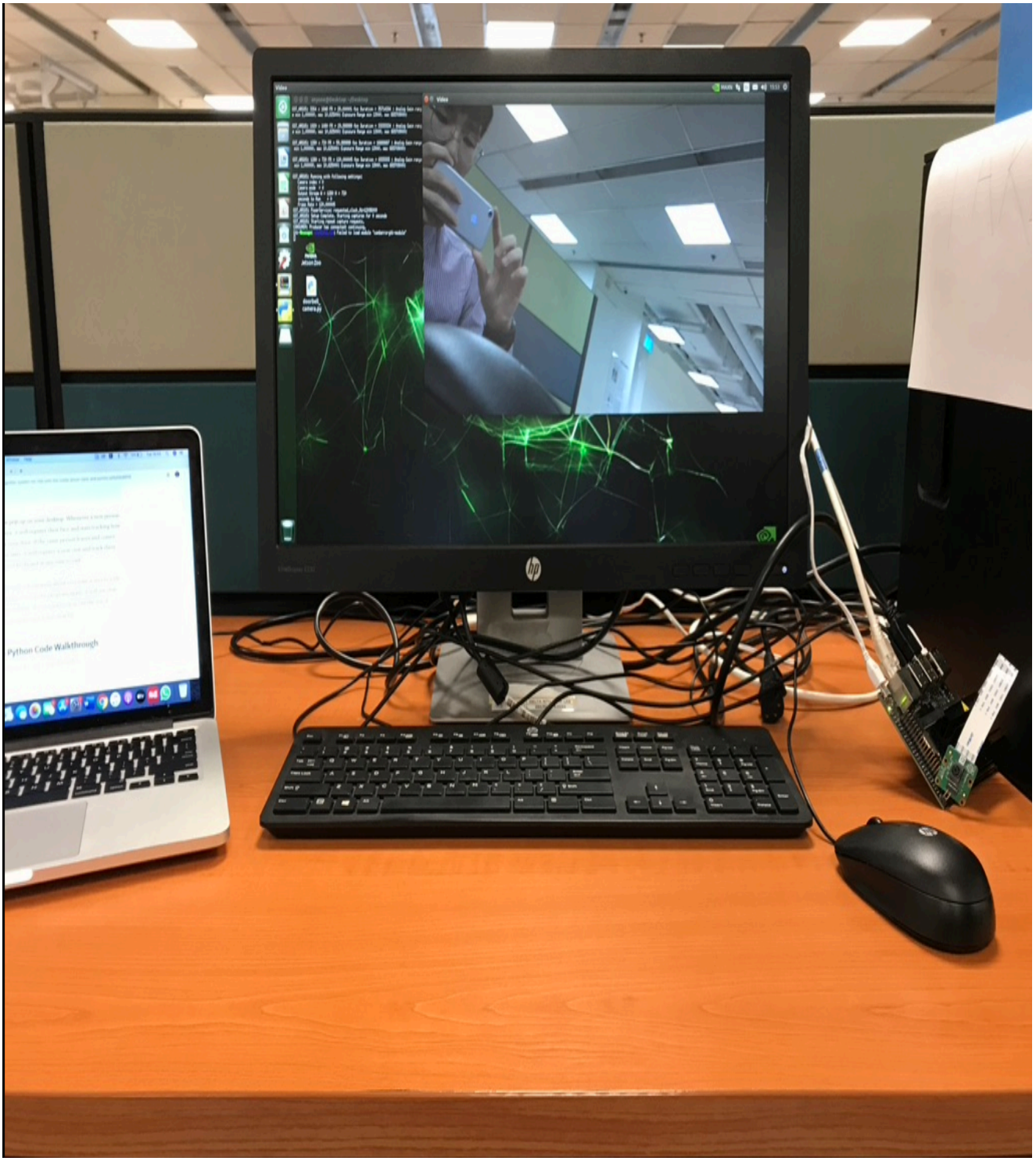
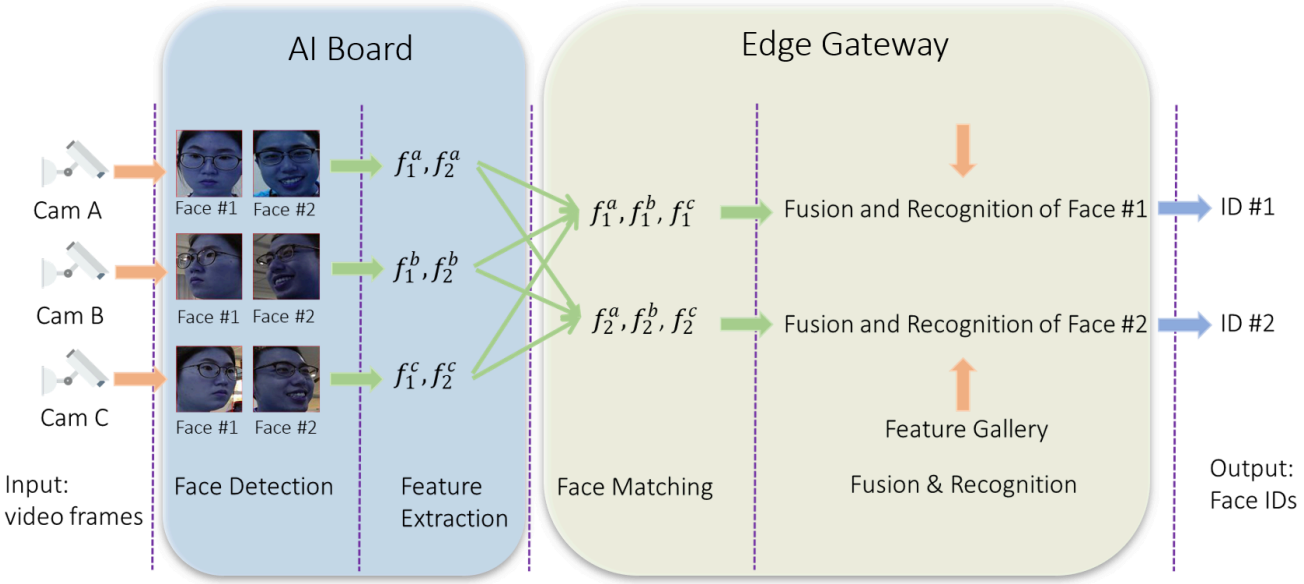


Figure 1.b Testing Nvidia Jetson Nano (*Group ONE*)

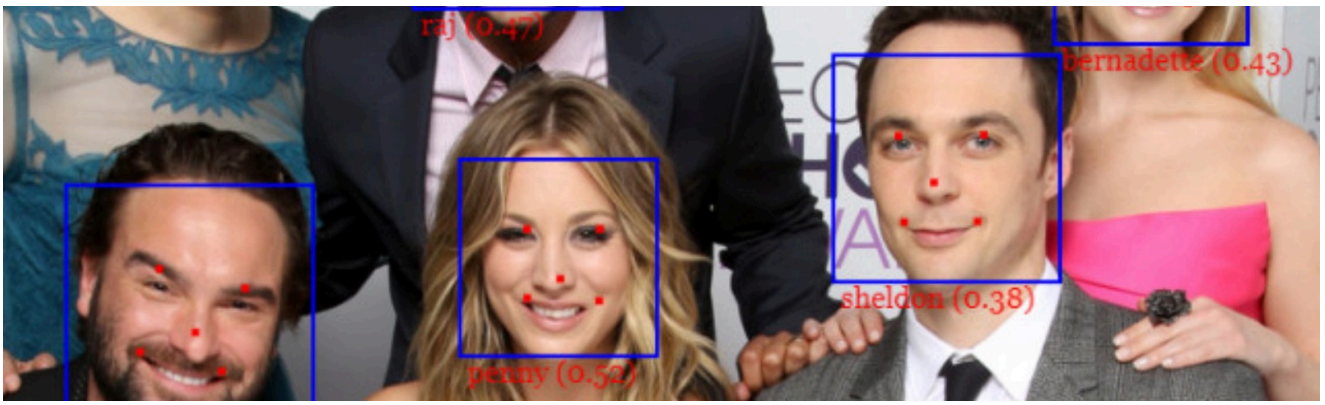
3. Scope

a. Project Deliverables

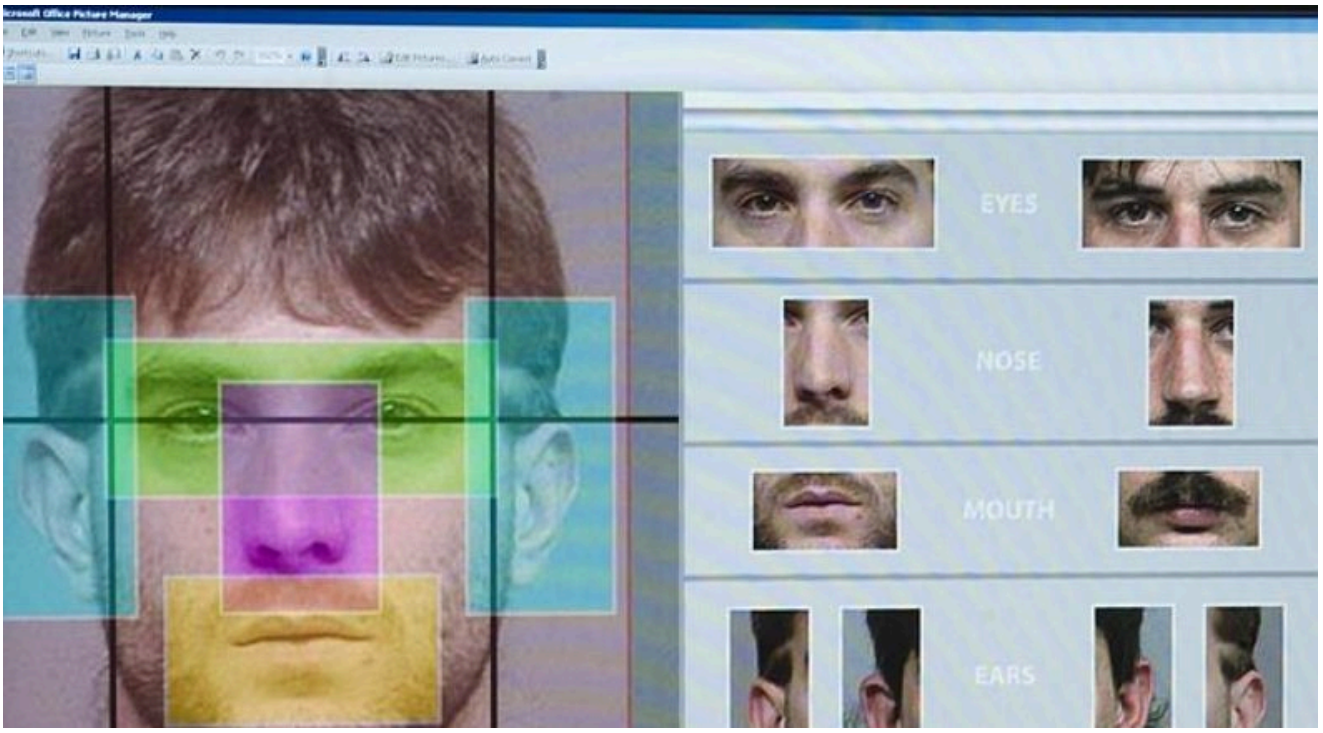
The goal of this project is to build a fully functioning real time multi-camera multi-face recognition system. The system is able to recognize faces of people that are in the database in real time.



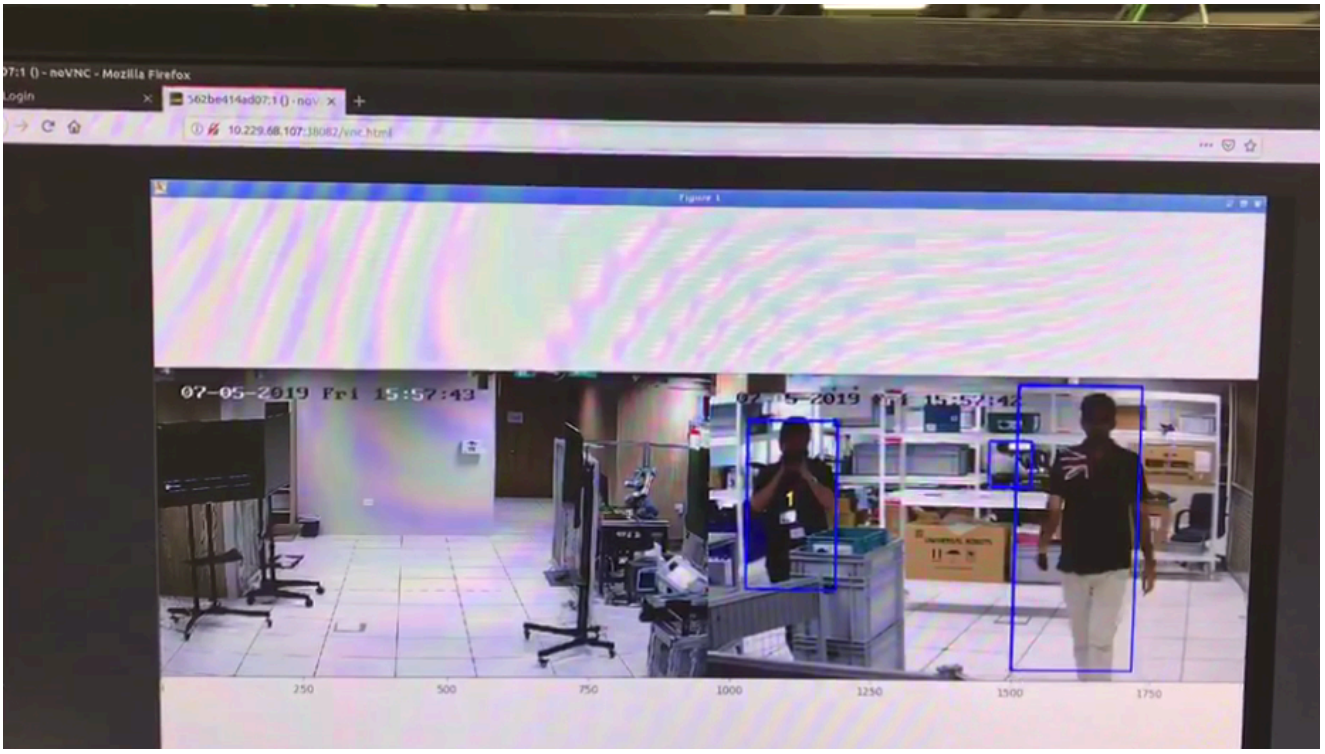
We will have multiple cameras working at the same time as shown in the picture above. The system could detect faces, extract their specific features, for example nose, eyes, lips, hair then fuse those features together and match them to existing data in the system (recognition). Then, on the GUI, name and information of that person will be shown in real time, as shown in the example below (output):



We constructed a database consisting of all faces as well as their respective features for the highest degree of precision. An example is presented below, as can be seen a face will be separated to different parts such as eyes, nose, mouth and ears.



In addition, API could be used to retrieve frames from a person or time stamp taken from processed video frames from the database. The GUI contains function to obtain these information that is user-friendly and easy-to-use. An instance is shown below, the system recognizes people inside the frame and can export with the time stamp on the top left corner of the frame.



The project is built based on computing power of a Raspberry Pi 4 board along with a Intel® Neural Compute Stick 2 to enhance computing potential.

b. Summary of activities

Our project is divided into three small groups: Hardware (Group 1), Software/GUI (Group 2), Software/Database (Group 3).

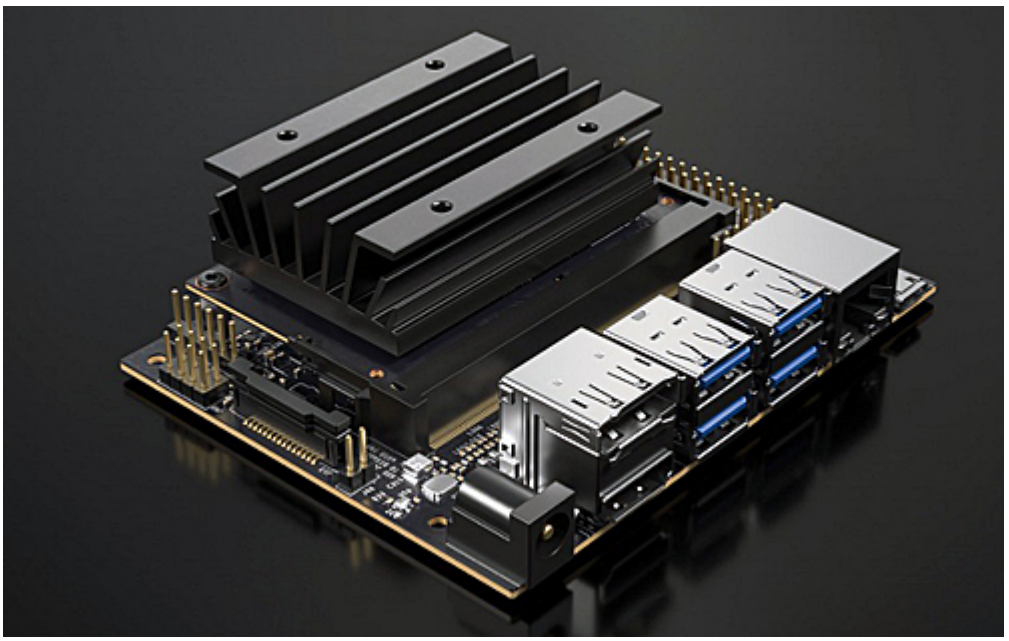
b.1. Hardware Implementation (Group 1)

Group 1 main objective is to figure out which board will be the most suitable for the project in terms of face detection time, face extraction time, power comsumption and compatibility. This group is also responsible for writing face detection and extraction algorithm. After that, they will pass the data from a number of input sets as well as the algorithm to group 2 and group 3.

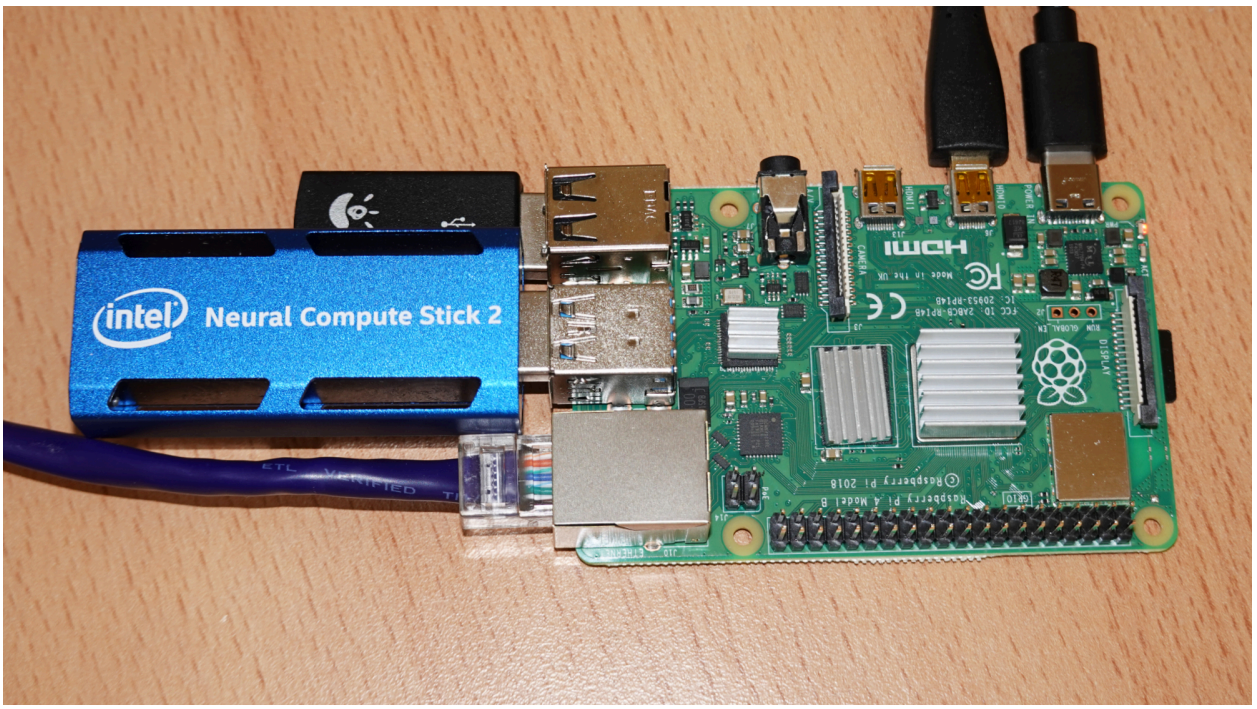
Group 1 planned to test four different setup:

Google Coral Edge TPU Board, Raspberry Pi (4 B) + Coral USB Accelerator: These hardware has not arrived in time so group 1 is not able to test it. Thus we left with 2 setups:

NVIDIA® Jetson Nano™: Despite the fact that NVIDIA Jetson Nano board has better processing ability, we decided to proceed with the Raspberry board due to the fact that it is more compatible with our source code.



Group 1 chose RaspBerry Pi 4 + Intel Neural Compute Stick 2 due to its compability with the whole system and the processing capability is not far behind Jetson Nano. We were not able to make the NVIDIA works as smoothly as RaspBerry Pi 4 due to the fact that our sourcecode is optimized for this specific board.

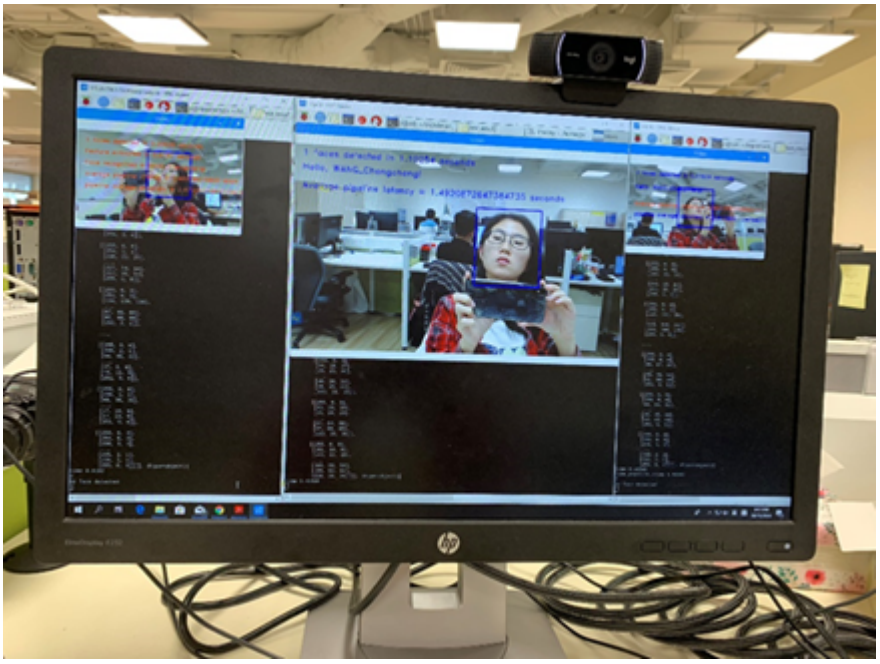


The tests are conducted using different datasets and various environment. We used photos and videos that have one faces or many faces with different size and resolutions. NVIDIA board came out as the fastest overall and RaspBerry Pi 4 came close at second place.

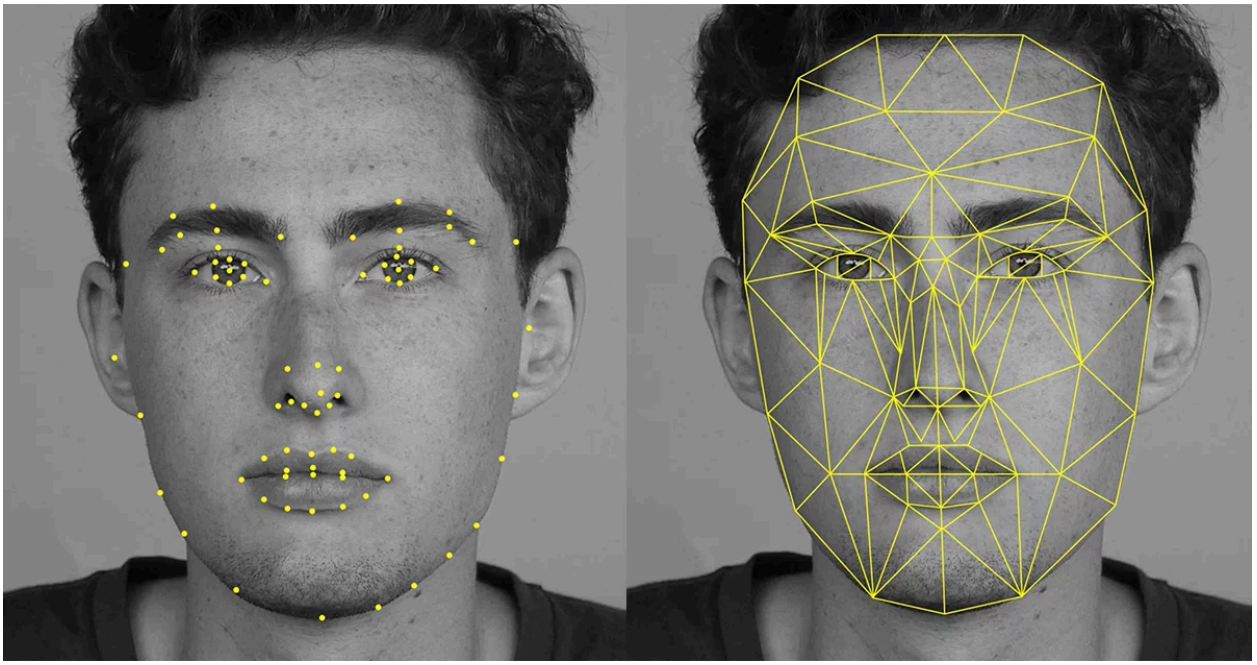
Group 1 is also responsible for the face detection and extraction algorithm. Face detection is built based on OpenCV and MTCNN library while feature extraction used MobileFaceNet, MobileNet v2 and ArcFace. The sourcecode is written in Python and work on most linux-based OS.

b.2. Software Implementation - GUI (Group 2)

Group 2 worked on the GUI and processed the data received from Group 1 as well as API provided by Group 3. Users could view results from any camera or all cameras via the GUI in real-time. They could choose to view based on time stamp or a specific person through the API.

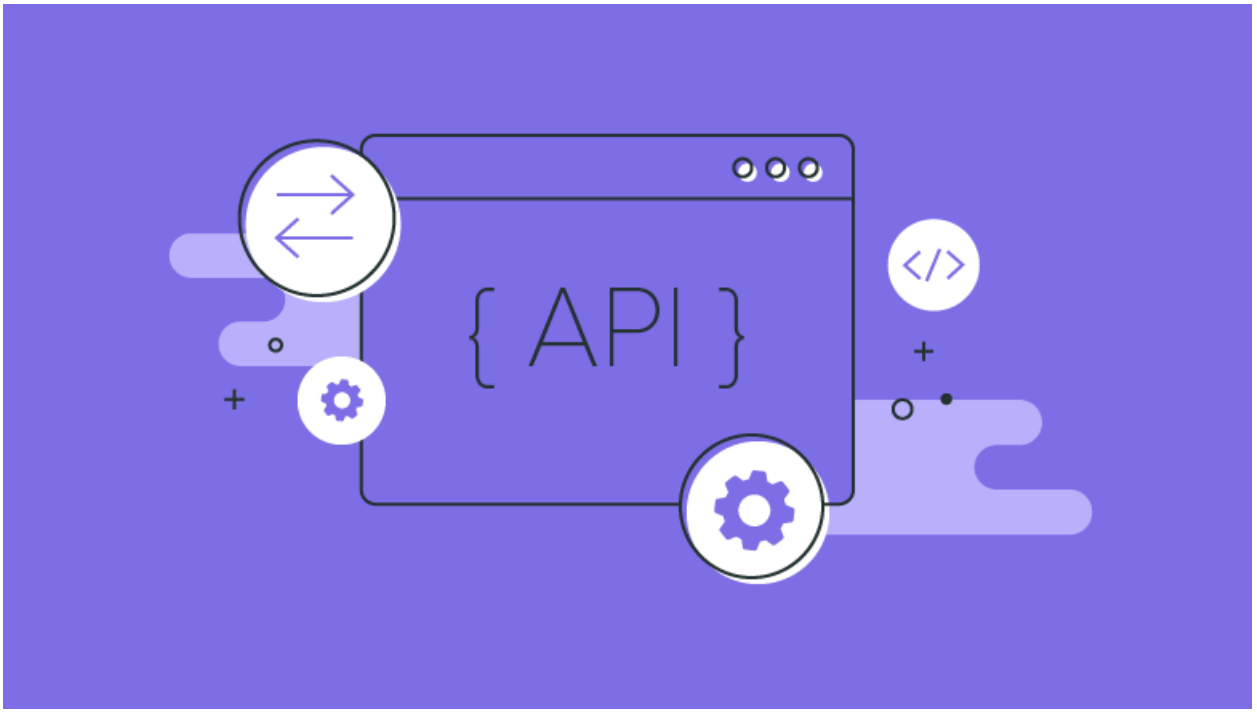


Group 2 received raw information so we also had to identify, match, fuse then recognize the person. From a dataset, we took extracted features from Group 1, identified then matched to person. After that, we fused all those fragments into one entity and performed facial recognition based on it. All of these happened in real time and users could witness it via our GUI. The whole program is written in Python.



b.3. Software Implementation - Database (Group 3)

Group 3 goal is to build a database using raw extracted data from Group 1 and provide the API for Group 2. The data provided is either name-based or time-based.



The Database is built using SQL and Python. We took raw data from group 1 and built a database with different people and we also stored time stamp along with the frames. Thus, users could easily called the necessary data via the API.

4. Schedule

PHASE	Planned Milestone Date	Actual Milestone Date
Initiating Phase	Week 1	Week 1
Planning Phase	Week 2	Week 2 - Week 6
Execution Phase	Week 3 - Week 11	Week 6 - Week 10
Closing Phase	Week 12 - 13	Week 11 - Week 12
Project End Date	Week 13	Week 13

Our planned milestone dates differ from actual milestone for quite a big part. One of the reasons was due to the fact that one of the boards was not here yet. The planning phase was actually throughout the whole process as there was certain misunderstanding on our scope of work in the initial part while we are executing individual responsibilities. All of us took quite a long time to execute our individual part as we were more focused on researching and understanding how to go about doing it. Programming was new to everybody in the team and we do not have strong fundamentals on coding. The only concept we had was Data Structure and Algorithm. It was hard to pinpoint the usefulness of our research resources as everybody was explaining many different ways to perform what we need to do. Other than researching based on our own project, we had to learn and understand the programming language of python. As we begin programming, we realised it's quite logical but it's very brain taxing because we had to think of the logic to let our code run. Concept such as while and for loop was very often used in the project and it is not as simple as how we were thought but essentially the concept was built upon on how we were taught previously in EE2008. There were a lot of complex issues that we faced during the execution part. The software library tends to be a little buggy at times. It varies from version

of CV and python. We had to try various way to get the result that we want when certain libraries does not work. Eventually towards the closing phrase, we were integrating programming from all three teams to ensure that all three of the programs can communicate with one another. Throughout the process, it was a lot of deadlocks and not knowing what to do next. We get stuck easily and very often we had to rest and wait a few days to get some ideas out. There was a lot of blank out periods due to that. It was pretty hard to actually express out the stuck that we had as some of it were minor after knowing how to fix. Whilst some was software issues that we were not able to explain and it eventually just got fixed.

As much as the project did not go according to our schedules, we benefitted from the process. We take longer time to research and to understand what we need to do. However, at the execution period, we took shorter time as we knew what we needed to do. We were able to pinpoint and be very focused on achieving the outcome that we want. I think lots of execution portion was mainly figuring out how to achieve that particular results that we want to see more than asking ourselves what is the next step. We were able to progressively get our project done in a rapid manner.

5. Cost

Each and every group of the Design and Innovation Project (DIP) is given a total budget of $1,600. Every purchase that is made should be approved by our Project Supervisor, Prof Tay Wee Peng. If the cost is less than $200, the components purchased can be taken as petty cash claim from Prof Tay. However, if the components were to cost more than $200, we - students, would have to inquire via email for approval from our supervisor, followed by the DIP Coordinator, Dr. Muhammad Faeyz Karim and the Area Lead. [1]

There is a difference between the planned cost and actual cost. The planned cost is cost that are needed and discussed in the initial stage of the planning, the actual cost is the total amount spent on the components after everything. Actual cost may differ from planned cost as there may be a discount factor or there maybe other equipment that can be used instead of getting a new one.

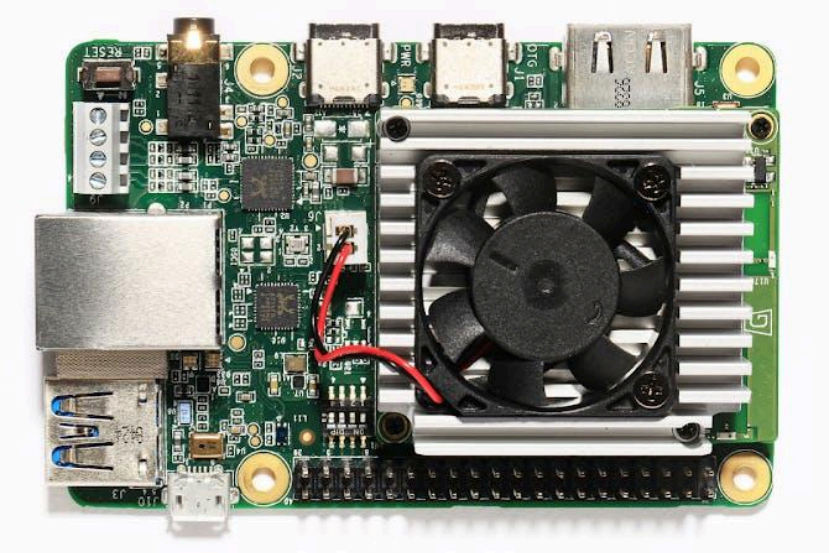
Hence, the table below are the Planned cost and Actual cost of our project:

PHASE	Planned Costs	Actual Costs
Initiating Phase	**$618.78**	0
Planning Phase	0	0
Execution Phase	0	0
Closing Phase	0	0
Project Total Costs	0	0

Break down of the planned costs:

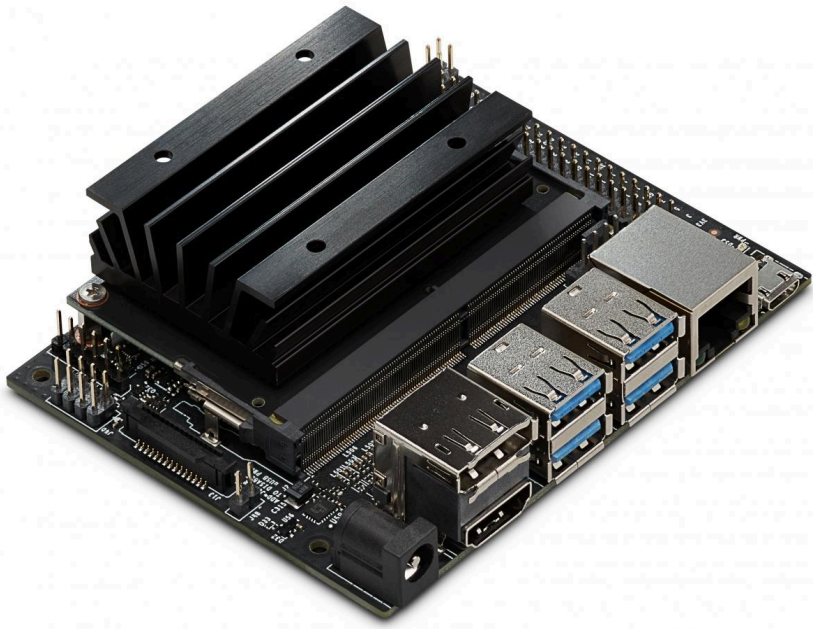
1.	Google Coral Edge TPU board	$204.36
2.	NVIDIA® Jetson Nano™ Developer Kit	$139.90
3.	Raspberry Pi (4 B)	$ 79.90
4.	Coral USB Accelerator	$ 84.95
5.	Intel® Neural Compute Stick 2	$109.67
		Total : $618.78

1. Google Coral Edge TPU board



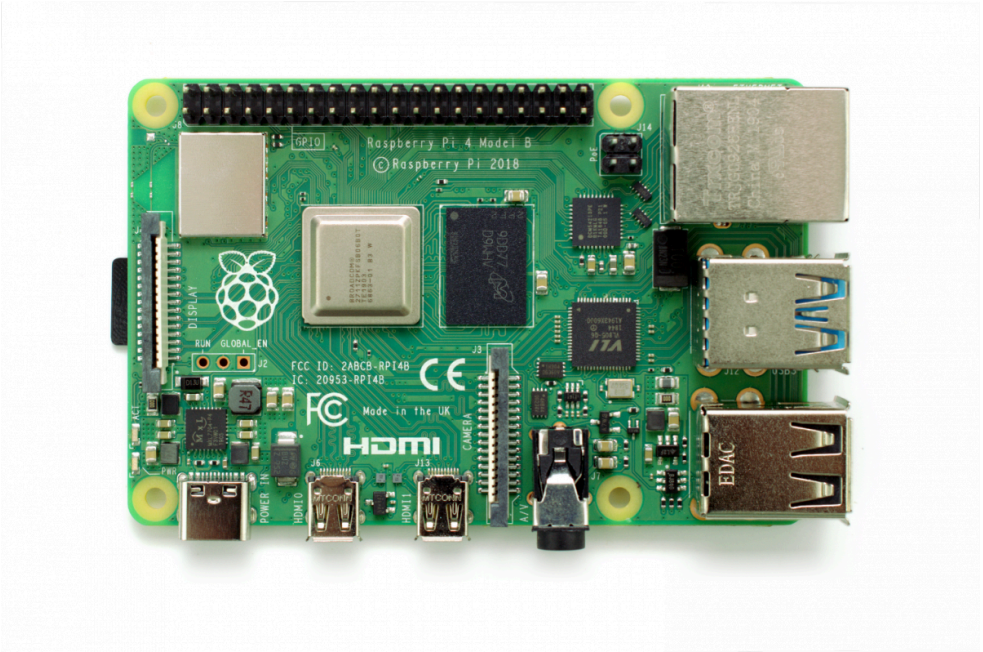
The Edge TPU is a small Application-Specific Integrated Circuit that was built by Google. It provides high performance Machine Learning interface at a low power cost. The Edge TPU enables users to execute multiple advanced Artificial Intelligence models per frame simultaneously, on a high-resolution video (30 frames per second), in a power-efficient manner. [2]

2. NVIDIA® Jetson Nano™ Developer Kit



The NVIDIA Jetson Nano Developer Kit may look small but it is a powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All these applications are put into a platform that only runs at 5 watts. [3]

3. Raspberry Pi (4B)



The Raspberry Pi 4 has a performance that is as good as a desktop personal computer (pc). It has a faster processor and Graphics Processing Unit (GPU), bigger and faster Random Access Memory (RAM) than compared to Raspberry Pi 3B+. Its microSD card slow is also twice as fast, the theoretical maximum of 50MBps versus 25MBps on the 3B+. [4]

4. Coral USB Accelerator



The Coral USB Accelerator is a device that allows microcontrollers to exploit the power of Artificial Intelligence applications such as image classification and object detection. This is more secure than having a cloud server which serves machine learning request and it can also reduce latency. [5]

5. Intel® Neural Compute Stick 2



The Intel Neural Compute Stick 2 is a simple device that just has to be fitted into any platform with a USB port to prototype and operate without cloud compute dependence. The Intel NCS 2 delivers 4 trillion operations per second with 8 times performance boost. It reduces time to prototype or tune neural networks with versatile hardware processing capabilities at a low cost. [6]

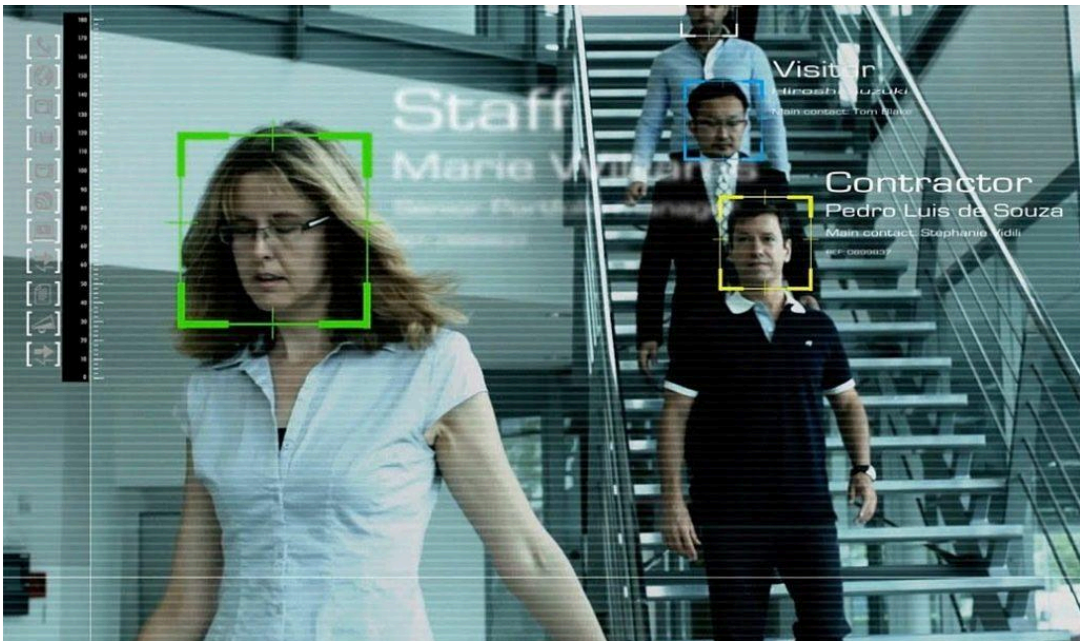
All these equipments are needed to test and select the board that works best for the purpose of this project. The Google Coral Edge TPU board was not bought as the delivery time of the board was unpredictable and hence was not ordered in time. The rest of the equipment was provided by the Teaching assistants as they had some spare and existing resources in labs should be used.

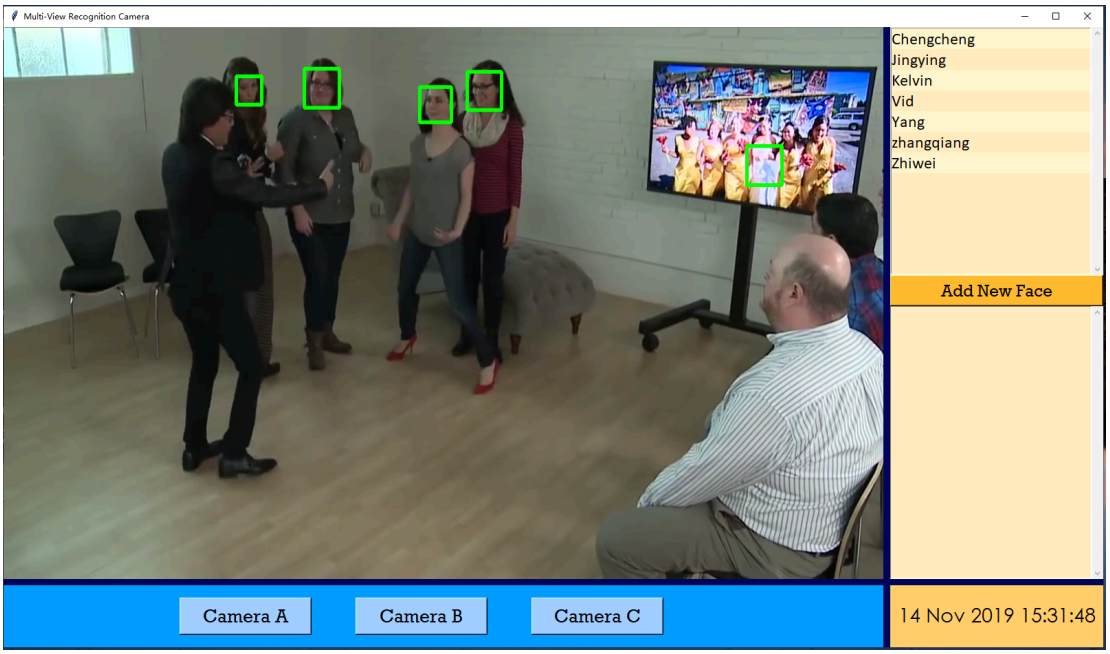
6. Outcomes / Benefits

- ***Outcomes***

The project has successfully produced a multiview face recognition system, using multiple video cameras, capable of detecting multiple human faces and identify them, together with a user interface that is convenient and user-friendly.

The system is able to detect faces on video cameras of different angles and store them in the database for future identification purposes. In each of the captured faces, the system will extract facial features and fuse them together to form a face vector to be classified under one person. The processing of images is achieved via implementing deep neural network module of the cv2 library version 3.3 by OpenCV. The new faces detected can be stored together with information of the people in a database, which includes the basic fields of Names, ID number and time stamp of the capturing of the faces. Moreover, further fields can be added to the database depending on the use of the system, such as surveillance, or attendance taking.





The user interface designed also allow users of the system to conveniently switch between cameras. Users can select between cameras to view the different angles of the faces being captured in real time. Moreover, users can choose to view the faces captured by searching the database. This will prompt the UI to make a request for the data from the database. From there the enquired data, if existing, will be sent over from the database through the API to be displayed on the UI. These data include the images captured itself, as well as information of the person and other relevant information depending on the application of the system.

The system run on Raspberry Pi, at low cost and power consumption. The Raspberry Pi, as mentioned above, cost SG$79.90, and the Neural Compute Stick 2 costs about

SG$95.90. Other equipment required includes Raspberry V2 camera module and SD memory cards. The facial detection process takes around 1. 24ms with using OpenCV model, while feature extraction process takes around 100ms. The idle current is at around 1.1A, while the peak current is at 1.4A which is not too much different. Compare to other boards that were tested, the combination of Pi 4 and Computing Stick is the most efficient in terms of performance.

- ***Benefits***

The system has a widespread range of applications, from surveillance to infrastructure access control, for both households and enterprises.

One of the applications is camera doorbell, where the facial recognition system is used instead of the actual bell. The system will be able to detect faces of those living in the household, when they reach the door and open the door automatically, while for those whose face is unknown, the system will be ringing the bell automatically to notify there is a guest. The user can choose to input the newly detected face of changer into the database and classify them as guests, so that next time they pay a visit, the host will know who is coming to visit them.



Another application of the system is building access, which can be used to replace the common method used nowadays such as RFID cards. The staff database of companies can be utilised for the system, where only one extra field is needed to store the face vectors of staff. Each face can be tagged with different level of access to different facilities in the building based on their designations. The time stamp of the image captured can also be used for another possible feature, which is monitoring signin and signout time of staff, mitigating the need of biometric fingerprint.

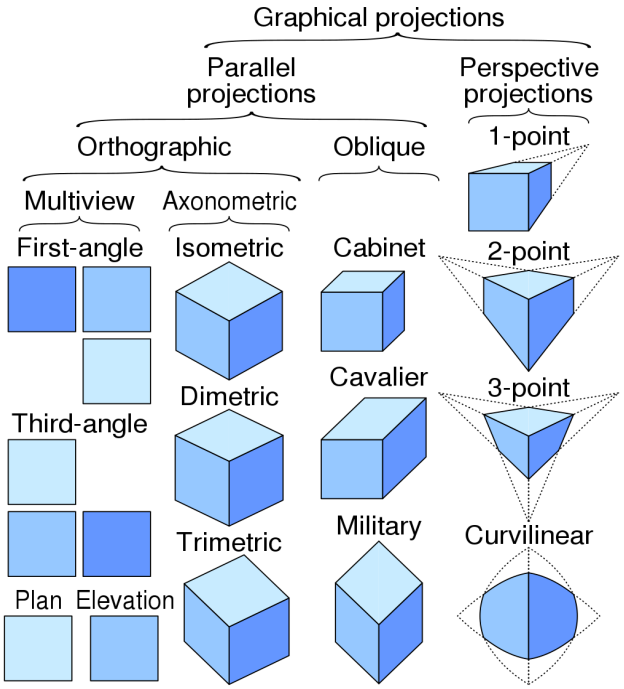


It can be seen that the applications of the system is various, which all serve to make life more convenient and efficient for everyone.

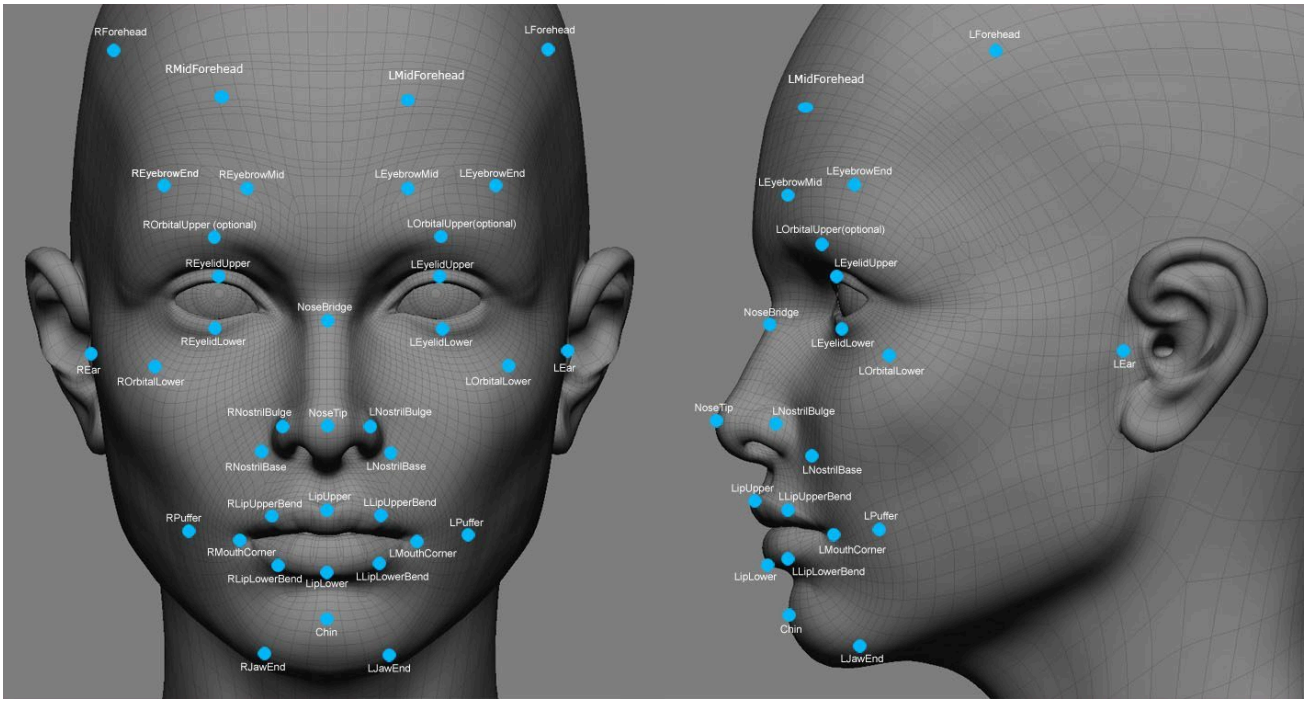
7. Reflection

1.Engineering knowledge

Multi-view: in technical drawing and computer graphics, a multiview projection is a technique of illustration by which a standardized series of orthographic two-dimensional pictures is constructed to represent the form of a three-dimensional object. Up to six pictures of an object are produced (called primary views), with each projection plane parallel to one of the coordinate axes of the object. The views are positioned relative to each other according to either of two schemes: first-angle or third-angle projection. In each, the appearances of views may be thought of as being projected onto planes that form a six-sided box around the object. Although six different sides can be drawn, usually three views of a drawing give enough information to make a three-dimensional object. These views are known as front view, top view and end view. Other names for these views include plan, elevation and section. In this experiment we are going to use three cameras to get multi-views of face of a person.



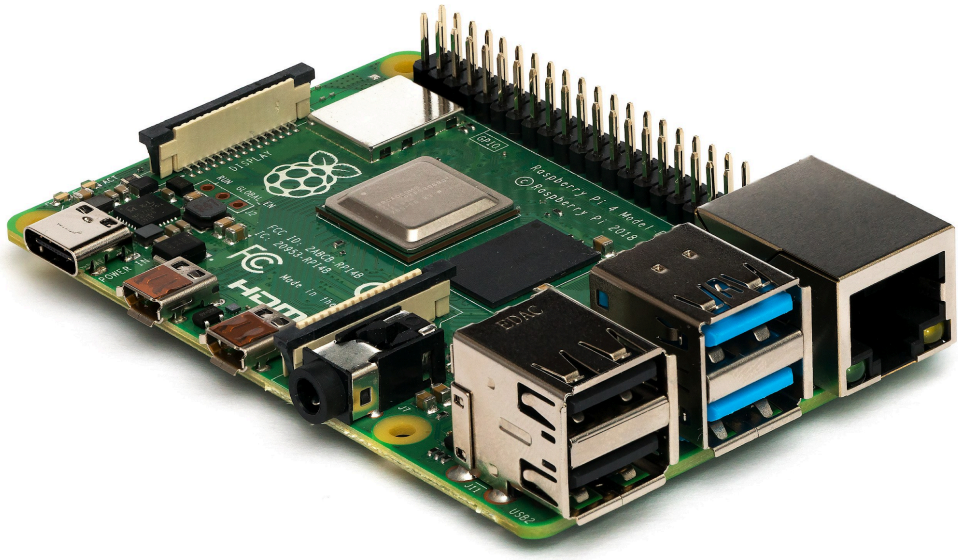
Face recognition: a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analysing patterns based on the person's facial textures and shapes.



And in this experiment, we are going to combine multi-view and face recognition together so that we can use cameras to capture different views of the face of a person then use face recognition system to know who he or she is.

2. Problem analysis

During this project, the largest problem we meet is how to combine hardware and software together. We are supposed to use computers to control Raspberry Pi, the hardware which contains cameras and is able to connect with computers.



And another big problem for everyone of us is we have no idea about what multi-view face recognition before is, most of us never even heard of Raspberry Pi.

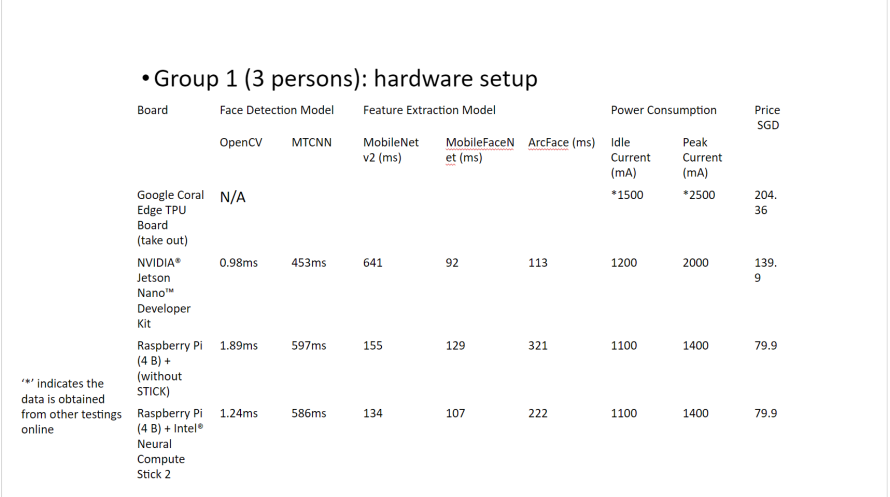
3.Design/development of Solutions

At the beginning of the project, we divided us 7 people into 3 teams so that everyone just need to do a small part of the project.

Then we try to figure out how to build the connection between software and hardware by searching from the internet or asking the mentors. At the first 6 weeks, we do our own job separately and at the last 6 weeks we all work together to combine three teams work. On week 8, we also did a process presentation, so that our mentors could know what we have done then they gave us a lot of suggestions.

4. Individual and Team Work

Our group of 7 are divided into 3 teams. Basically, Team 1 of three(**Liao Yan**, **NGO DAT PHUC** and **QUACH TRUNG QUAN**) are responsible for hardware setup, they are supposed to measure and compare efficiency among Google Coral Edge TPU board, NVIDIA Jetson Nano Developer Kit and Raspberry Pi then decide which board we are going to use. After that, they are going to figure out how to use that board they chose to connect to computer and how to use it to capture images from people.

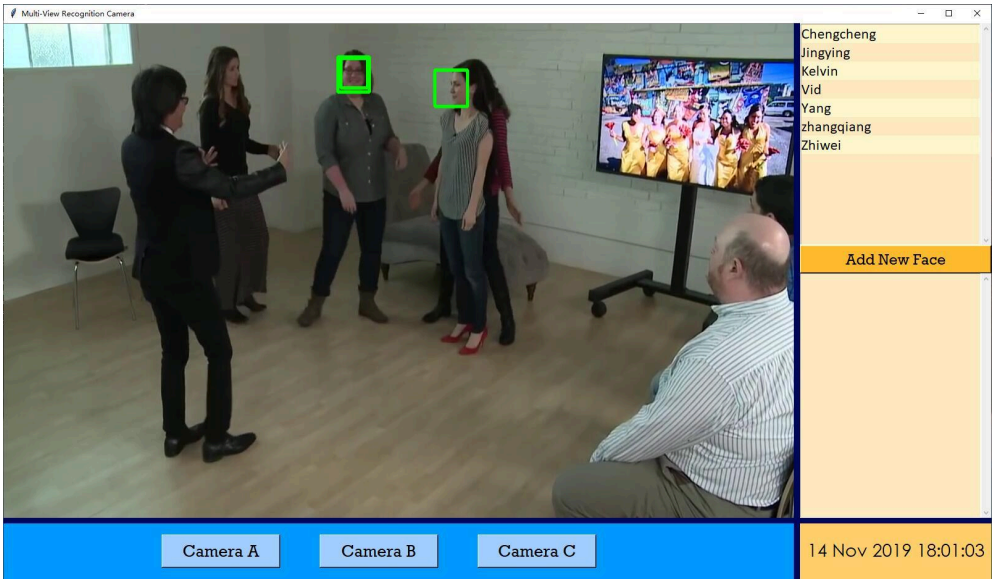


• Group 1 (3 persons): hardware setup

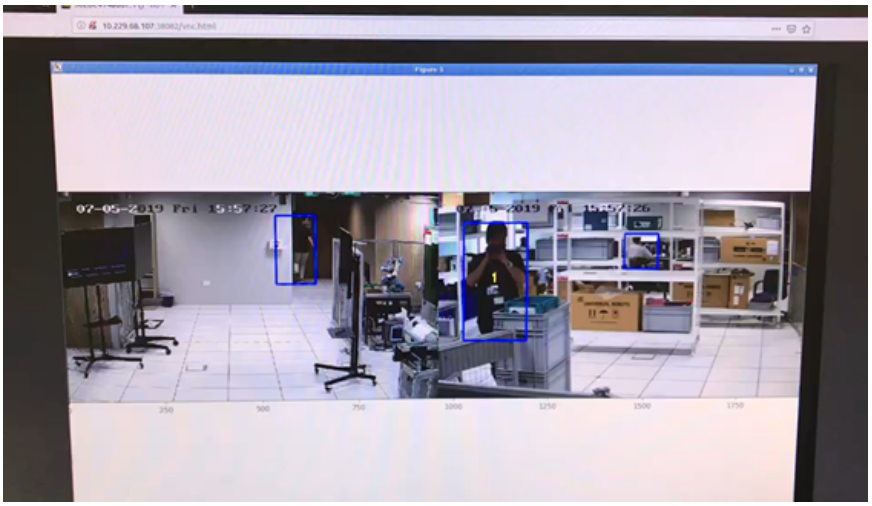
Board	Face Detection Model		Feature Extraction Model			Power Consumption		Price SGD
	OpenCV	MTCNN	MobileNet v2 (ms)	MobileFaceNet (ms)	ArcFace (ms)	Idle Current (mA)	Peak Current (mA)	
Google Coral Edge TPU Board (take out)	N/A					*1500	*2500	204.36
NVIDIA® Jetson Nano™ Developer Kit	0.98ms	453ms	641	92	113	1200	2000	139.9
Raspberry Pi (4 B) + (without STICK)	1.89ms	597ms	155	129	321	1100	1400	79.9
Raspberry Pi (4 B) + Intel® Neural Compute Stick 2	1.24ms	586ms	134	107	222	1100	1400	79.9

'*' indicates the data is obtained from other testings online

For team 2 of two(He Weiyang and Wen Xibo) are responsible for software implementation of a GUI. They will code a GUI to display real-time inference results from all cameras on gateway and To enable users to freely select to display results from specific cameras



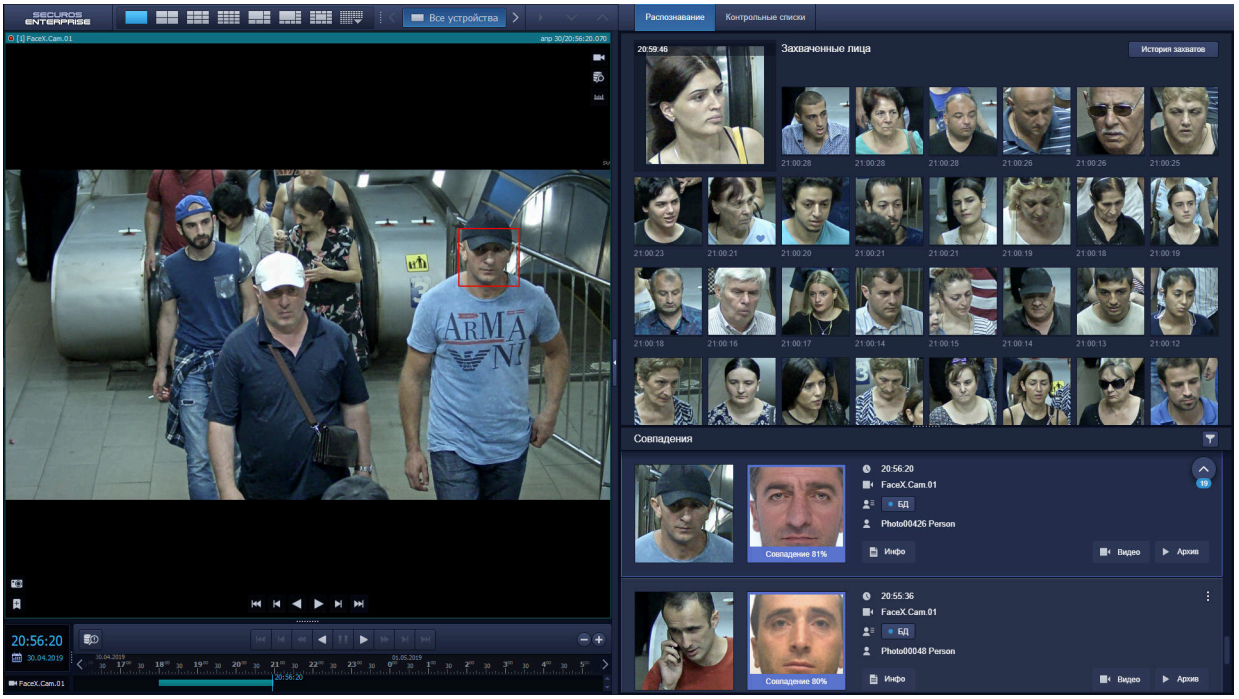
For team 3 of two(**WONG WENG SOON** and **SOH RUI XUAN DAPHNE**) are responsible for software implementation of a backend server/database. They will store processed video frames and code an API which is required to enable users to retrieve frames from specific persons and time instants



And **Liao Yan** is our group leader, he is not only going to do team 1's job but also mediate all 3 teams and tell mentors what we need.

5. Future recommendations

Actually what we have done in the last 12 weeks is a quite simple multi-view face recognition system. It looks not pretty enough and it's not as powerful and convenient as we want. What we want to build is like



Next time before the project, we think we may need to do more preparation about what we are doing and how to do it. And we need to practice more about our coding skills and learn more coding languages as during this project team 2 and 3 are supposed to use Python to code software part but most of us are not familiar with Python.

And we believe in the future, face recognition will perform a very important role in our daily life, even now, more and more phones are trying to use face ID to unlock. It is very convenient to identify or track people. What we have done is just the a of the iceberg.

References

[1] *Eee.ntu.edu.sg*, 2019. [Online]. Available: http://www.eee.ntu.edu.sg/programmes/CurrentStudents/undergraduate/undergraduatefull-time/Documents/dip/DIP_Briefing%20Slides.pdf. [Accessed: 14- Nov- 2019].

[2] "Edge TPU - Run Inference at the Edge | Edge TPU | Google Cloud", *Google Cloud*, 2019. [Online]. Available: https://cloud.google.com/edge-tpu/. [Accessed: 14- Nov- 2019].

[3]"Jetson Nano Developer Kit", *NVIDIA Developer*, 2019. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: 14- Nov- 2019].

[4]A. Piltch and G. Halfacree, "Raspberry Pi 4 Review: The New Gold Standard for Single-Board Computing", *Tom's Hardware*, 2019. [Online]. Available: https://www.tomshardware.com/reviews/raspberry-pi-4-b,6193.html. [Accessed: 14- Nov- 2019].

[5]G. Tanner, "Google Coral USB Accelerator Introduction", *Gilberttanner.com*, 2019. [Online]. Available: https://gilberttanner.com/blog/google-coral-usb-accelerator-introduction. [Accessed: 14- Nov- 2019].

[6]"Neural Compute Stick 2 - Intel | Mouser Europe", *Mouser.com*, 2019. [Online]. Available: https://www.mouser.com/new/intel/intel-neural-compute-stick-2/. [Accessed: 14- Nov- 2019].

Appendix A - Project Members Information

	Name	Project contributions	Report Contribution
1	He Weiyang	Project Report Co-writer **Group 2 Leader** UI designer Sample Code Editor Final Presentation Presenter	Pages 3 - 8
2	Liao Yan	**Project Leader** Project Charter Editor Project Report Co-writer **Group 1 Leader** Testing of Nvidia Jetson Nano Sample Code Editor Final Presentation Presenter	Pages 9 - 13 Appendix A, B References
3	QUACH TRUNG QUAN	Project Report Co-writer **Group 1 Member** Testing of Raspberry pi 4B (with Neural Compute Stick 2) Sample Code Editor	Pages 14 - 20
4	WONG WENG SOON	Project Report Co-writer **Group 3 Leader** API designer Final Presentation Presenter	Pages 21 - 22
5	SOH RUI XUAN DAPHNE	Project Report Co-writer **Group 2 Member**	Pages 23 - 27
6	NGO DAT PHUC	Project Report Co-writer **Group 1 Member** Testing of Raspberry pi 4B Sample Code Editor	Pages 28 - 32
7	Wen Xibo	Project Report Co-writer **Group 2 Member**	Pages 33 - 39

Sub-Group Members List

Group 1: Liao Yan, NGO DAT PHUC and QUACH TRUNG QUAN

Group 2: WONG WENG SOON and SOH RUI XUAN DAPHNE

Group 3: He Weiyang and Wen Xibo

Appendix B – (no page limit for appendices)

Data Sheets

1. Raspberry Pi 4B

Specification

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memory:	1GB, 2GB or 4GB LPDDR4 (depending on model)
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics
SD card support:	Micro SD card slot for loading operating system and data storage
Input power:	5V DC via USB-C connector (minimum 3A[1]) 5V DC via GPIO header (minimum 3A[1]) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature 0–50°C
Compliance:	For a full list of local and regional product approvals, please visit https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md
Production lifetime:	The Raspberry Pi 4 Model B will remain in production until at least January 2026.

2. Intel® Neural Compute Stick 2

INTEL® NEURAL COMPUTE STICK 2

Technical Specifications

Specifications	Intel® Neural Compute Stick 2
Vision Processing Unit (VPU)	The Intel® Movidius™ Myriad™ X VPU
Software development kit	The Intel® Distribution of OpenVINO™ toolkit
Operating Systems support	Ubuntu* 16.04.3 LTS (64 bit), Windows® 10 (64 bit), or CentOS* 7.4 (64 bit)
Supported framework	TensorFlow* and Caffe*
Connectivity	USB 3.1 Type-A, USB 2.0 Type-A
USB stick dimensions	72.5mm X 27mm X 14mm
Operating temperature	0° - 40° C
Material Master Number	964486
MSRP	$69 USD July 14, 2019

3. NVIDIA Jetson Nano

Feature	Specification
GPU	128-core NVIDIA Maxwell™
CPU	64-bit Quad-core ARM A57 (1.43 GHz)
Memory	4 GB 64-bit LPDDR4 (25.6 GB/s bandwidth)
Networking	10/100/1000 Mbit Ethernet
Camera	12 MIPI CSI-2 DPHY 1.1 lanes (1.5 Gb/s), Dual ISB (1.5 Gpix/s)
Module Mechanical Specifications	69.6 mm x 45 mm, 260-pin SODIMM connector
Module Interfaces	
USB	3x USB 2.0 ports and 1x USB 3.0 port
PCIe	1 (x1/x2/x4)
Camera	CSI 3 x4 **or** 2 x4 + 2 x2 **or** 1 x4 + 3 x2
Display	HDMI 2.0 **or** DP 1.2, eDP 1.4, DSI (1 x2) 2 simultaneous displays
Audio (I2S)	2x
SDIO	1x
Gigabit Ethernet	yes
I2C	3x
UART	3x
SPI	2x
Module Mechanical Specifications	69.6 mm x 45 mm, 260-pin SODIMM connector
Fan	PWM and Tach input
DevKit Interfaces	
USB	4x USB 3.0 Type A ports, 1x USB Micro-B
Display	HDMI 2.0, DisplayPort 1.2
40-Pin Header	GPIOs, I2C, I2S, SPI, PWM, UART
Camera	MIPI-CSI camera Connector
Storage	MicroSD card slot
Other IO	M.2 Wi-Fi card slot, Gigabit Ethernet, Fan connector, PoE connector, UART header

● Group 1 Board Testing Result Table

• Group 1 (3 persons): hardware setup

Board	Face Detection Model		Feature Extraction Model			Power Consumption		Price SGD
	OpenCV	MTCNN	MobileNet v2 (ms)	MobileFaceNet (ms)	ArcFace (ms)	Idle Current (mA)	Peak Current (mA)	
Google Coral Edge TPU Board (take out)	N/A					*1500	*2500	204.36
NVIDIA® Jetson Nano™ Developer Kit	0.98ms	453ms	641	92	113	1200	2000	139.9
Raspberry Pi (4 B) + (without STICK)	1.89ms	597ms	155	129	321	1100	1400	79.9
Raspberry Pi (4 B) + Intel® Neural Compute Stick 2	1.24ms	586ms	134	107	222	1100	1400	79.9

'*' indicates the data is obtained from other testings online