

**Face Reidentification System to Track Factory Visitors using OpenVINO**

BY

Wong Yiek Heng

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

For the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2020

## REPORT STATUS DECLARATION FORM

Title: \_\_\_\_\_

Face Reidentification System to Track Factory Visitors using OpenVINO

Academic Session: 202005

I WONG YIEK HENG

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

\_\_\_\_\_  
(Author's signature)

Verified by,

\_\_\_\_\_  
(Supervisor's signature)

Address:

H-71, Jalan Kelapa Sawit 14,

81000, Kulai, Johor

Teoh Shen Khang

\_\_\_\_\_  
Supervisor's name

Date: 4 SEPTEMBER 2020

Date: 10 September 2020

**Face Reidentification System to Track Factory Visitors using OpenVINO**

BY

Wong Yiek Heng

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

For the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS)

COMPUTER ENGINEERING

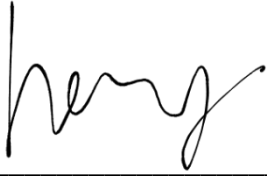
Faculty of Information and Communication Technology

(Perak Campus)

MAY 2020

## **DECLARATION OF ORIGINALITY**

I declare that this report entitled “**FACE REIDENTIFICATION SYSTEM TO TRACK FACTORY VISITORS USING OPENVINO**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : WONG YIEK HENG

Date : 4 SEPTEMBER 2020

## **ACKNOWLEDGEMENTS**

I would like to express my gratitude towards my supervisor, Mr. Teoh Shen Khang for giving me a chance and given me this bright opportunity to engage in this Machine Learning Project. It is my first step to establish a career in Artificial Intelligent field. A million thanks to you.

Besides, thanks to my friends for giving me unconditional support throughout the project, which directly encourage me to keep moving on. Finally, I must thank my parents for showing me their love, support and massive encouragement throughout the process.

## **ABSTRACT**

This paper introduces a facial recognition-based visitor tracking and reidentification system using OpenVINO. The system can track people based on their faces to track their identities. OpenVINO make CNN deep learning inference on the edge. It supports heterogeneous execution across computer vision accelerators, decrease the time to market through function of libraries which optimized calls for OpenCV and OpenVX and pre-optimized kernels.

Every book that I've referred to in the past few years has the author always mention that deep learning requires a lot of computational power to run on. That is the reason why we need GPU. GPU are usually used to train CNN models and include tons of calculations to make predictions. Company NVIDIA uses CUDA and cuDNN for deep learning, providing the best GPU and best software support. However, GPUs are relative extremely expensive which typically cost 2-3 times that of CPU compute instances. Hence, you may not run you deep learning model inferencing on inexpensive devices. For example, you won't apply an expensive GPU that costs thousands of ringgits to an IoT devices such as (Raspberry PI or Intel UP Squared board). Fortunately, developer may abstract away this difficulty by implement OpenVINO onto IoT devices and make it use CPU to inference on real time.

In addition, as the datasets grow larger, traditional neural network prediction methods will no longer be accurate and fast enough. OpenVINO affect the performance of inference, OpenVINO optimizes multiple calls in the traditional computer vision algorithm implemented in OpenCV, and performs specific optimizations for deep learning inference. Developer may get the benefit when using OpenCV with OpenVINO.

In this project, the development environment, running environment and the playground for running OpenVINO will be setup. At the end, two system will be developed which are "Face Registration System" and "Face Reidentification System". These two systems will apply to separate devices and make them run concurrently, so that the face reidentification process will be able run on embedded board without using any expensive component.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>ii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>ABSTRACT</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xii</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Project Scope	6
1.3 Project Objectives	7
1.4 Impact, significance and contribution	8
1.5 Background information	9
<b>Chapter 2: Literature Review</b>	<b>12</b>
2.1 ROS_OpenVINO	12
2.2 ROS2_OpenVINO	13
2.3 OpenFace	14
2.4 Person-Reidentification	16
2.5 MTCNN4Android	17
2.6 OpenVINO Smart Classroom C++ Demo	19
<b>Chapter 3: System Methodology</b>	<b>20</b>
3.1 Design Specifications	20
3.1.1 Hardware and software involved	20
3.1.2 System Development Method	28
3.1.3 Verification Plan	28
3.1.4 Expected System Testing and Performance	31

<b>3.2 System Design / Overview</b>	<b>33</b>
<b>3.2.1 face-detection-retail-0004</b>	<b>34</b>
<b>3.2.2 landmarks-regression-retail-0009</b>	<b>35</b>
<b>3.2.3 face-reidentification-retail-0095</b>	<b>36</b>
<b>3.3 Methodology</b>	<b>38</b>
<b>3.4 Project Milestone</b>	<b>39</b>
<b>3.4.1 FYP1 Milestone</b>	<b>39</b>
<b>3.4.2 FYP2 Milestone</b>	<b>40</b>
<b>Chapter 4: System Design and Implementation</b>	<b>41</b>
<b>4.1 System Flow (Software Architecture)</b>	<b>41</b>
<b>4.2 Functional Modules in the Basic Structure</b>	<b>42</b>
<b>4.2.1 Visualizer</b>	<b>42</b>
<b>4.2.2 Frame Processor</b>	<b>42</b>
<b>4.2.3 Face Detector</b>	<b>42</b>
<b>4.2.4 Landmark Detector</b>	<b>42</b>
<b>4.2.5 Faces Database</b>	<b>43</b>
<b>4.2.6 Face Identifier</b>	<b>43</b>
<b>4.3 System Flow Micro View</b>	<b>44</b>
<b>4.4 Hardware Connection (Hardware Architecture)</b>	<b>46</b>
<b>4.5 Flow Chart of Face Registration System</b>	<b>47</b>
<b>4.6 Flow Chart of Face Reidentification System</b>	<b>49</b>
<b>4.7 Refining Prototype</b>	<b>50</b>
<b>4.8 Final Product</b>	<b>51</b>
<b>4.8.1 Development Environment Setup</b>	<b>52</b>
<b>4.8.2 Face Registration System</b>	<b>56</b>
<b>4.8.3 GUI of Face Registration System</b>	<b>59</b>
<b>4.8.4 Face Reidentification System</b>	<b>63</b>
<b>4.8.5 GUI of Face Reidentification System</b>	<b>63</b>
<b>4.9 Use Cases</b>	<b>66</b>



<b>Chapter 5: System Evaluation and Discussion</b>	<b>67</b>
<b>5.1 Face Detection</b>	68
<b>5.1.1 Face Detection – 1 Face</b>	69
<b>5.1.2 Face Detection – 2 Face</b>	70
<b>5.1.3 Face Detection – 3 Face</b>	71
<b>5.1.4 Face Detection – 4 Face</b>	72
<b>5.1.5 Face Detection Test Result</b>	73
<b>5.2 Face Reidentification</b>	74
<b>5.2.1 Face Reidentification – 1 Face</b>	75
<b>5.2.2 Face Reidentification – 2 Face</b>	76
<b>5.2.3 Face Reidentification – 3 Face</b>	77
<b>5.2.4 Face Reidentification – 4 Face</b>	78
<b>5.2.5 Face Identification Test Result</b>	79
<b>5.3 Limitation and Future Enhancement</b>	80
<b>Chapter 6: Conclusion</b>	<b>81</b>
<b>Bibliography</b>	<b>83</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 1.1	Face Identification Process Flow	2
Figure 1.2	Artificial Neural Network	4
Figure 1.3	Convolutional Neural Network	4
Figure 1.4	Industry 4.0	11
Figure 1.5	Smart Factory	11
Figure 2.1	Hierarchical Architecture Design	13
Figure 2.2	Flow of OpenFace classification	14
Figure 2.3	ROC Curves	15
Figure 2.4	Architecture about improved reidentification	16
Figure 2.5	Result of MTCNN4Android	17
Figure 2.6	OpenVINO Smart Classroom Demo	19
Figure 3.1	Ubuntu 16.04 LTS	20
Figure 3.2	Python Programming	21
Figure 3.3	Intel OpenVINO	21
Figure 3.4	Camera	22
Figure 3.5	Intel UP Board	22
Figure 3.6	OpenCV	23
Figure 3.7	Python Tkinter	23
Figure 3.8	face-detection-retail-0004 IR files	24
Figure 3.9	face-detection-retail-0004 network architecture	24
Figure 3.10	landmarks-regression-retail-0009 IR files	25
Figure 3.11	landmarks-regression-retail-0009 network architecture	25
Figure 3.12	face-reidentification-retail-0095 IR	26
Figure 3.13	face-reidentification-retail-0095 network architecture	26
Figure 3.14	Python Socket Programming	27
Figure 3.15	Prototype	28
Figure 3.16	Simply Draft on Face Reidentification Process Flow	33
Figure 3.17	Model Structure of face-reidentification-retail-0095	36

Figure 3.18	SSD default boxes at 8x8 and 4x4 feature map	37
Figure 3.19	Face reidentification using one-shot learning	38
Figure 3.20	Comparing Facial Embedding	38
Figure 4.1	System Flow of registration and reidentification system	41
Figure 4.2	Basic System Data Flow	44
Figure 4.3	Hardware Architecture of system	46
Figure 4.4	Flow Chart of Face Registration System	47
Figure 4.5	Flow Chart of Face Reidentification System	49
Figure 4.6	Hardware Configuration of Face Registration System and Face Reidentification System	512
Figure 4.7	Ubuntu 16.04 LTS Version	52
Figure 4.8	pip3 command	52
Figure 4.9	Tkinter Version	53
Figure 4.10	OpenCV (OpenVINO)	53
Figure 4.11	Content of requirement.txt	53
Figure 4.12	Folder hierarchy of OpenVINO	54
Figure 4.13	Result of OpenVINO Verification Script	54
Figure 4.14	Successful of installing OpenVINO	55
Figure 4.15	Folder hierarchy of Face Registration System	56
Figure 4.16	Registered Face Image in gallery folder	56
Figure 4.17	Registered Face Image in gallery folder 2	56
Figure 4.18	IP_controller.py	57
Figure 4.19	Port Listening at 5001	58
Figure 4.20	DevicesList.json	58
Figure 4.21	GUI of Face Registration System	59
Figure 4.22	Save Face Window	59
Figure 4.23	New Face Detected	60
Figure 4.24	Registered Face Detected	60
Figure 4.25	Duplicated Face Image Save	61
Figure 4.26	Duplicated Face Image Save as new identity	61
Figure 4.27	Multiple Face at Face Registration System	62
Figure 4.28	Folder hierarchy of Face Reidentification System	63
Figure 4.29	GUI of Face Reidentification System	63

Figure 4.30	Register IP Window	64
Figure 4.31	Registered Face Detected	64
Figure 4.32	Unauthorize Face Detected	65
Figure 4.33	Use Case of Face Reidentification System	66
Figure 5.1	Flow Chart of Face Detection Verification Script	68
Figure 5.2	1 Face Detection	69
Figure 5.3	Accuracy of 1 Face Detection	69
Figure 5.4	2 Face Detection	70
Figure 5.5	Accuracy of 2 Face Detection	70
Figure 5.6	3 Face Detection	71
Figure 5.7	Accuracy of 3 Face Detection	71
Figure 5.8	4 Face Detection	72
Figure 5.9	Accuracy of 4 Face Detection	72
Figure 5.10	Relationship between accuracy and no of faces in face detection	73
Figure 5.11	Relationship between execution speed and no of faces in face detection	73
Figure 5.12	Flow Chart of Face Reidentification Verification Script	74
Figure 5.13	1 Face Reidentification	75
Figure 5.14	Accuracy of 1 Face Reidentification	75
Figure 5.15	2 Face Reidentification	76
Figure 5.16	Accuracy of 2 Face Reidentification	76
Figure 5.17	3 Face Reidentification	77
Figure 5.18	Accuracy of 3 Face Reidentification	77
Figure 5.19	4 Face Reidentification	78
Figure 5.20	Accuracy of 4 Face Reidentification	78
Figure 5.21	Relationship between accuracy and no of faces in face reidentification	79
Figure 5.22	Relationship between execution speed and no of faces in face reidentification	79

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 3.1	Confusion Matrix	28
Table 3.2	Testbench	31
Table 3.3	FYP1 Milestone	39
Table 3.4	FYP2 Milestone	40

## LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>ANN</i>	Artificial Neural Network
<i>AP</i>	Average Precision
<i>API</i>	Application Program Interface
<i>CNN</i>	Convolutional Neural Network
<i>CPU</i>	Central Processing Unit
<i>FLOPS</i>	Floating-point Operations Per Second
<i>FPGA</i>	Field Programmable Gate Arrays
<i>GPU</i>	Graphical Processing Unit
<i>GUI</i>	Graphical User Interface
<i>IoT</i>	Internet of Things
<i>LTS</i>	Long Term Support
<i>LBW</i>	Labeled Faces in the Wild
<i>MTCNN</i>	Multi-task cascade NN
<i>NASA</i>	National Aeronautics and Space Administration
<i>OpenVINO</i>	Open Visual Inference & Neural Network Optimization
<i>O-Net</i>	Output Network
<i>P-Net</i>	Proposal Network
<i>R-Net</i>	Refinement Network
<i>ROC</i>	Receiver Operating Characteristic
<i>SME</i>	Small and Medium Enterprise
<i>TCP</i>	Transmission Control Protocol
<i>VPU</i>	Movidius Vision Processing Units
<i>et al</i>	and others.

# Chapter 1: Introduction

## 1.1 Problem Statement and Motivation

There are always groups of visitors on a large company site, such as laboratory. It is possible that an unauthorized person may enter an area where he/she should not be. Hence, a system is able to handle such situation by display the position and smartly identify the visitors is needed. If not, the confidential document and technologies may be stolen and result in rate of commercial crime become higher. This system not only make sure the privacy and confidentiality of a company but also a step toward a brighter future. That's one small step for man, one giant leap for mankind.

Other than that, individualism and privacy and confidential of a person become more and more important. Everybody needs to have the right of their own so that identify a person become more and more important. There is a common way to differentiate and uniquely identify a person which is using the human face. Human face is multidimensional, complex, and dynamic object which has a high degree of variability in its appearance. Human face reveals a lot of information like telling us about mood, intention, attentiveness and the most important which is serve to identify a person. Moreover, facial identification is of a non-contact nature which is far better than other biometric techniques. Hence, face is one of the proper keys to help us to further identify a person instead of voice, finger print and other.

Facial identification is also known as a good security measure for attendance and time tracking. (Jason Brownlee, 2019) Human facial structure is an individual characteristic, no user can easily imitate another person. Factory visitors may be tracked by using camera to capture the face and identify the person and a person cannot imitate another person like student help his friend to sign even his friend is not in class. Enterprise, University and Government may need this kind of system to track their staffs, students, visitors instead of using sign or punch in system.

Besides that, one of the most important features of facial identification is facial identification is less processing involved which is cheaper than other techniques. Unlike the other biometric techniques, facial identification can be done by an embedded board and a camera, some algorithm to be performed on face identification without need of using the other expensive equipment such as vein recognition technology have to use multi infrared light sensor to sense the pulsation of vein and acquire and apply it into a high-performance computer to calculate and identify a person.

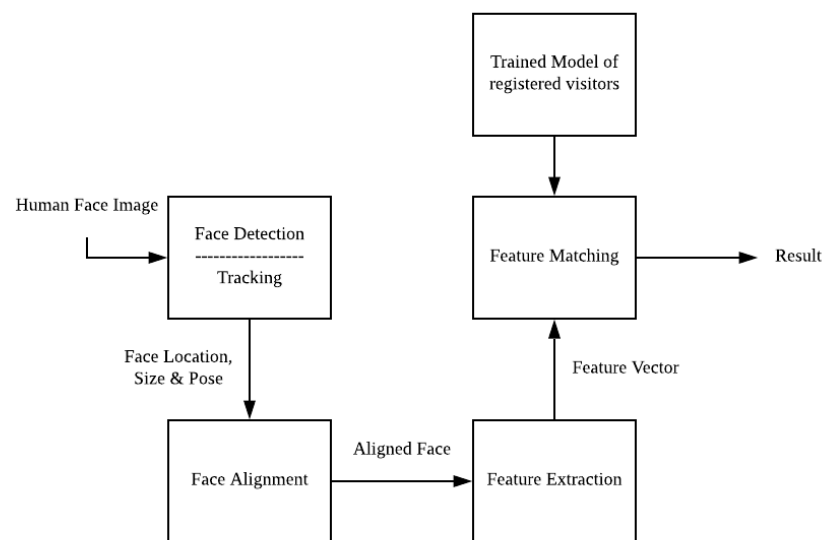


Figure 1.1: Face Identification Process Flow



Convolutional neural network (CNN) is an artificial neural network (ANN) used for image recognition and processing, and is dedicated to processing pixel data. (Damilola Omoyiwola, 2018) CNN is a powerful image processing, artificial intelligence (AI) that uses deep learning to perform generating and descriptive tasks. A neural network is a hardware and software system that composes images after the neurons in the human brain are running. Traditional neural networks are not suitable for image processing, and images must be input with reduced-resolution images. CNN's "neurons" are arranged more like the frontal lobe, which is the area responsible for visual stimulation of people and other animals. The neuron layer is arranged in such a way as to cover the entire field of view, avoiding the fragmentation image processing problems of traditional neural networks. The system used by CNN is very similar to a multi-layer perceptron designed to reduce processing requirements. The CNN layer consists of an input layer, an output layer, and a hidden layer. The hidden layer includes multiple convolutional layers, pooling layers, fully connected layers, and standardized layers. Eliminating restrictions and improving the efficiency of image processing has led to a system that is more efficient and easier for training limited by image processing and natural language processing.

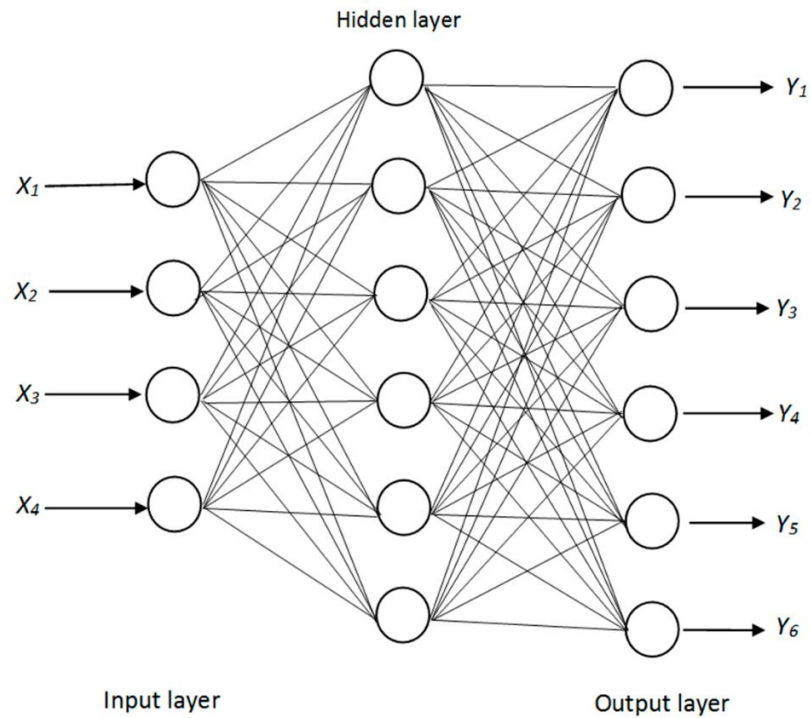


Figure 1.2: Artificial Neural Network (ANN)

Figure 1.2 is an example of artificial neural network. Features may be inputted (Input layer) and system will smartly calculate the weight (Hidden layer) and finally have the outcomes (Output layer) to identify.

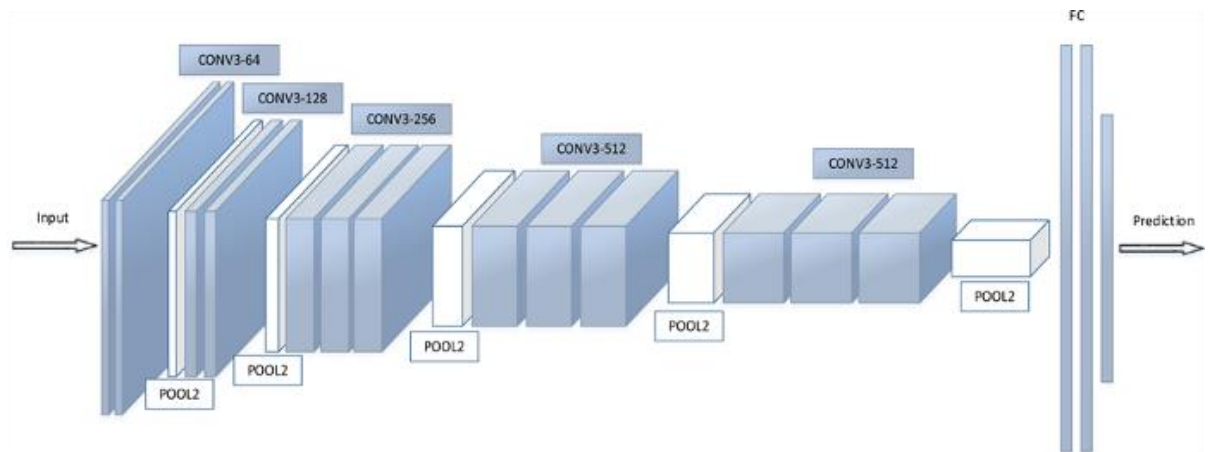


Figure 1.3: Convolutional Neural Network (CNN)

After a neural network is trained, it is deployed to run inference to classify, recognize and process new inputs and this can be speed up by using Intel Open Visual Inference & Neural Network Optimization (OpenVINO). OpenVINO toolkit is offered in the comprehensive line-up of Intel Vision Products that accelerate deep learning. (Intel Sdn Bhd,2018) Intel's regular central processing units and specialized accelerator hardware work with OpenVINO, which drives deep learning at the edge and positions Intel as the clear industry leader in enabling IoT solutions from the edge to the cloud. OpenVINO enables users to quickly track the development of high-performance computer vision applications, unleash deep learning inference capabilities across the entire Intel silicon portfolio, and meet their AI needs by using unparalleled solutions.

OpenVINO include three major APIs which are:

1. The Deep Learning Deployment toolkit
2. A common cross Intel Vision platform deep learning inference toolkit
3. OpenCV and OpenVX's optimized functions.

With the addition of OpenVINO to the Intel Vision Product portfolio, along with other AI products such as nGraph, Intel distribute AI solutions from the edge to the network to the cloud across their broadest set products. Because the OpenVINO is compatible with widely adopted AI frameworks, developers can seamlessly and effortlessly deploy their applications at the edge.

Human face can be acquired and send to the face features to pc for training purpose and will be trained through more layer convolutional neural network. However, face reidentification is not smart enough. In order to avoid repetitive training of face recognition models, we should teach the system how to differentiate the person, not just require large number of images to train and recognize a limited number of people. One shot learning will be used to train a model to calculate the cosine distance between two or more faces, so a truly smart face reidentification can be performed without iterate training. After trained, OpenVINO will be used to optimize the pretrained model and finally the pretrained model will able to faster, accurately and smartly re-identify human face.

## 1.2 Project Scope

In the end of project, a system which using OpenVINO technology to optimised the machine learning model in the area of face reidentification will be developed in order to amend the insufficient in the current methods. Factory visitors are allowed to register through a camera. Camera on the face registration system will capture the image and save the face as a reference. After that, the face will be used as a reference to differentiate with the real-time capturing face image by face reidentification system. The face reidentification system will use three pretrained model to perform the face reidentification process. These models will be “face detection model”, “landmark detection model” and “face reidentification mode”. As the result, these pretrained model will be optimized and make the reidentification process faster and able to run on embedded board. This system will differentiate the human features and smartly recognize who is he/she and instant show the name and location of the person on screen. If there is an unauthorized person came inside the factory, the “Unauthorized” will be show beside the frame and other developer may improve this system by invoke the notification call to notify admin.

### **1.3 Project Objectives**

In this project, three main objectives will guide the direction of whole project.

Objectives:

1. Design a face registration system to detect and crop out visitors' face and save it as a reference in face databases.
2. Design a face reidentification system that calculate and compare the cosine distance with built face databases which real-time identify, monitoring and tracking the factory visitors by using the face detection, landmark detection and reidentification model in any corner of the factory.
3. Design a secure connection between face registration systems and face reidentification system so that face registration system is able to transmit the reference images to all face reidentification systems.

## **1.4 Impact, significance and contribution**

Most of SME can get benefit from this proposed face reidentification system by using OpenVINO due to it help SME to track the visitors and employees inside the factory. By having this face reidentification system, SME may have better privacy which can make sure the safety of factory especially unmanned factory.

In this era which speed is the most concerned factor, success of system will depend on speed and accuracy. If inference of neural network not fast enough, unregistered faces were not recognized in time, the safety and integrity of factory will no longer exist. Company trade secrets could leak to some unauthorized people who will sell the company trade secrets to company competitors. However, we cannot give up accuracy and demand speed. Speed and accuracy are directly proportional in same condition of neural network, we can only improve the speed by apply new technology without lack of accuracy. Hence, this project will effectively improve the speed and accuracy of detect face and reidentify it which differ from traditional neural network without use of OpenVINO.

## 1.5 Background information

In this highly technology era, Internet application development demand is become higher. Internet of Things (IoT) is a major technology by which can produce various useful internet applications. Every devices and systems that human use every day in the analogue world have been added digital sensors and networking which is the product of IoT. Basically, IoT is a network in which all physical objects are connected to the Internet through network devices or routers and exchange data. The Internet of Things allows remote control of objects in existing network infrastructures, just as humans can use their phones to remotely unlock doors.

The Internet of Things is a very good intelligent technology, which can reduce manual work and easily access physical devices. The technology also has an autonomous control function, through which any device can be controlled without any manual intervention. For example, human do not need to adjust the temperature of refrigerator and the refrigerator will smartly adjust the temperature to ensure that foods will not easily decay. Nest and Ecobee smart thermostats are some of the most famous consumer examples. Smart thermostats have sensors in multiple rooms that can be connected to mobile phones via Internet, allowing you to extend temperature control. Smart thermostats can also be connected to some coded bot that help to control the temperature when you are away or depend on the weather. Not only that, Smart thermostats can even detect you are not home or leave and help you to adjust the temperature smartly. Industry 4.0, next generation of manufacturing will be the next target of IoT.

The OpenVINO toolkit enables developers to build and train AI models on cloud -- on popular frameworks such as TensorFlow, MXNet, and Caffe -- and deploy them to a board range cross platform product. (Mike Wheatley,2018) It takes advantage of Intel's investments in different AI accelerator technologies, including CPUs, field programmable gate arrays (FPGAs), and Movidius vision processing units (VPUs). For example, developers working for retail companies can use the toolkit to deploy computer vision capabilities in a series of edge applications such as security cameras, or digital signage.

OpenVINO provides a set of optimization capabilities and a runtime engine that allows developers to run their model on the architecture that best suits their needs, whether it's a highly-tuned FPGA, an efficient VPU, or another choice. For instance, in a retail setting, developers may wish to deploy computer vision capabilities in a range of edge applications, including point-of-sale, digital signage, and security cameras.

The traditional way of inferencing of TensorFlow is loading a model, followed by transforming data, run inference and interpreting output. After apply the OpenVINO to the system, vision apps for Intel hardware will be optimized by OpenVINO. Real-time object detection with OpenVINO will sees a significant speedup due to the acceleration library for optimized computing with Intel's hardware portfolio.

Industry 4.0 is a name for the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of Things (IoT), cloud computing and cognitive computing. Boston Consulting Group state that: Industry 4.0 is made up by Autonomous Robots, Simulation, Horizontal and Vertical System Integration, The Industrial Internet of Things, Cybersecurity, The Cloud, Additive Manufacturing, Data and Analytics and Augmented Reality nine principal technologies.

Smart factory represents a leap forward from more traditional automation to a fully connected and flexible system, which can learn and adapt to new demands through a constant stream of data from connected operations and production systems. Smart factory provides real-time operational awareness, flexible control and data-driven insights that enable smarter decisions for optimal process execution across the company. A true smart factory can drive manufacturing, maintenance, inventory tracking, digitization of operations, and other types of activities throughout the entire manufacturing network by integrate data from system-wide physical, operational, and human assets. The result may be more efficient, more flexible systems, less production downtime, and greater predictive power and the ability to adapt to facilities or wider network changes, which may lead to better positioning in a highly competitive market.



## Industry 4.0

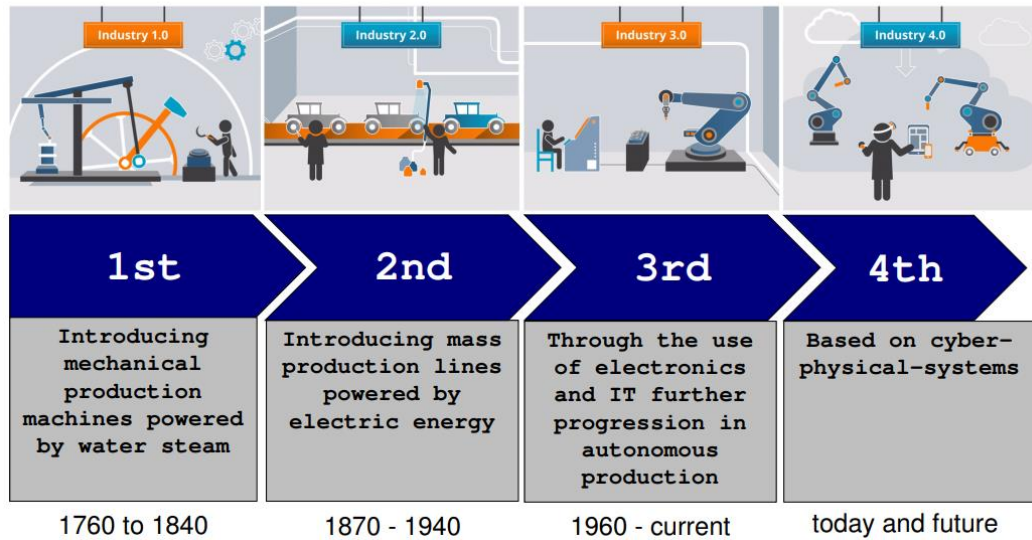


Figure 1.4: Industry 4.0



Figure 1.5: Smart Factory

## Chapter 2: Literature Review

### 2.1 ROS\_OpenVINO

A researcher Giovanni di Dio Bruno proposed a ROS package that used to wrap OpenVINO inference engine which worked with intel CPU, intel integrated GPU and intel VPU. (Giovanni di Dio Bruno,2019) He finally accelerates robotics developing using ROS with OpenVINO. The async API of Intel OpenVINO allows the robot to use a standard image topic and obtain a new topic with live result image. Depth analysis can be performed by adding a depth image topic to obtain a 3d box (displayed as a marker in rviz). The whole system tests are done UP2 with Intel Atom, 4GB RAM and 64GB eMMC and AI Core X. The result with a complete analysis with 4-5 object detected is about 30fps for rgb output and 16-18Hz for spatial calculation.

The one of strengths of his ROS system with OpenVINO is whole system is work on Intel UP square board including object detection, computation and analysis. He took out the computation and analysis function from pc and perform it on board which may make the whole system more portable and the system may be run everywhere but not only some fixed places. Other than that, the system is still able to detect the person even though the bottle is larger than person due to some shooting angle problem. Giovanni Bruno use a human model toy to test his system by placing a huge bottle beside the model and the system is still able to extract the correct features and smartly identify these entities.

Unfortunately, even though the system looks like perfect, there are still some issues exist which are: resize of input image is not available, it is still unable to track human face and the speed of analysis is still slow. If there is an image with 45 angles is input, the features of image will not be extracted properly and resulting in error detection and analysis. In simply say, the system is unable to analysis the image for different angle image shooting. Besides that, this system has the potential to track for human face but the researcher didn't do something like face recognition on it. Hence, he did not fully utilize the system and further push the system to become more effective. Finally, the speed of analysis is still very slow. The whole system is done on intel up square board, the central processing unit of up square board is still unable to handle some difficult algorithm computation, it had better to send the extracted features information over ROS to a high-end computer for computing purpose.

## 2.2 ROS2\_OpenVINO

A Chinese developer Lewis Liu and his teammate proposed a system that based on OpenVINO and a ROS-adapted runtime framework of neural network which quickly deploys applications and solutions for vision inference. (LewisLiew,2018) They divide the package into different functional components as shown in Figure 2.1.

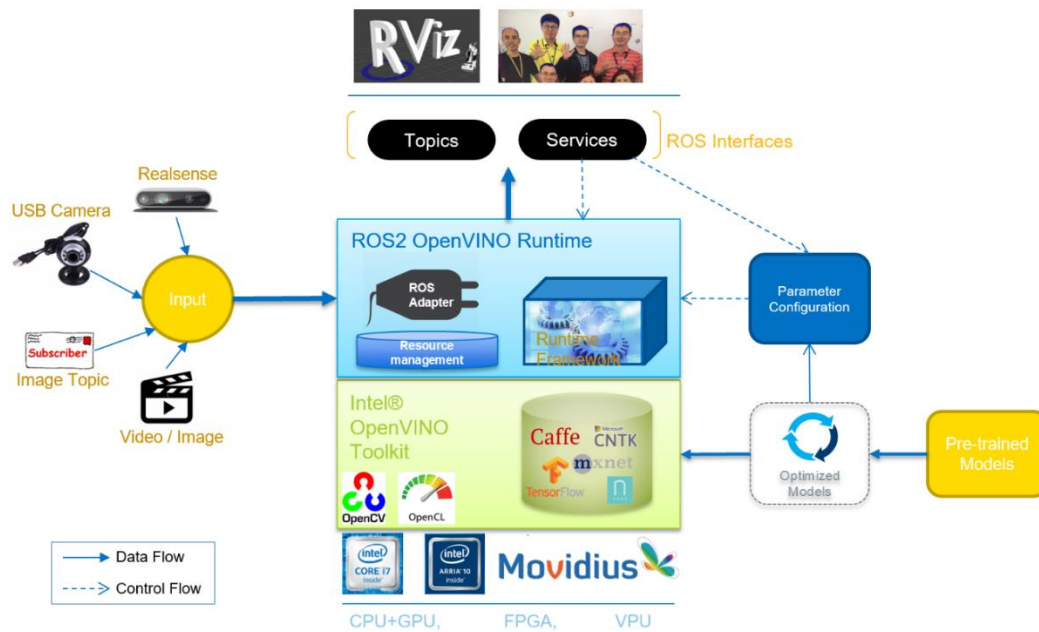


Figure 2.1: Hierarchical Architecture Design

This system is closely to perfect, which has speed, accuracy and functionality. The most significant strength of this system is able to perform multiple function like segmentation, object detection, person reidentification and face detection. This has the potential to further identify the object and filter it to different subsidiary inference. For example, given a car image, firstly do object detection on it, secondly pass car to vehicle brand recognition and pass license plate to license number recognition. The different structure outcome of the system will mainly depend on user requirement.

Unfortunately, the system is still unable to support result filtering for inference process which mean that inference results still cannot be filtered to different subsidiary inference. Besides that, emotion detection is only able to detect happy, neutral or sad. Recently, NASA Glenn Research Centre has released that webcam-pulse-detector, maybe the emotion detection can be combined together with pulse detector and further analysis on it to differentiate from more other kind of emotion like angry, idle, tired et al.

## 2.3 OpenFace

Researcher Carnegie Mellon University (Brandon Amos) proposed a project, the project is to use deep neural network for face recognition Python and Torch implementation, and is based on FaceNet Paper: "A Unified Embedding for Face Recognition and Clustering" (Brandon Amos, 2016) Torch allows CPU and CUDA to execute the network. Face detector dlib is used with pretrained model. Second, they transform faces for neural networks purpose. The repository tries to make the eyes and lower lip appear at the same position in each image by using dlib's real-time pose estimation and transformation. Third, they use deep neural networks to represent faces on a 128-dimensional unit hypersphere. Embedding is a common representation of anyone's face which has a very good attribute. Larger distance between two face embeddings means that these people may not be the same person. This attribute makes clustering, similarity detection, and classification tasks easier than other face recognition techniques. In other face recognition techniques, the Euclidean distance between features is meaningless. Finally, they applied their favourite clustering or classification techniques to the features to complete the recognition task.

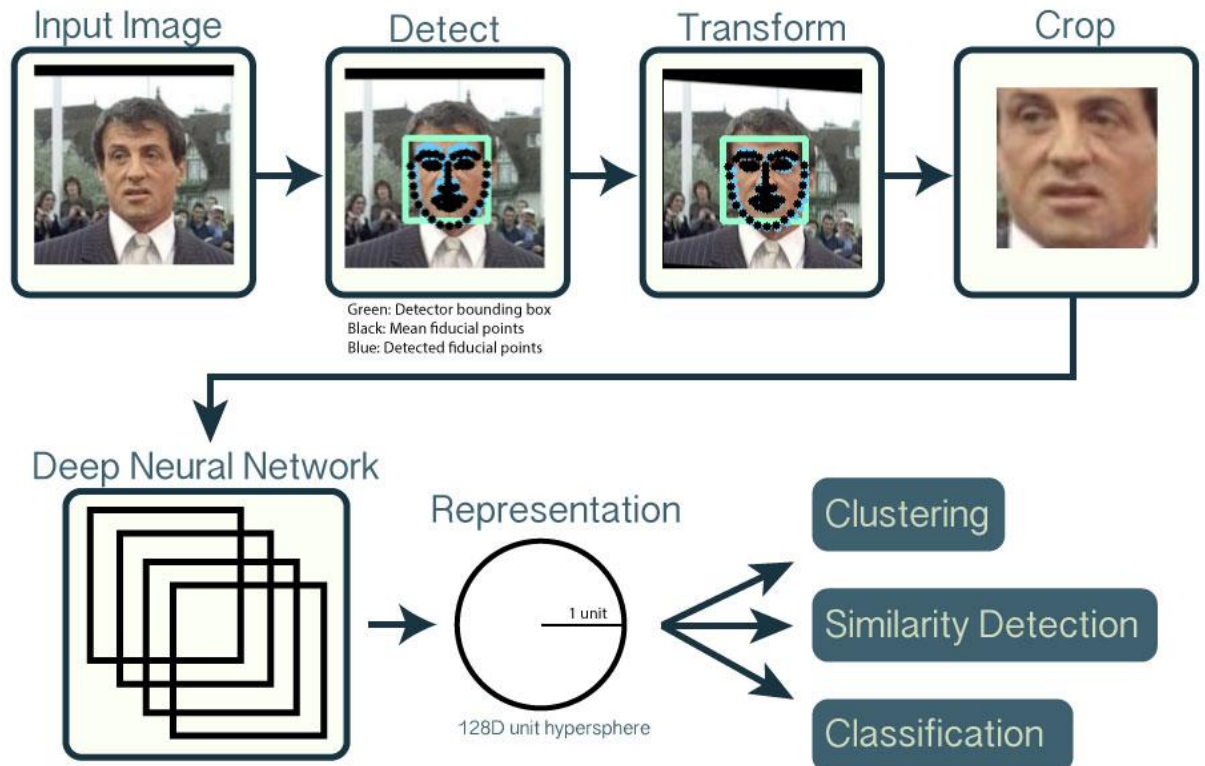


Figure 2.2: Flow of OpenFace classification techniques

The strength of this project is the accuracy of face recognition is very high. The accuracy test is work on the standard Labeled Faces in the Wild (LBW) benchmark and they get a remarkably high result on the benchmark. The ratio of true positive rate over false positive rate is higher than other method system like system with eigenfaces and OpenBR and only lower than human intelligence. This may highly increase the security level. Below is the result of OpenFace. Apart from that, another strength of this project is able to train a face model very face and differentiate the person instantly.

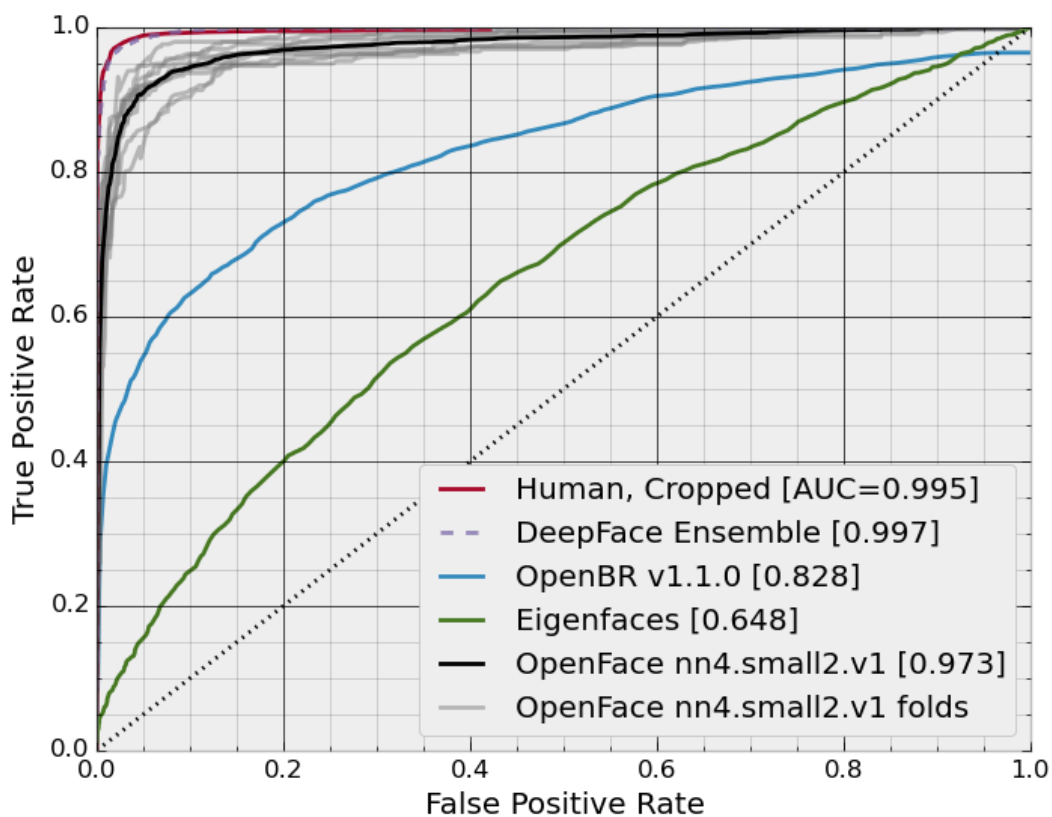


Figure 2.3: ROC Curves

Although the speed of training model is fast, but there is still not enough. The speed of training model is important because when the model is growing larger, the prediction time will become longer. This may lead to some resources and money wasting problem. This problem can be improved by applying the Intel OpenVINO to help to predict faster and more precise.

## 2.4 Person-Reidentification

This project is used to detect and track people in the video and list all the unique people appearing in the video. If someone appears some frames, then leaves the video for a period of time, and then appears again later, the model should recognize that it is the same person, rather than counting the person a second time. The project combines two methods: tracking and reidentification.

The advantage of this project is that it can learn features and corresponding similarity measures at the same time for reidentification of personnel. Project proposes a deep convolution architecture that contains layers specifically designed to solve the reidentification problem. Given a pair of images as input, our network will output a similarity value indicating whether the two input images represent the same person.

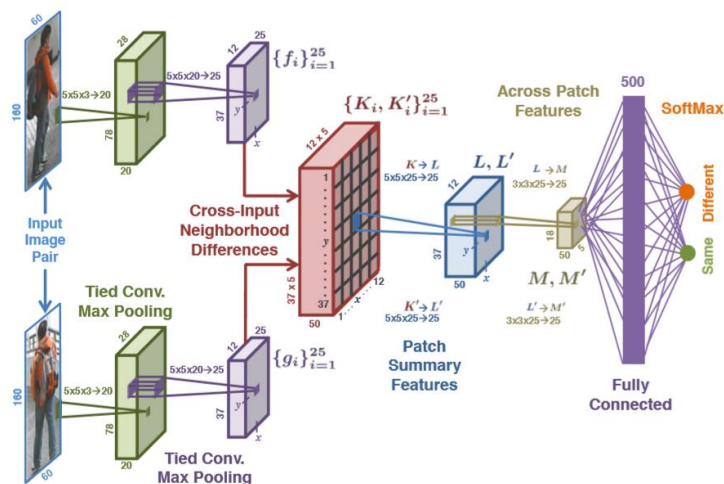


Figure 2.4: Architecture about improved reidentification

The biggest problem in this task or any tracking or reidentification task is dealing with occlusion. When there are many occlusions, the tracking mechanism still fails: when people are very close to each other and overlap each other, or when people cross each other and thus temporarily overlap. Therefore, occlusion processing is one of the areas that need improvement.

The use of tracking mechanisms can definitely make the execution faster, but it is not certain whether this is fast enough for real-time personnel tracking and reidentification. Therefore, the execution speed is also an area that needs to be improved.



## 2.5 MTCNN4Android

A researcher from Xiamen University Computer Science has proposed the android based implementation of MTCNN face detection. Author use the pretrained model in Google's FaceNet project and freeze the graph with .npy files into .pb file. The whole project is coded on Java based platform. System is able to detect the faces in images on mobile phone platform and able to figure it out and find out all the landmark on face.

According to the result, MTCNN has superior power in accuracy and it is capable of recognizing other facial features such as eyes and mouth. MTCNN always include facial detail which is important to be used in a classification network. Other than that, MTCNN is allow developer to be invoked cross platform. This show that MTCNN has excellent portability which not only allow user to invoke the application programming interface in different development environment, but also allow lower specification devices like mobile phone to use this framework.

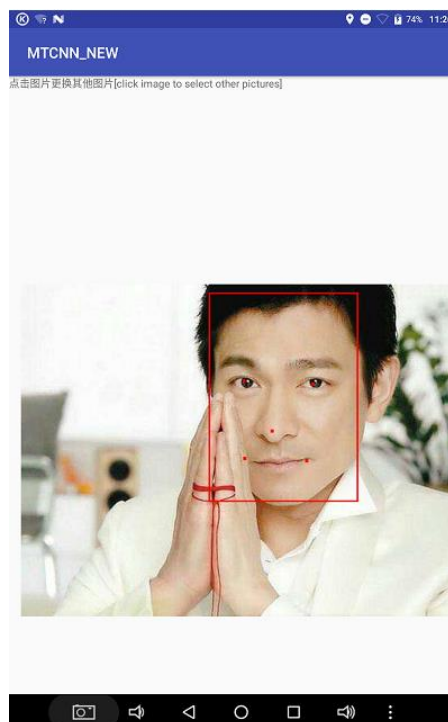


Figure 2.5: Result of MTCNN4Android

The weakness of this project could be the lack of speed. MTCNN is much slower than other face detector such as dlib and Haarcascade. Report state that Haarcascade has extremely fast processing speed which is 25 images per seconds, while it only 3 images per second for MTCNN. Author proposed some techniques to resolve this problem which adjust the minimum face detection size and satisfied some small faces. This could be double-edged sword, this method is workable on register human face into system but not on real-time detection of human face.



## 2.6 OpenVINO Smart Classroom C++ Demo

Precise tracking of person in a video is a common application of Computer Vision. Intel has proposed OpenVINO and came out with some demo to guide the software developer to develop system using OpenVINO. This project is a demo to show a program to join several neural networks and detect student actions and recognize people by faces in the classroom environment. Project demo use Async API for motion and face detection which allow face detection, person detection, person recognition and action recognition to be done in parallel on separate model accelerator.

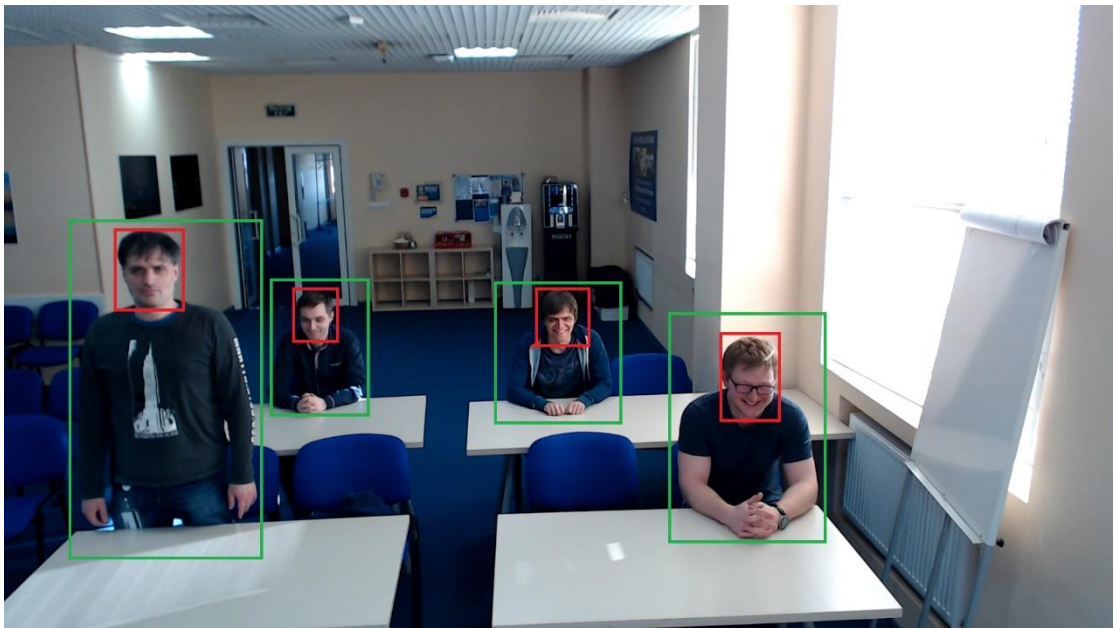


Figure 2.6: OpenVINO Smart Classroom Demo

One of the advantages of this project is speed is greatly increased by using Single Shot Detector (named SSD, a single deep neural network). SSD is a architecture like YOLO takes only one shot to detect multiple object present in an image using multibox which is a branch of one shot learning, it well trained to become an observer, not a discoverer. Moreover, these models are using OpenVINO model optimizer to break the model's graph representation to intermediate representation form (\*.xml + \*.bin, xml file will carry the architecture of the model and bin file will carry the bias and weight of the models.) and loaded into OpenVINO inference engine to make the inference stage significantly faster in speed and high accuracy face detection algorithm.

## Chapter 3: System Methodology

### 3.1 Design Specifications

In today's technology, there are two ways to complete face recognition. Training a neural network model (preferably a ConvNet model) is one of the methods. Training a neural network model can classify faces accurately but a classifier to be trained well, it needs millions of input data. It is not feasible to collect so many images from employees thereby this method seldom works. One-shot learning technique is another way to perform face recognition. One-shot learning aims to learn information about object categories from one or several training images. The model still needs to be trained on millions of data, but the dataset can be any that lie on same domain. Developers can train models with any dataset and use them to build our own model, and now the number of images from employees is very small.

#### 3.1.1 Hardware and software involved

1. Ubuntu 16.04 Long Term Support (LTS)

In this project, Ubuntu 16.04 LTS will be used due to ubuntu is a free and open-source software operating system that runs from the desktop, to the cloud, to all internet connected things. Which may help us to lower the cost of the whole system and don't need to bother about the license of the operating system. Other than that, LTS version will make sure that development of the operating system will still running and developer's code won't get expired specification so that developer may freely develop on it.



Figure 3.1: Ubuntu 16.04 LTS

## 2. Python3 Programming Language

In this project, Python will be the main programming language to be used to program the whole system. Python will be used to configure the inference engine of OpenVINO and invoke the `ie_core` of inference engine. Other than that, most of the popular library such as `shutil`, `socket` and `OpenCV` is built on python which mean that python will have absolute advantage compare to other language like Java.



Figure 3.2: Python Programming

## 3. Intel OpenVINO 2019R2 toolkit

OpenVINO is a free toolkit that can help optimize deep learning models from the framework and deploy them on Intel hardware using the inference engine. The toolkit has two versions: the OpenVINO toolkit supported by the open source community and the Intel(R) release version of the OpenVINO toolkit supported by Intel. OpenVINO was developed by Intel. The toolkit is cross-platform and can be used free of charge under the Apache License Version 2.0. OpenVINO is build on C++, but they also provide the python wrapper to use the inference engine in python code. In this assignment, python code will be used to invoke the OpenVINO inference engine.



Figure 3.3: Intel OpenVINO

#### 4. 2 x Camera

There will be two cameras in smart factory, one for register visitors' face and another one for servant capture the visitors' face in any location within factory. These cameras will connect to each intel up square board to capture the image.



Figure 3.4: Camera

#### 5. 2 x Intel UP board

The Intel UP board is a computer board for professional manufacturers and industrial applications. The wide range of features included in UP make it an ideal solution for applications such as robotics, drones, machine vision, smart home, education, digital signage, smart cars, and IoT solutions. The embedded board is based on the Intel Atom® processor 1.92GHz X5 Z8350 which compatible with OpenVINO. In this project, two UP board is required, one for running face registration program and other for running face reidentification program.

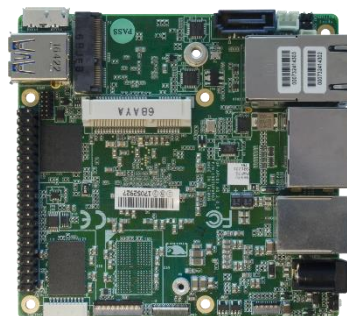


Figure 3.5: Intel UP Board

## 6. OpenCV

OpenCV is a programming library for real-time computer vision. It was originally developed by Intel and later supported by Willow Garage and Itseez. The library is cross-platform and can be used free of charge under an open source BSD license. OpenCV will be used to visualize the camera frame and draw the result on the captured frame.



Figure 3.6: OpenCV

## 7. Python Tkinter

Tkinter is a binding between Python and the Tk GUI toolkit. It is the standard Python interface of the Tk GUI toolkit and the de facto standard GUI of Python. Tkinter is included in the standard Python installations for Linux, Microsoft Windows and Mac OS X. Tkinter is free software released under the Python license.

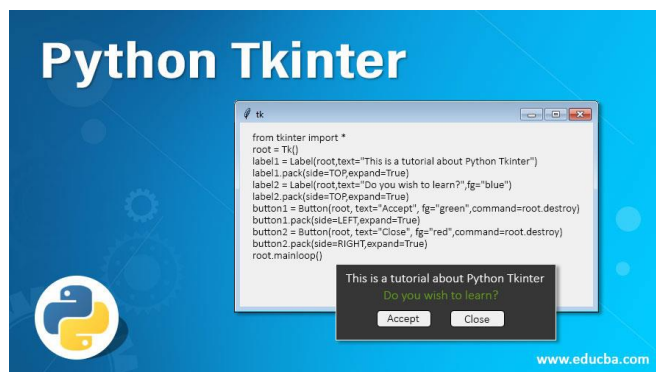


Figure 3.7: Python Tkinter

## 8. Pretrained model

Pretrained model is in intermediate representation form. For using this model, two files will be loaded (\*.bin + \*.xml) to OpenVINO inference engine, so that the inference engine is able to read the model. Developer may easily get these models by using “downloader.py” which included in the OpenVINO installation package. In case you want to use your own model, you may use the “model\_optimizer.py” to break the freeze graph into intermediate representation form, so that OpenVINO is able run on the model. Below is the list of used models in this project.

### a. face-detection-retail-0004 model



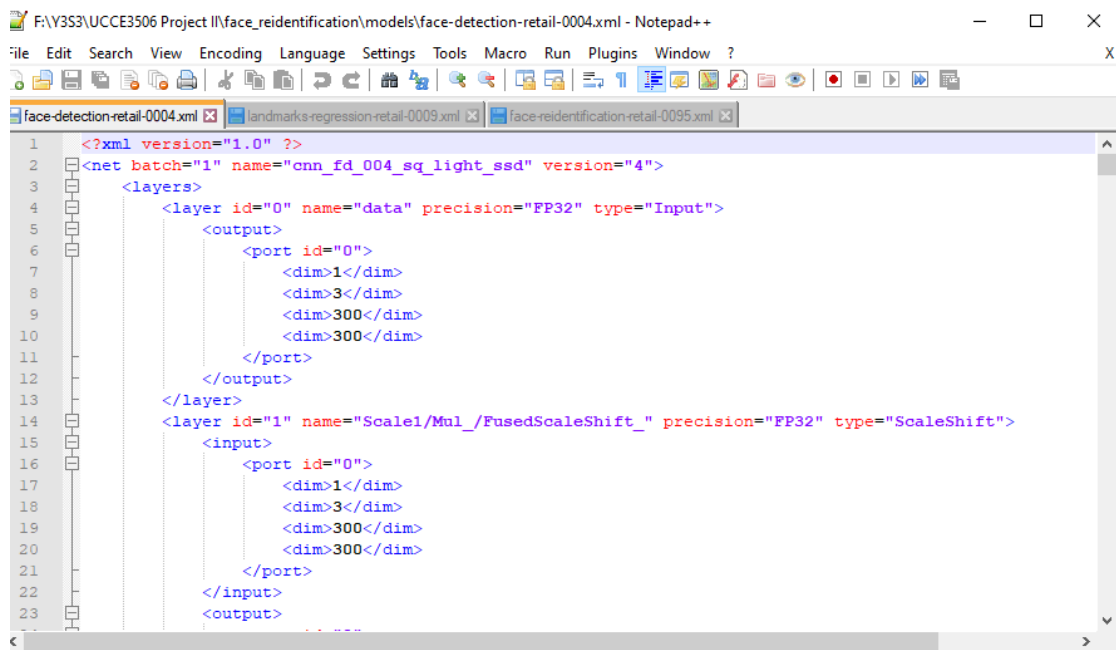
 face-detection-retail-0004.bin	23/8/2020 4:29 PM	BIN File	2,298 KB
 face-detection-retail-0004.xml	23/8/2020 4:29 PM	XML Document	48 KB

Figure 3.8: face-detection-retail-0004 IR files



```
1 <?xml version="1.0" ?>
2 <net batch="1" name="onn_fd_004_sq_light_ssd" version="4">
3   <layers>
4     <layer id="0" name="data" precision="FP32" type="Input">
5       <output>
6         <port id="0">
7           <dim>1</dim>
8           <dim>3</dim>
9           <dim>300</dim>
10          <dim>300</dim>
11        </port>
12      </output>
13    </layer>
14    <layer id="1" name="Scale1/Mul_/FusedScaleShift_" precision="FP32" type="ScaleShift">
15      <input>
16        <port id="0">
17          <dim>1</dim>
18          <dim>3</dim>
19          <dim>300</dim>
20          <dim>300</dim>
21        </port>
22      </input>
23    </output>
```

Figure 3.9: face-detection-retail-0004 network architecture

b. landmarks-regression-retail-0009 pretrained model


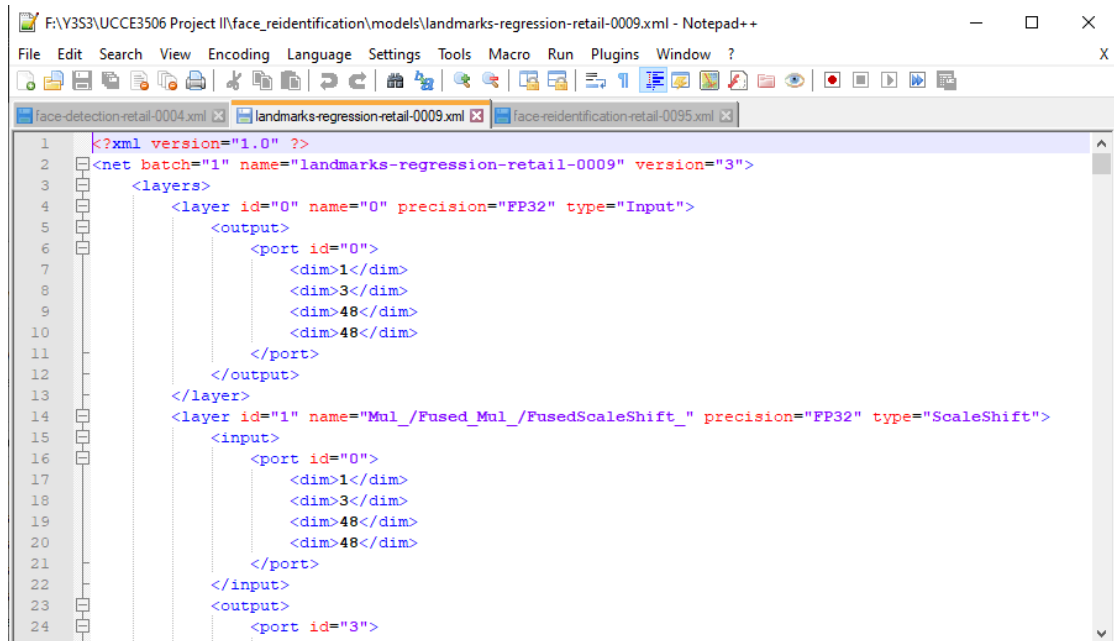
 landmarks-regression-retail-0009.bin	23/8/2020 4:29 PM	BIN File	745 KB
 landmarks-regression-retail-0009.xml	23/8/2020 4:29 PM	XML Document	18 KB

Figure 3.10: landmarks-regression-retail-0009 IR files



```
<?xml version="1.0" ?>
<net batch="1" name="landmarks-regression-retail-0009" version="3">
  <layers>
    <layer id="0" name="0" precision="FP32" type="Input">
      <output>
        <port id="0">
          <dim>1</dim>
          <dim>3</dim>
          <dim>48</dim>
          <dim>48</dim>
        </port>
      </output>
    </layer>
    <layer id="1" name="Mul_/Fused_Mul_/FusedScaleShift_" precision="FP32" type="ScaleShift">
      <input>
        <port id="0">
          <dim>1</dim>
          <dim>3</dim>
          <dim>48</dim>
          <dim>48</dim>
        </port>
      </input>
      <output>
        <port id="3">
```

Figure 3.11: landmarks-regression-retail-0009 network architecture

c. face-reidentification-retail-0095 pretrained model



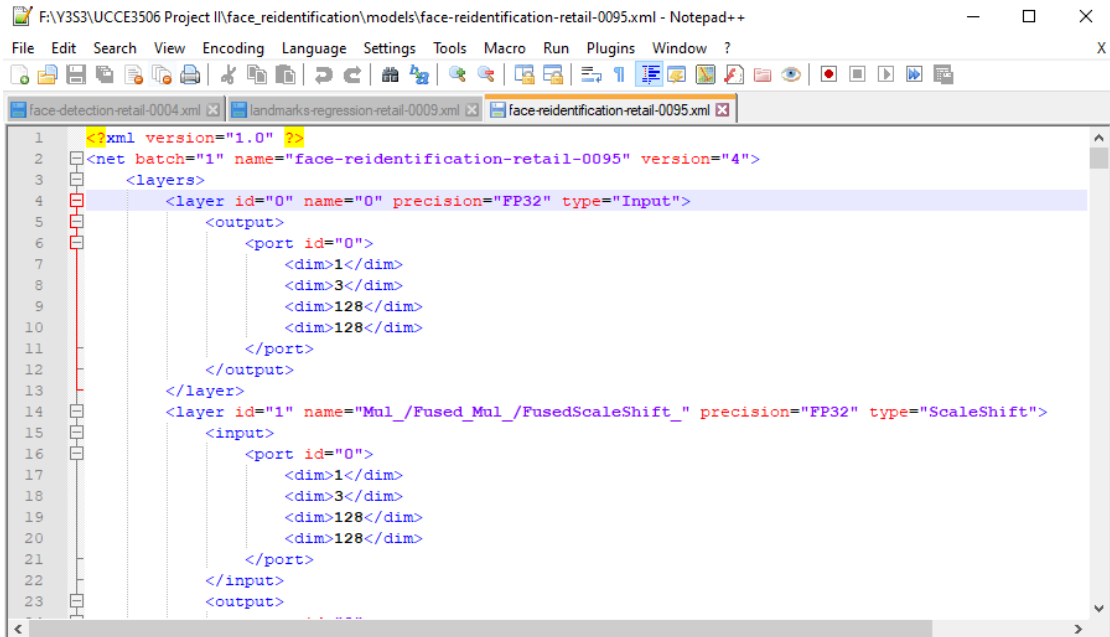
 face-reidentification-retail-0095.bin	23/8/2020 4:29 PM	BIN File	4,324 KB
 face-reidentification-retail-0095.xml	23/8/2020 4:29 PM	XML Document	117 KB

Figure 3.12: face-reidentification-retail-0095 IR



```
1 <?xml version="1.0" ?>
2 <net batch="1" name="face-reidentification-retail-0095" version="4">
3   <layers>
4     <layer id="0" name="0" precision="FP32" type="Input">
5       <output>
6         <port id="0">
7           <dim>1</dim>
8           <dim>3</dim>
9           <dim>128</dim>
10          <dim>128</dim>
11        </port>
12      </output>
13    </layer>
14    <layer id="1" name="Mul_/Fused_Mul_/FusedScaleShift_" precision="FP32" type="ScaleShift">
15      <input>
16        <port id="0">
17          <dim>1</dim>
18          <dim>3</dim>
19          <dim>128</dim>
20          <dim>128</dim>
21        </port>
22      </input>
23      <output>
```

Figure 3.13: face-reidentification-retail-0095 network architecture



## 9. Python Socket Programming

This module provides access to the BSD socket interface. It is available on all modern Unix systems, Windows, MacOS and possibly other platforms. The Python interface is a direct transliteration of Unix system calls and library interfaces. It is used to convert sockets into Python's object-oriented style: the socket function returns a socket object, and its methods implement various socket system calls. The parameter type is higher than the level in the C interface. In this project,

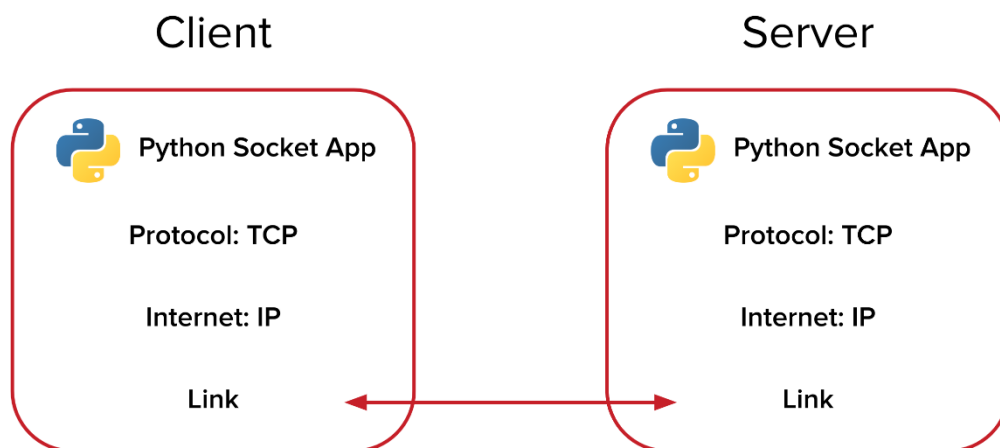


Figure 3.14: Python Socket Programming

### 3.1.2 System Development Method

In this project, Prototyping Model will be used as systems development method in project. (Margaret Rouse,2005) A prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

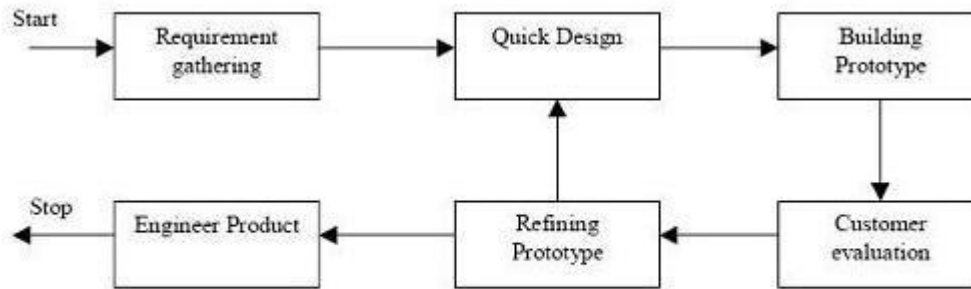


Figure 3.15: Prototype

### 3.1.3 Verification Plan

In this phase, evaluation from the user will be collected and confusion matrix will be implemented to statistically calculate face reidentification system accuracy and precision based on the results that I will test.(GeeksforGeeks,2019) In the field of statistical classification and machine learning, confusion matrix (also known as error matrix) which is a table, usually used to describe the performance of the classification model on a set of test data with known real values. It allows visualizing the performance of algorithms.

Table 3.1 Confusion Matrix

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Positive (P): Positive Observation.

Negative (N): Negative Observation.

True Positive (TP): Positive Observation, and is Positive Prediction.

False Negative (FN): Positive Observation, and is Negative Prediction.

True Negative (TN): Negative Observation, and is Negative Prediction.

False Positive (FP): Negative Observation, and is Positive Prediction.

Confusion Matrix allows easy identification of confusion between classes e.g. one class is commonly mislabelled as the other. Most performance measures are computed from the confusion matrix such as accuracy, recall, precision and f-measure. Below are equations:

$$Accuracy = \frac{TP + TN}{Total\ Cases}$$

Accuracy is defined as the ratio of the total number of cases that is correctly predicted to the total number of all condition examples. In vernacular, it means “How often is the classifier correct”.

$$Recall = \frac{TP}{TP + FN}$$

Recall is defined as the ratio of the total number of positive cases that is correctly classified to the amount of positive cases. High recall mean that system is correctly recognize every class.

$$Precision = \frac{TP}{TP + FP}$$

Precision is defined as the ratio of the total number of correctly classified positive samples divided by the total number of predicted positive samples.

High Recall, low precision means that most positive examples can be correctly identified, but there are many wrong examples. At the same time, "low recall" and "high precision" mean many positive examples, but those that are predicted to be positive are indeed positive.

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

F-measure is always having close distance to Precision or Recall. F-measure helps to make measurements that represent both of them. F-measure uses the harmonic mean which because F-measure penalizes the extreme value more.

### 3.1.4 Expected System Testing and Performance

Table 3.2 Testbench

Test Case	Testing Description	Input	Expected Output
Image Capture	Registration system is correctly getting the name and face image.	A user use registration system to register face.	Directory “gallery” contain registered face image and collected name is created.
Face Detection	1) Load face-detection-retail-0004 model 2)Get possible faces’ position in image.	Any frame that contain human face.	Region of interest will be output as [image_id, label, conf, x_min, y_min, x_max, y_max].
Landmark Detection	1) Load landmarks-regression-retail-0009 model. 2) Get possible landmark’s position in the human face	Human face’s image	Will output five landmarks coordinates which is left eye, right eye, left mouth, right mouth and nose’s position.
Face Reidentification	1) Load face-reidentification-retail-0095 model. 2) Get embedding feature vector of input face image	Human face’s image, region of interest and landmark coordinates	256 face embedding feature value will be ouput
Transportation between Registration System and Reidentification System	1) New start up face reidentification will register IP address at face registration system 2)Registered Face Gallery will send to every face reidentification system	1) New face reidentification will be setup and check it IP in face registration system 2) Gallery folder will be created and send it through TCP socket	1) IP address of face reidentification system will be found in face registration system 2) Gallery folder contain image will be found in face reidentification system after sent

Calculate the Accuracy of Face Detection	<p>4 Scenario will be gone through:</p> <ol style="list-style-type: none"> <li>1) 1 Face</li> <li>2) 2 Face</li> <li>3) 3 Face</li> <li>4) 4 Face</li> </ol> <p>Calculate the Accuracy of the model within 1000 captured frame</p>	<p>1 – 4 face will real-time capture and feed into testing program</p> <p>Testing program will prompt user to enter the number of faces to be detected</p>	Accuracy will display out
Calculate the Speed of Face Detection	Calculate the speed execution of the 4 scenarios of face detection	1 – 4 face will real-time capture and feed into testing program	Speed will display out (in term of fps)
Calculate the accuracy of true positive face reidentification	<p>Go through the same 4 scenarios stated above and calculate the accuracy</p> <p>Calculate the Accuracy of the model within 1000 captured frame</p>	<p>1 – 4 face will real-time capture and feed into testing program</p> <p>Testing program will prompt user to verify the face correctly reidentified</p>	Accuracy will display out
Calculate the speed of true positive face reidentification	Calculate the speed execution of the 4 scenarios of face reidentification	1 – 4 face will real-time capture and feed into testing program	Speed will display out (in term of fps)

### 3.2 System Design / Overview

Below is the whole reidentification system's process flow. Software Developer may easily design out the reidentification system based on the image shown.

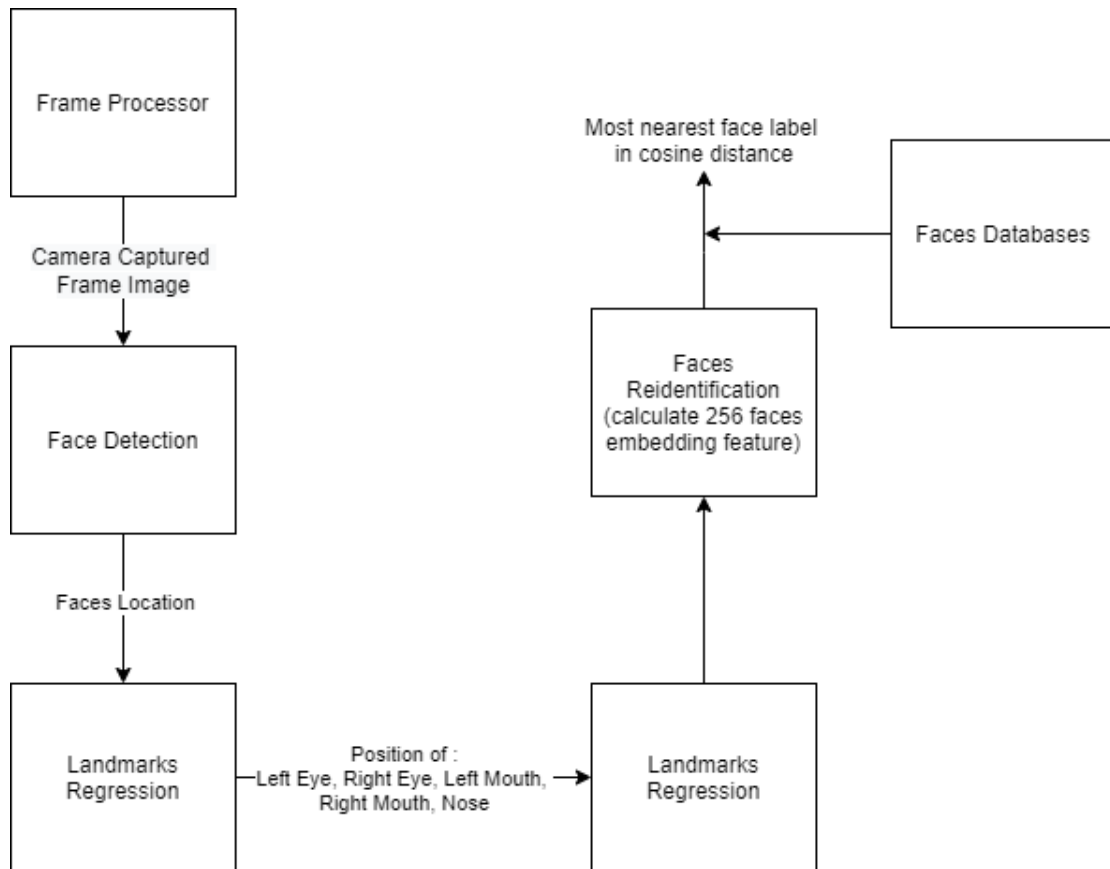


Figure 3.16: Simply Draft on Face Reidentification Process Flow

In this project, three pretrained model will be used which are face-reidentification-retail-0095, face-detection-retail-0004 and landmarks-regression-retail-0009. Intel OpenVINO provided these models in Open Model Zoo which published on OpenVINO official website and OpenVINO official Github site.

### 3.2.1 face-detection-retail-0004

SqueezeNet light is being used as a backbone in this model with a single SSD for indoor or outdoor scene shot. This SqueezeNet has AlexNet-level accuracy and less than 0.5 megabytes model's size which consist fire modules to reduce the amount of calculation. Models' single SSD head from the 1/16 scale feature map has nine cluster priority boxes. This model's source framework is Caffe with 83% of Average Precision (AP) with 1.067 gigaFLOPS ( $10^9$ ). AP is defined as the area under precision/recall curve.

Input of the model should be an image with colour order BGR and has the shape  $1 \times 3 \times 300 \times 300$  in term of  $B \times C \times H \times W$ . B is batch size, C is number of channels, H is image height and W is image width. When we deal with the model, we can found that model output shape is  $1 \times 1 \times N \times 7$  where N is the number of detected bounding boxes. In every detection will have the format image\_id, label, conf, position (x\_min, y\_min) and size (x\_max, y\_max) respectively. For example, outputs[0][0][i] will give the i-th box and each box has 7 numbers which has the format stated above.

Unfortunately, every predicted box should have false positive case which mean that confidence threshold value is needed to be set (Only the bounding boxes above this threshold will be accepted) to make sure the system will not get the unnecessary bounding boxes (BB with no face).



### 3.2.2 landmarks-regression-retail-0009

OpenVINO toolkit provide two type of landmarks regression model which are landmarks-regression-retail-0009 and facial-landmarks-35-adas-0001. The differences between these two models are first model provide 5 points of face and second model provide 35 points on face. For face reidentification, 5 points of face is enough to be used and 5 points will absolute faster than 35 points which may allow embedded board to run the face reidentification faster.

Landmarks-regression-retail-0009 is a lightweight landmarks regressor which has a classic convolution design (3x3 convolution, normalization, activation and merging function). Models predict five facial landmarks which are left eye, right eye, left mouth, right mouth and nose. This model's source framework is PyTorch with mean normed error value is 0.0705 and the gigaFlops is 0.021 which is very good to be used on low computation embedded board. Below is the equation of Normed Error (NE) for i-th sample:

$$\epsilon_i = \frac{1}{N} \sum_{k=0}^{N-1} \frac{\|\hat{p}_{i,k} - \vec{p}_{i,k}\|_2}{d_i}$$

Where N is the number of landmarks, p-hat and p are the prediction and ground truth vector of the k-th landmark of the i-th sample, and di is the eye distance of the i-th sample.

Input of the model should be an image with colour order RGB and has the shape 1x3x48x48 in term of BxCxHxW. B is batch size, C is number of channels, H is image height and W is image width. Model's output shape is 1x10 which contain 10 floating point values for five landmarks coordinates in term of (x0,y0, ... , x5,y5). All coordinates are rescaled into range 0 to 1.

### 3.2.3 face-reidentification-retail-0095

The model is based on MobileNet V2, which uses PReLU instead of ReLU6 activation. The network applies a global depth pool and uses 1x1 convolution to create a 256 face embedding vector. These face embedding vectors are generated for comparison in cosine distance. Similar faces will become closer, while different faces should be far away.

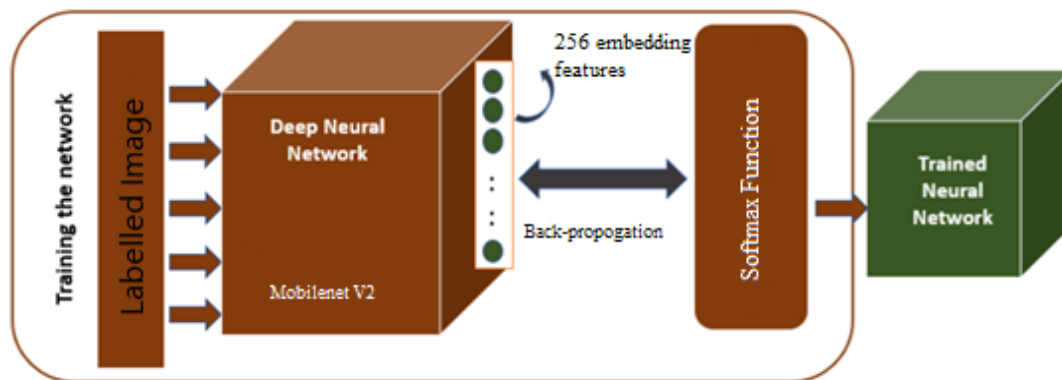


Figure 3.17: Model Structure of face-reidentification-retail-0095. Network consists of a batch input layer and a deep CNN followed by SoftMax function, which results in the 256 face embedding value.

This model's source framework is PyTorch with LFW accuracy 0.9947 and gigaFlops 0.588 which has low computing requirement and suitable for embedded board. If the input face is on the front and aligned, the model result will be best obtained. Input of the model should be an image with colour order BGR and has the shape 1x3x128x128 in term of BxCxHxW. B is batch size, C is number of channels, H is image height and W is image width. The input face image should be tight aligned crop and the five keypoints are located in the following points in range of [0-1,0-1].

```
[ (0.3155687500000000, 0.4615741071428571),  
  (0.6826229166666667, 0.4615741071428571),  
  (0.5002624999999999, 0.6405053571428571),  
  (0.3494718750000000, 0.8246919642857142),  
  (0.6534364583333333, 0.8246919642857142) ]
```

Model's output shape is 1x256x1x1 which contain 256 floating point values Output of the model is comparable in cosine distance.

For Single SSD process, SSD process used multi-box to predict the face which belong to one-stage. SSD is knew as faster and more accurate than Yolo. SSD uses CNN to perform detection directly instead of performing detection after fully connected layers like Yolo. In fact, direct detection of convolution is only one of the differences between SSD and Yolo. There are two important changes. One is that SSD extracts feature maps of different proportions for detection. Large feature maps can be used to detect small objects, and small feature maps can be used to detect large objects. The second is a priori box that uses different ratios and aspect ratios for SSDs. The shortcomings of the Yolo algorithm is that it is difficult to detect small targets and inaccurate positioning, but these important improvements enable SSD to overcome these shortcomings to a certain extent. Below we explain the principle of the SDD algorithm in detail, and finally show how to use TensorFlow to implement the SSD algorithm.

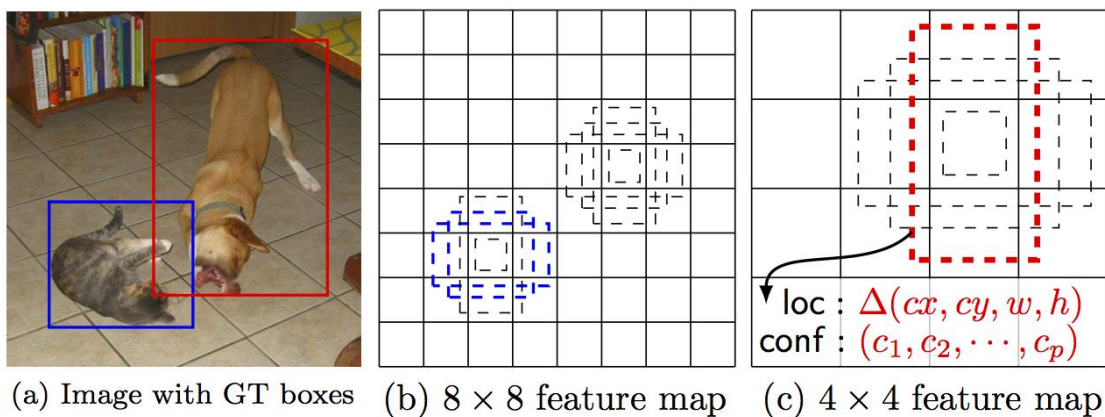


Figure 3.18: SSD default boxes at 8x8 and 4x4 feature map

### 3.3 Methodology

So far, we have understood the various models and architectures. Therefore, we may use these models to design the framework of the face recognition process. Below is a draft of the face reidentification process.

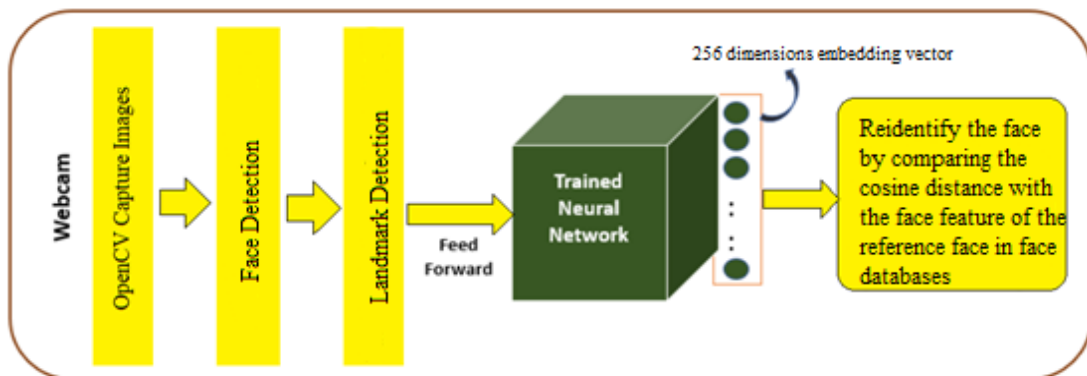


Figure 3.19: Face reidentification using one-shot learning

Face reidentification can be easily perform by comparing facial embedding instead of traditional training of limited face which require absolute less image of registered face than traditional training.

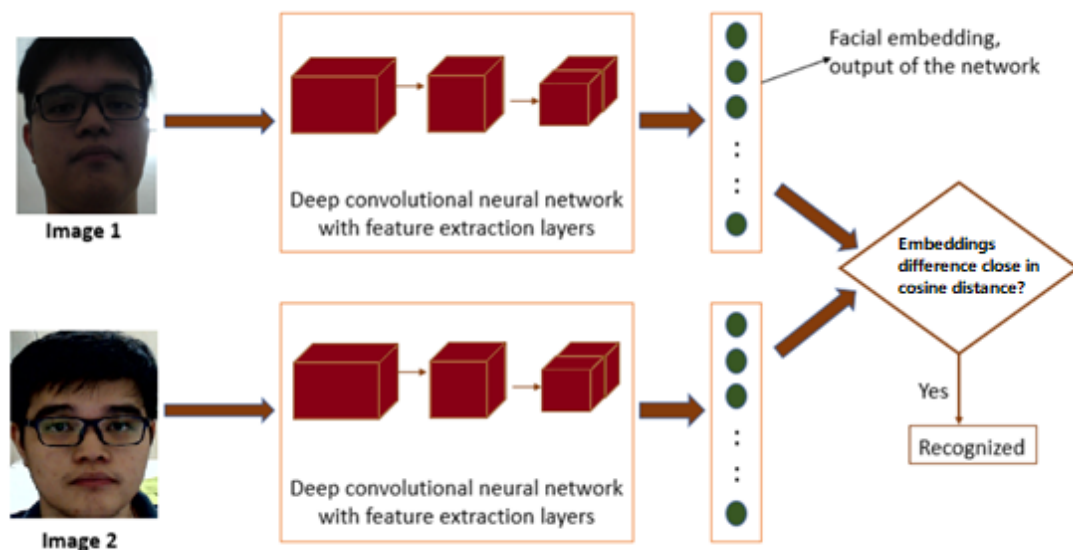


Figure 3.20: Comparing Facial Embedding

### 3.4 Project Milestone

#### 3.4.1 FYP1 Milestone

Table 3.3: FYP1 Milestone

Task	Project Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Revised Proposal														
Define project objective and scope														
Collection of data														
Analysis for Literature Review														
*Report with supervisor current progress														
Determine functional requirements														
Define technologies involved														
Determine system development model														
*Report with supervisor current progress														
Planning the system architecture														
Initializing Environment														
Building the system														
Refining Prototype														
Documentation														
*Report with supervisor current progress														
Presentation of FYP 1														

### 3.4.2 FYP2 Milestone

Table 3.4: FYP2 Milestone

Task	Project Week														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Build Classification System	█	█													
Applied OpenVINO into Classification System			█	█	█	█									
*Report with supervisor current progress							█								
Build the TCP connection between face registration system and face reidentification system							█								
Testing the system							█	█							
Debug the system							█	█	█						
Import Testing Data							█	█	█						
*Report with supervisor current progress									█						
Fixed error of system										█					
Documentation	█	█	█	█	█	█	█	█	█	█	█	█	█	█	
*Report with supervisor current progress												█	█		
Presentation of FYP2														█	

## Chapter 4: System Design and Implementation

### 4.1 System Flow (Software Architecture)

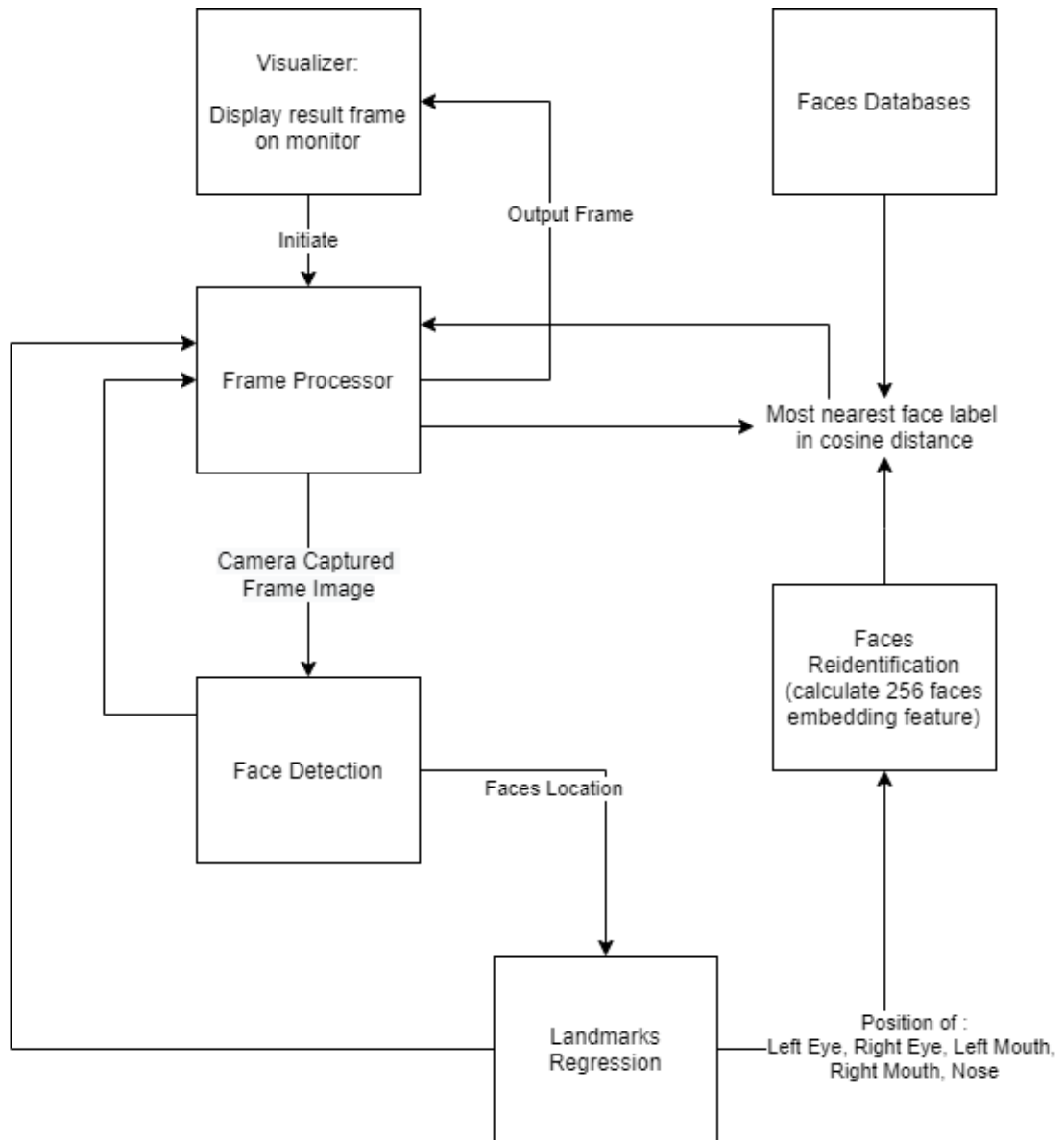


Figure 4.1: System Flow of registration and reidentification system

Both registration and reidentification system will have based on this basic system flow and work on it. Face registration system will have to used model to inference and check exist on Face databases. After that, face registration system will send the registered faces to all face reidentification system and make sure that face identification is able to reidentify registered person and unauthorized person on the spot.

## **4.2 Functional Modules in the Basic Structure**

### **4.2.1 Visualizer**

This module is mainly use OpenCV to create a visualize window which display the detected data that processed by Frame Processor on the captured frame. This module will draw the instruction of system, hardware status, detected region of interest, detected face label, detected landmark on the frame and display it on monitor.

### **4.2.2 Frame Processor**

Frame Processor built the communication between every module and make sure the flow of the reidentification is fluently launch. This module act like a manager which manage every inputs and outputs of Face Detector module, Landmark Detector module, Faces Database module and Face Identifier module.

### **4.2.3 Face Detector**

In this module, every face will be detected by the model. When a detected region has confidence threshold level is less than 0.5 will be eliminated. That mean the detected region is not likely a true face. At the end, a list of high confidence detected region of interest will be generated and pass it to the Frame Processor for landmark detecting.

### **4.2.4 Landmark Detector**

This module is to detect the position of five key points which are left eye, right eye, nose, left mouth, right mouth which to be ready to input into face identifier for generate 256 face feature embedding.



#### **4.2.5 Faces Database**

Faces Database is used to generate a database which contain all the 256 face feature embedding and the label. Face Database generate these data based on the face contained in gallery folder which mean that only registered face will occur in the gallery folder. Otherwise, the person will be mark as unauthorize.

#### **4.2.6 Face Identifier**

This module is used to generate the 256 face feature embedding based on the detected landmark. These values may be compared in cosine distance which is a measure of the similarity between two non-zero vectors in the inner product space. The calculated value will be in range from 0 to 1 and the

### 4.3 System Flow Micro View

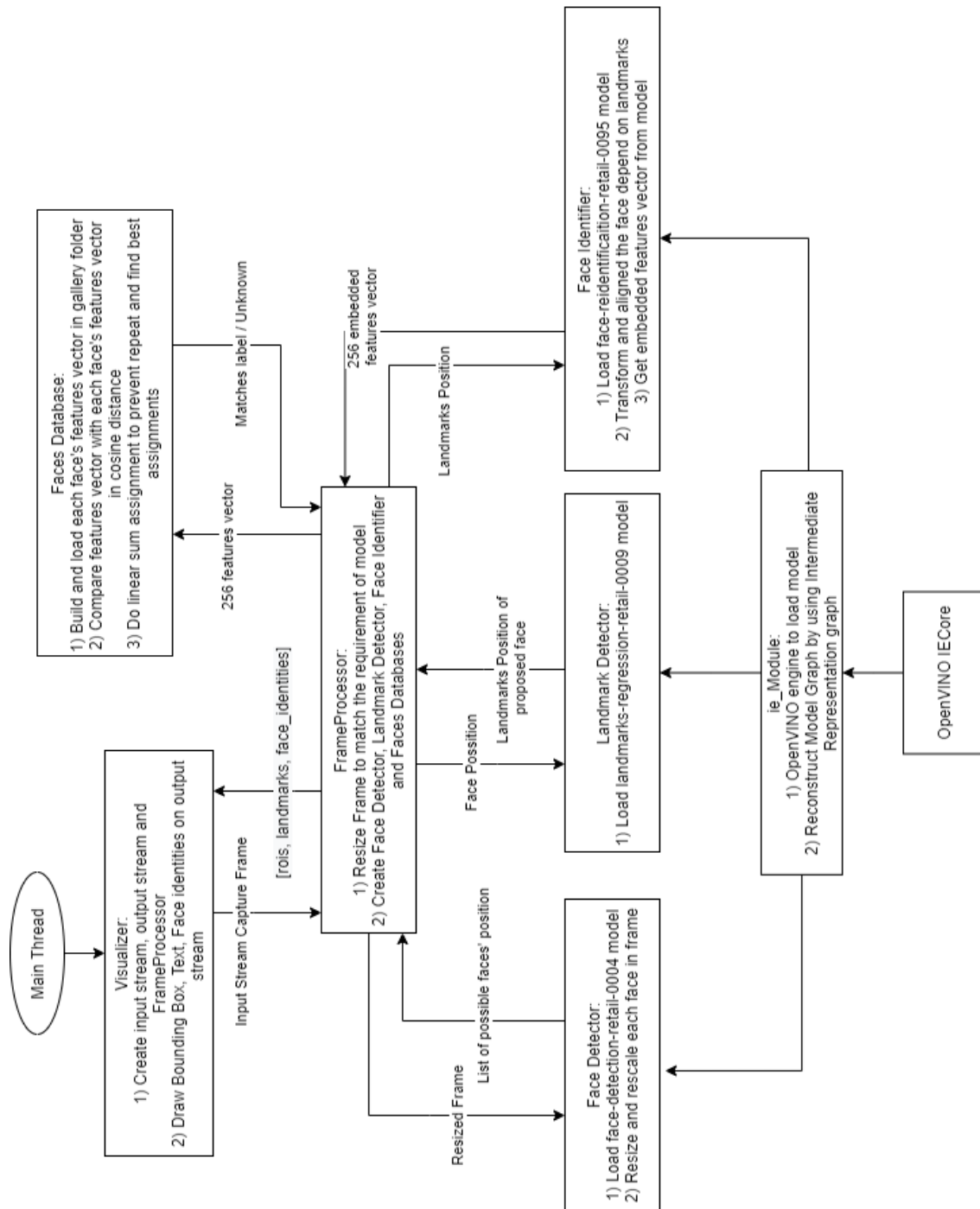


Figure 4.2: Basic System Data Flow

Figure above clearly state out the architecture and interaction between each module which are Visualizer, FrameProcessor, FaceDetector, LandmarkDetector, FaceIdentifier and FacesDatabase. All the input and output data are clearly stated out. The face registration system and face reidentification system will continue work based on this basic structure. The system will go further on this structure.

**Note:**

IENetwork retains the information about the model network read from intermediate representation and allows further modifications to the network. After necessary processing, it feeds the network to IECore, which creates an executable network. Hence, IECore is required during using OpenVINO to run the model network. IENetwork module definition can be found in OpenVINO official GitHub account.

Other than that, some layer of model is still remaining unsupported for CPU. We have to include the CPU extension from the OpenVINO installation folder. The extension named “cpu\_extension\_avx2.dll” is in the inference\_engine folder which include in deplotment\_tools folder. If CPU extension is not included, system might get error notification and force you to include the extension to support some layers of the model.

#### 4.4 Hardware Connection (Hardware Architecture)

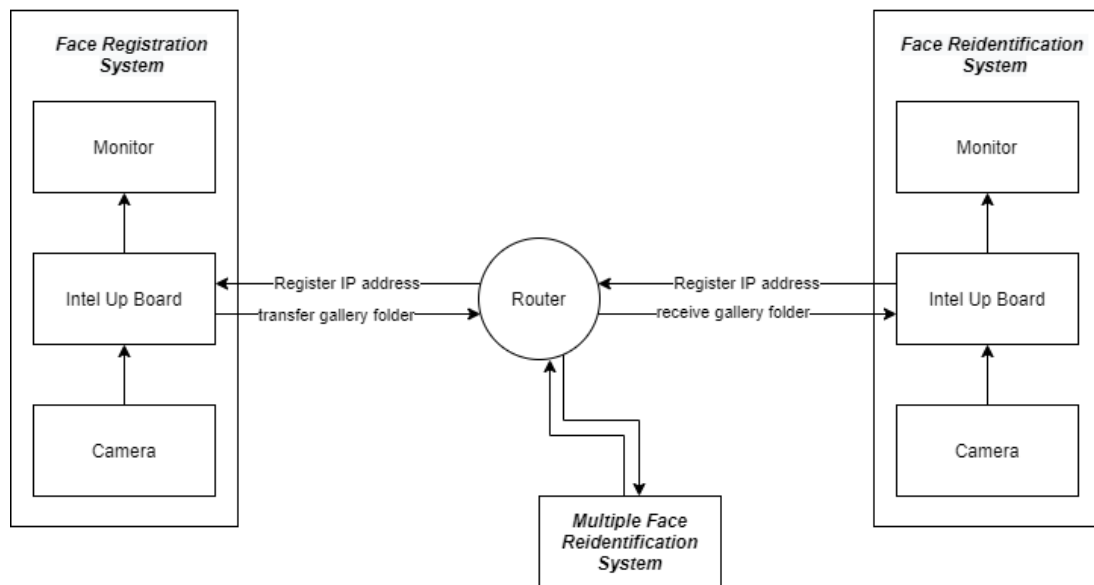


Figure 4.3: Hardware Architecture of system

This is the hardware architecture of the whole system. All intel up board connect to the router and DHCP will automatically assign the IP address to every system. It is recommended to configure static IP for face registration system such as 192.168.0.254 with subnet mask 255.255.255.0. The most recommended way is to configure the static IP table which may arrange these systems easily. Due to mine management's internet connectivity problem, I can't show the static IP table here. Developer may find the tutorial configuration of static IP table through online. The face registration system is act like server and face reidentification system act like client which can be multiple devices.

## 4.5 Flow Chart of Face Registration System

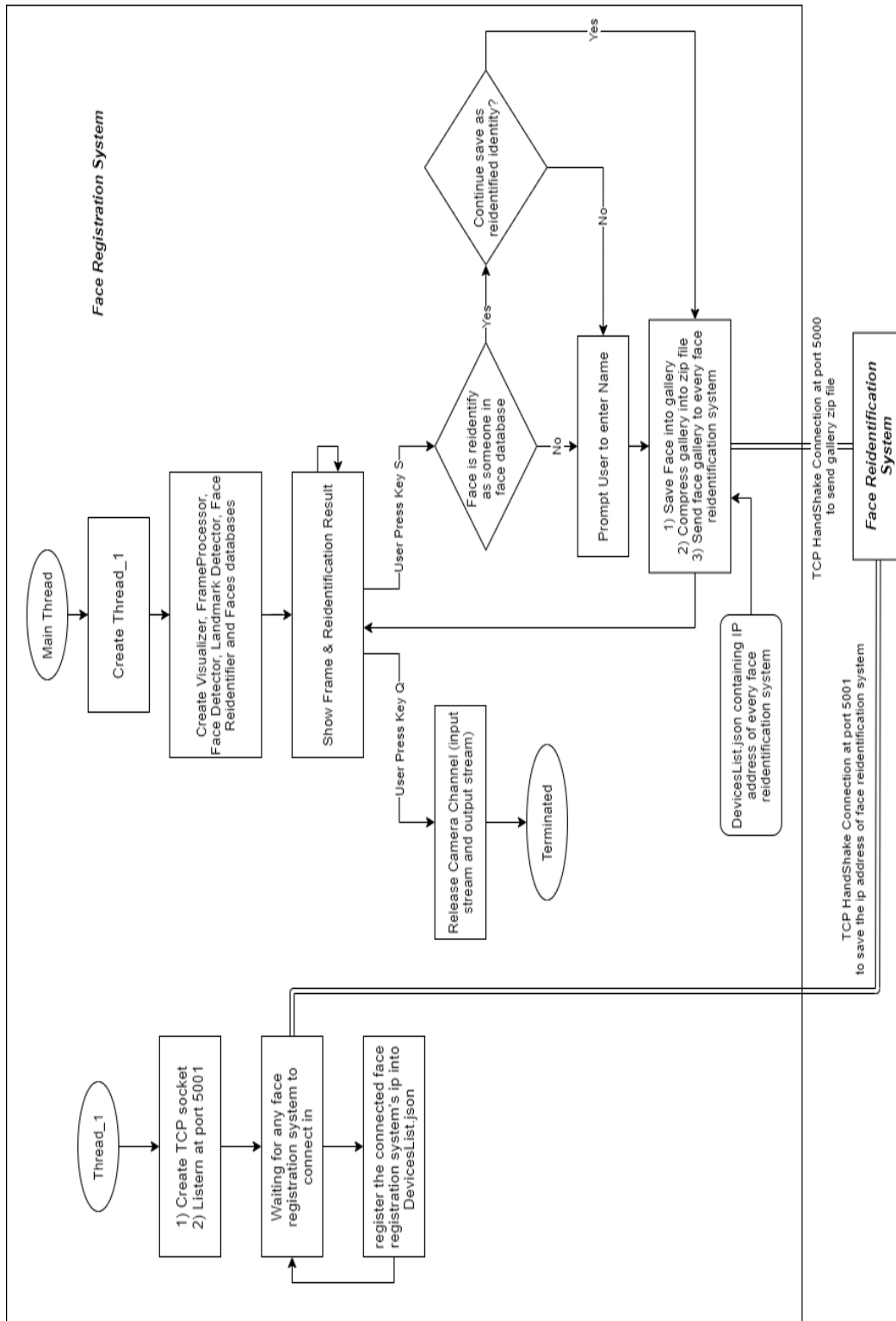


Figure 4.4: Flow Chart of Face Registration System

As mentioned above, face registration system is built on the basic flow of the face reidentification flow. Concern about parallel properties will be required. Hence, two thread is created which are main thread and thread\_1. Main thread is mainly used to manage the visualizer which has the graphical to interact with user. Main thread will always show the result frame and waiting for user to press the key Q and S. Once user press the key Q, the system will release the contained camera input and output stream channel and finally terminate all the thread and close the program. When key S is pressed, the program will flow into save progress which will pop out a window to let user to key in the name of the detected face. This progress will have two scenarios which are: face is registered and face is not registered. When program flow into face registered, program will pop out a message dialog to ask user continue save or save as a new face although it might lead system to confusing. Another scenario will directly prompt user to enter the name. After that, system will compress and send the gallery folder which containing the registered faces to all face reidentification system through TCP connection at port 5000 by reading the IP adderss from the “DevicesList.json” file.

Thread\_1 is used to create the TCP socket and listening at port 5001. Once a connection from face reidentification system is request to connect. System will perform three-way handshake to establish the connection and getting the IP address of the face reidentification system and create a device instance, parse it into json type and finally append it to the “DevicesList.json” file.

## 4.6 Flow Chart of Face Reidentification System

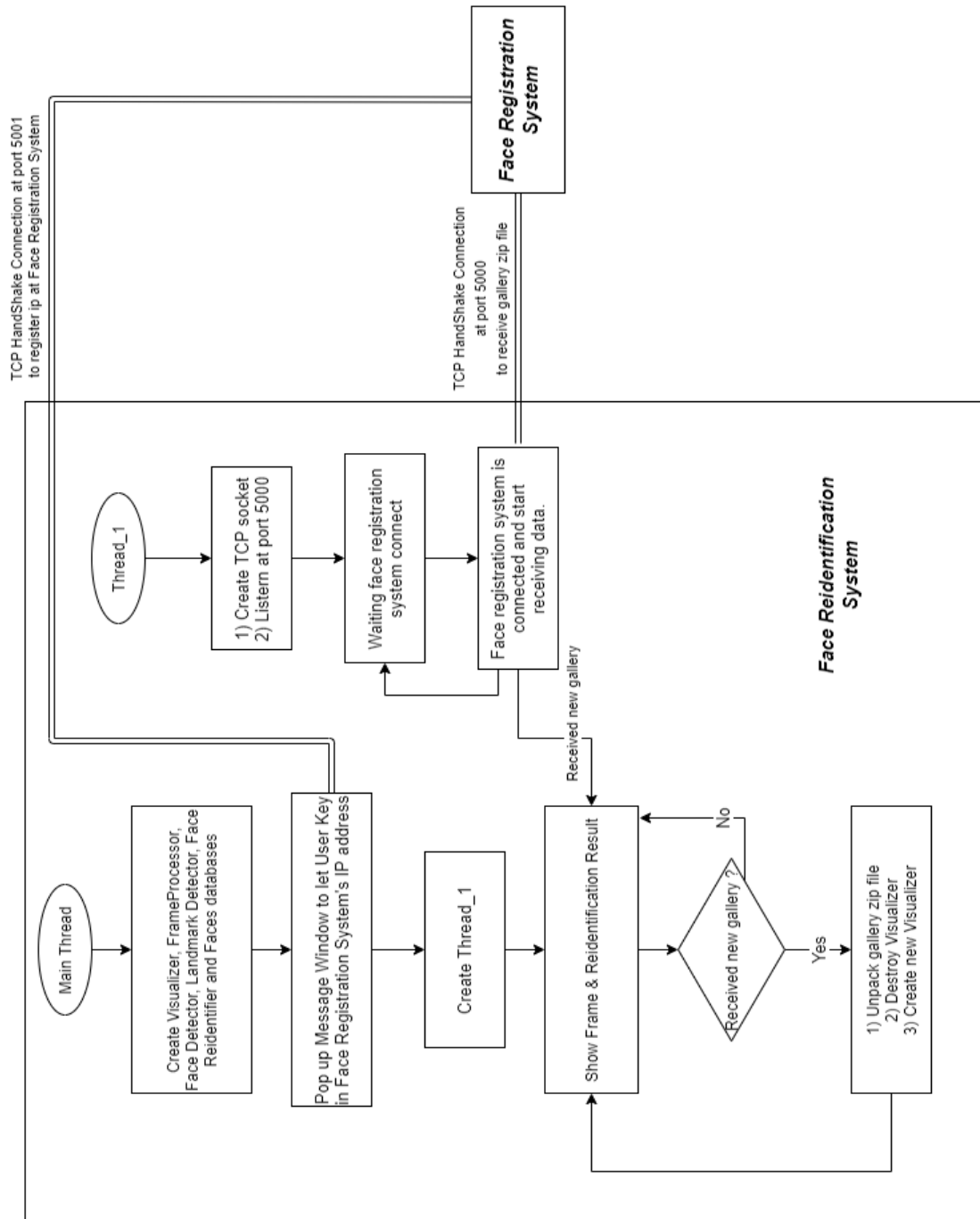


Figure 4.5: Flow Chart of Face Reidentification System

Face reidentification system is some sort like face registration system which is a further version of the basic structure. Face reidentification system will also have two thread to run parallel which are: Main thread and Thread\_1. Main thread is same like face registration system which mainly focus on manage visualizer. The little bit modification is program will register the IP address to the face registration system through TCP connection at port 5001 at the beginning.

Thread\_1 of the face reidentification system will also create a TCP socket and listening at port 5000 to waiting for face registration system to send compressed gallery folder. Once the compressed folder received, program will automatically unpack the archive and replace the existing gallery folder. After that, program will send a flag to main thread to force the main thread close the visualizer for updating the faces database.

#### **4.7 Refining Prototype**

After evaluation, the face reidentification system has to redesign based on the primary output such as the calculated accuracy is low, the speed of getting result is slow or anything else. After the changed, the second output will be evaluated in the same manner as was the primary output. These steps are reiterated persistently, till the users are satisfied with the output.



## 4.8 Final Product

The inference step will be repeated many times until all potential users of the system are satisfied that the prototype represents the desired final product and the final system will only be built based on the final prototype. The final system is fully evaluated and tested. Prevention of large-scale failures and minimize downtime will be carried out on a continuing.

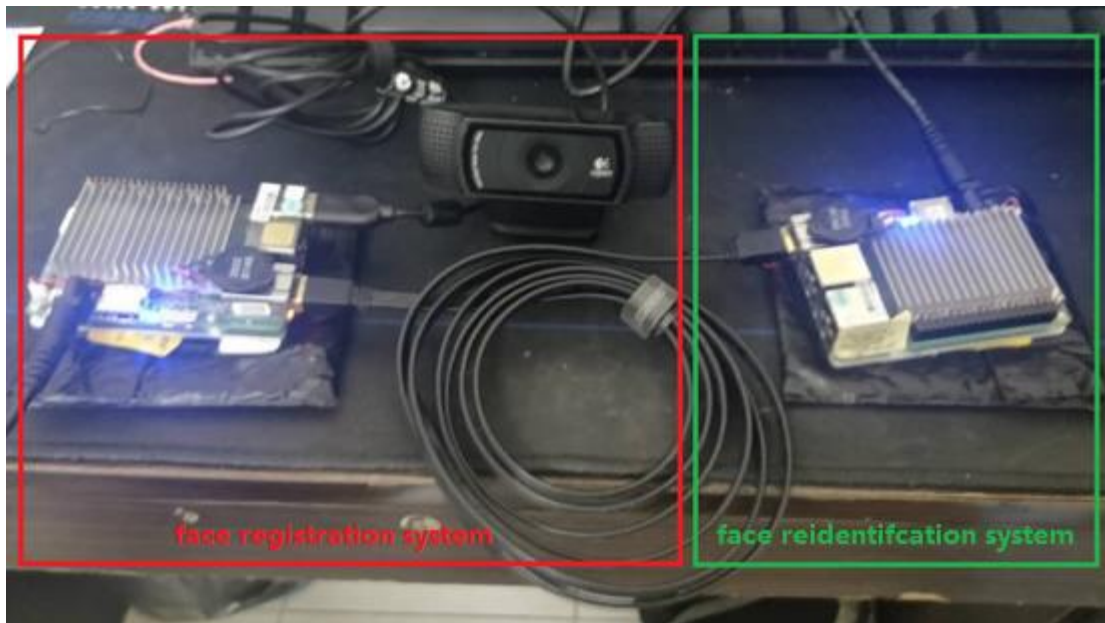


Figure 4.6: Hardware Configuration of Face Registration System and Face Reidentification System

Red frame is the face registration system and green frame indicate face reidentification system. These systems supposed to be connected to router, but due to limitation of resources, point-to-point network is connected through CAT7 Ethernet Cable which has same function as router.

### 4.8.1 Development Environment Setup

First and foremost, installing operating system is the most important step to do. Operating system Ubuntu 16.04 LTS installed as the below figure shown.

```
upboard2@upboard2-UP-CHT01:~$ hostnamectl
  Static hostname: upboard2-UP-CHT01
        Icon name: computer-tablet
        Chassis: tablet
        Machine ID: a35fb2f458ba423db2b06d88a018e39c
        Boot ID: f9d4fe45237d4c968e0f50a5db2a8c9d
  Operating System: Ubuntu 16.04.6 LTS
        Kernel: Linux 4.15.0-112-generic
        Architecture: x86-64
upboard2@upboard2-UP-CHT01:~$ █
```

Figure 4.7: Ubuntu 16.04 LTS Version

After that, python3 should be upgrade and download the necessary module through commands “pip3 install xxx”or “python3 -m pip install xxx”, where xxx is the module name.

```
upboard2@upboard2-UP-CHT01:~$ pip3 --help
Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash            Compute hashes of package archives.
  completion       A helper command used for command completion
  help            Show help for commands.
```

Figure 4.8: pip3 command

Python Tkinter is needed to build up the face registration GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python thereby developer may directly use “import tkinter” to use in python program.

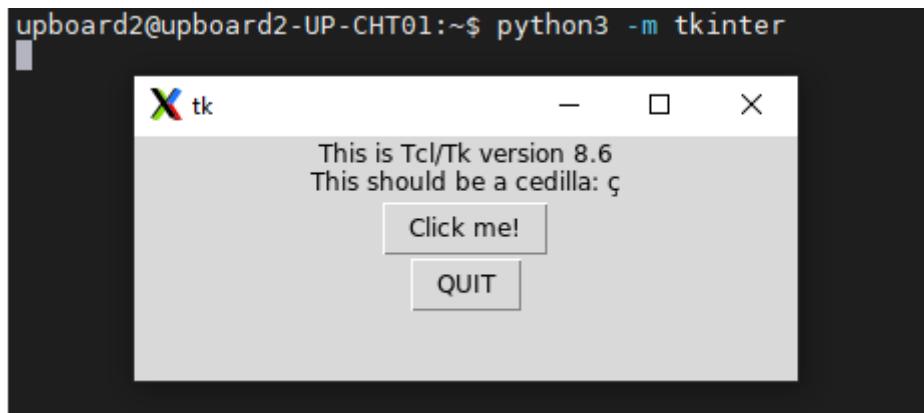


Figure 4.9: Tkinter Version

Besides that, OpenCV is also needed to invoke the driver of camera and use the camera to capture the frame of user’s face.

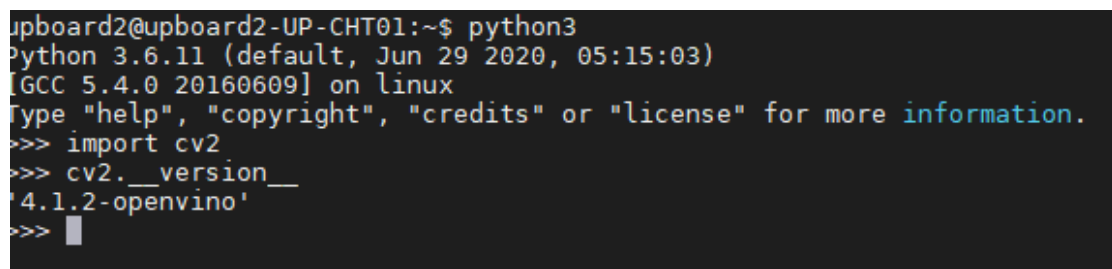


Figure 4.10: OpenCV (OpenVINO)

The needed module can be installed via the requirement.txt that prepared. Command “sudo pip3 install -r requirement.txt” can help to recursively install the correct version of module. The requirement.txt as the below figure shown.

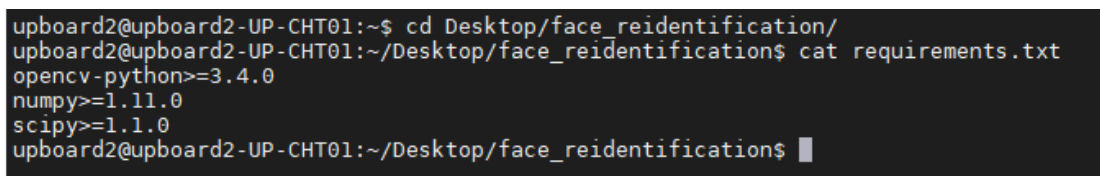


Figure 4.11: Content of requirement.txt

Other than that, installation of OpenVINO is an important way. OpenVINO can be download from the Intel OpenVINO official website. The installation way for ubuntu is different with window. For window user/developer may go through online search for installation way for window.

```

upboard@upboard-UP-CHT01:~$ ll /opt/intel/opencvino
opencvino/          opencvino_2019.3.334/
upboard@upboard-UP-CHT01:~$ ll /opt/intel/opencvino_2019.3.334/
total 44
drwxrwxr-x 11 root root 4096 0gos  6 15:41 ./
drwxrwxr-x  7 root root 4096 0gos  6 15:41 ../
drwxrwxr-x  2 root root 4096 0gos  6 15:39 bin/
drwxrwxr-x  7 root root 4096 0gos  6 15:41 deployment_tools/
drwxrwxr-x  2 root root 4096 Sep  24 2019 documentation/
lrwxrwxrwx  1 root root   33 0gos  6 15:39 inference_engine -> deployment_tools/inference_engine/
drwxrwxr-x  2 root root 4096 0gos  6 15:41 install_dependencies/
drwxrwxr-x  6 root root 4096 0gos  6 15:39 licensing/
drwxrwxr-x  8 root root 4096 0gos  6 15:40 opencv/
drwxr-xr-x  3 root root 4096 Sep  19 2019 opencvino_toolkit_uninstaller/
drwxrwxr-x  5 root root 4096 0gos  6 15:41 opencvx/
drwxrwxr-x  8 root root 4096 0gos  6 15:40 python/
upboard@upboard-UP-CHT01:~$ █

```

Figure 4.12: Folder hierarchy of OpenVINO

User and developer should download and install the needed dependencies of the OpenVINO by “sudo -E ./install\_opencvino\_dependencies.sh” and set the variables (source /opt/intel/opencvino/bin/setupvars.sh) to ~/.bashrc file.

```

Top 10 results:
Image /opt/intel/computer_vision_sdk_fpga_2018.2.298/deployment_tools/demo/./demo/car.png
817 0.8363345 label sports car, sport car
511 0.0946488 label convertible
479 0.0419131 label car wheel
751 0.0091071 label racer, race car, racing car
436 0.0068161 label beach wagon, station wagon, wagon, estate car, beach waggon, station waggon, waggon
656 0.0037564 label minivan
586 0.0025741 label half track
717 0.0016069 label pickup, pickup truck
864 0.0012027 label tow truck, tow car, wrecker
581 0.0005882 label grille, radiator grille

[ INFO ] Execution successful

#####

Demo completed successfully.

```

Figure 4.13: Result of OpenVINO Verification Script

After installation and configuration, make sure the inference pipeline verification script file (./demo\_security\_barrier\_camera.sh) in the folder “/opt/intel/opencvino/development\_tools/demo” is runnable. If not, please check with the Intel OpenVINO official website.

**Note:**

Squeezenet model is not included in the folder. Please download the model file from OpenVINO Github.



Figure 4.14: Successful of installing OpenVINO

If configuration is correct, an image display resulting frame will be displayed. This mean that OpenVINO had successfully installed and all the necessary tools exists and welcome to the OpenVINO world.

## 4.8.2 Face Registration System

```
upboard@upboard-UP-CHT01:~/Desktop/face_registration$ ll
total 80
drwxr-xr-x 5 upboard upboard 4096 Sep  2 16:05 ./
drwxr-xr-x 5 upboard upboard 4096 Ogos 27 16:12 ../
-rw-rw-r-- 1 upboard upboard  15 Ogos 26 14:52 DevicesList.json
-rw-rw-r-- 1 upboard upboard 32158 Ogos 26 16:19 face_registration.py
drwxrwxr-x 2 upboard upboard 4096 Ogos 26 17:38 gallery/
-rw-rw-r-- 1 upboard upboard 1026 Ogos 26 14:12 gallerySend.py
-rw-rw-r-- 1 upboard upboard 4150 Sep  18 2019 ie_module.py
-rw-rw-r-- 1 upboard upboard 2366 Ogos 16 05:39 IP_controller.py
-rw-rw-r-- 1 upboard upboard 1303 Ogos 23 16:16 main.py
drwxr-xr-x 2 upboard upboard 4096 Ogos 27 13:42 models/
drwxrwxr-x 2 upboard upboard 4096 Ogos 26 16:19 __pycache__/
-rw-rw-r-- 1 upboard upboard  48 Sep  2 16:05 requirements.txt
upboard@upboard-UP-CHT01:~/Desktop/face_registration$
```

Figure 4.15: Folder hierarchy of Face Registration System

Figure show that the files contained in face registration system. Program has an entry point which is “main.py”. User or developer may use “python3 main.py” to launch the program. “face\_registration.py” is the program that create visualizer, frame processor, face detector, landmark detector, face reidentifier and faces databases. “gallerySend.py” is the subprogram to send the gallery file to face reidentification system when a new face is saved. Other than that, “IP\_controller.py” is an extra program designed for debugging purpose which let developer check, add and remove the recorded IP address.

```
upboard@upboard-UP-CHT01:~/Desktop/face_registrations$ ll gallery
total 36
drwxrwxr-x 2 upboard upboard 4096 Ogos 26 17:38 ./
drwxr-xr-x 5 upboard upboard 4096 Ogos 26 16:21 ../
-rw-rw-r-- 1 upboard upboard 7947 Ogos 26 16:09 xiaaodoggy-0.jpg
-rw-rw-r-- 1 upboard upboard 11023 Ogos 26 14:08 yiekheng-0.jpg
-rw-rw-r-- 1 upboard upboard 6650 Ogos 26 16:21 yiyang-0.jpg
upboard@upboard-UP-CHT01:~/Desktop/face_registration$
```

Figure 4.16: Registered Face Image in gallery folder



Figure 4.17: Registered Face Image in gallery folder 2

Figure above showed the files that contained in gallery folder. Face reference image may be found in this folder. This folder act like a face database which provide a platform to let program compare the cosine distance with each image.

```
upboard@upboard-UP-CHT01:~/Desktop/face_registration$ python3 IP_controller.py
1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 2
Please input the ip address: 127.0.0.1

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 2
Please input the ip address: 192.168.0.1

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 2
Please input the ip address: 123.123.123.a
[ ERROR ] [ 09/02/2020 04:07:10 PM ] Invalid IP address

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 1
1 : 127.0.0.1
2 : 192.168.0.1

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 4
upboard@upboard-UP-CHT01:~/Desktop/face_registration$ █

upboard@upboard-UP-CHT01:~/Desktop/face_registration$ python3 IP_controller.py
1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 3
1 : 127.0.0.1
2 : 192.168.0.1
Please input the order to remove ip address: 2

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 3
1 : 127.0.0.1
Please input the order to remove ip address: 2
[ ERROR ] [ 09/02/2020 04:19:21 PM ] Invalid Order

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 1
1 : 127.0.0.1

-----

1. SHOW IP LIST
2. ADD IP
3. REMOVE IP
4. EXIT PROGRAME
Please select from the menu: 4
upboard@upboard-UP-CHT01:~/Desktop/face_registration$ █
```

Figure 4.18: IP\_controller.py



Although “IP\_controller.py” is an extra program, we still need to ensure this program will work smoothly and deal for any exception. This figure showed the testing of all possible input to “IP\_controller.py” which had ensured that program won’t get the error easily.

After we launched the program, program will start another thread to listening port 5001 to register incoming face reidentification system IP.

```
upboard@upboard-UP-CHT01:~/Desktop/face_registrations$ python3 main.py
[ INFO ] [ 09/03/2020 08:56:29 PM ] Loading plugins for devices: {'CPU'}
[ INFO ] [ 09/03/2020 08:56:39 PM ] Listening Port 5001
[ INFO ] [ 09/03/2020 08:58:39 PM ] Using CPU extensions library '/opt/intel/opencvino/i
pu_extension_sse4.so'
[ INFO ] [ 09/03/2020 08:56:39 PM ] Plugins are loaded
[ INFO ] [ 09/03/2020 08:56:39 PM ] Loading models
```

Figure 4.19: Port Listening at 5001

Figure showed the system is listening port 5001 which used to register incoming system’s IP.

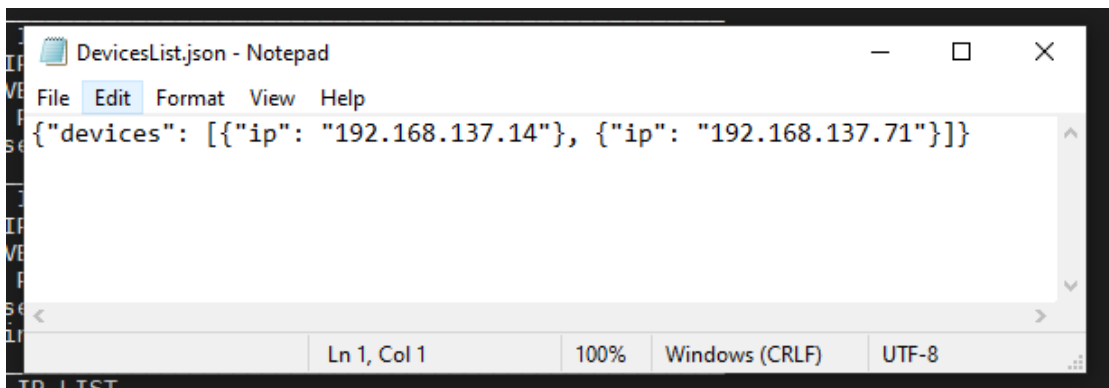


Figure 4.20: DevicesList.json

Figure showed the system’s registered IP address of each face registration system’s IP. 192.168.137.14 is the IP address of the first face reidentification system. Face registration system will send gallery file follow the sequence of the json file.



### 4.8.3 GUI of Face Registration System

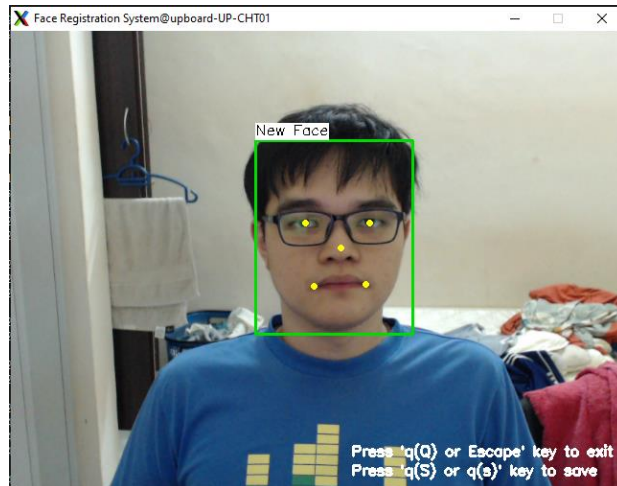


Figure 4.21: GUI of Face Registration System

A window will be brought to the front to display the captured and process frame result. User may click key “q” or “escape” to close the program or press s key to register and save the image into face database.

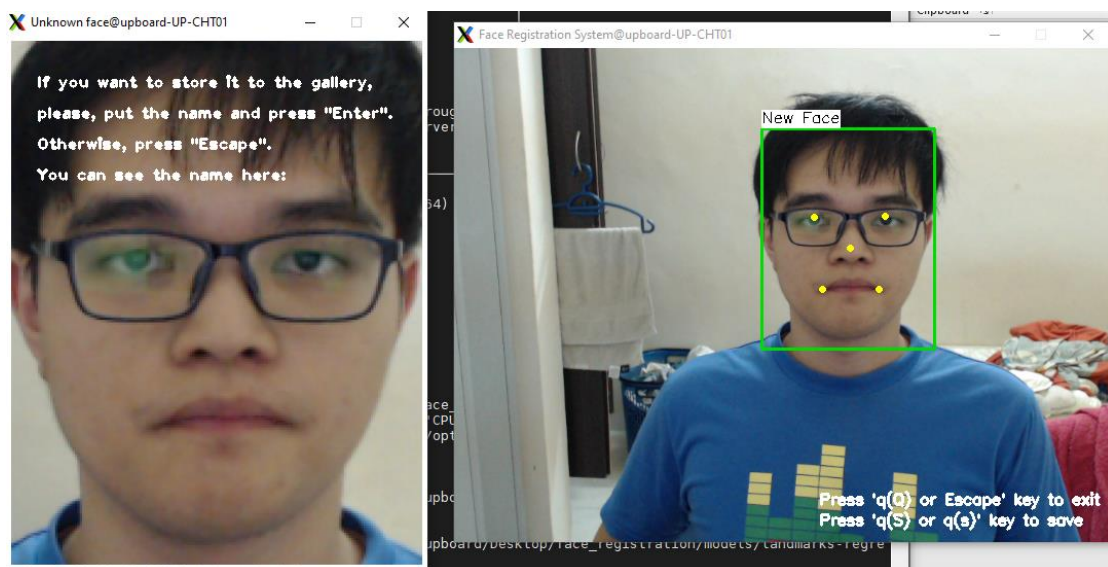


Figure 4.22: Save Face Window

When user click key ‘s’, detected face will be scale and display out to prompt user to key in the name. User may direct key in name into the pop out window and finally click key “enter” to save the image.

### New Face Scenario:

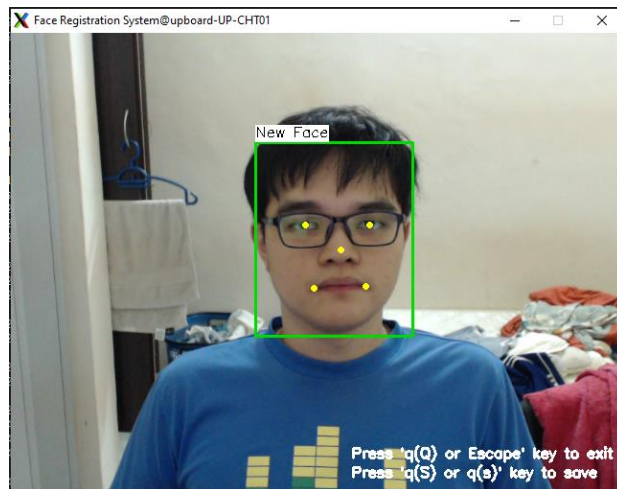


Figure 4.23: New Face Detected

When face registration system launched, system will detect the human face, if the face is a new face, system will show “New Face” above the detected face’s region of interest.

### Registered Face Scenario:

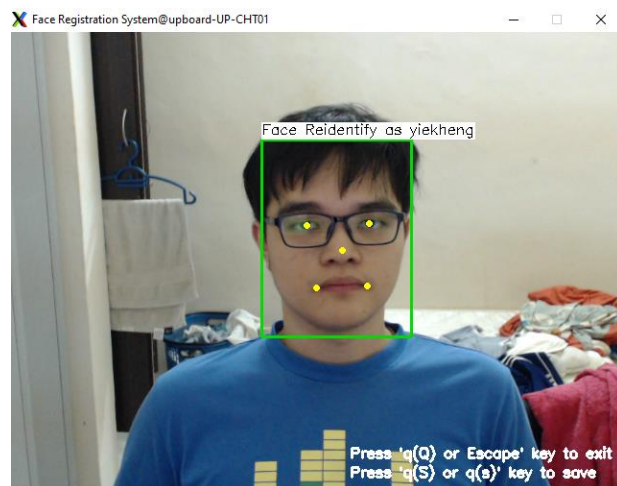


Figure 4.24: Registered Face Detected

If the face is registered in system, “Face Reidentify as XXX” will appear above the detected face’s region of interest.

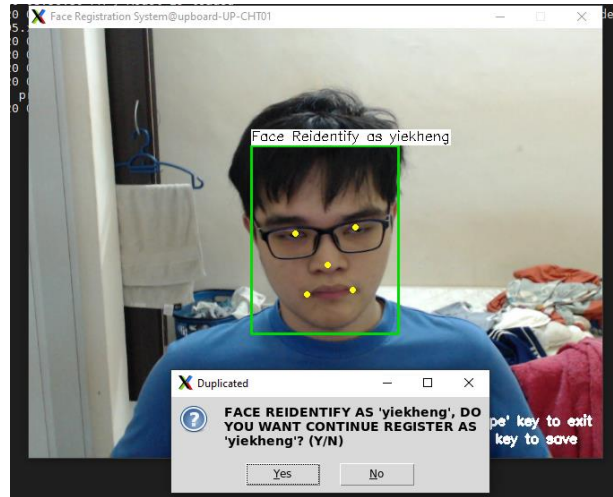


Figure 4.25: Duplicated Face Image Save

System will ask your choice for adding second image or create a new identify for the face when you click the key S to save the detected face image.

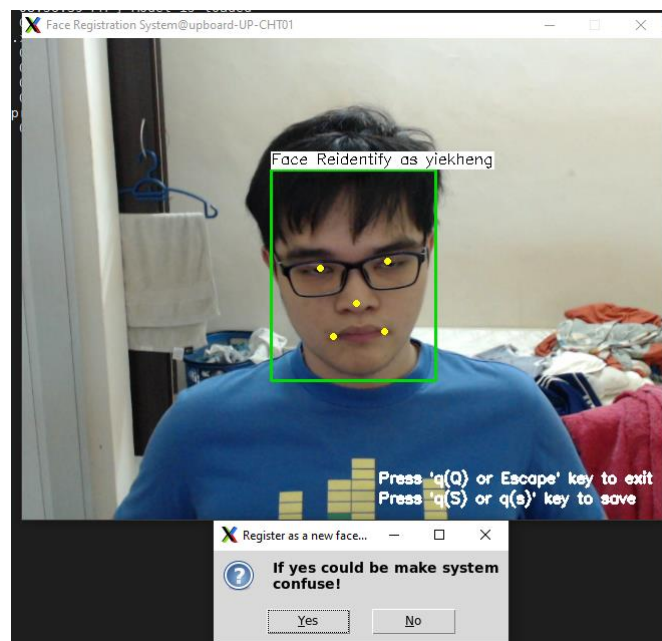


Figure 4.26: Duplicated Face Image Save as new identity

The worst-case scenario of the system is you reidentify as some other person, this mean that your face is similar to that person based on the model's algorithm. If keep adding your face into system may result in the person reidentify as you and you being reidentify as that person.

## Multiple Face Scenario:

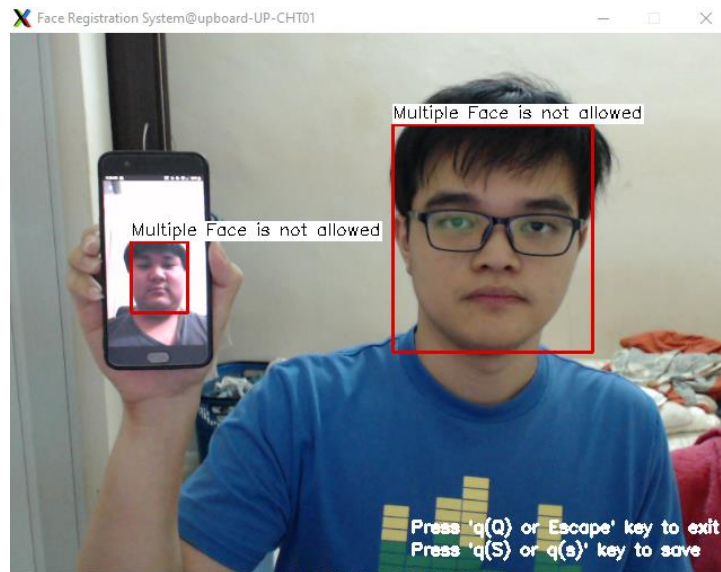


Figure 4.27: Multiple Face at Face Registration System

The registration system will detect the number of faces in the frame, it will automatically stop reidentify person when the number of faces is larger than two. If the face registration is register more than two face to one person, system will go confuse and will not be able to reidentify these two people anymore.

#### 4.8.4 Face Reidentification System

```
[setupvars.sh] OpenVINO environment initialized
upboard2@upboard2-UP-CHT01:~$ cd Desktop/face_reidentification/
upboard2@upboard2-UP-CHT01:~/Desktop/face_reidentification$ ll
total 60
drwxr-xr-x 4 upboard2 upboard2 4096 Ogos 23 16:20 ./
drwxr-xr-x 3 upboard2 upboard2 4096 Ogos 27 16:15 ../
-rw-rw-r-- 1 upboard2 upboard2 27961 Ogos 18 17:51 face_reidentification.py
-rw-r--r-- 1 upboard2 upboard2 4150 Sep 18 2019 ie_module.py
-rw-r--r-- 1 upboard2 upboard2 1970 Ogos 23 16:08 main.py
drwxr-xr-x 2 upboard2 upboard2 4096 Jul 5 19:02 models/
drwxrwxr-x 2 upboard2 upboard2 4096 Ogos 23 15:17 __pycache__/
-rw-r--r-- 1 upboard2 upboard2 48 Sep 18 2019 requirements.txt
upboard2@upboard2-UP-CHT01:~/Desktop/face_reidentification$
```

Figure 4.28: Folder hierarchy of Face Reidentification System

As the figure showed, face reidentification system will also have an entry point which is “main.py”. User and developer just need to type in “python3 main.py” to bring up the program.

#### 4.8.5 GUI of Face Reidentification System

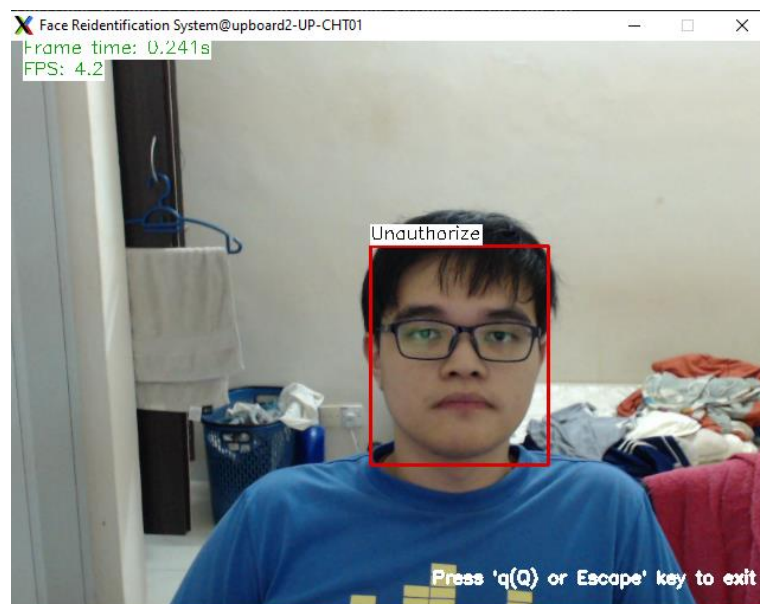


Figure 4.29: GUI of Face Reidentification System

GUI of face reidentification system is same like GUI of face registration system but less from the save key to press. Face reidentification no much work to do, it only need reidentify person.



### Register IP Scenario:

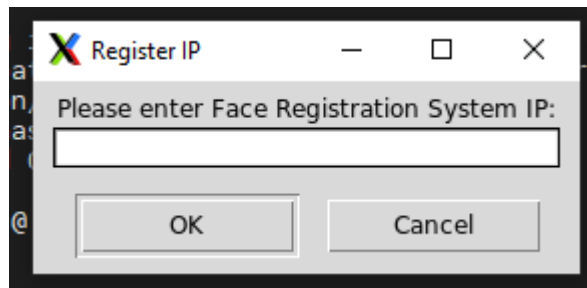


Figure 4.30: Register IP Window

Once the system brought up, system will prompt user to enter the IP address of the face registration system. This step is to register IP address of face reidentification system to face registration system.

### Registered Face Scenario:

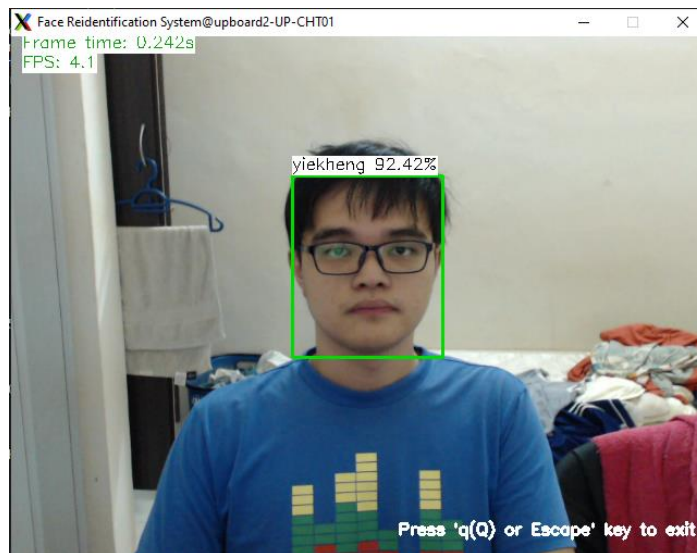


Figure 4.31: Registered Face Detected

The detected face will go through comparing process which compare with each face images in faces database in cosine distance. Result of the comparison will be get and output to the visualizer to display it out.

## Unauthorized Face Scenario:

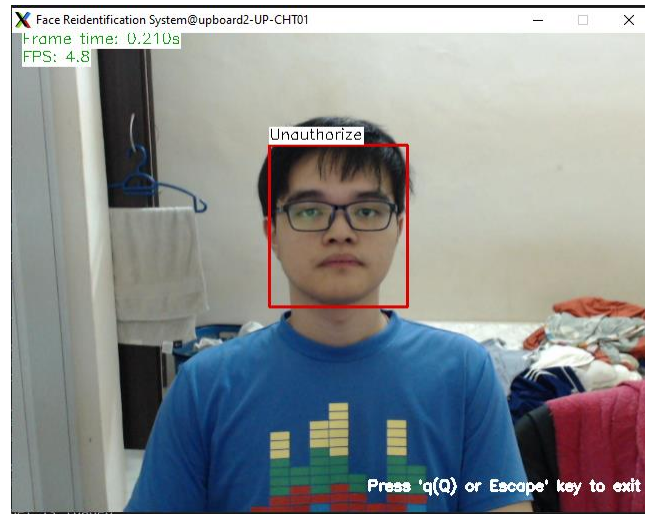


Figure 4.32: Unauthorized Face Detected

When some unauthorized person's face is captured by camera, system will draw it with red rectangle and showing "Unauthorized" label above the detected region of interest. If required, developer may easily add their Telegram bot into code to notify the manager that "there are some unauthorized person go into factory".

## 4.9 Use Cases

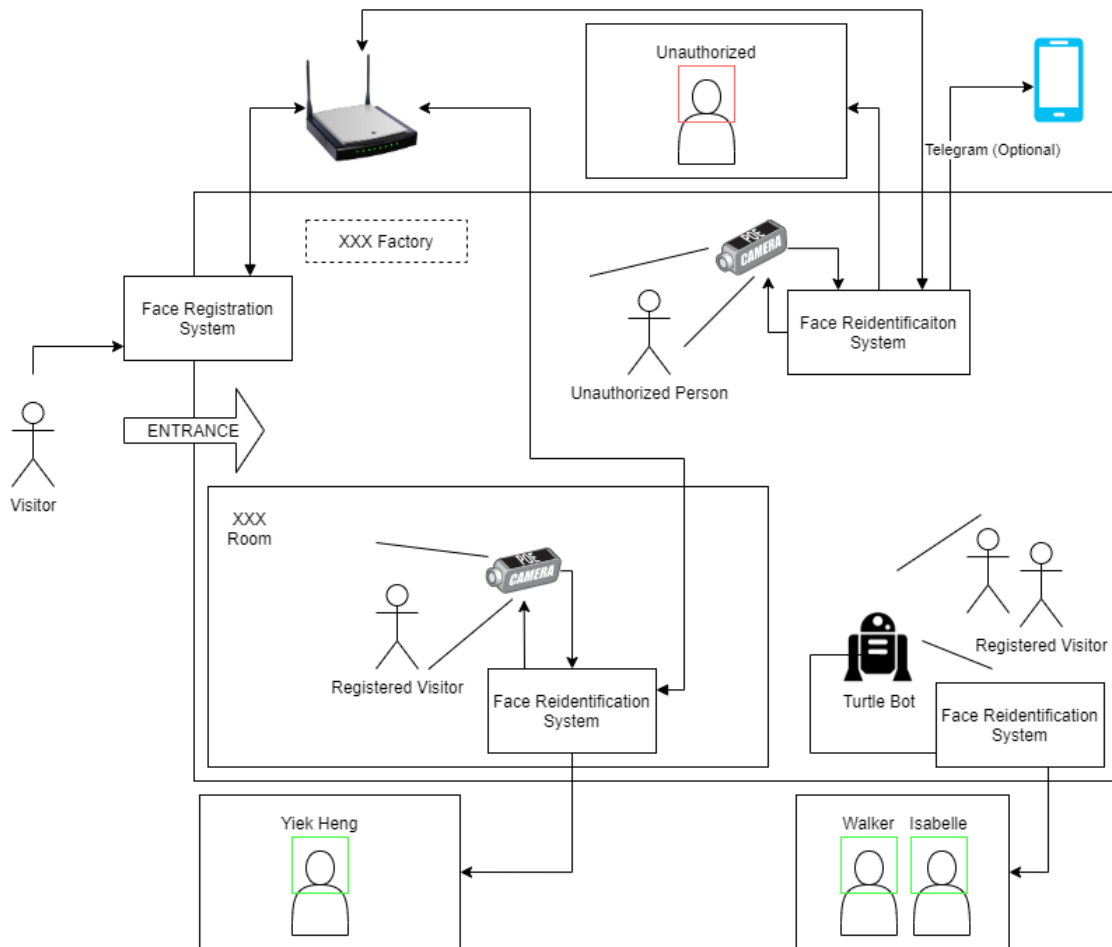


Figure 4.33: Use Case of Face Reidentification System

Figure above is the operating concept of whole system. When a new visitor is requesting to go into the factory, face registration is required. As mentioned above, face registration will send to all face reidentification system though TCP connection. That mean every face reidentification system will update the faces databases at the same time. There is no require to capture the visitor's face million time and only 1 face image required to reidentify the person. As the concept mentioned, Unauthorized Person will be detected as Unauthorized and the colour of the region of interest will go red. If require, developer may further code on the face reidentification system that may notify the manager through Telegram. Telegram bot may be built up through Telegram Bot API and developer may find the tutorial through its official documentation. The person who registered will be reidentify and the name of the person will appear above the region of interest. The colour of the region of interest will also go green.



## Chapter 5: System Evaluation and Discussion

In this chapter, verification plan that designed before will be carried out. Different scenarios will be tested such as 1 face, 2 face, 3 face and 4 face. First will test for accuracy of the face detection model within 1000 frames and face reidentification will also be tested within 1000 frames.

The accuracy of the face detection can be calculated by the equation:

$$\text{Accuracy (face detection)} = \frac{1 - \frac{|Y(x) - x|}{x}}{1000} \times 100\%$$

, where x is the ground truth of number of detection face.

Other than that, speed of the face detection is calculated by the equation:

$$\text{Speed (s/f)} = \frac{\sum_{i=0}^{999} (T(f(i+1)) - T(f(i)))}{1000}$$

, where f is frame and T is time.

The accuracy of the face reidentification is some sort like face detection's accuracy equation:

$$\begin{aligned} & \text{Accuracy (face reidentificaition)} \\ &= \frac{\left(1 - \frac{|Y(x) - x|}{x}\right) \left(\frac{x}{x+y}\right) + \left(1 - \frac{|Y(y) - y|}{y}\right) \left(\frac{y}{x+y}\right)}{1000} \times 100\% \end{aligned}$$

, where x is the number of known persons, y is the number of unknown persons.

The speed equation of face reidentification is same as face detection.

## 5.1 Face Detection

For calculating the accuracy and execution speed of the face detection. A script is needed to calculate the accuracy. Below is the script flow to detect the accuracy and speed.

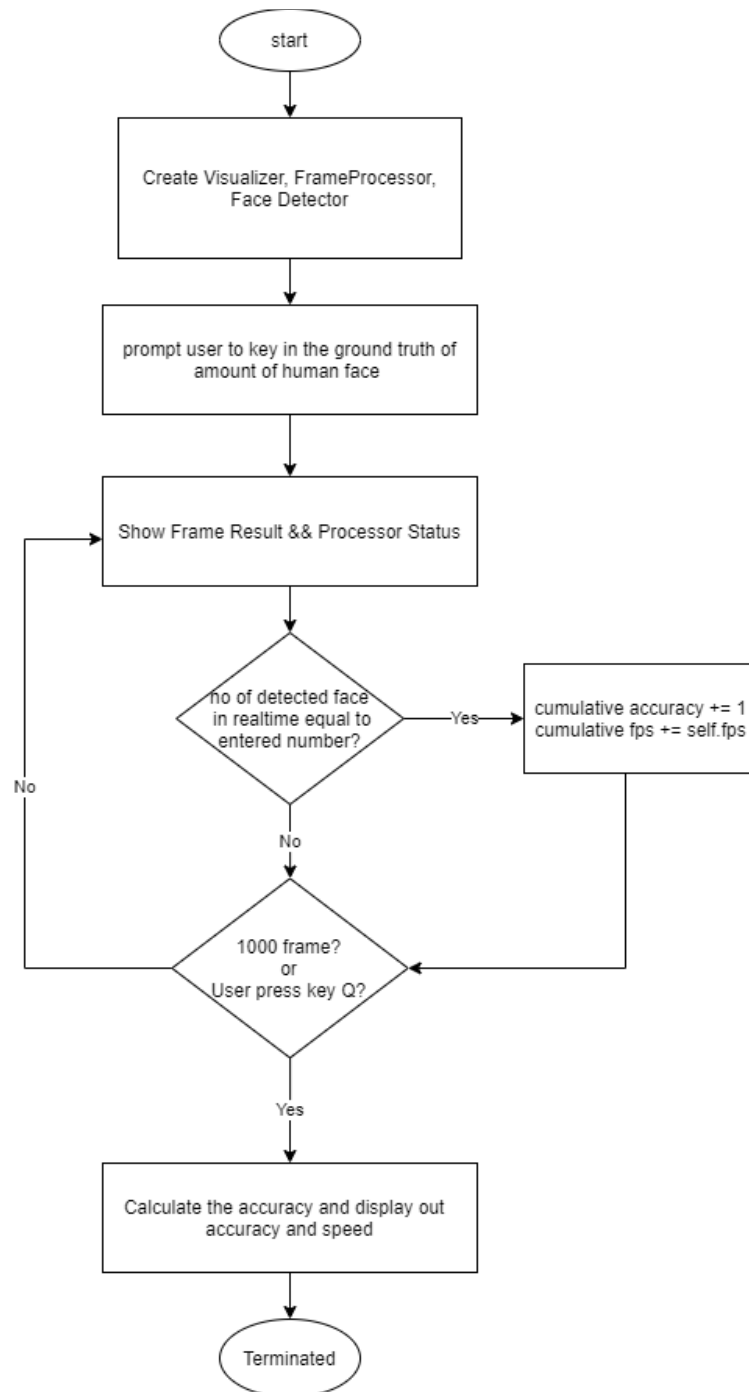


Figure 5.1: Flow Chart of Face Detection Verification Script

### 5.1.1 Face Detection – 1 Face

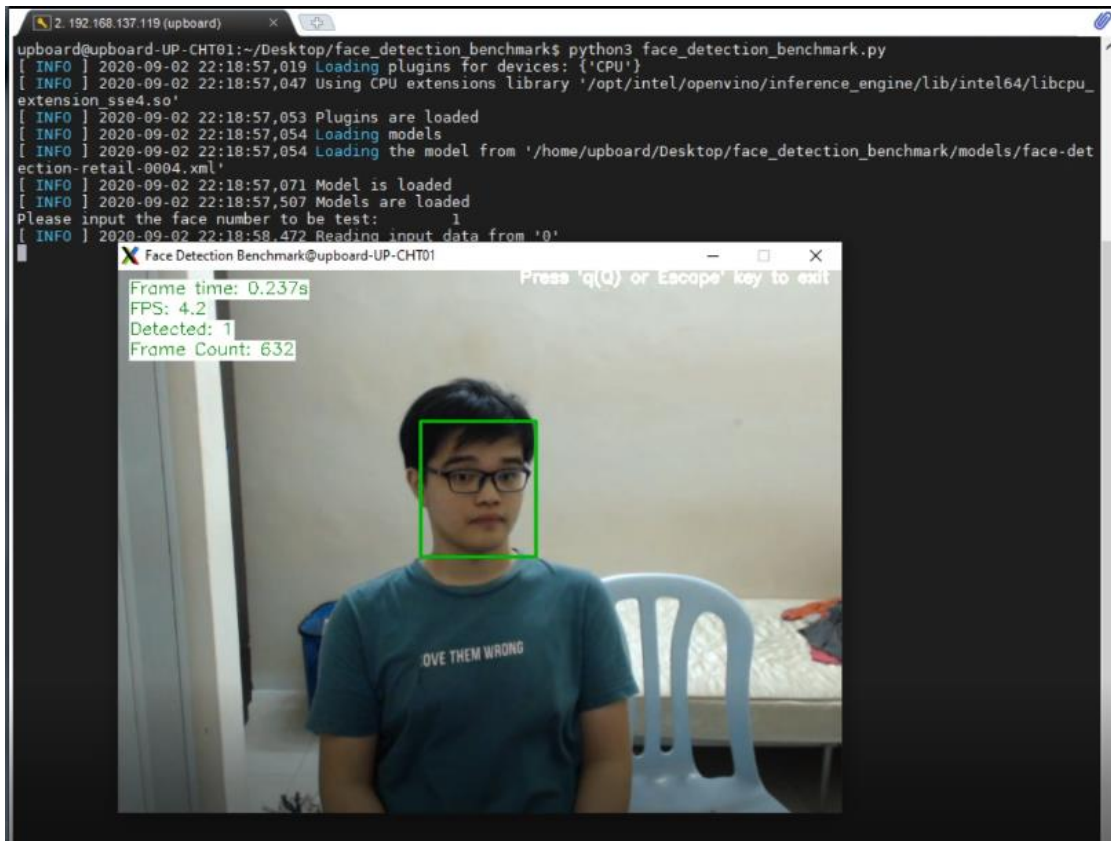


Figure 5.2: 1 Face Detection

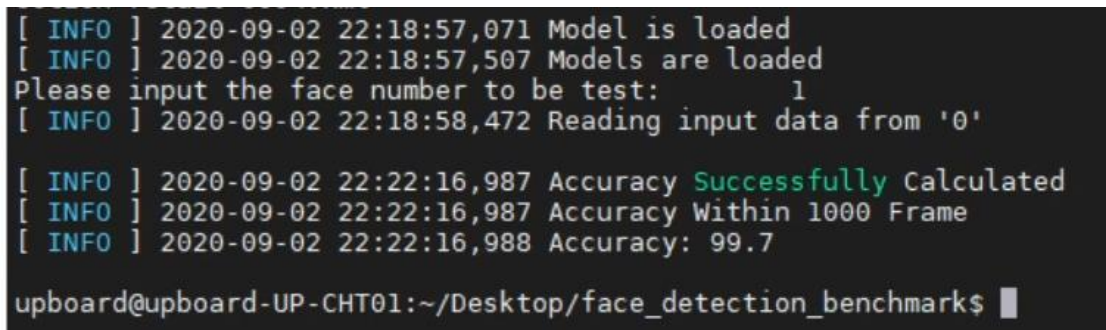


Figure 5.3: Accuracy of 1 Face Detection

Figure showed that face detection model has 99.7% accuracy when detecting 1 face and around 0.238 second to process 1 frame.

### 5.1.2 Face Detection – 2 Face

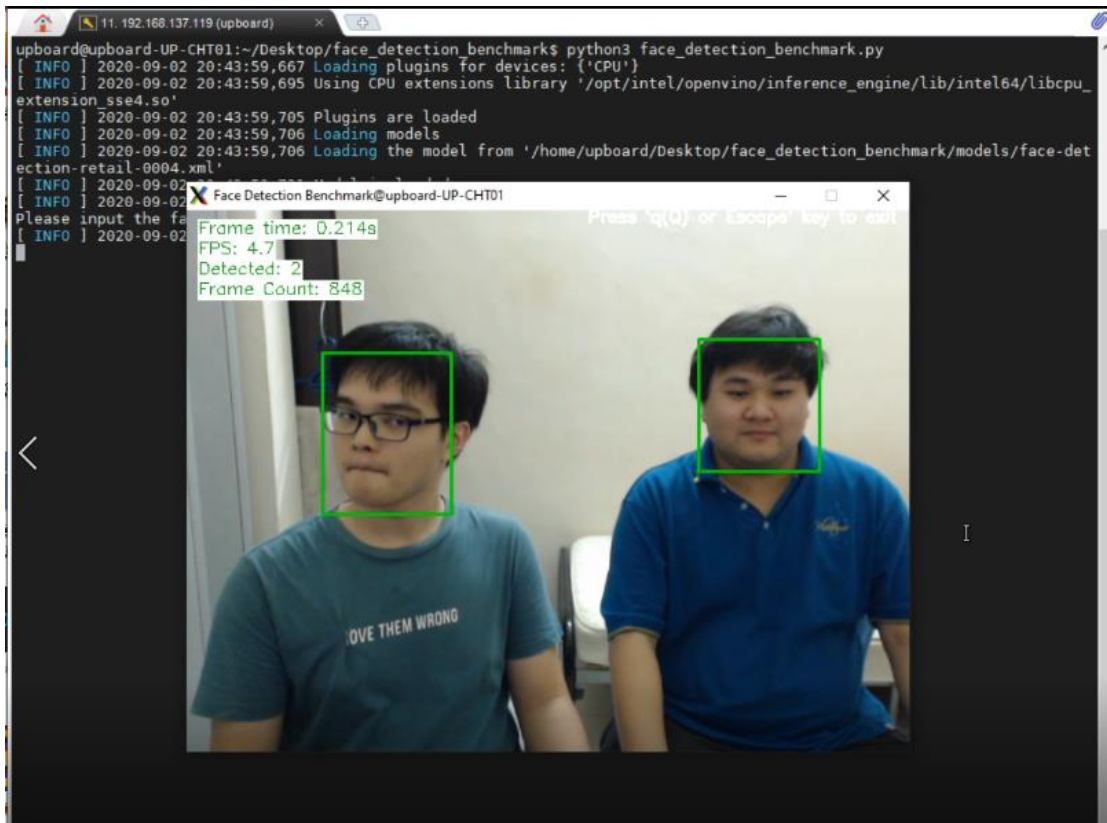


Figure 5.4: 2 Face Detection

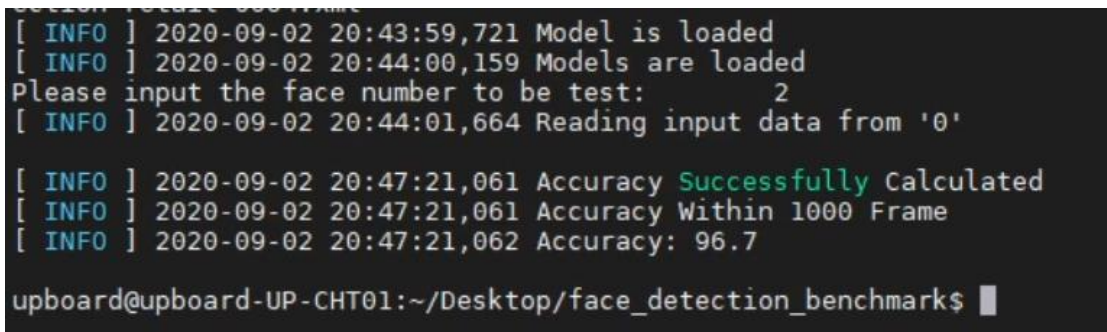


Figure 5.5: Accuracy of 2 Face Detection

Figure showed that face detection model has 96.7% accuracy when detecting 2 faces and around 0.213 second to process 1 frame.

### 5.1.3 Face Detection – 3 Face

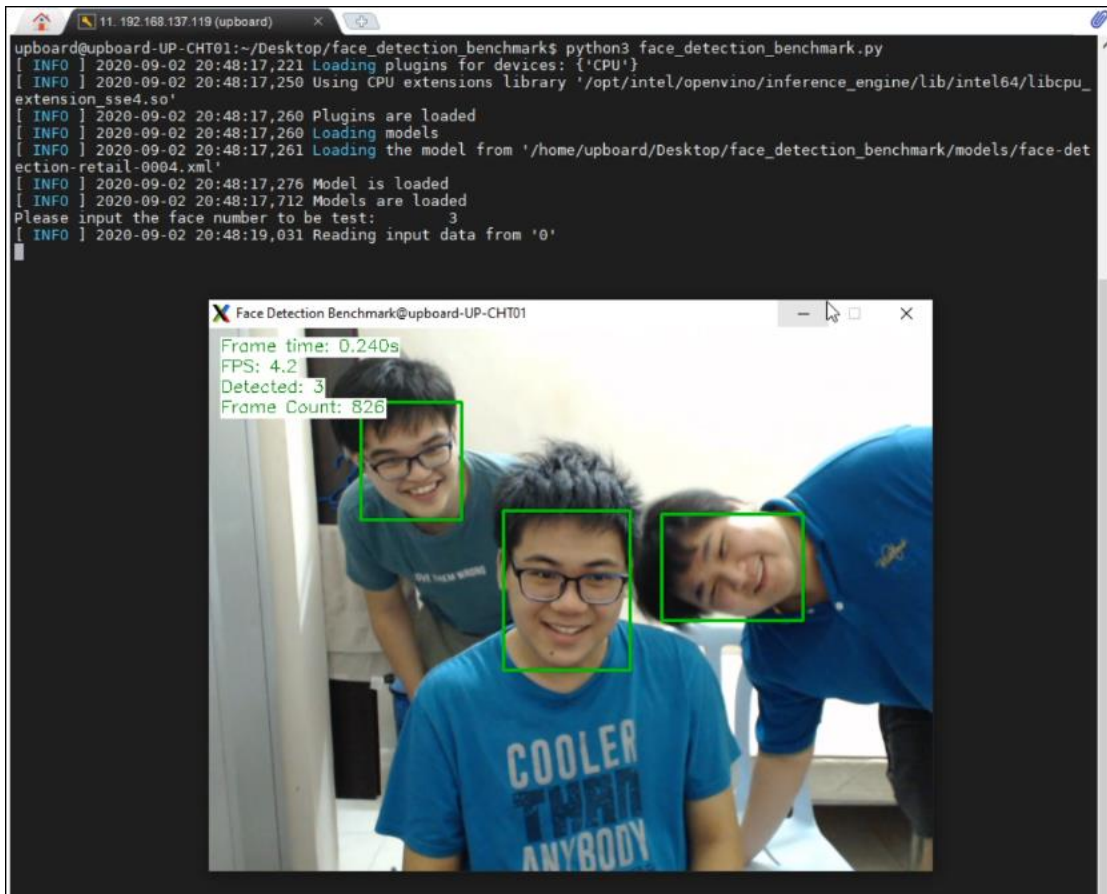


Figure 5.6: 3 Face Detection

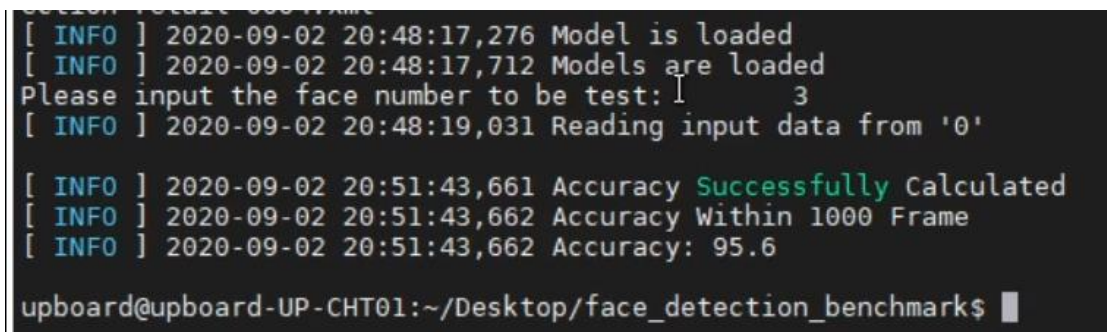


Figure 5.7: Accuracy of 3 Face Detection

Figure showed that face detection model has 95.6% accuracy when detecting 3 faces and around 0.238 second to process 1 frame.



### 5.1.4 Face Detection – 4 Face

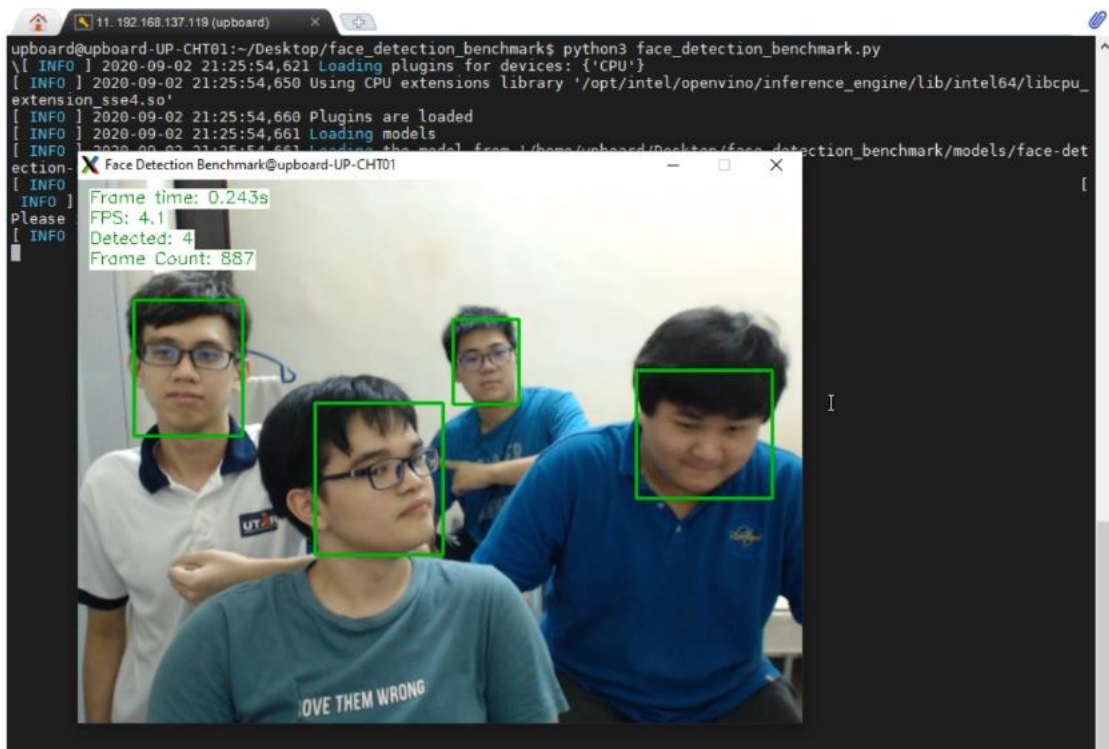


Figure 5.8: 4 Face Detection

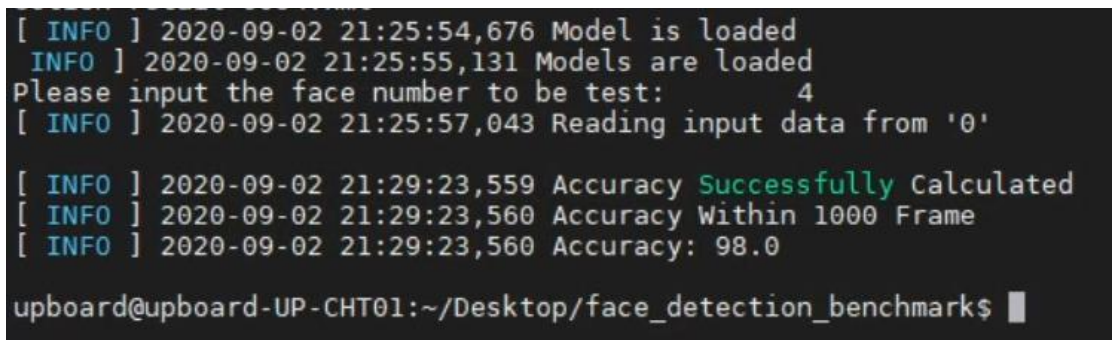


Figure 5.9: Accuracy of 4 Face Detection

Figure showed that face detection model has 98.0% accuracy when detecting 4 faces and around 0.244 second to process 1 frame.

### 5.1.5 Face Detection Test Result

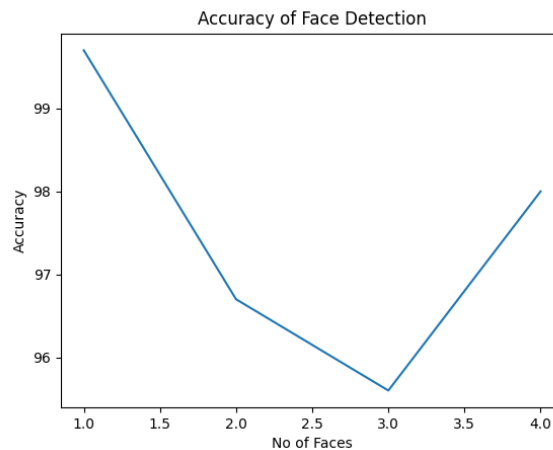


Figure 5.10: Relationship between accuracy and no of faces in face detection

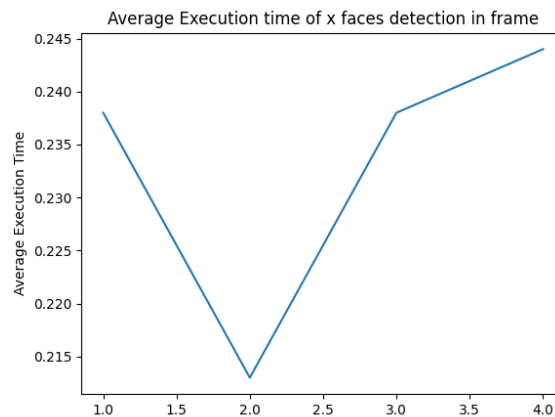


Figure 5.11: Relationship between execution speed and no of faces in face detection

Based on the data collected, a graph is plot out as the figure showed. From the graph, we knew that accuracy of the face detection model is going down when faces become more but the accuracy increases at faces equal to 4. This indicated that number of faces is not the only limited to decrease the accuracy. The other possible attribute to limit the accuracy of the model could be the complexity of the frame. The more transition of colour change of frame might affect the accuracy of model. Besides that, the average execution time of the face detection model has no huge differences. Hence, more face might not affect the model.

## 5.2 Face Reidentification

For calculating the accuracy and execution speed of the face reidentification system. A script is needed to calculate the accuracy. Below is the script flow to detect the accuracy and speed.

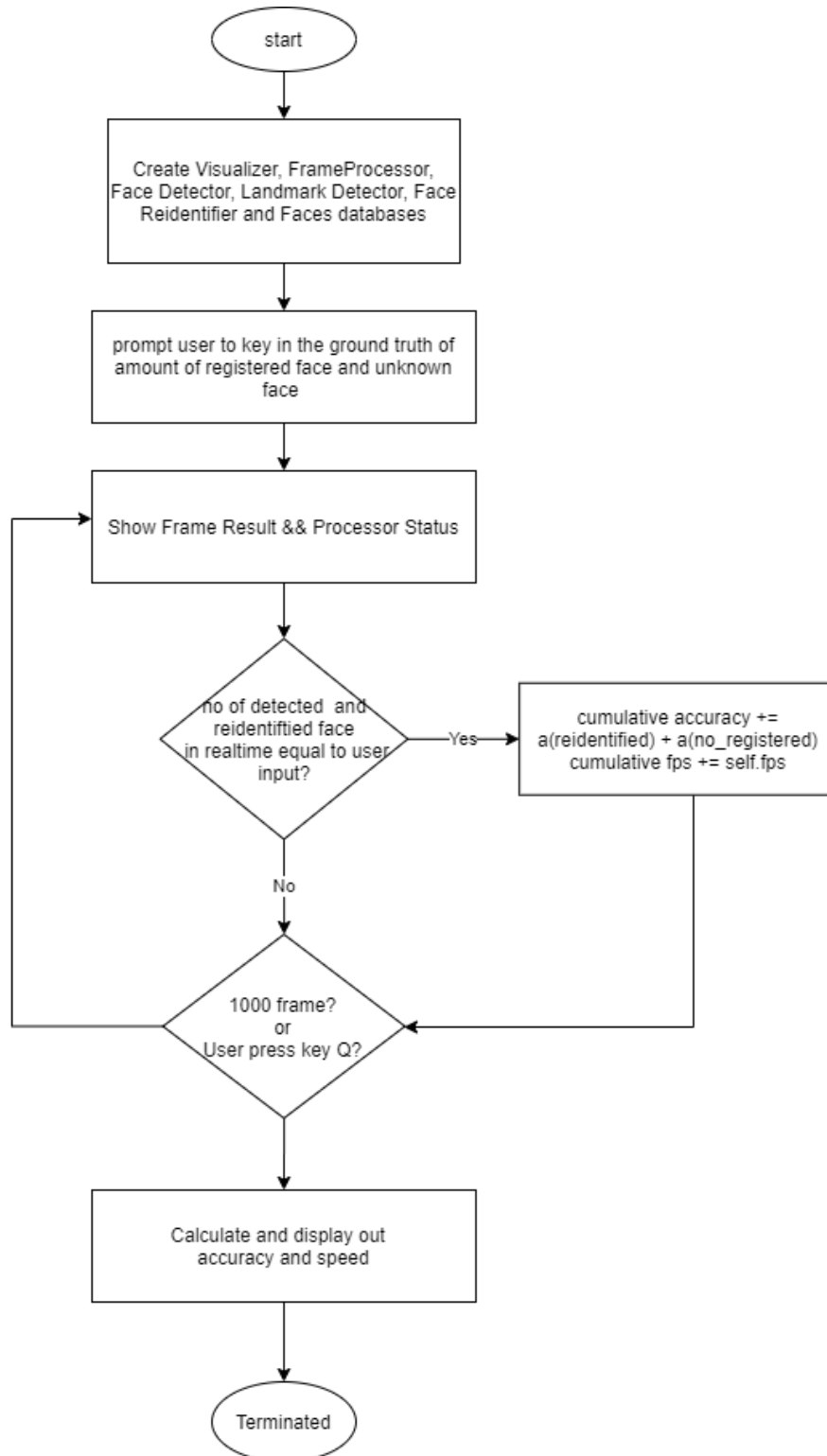


Figure 5.12: Flow Chart of Face Reidentification Verification Script



## 5.2.1 Face Reidentification – 1 Face

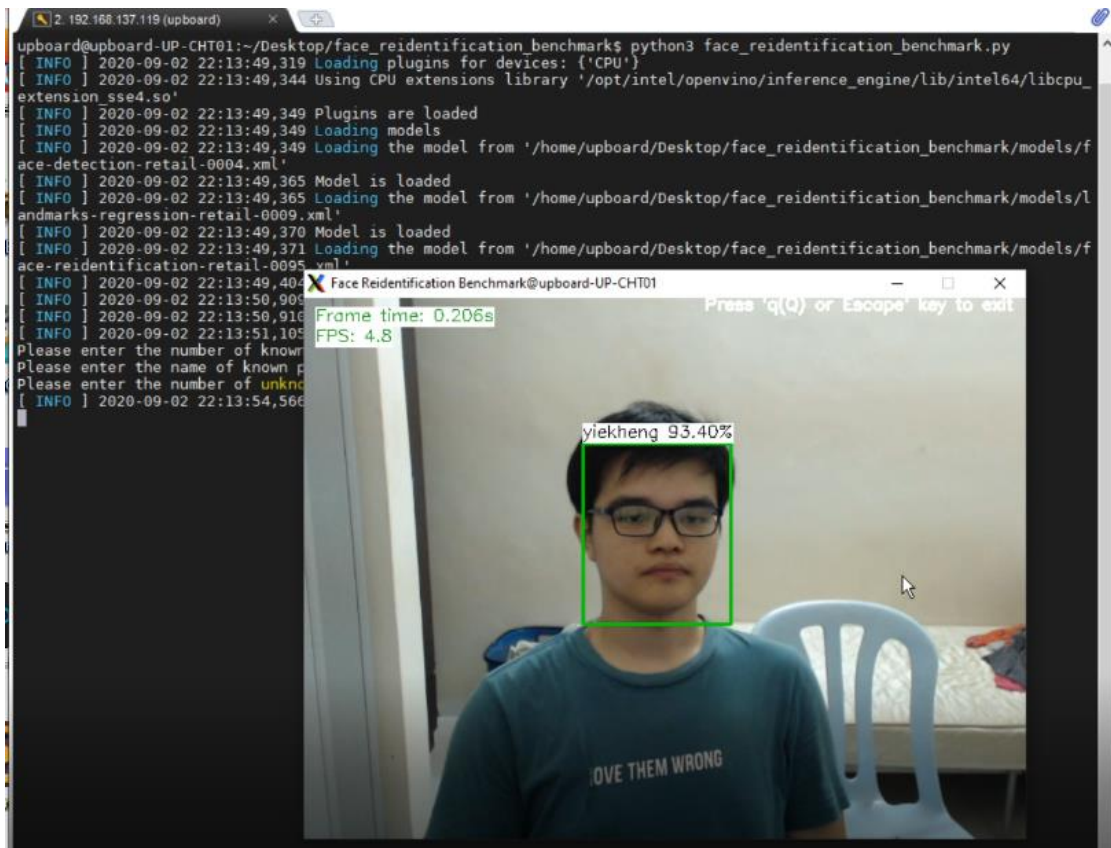


Figure 5.13: 1 Face Reidentification

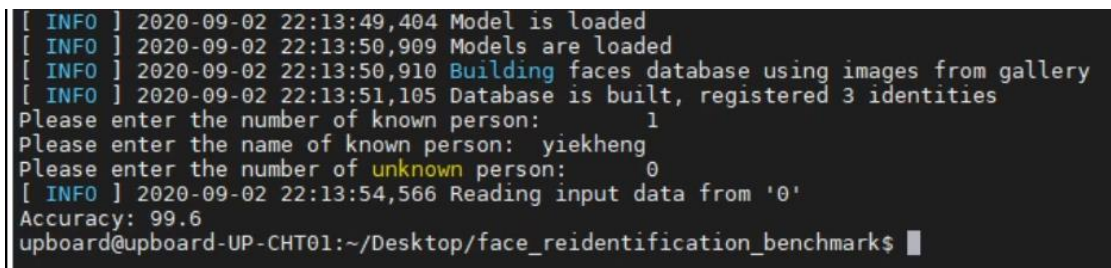


Figure 5.14: Accuracy of 1 Face Reidentification

Figure showed that face reidentification system has 99.6% accuracy when reidentifying 1 face and around 0.208 second to process 1 frame.

## 5.2.2 Face Reidentification – 2 Face

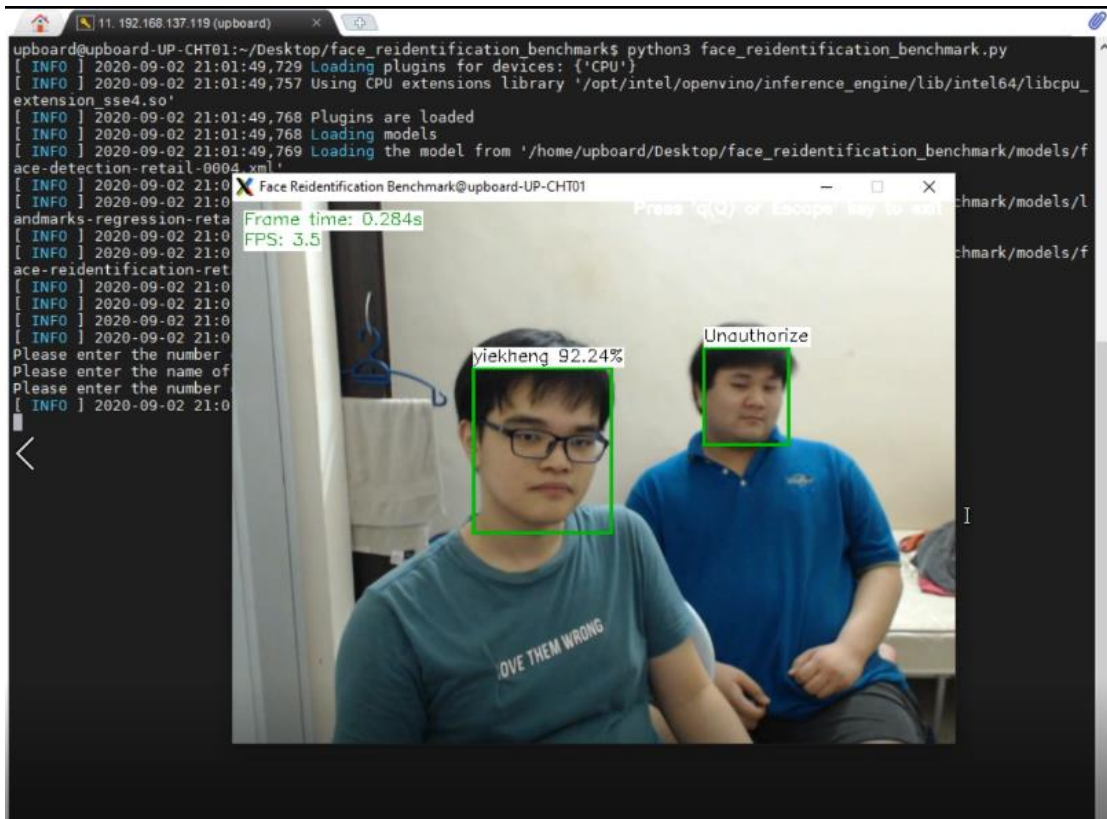


Figure 5.15: 2 Face Reidentification

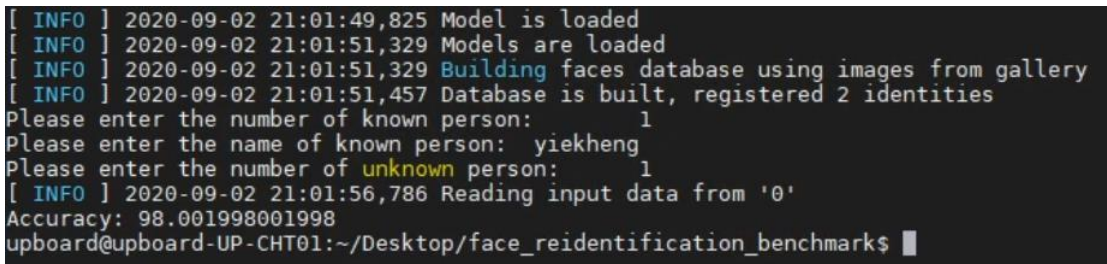


Figure 5.16: Accuracy of 2 Face Reidentification

Figure showed that face reidentification system has 98.0% accuracy when reidentifying 2 face which is 1 registered face and 1 unregistered face. System have around 0.286 second to process 1 frame.

### 5.2.3 Face Reidentification – 3 Face

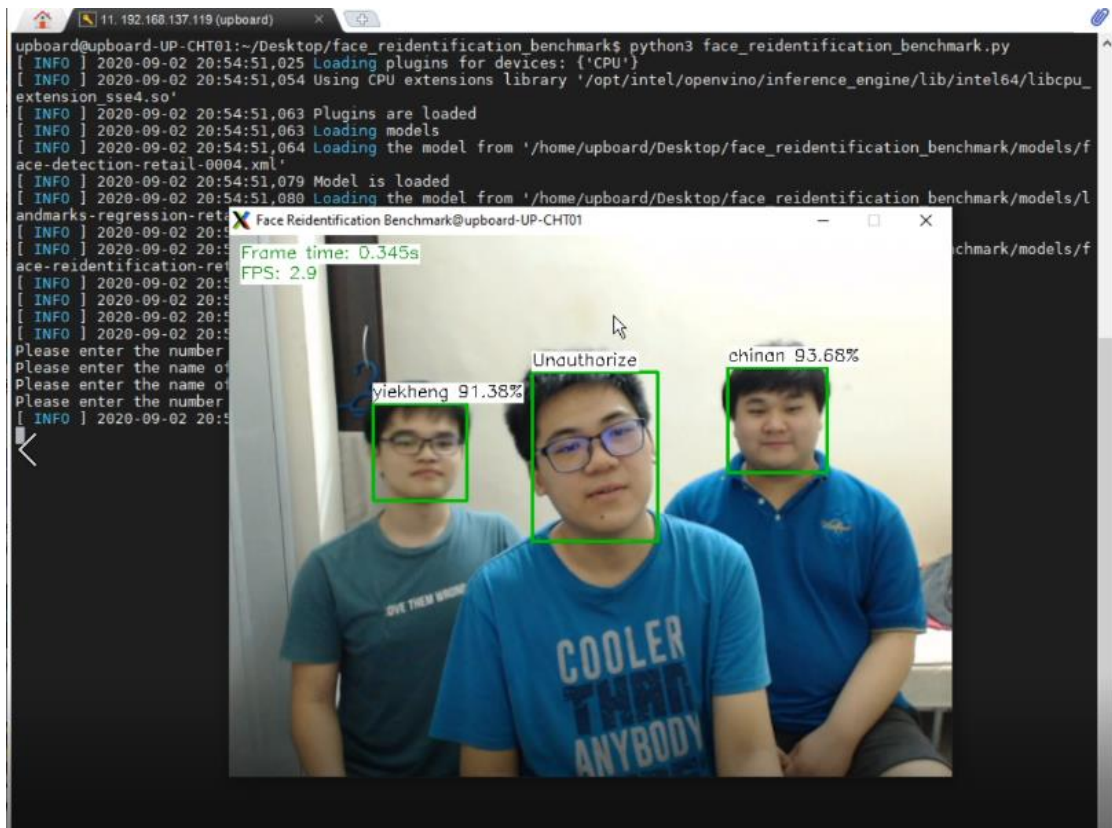


Figure 5.17: 3 Face Reidentification

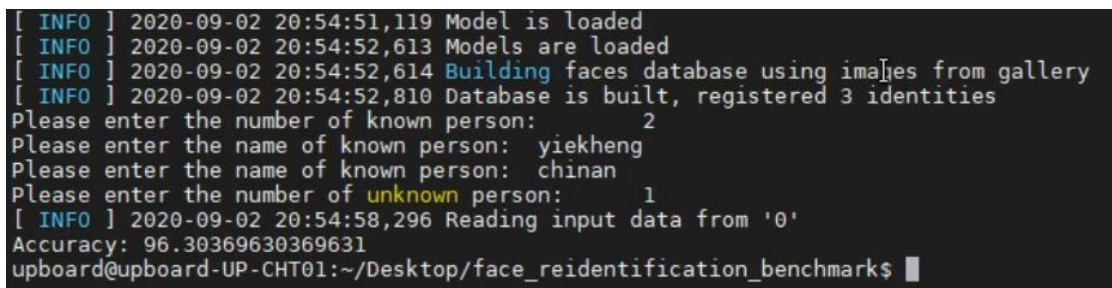


Figure 5.18: Accuracy of 3 Face Reidentification

Figure showed that face reidentification system has 96.30% accuracy when reidentifying 3 face which is 2 registered face and 1 unregistered face. System have around 0.345 second to process 1 frame.



## 5.2.4 Face Reidentification – 4 Face

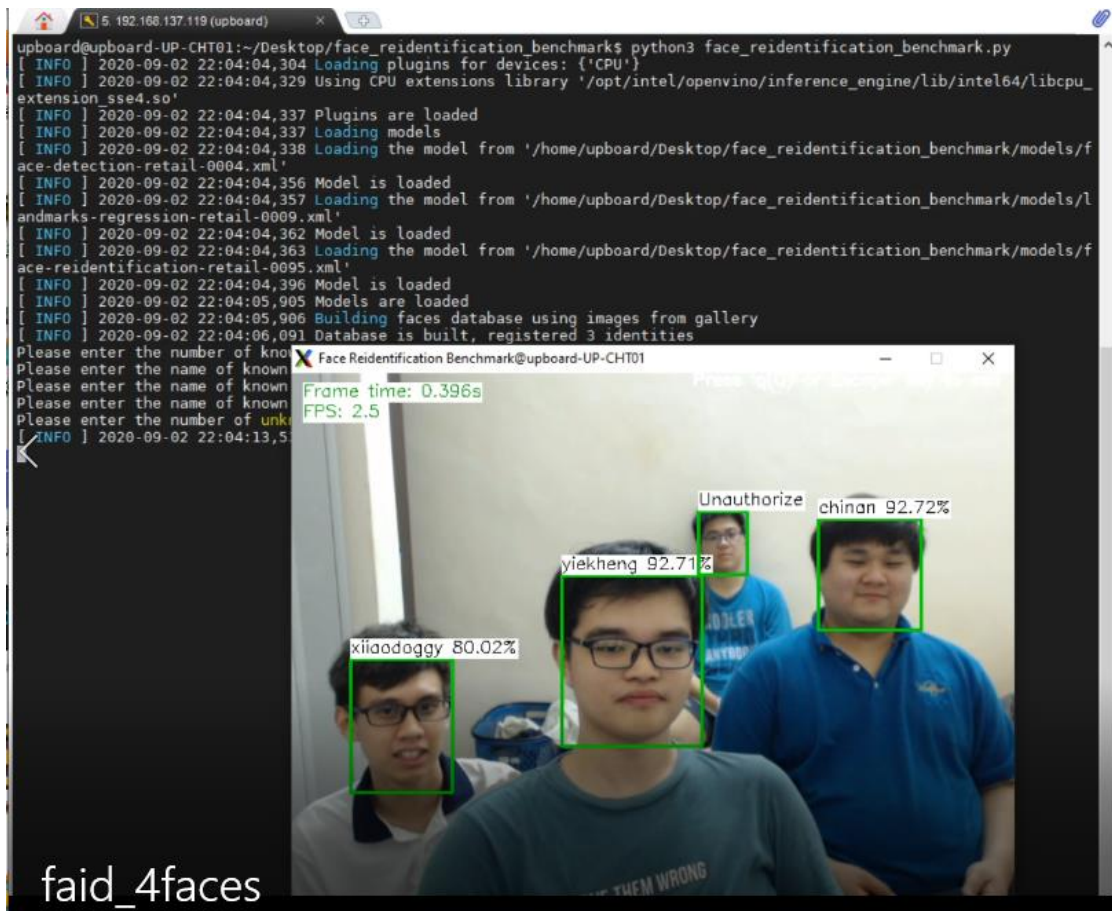


Figure 5.19: 4 Face Reidentification

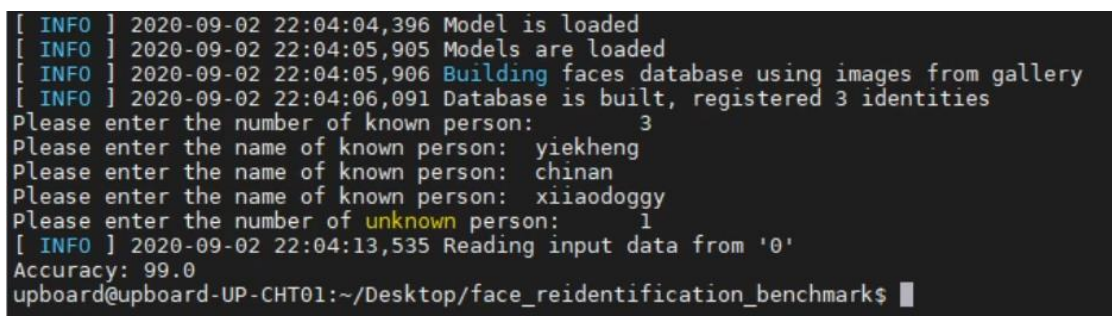


Figure 5.20: Accuracy of 4 Face Reidentification

Figure showed that face reidentification system has 99.0% accuracy when reidentifying 4 face which is 3 registered face and 1 unregistered face. System have around 0.4 second to process 1 frame.

## 5.2.5 Face Identification Test Result

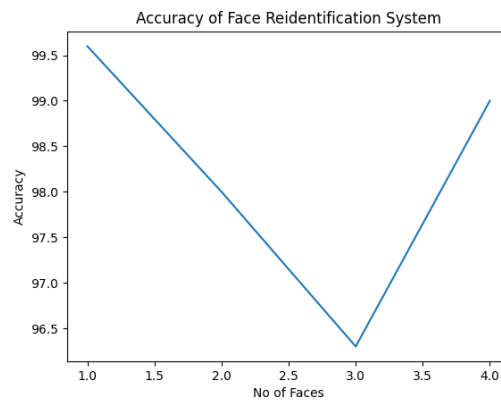


Figure 5.21: Relationship between accuracy and no of faces in face reidentification

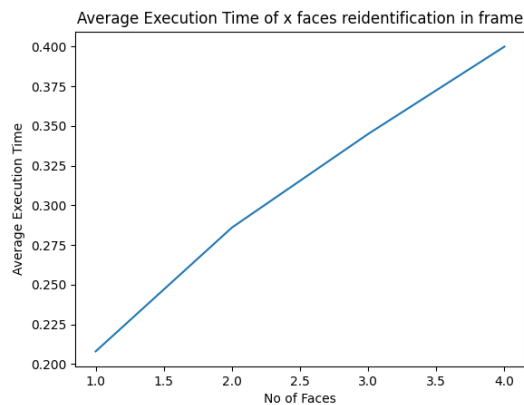


Figure 5.22: Relationship between execution speed and no of faces in face reidentification

Based on the graph above, we knew that accuracy of face reidentification system is going down when faces become more and accuracy increase at 4. This could be the same problem met in the face detection model. Hence, complexity of frame could be the key point that limited accuracy of face detection model. Other than that, average execution time of face reidentification system is going longer while the face become more. This noticed that number of faces will affect the execution times of face reidentification, high computation computer is required for more person reidentification. In a result, limited persons of reidentification is no problem for embedded board. If the number of people continues to rise, the board will take longer to reidentify every person's identity. If the person is limited to maximum 4 faces, up board with OpenVINO did very well already.

### 5.3 Limitation and Future Enhancement

The limitation and weakness of the system could be the up-board's computation power. Although up board is higher computation power compare than other low-end embedded board, up board is still not enough power to deal with more people reidentification at the same time. If possible, please change the embedded board to other embedded board with Intel CPU which is able to run the OpenVINO. The other way to solve this problem could be implement the GPU extension to OpenVINO. But this way will no have much differences from CPU based on current's Intel's GPU performance. If Intel's GPU performance becomes better in the future, user or developer could try to add the GPU extension of the OpenVINO to the code. If user and developer have high speed internet connectivity, user and developer could try to build OpenVINO in server and provide the application programming interface for client to connect. Then every face reidentification system could have low computation usage which send the frame to the server for reidentification purpose, and then send the processed frame back to the client for displaying purpose. This method is on the premise that the server is based on Intel architecture.

Other than that, this system also got another problem is required to be solve. When face reidentification system receives the update face database signal from face registration system, face reidentification system will destroy the visualizer and reopen to update the faces database due to camera is occupy the channel, faces databases cannot be updated. This is a little bug and needed to be solve.

## Chapter 6: Conclusion

With the rapid changes of technology, people are increasingly focusing on accuracy, speed and confidentiality of a system. These performances are directly proportional to the success rate and the amount of true positive cases. An optimized model of neural network model is become more and more important to solve the problem stated above. Therefore, developer have to implement some latest technology in this project to optimized the model and make the inference and prediction of neural network become faster. Intel as the leader of the technology, they release the latest technology which is OpenVINO. OpenVINO is highly possible to speed up and improve the accuracy of pretrained model, so that developer may easily perform the face reidentification in any field and place.

Other than speed and accuracy, almost all of the current IoT devices are not compatible to perform inference to classify the image due to lack of GPU. An GPU may cost thousands of ringgits and even more. Hence, OpenVINO is now released and allow developer to use Intel UP Squared board to perform inference, meanwhile, a system with lower cost may be developed easily and only one GPU needed in system.

Though this project is face reidentification in smart factory by using OpenVINO, but this project is more like a demonstration to guide developer to perform face reidentification in any filed even in an unremarkable house to prevent thief invasion. This project guide developer to develop a system which has some function like capture face, pre-processing to captured face (adjust the brightness of captured faces, rotate the image to make different view from captured faces), real time face recognition and applied OpenVINO to make inference stage become faster.

Although the project looks perfect, there are still some possible improvements that could be done in future, such as higher-end of Intel CPU can be applied for faster speed of inference on IoT devices or latest neural network architecture like SSD-MobileNet V3 can be trained lead into low memory usage and high-performance result.

The proposed system in this project make face recognition system faster, more accuracy and confidentiality. In the future, face recognition system using OpenVINO will have more advantages in any SME. These may help SME to track their employees, visitors and machines thereby lower their cost to build a smart factory. Other than that, SME is able to retrieve visitors' information and location anywhere and anytime.



## Bibliography


- Athul P (2019) Building a face recognition system with FaceNet. Available from: <https://medium.com/@athul929/building-a-facial-recognition-system-with-facenet-b9c249c2388a> [Access 12 February 2020]
- Brandon Amos (2016) OpenFace. Available from: <http://cmusatyalab.github.io/openface/> [Accessed 12 August 2019].
- Damilola Omoyiwola (2018) Machine Learning on Facial Recognition. Available from <https://medium.com/datadriveninvestor/machine-learning-on-facial-recognition-b3dfba5625a7> [Accessed 12 August 2019].
- Eddie Forson (2017) Understanding SSD MultiBox – Real-time object detection in deep learning. Available from: <https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab> [Access 11 February 2020]
- GeeksforGeeks (2019) Confusion Matrix in Machine Learning. Available from: <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/> [Accessed 14 August 2019].
- Giovanni di Dio Bruno (2019) ROS\_OpenVINO. Available from: [https://github.com/gbr1/ros\\_OpenVINO](https://github.com/gbr1/ros_OpenVINO) [Accessed 13 August 2019].
- Intel Sdn Bhd (2018) Deep Learning for Computer Vision. Available from: <https://software.intel.com/en-us/OpenVINO-toolkit/deep-learning-cv> [Accessed 10 August 2019].
- Intel (2019) Open Model Zoo Model and Source Code. Available from: [https://github.com/openvinotoolkit/open\\_model\\_zoo](https://github.com/openvinotoolkit/open_model_zoo) [Accessed 12 August 2019].
- Jason Brownlee (2019) A Gentle Introduction to Deep Learning for Face Recognition. Available from: <https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/> [Accessed 13 August 2019].

- Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li (2016) Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. Available from: [https://kpzhang93.github.io/MTCNN\\_face\\_detection\\_alignment/](https://kpzhang93.github.io/MTCNN_face_detection_alignment/) [Access 10 February 2020]
- LewisLiew (2018) ROS2\_OpenVINO. Available from: [https://mp.weixin.qq.com/s/BgG3RGauv5pmHzV\\_hkVAdw](https://mp.weixin.qq.com/s/BgG3RGauv5pmHzV_hkVAdw) [Accessed 13 August 2019].
- Margaret Rouse (2005) Prototyping Model. Available from <https://searchcio.techtarget.com/definition/Prototyping-Model> [Accessed 14 August 2019].
- Mike Wheatley (2018) Intel's OpenVINO toolkit enables computer vision at the network edge. Available from <https://siliconangle.com/2018/05/16/intels-OpenVINO-toolkit-enables-computer-vision-network-edge/> [Accessed 10 August 2019].
- Schroff, Florian, Dmitry Kalenichenko, and James Philbin. (2015) FaceNet: A Unified Embedding For Face Recognition and Clustering. Available from: <https://arxiv.org/abs/1503.03832> [Accessed 10 February 2020]
- Vcvycy (2019) MTCNN For Android. Java. Tensorflow. Face Detection. Available from <https://github.com/vcvycy/MTCNN4Android> [Accessed 10/4/2020]
- Vishwesh Shrimali (2019) Using OpenVINO with OpenCV. Available from: <https://www.learnopencv.com/using-openvino-with-opencv/> [Access 11 February 2020]

## Appendices


### Poster

Universiti Tunku Abdul Rahman




# FACE REIDENTIFICATION SYSTEM TO TRACK FACTORY VISITORS USING OPENVINO

by Wong Yiek Heng  
Supervisor: Mr. Teoh Shen Khang




### Introduction

- Improve the speed and accuracy from traditional neural network model
- Protect the privacy of factory visitors and employees
- Advancing Industry 4.0



### Objectives

- Create a Face Registration System
- Create a Face Reidentification System
- Create connection between Face Registration & Face Reidentification




### Description & Methodology

**START**

Capture and Register Face on Face Register System and register the face


Compare the Register Face in Cosine Distance and calculate the most similar face

Real-time reidentification of visitors on IoT devices



### Discussion & Conclusion

This system is an improvement of existing system. Almost all IoT devices cannot handle the traditional inference well because lack of powerful Graphical Processing Unit. A new technology, OpenVino, is applied to speed up the inference and make it workable on IoT devices to perform the real-time face reidentification on smart factory which greatly reduce the cost compare to the traditional neural network system.



Faculty of Information and Communication Technology  
BACHELOR OF INFORMATION TECHNOLOGY (HONS) COMPUTER ENGINEERING

## Plagiarism check result

### Face Reidentification System to Track Factory Visitors using OpenVINO

#### ORIGINALITY REPORT

<b>14%</b>	<b>11%</b>	<b>6%</b>	<b>9%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

#### PRIMARY SOURCES

<b>1</b>	<b>eprints.utar.edu.my</b> Internet Source	<b>1%</b>
<b>2</b>	<b>www.zdnet.com</b> Internet Source	<b>1%</b>
<b>3</b>	<b>medium.com</b> Internet Source	<b>1%</b>
<b>4</b>	<b>ijifr.com</b> Internet Source	<b>&lt;1%</b>
<b>5</b>	<b>Submitted to University of California, Los Angeles</b> Student Paper	<b>&lt;1%</b>
<b>6</b>	<b>www.teaminindia.com</b> Internet Source	<b>&lt;1%</b>
<b>7</b>	<b>lucidworks.com</b> Internet Source	<b>&lt;1%</b>
<b>8</b>	<b>www.groundai.com</b> Internet Source	<b>&lt;1%</b>

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	WONG YIEK HENG
<b>ID Number(s)</b>	17ACB06611
<b>Programme / Course</b>	CT
<b>Title of Final Year Project</b>	Face Reidentification System to Track Factory Visitors using OpenVINO

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>14</u> %</b>  <b>Similarity by source</b> Internet Sources: <u>11</u> % Publications: <u>6</u> % Student Papers: <u>9</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>0</u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

*Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.*

*7E04*

\_\_\_\_\_  
Signature of Supervisor

Name: Teoh Shen Khang

Date: 10 September 2020

\_\_\_\_\_  
Signature of Co-Supervisor

Name: \_\_\_\_\_

Date: \_\_\_\_\_



## UNIVERSITI TUNKU ABDUL RAHMAN

### FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

#### CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB06611
Student Name	WONG YIEK HENG
Supervisor Name	Mr. TEOH SHEN KHANG

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

\*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;"> </div> <p>(Signature of Student) Date: 4/9/2020</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <div style="text-align: center;"> </div> <p>(Signature of Supervisor) Date: 10 September 2020</p>
---	--