

PASSENGER COUNTING WITH FACE DETECTION

TEOH HAN WEI


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Mechatronics Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

January 2020

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : TEOH HAN WEI

ID No. : 15UEB05028

Date : 27th APRIL 2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**PASSENGER COUNTING WITH FACE DETECTION**” is prepared by **TEOH HAN WEI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature :



Supervisor :

IR. TS. DR. THAM MAU LUEN

Date :

27th APRIL 2020

Signature :



Co-Supervisor :

DR. CHEE PEI SONG

Date :

27th APRIL 2020

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2020, TEOH HAN WEI. All right reserved.

ACKNOWLEDGEMENTS

I will like to thank everyone who had contributed to the successful completion of this project. I will like to express my gratitude to my research supervisor, Ir. Ts. Dr. Tham Mau Luen for his invaluable advice, guidance and his enormous patience throughout the development of the research.

Besides, I cannot express enough thanks to UTAR lecturers, lab officers and staffs especially my co-supervisor, Dr. Chee Pei Song for their continued support, encouragement and assistance.

In addition, I will also like to express my gratitude to my loving parents and friends who had helped and given me encouragement and mentally support.

ABSTRACT

Passenger counting exhibits a wide variety of applications in the context of smart cities. Such applications range from retail analytics, queue management and space utilizations. Driven by the success of machine learning, this study aims to develop a real-time passenger counting system. Different from existing works, the designed solution is deployed on a resource limited Intel UP Squared (UP²) Board, inference of which is handled by an accelerator called Intel Movidius Neural Compute Stick 2 (NCS2). MobileNet-single shot detector (SSD) is chosen as the object detector model since it belongs to a class of efficient models that can execute on mobile and embedded systems. While running detections across every video frames, centroid tracking algorithm tracks every unique person in video streams, where Kalman filter is further applied to reduce the noise. The outcome can be visualized on different type of devices for further analysis through a central cloud server. The performance of the passenger counting system is evaluated in terms of accuracy and frame per second (FPS). Furthermore, the feasibility of the solution is demonstrated in several showcases.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF SYMBOLS / ABBREVIATIONS	xiii
LIST OF APPENDICES	xiv

CHAPTER

1	INTRODUCTION	1
1.1	General Introduction	1
1.2	Importance of the Study	1
1.3	Problem Statement	1
1.4	Aim and Objectives	2
1.5	Scope and Limitation of the Study	3
1.6	Contribution of the Study	3
1.7	Outline of the Report	4
 2	 LITERATURE REVIEW	 5
2.1	Introduction	5
2.2	Convolutional Neural Network (ConvNet/CNN)	5
2.2.1	Input Layer	6
2.2.2	Convolution Layer	7
2.3	Face Detection Methods	8
2.3.1	Knowledge-based Method	8

	2.3.2	Feature Invariant Approach	9
	2.3.3	Template Matching Method	10
	2.3.4	Appearance-based Method	10
2.4		Person Counting	11
	2.4.1	Camera Orientation	12
	2.4.2	Density of People	13
	2.4.3	Lighting Condition	13
	2.4.4	Presence of Occlusion	14
3		METHODOLOGY AND WORK PLAN	15
3.1		Introduction	15
	3.1.1	Prototype Implementation	16
3.2		Hardware Overview	18
	3.2.1	Intel UP Squared Board	18
3.3		Software Implementation	20
	3.3.1	OpenVINO	21
3.4		Face Recognising and Tracking	21
	3.4.1	Centroid Tracking Algorithm	22
	3.4.2	Kalman Filter	24
4		RESULTS AND DISCUSSIONS	26
4.1		Performance Measurement for Different Devices	26
	4.1.1	Tabulated Results of Performance Measurement for Different Devices	27
4.2		Performance Measurement for Different Input	31
	4.2.1	Performance Measurement for Different Revolutions Input	31
	4.2.2	Performance Measurement for Different Inputs Complexity	32
	4.2.3	Performance Measurement for Different Performances Complexity	32
4.3		Technical Challenges	33
	4.3.1	Camera View Calibration	33

4.3.2	Pedestrian Counting Method	34
4.3.3	Double Counting Problem	35
4.3.4	Counting Method	36
5	CONCLUSIONS AND RECOMMENDATIONS	38
5.1	Conclusions	38
5.2	Recommendations for future work	39
	REFERENCES	40
	APPENDICES	42

LIST OF TABLES

Table 3.1: Data Sheet of Intel UP ² Board	19
Table 4.1: Input Power and Output Power of UP ² Board and PC	27
Table 4.2: Results from UP ² Board with one MYRIAD	28
Table 4.3: Results from PC with one MYRIAD	28
Table 4.4: Results from UP ² Board with two MYRIAD	28
Table 4.5: Results from PC with two MYRIAD	29
Table 4.6: Comparison of Specification between UP ² and PC	29
Table 4.7: Comparison of Results between UP ² and PC for one MYRIAD	29
Table 4.8: Comparison of Results between UP ² and PC for two MYRIAD	30
Table 4.9: Comparison of Results between one and two MYRIAD for both UP ² and PC	30

LIST OF FIGURES

Figure 1.1: Example of dashboard	2
Figure 2.1: Subsets of artificial intelligence (AI)	5
Figure 2.2: Process of convolutional neural network (CNN)	6
Figure 2.3: 4x4x3 RGB plane	7
Figure 2.4: Convolution layer	7
Figure 2.5: Face Detection Methods	8
Figure 2.6: Knowledge-based Method	9
Figure 2.7: Feature Invariant Approach	9
Figure 2.8: Template Matching Method	10
Figure 2.9: Appearance-based Method	11
Figure 2.10: Person Counting	12
Figure 2.11: Different Camera View from Different Camera Orientation	12
Figure 2.12: High Density of People	13
Figure 2.13: Lighting Condition on Images' Quality	14
Figure 2.14: Presence of Occlusion	14
Figure 3.1: Flow chart of prototype	15
Figure 3.2: The Prototype	16
Figure 3.3: System Flow of Prototype	17
Figure 3.4: Flow Chart for System Flow	17
Figure 3.5: Communication of electronics components with microcomputer	18
Figure 3.6: Actual Model of Intel UP ² Board	20
Figure 3.7: Communication of software with microcomputer	20

Figure 3.8: Flow of face recognition, tracking and counting processes	21
Figure 3.9: Sample actual model of the passenger counting	22
Figure 3.10: Bounding Box with Unique ID	23
Figure 3.11: Euclidean Distance	23
Figure 3.12: Euclidean Distance for Two Points	24
Figure 3.13: Kalman Equation	24
Figure 3.14: Movement Path Prediction from Kalman Filter	25
Figure 3.15: Kalman Filter with Centroid Tracking Algorithm	25
Figure 4.1: Specification of Dell Inspiron 14-3443	27
Figure 4.2: Performance Measurement for Different Revolutions Input	31
Figure 4.3: Performance Measurement for Different Inputs Complexity	32
Figure 4.4: Performance Measurement for Different Performances Complexity	33
Figure 4.5: Changes of Camera View Calibration	34
Figure 4.6: Changes of Pedestrian Counting Method	35
Figure 4.7: Double Counting Problem	36
Figure 4.8: Changes of Counting Methods	37

LIST OF SYMBOLS / ABBREVIATIONS

5G	Fifth Generation Cellular Network Technology
AI	Artificial Intelligent
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
fps	frames per second
GPU	Graphics processing unit
Industry 4.0	Industry Revolution 4.0
IoT	Internet of Things
NCS2	Neural Compute Stick 2
OpenCV	Open Source Computer Vision Library
OpenVINO	Open Visual Inference and Neural Network Optimization

LIST OF APPENDICES

APPENDIX A: Datasheet of Intel UP Squared Board	42
APPENDIX B: Datasheet of Movidius Myriad VPU	44
APPENDIX C: Datasheet of AI Core X (Processor Chipset for Movidius)	46
APPENDIX D: Datasheet of UP HD Camera	47
APPENDIX E: main.py	49
APPENDIX F: CloudDataTransmitter.py	63
APPENDIX G: firebasecloud.py	66
APPENDIX H: dashingrequest.py	69

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Today, passenger counting has a capacious of applications and it is often developed on surveys, management and security. Passenger counting has the capability to provide a high accuracy and large amount of data for big data analysis, behaviour and management study. For instance, passenger counting is implemented in public transport system, where the management able to collect and study the data from this system. The management is then able to make a more precise decision on optimizing the public transport system.

Nowadays, technology trend grows dynamically due to the emerging trends of Industry 4.0 and 5G technology. Industry 4.0 was proposed to send all the data to the cloud while 5G technology will assist this idea and make it comes true. In the cloud, the huge amount of data will be collected and big data analysis will be done to collect the useful analysis results to the end users.

1.2 Importance of the Study

The result of this present study might have significant impact on solving the limitation of current face recognition technology by improving the flexibility, functionality and performance of this technology. The prototype from this study is able to embed, implement and solve multiple problems with one single solution.

The key concern of this study is to catch up the face recognition technology with the current technology tends such as Internet of Things (IoT), 5G, industry revolution 4.0 (Industry 4.0) and the big data analysis. In the coming next few years, the market will be analysed and studied by extracting the auxiliary information based on the biometric techniques especially the face recognition which has been wisely implemented in China. (Kumar, et al., 2017)

1.3 Problem Statement

Nowadays, passenger counting has been done using different type of algorithms and methods. The most common way is to use OpenCV to recognise the moving pixels in the frame. However, this method has a critical weakness where it might miss track and

count passengers if few passengers passed together or passed by each other. The best solution is to implement a deep learning algorithm which makes the system “smarter”. Furthermore, the passenger counting that is available in the open source shares some common weaknesses in terms of poor accuracy, limited features, limited application, lack of flexibility and high power consumption. These algorithms are easy to miscount when overlapping or multiple passing. Furthermore, the output data from these methods is traditionally stored or direct display without storing to the cloud.

In order to catch up with the technology trends for Industry 4.0 and 5G technology, the output data from the system has to share to the cloud where end users from every end corners of the Earth are able to access to it. By sharing the data to cloud, the end users are able to access the dashboard from far and receive the important plus simplified message from it. Figure 1.1 shows the example of dashboard.



Figure 1.1: Example of dashboard

1.4 Aim and Objectives

The aim of this project is to develop a passenger counting with face detection by using Intel UP Squared Board and Intel Movidius Neural Compute Stick 2 (NCS2). These components are embedded with the software coding in order to build a prototype that is able to accomplish the objectives of this project:

- To develop a video analytics with machine learning capability that performs the face detection technique and tracking of individual in real time.

- To integrate the technique with Internet of Things (IoT), industry revolution 4.0 (industry 4.0) and the big data analysis
- To create a cloud based system for the visualization of real time data analysis.

1.5 Scope and Limitation of the Study

This project is divided into few stages which are input data collection at frontend sensor, data processing and data analysis at central processing unit (CPU), data synchronization from the CPU to cloud or data storage and data dissemination to the end users. The challenge of this project is to combine all the stages and performed them in a single device.

In the data collection stage, the frames per second (fps) is the main challenge of this project. The communication speed between the sensor with the single board computer and image resolution of the sensor will be the barrier for this project. The pre-trained data sets, quantity of data and lack of suitable data sets would affect the output results from data processing and data analysis. (Fu, 2019)

The specification of the single board computer, transmission speed of Wi-Fi module and communication speed of the transmission system are important in this project where the response speed, training and inference time depend heavily on these factors and it might lead to the quality of the performance and results collected. (Kim, et al., 2019) Furthermore, the total power consumption required for this project is high as it might lead to few factors like heating problem, power source limitation, lifetime and costing.

The algorithms and platforms used for face recognition, face tracking and person counting are important for the accuracy, precision and recall of the results obtained. The parameters of algorithms and tracking method are tuned and adjusted to the optimum performance while at the same time the failure detection scheme was installed and implemented to recover from tracking failure. (Wang, et al., 2019)

1.6 Contribution of the Study

The contribution of this project might help to improve, enhance and combine the technologies from face recognition, face tracking and person counting into a single prototype. This system would build by available components, small in size, low power consumption and cheap in costing which is able to catch up with the technology and industry trends. Besides, the prototype from this project is a good solution and flexible

for many applications. For example, it can be implemented in attendance system, payments, access and security.

1.7 Outline of the Report

In this report, literature review will be discussed and commented in Chapter 2. Besides, the methodology involves fulfilling the aim and objectives in this study will be explained in Chapter 3. All the results and explanations will be discussed in Chapter 4 while the conclusion and recommendation will be written at Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Deep learning is a type of machine learning which trained a microprocessor or central processing unit (CPU) to perform a high level jobs or humanlike tasks such as image recognition, passenger tracking and speech recognition. Deep learning will help the computer to study on its own recognizing patterns through multiple layers of processing. The divergences between the artificial intelligence (AI), machine learning and deep learning are artificial intelligence which is a technique that enabled computer to imitate humanlike behavior while machine learning is a subdivision of artificial intelligence technique which enabled machines to improve itself with experiment by using statistical method and deep learning is a subdivision of machine learning where the multi-layer neural networks feasible is made with computation process. Figure 2.1 shows the subsets of artificial intelligence.

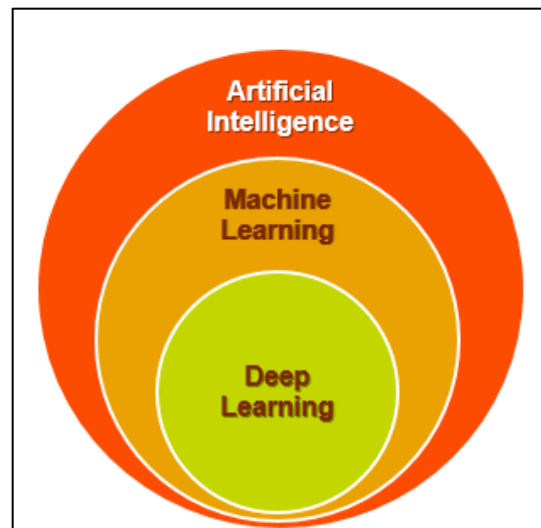


Figure 2.1: Subsets of artificial intelligence (AI)

2.2 Convolutional Neural Network (ConvNet/CNN)

In the world of deep learning, a convolutional neural network (CNN) is a class of deep learning architecture that contains various layers such as pooling layer, convolutional layer and activation layer. In CNN, it takes in an input data such as images, videos or any real time data and it will assign learnable biases and weights to the targets in the

input data and it will be able to classify and differentiate the targets. However, CNN requires fine-tuned weights and biases to obtain the best result with the help of filters, pre-processing and number of trainings. For CNN, the pre-processing required is much lower as compared to other methods of classification. (Wu, 2019) Figure 2.2 shows the sequence of CNN from the input raw data until the output result. (Saha, 2018)

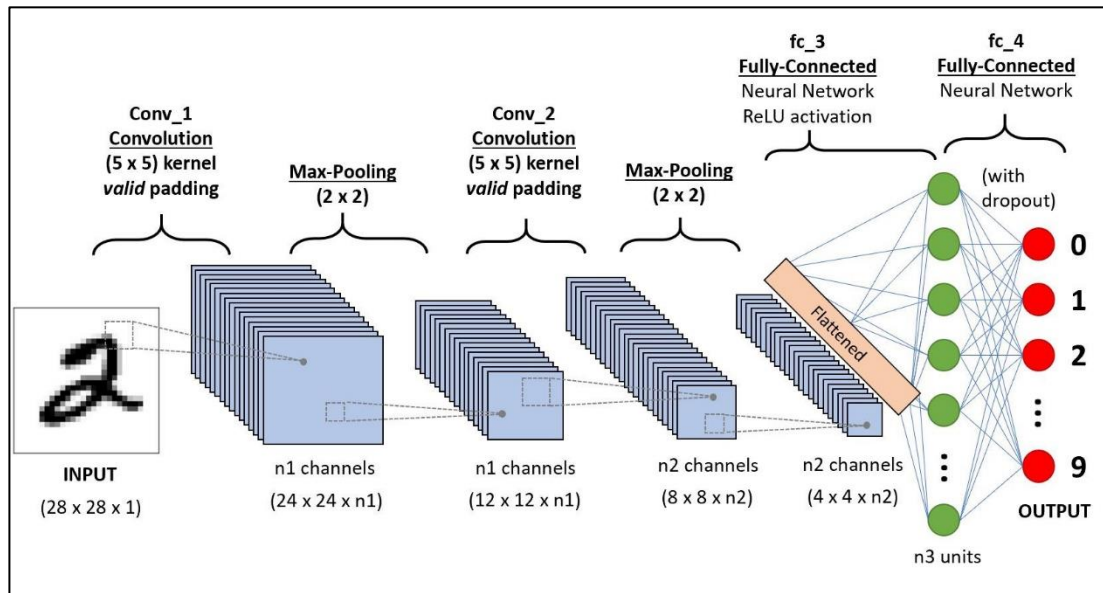


Figure 2.2: Process of convolutional neural network (CNN)

2.2.1 Input Layer

The data collected from the input stage will be processed by separating the data into three types of colour planes which are the red, green and blue planes where they are also known as the RGB plane. (Atmaja, et al., 2016) The purpose of this process is mainly on minimizing the data into a more simple form that is easier for the following processes and at the same time, all the important features from the data will be remained which is important and critical for getting a good prediction. (Prabhakar, et al., 2017) In Figure 2.3 shows the RGB plane is separated from a raw data. (Saha, 2018)

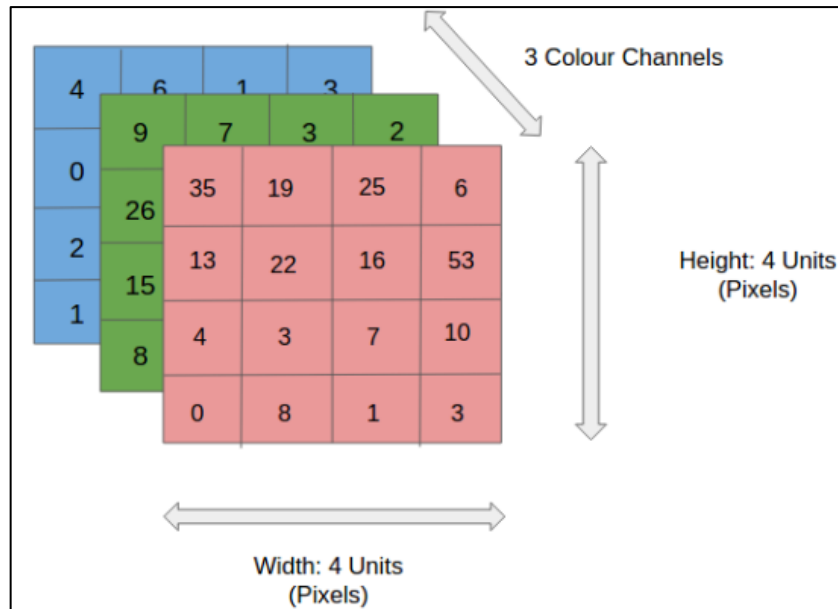


Figure 2.3: 4x4x3 RGB plane

2.2.2 Convolution Layer

Convolution layer is the major role in CNN. A convolution layer consists of a set of filters where those variables of filter are required to be trained and learned. The filters contain of weight and height where the weight and height of the filters have to be smaller than the input volume. Input volume will be convolved by each filter and formed neurons which will compute an activation map. For example, Figure 2.4 shows an example of 5x5 matrix of input data and it is convolved to a feature map. (Dertat, 2017)

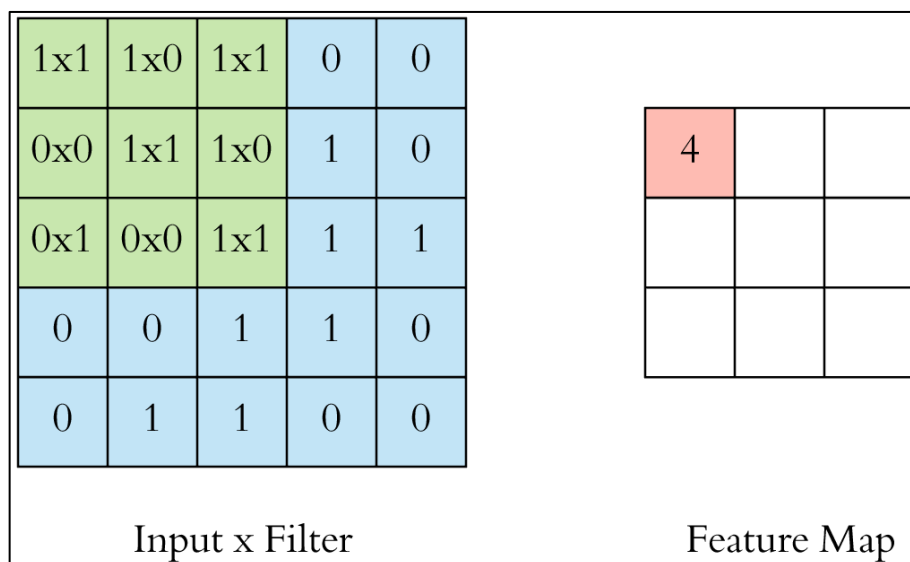


Figure 2.4: Convolution layer

2.3 Face Detection Methods

There are several existing and mature techniques used in the market to detect the faces from any single intensity or colour image. These face detection techniques can be classified into four major categories which are knowledge-based method, template matching method, appearance-based method and features invariant approach. (Qaim Mehdi Rizvi, 2011) Face detection can be consider as a substantially part of face recognition. The method of face detection is complicated due to various features presented across the human faces such as pose, skin colour, position and orientation, lighting condition and image resolution. (Chakravartula Raghavachari, 2015) Figure 2.5 shows the list of common face detection methods used.

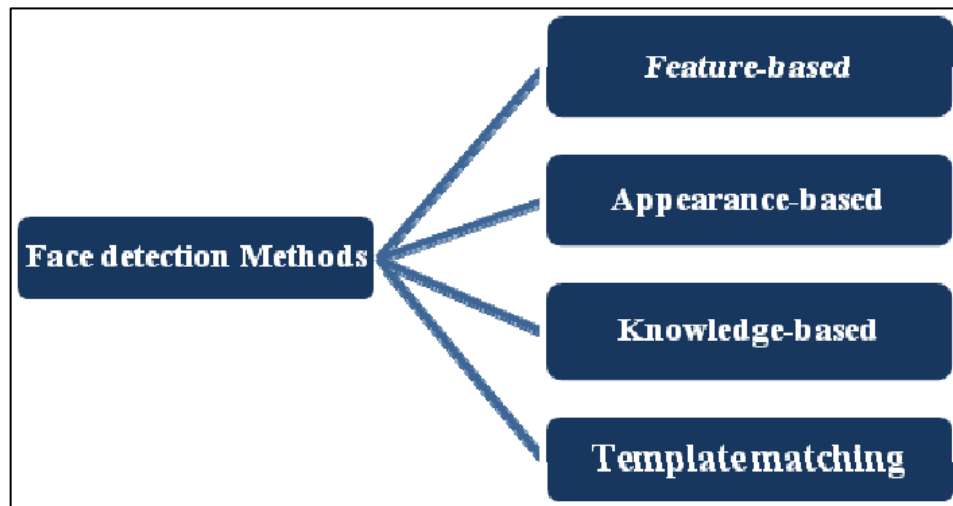


Figure 2.5: Face Detection Methods

2.3.1 Knowledge-based Method

First of foremost, knowledge-based method also known as the rule-based method where it will encode human knowledge that the features that should be constituted in a typical face. (Mayank Chauhan, 2014) Normally, these rubrics capture the relationships between facial features. This method is designed mainly for face localization. (Qaim Mehdi Rizvi, 2011) Figure 2.6 shows the knowledge-based method's architecture.

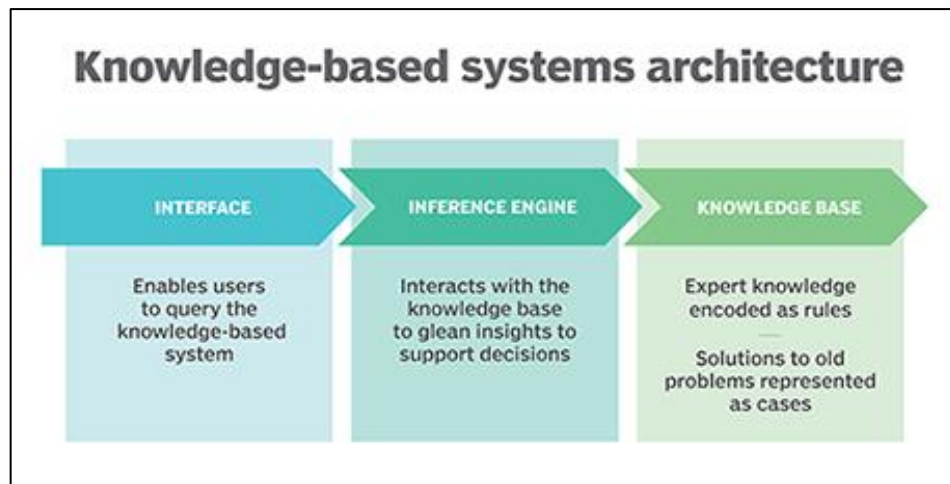


Figure 2.6: Knowledge-based Method

2.3.2 Feature Invariant Approach

For feature invariant approach, the algorithm in this approach is mainly to find for the structural features that may exist at certain position even when the pose, gestures, viewpoint or lighting conditions vary and this approach is to detect the faces. This method is designed mainly for face localization. (Qaim Mehdi Rizvi, 2011) Figure 2.7 shows the sample of feature invariant approach which found out the relative position from the random listing.

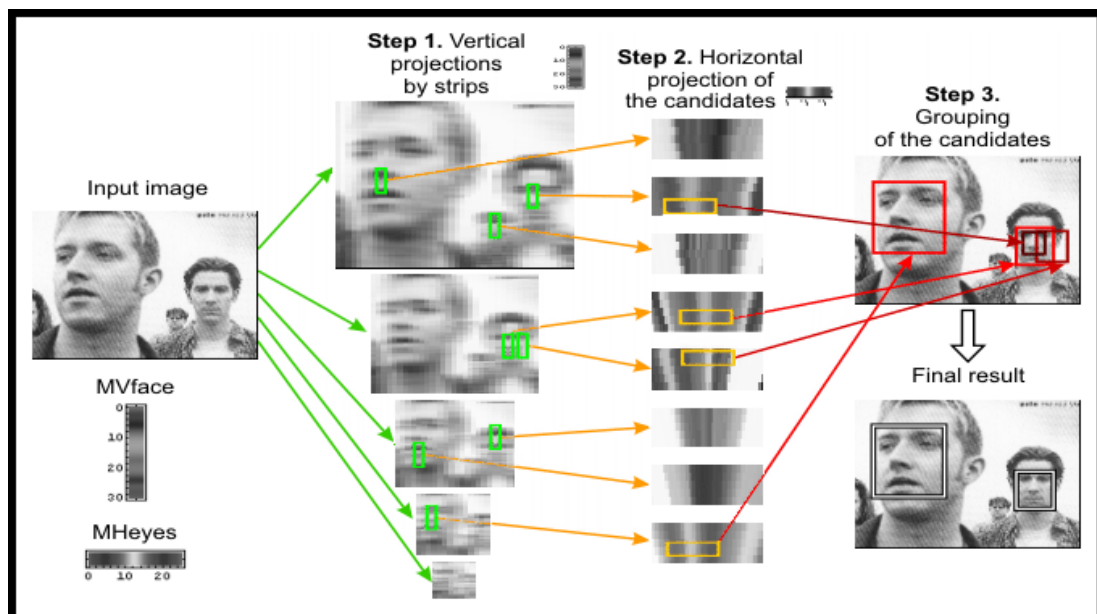


Figure 2.7: Feature Invariant Approach

2.3.3 Template Matching Method

In template matching method, a group of standard patterns of typical faces will be stored to describe the face as a whole or the facial features separately. (Priyanka R.Borude, 2015) The correlations between an input image and the stored patterns are computed for detection. This method is designed mainly for both face localization and detection. (Qaim Mehdi Rizvi, 2011) This method has been widely used in image face detection algorithm and programming platforms like Matlab, OpenCV and Python. Figure 2.8 shows the sample architecture of template matching method.

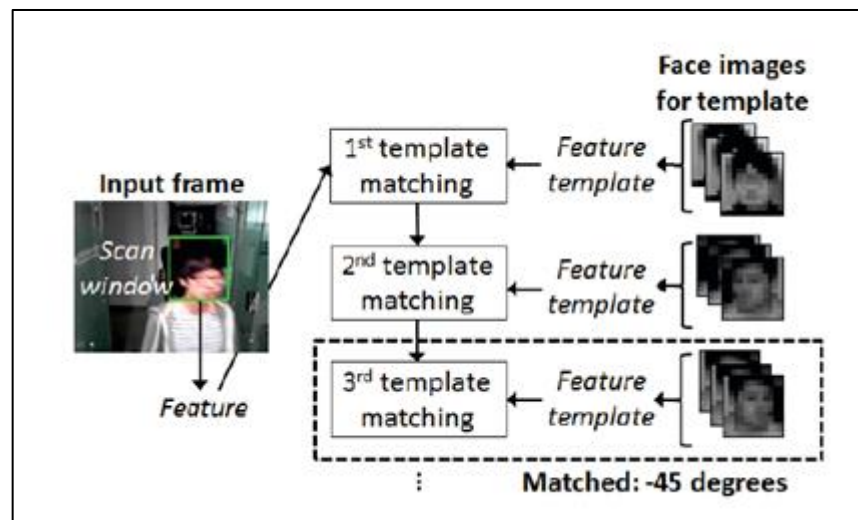


Figure 2.8: Template Matching Method

2.3.4 Appearance-based Method

In divergence to template matching method, the models are learned from a set of training images which should capture the representative variability of facial appearance. (Priyanka R.Borude, 2015) These learned models will be used for face detection. These methods are designed mainly for face detection. (Qaim Mehdi Rizvi, 2011) This method has been widely used in motion face detection algorithm and programming platforms like OpenCV and Python where this method able to track the recognized features or points when the faces are moved. Figure 2.9 shows the sample result of appearance-based method.



Figure 2.9: Appearance-based Method

2.4 Person Counting

In person counting, there are several vision based approaches that need to be consisted in different scenarios. There are several factors that may affect the performance of person counting. The factors are camera orientation, density of people, lighting conditions and presence of occlusion. (Chakravartula Raghavachari, 2015) Figure 2.10 shows sample person counter using OpenCV.



Figure 2.10: Person Counting

2.4.1 Camera Orientation

The camera can be installed either in an overhead position or in an inclined position. The camera installs in an overhead position will have a better result of person counting due to the occlusion noise caused by person can be reduced as the camera had a better view of the people arrangement. However, the camera installs in an inclined position had a better view and enables to extract the human features well. (Chakravartula Raghavachari, 2015) Figure 2.11 shows the camera view from overhead position and inclined position.

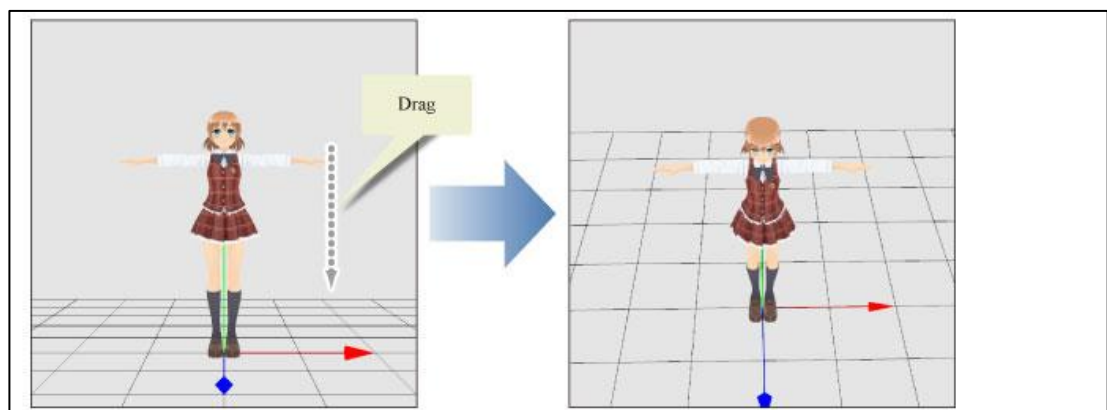


Figure 2.11: Different Camera View from Different Camera Orientation

2.4.2 Density of People

The person counting is highly effected by the human detection accuracy and the human detection accuracy is highly effected the density of people in the detection region or coverage region of camera. The person counting has a better result when the density of people in low. This is due to the high density of people, the people are moving close to each other and the noise like occlusion may happen and effect the final result. (Chakravartula Raghavachari, 2015) Figure 2.12 shows the camera view from a high density of people where the people stood closer to the camera can be viewed but the people behind are blocked.



Figure 2.12: High Density of People

2.4.3 Lighting Condition

Lighting condition and the environment condition are highly effected the vision based algorithms. The person counting results are robust to the low lighting condition. The accuracy under day light condition is less lower that the night mode which may due to the presence of shadow in day light condition. This noise is effected the accuracy of human detection which effects the person counting results as well. (Chakravartula Raghavachari, 2015) Figure 2.13 shows the level of lighting condition and the changes on the quality of image taken.

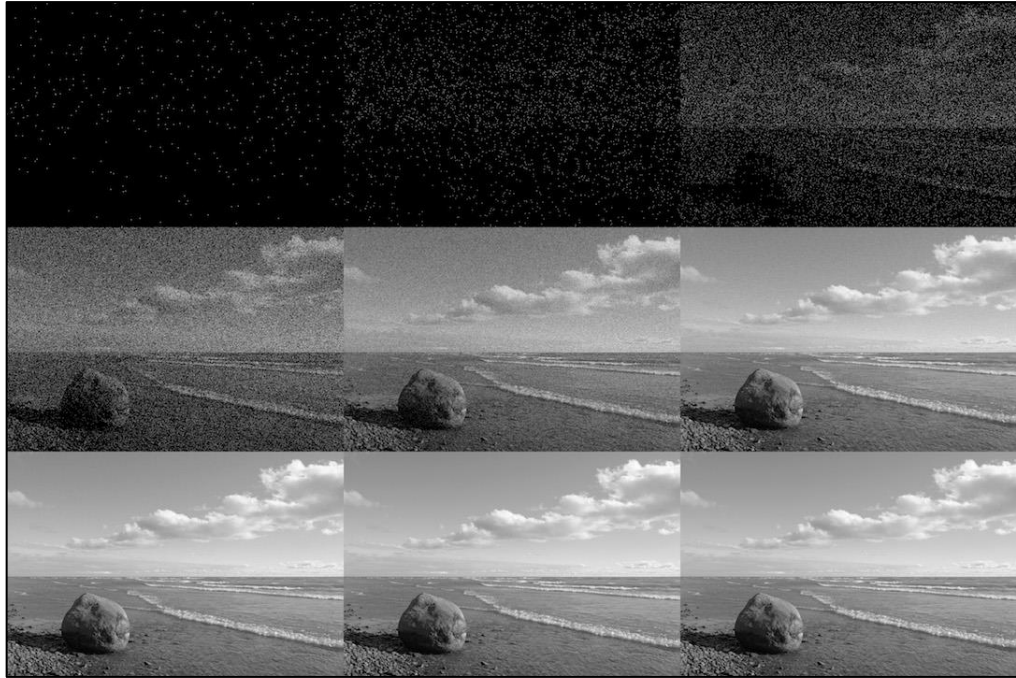


Figure 2.13: Lighting Condition on Images' Quality

2.4.4 Presence of Occlusion

In camera view, the detected person is highly sensitive to the obstacles around him. For instance, someone passes by him or he passes by some objects and the objects blocked him. The effectiveness of the person counting results is depended on the number of occlusions occurred. (Chakravartula Raghavachari, 2015) Figure 2.14 shows the effect of occlusion on person tracking.

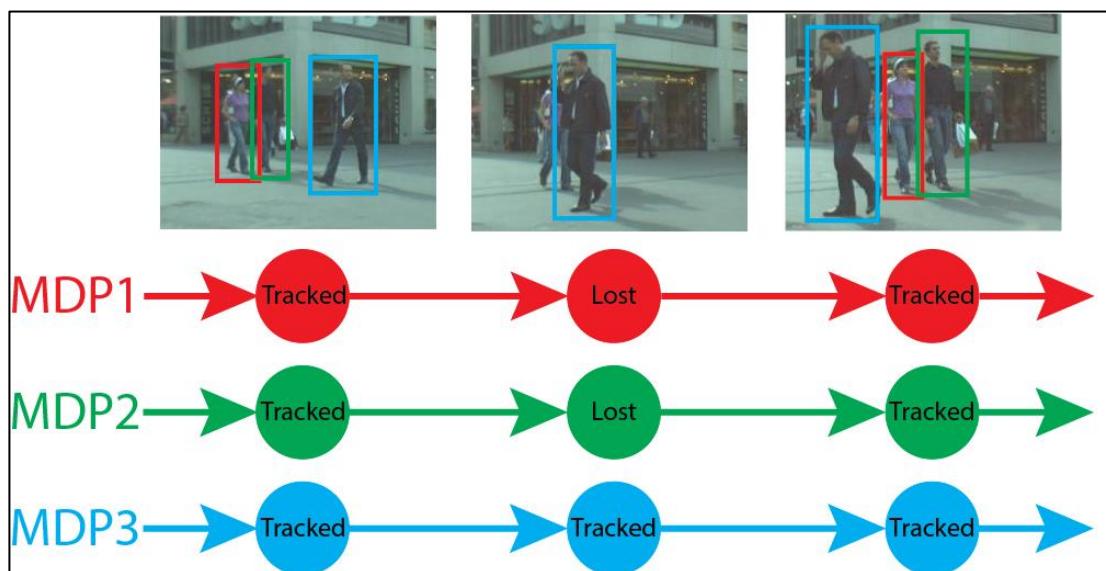


Figure 2.14: Presence of Occlusion

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this project, the prototype is made up of three major stages which are input stage, data processing stage and output stage. In input stage, the sensor used is the UP HD camera which to collect the real time video analysis as input data. In the processing stage, the input data collected will be processed by the microcomputer with the support of Intel Movidius Neural Compute Stick 2 (NCS2), assistance modules and software algorithms implementation. The last stage is the output stage where the output result from the previous stage will be collected and stored in cloud storage, displayed to the end user directly or extracted by the end user when needed. Figure 3.1 shows the flow chart of the prototype.

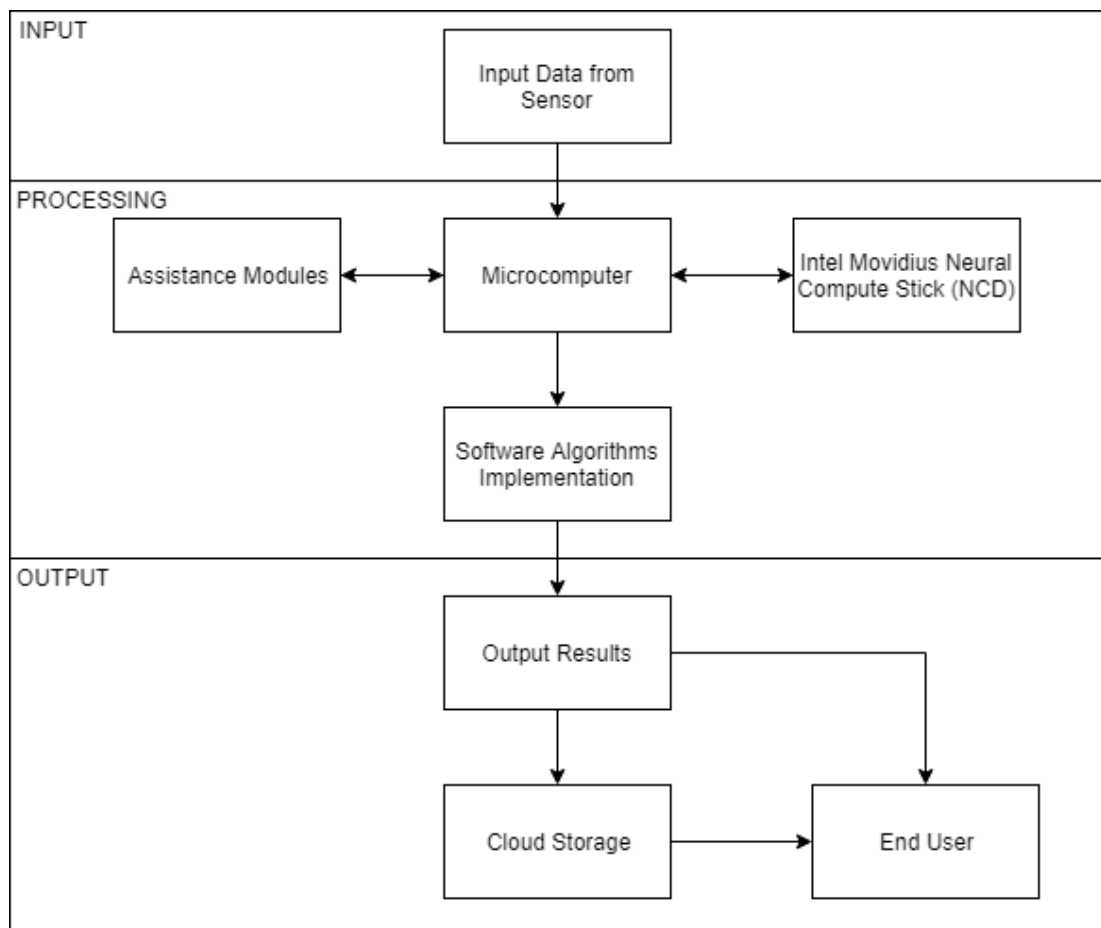


Figure 3.1: Flow chart of prototype

3.1.1 Prototype Implementation

The prototype of this project is made from Intel UP Squared (UP²) Board, UP HD camera and Intel Movidius Neural Compute Stick 2 (NCS2) where is showed in Figure 3.2.



Figure 3.2: The Prototype

This prototype is consisting of Intel UP Squared (UP²) Board, UP HD camera and Intel Movidius Neural Compute Stick 2 (NCS2) in term of hardware. It is consisting Python 3.6, OpenCV, Linux Operating System (OS) and OpenVINO in term of software. The prototype will send the results collected to central cloud server which is the Firebase. The results will be displayed to the users through the monitor screen, dashboard, phone APP (Android and IOS). Figure 3.3 shows the complete system flow of the prototype and Figure 3.4 shows the complete flow chart for the system flow.

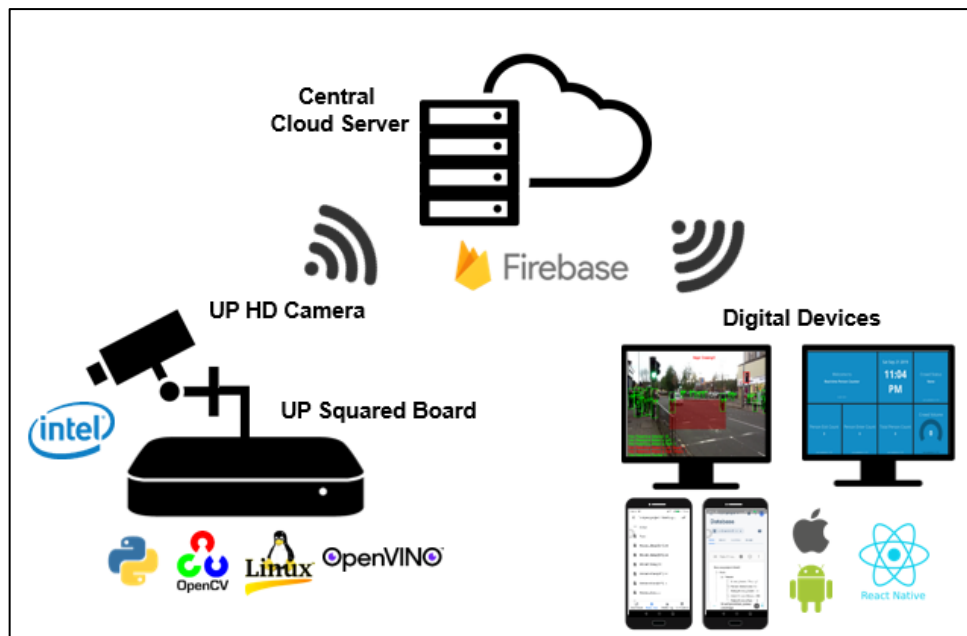


Figure 3.3: System Flow of Prototype

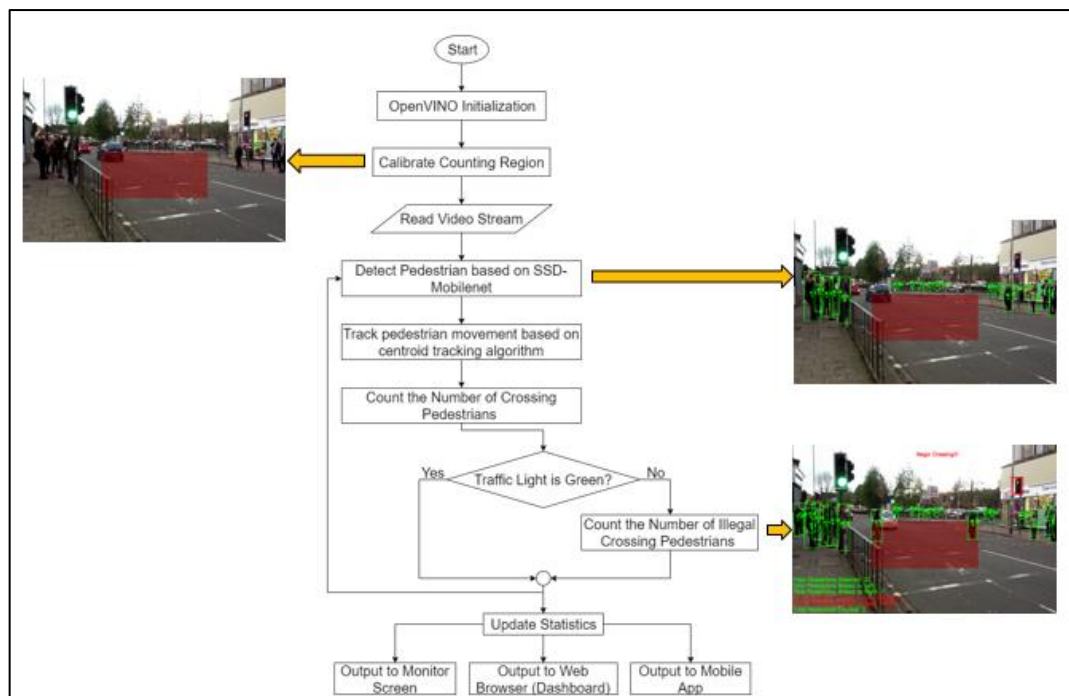


Figure 3.4: Flow Chart for System Flow

3.2 Hardware Overview

In this project, a prototype is built with the capability to perform deep learning and network monitoring tasks, several electronics hardware and microcomputer that required to fulfil the objectives of this project. The microcomputer used is the Intel UP Squared (UP²) Board and it will communicate with the UP HD camera where the camera will act as sensor to collect the input data.

Due to the limitation of the Intel UP² Board, a mSATA SSD module and M2 Wi-Fi module are added to assist the Intel UP² as this board did not had a build in Wi-Fi module and the build in memory storage is limited which needs an external memory storage to support it. Besides, the Intel Movidius Neural Compute Stick 2 (NCS2) is used to support the Intel UP² Board as well which will increase the deep learning performance up to seven times. Figure 3.5 shows relationship between the electronics components with the microcomputer.

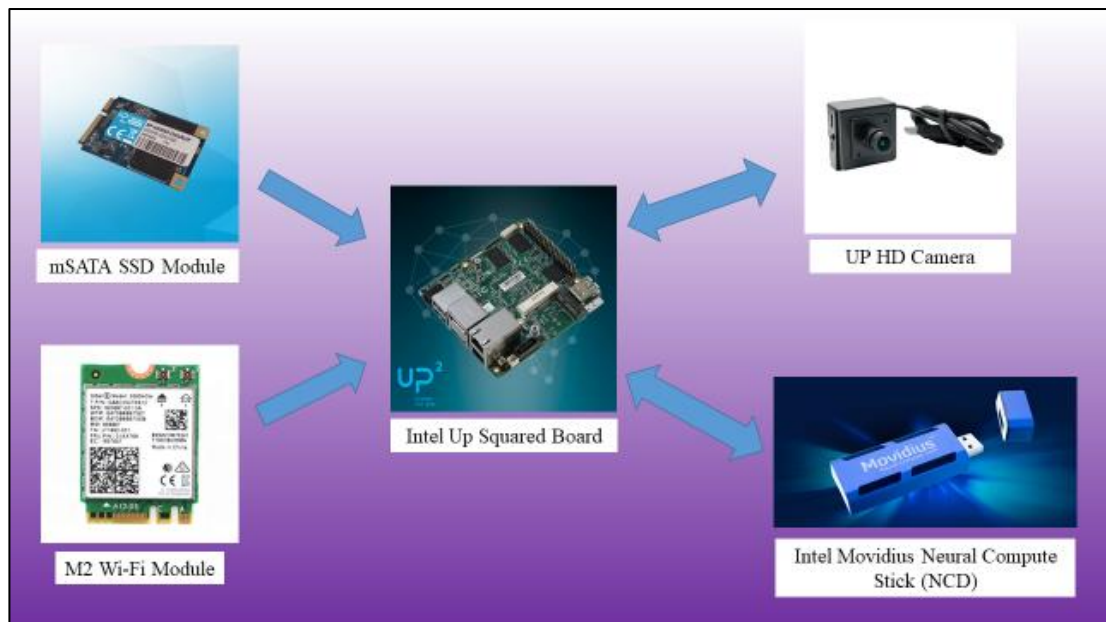


Figure 3.5: Communication of electronics components with microcomputer

3.2.1 Intel UP Squared Board

Intel UP Squared (UP²) Board is a single board microcomputer or also known as edge computer which is developed by Intel Corporation (Intel). This board can function like a mini computer and poared by 5V at 6A which had a dimension of 85.6mm x 90.0mm x 50.0mm (3.37" x 3.54" x 1.97"). Intel claimed that this board is the world's faster maker board with a high performance and low power consumption features of Intel's

Apollo lake processors. The 40 pins I/O connector, three USB 3.0 ports with one OTG, two gigabit Ethernet and more other powerful features made this board a perfect choice for different developments and applications like intelligent cars, smart city, smart home, robotics, machine vision, drone, Internet of Things (IoT) and deep learning.

This board supports the AI Core X mPCIe module and Intel Movidius Neural Compute Stick 2 (NCS2) which may speed up to 105 fps and 1 trillion floating point operations that made the a powerful and smooth solution for real time performances and visual inspections with deep learning. In Table 3.1 shows the data sheet of Intel UP² Board and Figure 3.6 shows the actual model of Intel UP² Board that is used for the prototype in this project.

Table 3.1: Data Sheet of Intel UP² Board

Features	Descriptions
UP board version	UP Squared
Graphics	Intel HD Graphic 505
System memory	4GB RAM
Storage capacity	64GB eMMC
WOL	YES
Video output	HDMI+DP
CEC	Optional
RTC	YES
PXE	YES
mPCI-e	x1
M2 2230 E-key	x1
SATA	x1



Figure 3.6: Actual Model of Intel UP² Board

3.3 Software Implementation

In this project, a prototype is built with the capability to perform deep learning and network monitoring tasks, several electronics hardware and microcomputer that required to fulfil the objectives of this project. The software environment used is the Ubuntu 18.04 LTS operating system and the software installed is the OpenVINO while OpenCV and TensorFlow are embedded in the OpenVINO where they are the fundamental structure of the OpenVINO. Figure 3.7 shows communication of the software with the microcomputer.

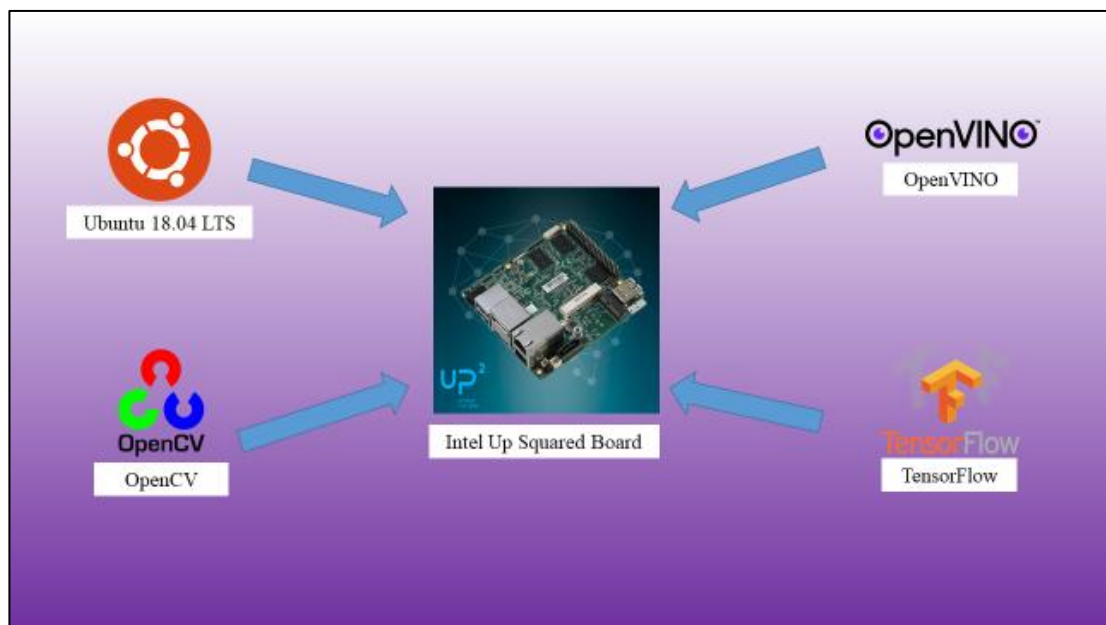


Figure 3.7: Communication of software with microcomputer

3.3.1 OpenVINO

Open Visual Inference and Neural Network Optimization (OpenVINO) is a toolkit that introduced by Intel which is used to perform an optimized deep learning. OpenVINO is a toolkit that build from the fundamental of OpenCV, in order words OpenVINO is the update version of the OpenCV. Eventually, OpenCV does not equipped with the deep learning feature and Intel optimized the OpenCV by combining the free and open-source software library, TensorFlow and OpenVINO are the product formed.

Intel claims that OpenVINO able to accelerate the deep learning performance by accessing the Intel computer vision accelerators, speed code performance and it also supports heterogeneous processing with asynchronous execution. Intel tries to run with and without OpenVINO under an unleash convolutional neural network (CNN) based deep learning inference across using a common API & ~10 pre-trained models and the result shows that OpenVINO able to integrates the deep learning process seven times faster.

3.4 Face Recognising and Tracking

At the beginning stage, the face recognition technique will detect the human face in the every frames captured with the help of deep learning. Next, the system will track the face detected and count the number of passengers when they pass through a certain area. Figure 3.8 shows the flow of face recognition, tracking and counting processes and Figure 3.9 shows the sample actual model of the passenger counting.

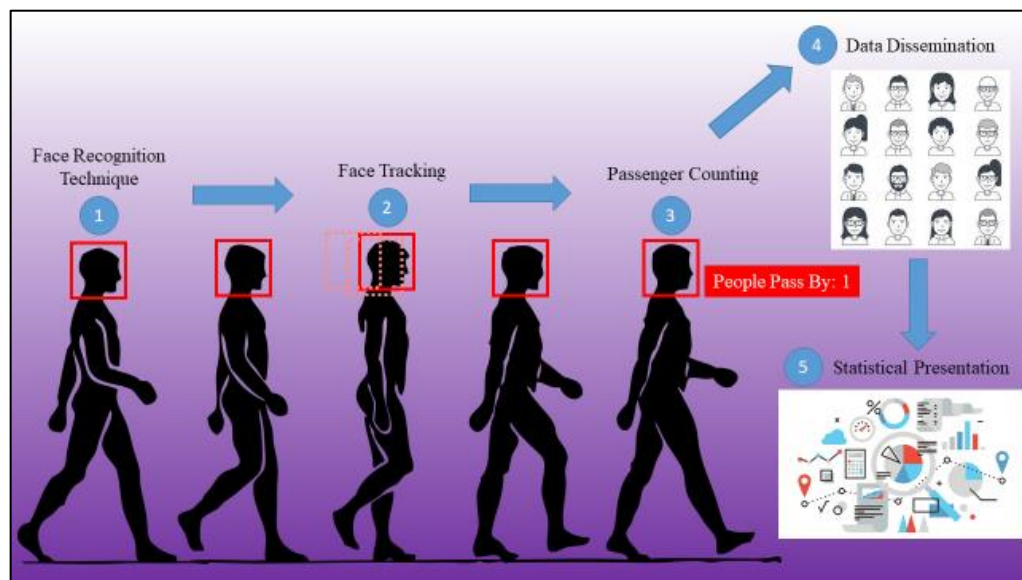


Figure 3.8: Flow of face recognition, tracking and counting processes



Figure 3.9: Sample actual model of the passenger counting

3.4.1 Centroid Tracking Algorithm

At the beginning state of centroid tracking algorithm, the algorithm will detect the human through the convolution neural network (CNN). The algorithm will initial a bounding box which will fit the entire human gesture into the region. Next, the algorithm will try to calculate the centroid point coordinate for the bounding box plus assigning a unique ID for each bounding boxes formed. The corners bounding box will generate XY coordinate ($[x1, y1], [x1, y2], [x2, y1], [x2, y2]$) which is shows in Figure 3.10. The centroid coordinate will be generated by referring the XY coordinate (x, y) of the corners. The point x is calculated from the submission of $x1$ and $x2$ then divided by 2 while the point y is calculated from the submission of $y1$ and $y2$ then divided by 2. The algorithm will repeat this step for every new detected human and assign a unique ID for all of them.

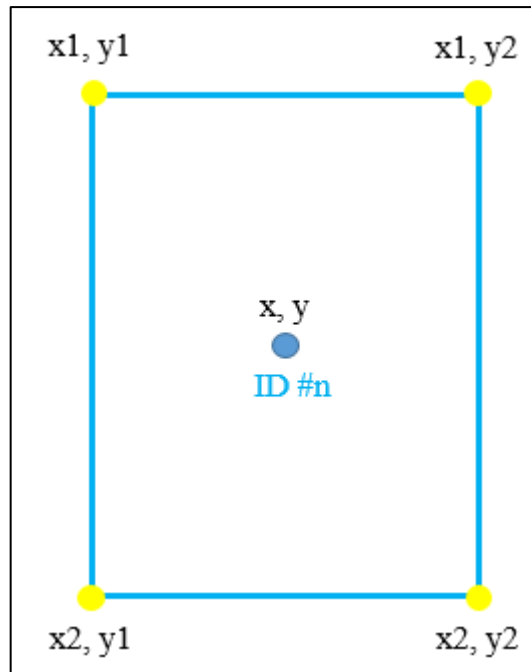


Figure 3.10: Bounding Box with Unique ID

From the centroid point generated with unique ID, the algorithm will try to track the human using the centroid point and the Euclidean distance comes in to assist the algorithm to track human movement. In Figure 3.11, the Euclidean distance, d between the initial point, $P_1 (x_1, y_1)$ and the final point, $P_2 (x_2, y_2)$ is determined with the help of formulae. The initial point is the human position at frame t and the final point is the human position at frame $t+1$ where the algorithm will get Euclidean distance to predict the human movement direction. In other way round, the human is moving from P_1 to P_2 with certain angle and direction. All the description above are concluded and shows in Figure 3.12.

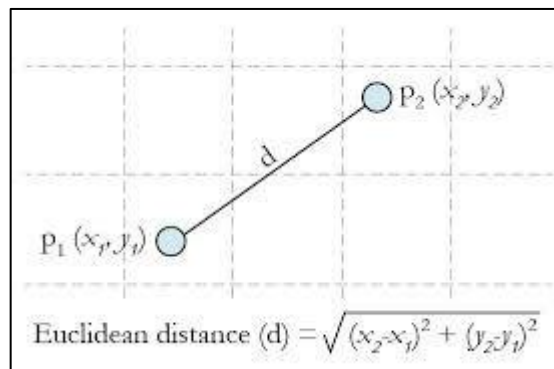


Figure 3.11: Euclidean Distance

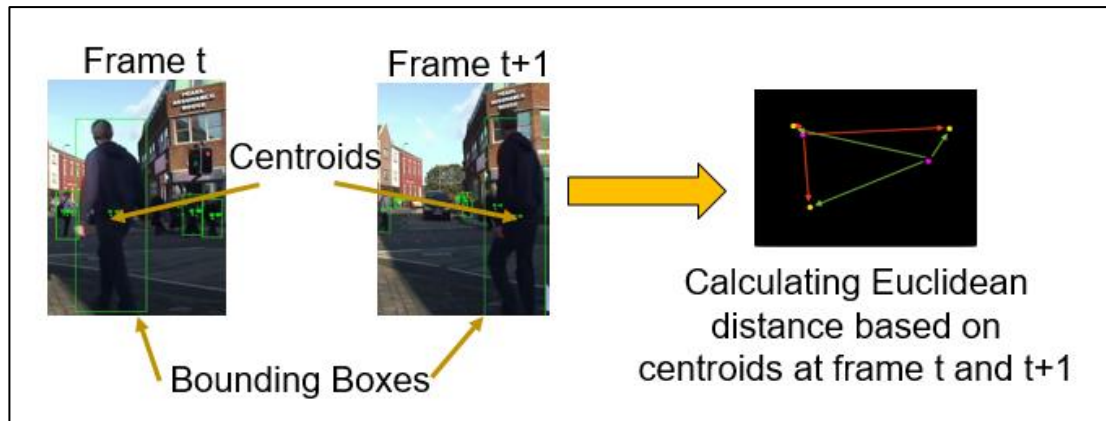


Figure 3.12: Euclidean Distance for Two Points

3.4.2 Kalman Filter

With the centroid tracking algorithm only, the algorithm does not performed well in real time tracking. During real time tracking, the centroid point formed is not constantly formed and it will miss in between. The problem causes the missing tracking of human and the algorithm will assign a new ID again for the same human again and again. For instance, person A is moving from point A to point B and in this moving path, the algorithm misses track him for 3 times due to some noises appeared. The algorithm may assign 3 new ID to him for every miss tracking where the algorithm will track him as new person after every miss track. When came to the end, this person A with an initial ID of ID 1 will end up with ID 4 after 3 miss track. In order to overcome with this problem, Kalman filter is added to assist the centroid tracking algorithm where this filter will help the algorithm to predict the possible future movement with the help of formula in Figure 3.13.

$$\hat{X}_k = K_k \cdot Z_k + (1 - K_k) \cdot \hat{X}_{k-1}$$

Labels in the diagram: current estimation (pointing to \hat{X}_k), measured value (pointing to Z_k), Kalman Gain (pointing to K_k), previous estimation (pointing to \hat{X}_{k-1}).

Figure 3.13: Kalman Equation

The Kalman filter able to predict the future movement until the human is really disappeared from the screen and Figure 3.11 shows the predicted movement path from Kalman filter with the points collected from algorithm. Besides, the Kalman filter able to eliminate the noise from the points collected by the algorithm during the movement

path. This filter plays an important role especially in real time tracking as the centroid generated may be keep fluctuating and not consistent which may act as noise for the centroid tracking algorithm and effect the accuracy of the result. With this filter, the algorithm able to come out a smooth moving path like shows in Figure 3.14 where the centroid points collect along the movement path is not consistent and fluctuating but the filter will come out the a smooth predicted movement path. Figure 3.15 shows the description and explanation above.

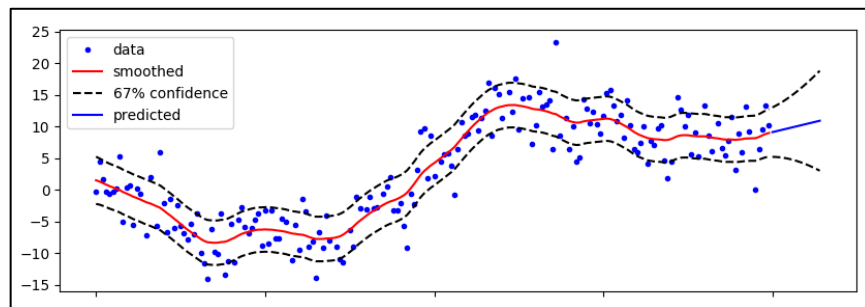


Figure 3.14: Movement Path Prediction from Kalman Filter

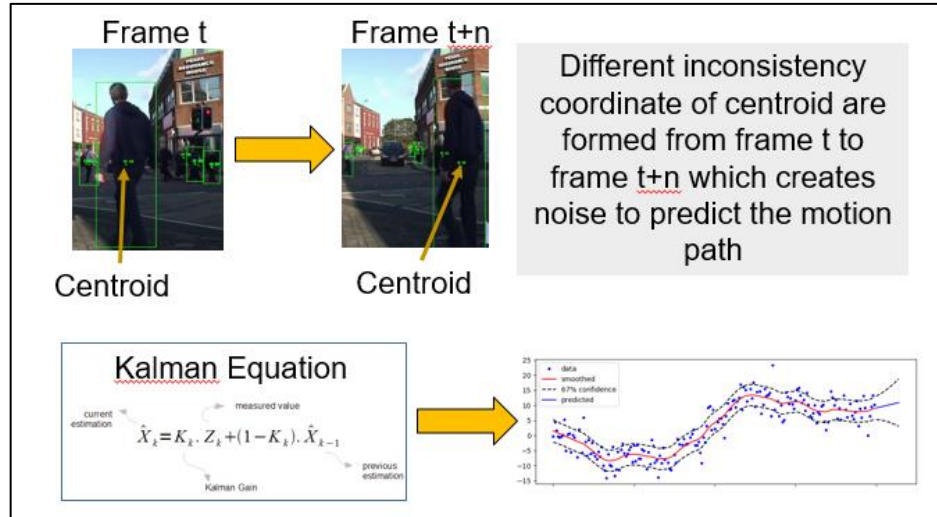


Figure 3.15: Kalman Filter with Centroid Tracking Algorithm

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Performance Measurement for Different Devices

The power specification of Intel UP Squared (UP²) Board and personal computer (PC) are recorded and tabulated in Table 4.1. The input power of devices are not focused in this study as they just effected the power consumption and it will not affect the performance. Thus, input power of devices will not be discussed but the output power of devices are highlighted as it will affect the performance in term of time needed, s and FPS. From Table 4.6, a 117% of output power increased in order to have an improvement of FPS of 42% where by increasing 1% of output power, the FPS will increase 0.36% which is around 0.01 FPS. From the improvement, a small conclusion can be made that the UP² did not have enough power to fully power up the movidius (MYRIAD). In short, the FPS performance is highly effected by the output power devices.

From Table 4.9, the results are not so positive and they did not behaviour as their expected performances. One of the problems may be due to the maximum output current of USB port which is not enough for the MYRIAD to work efficiently. A unit load is defined as 100 mA in USB 2.0 and 150 mA in USB 3.0. A device may draw a maximum of 5 unit loads (500 mA) from a port in USB 2.0 and 6 unit loads (900 mA) in USB 3.0. This is the ideal output power from USB and in real case for laptop or any electronic devices, the output power will be lower due to the power consumption in CPU and power loss. From the datasheet provided, although Intel did not mentioned the maximum current for the processor (Myriad X) in MYRIAD but an estimation can be made that it should be optimum around 1V as Intel processor worked under 1.025V and overclocked processor worked around 1.2V.

The connection of USB ports hub is another problem as well where it will share the output power among the devices connected. For instance, this hub is connected to USB 3.0 of laptop and 2 MYRIAD are connected to the hub. The 900mA from laptop will be share among the MYRIAD where each MYRIAD will receive a power of 450mA. One experiment is done by connecting two MYRIAD directly to the PC but my PC only had 2 USB 2.0 and 1 USB 3.0 and the results are not that good as well where the MYRIAD connected to USB 3.0 is heater than the MYRIAD connected to

USB 2.0. This phenomena is similar to the UP² when it is connected with two MYRIAD where one is heater than another.

The cooling step should be made as a small precaution step when the MYRIAD to high power port. As the MYRIAD will really heat up and it may spoil the MYRIAD and USB port as datasheet mentioned that MYRIAD worked under 0°C to 40°C and the USB port may spoil after running the footage under high temperature.

4.1.1 Tabulated Results of Performance Measurement for Different Devices

The PC named is Dell Inspiron 14-3443 and the specification of this PC is shows in Figure 4.1. All the testing and performance running are done under this PC with the specification listed.

TECH SPECS	
Processor	: 5th Generation Intel(R) Core(TM) i7-5500U Processor (4M Cache, up to 3.00 GHz)
Storage	: 500GB SATA 6Gb/s
Memory	: 4GB Single Channel DDR3L 1600MHz (4GBx1)
Graphics	: 2GB NVIDIA GeForce 840M DDR3
Bluetooth	: Bluetooth 4.0
Display	: 14.0-inch HD (1366 x 768) Truelife LED-Backlit Display
Camera	: 1.0MP
Wireless	: DW 1705 + BT4.0 [802.11bgn + Bluetooth 4.0, 2.4 GHz, 1x1]
Card Reader	: 3 in 1 Card Reader (SD, SDHC, SDXC)
Optical Drive	: Tray load DVD Drive (Reads and Writes to DVD/CD)
Battery	: 40 WHr, 4-Cell Battery (removable)
OS	: Windows(R) 8.1 Single Language (64Bit) English
Antivirus	: McAfee Security Center 15 month subscription
Sound Card	: High Definition Audio enhanced with Waves MaxxAudio®
Network	: Ethernet 10/100

Figure 4.1: Specification of Dell Inspiron 14-3443

Table 4.1 shows the comparison of input power and output power between the UP² Board and PC.

Table 4.1: Input Power and Output Power of UP² Board and PC

Specification	Input		Output	
	UP ²	PC	UP ²	PC
Voltage, V	240	240	5	19.5
Current, A	1.5	1.7	6	3.34

Power, W	360	408	30	65.13
-----------------	-----	-----	----	-------

Table 4.2 shows the results collected from UP² Board with one MYRIAD only.

Table 4.2: Results from UP² Board with one MYRIAD

Device:	MYRIAD			
Platform:	UP²			
	Result 1	Result 2	Result 3	Average
Enter/person	1104	1104	1104	1104
Exit/person	726	726	726	726
Time/s	26659.26	26327.63	27175.65	26720.85
FPS	3.34	3.39	3.28	3.34

Table 4.3 shows the results collected from PC with one MYRIAD only.

Table 4.3: Results from PC with one MYRIAD

Device:	MYRIAD			
Platform:	PC			
	Result 1	Result 2	Result 3	Average
Enter/person	1104	1104	1104	1104
Exit/person	726	726	726	726
Time/s	19186.38	18602.61	18539.86	18776.28
FPS	4.65	4.79	4.81	4.75

Table 4.4 shows the results collected from UP² Board with two MYRIAD.

Table 4.4: Results from UP² Board with two MYRIAD

Device:	MYRIAD, MYRIAD			
Platform:	UP²			
	Result 1	Result 2	Result 3	Average
Enter/person	1104	1104	1104	1104
Exit/person	726	726	726	726
Time/s	26365.16	27139.80	26310.89	26605.28

FPS	3.38	3.28	3.39	3.35
------------	------	------	------	-------------

Table 4.5 shows the results collected from UP² Board with two MYRIAD.

Table 4.5: Results from PC with two MYRIAD

Device:	MYRIAD, MYRIAD			
Platform:	PC			
	Result 1	Result 2	Result 3	Average
Enter/person	1104	1104	1104	1104
Exit/person	726	726	726	726
Time/s	18884.83	18658.45	18507.09	18683.46
FPS	4.72	4.78	4.82	4.77

Table 4.6 shows the comparison of specification between UP² Board and PC.

Table 4.6: Comparison of Specification between UP² and PC

Specification	Percentage Change from UP² to PC, %	Status
Input Voltage, V	0.00	Remained
Input Current, A	13.33	Increased
Input Power, W	13.33	Increased
Output Voltage, V	290.00	Increased
Output Current, A	-44.33	Decrease
Output Power, W	117.10	Increased

Table 4.7 shows the comparison of results between UP² Board and PC for one MYRIAD only.

Table 4.7: Comparison of Results between UP² and PC for one MYRIAD

Device:	MYRIAD	
Specification	Percentage Change from UP² to PC, %	Status
Enter/person	0.00	Remained
Exit/person	0.00	Remained

Time/s	-29.73	Decrease
FPS	42.36	Increased

Table 4.8 shows the comparison of results between UP² Board and PC for two MYRIAD.

Table 4.8: Comparison of Results between UP² and PC for two MYRIAD

Device:	MYRIAD, MYRIAD	
Specification	Percentage Change from UP ² to PC, %	Status
Enter/person	0.00	Remained
Exit/person	0.00	Remained
Time/s	-29.78	Decrease
FPS	42.49	Increased

Table 4.9 shows the comparison of results between one and two MYRIAD for UP² Board and PC.

Table 4.9: Comparison of Results between one and two MYRIAD for both UP² and PC

Specification	Percentage Change from MYRIAD to MYRIAD, MYRIAD, %		Status
	UP ²	PC	
Enter/person	0.00	0.00	Remained
Exit/person	0.00	0.00	Remained
Time/s	-0.43	-0.49	Decrease
FPS	0.40	0.49	Increased

4.2 Performance Measurement for Different Input

Several performance measurements are done in order to figure out the optimum conditions to fully utilise the UP² Board with the best outcome. The performance measurements are done in term of frame per second (fps), accuracy, speed, time needed and duration. The performance measurements are done under fixed conditions and during the measurements, the same devices, items and footage are used until the end of experiments.

4.2.1 Performance Measurement for Different Resolutions Input

First of foremost, the first performance measurement is on the different resolutions input. A 720p resolution footage is downgraded to 480p, 360p, 240p and 144p. The footages are ran under the same algorithm and the number of pedestrians detected by the algorithm is the measurement for this experiment. The fixed conditions in this measurement are only one movidius (NCS2) and the same UP² Board are used. From Figure 4.2, the results shows that better resolution footage has a better result and only 720p resolution footage has detected all the pedestrians in the footage.

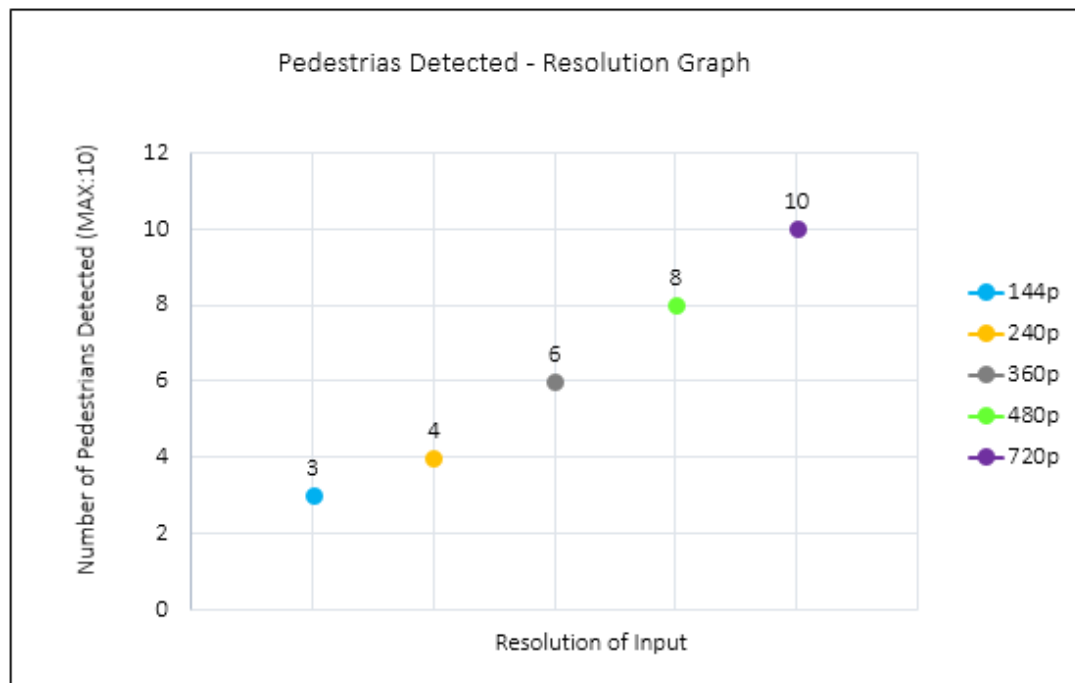


Figure 4.2: Performance Measurement for Different Resolutions Input

4.2.2 Performance Measurement for Different Inputs Complexity

Next, the second performance measurement is on the different inputs complexity. Several 720p resolution footages with different number of pedestrians in the footage are prepared. The footages are ran under the same algorithm and the frame per second (fps) is the measurement for this experiment. The fixed conditions in this measurement are only one movidius (NCS2) and the same UP² Board are used. From Figure 4.3, the results shows that lesser pedestrians has a better result and when the number of pedestrians increases, the fps will drop a lot which may affect the accuracy especially in real time demo as fps is the key in real time demo. In real time demo, the changes are every seconds. If the fps is not high enough, the algorithm may miss out few important frames that may affect the final results and accuracy.

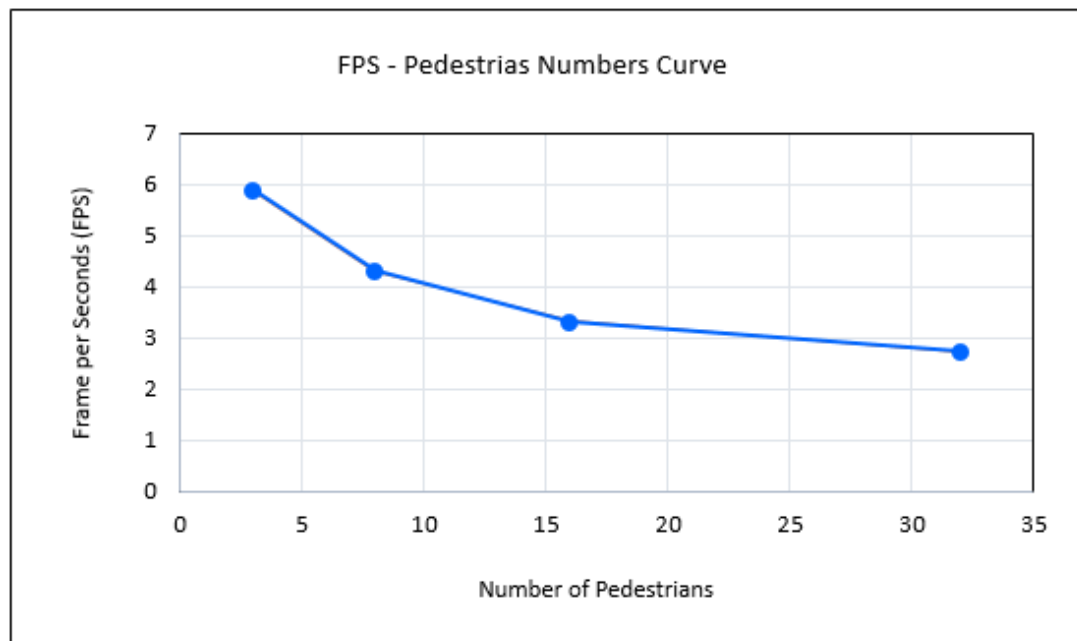


Figure 4.3: Performance Measurement for Different Inputs Complexity

4.2.3 Performance Measurement for Different Performances Complexity

The third performance measurement is on the different performances complexity. A 720p resolution footage is prepared and it is ran under the same algorithm and the frame per second (fps) is the measurement for this experiment. The changes conditions in this measurement are the performances which are the OpenCV display, pedestrian detection, tracking and counting with and without OpenCV effects. From Figure 4.4, the results shows that lesser performances had a better result and when the

performances are getting complicated, the fps will drop a lot which may affect the accuracy especially in real time demo as fps is the key in real time demo. In real time demo, the changes are every seconds. If the fps is not high enough, the algorithm may miss out few important frames that may affect the final results and accuracy.

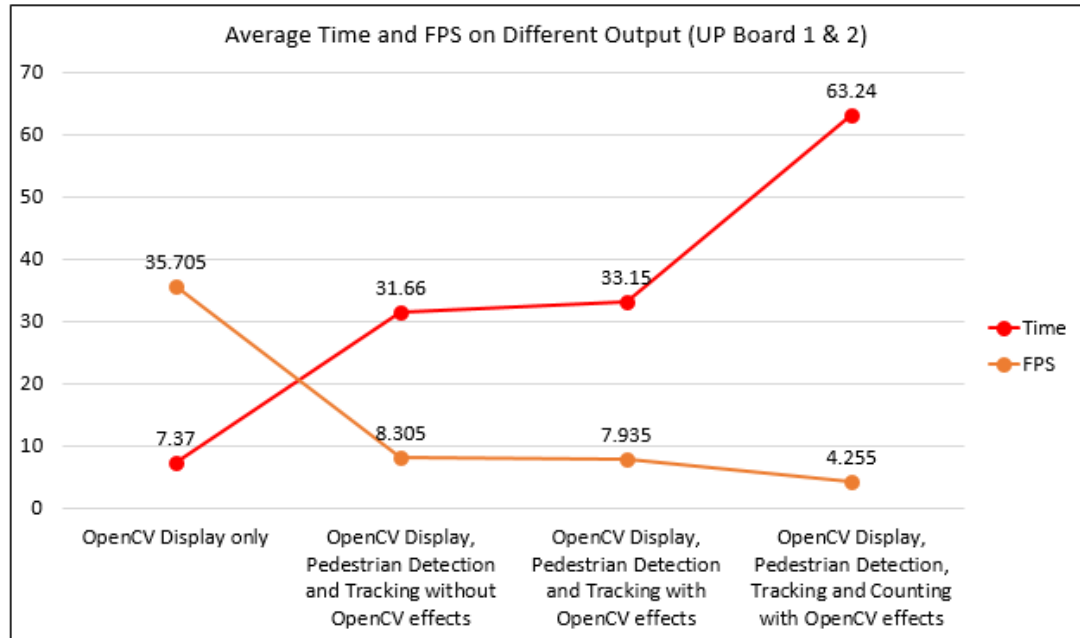


Figure 4.4: Performance Measurement for Different Performances Complexity

4.3 Technical Challenges

After several demo and performance testing, several technical challenges are overcome which had improved the accuracy of results obtained. The technical challenges overcome are the camera view calibration, pedestrian counting method, double counting problem and counting method.

4.3.1 Camera View Calibration

The first technical challenge is the camera view calibration. The camera is adjusted to a higher position which is suggested to be higher than human high. As camera at a higher position, it has a better view angle and the camera is able to capture a better human posture which will help in the sensitivity of detecting the human. A better view angle which also gives the algorithm a better detection region. Besides, at a higher position, the camera will capture less noise and would not be blocked by any obstacles. This is

proved by the closed circuit television (CCTV) and surveillance camera industry. Figure 4.5 shows the before and after calibration of camera view.

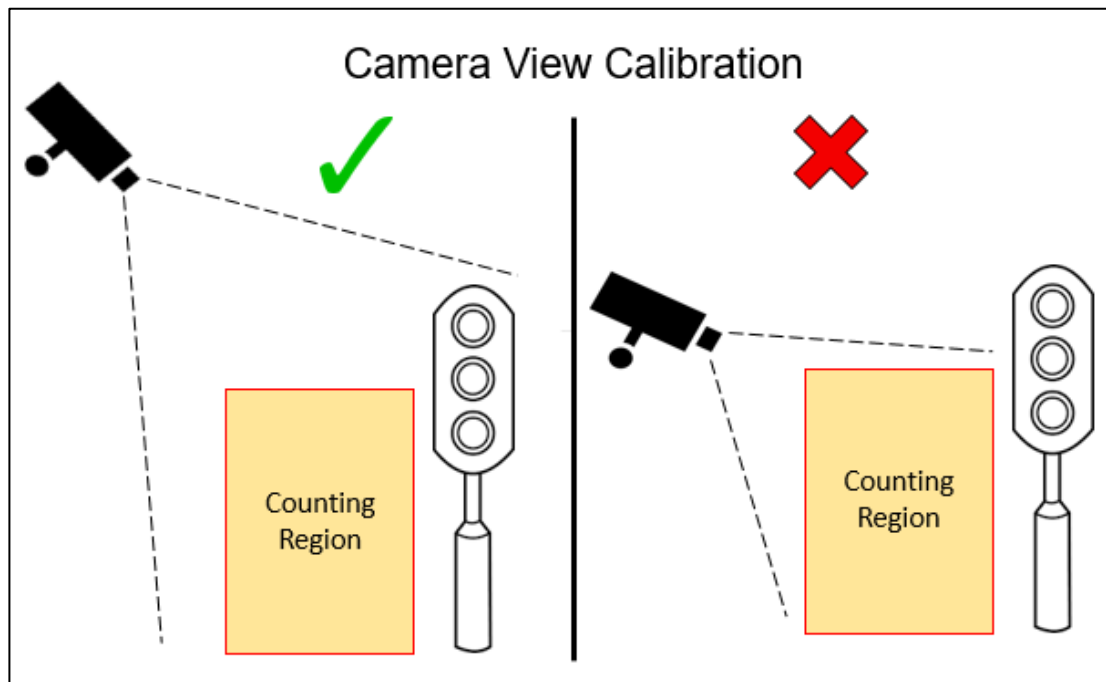


Figure 4.5: Changes of Camera View Calibration

4.3.2 Pedestrian Counting Method

The second technical challenge is the pedestrian counting method. The first version and also the most popular version is the bounding box counting. This method is simple and easy to be done. However, it has a big weakness which is this method unable to detect the pedestrian if the bounding box is greater than the counting region. For instance, the pedestrian is closed to the camera or the prototype is implemented in a small and narrow area. This problems will affect the flexibility of the prototype where it only able to work under limited condition which does not fulfilled this project object. Thus, some modifications are done and the modifications are adding in centroid point and centroid tracking algorithm. The algorithm will track the centroid point only in state of whole bounding box. This modifications also solve the problem and improve the flexibility of the prototype where this prototype able to work in any narrow and small area. Figure 4.6 shows the before and after pedestrian counting method.

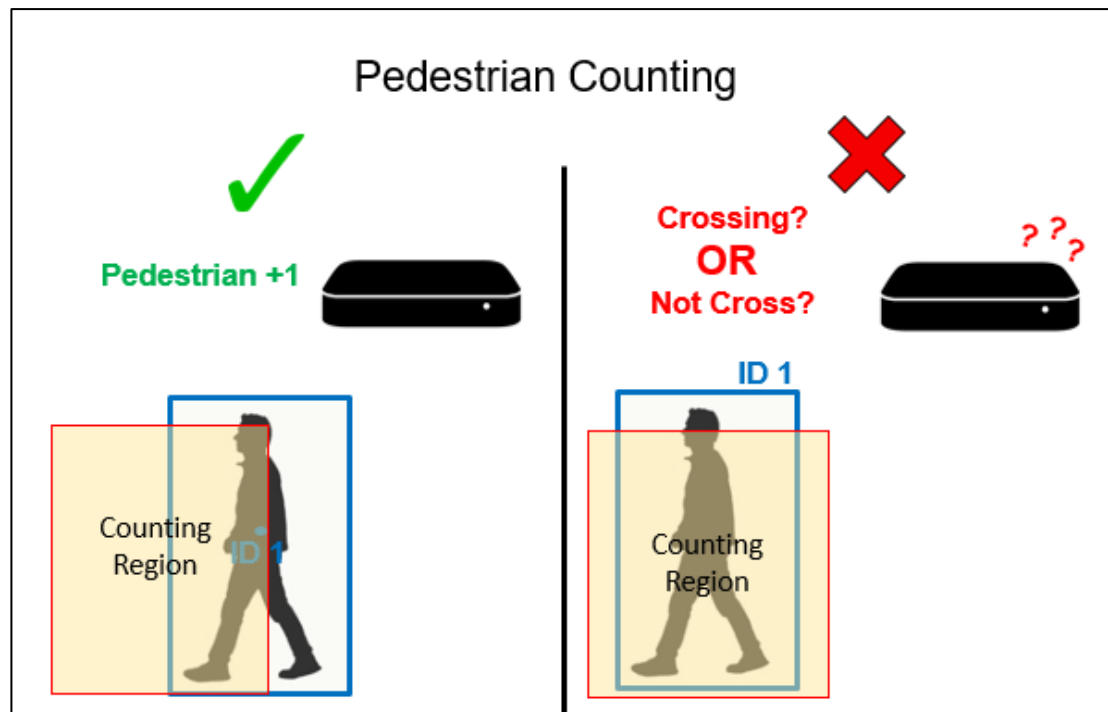


Figure 4.6: Changes of Pedestrian Counting Method

4.3.3 Double Counting Problem

The third technical challenge is the double counting problem. Most of the counting systems in the internet are having this problem and this may due to the complexity of the design and the application of the systems. For example, in the manufacturing industry, the system only needs to count the carriers that passed through on the conveyor belt and most of the time, the conveyor system works in one way direction only. In this project, the prototype is to count the people and human behaviours are the most challenging part for AI industry. For instance, people may walk around the region, testing the limitation of the prototype and trying to confuse the algorithm as well. In order to overcome, the errors caused by human behaviours. Some modifications are done on the algorithm where more terms and conditions are set around the counting region to avoid recounting the same person again and again. In short, the algorithm will only count the person when he is passed the region. Figure 4.7 shows the double counting problem that caused by human behaviour.

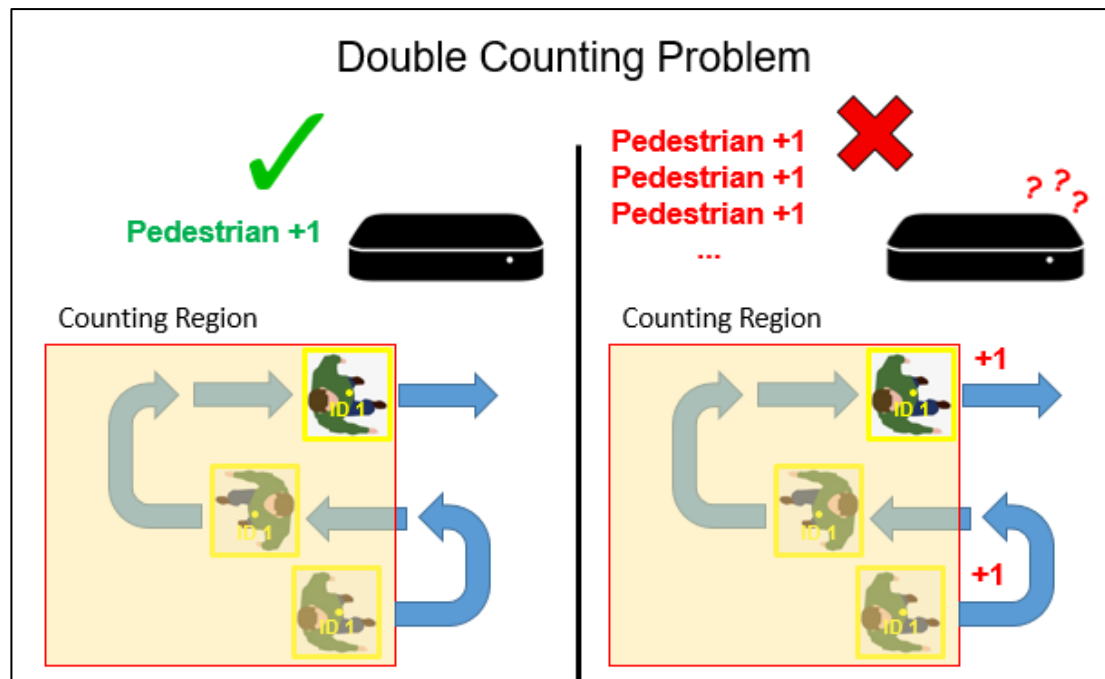


Figure 4.7: Double Counting Problem

4.3.4 Counting Method

The fourth technical challenge is the counting method. One of the problem that real time demo may be occurred is frog problem when the frame per second (fps) dropped. In real time demo, the fps dropped may be due to the device is overload or too much information received at the same time. For instance, there are 50 people appears in the frame during peak hour, the device may not able to handle this situation on time and caused delay problem. This problem will cause some frames are missed to process by the algorithm and the algorithm may miss counted this people that crossed during these missing frames. In order to solve the frog problem, the line counting method is changed to region counting method. The region counting method is more complex than the line counting method and it had more conditions needed to fulfil as well. This counting method able to solve the frog problem as when the people appeared before counting region, the algorithm will remember him if he suddenly disappeared and appeared at counting region, the algorithm will understand that this is frog problem and it will counted the people as well. Figure 4.8 shows the before and after counting method.

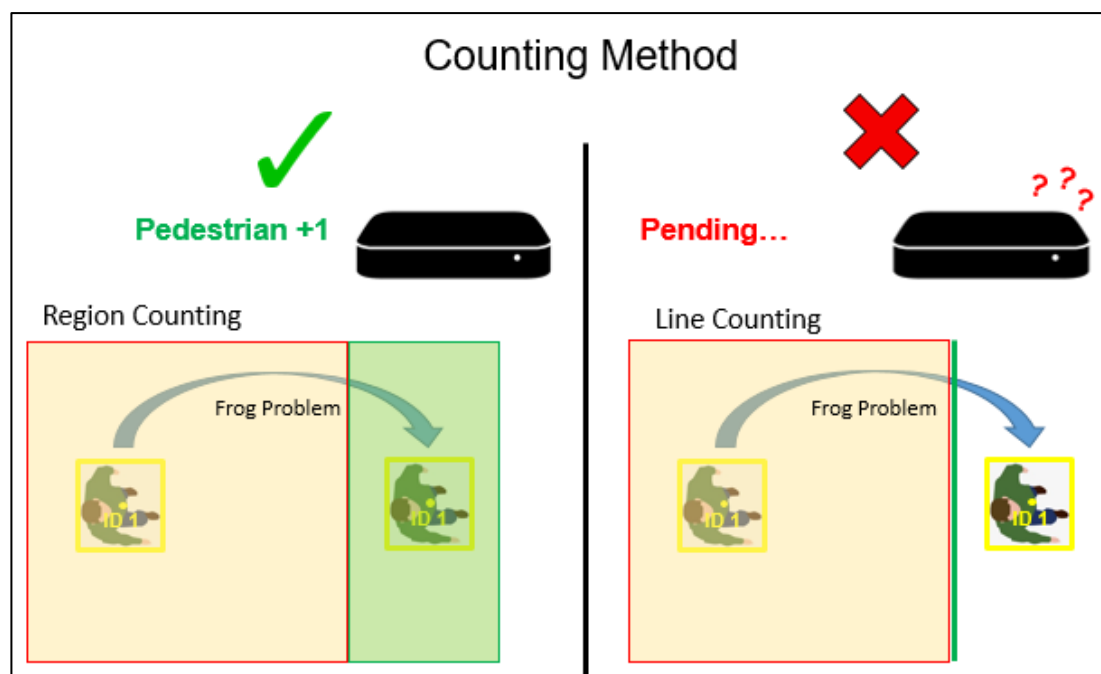


Figure 4.8: Changes of Counting Methods

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The aim and objectives of this project are achieved where a passenger counting with face detection by using Intel UP Squared Board and Intel Movidius Neural Compute Stick 2 (NCS2) are successfully developed and implemented into several applications. These components will be embedded with the software coding to build a prototype with a developed video analytics with machine learning capability that performs the face detection technique and tracking of individual in real time. The prototype also integrated the technique with Internet of Things (IoT), industry revolution 4.0 (industry 4.0) and the big data analysis. It also created with a cloud based system for the visualization of real time data analysis.

Besides, the prototype has solved and overcame the problems that listed in the problem statement where the prototype has a better accuracy, more features, more flexibility, better power consumption and wide applicable applications. The prototype has been tested and able to overcome the real time demo's limitations. It also has a better flexibility which able to perform at any environment and condition. The power consumption problem also overcomes by running through multiple performance measurements to figure out the optimum performance with low power consumption.

In short, this project is successfully met the aim, objectives and some expected outcomes.

5.2 Recommendations for future work

There are several recommended future works that can be done in order to improve this prototype. First of all, specification of the device selected especially the power supply and USB port. Sufficient power needed in order to drive multiple devices. For example, in the Chapter 4, the UP² board does not had sufficient energy to drive two movidius (MYRIAD). If the device is powerful enough, it able to drive multiple movidius at the same time and the performance may improve and optimize. The USB port also plays an important role where the PC did not had sufficient USB 3.0 port and the result is not that positive.

Next, the processor selected also needs to be powerful as the processor plays an important role to split the workload to other devices like movidius and GPU. If the processor is not powerful enough, it will not able to split the workload consistently and equally to other devices which may cause the problem of only one movidius heavy loaded while another lightly loaded.

Some cooling effects need to be done as well. While the movidius is running under the PC, the temperature is high and may be out of the safety working temperature range provided by supplier. This may burn the device in long run.

REFERENCES

- Atmaja, R., Murti, M., Haloman, J. & Suratman, F. Y., 2016. An Image Processing Method to Convert RGB Image into Binary. 3(2), pp. 377-382.
- Chakravartula Raghavachari, A. V. C. S. V. B., 2015. A Comparative Study of Vision based Human Detection Techniques. *Procedia Computer Science*, Volume 58, pp. 461-469.
- Dertat, A., 2017. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. [Online]
Available at: <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>
[Accessed 19 AUGUST 2019].
- Fu, Y. a. A. C., 2019. Flotation froth image recognition with convolutional neural networks. *Minerals Engineering*, Issue 132, p. 183–190.
- Kim, J. H., Moon, J. H., Hwang, E. J. & Kang, P. S., 2019. Recurrent inception convolution neural network for multi short-term load forecasting. *Energy & Buildings*, Issue 194 , p. 328–341.
- Kumar, S., Singh, S. & Kmar, J., 2017. A Study on Face Recognition Techniques with Age and Gender Classification. *IEEE, At Greater Noida*.
- Mayank Chauhan, M. S., 2014. Study & Analysis of Different Face Detection. *International Journal of Computer Science and Information Technologies*, 5(2), pp. 1615-1618.
- Prabhakar, A., Neeti & Devi, R., 2017. DIFFERENT COLOR DETECTION IN AN RGB IMAGE. 7(8), pp. 14503-14506.

Priyanka R.Borude, S. P. G., 2015. Identification and Tracking of Facial Features. *Identification and Tracking of Facial Features*, Volume 49, pp. 2-10.

Qaim Mehdi Rizvi, P. B. G. A. D. R. B., 2011. A Review on Face Detection Methods. *Journal of Management Development and Information Technology* , p. 11.

Saha, S., 2018. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [Online]

Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[Accessed 19 August 2019].

Wah, D. T. W., 2015. *theSundaily*. [Online]

Available at: <https://www.thesundaily.my/archive/1333889-KRARCH296470>


[Accessed 30 October 2019].

Wang, Y. et al., 2019. Detection based visual tracking with convolutional neural network. *Knowledge-Based Systems*, Issue 175 , p. 62–71.

Wu, J. X., 2019. Convolutional neural networks. pp. 13-32.

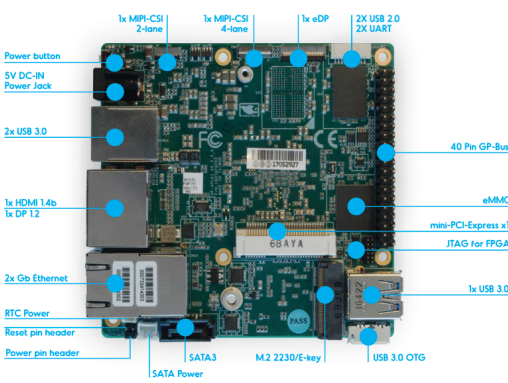
APPENDICES

APPENDIX A: Datasheet of Intel UP Squared Board

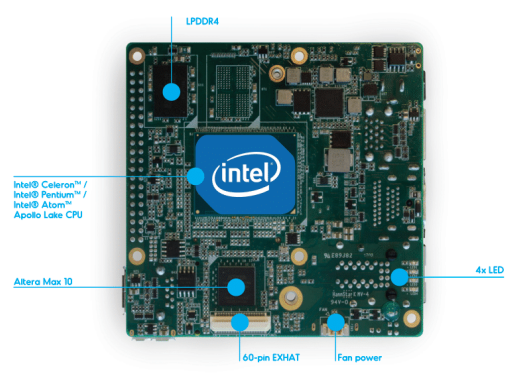


bridge
the gap

specification



Labels for front view: Power button, 5V DC-IN Power Jack, 2x USB 3.0, 1x HDMI 1.4b, 1x DP 1.2, 2x Gb Ethernet, RTC Power, Reset pin header, Power pin header, SATA Power, 1x MIP-CSI 2-lane, 1x MIP-CSI 4-lane, 1x eDP, 2x USB 2.0, 2x UART, 40 Pin GP-Bus, eMMC, mini-PCI-Express x1, JTAG for FPGA, 1x USB 3.0, USB 3.0 OTG, M.2 2230/E-key, SATA3.



Labels for back view: LPDDR4, Intel Celeron™ / Intel Pentium™ / Intel Atom™ Apollo Lake CPU, Altera Max 10, 4x LED, 60-pin EXHAT, Fan power.


UP² (UP Squared) is world's fastest maker board with the high performance and low power consumption features of **Intel® Celeron™, Pentium™ and Atom™ Processors (codename Apollo Lake)**.

The internal GPU is the new **Intel Gen 9 HD** with **12 / 18 Execution Units**, supporting **4K Codec Decode and Encode** for HEVC¹, H.264 and VP8. Thanks to the Vector Units Image Processing Unit and Precision Timing Management to synchronize CPU with I/O, improved determinism (cache QoS, Intel Virtualization Technology), all the graphic processing is effortless to UP² (UP Squared).


UP² (UP Squared) comes with **2GB/4GB/8GB LPDDR4** and **32GB/64GB/128GB eMMC**. A **40-pin GP-bus** provides the freedom for makers to build up their module. Additionally, there is a **60-pin EXHAT** for embedded applications. This allows for the exploration of more possibilities. The expansion capabilities of UP² (UP Squared) goes much further than this. Native **mini-PCI-e, M.2 2230 and SATA3** are all built in on the board. What more could one desire?

The board supports **Windows 10, Windows IoT Core, Ubuntu, Ubuntu, Yocto and Android Marshmallow**. It's really UP to you to decide which operating system is best for your application. Now, all you need is an UP² (UP Squared) to begin your project!


UP - Applications




Drones




Education




Robotics



Media Center



Internet of Things



Home Automation

www.up-board.org / www.up-shop.org / www.up-community.org

UP² - Specifications

APPENDIX B: Datasheet of Movidius Myriad VPU

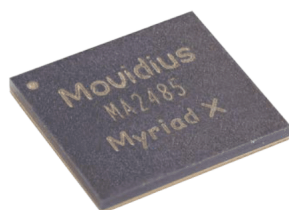
PRODUCT BRIEF

Movidius® Myriad™ X VPU

Movidius
 an Intel company

Enhanced Visual Intelligence at the Network Edge

Movidius® Myriad™ X VPU with Neural Compute Engine



Take your imaging, computer vision and machine intelligence applications into network edge devices with the newest Movidius family of vision processing units (VPUs) by Intel.

Industry Leading Performance at Ultra-Low Power

Intel's Myriad X third generation VPU delivers class-leading performance in computer vision and deep neural network inferencing applications. As the newest member of the Movidius VPU family known for ultra-low power consumption, the Myriad X VPU is capable of delivering a total performance of over 4 trillion operations per second (TOPS).¹ With new performance enhancements, the Myriad X VPU is a power efficient solution that brings advanced vision and artificial intelligence applications to devices such as drones, smart cameras, smart home, security, VR/AR headsets, and 360 cameras.

New Generation of Deep Neural Network Performance

Intel has introduced an entirely new deep neural network processing unit into the Myriad X VPU architecture: the Neural Compute Engine. Specifically designed to run deep neural networks at high speed and low power, the Neural Compute Engine enables the Myriad X VPU to reach over 1 TOPS of compute performance on deep neural network inferences.² The Neural Compute Engine is integrated as part of the power efficient Movidius VPU architecture which minimizes power by reducing data movement on-chip. While the Myriad 2 VPU has provided superior deep neural network support at low power, the Myriad X VPU can now reach 10X higher performance for applications requiring multiple neural networks running simultaneously.²

Customizable Imaging & Vision Pipelines

The Movidius family of VPUs have always provided a unique, flexible architecture for image processing, computer vision, and deep neural networks. The architecture provides a modular approach to configuring imaging and vision workloads because it combines a set of imaging and vision hardware accelerators, such as stereo depth or the Neural Compute Engine, with an array of C-programmable VLIW vector processors, all accessing a common on-chip memory. This approach enables world-class image signal processing (ISP) without the need to make trips to memory for best power efficiency, in addition to interleaved computer vision and deep neural network inference application pipelines, all with a dataflow methodology that reduces power by minimizing data movement. Movidius VPUs deliver an optimal balance between programmability and performance at low power.

Support for 8 HD Sensors and 4K Encoding

The Myriad X VPU features 16 MIPI lanes, which supports up to 8 HD resolution RGB sensors to be connected directly. The high-throughput inline ISP ensures streams are processed at high speeds, while new hardware encoders provide support for 4K resolutions at both 30 Hz (H.264/H.265) and 60 Hz (M/JPEG) frame rates. Other featured interfaces include USB 3.1 and PCI-E Gen 3.

Software Development Kit (SDK) and Tools

The Myriad X VPU ships with a rich SDK that contains all of the software development frameworks, tools, drivers and libraries to implement custom imaging, vision and

deep learning applications on Myriad X VPU. The SDK also includes a specialized FLIC framework with a plug-in approach to developing application pipelines including image processing, computer vision, and deep learning. This framework helps developers focus on the processing, leaving dataflow optimization to the tools. For deep neural network development, the SDK includes a neural network compiler that enables developers to rapidly port neural networks from common frameworks such as Caffe* and Tensorflow* with an automated conversion and optimization tool that maximizes performance while retaining network model accuracy.

Where to Get More Information

For more information, visit www.movidius.com/MyriadX

Movidius® Myriad™ X VPU at a Glance

FEATURES	BENEFITS
Neural Compute Engine	With this dedicated on-chip accelerator for deep neural networks, the Myriad X VPU delivers over 1 trillion operations per second of DNN inferencing performance. ² Run deep neural networks in real time at the edge without compromising on power consumption or accuracy.
16 Programmable 128-bit VLIW Vector Processors	Run multiple concurrent imaging and vision application pipelines with the flexibility of 16 vector processors optimized for computer vision workloads.
16 Configurable MIPI Lanes	Connect up to 8 HD resolution RGB cameras directly to the Myriad X VPU with support for up to 700 million pixels per second of image signal processing throughput.
Enhanced Vision Accelerators	Utilize over 20 hardware accelerators to perform tasks such as optical flow and stereo depth without introducing additional compute overhead. For example, the new stereo depth accelerator can simultaneously process 6 camera inputs (3 stereo pairs) each running 720p resolution at 60 Hz frame rate.
2.5 MB of Homogenous On-Chip Memory	The centralized on-chip memory architecture allows for up to 400 GB/sec of internal bandwidth, minimizing latency and reducing power consumption by minimizing off-chip data transfer.
2 chip packages offered	MA2085: No memory in-package; interfaces to external memory MA2485: 4 Gbit LPDDR4 memory in-package



¹ Overall performance is the architectural calculation based on maximum performance of operations-per-second over all available compute units. Application performance varies based on the application.

² Maximum performance based on peak floating-point computational throughput of Neural Compute Engine. Actual results on deep neural networks may achieve less than peak throughput.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Intel, Movidius, and Myriad are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. * Other names and brands may be claimed as the property of others.

© 2017 Movidius, an Intel Company.

Printed in USA

0116/LTW/MIM/PDF

Please Recycle

333867-001-001US


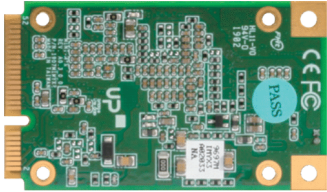
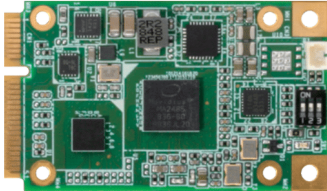
APPENDIX C: Datasheet of AI Core X (Processor Chipset for Movidius)

01

AI Edge Computing Solutions

AI Core X

AI Edge Computing Module with Intel® Movidius™ Myriad™ X VPU



Features

- Intel® Movidius™ Myriad™ X VPU
- 1x MYDX on mPCIe
- Intel® Vision Accelerator Design SW SDK
- Supported Framework: TensorFlow, Caffe, MXNET
- Ubuntu 16.04, Windows® 10

CE FCC

Specifications

System	
IC	Intel® Movidius™ Myriad™ X VPU, MA2485
Support frame work	Tensorflow, Caffe, MXNET
Others	
Form Factor	Mini PCIe
Dimension	2.01" x 1.18" (51 mm x 30 mm)
Certification	CE/FCC Class A
Operating Temperature	32 °F ~ 140 °F (0 °C ~ 60 °C)
Operating Humidity	10% ~ 80% relative humidity, non-condensing

Packing List

- M2 screw x 1
- AI Core X x 1

Ordering Information

Part Number	Description
PER-TAICX-A10-001	AI CORE Movidius™ Myriad™ X 2485, PCIe interface.Rev.A1.0
PER-TAICX-A10-002	AI Core X mPCIe card, with single Myriad™ X, 15 mm heat sink with FAN
PER-TAICX-A10-003	AI Core X mPCIe card, with single Myriad™ X, w/o heat sink

Note: All specifications are subject to change without notice.

www.aaeon.com

01-2

APPENDIX D: Datasheet of UP HD Camera


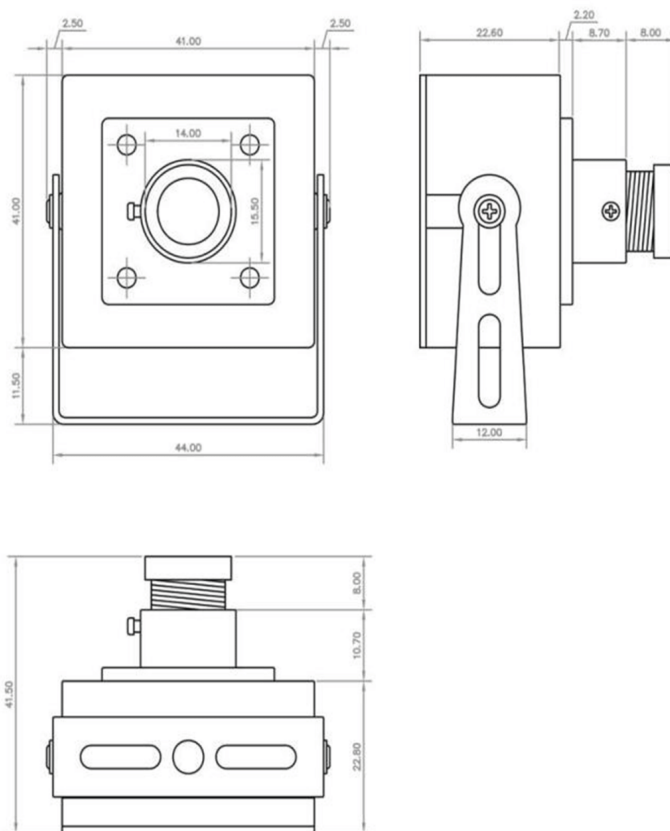
 USB 1080P CAMERA SPECIFICATION	
Model	AD-BHC7350U
Max. Resolution	Full HD 1920*1080
Resolution & frame	320X240 QVGA MJPEG @60fps/ 352X288 CIF MJPEG @60fps 640X480 VGA MJPEG@60fps 1024X768 XGA MJPEG@30fps / 1280X720 HD MJPEG@30fps 1280X1024 SXGA MJPEG@30fps / 1920X1080 FHD MJPEG@30fps
Picture format	MJPEG/ YUV2 (YUYV)
Focus mode	Manual focus
Sensor	1/2.7" OV2735
Mini illumination	0.05LUX
Interface	USB2.0 High Speed
Lens	F3.6mm
Support OTG	USB2.0 OTG
Support free driver	USB Video Class (UVC)
Size:	41x41x41.5mm/100g
Exposure	Auto
Auto white balance	Auto
AGC	Support
Work temperature	-20°C to 80°C
Operating Humidity	30%~90%Rh
Working power	DC5V
Cable	1M
Adjustable parameters	Brightness/Contrast/Color saturation /Definition/Gamma/WB
Support OS	WinXP/Vista/Win7/8/10 Wince with UVC/Linux/MAC-OS/Android
Application area	Entrance guard / high-speed capture / face recognition / advertising machine etc
Certificate	ROHS,CE,FCC

Figure and Mechanical Dimension (mm)



APPENDIX E: main.py

```
from __future__ import print_function
from opencvino.inference_engine import IENetwork, IEPlugin
from argparse import ArgumentParser, SUPPRESS
from pyimagesearch.customcentroidtracker import PersonCentroidTracker
from pyimagesearch.customtrackableobject import PersonTrackableObject
from imutils.video import FPS

import numpy as np
import imutils
import sys
import os
import csv
import cv2
import time
import datetime
import pyautogui
import logging as log

(H, W) = (None, None)

personct = PersonCentroidTracker()

persontrackableObjects = {}

totalcount = 0

persontotalminus = 0
persontotalplus = 0

status = "[Unavailable]"
```

```
detections = 0
```

```
writer = None
```

```
leftx1 = 0
```

```
lefty1 = 0
```

```
leftx2 = 0
```

```
lefty2 = 0
```

```
rightx1 = 0
```

```
righty1 = 0
```

```
rightx2 = 0
```

```
righty2 = 0
```

```
topx1 = 0
```

```
topy1 = 0
```

```
topx2 = 0
```

```
topy2 = 0
```

```
bottom = 550
```

```
frame_X = 300
```

```
frame_Y = 250
```

```
region = 30
```

```
#band = 200
```

```
band = 0
```

```
enter = False
```

```
exit = False
```

```
def build_argparser():
```

```
    parser = ArgumentParser(add_help=False)
```

```
    args = parser.add_argument_group('Options')
```



```

    args.add_argument('-k', '--heightmultiplier', type=int, default=10, help='line
multiplier 1/20')
    args.add_argument('-j', '--widthmultiplier', type=int, default=2, help='line
multiplier 1/10')
    args.add_argument("-o", "--output", type=str, help="path to optional output video
file")
    args.add_argument('-h', '--help', action='help', default=SUPPRESS, help='Show
this help message and exit.')
    args.add_argument("-m", "--model", help="Required. Path to an .xml file with a
trained model.", required=True, type=str)
    args.add_argument("-i", "--input", help="Required. Path to video file or image.
'cam' for capturing video stream from camera",
                      required=True, type=str)
    args.add_argument("-d", "--device", help="Optional. Specify the target device to
infer on; CPU, GPU, FPGA, HDDL or MYRIAD is "
                      "acceptable. The demo will look for a suitable plugin for device
specified. "
                      "Default value is CPU", default="CPU", type=str)
    return parser

log.basicConfig(format="[ %(levelname)s ] %(message)s", level=log.INFO,
stream=sys.stdout)
args = build_argparser().parse_args()
model_xml = args.model
model_bin = os.path.splitext(model_xml)[0] + ".bin"

print(os.path.splitext(model_xml)[0])

if os.path.splitext(model_xml)[0] == "models/mobilenet-ssd":
    CLASS_PERSON = 1
else:
    CLASS_PERSON = 1

```

```

# Plugin initialization for specified device and load extensions library if specified
log.info("Initializing plugin for { } device...".format(args.device))
plugin = IEPlugin(device=args.device, plugin_dirs=None)

# Read IR
log.info("Reading IR...")
net = IENetwork(model=model_xml, weights=model_bin)

if plugin.device == "CPU":
    supported_layers = plugin.get_supported_layers(net)
    not_supported_layers = [l for l in net.layers.keys() if l not in supported_layers]
    if len(not_supported_layers) != 0:
        log.error("Following layers are not supported by the plugin for specified device
{ }:\n { }".
                  format(plugin.device, ', '.join(not_supported_layers)))
        log.error("Please try to specify cpu extensions library path in demo's command
line parameters using -l "
                  "or --cpu_extension command line argument")
        sys.exit(1)

assert len(net.inputs.keys()) == 1, "Demo supports only single input topologies"
assert len(net.outputs) == 1, "Demo supports only single output topologies"

input_blob = next(iter(net.inputs))
out_blob = next(iter(net.outputs))
log.info("Loading IR to the plugin...")
exec_net = plugin.load(network=net, num_requests=2)

# Read and pre-process input image
n, c, h, w = net.inputs[input_blob].shape
del net

if args.input == 'cam':
    input_stream = 1

```

```

else:
    input_stream = args.input
    assert os.path.isfile(args.input), "Specified input file doesn't exist"

k = args.heightmultiplier
j = args.widthmultiplier

cap = cv2.VideoCapture(input_stream)

if cap is None or not cap.isOpened():
    cap = cv2.VideoCapture(0)

cur_request_id = 0
next_request_id = 1

log.info("Starting inference in async mode...")
is_async_mode = True

show_stats = True
flip_mode = False

render_time = 0
ret, frame = cap.read()

print("")
print("To close the application, press 'CTRL+C' or any key with focus on the output window")
print("")
print("To flip video output, press 'f' on the output window")
print("")
print("To hide or show stats panel, press 's' on the output window")

fps = FPS().start()

```

```

while cap.isOpened():

    W = int(cap.get(3))
    H = int(cap.get(4))

    center = (W / 2, H / 2)

    if is_async_mode:
        ret, next_frame = cap.read()
        if flip_mode:
            M = cv2.getRotationMatrix2D(center, 180, 1)
            next_frame = cv2.warpAffine(next_frame, M, (W, H))
    else:
        ret, frame = cap.read()
    if not ret:
        break

    output = frame.copy()

    topx1 = j * W // 40
    topy1 = k * H // 40
    topx2 = (40 - j) * W // 40
    topy2 = k * H // 40

    leftx1 = j * W // 40 - band
    lefty1 = H
    leftx2 = j * W // 40
    lefty2 = k * H // 40

    rightx1 = (40 - j) * W // 40 + band
    righty1 = H
    rightx2 = (40 - j) * W // 40

```

```

    righty2 = k * H // 40

    alpha = 0.5

    rectangle = np.array([[(leftx1, lefty1 - bottom), (topx1, topy1), (topx2, topy2),
    (rightx1, righty1 - bottom)]], np.int32)

    cv2.fillPoly(output, rectangle, (0, 150, 0), 255)
    cv2.addWeighted(output, alpha, frame, 1 - alpha, 0, frame)

    initial_w = cap.get(3)
    initial_h = cap.get(4)

    inf_start = time.time()
    if is_async_mode:
        in_frame = cv2.resize(next_frame, (w, h))
        in_frame = in_frame.transpose((2, 0, 1)) # Change data layout from HWC to
CHW
        in_frame = in_frame.reshape((n, c, h, w))
        exec_net.start_async(request_id=next_request_id, inputs={input_blob:
in_frame}))

    if exec_net.requests[cur_request_id].wait(-1) == 0:
        inf_end = time.time()
        det_time = inf_end - inf_start
        personrects = []
        detections = 0
        res = exec_net.requests[cur_request_id].outputs[out_blob]

        for obj in res[0][0]:
            personobjects = []

            if obj[2] > 0.3:

```

```

if int(obj[1]) == CLASS_PERSON:
    detections += 1
    x1 = int(obj[3] * initial_w)
    y1 = int(obj[4] * initial_h)
    x2 = int(obj[5] * initial_w)
    y2 = int(obj[6] * initial_h)
    box = np.array([x1, y1, x2, y2])
    personrects.append(box.astype(int))
    (startX, startY, endX, endY) = box.astype('int')
#     cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 255, 0), 2)
    personobjects = personct.update(personrects)

if len(personobjects) != 0:
    for (personobjectID, personcentroid) in \
        personobjects.items():

        if show_stats:
            text = 'ID {}'.format(personobjectID)
            cv2.putText(frame, text, (personcentroid[0] - 10, personcentroid[1] -
10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2, )
            cv2.circle(frame, (personcentroid[0], personcentroid[1]), 4, (0, 255, 0), -
1)

            persononto = persontrackableObjects.get(personobjectID, None)

            if persononto is None:
                persononto = PersonTrackableObject(personobjectID, personcentroid)
            else:
                x = [c[0] for c in persononto.personcentroids]
                y = [c[1] for c in persononto.personcentroids]

                direction = personcentroid[1] - np.mean(y)
                persononto.personcentroids.append(personcentroid)

```

```

        if not person.to.counted:
            if direction > 0 and person.centroid[1] > (righty1 - bottom - region)
and person.centroid[1] < (righty1 - bottom) and person.centroid[0] > leftx1 and
person.centroid[0] < rightx1 and np.mean(y) < (righty1 - bottom - region) and
np.mean(x) > leftx2 and np.mean(x) < rightx2:
                enter = True

            if direction > 0 and person.centroid[1] > (righty1 - bottom) and
person.centroid[1] < (righty1 - bottom + region) and person.centroid[0] > leftx1 and
person.centroid[0] < rightx1 and np.mean(y) < (righty1 - bottom) and np.mean(x) >
leftx2 and np.mean(x) < rightx2 and enter == True:
                cv2.line(frame, (topx1, topy1), (topx2, topy2), (0, 0, 255), 5)
                cv2.line(frame, (leftx1, lefty1 - bottom), (leftx2, lefty2), (0, 0,
255), 5)

                cv2.line(frame, (rightx1, righty1 - bottom), (rightx2, righty2), (0,
0, 255), 5)

                cv2.line(frame, (leftx1, lefty1 - bottom), (rightx1, righty1 -
bottom), (0, 0, 255), 5)

                person.totalplus += 1
                person.to.counted = True

                with open('Data/HistoricalData.txt', 'a') as txtfile:
                    txtfile.write("Total People Enter: %d,
" %(int(person.totalplus)))

                    txtfile.write("Total People Exited: %d,
" %(int(person.totalminus)))

                    txtfile.write("Total People in Hall: %d, " %(int(person.totalplus
- person.totalminus)))

                    txtfile.write(str(datetime.datetime.now()))

                    txtfile.write("\n")

                txtfile.close

                with open('Data/HistoricalArray.txt', 'a') as txtfile:
                    txtfile.write("[%d, " %(int(person.totalplus)))
                    txtfile.write("%d, " %(int(person.totalminus)))

```

```

        txtfile.write("%d, " %(int(persontotalplus -
persontotalminus)))

        txtfile.write(str(datetime.datetime.now()))
        txtfile.write("]\n")
        txtfile.close
        enter = False

        if direction < 0 and personcentroid[1] > lefty2 and
personcentroid[1] < (lefty2 + region) and personcentroid[0] > leftx2 and
personcentroid[0] < rightx2 and np.mean(y) > lefty2 and np.mean(x) > leftx2 and
np.mean(x) < rightx2:
            exit = True

        if direction < 0 and personcentroid[1] > (lefty2 - region) and
personcentroid[1] < lefty2 and personcentroid[0] > leftx2 and personcentroid[0] <
rightx2 and np.mean(y) > (lefty2 - region) and np.mean(x) > leftx2 and np.mean(x) <
rightx2 and exit == True:
            cv2.line(frame, (topx1, topy1), (topx2, topy2), (0, 0, 255), 5)
            cv2.line(frame, (leftx1, lefty1 - bottom), (leftx2, lefty2), (0, 0,
255), 5)

            cv2.line(frame, (rightx1, righty1 - bottom), (rightx2, righty2), (0,
0, 255), 5)

            cv2.line(frame, (leftx1, lefty1 - bottom), (rightx1, righty1 -
bottom), (0, 0, 255), 5)

            persontotalminus += 1
            personto.counted = True
            with open('Data/HistoricalData.txt', 'a') as txtfile:
                txtfile.write("Total People Enter: %d,
" %(int(persontotalplus)))
                txtfile.write("Total People Exited: %d,
" %(int(persontotalminus)))
                txtfile.write("Total People in Hall: %d, " %(int(persontotalplus
- persontotalminus)))

```



```

        txtfile.write(str(datetime.datetime.now()))
        txtfile.write("\n")
    txtfile.close
    with open('Data/HistoricalArray.txt', 'a') as txtfile:
        txtfile.write("[%d, " %(int(persontotalplus)))
        txtfile.write("%d, " %(int(persontotalminus)))
        txtfile.write("%d, " %(int(persontotalplus -
persontotalminus))))
        txtfile.write(str(datetime.datetime.now()))
        txtfile.write("]\n")
    txtfile.close
    exit = False

    if direction > 0 and personcentroid[1] < (righty1 - bottom - region):
        personto.counted = False

    if direction < 0 and personcentroid[1] > (lefty2 + region):
        personto.counted = False

    persontrackableObjects[personobjectID] = personto

    resizedframepic = cv2.resize(frame, (512, 288))
    cv2.imwrite("snapshot/1.jpg", resizedframepic)

    if show_stats:
        cv2.putText(frame, "Total People Enter: " + str(persontotalplus), (10, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 4)
        cv2.putText(frame, "Total People Exited: " + str(persontotalminus), (10, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 4)
        cv2.putText(frame, "Total People in Hall: " + str(persontotalplus -
persontotalminus), (10, 150), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 4)

```

```

row = [int(persontotalplus), int(persontotalminus), int(persontotalminus -
persontotalplus), int(detections), datetime.datetime.now()]

with open('Data_File.csv','r') as readfile:
    output_reader = csv.reader(readfile)
    lines = list(output_reader)
    lines[0] = row

with open('Data_File.csv', 'w') as writefile:
    output_writer = csv.writer(writefile)
    output_writer.writerows(lines)
readfile.close()
writefile.close()

with open('Data/RealTimeUpdateData.txt', 'w') as txtfile:
    txtfile.write("Total People Walked to Left: %d\n" %(int(persontotalplus)))
    txtfile.write("Total People Walked to Right: %d\n" %(int(persontotalminus)))
    txtfile.write("Total People in Hall: %d\n" %(int(persontotalplus -
persontotalminus)))
    txtfile.write(str(datetime.datetime.now()))
txtfile.close

with open('Data/RealTimeUpdateArray.txt', 'w') as txtfile:
    txtfile.write("[%d, " %(int(persontotalplus)))
    txtfile.write("%d, " %(int(persontotalminus)))
    txtfile.write("%d, " %(int(persontotalplus - persontotalminus)))
    txtfile.write(str(datetime.datetime.now()))
    txtfile.write("]")
txtfile.close

resizedframe = cv2.resize(frame, (1080, 720))
render_start = time.time()
cv2.imshow("Detection Results", resizedframe)

```

```

render_end = time.time()
render_time = render_end - render_start

if writer is not None:
    writer.write(frame)

if is_async_mode:
    cur_request_id, next_request_id = next_request_id, cur_request_id
    frame = next_frame

fps.update()

key = cv2.waitKey(1)
if key == 27:
    break

if (115 == key): #s, show
    show_stats = not show_stats
    log.info("Show Stats Panel" if show_stats else "Hide Stats Panel")

if (102 == key): #f, flipped
    flip_mode = not flip_mode
    log.info("Video Output is NOT flipped" if flip_mode else "Video Output is
flipped")

if writer is not None:
    writer.release()

fps.stop()

print("[ INFO ] Elapsed Time: {:.2f}s".format(fps.elapsed()))
print("[ INFO ] Approx. FPS: {:.2f}".format(fps.fps()))
print(persontotalplus)

```

```
print(persontotalminus)
```

```
try:
```

```
    os.remove("snapshot/1.jpg")
```

```
except:
```

```
    pass
```

```
cv2.destroyAllWindows()
```

APPENDIX F: CloudDataTransmitter.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from pyimagesearch.firebasecloud import firebasecloud
from pyimagesearch.dashingrequest import dashingrequest
from imutils.video import FPS

import numpy as np
import argparse
import time
import cv2
import os
import csv
import subprocess
import requests

firebasecloud = firebasecloud()
dashingrequest = dashingrequest()
status = 'Unavailable'
cycle = 0

print("[ INFO ] Starting data transmission to Cloud Server...")
print("[ INFO ] Transmission Started. To close the application, press 'CTRL+C' in
terminal")

while True:
    try:
        fps = FPS().start()

        with open('Data_File.csv', 'r') as readfile:
            output_reader = csv.reader(readfile)
            last_line = list(output_reader)
```

```
cycle += 1
print("[ INFO ] Update Cycle: " + str(cycle))
print(last_line)
print("")

persontotalplus = int(last_line[0][0])
persontotalminus = int(last_line[0][1])
persontotalcount = int(last_line[0][2])
persondetections = int(last_line[0][3])

if persondetections > 10:
    status = 'Heavy'
elif persondetections >= 5 and persondetections <= 10:
    status = 'Moderate'
elif persondetections > 0 and persondetections < 5:
    status = 'Low'
else:
    status = 'None'

firebasecloud.uploadstats(
    persontotalplus,
    persontotalminus,
    persontotalcount,
    persondetections,
    status
)

dashingrequest.updatepersonplus(persontotalplus)
dashingrequest.updatepersonminus(persontotalminus)
dashingrequest.updatetotalcount(persontotalcount)
```

```
dashingrequest.updatetrafficvolume(persondetections)
dashingrequest.updatecrowdstatus(status)

fps.update()

except requests.exceptions.ConnectionError:

    pass
except IndexError:

    pass
except ValueError:

    pass
except KeyboardInterrupt:
    fps.stop()
    break

print("[ INFO ] Elapsed Time: {:.2f}s".format(fps.elapsed()))
print("[ INFO ] Total Update Cycles: " + str(cycle))
```

APPENDIX G: firebasecloud.py

```

from firebase import firebase
from google.cloud import storage
import os

os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="gcloud/FinalYearProject-a9114927741e.json"

firebase = \
    firebase.FirebaseApplication("https://finalyearproject-96e65.firebaseio.com/"
                                , None)

client = storage.Client()
bucket = client.get_bucket('finalyearproject-96e65.appspot.com')
imageBlob = bucket.blob("snapshot/")
textBlob = bucket.blob("Data/")

class firebasecloud:

    def uploadstats(self, persontotalplus, persontotalminus, persontotalcount,
persondetections, status):

        persondata = {'Total_Person_Plus': int(persontotalplus),
                        'Total_Person_Minus': int(persontotalminus),
                        'Total_Person_Count': int(persontotalcount),
                        'Person_Detections': int(persondetections),
                        'Crowd_Status': status}

        #firebase.put('/Stats/', 'Person', persondata)
        #firebase.put('/UP1/', 'Person', persondata)
        firebase.put('/UP2/', 'Person', persondata)

```



```
if os.path.exists('snapshot/1.jpg') == True:

    imagePath = "snapshot/1.jpg"
else:
    imagePath = "snapshot/sub1.jpg"

imageBlob = bucket.blob("1.jpg")
imageBlob.upload_from_filename(imagePath)

imageBlob = bucket.blob("IllegalCrossing(UP2).jpg")
imageBlob.upload_from_filename(imagePath)

if os.path.exists('Data/HistoricalArray(UP2).txt') == True:
    textPath = "Data/HistoricalArray(UP2).txt"

textBlob = bucket.blob("HistoricalArray(UP2).txt")
textBlob.upload_from_filename(textPath)

if os.path.exists('Data/HistoricalData(UP2).txt') == True:
    textPath = "Data/HistoricalData(UP2).txt"

textBlob = bucket.blob("HistoricalData(UP2).txt")
textBlob.upload_from_filename(textPath)

if os.path.exists('Data/RealTimeUpdateArray(UP2).txt') == True:
    textPath = "Data/RealTimeUpdateArray(UP2).txt"

textBlob = bucket.blob("RealTimeUpdateArray(UP2).txt")
textBlob.upload_from_filename(textPath)

if os.path.exists('Data/RealTimeUpdateData(UP2).txt') == True:
    textPath = "Data/RealTimeUpdateData(UP2).txt"
```

```
textBlob = bucket.blob("RealTimeUpdateData(UP2).txt")  
textBlob.upload_from_filename(textPath)
```

APPENDIX H: dashingrequest.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import requests # pip install requests
import simplejson as json # pip install simplejson
from urllib import request, parse

url = 'http://localhost:3030'
headers = {'Content-type': 'application/json'}

class dashingrequest:

    def updatepersonplus(self, persontotalplus):

        widget = 'totalpersonplus'
        data = {'auth_token': 'YOUR_AUTH_TOKEN',
                'text': str(persontotalplus)}

        fullUrl = '%s/widgets/%s' % (url, widget)

        requests.post(fullUrl, data=json.dumps(data), headers=headers)

    def updatepersonminus(self, persontotalminus):

        widget = 'totalpersonminus'
        data = {'auth_token': 'YOUR_AUTH_TOKEN',
                'text': str(persontotalminus)}

        fullUrl = '%s/widgets/%s' % (url, widget)

        requests.post(fullUrl, data=json.dumps(data), headers=headers)
```

```
def updatetotalcount(self, persontotalcount):
```

```
    widget = 'totalcount'
```

```
    data = {'auth_token': 'YOUR_AUTH_TOKEN',  
           'text': str(persontotalcount)}
```

```
    fullUrl = '%s/widgets/%s' % (url, widget)
```

```
    requests.post(fullUrl, data=json.dumps(data), headers=headers)
```

```
def updatetrafficvolume(self, persondetections):
```

```
    widget = 'trafficvolume'
```

```
    data = {"auth_token": 'YOUR_AUTH_TOKEN', 'value': persondetections}
```

```
    fullUrl = '%s/widgets/%s' % (url, widget)
```

```
    requests.post(fullUrl, data=json.dumps(data), headers=headers)
```

```
def updatecrowdstatus(self, status):
```

```
    widget = 'status'
```

```
    data = {'auth_token': 'YOUR_AUTH_TOKEN',  
           'text': str(status)}
```

```
    fullUrl = '%s/widgets/%s' % (url, widget)
```

```
    requests.post(fullUrl, data=json.dumps(data), headers=headers)
```