PRESENTER TRACKING FOR VIDEO RECORDING ON INTEL GALILEO BOARD

By

SURAYA NAFILA BT ROZAIDI

14810

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical and Electronic Engineering)

JANUARY 2015

Univeristi Teknologi PETRONAS
Bandar Sri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

PRESENTER TRACKING FOR VIDEO RECORDING ON INTEL GALILEO BOARD

by

Suraya Nafila bt Rozaidi 14810

A project dissertation submitted to the

Department of Electrical and Electronic Engineering

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

Bachelor of Engineering (Hons)

(Electrical and Electronic Engineering)

Approved:

____

AP Dr Fawnizu Azmadi Hussin

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

JANUARY 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

____

Suraya Nafila bt Rozaidi

ABSTRACT

Video Recording plays a crucial role in technological world nowadays. Be it online learning, distance education, to lecture recording. In UTP, there are many lecture rooms. For example, there are several large lecture rooms in Pocket C and D, as well as medium sized lecture rooms in academic blocks. To ensure an effective video recording many camera operators are needed. Thus, to accommodate all lecture rooms with all the camera operators for lecture video recording would be very costly. Thus, to support the need of a cameraman to do video recording for lecture, a more automated and smart video recording by employing a presenter tracking mechanism is proposed. The algorithm for presenter tracking is divided into two parts. The first part is face detection using OpenCV in C++ to detect the presenter. The second part is designed to control the camera's direction from the output of the first part to ensure that the face of the presenter is always at the center of the video. The project will produce a system that can detect faces autonomously and ensure that the camera will always be following the presenter or lecturer along the presentation to ensure an effective video recording and the implementation will be done on Intel Galileo board.

ACKNOWLEDGEMENT

Throughout this project, many people has been contributing and assisting the author in succeeding the project. The author would like to acknowledge:

First and foremost, the project supervisor AP Dr Fawnizu Azmadi Hussin for the guidance given, the knowledge taught and all his effort in helping the author in her project. He helped a lot in improving the author's work. In fact, the author was able to stay in progress because of the kind assistance from him in discussing the problems and solutions.

The author would like to extend a sincere gratitude and appreciation to the author's family for both the mental and physical support. This helps the author to progress the project smoothly. Besides, a special thanks to Ahmed Hesham, for the assistance in troubleshooting, Mr Lai Soon Chong for the help in understanding Intel Galileo better, Khalida, Aisyah, Atika, Frenchlyn and Rifqi for the endless support.

Last but not least, the author would like to thank all parties who have helped in this project and making it a success.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1. Background

Video Recording of a presentation or a video conference would be more effective if the camera(s) can track the location of the presenter or the face of the presenter. Several approaches have been done to detect faces. In this project, the aim is to use the face detection output from OpenCV to control the servos that are controlling the camera's direction during recording. The implementation will be done on Intel Galileo board, which will be used as embedded platform.

1.2. Problem Statement

The effective recording of a presentation or video conference requires the camera to track the position of the speaker. This needs to be implemented in an embedded platform that receives the location information from face detection. Using that information, the camera position needs to be updated to ensure the speaker's face remain in the center position.

1.3. Objectives and Scope of Study

- To integrate video sensor and servo motor to Intel Galileo board for camera rotation
- To implement the control algorithm that tracks the speaker location in the Intel Galileo platform
- To develop face detection algorithms that will start with detecting one face and ignore any other faces on the stage once the face is locked.
- To create a presenter tracking algorithm for video recording based on indoor classroom setup.

CHAPTER 2
LITERATURE REVIEW

2.1 Face Detection

Face detection system is one of the promising applications in the world of intelligent environments to support human activities. Face detection is a way to track a person in a video sequence for the purpose of efficiency in video recording. There are numerous methods in the technology of vision system, be it face detection, gesture recognition, action recognition, or gaze detection [1]. However, this paper aims to focus on the various ways of implementing face detection for presenter tracking in video recording.

In [2], it is proposed to use sensor-based tracking for automatic camera control. The implementation of the algorithm is by detecting a person by using depth imaging by Kinect depth camera. Then, the person is locked and thus, any other people entering the frame will be disregarded. The distinguishing of a tracked person and other person entering the scene is done via distance information. This method is sufficient for a small lecture room because the optimal distance for Kinect camera to do the sensing is up to 4 meters. The methodology in [2] uses two cameras, one is for tracking the lecturer and move accordingly, and the other camera is for video recording.

The method proposed by [3] which is using OpenCV is perhaps the simplest. First, the lecturer is detected using face detection method based on Adaboost algorithm, which is proposed by Viola and Jones [4]. It is a method where the face of person is marked using a square[4]. Then, after detecting the face, the screen is detected using the brightness information as the screen brightness is higher than the others. Then, the coordinates of the screen region is recognized. After the location of the presenter and the screen is secured, a relationship between the two is developed into an action table. The camera is operated according to the action table[3].

Another popular technique is temporal differencing[1]. This technique is robust because it does not depend on steady lighting level or particular colour characteristic for the presenter. This method is suitable for motorized camera tracking as the condition imposed is the camera adjustment will only be made if difference picture reflects development over multiple frames. Otherwise, any change in the frame is thought to be commotion. In this technique, histogram analysis is used to track the individual. If the position of the presenter is outside the center by more than a certain threshold for a pre-set quantity of frames, then only the direction of camera is attuned. This is to avoid unwanted camera reactions.

Another approach for person tracking is using visual processing modalities- stereo, colour and pattern [5]. Darrel, Gordon and Woodfill proposed visual person tracking system using integration of the three visual processing modalities mentioned earlier. This method uses multiple fixed cameras to estimate the depth. They use intensity- invariant colour classifier to detect regions of flesh tone on the presenter's body part region. Then, face detection module is used to distinguish between head region from hands, legs, and other body parts.

Cascade object tracking, which is another effective algorithm is proposed in [6], where the algorithm is divided into three parts. The first part is to find the head area of the presenter using context based edge, from front, back and side of the presenter's face. Then, the second part, comprising of geometric recognizer, is where the system validate if the tracked head area is really a human's head using histogram-based and geometric models. Next, the third part is Cascade face validation using pre-determined face data for face recognition.

Another recent technique used in [7] is useful for autonomous lecture recording. This method is implemented by using a PTZ camera, it is able to track one person, change type of shots, listen to commands by director, and it complies with cinematographic rules. The algorithms of the approach in [7] is divided into three parts. The first part is on the lecturer's position detection, where the detection depends on the type of shots the camera is in. For long shot, the technique used is pedestrian detection based on Histogram of Oriented Gradients (HOG/SVM) whilst for medium shot, Haar/Adaboost technique as proposed in [4] is used to detect face as both methods are known for the fast and excellent calculation. Next part in the algorithm is on the conformation with cinematographic rules. The first rule is about *rule of thirds* where the person is never in the middle of the image but always 1/3th left or right in the video frame, depending on the direction of presenter's action. In this rule, the aim is to find the action axis by using gaze orientation and motion direction. The last part is to perform image-based visual servoing, where the desired direction and speed of camera is calculated.

In a study conducted by [8], the method uses web camera to implement face detection in real time. The study claims that many face detection method uses skin colour detection as it is a unique feature of human face. By using skin colour detection, the face is segmented easily from the dynamic background. Besides, this method can detect face of different poses in real time and uses contour detection. The algorithm used in [8] includes three parts. The first part is skin colour analysis, where the colour space of image of Red, Green and Blue (RGB) is converted into Hue, Saturation, and Value (HSV). Then, skin colour and the background colour are separated by determining the threshold values for HSV for indoor and outdoor. The second part is morphology process where the undesirable pixels are removed by binary morphology process. The third part is the contour detection by using active snake contour method in which the curves of the contours of the face is linked and calculated to get the boundary of each object detected. The final result will be a white ellipse used to mark the contour area. A bounding box is created.

2.2 Intel Galileo

An embedded system is a special purpose computer system designed to perform one or a few dedicated functions. As for this project, Intel Galileo board is going to be utilised to perform a presenter tracking algorithm for video recording which has been done by many other people using other platform. Galileo is an Arduino-compatible board but is more advanced. It is a version of Linux, but an embedded form. Arduino platform is not sufficient to access a webcam to grab a photo or catch a video stream but all these could be done with Linux. On top of that, Galileo supports on-board flash memory but it is limited to install Linux[9]. Thus, the microSD card will be used to access the required features on Galileo. The features include OpenCV (an open-source computer vision application) and Video4Linux2 (V4L2), which will be explained in next section.

This project is utilising webcam to send video frames to OpenCV which is running in Linux of Intel Galileo. OpenCV is used to detect the face of the presenter and track it. The result of the face detection will be used to perform calculation to find the x and y-axis of the center of the face. The coordinates are the information used as a reading via the Linux of Galileo. The reading of x and y-axis is used to control the movement of the webcam with the assistance of two servos to track the detected face of presenter. This project is utilising methods that are previously done by other researchers and implementing it on new technology which is Intel Galileo embedded platform.

2.3 Integration of Video4Linux2 (V4L2) with OpenCV and Webcam

V4L2 is a video record and play utility for Linux. According to [9], it is a set of Application Program Interface (API) and drivers developed to allow Linux operating system to communicate with devices that receive and transmit audio and video. As for this project, V4L2 is needed to communicate with the webcam. The working principle of V4L2 is required to be understood as this project is using OpenCV, where OpenCV often have issues with V4L2. Most of the debugging errors occur due to V4L2 and not OpenCV, thus it is crucial to properly know how to fix the issues which are confusing.

Using Intel Galileo requires the webcam to be compatible with USB Video Class (UVC) standard, which will work with programs involving OpenCV. Before using OpenCV it is important to understand the following aspects of the webcam:

- The encode/pixel formats supported
- The resolutions supported to capture images
- The resolutions supported to capture video
- The frames per second (fps) supported in different encode modes
- The resolutions that really work

The Logitech webcam C270 is the best to be used for this project because it is an affordable camera (US$ 26.00) and complies with the USB Video Class (UVC), and works with face detection projects [9]. The OpenCV libraries offer a prevailing structure to make advanced computer vision applications, abstracting all mathematic, static, and machine learning models out of the application setting. To concentrate specifically on the projects including OpenCV, it is important to understand the capacities of the webcam to be used,as in the points above. Understanding these capabilities utilizing V4L2 will avoid from wasting hours attempting to solve problems which are not coming from OpenCV but V4L2.

2.4 Servo

Servo motors are generally utilized as part robots and gadgets controlled remotely, as in RC planes. However, they are perplexing because of the fact that it functions differently in theory and practically. In theory, the servo rotates between 0 to 180 degrees using PWM pulses that vary between 1 to 2 milliseconds in a period of 20 milliseconds (50Hz) and operating between 4.5 to 6 VDC [9]. With only three wires, the VCC wire is usually red, the ground is usually black, and the pulse signal wire comes in different colours depending on the manufacturer. It is possible to move the servo to a specific position between 0 to 180 degrees [9]. For this project, the suitable servo will be Servo Futaba S3003 or equivalent. However, the author is utilising the normal micro servo HD-1160A. As shown in figure below:

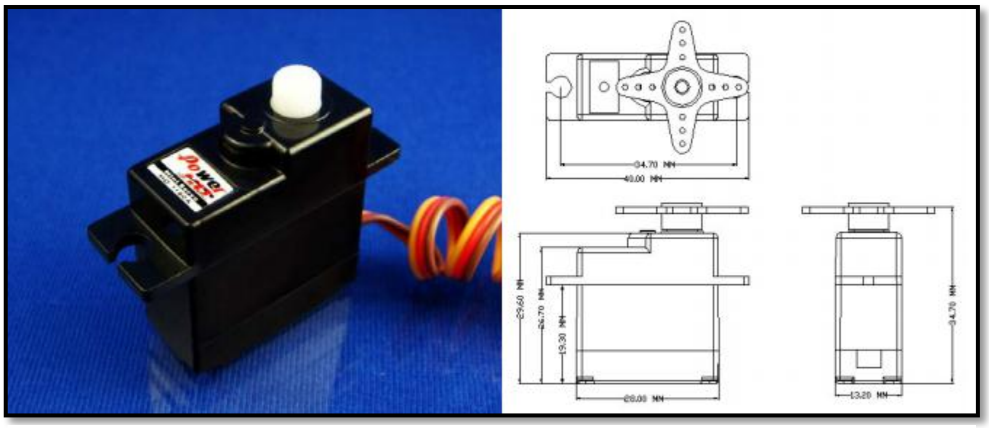


Figure 1 Servo HD-1160A to move the camera

The working principle of a servo is shown below.

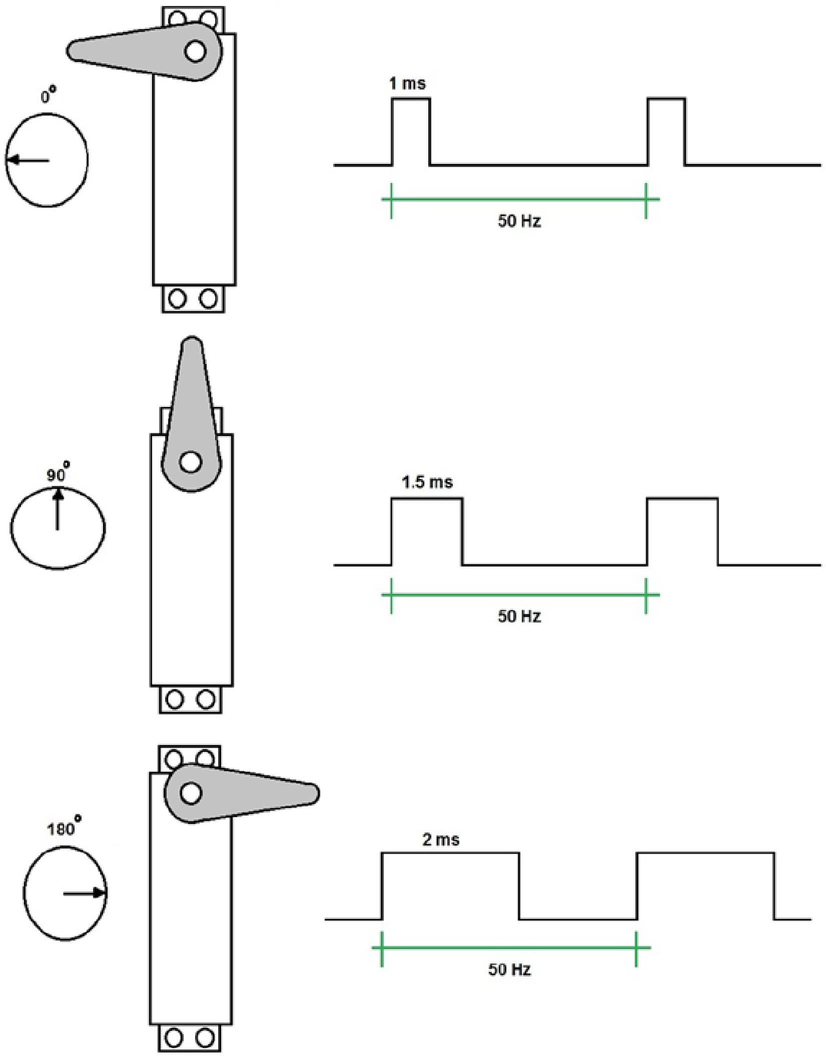


Figure 2 How servo works in theoretically

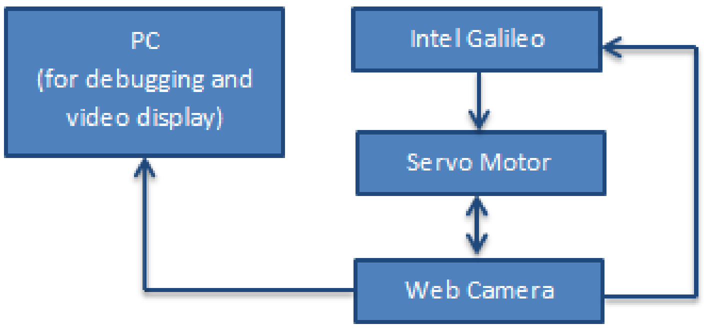
CHAPTER 3
METHODOLOGY

3.1 Use Case

This project intended to produce a video recording system that will use interfacing between Intel Galileo board and a web camera for presenter tracking purpose. The system is implementing an existing technique of face detection that is reliable and integrating it on a new technology from Intel platform, Galileo. The system will be used for lecture recording purpose wherein the setup is for indoor light intensity only. The camera will only be able to track one person at a time. The device will be used by the user in a manner that it needs to be turned on manually every time the lecture needs to be recorded, requiring a switch as an input to start the device.

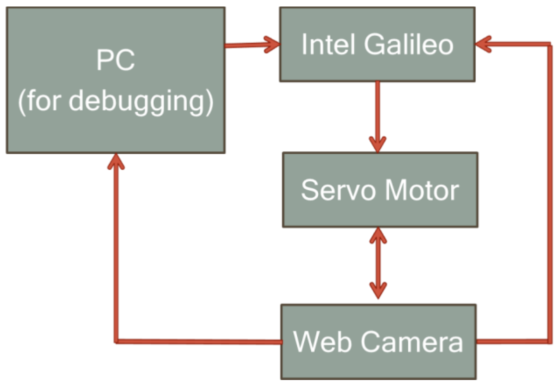
3.2 Research Methodology

Architecture of presenter tracking system

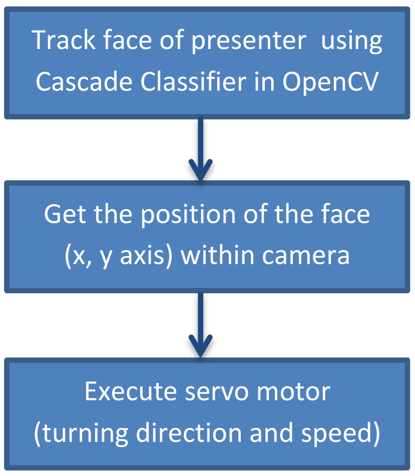


The computer will be used for debugging the Galileo board. All the face detection processing will be done on Linux image of Intel Galileo. The servo movement will be programmed by Arduino IDE of the Galileo. Thus, all the processing for the whole system of presenter tracking will be done on a single board of Intel Galileo. The camera will be attached to the servos and move accordingly to follow the presenter's face.

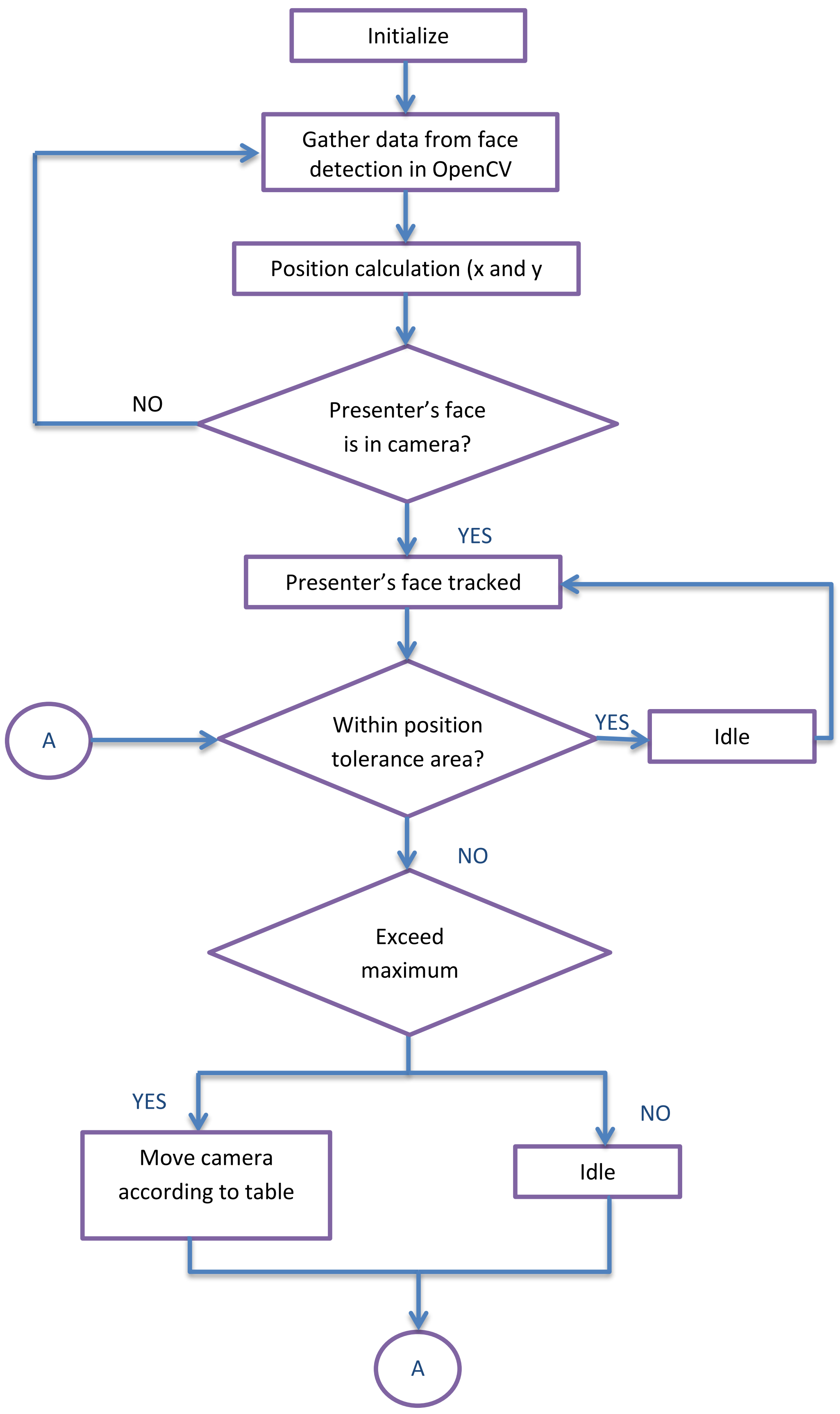
However, due to the technical challenges in building the Linux image on Intel Galileo that can run the face detection code, the methodology in the final stage of prototyping is as below



Architecture of Software

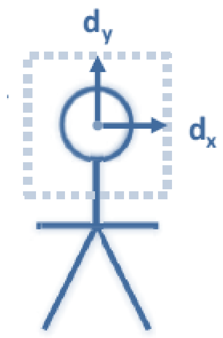

Track face of presenter using Cascade Classifier in OpenCV
Get the position of the face (x, y axis) within camera
Execute servo motor (turning direction and speed)

Presenter Tracking Algorithm



Initialize
Gather data from face detection in OpenCV
Position calculation (x and y
NO
Presenter’s face is in camera?
YES
Presenter’s face tracked
A
Within position tolerance area?
YES
Idle
NO
Exceed maximum
YES
NO
Move camera according to table
Idle
A

Camera action table



d_y	d_x	Camera action
>Max	-	Tilt up
<Min	-	Tilt down
-	>Max	Move Right
	<Min	Move Left
<Max or >Min	<Max or >Min	Idle

Table 1 Conditions and camera actions result

3.3 Project Implementation Activities

The aim of the project is to create a presenter-tracking camera using a web camera and servo motor to Intel Galileo board for camera rotation. To produce the system that will be able to carry out the presenter-tracking algorithm, below are the activities to be done:

i. Literature Review on how to perform face detection and tracking the position of the presenter.

ii. To study a suitable theory to be implemented on the Intel Galileo and to decide on the design of algorithm for the system in which compatible webcam and servo motor will be used for the presenter tracking.

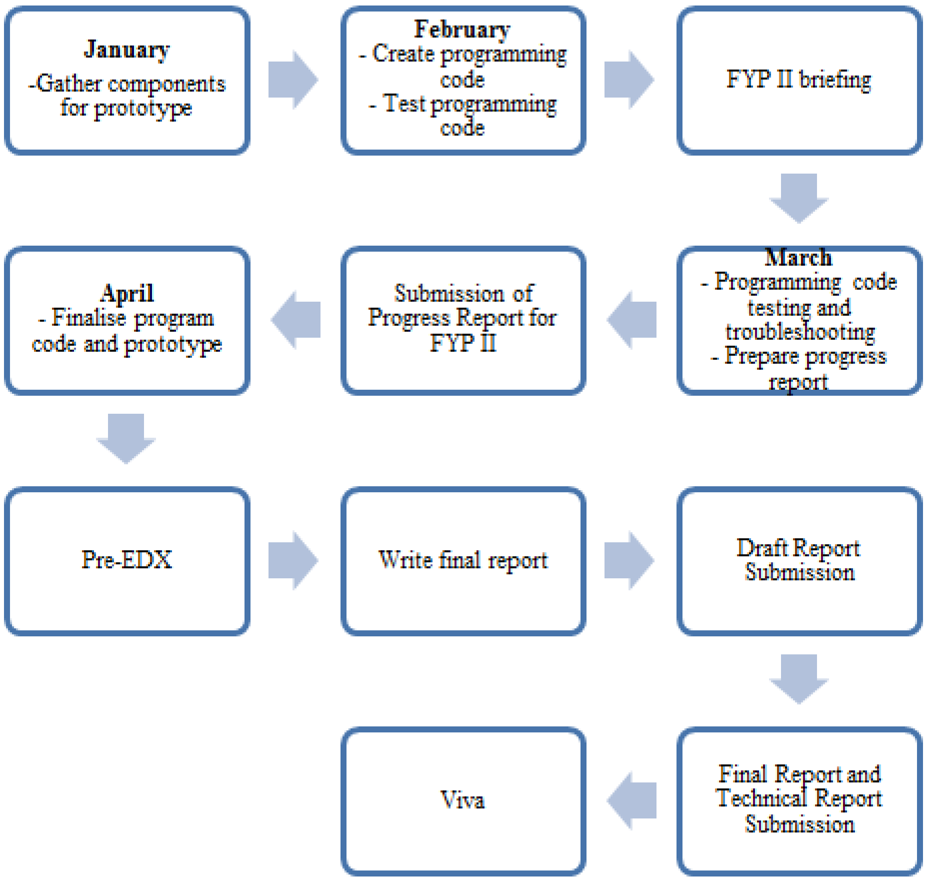
iii. To design algorithms and code it into programs. Camera control algorithm will also be developed to interface between the camera and the Intel Galileo board.

iv. To test the design concept and algorithm approach and execute debugging work if needed.

v. To deliver a working prototype for project demonstration of presenter tracking for video recording.

vi. To create a final report and documentation for future development reference.

3.4 Key Milestones


January
-Gather components for prototype
February
- Create programming code
- Test programming code
FYP II briefing
March
- Programming code testing and troubleshooting
- Prepare progress report
Submission of Progress Report for FYP II
April
- Finalise program code and prototype
Pre-EDX
Write final report
Draft Report Submission
Final Report and Technical Report Submission
Viva

Final Year Project 1															
No	**Item/Week**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	Project title selection		▨												
2	Extended proposal submission						▨								
3	Proposal defence									▨					
4	Draft report submission													▨	
5	Final report submission														▨

Table 2 Key Milestone of FYP1

3.5 Gantt Chart/ Schedule

A Gantt Chart is demonstrated in the following page. Below are the details on the Gantt Chart components:

i. Research

This is the place different resources are explored to help in the creation of the model and system. In view of the researches, an information investigation is done to study a suitable hypothesis and algorithm that is upheld to be incorporated with the framework.

ii. Designing

Amid this stage, the framework of technical setup of program is done. The model parts are collected. The sensors, webcam and drivers are integrated onto the prototype.

iii. Implementation and Testing

The codes are finished and prepared to be troubleshot. A system test is conducted orderly and every sensor is tested independently. At that point the full framework is tried with all sensors, webcam, and drivers incorporated together. From that point on, debugging work is performed and more reliable features are to be incorporated if possible. All results are recorded in a report.

iv. Deadlines

These are the due dates to be noted for the entries of reports and presentations.

Final Year Project 1															
No	**Item/Week**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	Project title selection	▨	▨												
2	Background study and literature reviews on the face detection and Intel Galileo architecture			▨	▨										
3	Identifying suitable design					▨	▨								
4	Extended proposal			▨	▨	▨	▨								
5	Study on the improvement for project							▨	▨						
6	Proposal defence								▨	▨					
7	Familiarize with software and coding										▨	▨			
8	Coding on the face detection output											▨	▨	▨	
9	Draft report												▨	▨	
10	Final report												▨	▨	▨
Final Year Project 2															
No	Item/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
11	Compile face detection coding on Linux image of Intel Galileo	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
12	Communication between face detection and servo movement	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Table 3 Gantt Chart of FYP 1 & FYP 2

3.6 Hardware and Software

Hardware	Description
Intel Galileo board (Gen1)	To compute algorithm and interact with sensors
Webcam	To detect face of presenter and for video display/recording
Servo motor	To control the angle of rotation of webcam according to program code
Miscellaneous (batteries, wires, etc.)	Other parts of the prototype
	Description
Linux OS	For debugging purpose
Arduino IDE for Galileo	To create and upload sketches into Galileo board using USB
OpenCV C++	Computer vision application to detect face of presenter.
Microsoft Office	For all the documentation purposes

CHAPTER 4
RESULTS AND DISCUSSION

As discussed in Chapter 2, this project implementation will be divided into three parts. The first part is the face detection using OpenCV in C++ which is implemented in Linux, where the result of the face detection will be used to perform calculation to find the x and y-axis of the center of the face. Next part is the integration between the output of face detection in Linux image in Intel Galileo and the input to Arduino side of Galileo which is using Arduino IDE. The third part is on the servo movement to move the webcam according to the position of presenter's face within the camera frame.

The reading of x and y-axis is used to control the movement of the webcam with the assistance of two servos to track the detected face of presenter. This project is utilising methods that are previously done by other researchers and implementing it on new technology which is Intel Galileo embedded platform.

4.1 Integration between OpenCV and C++ in Linux

Arduino sketches scripts are great but it has certain limitations. A better performance can be achieved by using certain C libraries. By using Linux application code reads from and writes to files that are actually driver interfaces. It will enable C++ programs to be written and compiled. The source code is provided in OpenCV face detection library. Then, the program is compiled and linked. Then the program will be able to be executed. The C++ code of face detection from OpenCV is shown in Appendix 1.

The face detection code is provided by OpenCV examples to detect face and eyes, named as Cascade Classifier. The essential idea from the code is it uses some XML files that utilize the classifier model. Thus, in the project file, two XML files are loaded, named haarcascade_frontalface_alt.xml and haarcascade_eye.xml. Each file is like a sample model that represents how a human face and eyes are based on the sum of intensity of pixels in a series of rectangles. The files are useful that it has the characteristics of human faces and eyes and the Cascade Classifier can detect the face and eyes when the detectMultiscale() is invoked. When a face is detected, a rectangle is drawn around the face and when the eyes are detected, a circle is drawn around the eyes. These drawings are done by the function rectangle() and circle(). The code is modified to calculate the center of the face and produce and output coordinates to be sent through serial USB connection to the Galileo board. Besides, the function rectangle() that is used to draw rectangle around the face can be modifies to draw circle. The results of face detection are as shown below:

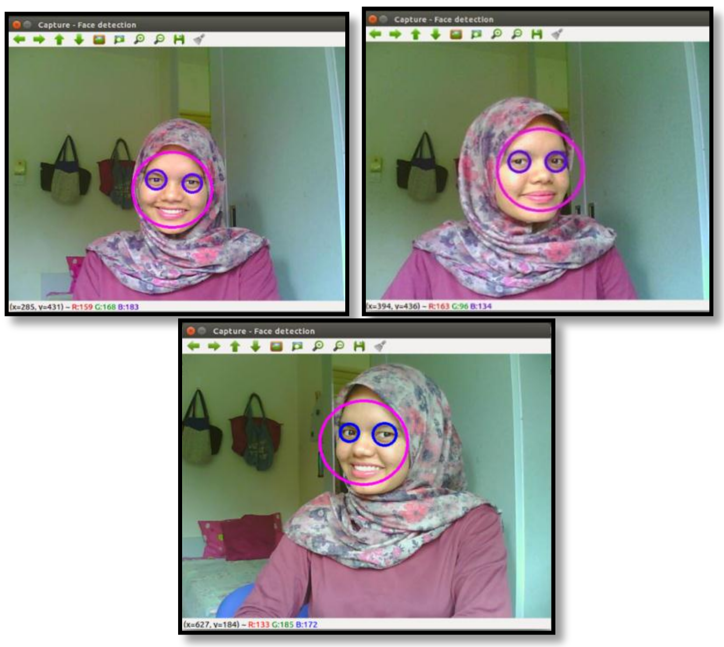


Figure 3 Result of face detection using OpenCV in Linux

The front face of presenter can be detected accurately by the face detection algorithm in OpenCV. However, a face facing sideways will cause some difficulty for the algorithm to detect the face as the points of curve vary according to angle of face direction. The code is then compiled using Yocto Toolchain and uploaded into Linux image of Galileo. However, due to the technical challenges that the author faced, where there is not enough support for the build packages as well as the sources from third party, there is insufficient time to resolve all the issues. Thus, the processing of face detection is transferred to the laptop Linux Operating System.

4.2 Booting Linux in Galileo

It is necessary to download the Intel provided operating system and boot the Galileo from the microSD card on the Galileo board. Once the SD card version of Linux is running, then Galileo board can communicate with the Operating System. The Linux operating system is built from the Kernel Source code from kernel.org. Using a build tool called Yocto. Yocto is a system generation build tool where it will scout the Web looking for different source code and build up a complete Linux Operating System from scratch. It will also generate a cross compiler tool chain. Hidden in the Arduino program is a cross compiler which can generate binary program targeting the Galileo board.

4.3 Linking Linux Program with Arduino Sketches

If OpenCV detects a face it will track it and calculate its center's X,Y coordinates. The coordinates are then passed on to the Galileo. Since there will be two processes inside Galileo board, the two processes need to communicate. This is called inter-process communications (IPC). A good method of IPC will be socket communication, which can transfer data between any processes running on the same computer or through network connection. However, since the image of Linux on Intel Galileo cannot be built and the processing of face detection is done on the laptop, thus serial communication is used via serial USB connection.

4.4 Servo Movement for Camera Control

The coding to move the servo will be based on the output reading from face detection which generates the x and y-axis of location of face in the camera frame. Two servos will be used to control the x and y-axis of the camera according to the position of the presenter's face. Intel Galileo does not provide a sufficient PWM signal resolution in that the PWM signal does not offer a good resolution. It means if more number of servos is used, the movement will be seen to be choppy due to lack of precision in the PWM. However for this project, as only two servos will be used, Intel Galileo is still capable to produce an acceptable result. The pseudocode used to move the servo according to the coordinates received from the Linux serial port is attached in Appendix 2. The final attached prototype is shown in the picture below. Extra power supply which is 6V is needed to ensure that the servo is working properly, not draining out the power from Galileo board and causing problem in the whole functionalities of tracking.

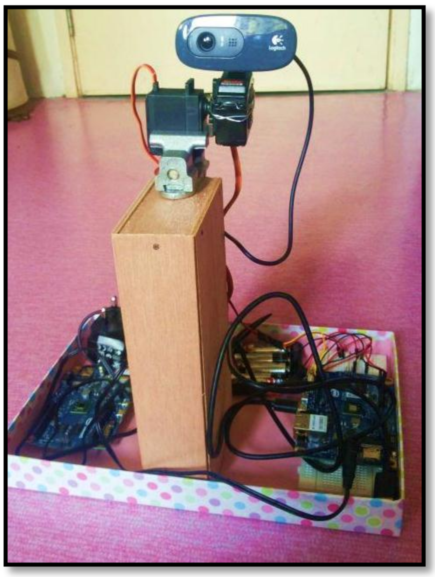


Figure 4 Final prototype of presenter tracking

The biggest challenge when working with servos is the different standardization and varieties of servos available in market, with different specifications, including the degrees of rotation in certain milliseconds causing the difference in proportionality to pulse length in one direction.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

This project implementation is using an existing method of face detection that is using OpenCV in C++ using Linux as operating system and implementing it on a new embedded platform of Intel Galileo. This project is done to ensure an effective video recording by using an automated and smart video recording by employing a presenter tracking mechanism. There are many methods of face detection that can be employed for a better and robust presenter tracking algorithm. As for this project OpenCV is utilized for the face detection part where the Cascade Classifier method is applied.

The uniqueness of Intel Galileo is that it has its own image of Linux in the memory card as well as the Arduino side of it. Thus, comparing it to Arduino, this project of presenter tracking can be done on the Galileo board itself rather than using a separate Operating System outside the microcontroller to carry out the face detection process and yields the output for servo tracking which can be done in the sketch. It can be seen how powerful Intel Galileo is and how easy it is to integrate the Arduino reference APIs into the Linux libraries.

On the other hand, Intel Galileo is occasionally not that easy to work with for a beginner. It requires some trainings as well as learning from specific book about it. There are fewer samples on coding for Intel Galileo as compared to Arduino open sources. Intel Galileo requires having a specific compiler which is a Yocto Toolchain to generate an executable file. Besides, Intel Galileo needs a 32-bit operating system and if you have a 64-bit operating system, it will yield some error, even it is possible to fix it, but it requires some trick.

As recommendations to future researchers, further implementation in outdoor setting and multiple presenter mode should be investigated as Intel Galileo has more processing power potential for further application. The setting for outdoor can be done in the OpenCV itself by changing the setting of light intensity. Besides, the Linux image on Galileo should be utilised to process the face detection using OpenCV. On top of that, with features of Internet of Things (IoT) by Intel, future researchers should consider on utilising it for instant upload of video to the web.

REFERENCES

[1] S. Arseneau and J. R. Cooperstock, "Presenter tracking in a classroom environment," in *Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE*, 1999, pp. 145-148 vol.1.

[2] M. B. Winkler, K. M. Hover, A. Hadjakos, and M. Muhlhauser, "Automatic Camera Control for Tracking a Presenter during a Talk," presented at the Proceedings of the 2012 IEEE International Symposium on Multimedia, 2012.

[3] C. Han-Ping, W. Jung-Ming, F. Chiou-Shann, L. Shih-Chi, and C. Sei-Wang, "Automated lecture recording system," in *System Science and Engineering (ICSSE), 2010 International Conference on*, 2010, pp. 167-172.

[4] P. Viola and M. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision,* vol. 57, pp. 137-154, 2004/05/01 2004.

[5] T. Darell, G. Gordon, J. Woodfill, and M. Harville, "Tracking people with integrated stereo, color, and face detection," in *Proceedings of the 1998 AAAI Spring Symposium on Intelligent Environments*, 1998, pp. 44-49.

[6] K.-D. Lee, M. Nam, K.-Y. Chung, Y.-H. Lee, and U.-G. Kang, "Context and profile based cascade classifier for efficient people detection and safety care system," *Multimedia Tools and Applications,* vol. 63, pp. 27-44, 2013/03/01 2013.

[7] D. Hulens, Goedeme, x, T., and T. Rumes, "Autonomous Lecture Recording with a PTZ Camera While Complying with Cinematographic Rules," in *Computer and Robot Vision (CRV), 2014 Canadian Conference on*, 2014, pp. 371-377.

[8] N. B. Zahir, R. Samad, and M. Mustafa, "Initial experimental results of real-time variant pose face detection and tracking system," in *Signal and Image Processing Applications (ICSIPA), 2013 IEEE International Conference on*, 2013, pp. 264-268.

[9] M. C. Ramon, "Intel Galileo and Intel Galileo Gen 2," in *Intel® Galileo and Intel® Galileo Gen 2,* ed: Springer, 2014, pp. 1-33.

[10] W. Mohsin, N. Ahmed, and C. Mar, "Face Detection Project," *EE368 Digital Image Processing,* 2003.

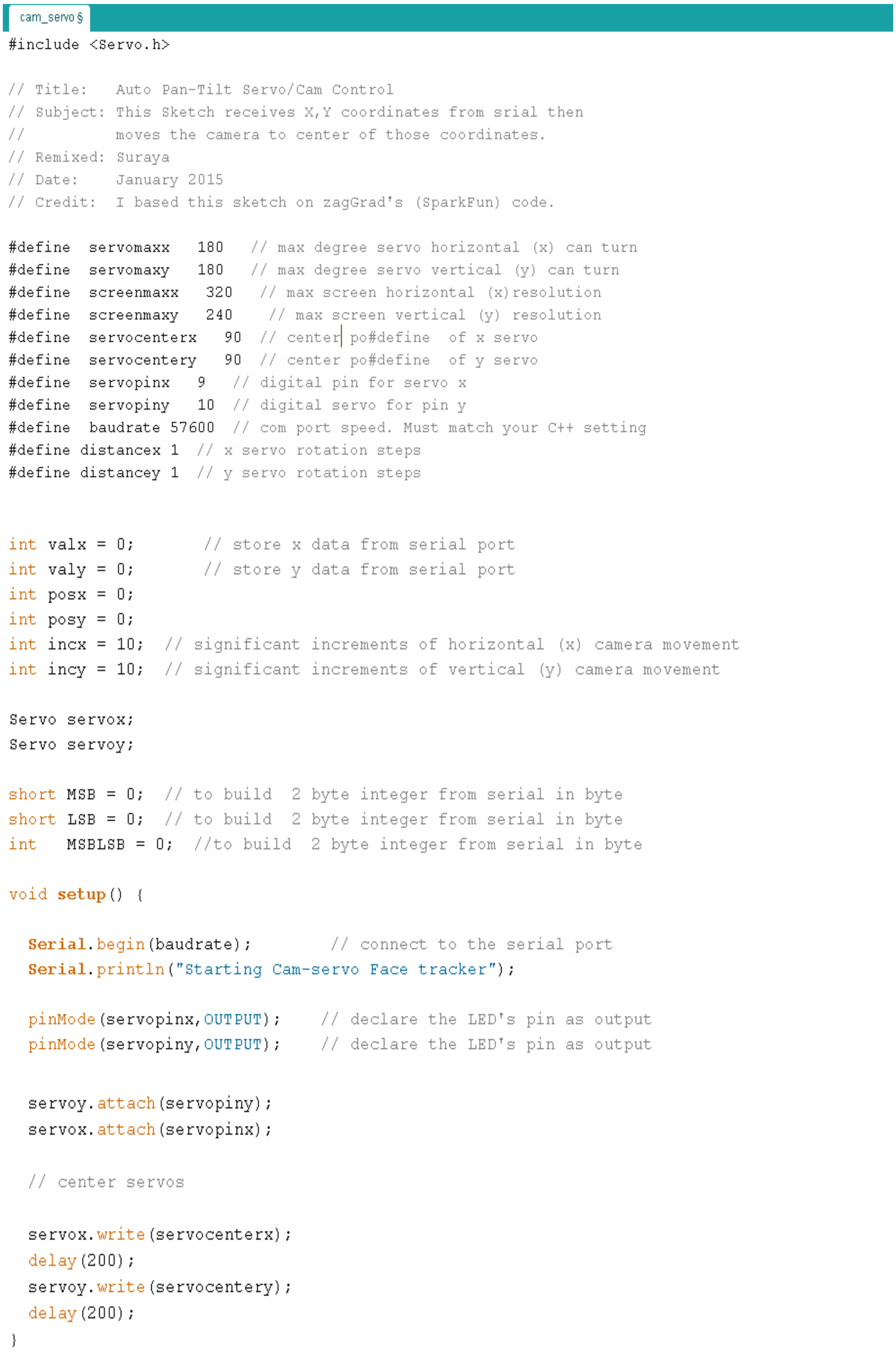
[11] R. E. Kahn and M. J. Swain, "Understanding people pointing: The perseus system," in *Computer Vision, International Symposium on*, 1995, pp. 569-569.

[12] M. Baykara and R. Das, "Real time face recognition and tracking system," in *Electronics, Computer and Computation (ICECCO), 2013 International Conference on*, 2013, pp. 159-163.

APPENDICES

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>
#include <cstdio>
#include "Eserial.h"

using namespace std;
using namespace cv;

/** Function Headers */
void detectAndDisplay (Mat frame);

/** Global variables */
//-- Note, either copy these two files from opencv/data/haarscascades to
your current folder, or change these locations
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
string window_name = "Captcha";

// Serial to Galileo global declarations
int galileo_command;
Eserial * galileo_com;
short MSBLSB = 0;
unsigned char MSB = 0;
unsigned char LSB = 0;


// Serial to Galileo global declarations
int
main (int argc, const char **argv)
{
  CvCapture * capture;
  Mat frame;

  if (argc != 2)
    {
      cerr << "Usage: techbitarFaceDetection <serial device>" << endl;
      exit (1);
    }

  // serial to Galileo setup
  galileo_com = new Eserial ();
  if (galileo_com != 0)
    {
      //Standard Galileo uses /dev/ttyACM0 typically, Shrimping.it
      //seems to use /dev/ttyUSB0 or /dev/ttyUSB1.
     galileo_com->connect(argv[1], 57600, spNONE);
    }

  // serial to Galileo setup
  //-- 1. Load the cascades
  if (!face_cascade.load (face_cascade_name))
    {
      printf ("--(!)Error loading\n");
      return -1;
    };
  if (!eyes_cascade.load (eyes_cascade_name))
    {
      printf ("--(!)Error loading\n");
      return -1;
    };

  //-- 2. Read the video stream
  capture = cvCaptureFromCAM (-1);

  if (capture)
    {
      while (true)
        {
          frame = cvQueryFrame (capture);
          //-- 3. Apply the classifier to the frame
          if (!frame.empty ())
            {
              detectAndDisplay (frame);
            }
          else
            {
              printf (" --(!) No captured frame -- Break!");
              break;
            }
          int c = waitKey (10);
          if ((char) c == 'c')

            {
              break;
            }
        }
    }

  // Serial to Galileo - shutdown
  galileo_com->disconnect ();
  delete galileo_com;
  galileo_com = 0;

  // Serial to Galileo - shutdown
  return 0;
}


/**
 * @function detectAndDisplay
 */
  void
detectAndDisplay (Mat frame)
{
  std::vector < Rect > faces;
  Mat frame_gray;
  cvtColor (frame, frame_gray, CV_BGR2GRAY);
  equalizeHist (frame_gray, frame_gray);
  //-- Detect faces
  face_cascade.detectMultiScale (frame_gray, faces, 1.1, 2,
                                  0 | CV_HAAR_SCALE_IMAGE, Size (30, 30));
  for (int i = 0; i < faces.size (); i++)
    {

      Point center (faces[i].x + faces[i].width * 0.5,
                    faces[i].y + faces[i].height * 0.5);
      ellipse (frame, center,
               Size (faces[i].width * 0.5, faces[i].height * 0.5), 0, 0,
360,
               Scalar (255, 0, 255), 2, 8, 0);

      //  cout << "X:" << faces[i].x  <<  "  y:" << faces[i].y  << endl;

      // send X,Y of face center to serial com port
      // send X axis
      // read least significant byte
      LSB = faces[i].x & 0xff;

      // read next significant byte
      MSB = (faces[i].x >> 8) & 0xff;
      galileo_com->sendChar (MSB);
      galileo_com->sendChar (LSB);
      cout << "X MSB: " << hex << (int) MSB << ", X LSB: " << (int) LSB
<< endl;

      // Send Y axis
      LSB = faces[i].y & 0xff;
      MSB = (faces[i].y >> 8) & 0xff;
      galileo_com->sendChar (MSB);
      galileo_com->sendChar (LSB);
      cout << "Y MSB: " << hex << (int) MSB << ", Y LSB: " << (int) LSB
<< endl;

      // serial com port send
      Mat faceROI = frame_gray (faces[i]);
      std::vector < Rect > eyes;


    }
  //-- Show what you got
  imshow (window_name, frame);
}
```

Appendix 1: C++ code for face detection in Linux



```
#include <Servo.h>

// Title:   Auto Pan-Tilt Servo/Cam Control
// Subject: This Sketch receives X,Y coordinates from srial then
//          moves the camera to center of those coordinates.
// Remixed: Suraya
// Date:    January 2015
// Credit:  I based this sketch on zagGrad's (SparkFun) code.

#define  servomaxx   180   // max degree servo horizontal (x) can turn
#define  servomaxy   180   // max degree servo vertical (y) can turn
#define  screenmaxx   320   // max screen horizontal (x)resolution
#define  screenmaxy   240    // max screen vertical (y) resolution
#define  servocenterx   90  // center po#define  of x servo
#define  servocentery   90  // center po#define  of y servo
#define  servopinx   9   // digital pin for servo x
#define  servopiny   10  // digital servo for pin y
#define  baudrate 57600  // com port speed. Must match your C++ setting
#define distancex 1  // x servo rotation steps
#define distancey 1  // y servo rotation steps


int valx = 0;       // store x data from serial port
int valy = 0;       // store y data from serial port
int posx = 0;
int posy = 0;
int incx = 10;  // significant increments of horizontal (x) camera movement
int incy = 10;  // significant increments of vertical (y) camera movement

Servo servox;
Servo servoy;

short MSB = 0;  // to build  2 byte integer from serial in byte
short LSB = 0;  // to build  2 byte integer from serial in byte
int   MSBLSB = 0;  //to build  2 byte integer from serial in byte

void setup() {

  Serial.begin(baudrate);        // connect to the serial port
  Serial.println("Starting Cam-servo Face tracker");

  pinMode(servopinx,OUTPUT);    // declare the LED's pin as output
  pinMode(servopiny,OUTPUT);    // declare the LED's pin as output


  servoy.attach(servopiny);
  servox.attach(servopinx);

  // center servos

  servox.write(servocenterx);
  delay(200);
  servoy.write(servocentery);
  delay(200);
}
```

```
void loop () {
  while(Serial.available() <=0); // wait for incoming serial data
  if (Serial.available() >= 4)  // wait for 4 bytes.
  {
    // get X axis 2-byte integer from serial
    MSB = Serial.read();
    delay(5);
    LSB = Serial.read();
    MSBLSB=word(MSB, LSB);
    valx = MSBLSB;


delay(5);

// get Y axis 2-byte integer from serial
MSB = Serial.read();
delay(5);
LSB = Serial.read();
MSBLSB=word(MSB, LSB);
valy = MSBLSB;
delay(5);

// read last servos positions
posx = servox.read();
posy = servoy.read();

//Find out if the X component of the face is to the left of the middle of the screen.
if(valx < (screenmaxx/2 - incx)){
  if( posx >= incx ) posx += distancex; //Update the pan position variable to move the servo to the left.
}
//Find out if the X component of the face is to the right of the middle of the screen.
else if(valx > screenmaxx/2 + incx){
  if(posx <= servomaxx-incx) posx -=distancex; //Update the pan position variable to move the servo to the right.


 }

 //Find out if the Y component of the face is below the middle of the screen.
 if(valy < (screenmaxy/2 - incy)){
   if(posy >= 5)posy += distancey; //If it is below the middle of the screen, update the tilt position variable to lower
 }
 //Find out if the Y component of the face is above the middle of the screen.
 else if(valy > (screenmaxy/2 + incy)){
   if(posy <= 175)posy -= distancey; //Update the tilt position variable to raise the tilt servo.
 }

 // Servos will rotate accordingly
 servox.write(posx);
 servoy.write(posy);

}
```

Appendix 2: Servo Direction Sketch