Executive Summary

Robot is a complex machinery equipment with high technology, which has a wide range of applications. As a new industry witch contains the mechanical technology and computer science, robotic industry will plays an increasingly important role in future production and social development.

This project aim to create a detection and tracking robot with movable camera. This robot can detect the person who in front of it, and move its camera to tracking the person. This robot can be installed in a humanoid robot to simulate the human behaviour(follower the user who speak to the robot). And it also can be used to improve existing CCTV system. To make the CCTV surveillance movable, which will help to save energy and storage. Another purpose has been achieved which is a robot for exhibition (this robot is already used in the Open Day of Bristol University).

The works have been done in this project were summarised as follows:

- The design of robot's physical structure. The structure of robot is based on Lego NXT robot, two structures of movement are discussed in this thesis.
- An ultrasound sensor for distance measuring. Use ultrasonic ranging the detect the distance between the robot and the person in front of it.
- One Face Detection algorithm is implemented to the robot system. An effective detection algorithm(Viola-Jones Method) is used to find the user's face in this project.
- Two Face Tracking algorithm is implemented to the robot system. A colour based tracking algorithm(CamShift) and an online learning algorithm(TLD) are used in the project.
- Combine these algorithms to create 3 versions of robot system and analysis the pros and cons of each version.

Contents

1.Introduction

According to the World Industrial Robotics 2011 report [1], the worldwide market value for industrial robot systems in 2010 is estimated to be $17.5 billion and the robot sales will increase by about 6% per year on average between 2012 and 2014.

Today, the robots not only play an important role in industrial production but also enter our normal lives. Although we can choose many intelligent robot products in the market, this 'intelligence' is weak and low-level. Scientists are working to make robot smarter, with the development of the discipline of Artificial Intelligence (Machine Learning, Data Mining, Neuroscience, etc.), some new techniques have been proposed [9].

This project aim is to use those Artificial Intelligence techniques (Face Detection, Face Tracking, etc.) to design a robot which can move its camera to track a person in front of it.

1.1. Some Future Application of Tracking Robot

Now in my project a demonstrate system had been established. The system can be easily transfer to other applications and devices to achieve different functions. There are two possible future applications can use the tracking system.

Exhibition robot in the Open Day

Open Day is an annual event in University of Bristol. Many potential students who are interesting to study in University of Bristol and their parents will participate this event. It is a perfect opportunity for those students to visit the school and find everything they want to know about the university. And the university will also offer a range of exhibitions and welcome sessions to help the visitors find out more university life.

This Tricking Robot will be an object to display in the exhibition of engineering department at CDO(Central Design Office) room in Queen's Building. I will explain and show this robot to the potential student and they can know what a Msc computer science student will working for and hopefully they will interested in it.

Humanoid robot

A humanoid robot is a robot with all or partly of human appearance, and can interaction with human or use man-made tools. The function of face tracking can be applied to a humanoid robot's head, put the camera to the robot's eye, use robot built-in computer to instead the external PC. When a user is talking or interact with the robot, the robot will moving its head follow the user's face. This function can make the humanoid robot more realistic.

Movable CCTV surveillance

CCTV surveillance is widely used in our daily lives. It is estimated that there are more than 4 million cameras distributed in the United Kingdom. It is said that people who living in London are probably to be caught on CCTV camera 300 times per day. The UK

announced that it has the highest surveillance cover rate in the world. These CCTV cameras cover almost everywhere of people's life such as public building, highway, remote areas, and private place such as shops and clubs [3]. But almost all of those CCTV cameras are fixed or only can be manipulate by human, they have no intelligence. So we must put several cameras in one place to ensure that all areas are monitored. In some place where has few people, for example, indoor parking, night road, too much fixed CCTV camera will lead to energy and device waste in those place. A movable tracking camera can solve this problem. It can automatically Sleep when there are no movable objects in its view, and when the object have been caught it has more widely rotation range. So if we use the tricking system into CCTV surveillance can decrease the number of camera and save the storage space and energy.

1.2. General Specifications and Scope

The robot consists of a Lego NXT robot, a camera, and a PC. Lego NXT has provided three motors, two of them used to control the head movement and the other one is used to move the robot arm, camera can collect the image of visitor. All of the data processing and computing rely on the PC, which is connected with the robot and camera. PC will send the signal to control the movement of Lego NXT robot.

The system was designed to satisfy the following requirements:

1. An ultrasound sensor can real-time monitor if someone approaches the robot.
2. If the robot detects a user, then it raises its camera, and captures the video to PC. PC will locate the position of user's face.
3. If the user is in the view of camera, the robot will move the camera to track the user. If the user goes out of the view (the camera has a maximum rotation angle), robot will come into sleep mode until next user comes.

1.3. Thesis Organisation

Section 2 talks about the history of the robots, and shows a general introduction of the various algorithms in face detection and face tracking area. A summary of robot structure and ultrasonic are also presented in this section.

Section 3 is the main design and implementation part in this thesis. The system is divided into 4 parts: Ultrasonic Ranging, Face Detection, Face Tracking and Robot Movement. There are two face tracking algorithms are used in this project(three versions of programs are established). There is one thing need to declare that the main idea of Face Detection and Face Tracking are comes from their own author's paper(had been cited in the thesis). The reason I didn't put the details of those algorithms on the section 2 is these original papers are only rough introduced the main idea of algorithm, I did some specific descriptions in each sections. It will be more easy for the reader to understand the part of implementation after reading those specific descriptions. And

some derivation of the formula and figures is also done by myself (for example, the **Figure 15.** and the formula in section 3.3.3.).

Section 4: There are three visions of programs had been established in this project(face detection, face detection + CamShift tracking and face detection + TLD tracking). In **Section 4**, I choose three scenarios to test to measure these visions. And also discuss the pros and cons for each vision base on the test data. A further work and improvements for this project is presented at the end of this section.

2.Literature Review

This project is aim to design and build a recognition and tracking robot. The robot system will use the knowledge of computer version (Face Detection and Face Tracking) and robotic system (Ultrasonic Ranging and Robot's Morphology).

The scope of my literature review is mainly divided into three parts:

Face Detection: The purpose of Face detection is recognising the presence of human face in an image. Face detection combines image processing, pattern recognition, computer version and other multidirectional technology. There are two mainly directions of face detection: based on feature of face image components and based on overall characteristics of face images. This project chooses the Haar Cascade Classification [4], which is the second direction.

Face Tracking: The purpose of Face detection is tracking an object, which the position of object is provided by face detection algorithm. Face tracking is a part of object tracking, the tracking methods can be divided into four major categories: region-based tracking, active-contour-based tracking, feature- based tracking, and model-based tracking [2]. This project will use two algorithms to build the system, which are Cam-Shift algorithm and TLD algorithm.

leJOS NXJ: leJOS NXJ is a Java programming environment for the LEGO MINDSTORMS NXT ®. It allows you to program LEGO ® robots in Java [6][7]. This project should control 3 motors and an ultrasound sensor with a LEGO NXT Robot by leJOS NXJ.

This section will describes the general robot history review and introduces four techniques which are used to built the robot system.

2.1. History of robots

Robots have more and more important role in our daily life over the last half-century. But at the dawn of the 20th century, robot was just a new thing in science fiction. The word 'Robot' was first announced by a Czechoslovakia writer Karel Capek in his science fiction, according to the word 'Robota' (the original meaning in Czech is "labour") and 'Robotnik' (the original meaning in Polish is "worker"), to create the word of 'Robot' [10].

Now, the concept of robot in the international view has been gradually approaching the same. People can accept this general argument, that the robot is a machine, which can realise its own power and ability to control various functions. International Organisation for Standardisation adopted the definition of Robot Institute of America (1979) to the robot [14]: A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialised devices through various programmed motions for the performance of a variety of tasks.

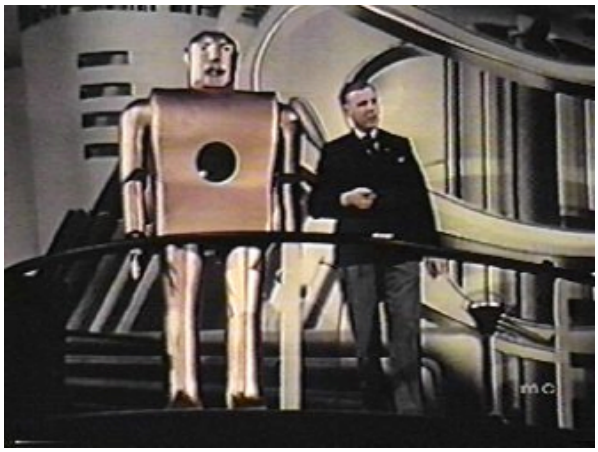


Figure 1. Elektro

The robot shows in the **figure 1** is "Elektro", which is a famous robot in the early year of robotic history [11]. Elektro is built by Westinghouse Electric Corporation between 1937 and 1938 and first exhibit at the 1939 New York World's Fair. He could walk by voice command, speak about 700 words, and even smoke, but "Elektro" is still far away from the real housework. But it makes people longing for domestic robots become more specific.

A science fiction writer Isaac Asimov coined the word 'robotic' used to describe the robot study in his short story *'I, Robot'*, which published in 1942 [12]. Asimov also created his "Three Laws of Robotics", and he later expansion for Four Laws:

> ***Law Zero:*** *A robot may not injure humanity, or, through inaction, allow humanity to come to harm.*
>
> ***Law One:*** *A robot may not injure a human being, or, through inaction, allow a human being to come to harm, unless this would violate a higher order law.*
>
> ***Law Two:*** *A robot must obey orders given it by human beings, except where such orders would conflict with a higher order law.*
>
> ***Law Three:*** *A robot must protect its own existence as long as such protection does not conflict with a higher order law.*

Although this is only the creation of science fiction, but later it developed in academic research [13].

Artificial Intelligence Centre of Stanford Research Institute (now called SRI International) announced their robot named Shakey [15] in 1968. It had a visual sensor that could discover and grasp the blocks by command of controller. But the volume of Shakey's control computer was as large as a room. Shakey can be regarded as the world's first intelligent robot.

In 2002, the iRobot Corporation launched a vacuum cleaner robot, called Roomba. It can avoid obstacles, and automatic design the route, heading for automatic charging when the power is low [16]. Now, Roomba is the largest-selling, most commercial home robot in the world.

Latest social robots

Social robot is a humanoid robot which can communicates and interacts with human by human social rules. Social robot normally has human-like appearance, can replace or partly replace the human work in some area. For example, a receptionist robot is a part of social robots family, and it can talk with the visitors and show the direction where they want to go or answer some simple question likes, 'what's the weather?', 'What's the date?' [17]. The final purpose of receptionist robot is to replace the reception staff in a building or public places.

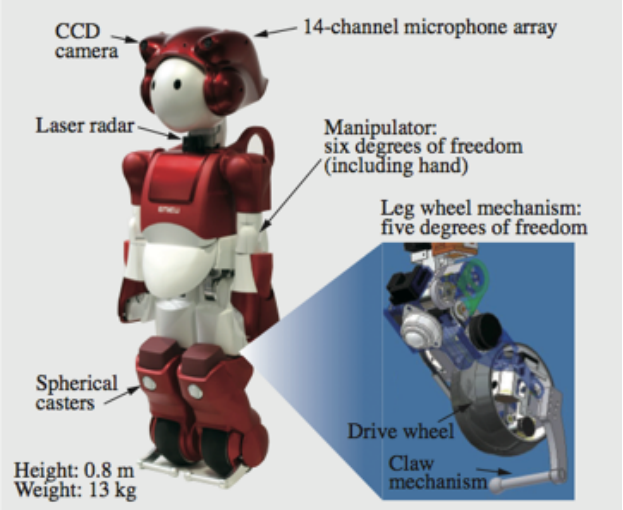


Figure 2. EMIEW2 (figure taken from [18])

• **EMIEW** (**E**xcellent **M**obility and **I**nteractive **E**xistence as **W**orkmate) is a robot developed by Hitachi, the last version is EMIEW2 (2007). EMIEW is a humanoid robot, moving with wheels, maximum moving speed is 6 km/hour [18]. EMIEW 2 has better identification to voice command, the technology of dynamic balance makes it not easily fall down. The function of obstacle avoidance can keep it walk fluently between the multi-obstacle environments. As a receptionist robot, EMIEW 2 uses a laser ranging sensor, the map generation accuracy can reach to 5 cm. It can be through the crowd to reach the destination and find its own route according to the location of the channel and table in the office. EMIEW 2 will easily take the visitors to the designated location and also can help stuff to organise files.

Figure 3. SAYA's facial expressions (taken from [19])

• **SAYA** is humanoid receptionist robot developed by Tokyo University which intended to replace human secretaries in the future. SAYA can talk to visitor and answer some general questions with about 300 words or 700 phrases. She can guides guest to the lift or show the direction where guests want to go. The most special characteristics of SAYA are express humanoid facial expressions (**Figure 3**). The facial skin of robot is made by soft urethane resin can recreate the humanoid texture to robotic face. The facial expressions is created by 19 control point which closed to the skin, with the moving of those 19 points, the robotic face has 19 degrees of freedom to generating facial expressions [19]. SAYA has been applied in some Japanese places, such as department store receptionist and secondary school teacher.

2.2. Introduction of Ultrasonic

The frequency of the sound is the number of vibrations per second, its unit is the Hertz (Hz). Our human ears can hear the sonic frequency between 20Hz and 20000Hz (the acoustic line of **Figure 4**). Human cannot detect the sound wave when its frequency is less than 20 Hz or greater than 20KHz. Therefore, we call the sound wave which is lower than 20Hz as infrasound and a higher than 20kHz sound wave is defined as ultrasound.

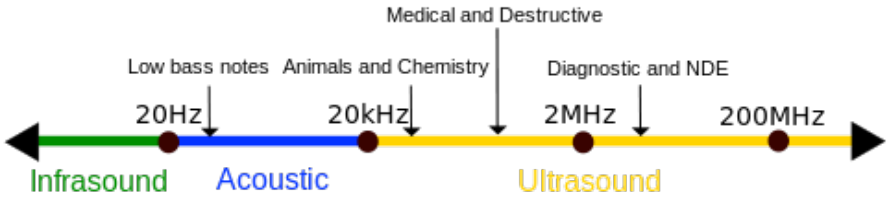


Figure 4. Ultrasound range diagram

The ultrasound can be propagated in different mediums such as gas, liquid, solid, solid melts. Some phenomena will be produced during the propagating of ultrasound such as reflection, interference, superposition and resonance. Because of those feature, ultrasonic widely used in diagnostics, therapeutics, engineering, biology, and other fields.

Ultrasonic Ranging

Ultrasonic Ranging is an application of the ultrasonic. Ultrasound has a strong directionality characteristic and can propagate long distance in the medium, thus the ultrasonic often used for distance measurement. Using ultrasonic to measure the distance is often more rapid, convenient, calculation is simple and easy to achieve real-time control. The accuracy of ultrasound measurement can achieve the requirements of the industrial need, therefore it is widely used in some mobile robots.

The principle of ultrasonic ranging: ultrasonic transmitter send ultrasound, ultrasonic receiver can receive the reflected signal. The distance can be calculated by the difference of send time and receive time.

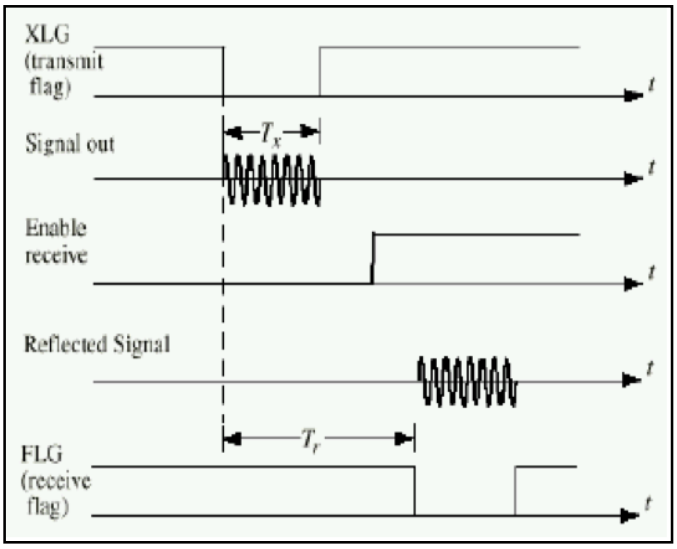


Figure 5. Sample of ultrasonic ranging[5]

This principle is similar to radar range, a sample of ultrasonic ranging shows in **Figure 5**. The speed of ultrasound propagate in the air is 340m/s, after ultrasonic transmitter sending a piece of ultrasound, the timer is started, the ultrasonic wave propagation in the air, and will return immediately if meet an obstacle, the timer will be stop when the receiver get the reflect wave. So in the **Figure 5**, T_r is the time difference between sent and receive, the distance(S) between ultrasonic transmitter and the obstacle can be calculate as $S = 340\times T_r/2$ m.

2.3. Introduction of Face Detection

The techniques of face detection can be divided into four categories [23]:

1.Knowledge-based methods.

2.Feature invariant approaches.

3.Template matching methods.

4.Appearance-based methods.

Of course, many face detection methods are combination of several categories, so cannot simply be attributed to one category in the above. Some researchers also simply divided face detection into two categories [24]: Feature-based approaches and Image-based approaches. Feature-based approaches use some feature of face as its minimum processing unit. Image-based approaches use the pixels of image as its minimum processing unit. Next, I will use the classification of Yang Ming-Hsuan [23] to introduce and analysis those four face detection technologies.

2.3.1.Knowledge-based methods

Knowledge-based methods encode the features of typical human face into some rules. Generally, these rules include knowledge of the relationships between facial features [23]. The knowledge of the human face can be summarised into the following few simple rules:

1. Contours of face

The contours of the face can be approximated as an ellipse, so face detection can be formulated as ellipse detection. Goyindaraju had proposed a cognitive model, the human face is modelled as two lines (the right and left cheek) and two arcs (head and chin). Detect the straight line and arc by amending the Hough transform [25].

2. Facial organ distribution

Faces vary from person to person, but they all follow a few of general rules, the geometry of facial organ distribution rule. For example, a face in an image often includes two eyes, a nose and a mouse, these organs always are in a similar geometric distribution. The distances and positions between those organs can be the features to define a face [23].

3. Symmetry

The human face has a certain degree of axial symmetry, each organ also has certain symmetry, and this feature can be a rule to detect a face. In order to detect the human face in image, Zabrodsky [26] had proposed a continuous symmetry detection method to detect the symmetry of a circular area.

In addition, the colour and texture of human skin and the human movement in the background also can be used as a rule in knowledge-based method.

Recently, the knowledge-based method is no longer as the focus of research, because the problem of this approach is difficulty in translating human knowledge into well-defined rules. If the rules are too strict, a lot of faces may not pass the rule, and if the rules are too general, much error detection may happen [23].

2.3.2.Feature invariant approaches

There are some structural features that still exist irrespectively of the position, angle and lighting change in the face image. The goal of feature invariant approaches is find these features and use these features to detect the face [4]. Human beings can effortlessly “see” the faces and objects in a different light and posture, so the researchers believe that there is a potential assumption: there are some features and characteristics, which do not depend on outside conditions. Feature invariant approaches are in accordance with the underlying assumption, first to find this facial features (by analysis large number of samples), and then use these features to detect the face [23].

The colour of human skin has proven to be an effective feature for face detection [23], because skin colour is gathered in a small colour space. The result may not be accurate if only detected by skin colour, but as a rough location of entire system, it is intuitive, simple and fast, which can create a good condition to next precise position.

The approach used in this project namely Viola-Jones Object Detection [29] is based on feature invariant. The next section will describe the detail process of this approach.

2.3.3.Template matching methods

The template matching method is a classical pattern recognition method. The process is as follows [28]:

1. Pre-treat a standard human face. Generally, the pre-treatment will do two steps of scale normalise and grey normalise. A simple model template face as an ellipse and a more complex model template face which shown in **Figure 6**.
2. Calculate the relevant value between the input image and the standard face. The relevant values are independently calculate by the face, glasses, nose, mouth, then comprehensive them together. In the **Figure 5**, the standard face is divided into 16 areas with 23 kinds of connections, these 23 connections are used to calculate the relevant value.
3. In accordance with the relevant values and pre-set threshold value to determine if someone's face is in the image.

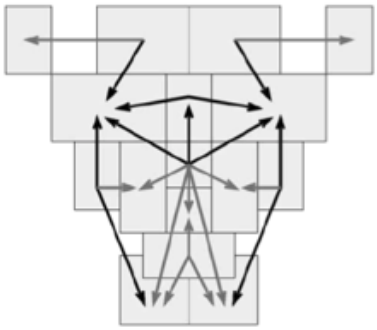


Figure 6. The template is composed of 16 regions (the grey boxes) and 23 relations (shown by arrows)[28]

The Template matching method is a mature method, its implementation is relatively simple, but this method is not very efficient for face detection [23].

2.3.4.Appearance-based methods

Contrasted to the Template matching method in which the templates are defined by experts, the templates in appearance-based methods are learned from some example images. Generally, appearance-based methods rely on statistical analysis and machine learning to find the relevant characteristics of face images and non-face images. The learning result is a distribution models or discriminant functions. These models and functions are used to detect an unknown image. Meanwhile, dimensionality reduction is usually used to increase the efficiency [23].

Many appearance-based methods can be understood as a probabilistic theory. A feature vector derived from an image is views as a random variable ***x***, this random variable ***x*** can be related to face and non-face by conditional density function $p(\boldsymbol{x}|\textit{the faces})$ and $p(\boldsymbol{x}|\textit{non-faces})$.

The position of face or non-face in the unknown image can be found by the Bayesian classification or maximum likelihood method. Unfortunately, it is not feasible by simply application of Bayesian, because [23]:

1. $\boldsymbol{x}$ is high dimensionally.
2. $p(\boldsymbol{x}|\textit{the faces})$ and $p(\boldsymbol{x}|\textit{non-faces})$ are multimodal.
3. It is not yet understood if there are naturally parameterized forms for $p(\boldsymbol{x}|\textit{the faces})$ and $p(\boldsymbol{x}|\textit{non-faces})$.

Hence, confirm parametric and nonparametric approximations to $p(\boldsymbol{x}|\textit{the faces})$ and $p(\boldsymbol{x}|\textit{non-faces})$ is an important part of appearance-based methods.

2.4. Introduction of Face Tracking

Face tracking is divided into single-face tracking and multiple face tracking. It mainly has three cases: The first one is that the face does not move, the camera moves; secondly, camera does not move, face moves; the third is the human face and camera both move. The first case is most constrained and third case is more universal. This section I will introduce some basic concepts of face tracking, which will used in my project.

2.4.1.Region-based Tracking

Region-based tracking method mainly depends on tracking of regional block to accomplish the tracking. How to hand the shadow of the moving target and block is a difficult problem in this method. A region tracking method proposed by Meyer and Bouthemy [33], uses the affine model of the density flow field to segment the movement and demarcate the regional border. This algorithm uses two Kalman filters, one motion filter is used to track the movement of the affine model, another geometry filter used to track the regional boundaries.

2.4.2.Template Matching Based Tracking

The template matching based tracking uses the template to indicate the tracking target, and then tracks the template in the image sequences. In the early year of this area the template was rigid, however, in practical applications, the tracking target is not always a rigid object, and the exact geometric template of the target object is not easy to obtain. Deformed template can be deformed to match the target shape, it is more flexible to tracking non-rigid object, so now we mainly use the deformable template. There are

two type of deformed template: non-parametric deformed template and parametric deformed template.

The typical Snake algorithm is a non-parametric deformed template [35]. The parametric deformed template has a better performance in face tracking. Liang Wang [34] first uses a used motion detection method to decrease the range of searching template. Then matching the correlation between template and object, and dynamic adjust the template during the tracking.

2.4.3.Feature Point Based Tracking

Feature point cased tracking includes two processes: feature points selection and features matching. In the area of face tracking, a rectangular box can close the face image, the centroid of this box can be chosen as the feature point. In the tracking process, if two faces overlap, as long as the speed of the centroid can be separated, the tracking can still be successfully implemented. The advantage of this method is simplicity, and the use of facial movement to solve the blocking problem.

2.5. Introduction of Robotic Morphology

There are four common robot movement structure: cartesian coordinate type, cylindrical coordinate type, polar coordinate type and joint type. The principle of designing a robot is to choose the simplest movement structure to meet the need of robot with least **DOF** and. (**DOF**: Degrees Of Freedom is the minimum number of coordinates required to represent the range of system motion.)

The samples of those four movement structure types shows below:

a. Cartesian coordinate robot

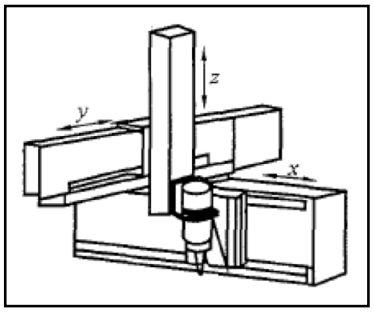


Figure 7. Cartesian coordinate structure[8]

Cartesian coordinate robot has simple structure (**Figure 7**) and high stability. There are three straight moving directions, *x, y* and *z*. Those three moving directions are mutually independent with no coupling, so the physics calculation of this structure is simple.The disadvantage of this structure is need a large area, low range of motion, and inability to achieve some complex actions.

b. Cylindrical coordinate robot

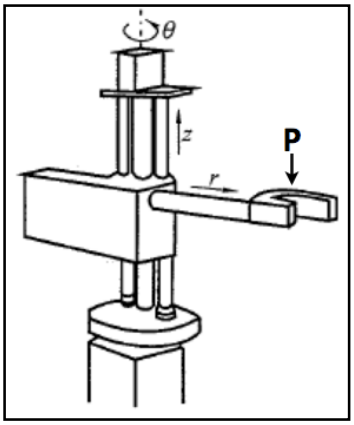


Figure 8. Cylindrical coordinate structure[8]

Figure 8 is a structure of robot arm which use the cylindrical coordinate. Cylindrical coordinate robot use θ, z and r as parameters to constitute the coordinate system. The coordinate position of the robot arm **P** can be expressed as **P** = f (θ, z, r), where r is the radial length of the arm, and θ is the length of arm in the direction of radius, z is height of the vertical axis. This structure need small space and simple structure. And the telescopic and lift of arm is conducive to reduce the inertia. The disadvantage of this structure is also cannot achieve some complex actions (can be improved, if increasing the degree of freedom).

c. Polar coordinate robot

This structure is a spherical surface which is formed by rotate by a fulcrum. The radial of this spherical surface is r. θ and ϕ are the angle parameter, a point **P** which locate in the spherical surface can be expressed as **P** = f (θ, ϕ, r). It is difficult to control this kind of robot, so such a structure of the robot is less common.

d. Joint robot

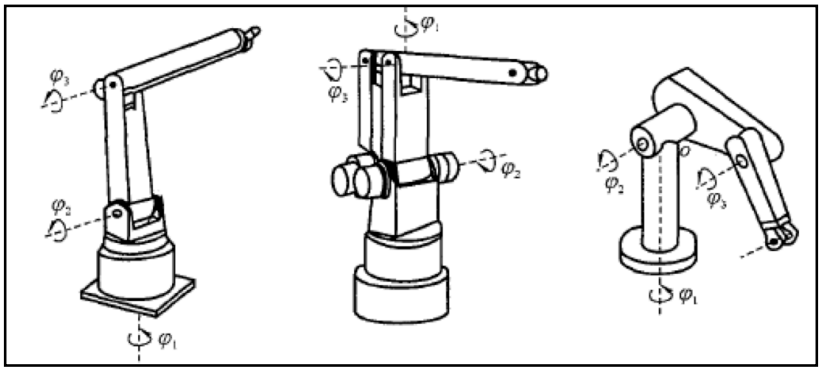


Figure 9. Three kind of joint structures[8]

Figure 9 shows three kind of joint structures. This structure is designed to simulate human's waist joint, shoulder and elbow joint, which is widely used in anthropomorphic robot. It is a compact structure, need small space and can achieve flexible action. The biggest drawback is the complex physics calculations and need large computation.

3.Design and Implementation

3.1.General Design

A simple general flow chart of this robot system is shows below:

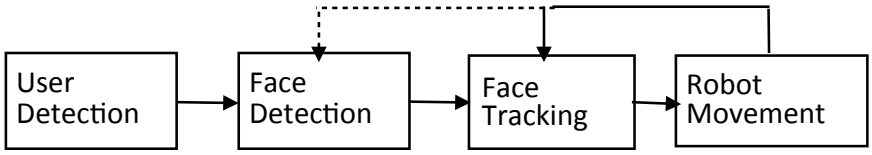


Figure 10. Simple general flow chart

This chart can split the system into Four modules: User detection, Face detection, Face tracking, Head movement. This section will talks the general design of the system first, then introduces the specific implementation of each modules with the following order.

1. **User detection:**

   Robot detect its surround environment, and start the system when someone is closing. In this robot system will use the distance between the user and robot as the signal to determine whether a potential user is coming. If the distance is less than a threshold value, the next step face detection module will be started.

   This project will use an ultrasound sensor to detect the distance. A more specific implementation will be shown in section 3.2.

2. **Face detection**

   Rise the robot camera, start to catch the video, this module will try to detect the face which appear in the screen, until a stable face is found, then return the location information to face tracking modules.

   In section 3.2, an AdaBoost based object detection algorithm is used to achieve the function of face detection.

3. **Face tracking**

   Receive the location information and track the object (user's face) in the location, and if the face tracking module can track the object, then it will sent the track information (a coordinate) to head movement modules. The dashed arrows in the **figure 10.** means if the robot lost the target during the tracking, the system will restart to face detection.

   Two tracking algorithms, which were tested in this project are CamShift Tracking and TLD. Two versions of robot system were built base on these two algorithms. Two algorithms have their own advantages and disadvantages. The comparison and evaluation will be discussed in section 4.

4. **Robot movement**

Receive the tracking information, if the object is not located in the centre of video, move the camera. The challenge of this module is design a structure which can ensure the accuracy and stability during the rotation.

3.2.User Detection

In order to wake the system up from an idle state, there should be a signal to measure if a potential user is coming. In the design stage, I found out some solutions to solve this problem. For example, use the face detection during the idle state, if found a face size is large enough, that means somebody is very close to the robot, then start the system. But this method will waste a lot of computing, and it is hard to choose the start face size. Or use the sound as a key of start. The system will be turn on if a user says 'Hello','How are you' or other similar words. This method is interesting but not robust, because the user need to know the keyword in advance and it is easy to be confused by background sound. Finally, the automatic doors of the supermarket gave me the inspiration, the distance can be a signal of user detection!

The automatic door use the inductor(mainly are infrared sensors and microwave sensors) to detect if someone comes. Lego NXT Robot provides an ultrasound sensor which can detect the distance by ultrasound. The principle of Ultrasonic Ranging had been mentioned in section 2.2. This sensor can be perfectly used in my project. So I will introduce this ultrasound sensor and design an experiment to determine how much distance is more accurate.

3.2.1.Lego NXT Ultrasonic Sensor

Lego NXT Ultrasonic Sensor is one of sensors used by NXT robot. This ultrasonic sensor to just like the 'eye' of robot, which can measure the distance between the obstacle, and help the robot detect the surrounding environment. The picture of this sensor shows below.

Figure 11. Ultrasonic Sensor

We can see from **Figure 11** the ultrasonic sensor consists of two parts. One is an ultrasonic receiver and the other one is an ultrasonic sender (the orange circle in the picture). According to the official data[20], this ultrasonic sensor can measure distance in centimetres and in inches. It can detect the distance 0-255cm with error of +/-3cm, and the range of object detection is 150 degrees. In actual use, the data measure measurement of this sensor is not very accurate and sometimes with fluctuations, so I designed an experiment to found a reliable value of this sensor to make the system more robustness.

3.2.2. Implementation

The NXT Ultrasonic Sensor will return a centimetre number between 0 to 255. If the sensor did not detect anything or the distance of obstacle is more than 255cm, it will return 255. That means If the return value is less than 255 we can believe that it is reliable. There are mainly two lines of code had been used in the program:

```
UltrasonicSensor ultrasonicPort = new UltrasonicSensor(SensorPort.S1);
ultrasonicPort.getDistance();
```

The first line is declare a new ultrasonic sensor, and the second line, the function of gerDistance() will return the distance.

During the using, the sensor sometimes cannot detect an accurate distance.In order to find a best detect distance I did a few experiments and record the correct rate at different distances.

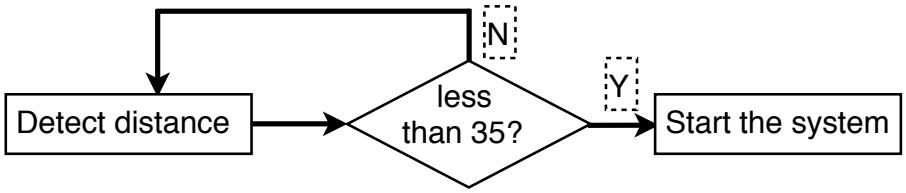
I chose a few of data which from 10cm to 35cm, and simulate the user close to the robot with this distance, record and calculate the correct rate for each value. Each distances test 20 times. And the result is shown below:

actual distance	10cm	15cm	20cm	25cm	30cm	35cm	40cm
Detection rate	100%	100%	100%	90%	80%	90%	75%
Detection distance(Avg.)	21.6	21.1	23	26.1	32.1	36.3	42.9
Error	+11.6	+6.1	+3	+1.1	+2.1	+1.3	+2.9

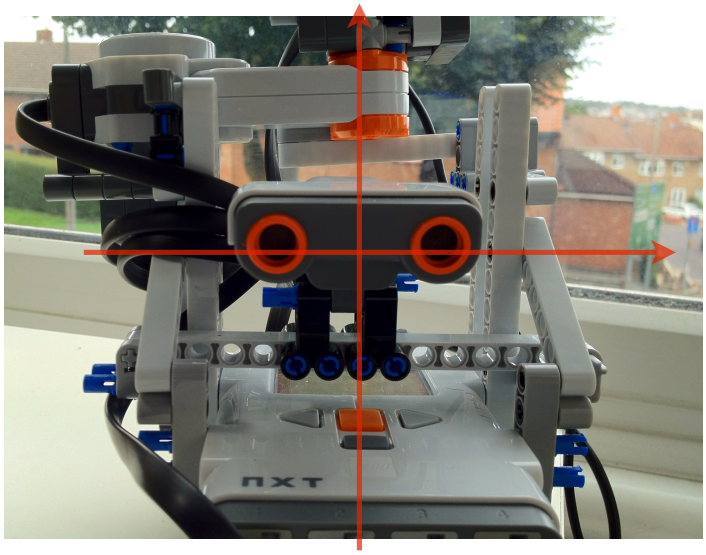
The Detection rate = the times of legal return value(less than 255)/the times of experiment

From the form we can find that the robot can set a better detection rate in short distance. And got the lowest error in 15cm and 35cm. Taking into account that the 30cm is larger than 25cm and 90% detection rate is also acceptable, so finally the detection distance is 35cm in the program.

The **main flow chart** of the user detection part is shown below:



The **picture** of ultrasonic sensor in the robot:



The ultrasonic sensor is located in the middle of robot and it is perpendicular to the robot body. This structure can make sure robot can detect the user's face when the system started.

3.3.Face detection

This project use the AdaBoost algorithm and Haar Cascade Classifier to achieve a real-time face detection system, which is proposed by Paul Viola and Michael Jones [27,29].

3.3.1. General Structure

This method mainly including the integral image, cascade classifier, AdaBoost algorithm, the general framework can be divided into the following three parts [4]:

1. Use the Harr features to represent the face, use the integral image to achieve the fast calculation of the characteristic values;
2. Use the AdaBoost algorithm to pick out some of the most representative rectangular features (weak classifiers), construct a strong classifier with some weak classifiers by the weighted vote method.
3. Combine some strong classifiers into a cascade classifier, the cascade structure can improve the speed of detection effectively.

3.3.2. Introduction of AdaBoost algorithm

Freund and Schapire [30] proposed the AdaBoost algorithm at the year of 1995. The **AdaBoost** full called **Ada**ptive **Boost**ing, the reason of author called it AdaBoost is because this algorithm is pretty different from the Pre-Boosting algorithm (the original Boosting algorithm needs to know the hypothesis minimum error rate in advance), AdaBoost is automatically adapt the error rate by weak learners, that means, the AdaBoost algorithm does not require any priori knowledge of weak learners, and it has similar efficiency with original Boosting, so it can be easily applied to practical problems.

Weak Classifier and Strong Classifier

Random guessing a yes or no problem, there will be 50% corrective rate. If a hypothesis can slightly increase the probability of corrective rate, then this assumption is the **weak classifier**, and the process of this classifier is called weak learning. If a hypothesis can significantly increase the probability of corrective rate, then this assumption is called the **strong classifier**. It is easy to create the weak classifier by experience, but hard to create strong classifier. The method of AdaBoost to create strong classifier is to combine many weak classifiers [29].

AdaBoost detect objects by classifier, the main idea is: given a higher weight to a better performance weak classifier and given a lower weight to a poor performance weak classifier. Singled out a number of key weak classifier and combine into a strong classifier to achieve better classification results.

Kearns and Valiant had proved [31]: If sufficient data has been provided, the assumption of weak learning algorithm will be able to generate arbitrary precision assumption (strong classifier). This proves can support the AdaBoost algorithm.

Firstly, Freund and Schapire [30] describe the general structure of the AdaBoost algorithm. Viola and Jones [27] proposed a modified AdaBoost algorithm to the specific application for face detection. Viola's algorithm makes weak classifier and weak feature (one of optional feature) as equivalent. Each weak classifier is only use one feature, and combines the weak classifiers by AdaBoost algorithm. In order to detect face, Viola defines a large number of rectangular features, which are called Harr characteristics (can be extracted more than 100 thousands of features from a 20 * 20 image).

3.3.3.Harr Features

Haar features can be divided into three categories: **edge features**, **linear features**, and **diagonal features**. Haar features are computed on the basis of rectangular features of image, so we also call them rectangular features.

Haar features are sensitive to simple graphical structures (such as edges, line segments), but it can only describe the specific trend structure (horizontal, vertical, diagonal), which shown in **Figure 10**, some of the characteristics can be described haar features, for example, usually the colour of eyes is deeper than the cheek, the colour of nose is deeper than its both sides, the mouth is darker than its surrounds.

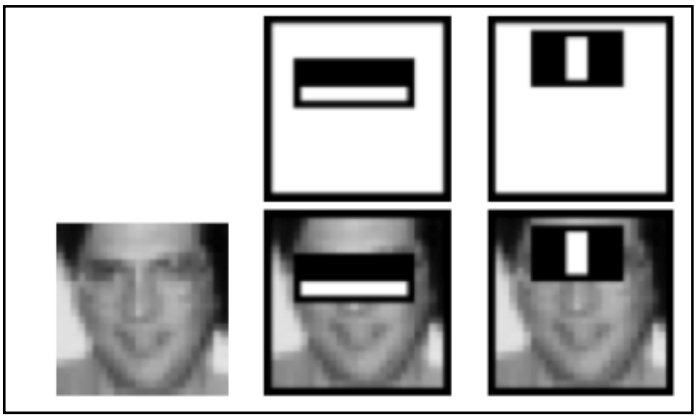


Figure 12. Harr features, Figure from [29]

For a 24 × 24 images, there are more than 160,000 harr features inside, so we must select some suitable features, and combine them into a strong classifier.

Viola and Jones use a simple combination of rectangles as harr features, and they are all combined by two or more congruent rectangles. The rectangles are drawn with white or black, and define the upper-left rectangle is white. **Characteristic value is defined as the difference of white rectangular pixels minus black rectangular pixels.**

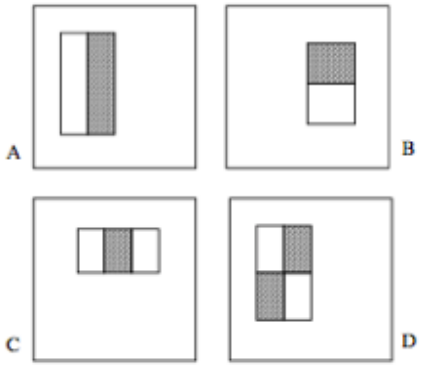


Figure 13. Four example rectangle features, Figure from [29]

Four example harr features are shown below: A and B are the **edge features**, C is the **linear features**, and D is the **diagonal features**.

Condition Rectangle

The harr features can be placed in an image by “any” size and “any” position. Therefore we need to identify all possible positions of the features.

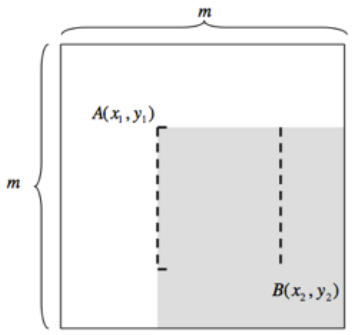


Figure 14. Rectangle features in a $m \times m$ image

For example, there is a *m×m* image shown in **Figure 14**, a upper-left vertices ***A(x₁, y₁)*** and a lower right vertices ***B(x₂, y₂)*** can only determine a rectangle, the length of the rectangle is ***y₂ - y₁*** and the width of the rectangle is ***x₂ - x₁*** , if this rectangle also meet the following two conditions (referred to **(s, t) conditions**), we call it **condition rectangle** :

1), The length of the rectangle **(*x₂ - x₁*)** can be divided by the natural numbers **s** (can be divided into **s** equal segments);

2), The width of the rectangle **(*y₂ - y₁*)** can be divided by the natural numbers **t** (can be divided into **t** equal segments);

So the minimum size of this rectangle is the **s × t** or **t × s**, the maximum size is **[m/s]·s × [m/t]·t** or **[m/t]·t × [m/s]·s**. Where '**[]**' is the rounding operator.

In fact, **(s, t) conditions** describes the characteristics of rectangular features, **Figure 15** listed the **(s, t) conditions** of the example rectangle features:

Rectangle Features				
(s, t) conditions	(2,1)	(1,2)	(3,1)	(2,2)

Figure 15 (s, t) conditions of example rectangle features

The Number of Condition Rectangles

We can locate a condition rectangle by the following two steps [32]:

1) Select ***A(x₁, y₁)*** where $x_1 \in \{1,2, \cdots, m-s, m-s+1\}$, $y_1 \in \{1,2, \cdots, m-t, m-t+1\}$;

After ***A*** has been selected, ***B(x₂, y₂)*** can be only located in the shaded of **Figure 14** (including marginal), the condition of ***B*** is :

$$x_2 \in X=\{x_1+s-1, x_1+2\cdot s-1, \cdots, x_1+(p-1)\cdot s-1, x_1+p\cdot s-1\},$$

$$y_2 \in Y=\{y_1+t-1, y_1+2\cdot t-1, \cdots, y_1+(q-1)\cdot t-1, y_1+q\cdot t-1\},$$

Where $p=\left[\frac{m-x_1+1}{s}\right], q=\left[\frac{m-y_1+1}{t}\right]$. And $|X|=p, |Y|=q$.

So, in a *m×m* image, the number of condition rectangular which satisfy **(s, t) conditions** can be summarized as a formula:

$$\Omega^{m}_{(s,t)} = \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} p \cdot q$$
$$= \sum_{x_1=1}^{m-s+1} \sum_{y_1=1}^{m-t+1} [\frac{m-x_1+1}{s}] \cdot [\frac{m-y_1+1}{t}] = \sum_{x_1=1}^{m-s+1} [\frac{m-x_1+1}{s}] \cdot \sum_{y_1=1}^{m-t+1} [\frac{m-y_1+1}{t}]$$
$$= \left(\left[\frac{m}{s} \right] + \left[\frac{m-1}{s} \right] + \cdots + \left[\frac{s+1}{s} \right] + 1 \right) \cdot \left(\left[\frac{m}{t} \right] + \left[\frac{m-1}{t} \right] + \cdots + \left[\frac{t+1}{t} \right] + 1 \right)$$

3.3.4.Integral Image

To rapidly compute the rectangle features Viola and Jones [29] first use **integral image,** which defined as:

A(x, y) is a point in the integral image (shown in **Figure 16**), the integral image *ii(x,y)* is defined as below:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

Where *i(x',y')* is the colour value of position *(x',y')* in the original image; if it is a grey image, the value range is 0 to 255; if it is a colour image we should change it to grey image.

We can get an integral image by only a small amount of computational work (as shown in next section). One image only corresponding to one integral image, integral image can use to calculate different rectangle features in same time, thus it can improve the detection speed greatly. So, firstly, I will simply introduce the calculation method of integral image.

Calculate Integral Image

According the definition formula of integral image, *ii(x,y)* can also be obtained by the following iterative formula [29]:

$$ii(x, y) = ii(x-1, y) + s(x, y)$$
$$s(x, y) = s(x, y-1) + i(x, y)$$

Where *s(x,y)* is the sum of grey value on the *y* direction (**Figure 16**), *s(x,0)=0, ii(0,y)=0*.

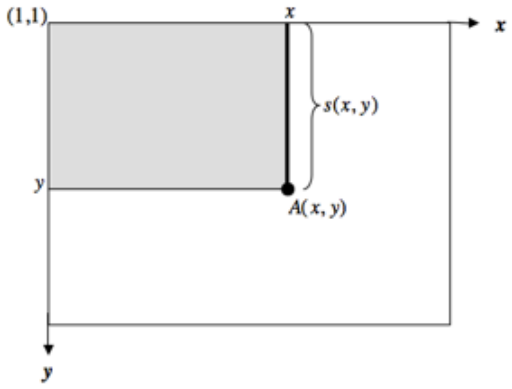


Figure 16 The value of Integral Image in *A(x, y)* is the sum of its the upper-left rectangle's grey value (the shaded in figure). *s(x,y)* is the sum of grey value on the *y* direction (the thick black line in figure).

If the image size is *m*×*n*, the integral image is:

$$\begin{pmatrix} ii(1,1)=ii(0,1)+s(1,1) & ii(2,1)=ii(1,1)+s(2,1) & \cdots & ii(m-1,1)=ii(m-2,1)+s(m-1,1) & ii(m,1)=ii(m-1,1)+s(m,1) \\ s(1,1)=s(1,0)+i(1,1) & s(2,1)=s(2,0)+i(2,1) & & s(m-1,1)=s(m-1,0)+i(m-1,1) & s(m,1)=s(m,0)+i(m,1) \\ ii(1,2)=ii(0,2)+s(1,2) & & & & ii(m,2)=ii(m-1,2)+s(m,2) \\ s(1,2)=s(0,1)+i(1,2) & & \ddots & & s(m,2)=s(m,1)+i(m,2) \\ \vdots & & & & \vdots \\ ii(1,n-1)=ii(0,n-1)+s(1,n-1) & & & & ii(m,n-1)=ii(m-1,n-1)+s(m,n-1) \\ s(1,n-1)=s(1,n-2)+i(1,n-1) & & & & s(m,n-1)=s(m,n-2)+i(m,n-1) \\ ii(1,n)=ii(0,n)+s(1,n) & ii(1,n-1)=ii(0,n-1)+s(1,n-1) & \cdots & ii(m-1,n)=ii(m-2,n)+s(m-1,n) & ii(m,n)=ii(m-1,n)+s(m,n) \\ s(1,n)=s(1,n-1)+i(1,n) & s(1,n-1)=s(1,n-2)+i(1,n-1) & & s(m-1,n)=s(m-1,n-1)+i(m-1,n) & s(m,n)=s(m,n-1)+i(m,n) \end{pmatrix}$$

So, obtain the integral image only need to traverse the original image once and calculate *m*×*n*×*2* times.

Calculate the Integral Value in One Area

In the **Figure 17**, the sum grey value of **D** area can be calculated by integral value of location 1,2,3 and 4.

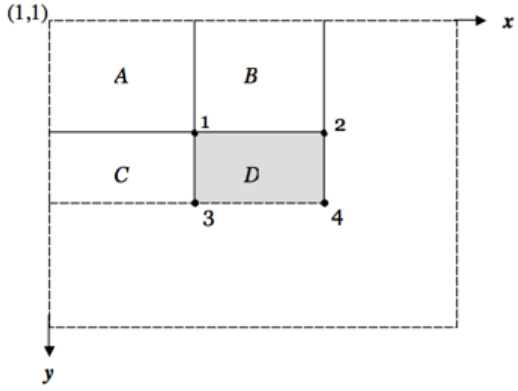


Figure 17. The sum grey value of **D** area.

Explain:

ii_1 is the sum of grey value in rectangle A, ii_2 is the value of A+B, ii_3 is the value of A+C, ii_4 is the value of A+B+C+D.

So, the sum with D is equal to $ii_4 + ii_1 - (ii_1 + ii_3)$.

Calculate the Characteristic Values of Rectangle Feature

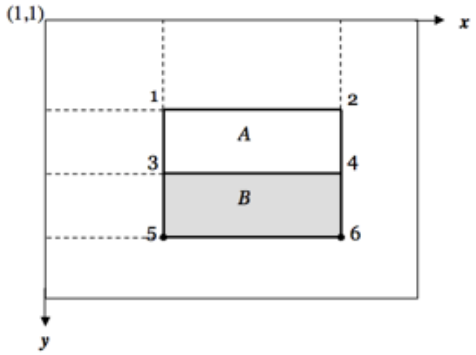


Figure 18 Characteristic values are only related to the endpoints of this rectangle feature.

The example rectangle feature A (**Figure 13**) is shown in **Figure 18**. According the definition of section 3.3.3. the characteristic value is:

The sum of pixels in area A - The sum of pixels in area B =

$$ii_4 + ii_1 - (ii_2 + ii_3) - (ii_6 + ii_3) + ii_4 + ii_5$$

Thus, characteristic values are only related to the endpoints of this rectangle feature. Therefore, no matter whatever the size of rectangle feature, the time spent in the calculation is constant, and it is simple addition and subtraction. For this reason, the integral image enhances the speed of detection greatly.

3.3.5.AdaBoost Training Algorithm

AdaBoost is an iterative algorithm, which trains a lot of different **weak classifiers**, then combines these weak classifiers together as a **strong classifier**. This section will introduce the process of AdaBoost Training Algorithm. The definition of **weak classifier** and **strong classifier** had been mentioned in section 3.3.2. First is the introduction of **weak classifier**.

Weak Classifier

A weak classifier ($h(x, f, p, \theta)$) consists with a feature function f, a threshold θ and a polarity p indicating the direction of the inequality [29]:

$$h(x,f,p,\theta)=\begin{cases}1 & if\ \ pf(x)<p\theta \\ 0 & otherwise\end{cases}$$

where **x** is an input image.

Characteristic value function ***f(x)***:

The output of function f(x) is the characteristic value of a rectangle feature, the characteristic value has been define in the section 3.3.3 . During the training, the size of detect window is equal to the size of training image. The size of detect window decide the number of rectangle features, each example of training set has same size, so they have same number of rectangle features (3.3.3). For example, according the Condition Rectangles number formula, a 20×20 image has 78,460 rectangle features.

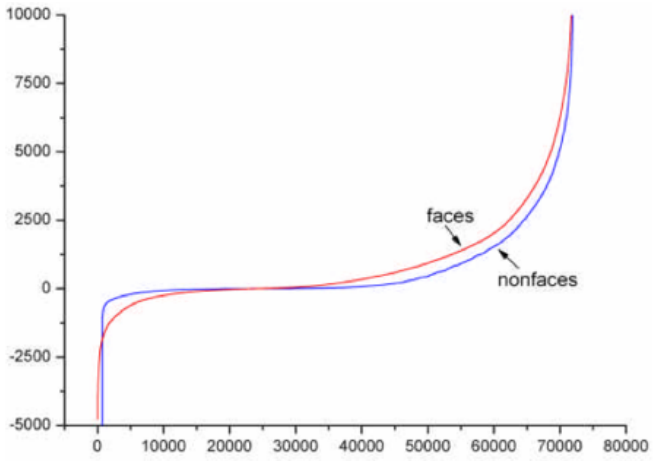


Figure 19 The average characteristic value distribution of all rectangle features (The horizontal axis is the number of the rectangular features). Figure from [32].

To each rectangle feature, calculate the average characteristic value of training set (include the face image and non-face image), and get the average distribution shown in the **Figure 19**. The distribution shows that most characteristic value of rectangle feature is distributed in the range of near 0. And there is only little difference in the distribution line between face images and non-face images. But when the characteristic value larger or less than a threshold value, the distribution lines have a consistency of difference. That means most features have weak ability to recognize the face image and non-face, but there are still some features and the threshold value can help us to divide the face and non-face effectively.

The threshold θ and polarity p :

A weak classifier has only two outputs, 0 or 1. An image input to the $f(x)$, and output a characteristic value, θ is the threshold value to distinguish the image belong to face image or non-face image. And polarity p is indicating the direction of the inequality.

The requirement of a weak classifier is: to distinguish face and non-face by slightly less than 50% error rate. Almost all of the rectangle features can satisfy the requirement by set an appropriate θ.

How to find the optimistic threshold θ to a weak classifier?

Each image for the training set has a weight w, which is the importance of this image. The purpose of training weak classifier is to find a θ, which can get the lowest error rate base on current weight distribution.

The first step in the algorithm is to sort the characteristic values of all training set by a rectangle feature. The optimistic threshold θ can be obtained by scan the sorted list once. Specific needs to calculate the following four sums [29]:

1). T^+ is the total sum of face example weights.

2). T^- is the total sum of non-face example weights.

3). S^+ is the sum of face image weights below the current example.

4). S^- is the sum of non-face weights below the current example

Choose the threshold θ between maximum and minimum of characteristic value list. So the weak classifier can distinguish the face and nun-face example, the characteristic value of image larger than threshold θ classified as face (or non-face), the characteristic value of image less than threshold θ classified as non-face (or face). The direction depends on the polarity p.

The error for this threshold, which splits the range between the current and previous example in the sorted list is [29]:

$$e = \min\left(S^+ + \left(T^- - S^-\right), S^- + \left(T^+ - S^+\right)\right)$$

Thus, scan the sort list can choose the threshold θ, which can get the minimise error rate. After determining the value of θ, the training of weak classifier is completing.

Strong Classifier Training Algorithm

The strong classifier training algorithm can be described shown below [29]:

A. For a given example set $\left(x_1, y_1\right), \left(x_2, y_2\right), \ldots, \left(x_n, y_n\right)$ where $y_i = 0$ for negative examples (face), $y_i = 1$ for positive examples (non-face). n is the totally number of examples.

B. Initialise weight $w_{1,i} = D(i)$, where $D(i) = \frac{1}{2m}$ (negative examples), $D(i) = \frac{1}{2l}$ (positive examples), m and n is the number of negative and positive example respectively and $n = m + l$.

C. For $t = 1, \ldots, T$:

1. Normalise the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$.

2. Training a weak classifier for each rectangle features. And calculate the weighted error rate e_f :

$$\varepsilon_f = \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|.$$

3. Choose the optimistic weak classifier $h_t(x)$, which has a minimum error rate:

$$\varepsilon_f = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|.$$

4. Update the weights base on the optimistic weak classifier:

$$w_{t+1,\,i} = w_{t,\,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta = \frac{\varepsilon_t}{1-\varepsilon_t}$.

D. The final strong classifier is:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0 & \text{otherwise} \end{cases}$$

where $a_t = \log \frac{1}{\beta_t}$.

Strong Classifier

After T times of iteration, we obtain T optimistic weak classifier $h_1(x), ..., h_T(x)$, which can composite a strong classifier by the following combinations [29]:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0 & \text{otherwise} \end{cases},$$

where $a_t = \log \frac{1}{\beta_t} = \log \frac{1-\varepsilon_t}{\varepsilon_t} = -\log \varepsilon_t$.

Use this strong classifier to detect an image is equal to let all weak classifiers to vote, then sum the voting results by the weight of each weak classifier. And get the final result by compare the sum and the average voting results.

The average voting results means assume each weak classifier has same probability to vote 'agree' or 'disagree', $\frac{1}{2}\left(\sum_{t=1}^{T} a_t \bullet 1 + \frac{1}{2} \sum_{t=1}^{T} a_t \bullet 0 \right) = \frac{1}{2} \sum_{t=1}^{T} a_t$, the average probability is the average voting results.

The Attentional Cascade

AdaBoost algorithm can obtain a strong classifier with high detection rate, but the single classifier stronger the detecting time spends more. In order decrease the time cost,

Viola [29] proposed a cascade of classifiers which achieves increased detection performance while radically reduction computation time.

The cascade classifier is a combination of the strong classifiers. As the **Figure 18** shows, each level of cascade classifier is a strong classifier (the circle 1,2,3 in the figure), which is calculated by AdaBoost algorithm. Making each strong classifier can detect almost all the face examples and deny a large part of non-face samples by adjust the threshold θ.

Moreover, the previous layer will use less number of rectangle features to make the calculation very fast. And the higher layer the less candidate matching images. Despite the rectangle features increased makes the time of single image calculation longer, but in the actual testing, the number of input image which will through the behind level is very small. So the number of sub-window, which cause all layers (even able to reach behind layers) have calculated is very small.

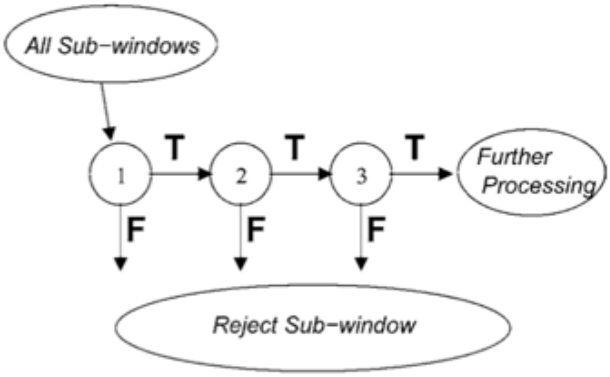


Figure 20 The schematic of cascade classifier. Figure from [29]

Similar with the decision tree, the each layer of cascade classifier is constructed by all training examples. The strong classifier in the current layer should face a more difficult image, which cannot be distinguished by the former layer. And we can change the weights of each example image to make the training of behind layer has more target to these undetected image.

3.3.6.Implementation

Some assumptions on user behaviour

1. Users will face toward to the robot.

   The object detection classifier is trained by one kind of object, in this project the classifier is trained by front face, because most people who want to interact with robot may face to the robot that means the camera will capture user's front face.

2. A more relevant possible user is closer to the robot and has a bigger face in video.

   Currently, the robot is not intelligent enough to distinguish the people's intent by facial expressions and demeanour. So multiple faces in the video will lead to a problem, who is the person the robot need to track. An imprecise method is used to solve the problem. Robot will default the person who has a biggest face in the video as the tracking target. Because the person who stand most close to the

robot may have most potential to be the target. And the size of human face is similar, so the closer person has a bigger face in the video.

The choice of coding language

Two options had been consider to implement the AdaBoost Algorithm, which are Matlab and OpenCv. Both of them provide the library to achieve the Viola and Jones face detection method. As a scripting-style language Matlab can easily implement the face detection module. On the drawback, it has less lower efficiency. OpenCv (Open Source Computer Vision Library) was established by Intel in 1999, and now it is supported by Willow Garage. OpenCV is a is lightweight and efficient Open Source Computer Vision Library, which consists of a set of C functions and a few C + + classes. It can implement many common algorithms for image processing and computer vision.

After considering the pros and cons of both OpenCv and Matlab, finally, OpenCv become the coding tool in this project, there are two mainly reasons for this choice: first is the speed, the robot should achieve the real-time detection and will be placed in a multi-people environment, the speed is a very important parameter for this system. The second reason is C/C++ is computing easier to achieve a cross-language programming than Matlab. Because in this project the robot's main body is build by Lego NXT, which only provide JAVA interface to control the robot. JNI(Java Native Interface) is a function of JAVA, it allow JAVA program use the library which is compiled by other language especially for C and C++.

OpenCv provide several classifiers which are trained by different objects. A frontal face classifier 'haarcascade_frontalface_alt.xml' is used in this project, it can already obtain a satisfactory accuracy. And a stable algorithm(which will be explained below) is used to increase the robustness of the system.

Sample of Face Detection

In order to check the accuracy of face detection, I wrote a face detection sample program for picture. The result is shown in **Figure 21**.

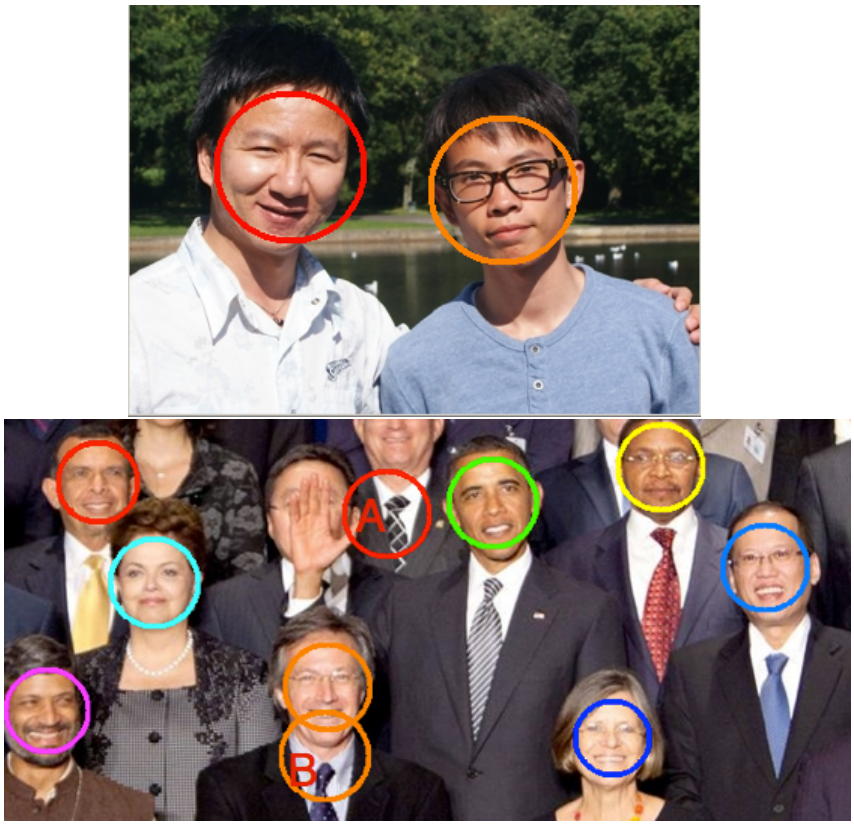


Figure 21. The result of face detection(the second picture comes from news.ifeng.com)

There are two pictures in the **Figure 21**, the first picture has only two faces, the program can find both two faces. And the second picture is more complex, we can find that all of the unblocked faces can be detected. But the program cannot detect the blocked or incomplete face in the picture. And some non-face area will be detected as a face (place A and B in the second picture).

According the test of Zhao[32], the accuracy of this algorithm is close to 90%, but always with some misdetection. After this test this classifier can satisfy the requirement of this project, and a stable algorithm will used to avoid the misdetection.

How to avoid the misdetection

The face detection cannot guarantee 100% to find all of faces in the video, sometime misdetection will happen. There are three screenshot during the program running in the **Figure 22**. The left picture is a correct detection, the program found the correct face position. No face had been found in the middle picture, and a wrong place was marked in the right picture.

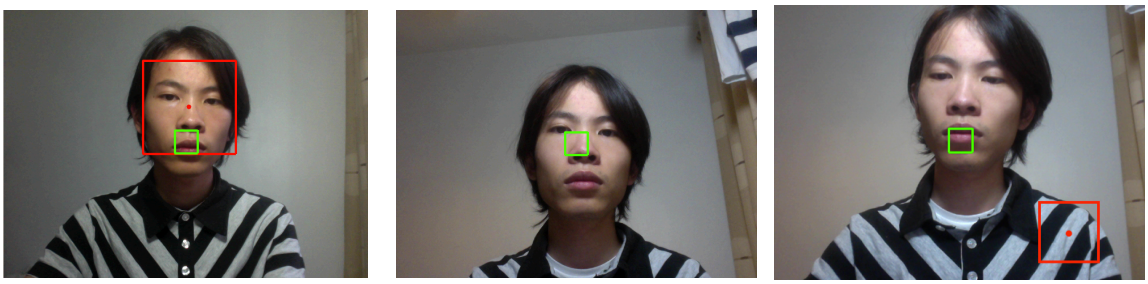


Figure 22. A correct detection and two misdetection (the Green Box is the centre of video, the red box is the detected face position, the red point is the middle of red box)

A stable algorithm had been announced to avoid the misdetection in this module. The processing unit of face detection module is a frame of video. If only use one frame to detect the face may easily get a wrong result. I found that most picture can be classified correctly, the mistake only happened in some special angle. And a user will not keep a stable pose for too longtime. So the detect result for a series of consecutive frames may like this: correct, correct, correct, correct, correct, misdetection, correct, correct, correct, misdetection, correct, correct, correct, correct... Misdetection will be interspersed among a series of correct result with a low probability.

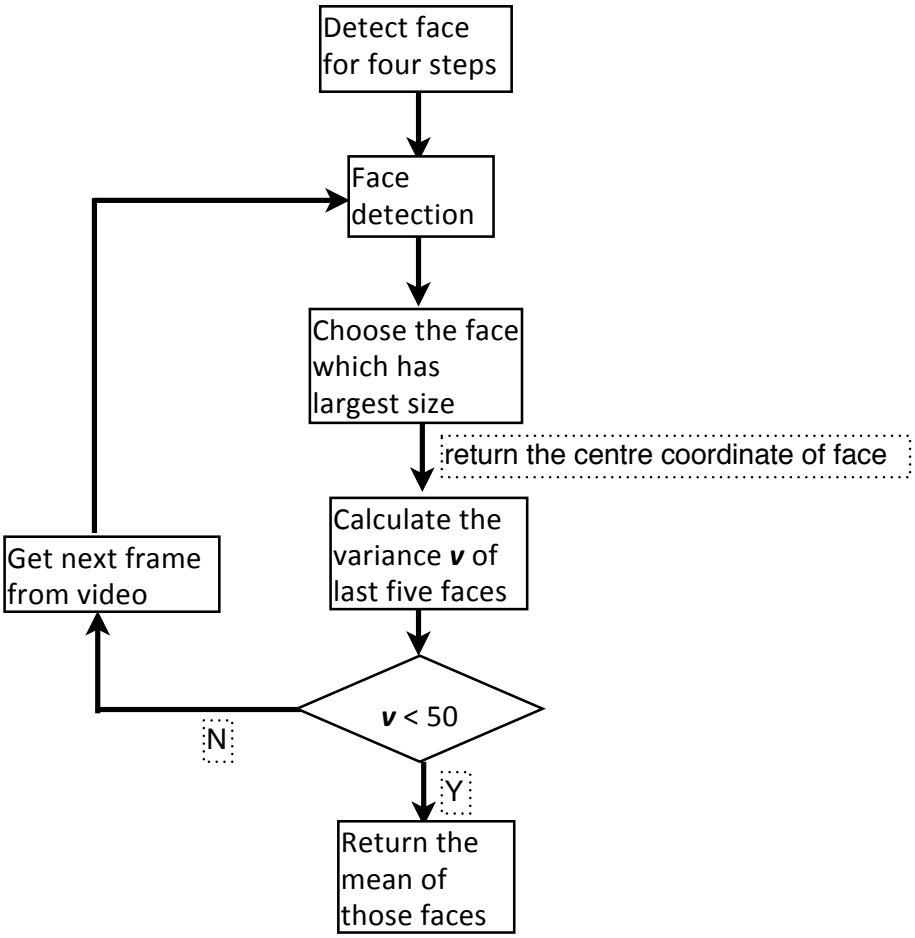
So the main idea of stable algorithm is: a face position is reliable only if several consecutive frames all detect a face in a close area. I use the variance of face position***(x,y)*** to measure the degree of stability between consecutive frames. If both variance $v(x_v, y_v)$ of face abscissa and ordinate are less than a threshold means get a stable face position. The $v(x_v, y_v)$ is defined as:

$$v_x = E\left(\left(X - E(X)\right)^2\right), X = x_1, x_2, x_3 \ldots x_n$$

$$v_y = E\left(\left(Y - E(Y)\right)^2\right), Y = y_1, y_2, y_3 \ldots y_n$$

Where, E(X) is the mean of X, *x* is the face abscissa of one frame, *y* is the face ordinate of one frame, *n* is the number of consecutive frames, the bigger value of n can get a better accuracy but will take more time.

In the program, n = 5 and the threshold = 50, the flow chart shows below:



So that we can evaluate this method, if the error rate of each frame is 10%, the wrong face position will be returned only under the situation that the misdetection continuous occurred five times. So the probability of this method returns a error result is very low. The time cost of process one frame is less than 0.05s. Actually, this algorithm can get a good performance to avoid the misdetection with low time cost.

3.4.CamShift Face Tracking

The CamShift(Continuously Adaptive Mean-Shift) Algorithm is first proposed by Gary R. Bradski[21] in 1998. CamShift is a tracking algorithm based on colours probability distribution, and its core idea is the Mean Shift iteration.CamShift extends the Mean Shift algorithm from a single image to a continuous video sequence frame images. CamShift algorithm uses the colour histogram of the target area, which is the H-component(Hue) of the HSV colour space. To each single image it uses the Mean Shift algorithm to find the destination window, then adjust the size of window. In a continuous video, and the detection result of last frame become the initial window of next frame to achieve continuous tracking.

In this section, I will introduce and analyse the CamShift Algorithm by three step (H-component Back projection, MeanShift and CamShift). And the implementation will be discussed in section 3.4.4.

3.4.1. H-component Back projection

The first step of CamShift is calculating the back projection image for each frame. The general process is: a.To each frame, transfer the image from RGB colour space to HSV colour space and extract the H-component(Hue). b.Create the H-component probability distribution histogram for the tracking area. c.Calculate the back projection image of this frame by this probability distribution histogram.

HSV colour space

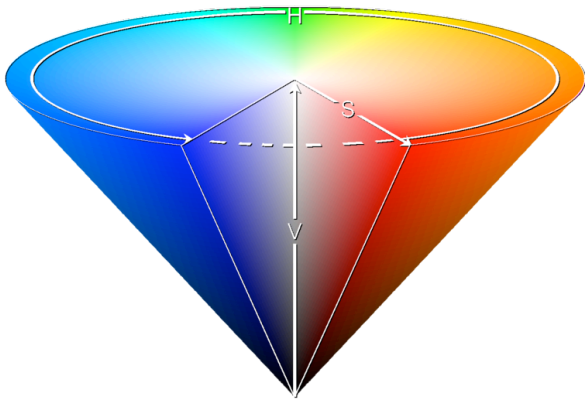


Figure 23. HSV Colour Space[22]

As shown in **Figure 23.**, HSV color space is a method to determines colour by three basic colour attributes: hue(H), saturation(S) and brightness/value(V).

Hue(H) is the basic attributes of colour, it is usually the name we call for a colour, such as red, yellow,blue. The value of H can be 0-360 which are all located in the H circle of **Figure 23**.

Saturation (S) refers to the purity degree of the colour, the higher value of S the colour is more pure, the lower value of S the colour is dimmer, S usually take a value of 0-100%.

Value(V) is also called brightness, it represent the degree of bright, usually take 0-100%.

Why choose HSV colour space?

RGB colour space is more sensitive to brightness. In order to keep the robustness of the algorithm, we need to reduce the affect of light brightness. In a variety of colour spaces, only the H-component of HSV colour space can express the colour information without the affect of brightness.

Transfer from RGB to HSV

A general formula to convert from RGB to HSV is defined like this[22]:

(r,g,b) are the red, green and blue coordinates from RGB colour space, their value is a real number between 0-1.

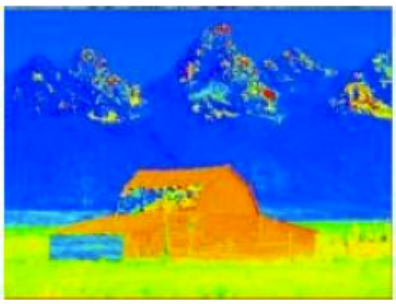
$$\text{let } \max = \max(r,g,b), \min = \min(r,g,b),$$

$$h = \begin{cases} 0^\circ, & \text{, if max=min} \\ 60^\circ \times \dfrac{g-b}{\max - \min} + 0^\circ & \text{, if max=}r \text{ and g} \geq \text{b} \\ 60^\circ \times \dfrac{g-b}{\max - \min} + 360^\circ & \text{, if max=}r \text{ and g} < \text{b} \\ 60^\circ \times \dfrac{g-b}{\max - \min} + 120^\circ & \text{, if max=g} \\ 60^\circ \times \dfrac{g-b}{\max - \min} + 240^\circ & \text{, if max=b} \end{cases}$$

$$s \begin{cases} 0, & \text{if max=0} \\ \dfrac{\max - \min}{\max} = 1 - \dfrac{\min}{\max}, & \text{otherwise} \end{cases}$$

$$v = \max$$

A sample of transfer by OpenCV shows below:



The left picture is original image use RGB colour space, the right picture is the H-component of this image.

Probability distribution histogram

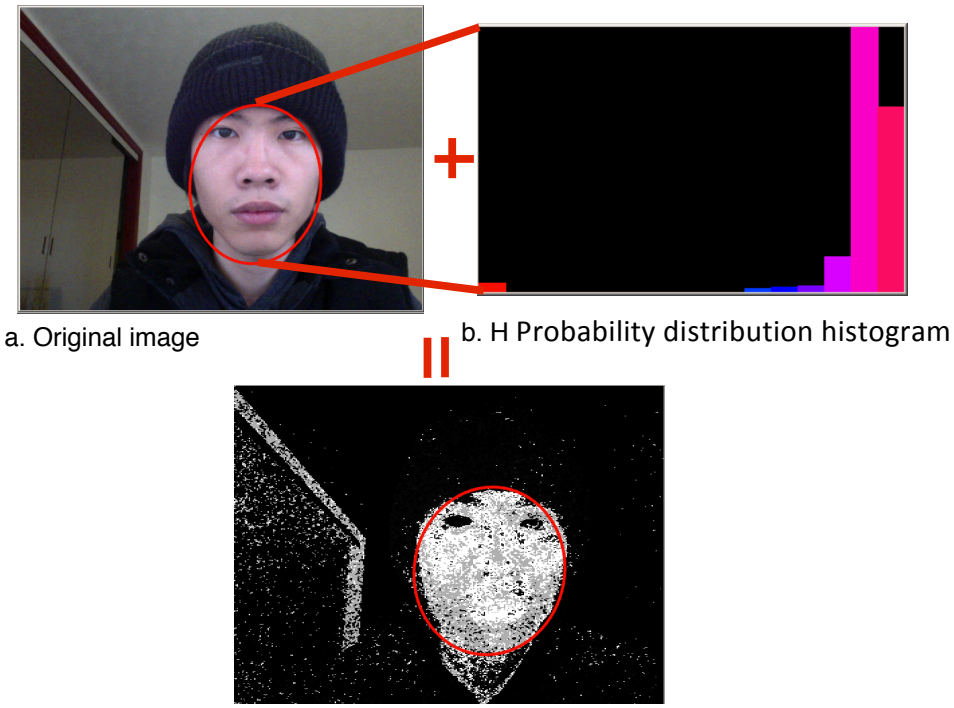


Figure 24. Face colour probability distribution histogram and back projection

After face detection, we can get a face position, transfer the face area from RGB to HSV. And we can create the probability distribution base on the H-component of face image. The value of H is 0-360, so the colour space can be divided into 360 colours. The H probability distribution histogram is the probability of each colour which is appeared in the face area.

There is a sample shows on **Figure 24**, picture **a.** is an original frame which the face position had been circled out. Picture **b.** is the H probability distribution histogram of this face. The abscissa of this histogram is the H value, and the ordinate is the probability of H colour. Because of the human face has a concentrated colour distribution, so the histogram always concentrated in some range. We can found from picture **b.**, that almost of the face colour in picture **a.** is pink and cherry. This histogram can be summed as a formula:

$f(h)=p$, where *h* is the value of abscissa, p is the value of ordinate.

An example to explain this histogram: if abscissa value h=100, the ordinate value $f(h)=25\%$, that means there are 25% pixels' H value is 100 in this face area.

Back projection

After getting the probability distribution histogram, in oder to get a grey image under this distribution, we need to transfer the value of histogram from 0-360 to 0-255 (because the value of pixels in a grey image is 0-255).

This formula can achieve this transfer[36]:

$$p_n = \frac{255}{360} h_n, h_n \subseteq [0,360]$$

Apply this formula to $f(h) = p$, a new distribution formula can be built:

$$f'(h') = p, where\ \mathrm{h}' \subseteq [0,255],\ \sum_{h'=0}^{255} f'(h') = 1.$$

The grey value of each pixels under this distribution is $v = f'(p_n) \times 255$, calculate ever pixels in the original image can get the back projection image. In the **Figure 24.**, picture **c.** is the back projection image of picture **a.** We can found that the pixels which had similar colour with face is lighter than other pixels.

An example can explain to explain this process: assume the H-component formula for the face window is $f(h) = p$, transfer the value of h from from 0-360 to 0-255, get a new formula $f'(h') = p$. To each pixel in the image, calculate it H-component value h_1 , then use the formula $h_1' = \frac{255}{360} h_1$, now $h_1' \subseteq [0,255]$, the grey value of this pixel in the grey image can be calculate by $v = f'(h_1') \times 255$,so that $v \subseteq [0,255]$. As a result, if the colour of pixel is more similar with the face window, it will get a higher value of grey value(will be more white). So we can see from the picture **c**. of **Figure 24,** the pixels in original image which are more close to red and orange will be more white in the grey image.

3.4.2.MeanShift Algorithm

After Establishing the grey image, we need to use the MeanShift algorithm for each frame to get a convergence window. The general steps of MeanShift used in CamShift algorithm is[37]:

Step 1. Initialise the detection window (this initial window comes from face detection).
Step 2. Calculate the mass centre of this window.
Step 3. Adjust the centre of window to the mass centre.
Step 4. If the window is not convergence, go to Step 2(convergence has been proved [21])

How to calculate the mass centre?

The grey image can be considered as a 2-dimensional probability density function $f(x,y)$, then the (p+q)th moment of this image can be defined as:

$$M_{pq} = \iint x^p \times y^q \times f(x,y) dxdy,\ p,q = 0,1...\infty$$

The mass centre can be calculated by zeroth moment and first moment[21]:

(1) Zeroth moment

$$M_{00} = \iint f(x,y)\,dx\,dy$$

(2) First moment of x and y

$$M_{10} = \iint x \times f(x,y)\,dx\,dy, M_{01} = \iint y \times f(x,y)\,dx\,dy$$

(3) Find the mass centre

$$x_c = \frac{M_{10}}{M_{00}}, y_c = \frac{M_{01}}{M_{00}}$$

(4) The adaptive window size s for next MeanShift

$$s = 2\sqrt{\frac{M_{00}}{256}}$$

(x_c, y_c) is the mass centre of this window. Then move the window to the mass centre and recalculate mass centre until convergence. **Figure 25.** is a process of MeanShift.

The arrow shows the movement trajectory of the window, finally the window will move to the local peak of data distribution.

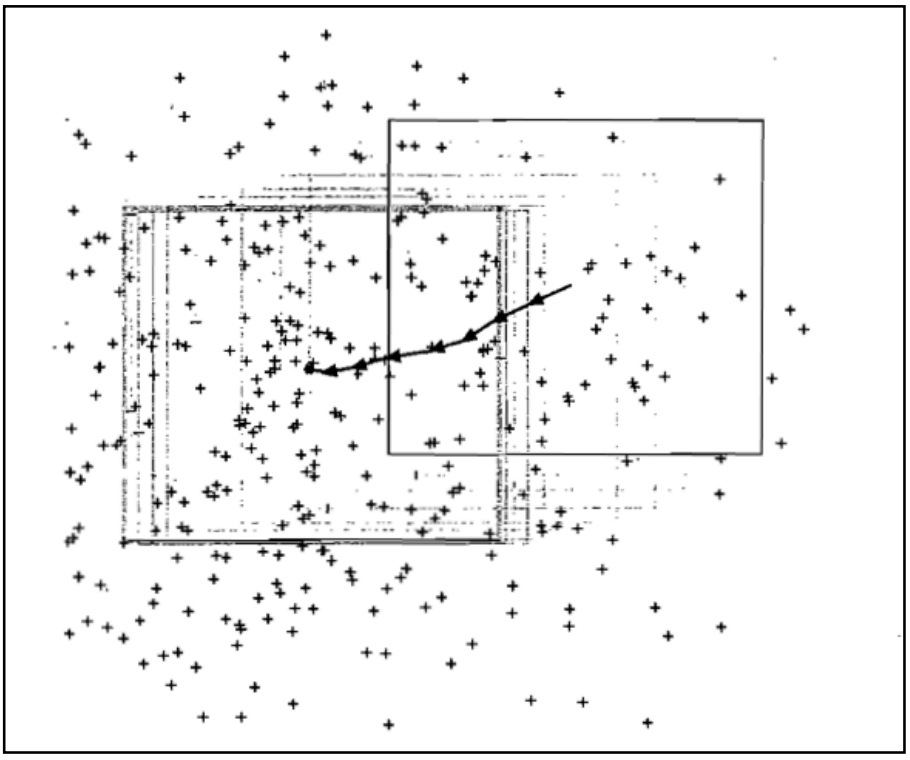


Figure 25. Process of MeanShift. Where the plus means the pixels have high grey value(more white)

3.4.3.CamShift Algorithm

MeanShift algorithm is used for a single frame, in order to tracking face in a continuous video, the CamShift use the output window of last frame become the initial window of next frame. MeanShift will got a convergence window, before send the windows to next frame, CamShift algorithm will adaptive adjust the search window for next frame, the calculate process is[21]:

(1) Second moments

$$M_{20}=\iint x^2\times f(x,y)dx\,dy, M_{02}=\iint y^2\times f(x,y)dx\,dy, M_{11}=\iint x\times y\ \times f(x,y)dx\,dy$$

(2) Calculating the direction angle of the target spindle(the major axis of the ellipse in the left picture of **Figure 26.**):

$$\theta=\frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}}-X_cY_c\right)}{\left(\frac{M_{20}}{M_{00}}-X_c^2\right)-\left(\frac{M_{02}}{M_{00}}-Y_c^2\right)}\right)}{2}$$

(3) In order to simplify the formula, three parameters had been introduced:

$$a=\frac{M_{20}}{M_{00}}-x_c^2,$$
$$b=2\left(\frac{M_{11}}{M_{00}}-x_cy_c\right),$$
$$c=\frac{M_{02}}{M_{00}}-y_c^2,$$

(4) Adaptively calculating the width w and height h for next search window are as follows(the ellipses show in the **Figure 24.** are also drawn by the parameter h,w,θ):

$$w=\sqrt{\frac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}},$$
$$h=\sqrt{\frac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}}.$$

In order to track in consecutive frames, CamShift will make the output window of last frame become the initial window of next frame. The *w* and *h* which calculated by above formula is the length and width of the tracking window for next frame.

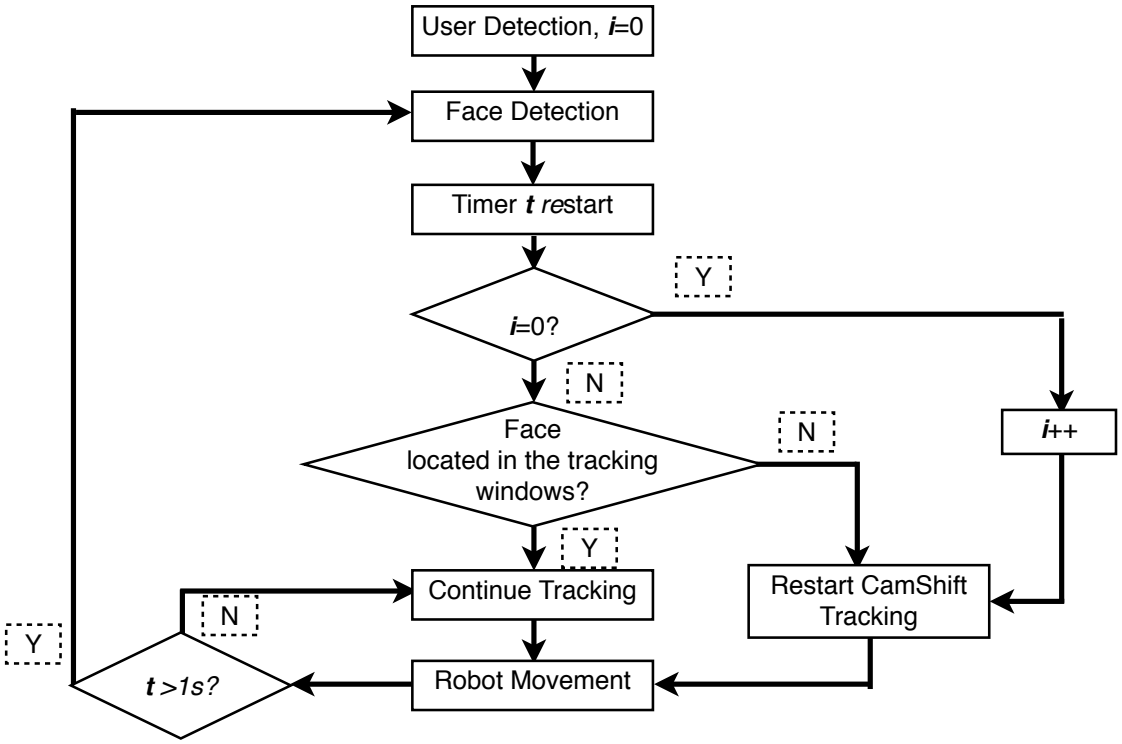
Figure 26. Result of CamShift Tracking

3.4.4.Implementation

The CamShift algorithm of this project is build base on OpenCV, this algorithm has fast tracking speed. But as an algorithm base on colour feature, the biggest problem is sometimes the tracking result will be confused by the similar colour in the background or other person who has same skin colour in the video. An mis-tracking example shows in **Figure 25**. If I move the hand in front of the face slowly, the tracking windows sometimes may remove to my hand, because the hand has similar colour with the face.

Figure 27. CamShift will be confused by hand

To solve this problem, a **correction method** is proposed in this project to reduce the interference of background. The program will restart the face detection during the face tracking per second, if the detected face position is coincided with the tracking window, that means the tracking is correct, program continue. And if the detected face position is not in the tracking window, that means the mis-tracking is happened, system will restart tracking algorithm for this face position. This method not only keep the fast speed of the program, and also increased the robustness. The flow chart shows below:



3.5. TLD Face Tracking

TLD (**T**racking-**L**earning-**D**etection) is a single target long time tracking algorithm which propose by Zdenek Kalal[38]. The novelty of this algorithm is it combines the conventional tracking algorithms and conventional detection algorithm to solve the problem of deformation, partial occlusion during the tracking. At the same time, it uses an improved online learning mechanism to continuously update the feature of the target and the parameters of the detection module to make the track more stable, robust, and reliable.

3.5.1 General Structure of TLD

For a long time Tracking, a key question is: when the target is to re-appear in the video, the system should be able to re-detected it, and restart tracking. However, during the tracking process for a long period, the shape and light of the target will inevitably be changed.The conventional tracking algorithms require cooperation with the detection module, when the target had been detected, the tracking module start. And thereafter, the detection module will not intervene into the tracking process.However, this method has a flaw: when the shape of tracking target was changed or blocked, the track it is easy to fail. Therefore, in this situation, some people only use the detection algorithm to replace the tracking algorithm. Although this method can improve the tracking effect in some cases, but it requires a offline learning process. That means, before detection, we

need select a large number of tracking target samples to study and train. So the training samples should cover all possible target images under a variety of scales, attitudes and illuminations. In other words, the quality of training samples are very important to an off-line detection algorithm, otherwise, the robustness is difficult to guarantee.

Taking into account a simple tracking or a simple detection algorithm cannot achieve a satisfied effect, TLD method consider to combine those two algorithm, and added an improved online learning mechanism, making the tracking more stable and effective. A general structure of TLD can be divided into three parts, Learning module, Tracking module and Detection module, as shown in the following figure:

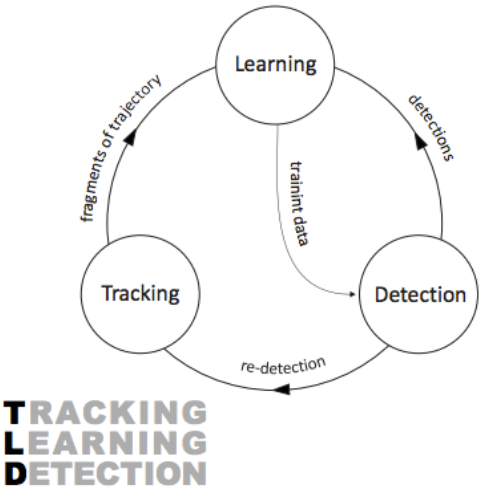
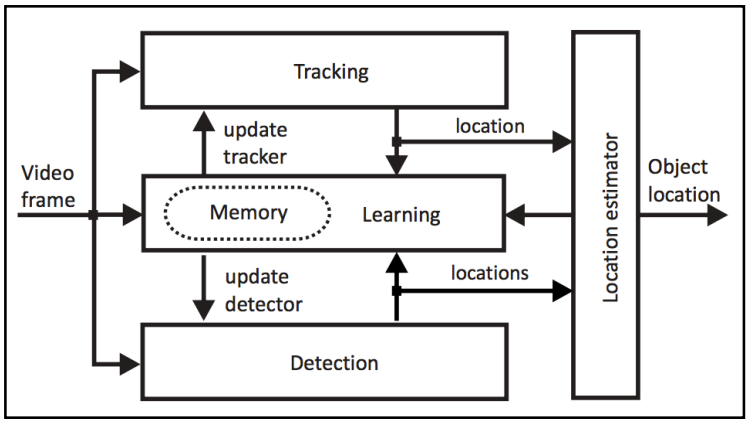


Figure 28. General structure of TLD[39]

The process of TLD is: the detection module and a tracking module will parallel processing. First of all, in order to estimate the movement of the target, the tracking modules assume the movement of target is limited between adjacent frames in the video, and the target is visible. If the target disappeared from the video the tracking will fail. The detection module assumes that each frame is independent, and searches the full frame to locate the possible target area. The target model used to detect is learned by previous experience. TLD detection module is also likely to make mistakes, and the error is nothing more than a false negative sample and a false positive sample. The learning module is used to correct those errors. It will assess these two errors based on the results of the tracking module and update the target model for the detection by the results of assessment. And update the key feature points of tracking module to avoid similar mistakes in the future.

A more detail process diagram of TLD is shown below[40]:



TLD use **Optical Flow** with **Forward-Backward Error**[41] in tracking module and a method named **P-N learning** is implemented in learning module, the detection part uses the same features with **section 3.3** . So I will discuss the **Optical flow** and **P-N learning** in the next sections.

3.5.2 Optical Flow with Forward-Backward Error

The concept of optical flow is first proposed by Gibson in 1950 [37]. It is a method to calculate the motion information between adjacent frames. It use the change of pixels in the time domain and the change of pixels between the adjacent frames to find the trajectory of pixels, and calculate the trajectory of full object. In general, the optical flow is produced by the movement of tracking object and the movement of camera. **Figure 29.** can simple explain what is optical flow:

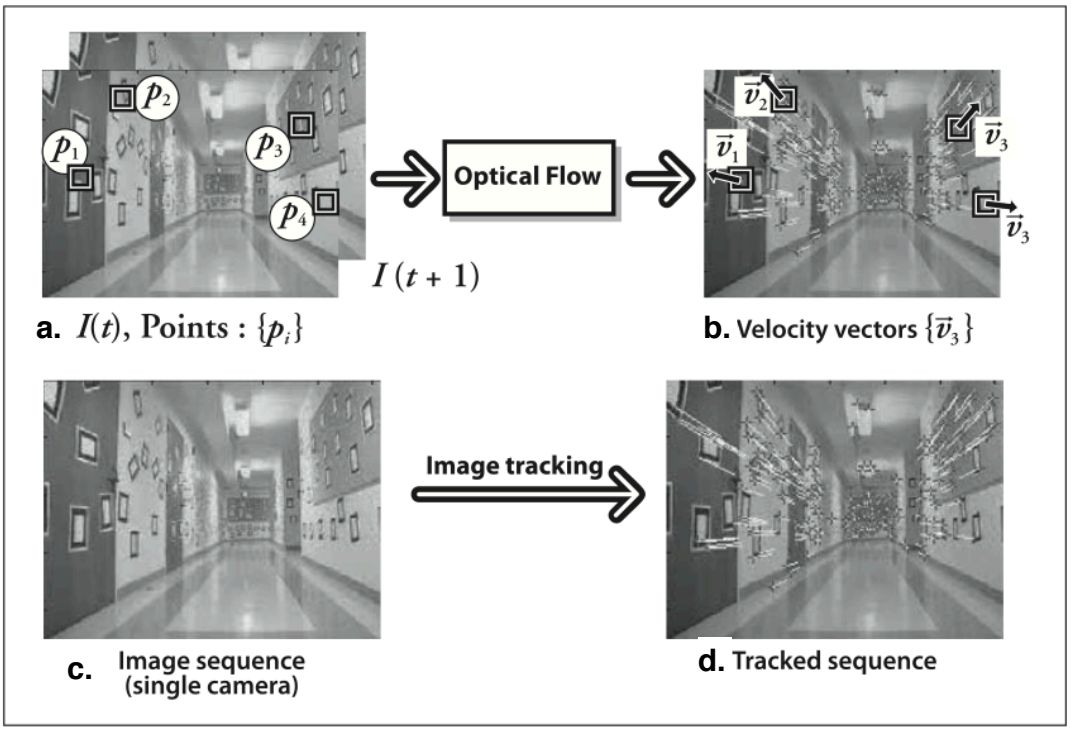


Figure 29. Optical flow[37]

In the picture **a**, there are two adjacent frames, *I(t)* and *I(t+1),* and some points had been marked in *I(t)* (Points: $\{p_i\}$). The optical flow is to find the position of those points in next frame by search the surrounding positions of those points. If a point had been tracked its movement can be converted to the velocity vector, which is shown in picture **b**. ,$\{\vec{v}_n\}$ is the velocity vectors of Points: $\{p_i\}$. Picture **c** & **d** are the original frame and the optical flow frame. Some points had been marked by velocity vectors in **d**.

Optical Flow in TLD

Zdenek[41] use Lucas-Kanade Method[37], which is a kind of optical flow method, and introduced a Median-Flow tracker to increase the tracking effect and enhance the robustness. Median-Flow tracker will calculate the Forward-Backward Error for each tracking point to remove some of the large error points, thereby improving the accuracy. A sample of Forward-Backward Error shows in **Figure 30**.

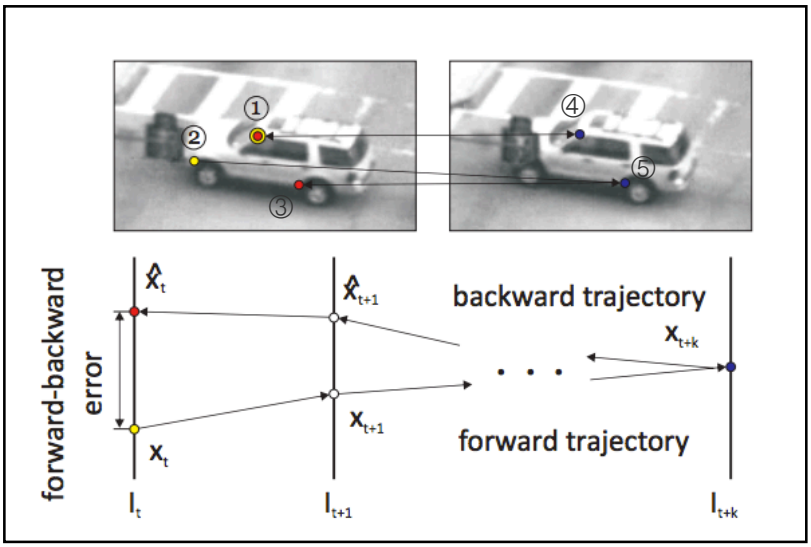


Figure 30. Forward-Backward Error[41]

Median-Flow tracker will tag number of pixels as the feature points in the tracking window, and use Lucas-Kanade Method to find the position of those feature points in next frame. This is forward tracking. Then use those point to find their pre-position in the pre-frame. This is backward tracking. And calculate the euclidean distance between the results of forward tracking and backward tracking. The euclidean distance is what Zdenek called Forward-Backward Error. The left picture in the Figure 28., there are two point①&②. After doing the forward tracking, point① found point④ in next frame and point② found point⑤. Then started the backward tracking, point ④ found its original point in pre-frame, and point⑤ found a new point③ in pre-frame. So the forward-backward error of point① is 0, the forward-backward error of point② is the euclidean distance between point②&③.

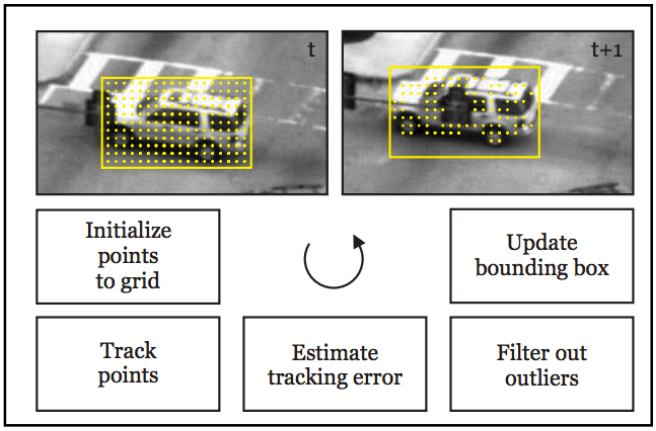


Figure 31. The process of Feature point filtering[41]

After getting all forward-backward errors of those tagged pixels, Median-Flow tracker will discard the 50% of points which had a larger errors. As **Figure 31.** shows, before

tracking, a lot of feature point had been tagged, and after tracking many unreliable points are filtered. Moving position of tracked object can be calculated through the remaining points in the next step.

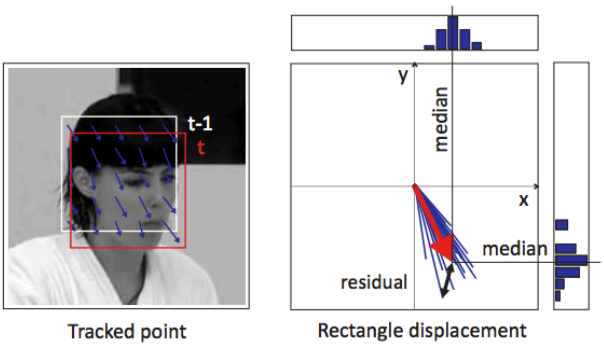


Figure 32. Tracking target movement[39]

Finally, calculate the median of those remaining points, TLD will get a motion vector for the tracking window. As **Figure 32** shows, the motion vector is calculated by median of all points. The residual is the distance between feature point and motion vector, if the median residual is larger than a threshold the tracking is failure. The threshold value is given by experience. In Zdenek's paper[41], he use the value of 10. If tracking is failure, the detection module will start to find the target position.

3.5.3 P-N Learning

P-N Learning is an online learning method[42] used in the TLD learning module. **P** refers Positive Constraint, also called P-expert or P-constraint, **N** refers Negative Constraint, also called N-expert or N-constraint.

Positive sample is an image include the target, negative sample is an image without the target. Positive samples and negative samples can used to build the classifier.

P-expert is used to found the new appearance (deformation) of target, P-expert can define the positive sample from the unlabelled data set and correct the positive example which is misclassified by classifier. So P-expert will making detection module more robust;

N-expert is used to found the negative training samples. N-expert build on the assumptions, there are only one target in the video frame and it only appear in one place in same time. Therefore, if the position of the target is determined, then the picture surround the target can be defined as negative sample.

PN learning will online processing the video sequence to improve the performance of the detection module. For each frame of the video, we hope we can find the false detection and correct it, then update the target model. So that we can avoid similar

mistakes happen again in the future video frame processing. The key point of the PN learning is the two types of "experts"[42]: P-experts to check the samples which are tagged as negative label by classifier; N-experts to check the samples which are tagged as positive label by classifier.

PN learning consists of four parts: (1)a classifier; (2) training set - some samples with positive or negative labels; (3) training method - a training method which can use the training set to update the classifier; (4) PN experts - constraint functions to relabel the result of classifier. The relationship between the four parts shown in the following figure.

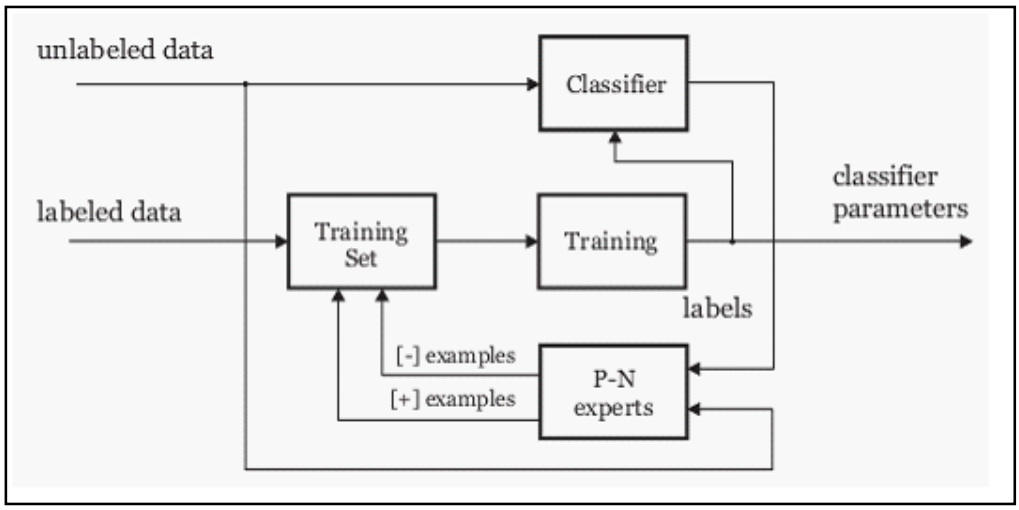


Figure 33. Process of PN learning[42]

Firstly, training the initial classifier according to the initial samples. And start the iterative learning for next frames. Use the pre-classifier to classify the samples from next frames. PN experts will correct the misclassified, the performance of next iterative learning will be improved.

Here is an example to illustrate PN learning in **Figure 34.**: there are three consecutive video frames shows below, each frames have several scanning windows, such as shown in (a):

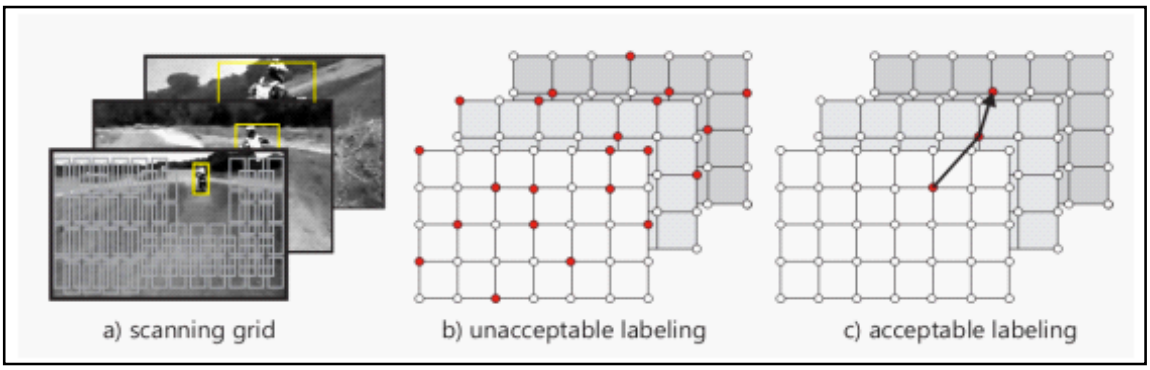


Figure 34. Example of PN learning[39]

Each scan window is an image patch, those image patches are unlabelled sample. After classifying those samples, the coloured dots in the picture (b) represent the label of sample which in the same position, red dots means positive, white means negative. The classifier is independent of each sample, therefore, there maybe more than one position will be labeled as positive. There are several positions in picture(b) is marked as positive. And we can hardly to find the continuity of the movement. Obviously, this kind of labelling is unacceptable. In contrast, an acceptable labelling shows in (c), only one target occurred in each frame, and the target in continuous frame can consist a trajectory. The role of PN-experts is to retain the right labels and to correct the wrong labels by the constraints.

The constraints of P-expert in TLD:

The principle of p-expert constraints in TLD is the movement of target will follow the trajectory line, that means the displacement of the target in the adjacent frames is small and the adjacent frames have a certain correlation. P-expert record the target location in last frame and predict the position of the target in the current frame by tracking algorithm(optical flow method). If the sample located in predicted position is labelled as negative by the classifier, then the P-expert will produce a positive training sample to training set.

The constraints of N-expert in TLD:

N-expert assume that a target on a frame can only appear in one location. The classifier will mark for all of the unlabelled samples, if the mark is larger than a threshold, the sample will be labelled as positive. N-Expert will analysis all of the positive samples, and find the region having the maximum likelihood(the sample has highest mark). N-expert will relabel those positive samples which have no overlap with this region as negative. Further, the region with maximum likelihood will be used to re-initialize the tracking module.

Another specific example to illustrate the PN-experts[42]:

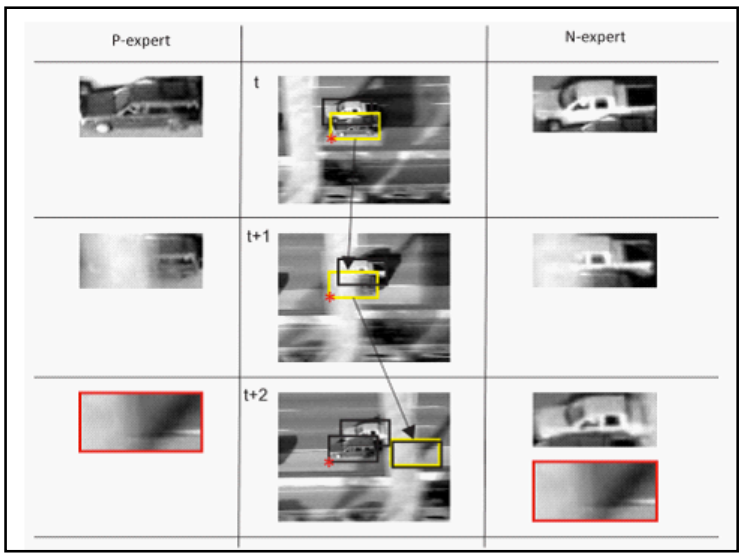


Figure 35. An example of error correction between PN-experts

Figure 35. shows three consecutive frames. PN learning needs to process the car in the yellow box at the time *t*. Tracking module gives the car's position between adjacent frames, from the previous analysis we know that the result of tracking module can be used to process positive samples by P-expert. However, because of the occlusion, a error positive sample is processed by P-expert at the time of *t+2*. At the same time, N-experts had identified the most likely position of the target (red asterisk mark), and all of other area samples had marked as negative. So at the time of *t+2*, N-expert fixed the error of P-expert.

3.5.4 Implementation

In the part of TLD, Arthurv[43] had already provide a TLD framework on C++, all things I need to do on TLD implementation is modify Arthurv's code and make a combination of TLD tracking and face detection. The flow chart is similar with **Figure 10.** and the tracking results as shown below:

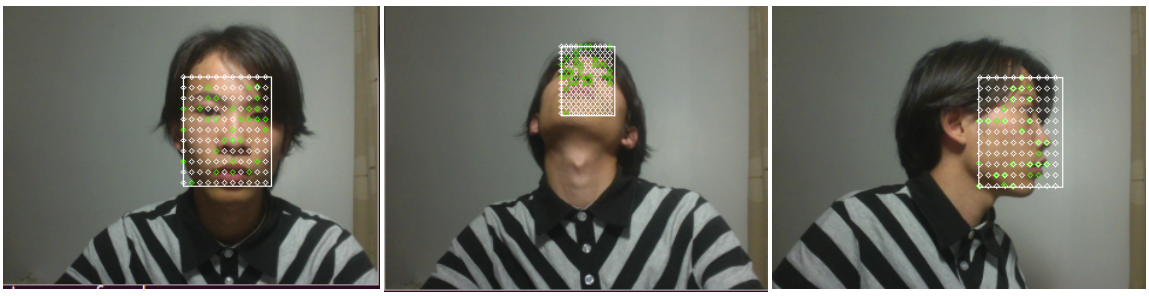


Figure 36. TLD tracking can identify the face of each angle (the white box is the tracking windows, and the green dots is the most reliable pixels in the)

As an online learning algorithm, TLD will create positive and negative test samples during the tracking by PN learning, so that means in a long time tracking, the performance of classifier will be more and more better. A specific performance and efficiency analysis of TLD will be discussed in the section 4.

3.6.Robot Movement

The robotic camera should be able to move in a 3D space to keep the target face always in the centre of video. I had two servo motors to create the movement system. In this section, the servo motors will be introduced first, and the camera structure will be discussed in section 3.6.2. After building the physical structure some question of movement should be solved by the program on PC. 'How to protect the robot itself', 'How to avoid over rotation'... those questions will be solved in section 3.5.3.

3.6.1. Lego NXT Servo Motors

There are three servo motors can be installed in a NXT robot. The inside and outlook picture shows below:

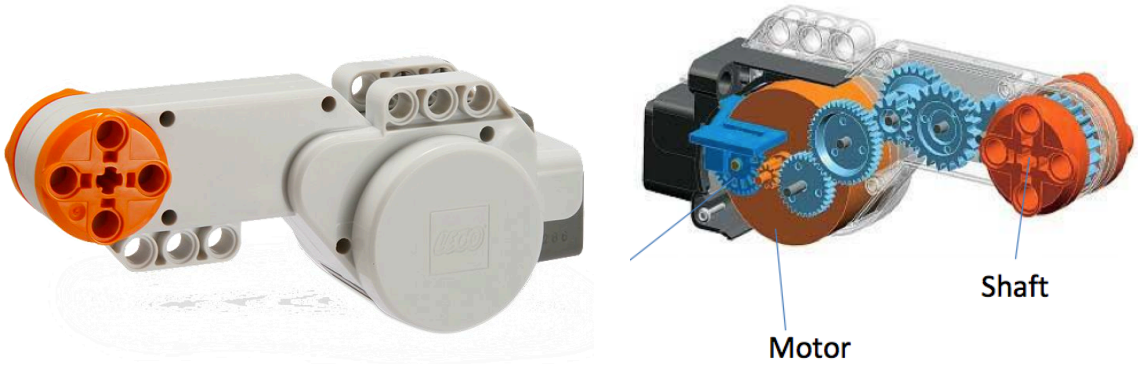


Figure 37. Servo Motors [5]

The servo motor has a built-in Rotation Sensor[20]. This sensor can control the accuracy of robot's movements to +/- one degree. Each rotation can be divided into 360 degrees, so if the motor turn 360 degrees, the shaft will turn around. And the turning speed of the motor is also can be set.

3.6.2.Structure Design and Implementation

Walterio[5] provided two 360 degreed turning structures, parallel robot and serial robot. **Figure 38.&39.** are the design of those two structures for this project. Theoretically, these two structures both can reach 360 degree rotation. I had tried those two structures, after comparing the stability and the rotation speed I finally chose one.

Parallel Robot

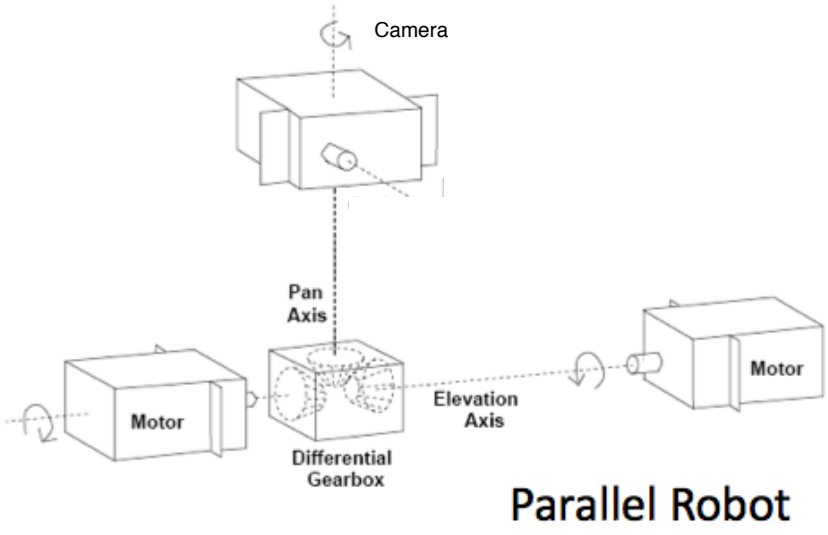
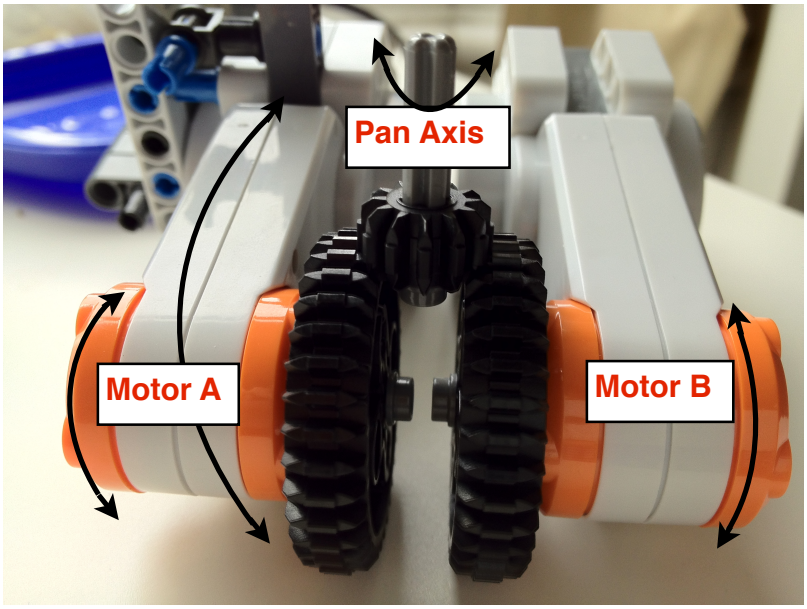


Figure 38. Structure of Parallel Robot, figure is modified by[5]

Parallel Robot use two motors to control the camera, we can see from **Figure 22.** there are two motors in the left and right sides of differential gearbox. The camera is connected with a pan axis. The pan axis and motors are linked together through the gears. The specific implementations for this project is shown below:



Motor A and Motor B both can turn forwards and backwards, camera will be fixed at the top of pan axis. If Motor A&B turn forward or backward together, the camera will going forward or backward. If A&B one forward rotation other one backward rotation, the camera will rotate to the left and right.

Serial Robot

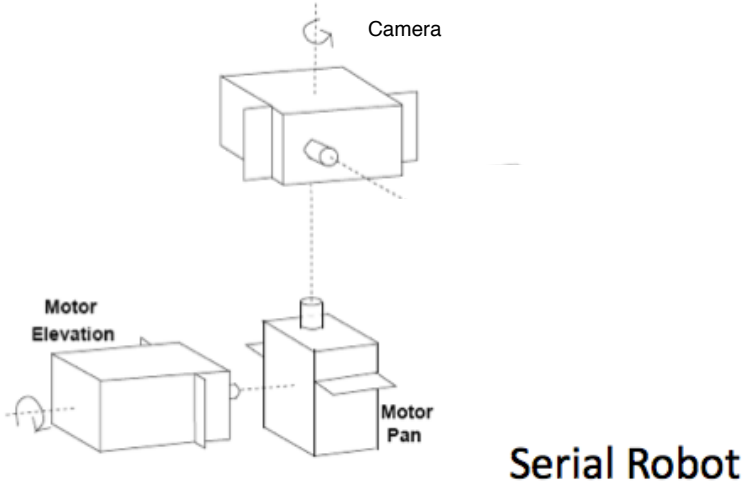


Figure 39. Structure of Serial Robot, figure is modified by[5]

Serial Robot is also use two motors to reach 360 degrees rotation. The motor elevation can provide the rotation power of forward and backward. The motor pan can provide the rotation power of left and right. The specific implementations for this project is shown below:

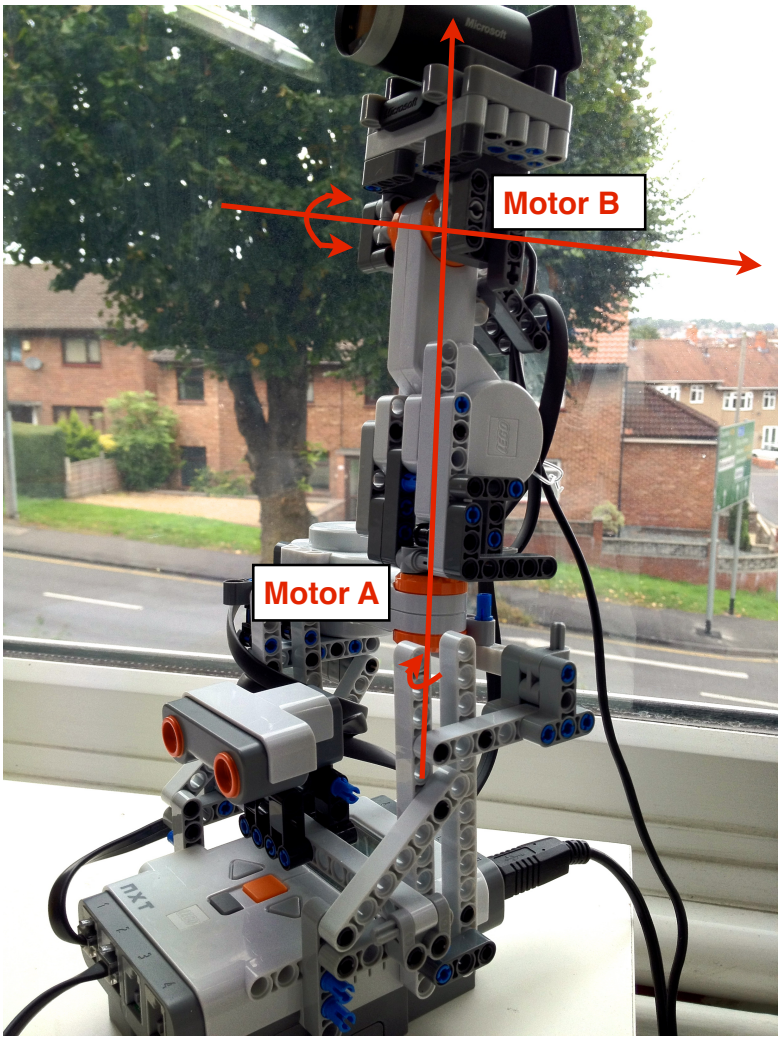


Figure 40. Serial Structure

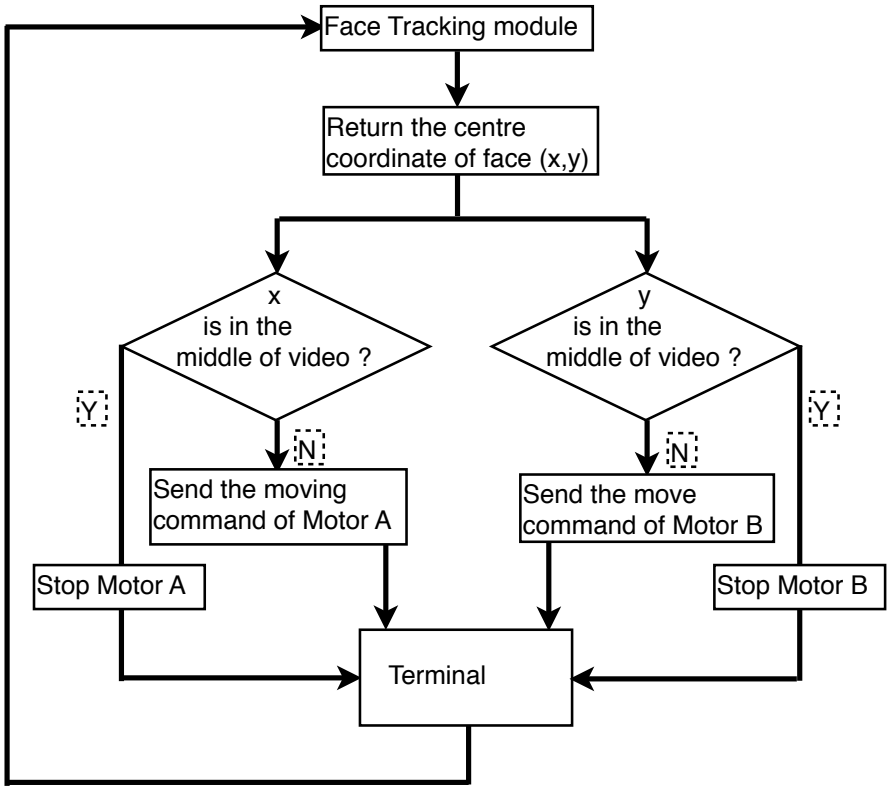
Camera is fixed in Motor B, and Motor B is connected with shaft of Motor A. So we can see that, the rotation of Motor A can drive the left or right rotation of Motor B and camera. And the Motor B can help the camera forward or backward rotation.

Comparison and Selection of Structure

The Parallel Robot can easier to achieve a higher accuracy, but also requires more precise robot bricks. Because the limited of the existing Lego bricks, it is hard to create the parallel structure. And in the part of program I had design a moving method,which do not need to know the movement angle in advance. So finally I choose the serial structure to create the robot.

3.6.3.Program Design and Implementation

The general flow chart of camera movement is:



Face tracking module will return the target coordinate. The middle of video is defined as a 60*60 box which located in the centre of video (the green box in shows in the **Figure 22.**). If the target coordinate is not located in the middle of video then move the motors. Motor A control the x-axis, Motor B control the y-axis.

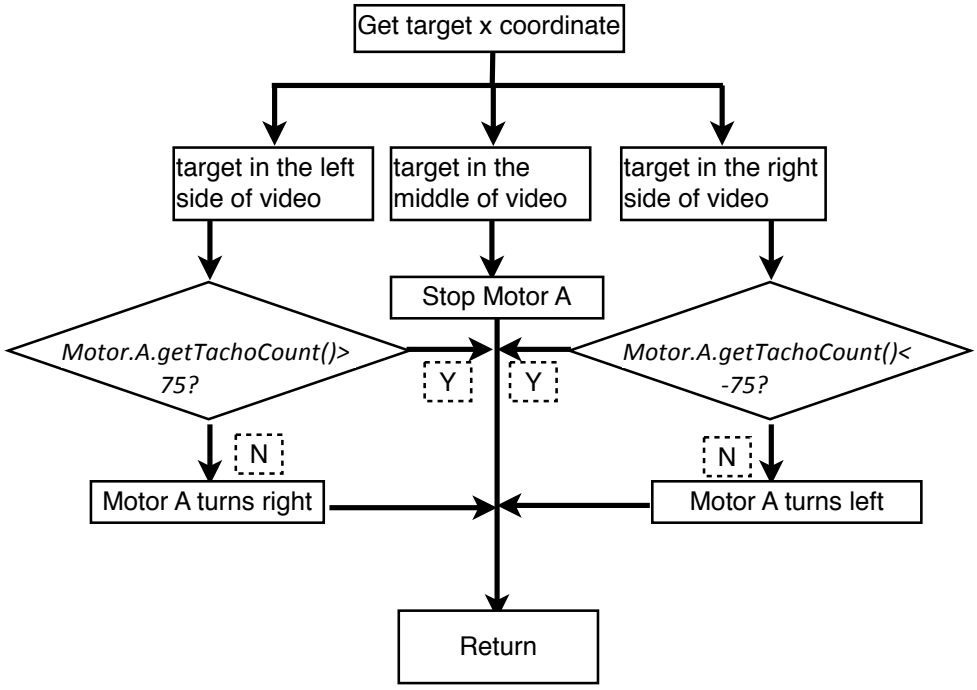
We can see from the **Figure 40.** the motors are connected with a data line to NXT body, so the motors cannot reach 360 degree rotation. A too large rotation angle will damage the robot. So the first problem should be solved is to limit the angle of rotation.

Limit rotation angle

There is a function of NXT '*Motor.A.getTachoCount()*' can return the current rotation angle. The limit angle in my program is 150° in x and y directions, which is shows in **Figure 41.**

Figure 41. The rotation angle limit in top view and side-view

The flow chart of x-axis shows below, the y-axis is similar.



Avoid over rotation

In the beginning of the design, the motors will rotate ***N*** angles after one frame is processed. If the face location is far alway from the centre the value of ***N*** will increase. This method had fast tracking speed. But the problem of this method is sometimes over rotation occurs. If the target is very close to the camera, even ***N*** takes a very small value, each rotate will lead to a large offset of image. For example, last frame the target located in the left of the video, after the ***N*** angles rotation, the target crossed centre to the right of the video. So the camera will move back again, cannot reach a stable place. This situation was what I called over rotation.

In order to solve this problem, to each frame, the motor will not rotate a fixed angle. If the target coordinate in the centre, the PC will sent a stop commend, if the target

coordinate not in the centre, the PC will sent a move commend(left or right), the motor will stop until next any frame sent a stop commend. And set different rotate speeds to increase the tracking speed. The rotate speed will higher when the target located farther from centre.

4.Evaluation and Conclusion

In order to achieve the functions of the tracking robot, I had established three versions systems, each system has its own advantages and disadvantages, a brief introduction of each version shows below:

Version 1: Only use Face Detection

Only use the face detection algorithm to calculate the target location without face tracking algorithm.

Version 2: Use Face Detection with CamShift

Use the face detection to initialise the tracking window, and CamShift is implied in tracking part. And use a correction method(section 3.4.4) to reduce the interference of similar colour in background.

Version 3: Use Face Detection with TLD

Use the face detection to initialise the tracking window, and use TLD to tracking the target.

4.1.Tracking Effect Test and Analysis

I asked three students to help me test programs and record the tracking effect. The result of test shows below:

Test place: Merchant Venturers Building(**MVB**), Queen's Building(**QB**), My dorm(**Dorm**).

Tester: Pengcheng Xu, Steven Shi, Eric

Test content: In each test place, as a tester, the test program will to track the tester 4 times, each lasting one minute, where 2 times with simple background(no other person), and 2 times with complex background(with some person and similar colour). So each version will be test for 12 times, 6 times with simple background, 6 times with complex background.

The recorded data: a). The times of correct tracking. b). The number of confusion by background(other person in the video or similar colour in background). c). The number of times the target was not detected (some times the algorithm can not found the target).

The result are shown below:

Happen/Total	Places	**Times of correct tracking**	**Confused by other person**	**Confused by similar colour in background**	**Mis-detect target**
Face Detection	MVB	6/12	3/12	0/12	3/12
	QB	8/12	3/12	0/12	1/12
	Dorm	5/12	5/12	0/12	2/12
	Avg.	6.33/12	3.67/12	0/12	2/12
	Total	**52.7%(19/36)**	**30.55%(11/12)**	**0%(0/36)**	**16.67%(6/36)**
Face Detection with CamShift	MVB	10/12	1/12	1/12	0/12
	QB	8/12	1/12	3/12	0/12
	Dorm	8/12	1/12	3/12	0/12
	Avg.	8.67/12	1/12	2.33/12	0/12
	Total	**72.22%(26/36)**	**8.33%(3/36)**	**19.44%(7/36)**	**0%(0/36)**
Face Detection with TLD	MVB	11/12	0/12	0/12	1/12
	QB	12/12	0/12	0/12	0/12
	Dorm	10/12	1/12	0/12	1/12
	Avg.	11/12	0.33/12	0/12	1.67/12
	Total	**91.67%(33/36)**	**2.78%(1/36)**	**0%(0/36)**	**5.56%(2/36)**

Analysis

Figure 41. The samples of test, the red eclipse is the tracking windows, the green dot is the centre of the tracking windows. In those samples, person wore grey clothes was the target.

Figure 41. is the sample pictures in the test. The picture **A** is the sample of simple background, which the robot got a correct tracking. The rest pictures **B**,**C**,**D** are the complex background (with other person or similar colour background). Picture **B** is a correct tracking sample, even there was another person in the video, but the robot can still found the correct target. Picture **C** is an incorrect tracking sample, which the robot mistaken the curtains as the target. Picture **D** is also an incorrect tracking sample, because the robot mistaken the person in the background as the target.

From this test, we can find that, if we only use Face Detection the tracking result is not reliable, which has only 52.7% correct rate in 36 times test. The biggest problem of this version is it will strongly be confused by other person who is also appeared in the video. And although there is a high recognition rate of face detection, the mis-detection still occur. The mis-detection rate is 16.67%. And in this version, the similar colour in background will not interfere with the tracking.

In the version of Face Detection with CamShift, the performance is better than first version. The disadvantage of this version is sometimes the tracking result will be influence by the similar colour in the background or the person who has similar skin colour with target. After applied a correction method(section 3.4.4), this defect has improved, but still cannot achieve perfect results.

Use Face Detection with TLD get a best performance in the test, which can get 91.67% correct rate. During 36 times testing, it only once mis-tracking by other person, and mis-detection is only happened twice. And as an online learning algorithm, during the tracking, those mistakes can be automatically corrected. But the performance of TLD version is not always best in the test. A biggest problem of TLD version is the running speed,sometimes the frames of video will be delayed. Because, there are three parts in TLD, learning, tracking, detection, each frame should be processed by all of those three modules. The most time-consuming part is learning, we should calculate a large number of features(see section 3.3.3) to update the classifier. So it will decrease the speed of system.

Here is a summary of the performance indicators in each versions.

	Track specific person	**Interference**	**Lost target**	**Detect area**	**Speed**
Face Detection	No	By Other Faces	Yes	Infront of Face	Medium
Face Detection with CamShift	No	By Similar Color	Yes	180º of Head	Fast
Face Detection with TLD	Yes	No	Yes	360º of Head	Slow

Track specific person means during the tracking robot will only tracking a specific person. **Lost target** means sometimes the robot will not recognise the target even the target is in the video. Unfortunately, this phenomenon occurred in all three versions. Over all, the version of Face Detection with TLD has best performance but compare with other two versions, the only disadvantage is the running speed. The reason is that the learning part of TLD will take a lot of time to update the classifier.

The **Open Day** held in the Central Design Office (stand of computer science department at University of Bristol Queen's Building, Date: 22 Sep 2012), hundreds of people are expected to come to join the event. There will be a lot of person in background, and currently the speed of TLD is acceptable for a short time display, so finally, I choose the TLD version to run the robot in **Open Day**. In actual use, the effect is also very satisfactory. I have shown to over 40 visitors, each demonstrate lasted more than two minutes. The demonstrate process was like this: firstly, I invited a visitor to stand in front of the robot, the robot will detected the visitor and start to track, and I asked the visitor to try to move the body to test the tracking result. Then I asked the user to move out of the video, so that the robot would stop, and the target(visitor) return back again, robot restart to track. And during the demonstrate I had also invited other person came into the video to show that the tracking effect was not influenced by other person's face. In the Open Day, almost every tests are successful (only two times the NXT was crashed because of long time running).

Over all, if you didn't care about the tracking speed(used in the situation that the target moving slow or the controlling computer has a high performance), the TLD version is the best choice. Face Detection with CamShift suitable for this situation that if you need a high real-time performance and there is less similar colour in background. And the only version only with Face Detection can get a better performance in the place with less people.

4.2. Further Work

Currently the robot can meet the basic tracking task, I think the further work can be divided into two three part:

1. Optimise the algorithm. Currently the TLD algorithm is modified by Arthurv's code, if we can reprogram the code to make it focus the face tracking, or optimise the algorithm to improve the processing time, the robot will get a better performance.
2. Improving the physical structure and add new device. Now the robot is built by the Lego bricks. Because the limit of brick's area, movement of the camera cannot reach a high accuracy. If can find a new element to build the robot body can make it more flexible. On the other hand, the Lego NXT supports Bluetooth. We can find a mobile device(smart phone, tablet,ect.) to replace the PC. So that the robot can be more portable.
3. Applied to CCTV surveillance or a receptionist robot system. As mentioned in section 1.1., there are two possible future application of this robot. Dr Rafal had already got a receptionist robot that we can applied the face tracking function to it. A CCTV surveillance is also a direction to extent the project.

5.Reference

[1] IFR Statistical Department, World Industrial Robotics 2011 report, World Robotics - Industrial Robots 2011, http://www.worldrobotics.org/.

[2] HU Weiming,TAN Tieniu,WANG Liang,MAYBANK S. A survey on visual surveillance of object motion and behav-iors[J].IEEE Transactions on Systems,Man,and Cyber- neties, Part C: Applications and Reviews,2004,34(3): 334-352.

[3] James Anthony Humphreys, The Automated Tracking Of Vehicles and Pedestrians In CCTV For Use In The Detection Of Novel Behaviour,Durham University, Department of Computer Science 2002-2004

[4] Ian Fasel, Bret Fortenberry and Javier Movellan, A generative framework for real time object detection and classification, Computer vision and image understanding, Volume 98, Issue 1, Pages 182-210, 2005.

[5] Walterio Mayol-Cuevas, Andrew Conn, Lecture of Robotic Systems, University of Bristol, 2012.

[6] Lee and Tae-Hoon, Real-time face detection and recognition on LEGO Mindstorms NXT robot, Advance in Biometrics: Lecture Notes in Computer Science, Volume: 4642, Pages 1006-1015, 2007.

[7] Java for LEGO Mindstorms, Available from: http://lejos.sourceforge.net/.

[8] Shiqiang Zhu, Xuanyin Wang, Lecture of <The technology and application of robot>, Zhejiang University, China.

[9] M. Negnevitsky, Artificial intelligence: A guide to intelligent systems, Addison-Wesley, Reading, MA, 2005.

[10] Josf Capek, Opilec, In: Lelio A Pro DelWna, Aventinum, Pra- gue, 1920.

[11] N Sharkey, A Sharkey, “Electro-mechanical robots before the computer”, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, Pages 235- 241, 2009.

[12]I. Asimov, I, Robot (a collection of short stories originally published between 1940 and 1950), Grafton Books, London, 1968.

[13] Clarke, Roger, "Asimov's Laws for Robotics: Implications for Information Technology", Part 1 and Part 2, Computer, December 1993, Pages. 53-61 and Computer, January 1994.

[14] Bartneck, C. Forlizzi, J., A Design-Centred Framework for Social Human-Robot Interaction, Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on, Pages. 591-594, 2004.

[15] Nilsson, Nils J., Shakey the Robot, Technical note no. 323, Pages 135, 1984

[16] Tribelhorn, B., Dodds, Z. Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education, Robotics and Automation, 2007 IEEE International Conference on, Pages 1393-1399, 2007.

[17] Min K. L., Sara K., Jodi F., Receptionist of Information Kiosk: How Do People Talk With a Robot? , CSCW '10 Proceedings of the 2010 ACM conference on Computer supported cooperative work, 2010.

[18] M. Koga, Y. Hosoda, T. Moriya. Humanoid robots. Hitachi Review, Vol.58, No. 4, Pages. 151–156, 2009.

[19] Takuya Hashimoto, Naoki Kato and Hiroshi Kobayashi, Development of Educational System with the Android Robot SAYA and Evaluation, International Journal of Advanced Robotic Systems, Vol. 8, No. 3, Special Issue Assistive Robotics, Pages.51-61, 2011.

[20] Introduction of NXT Sensors, http://legoengineering.com/nxt-sensors-2.html

[21] Gary R. Bradski, Computer Vision Face Tracking For Use in a Perceptual User Interface, Microcomputer Research Lab, Santa Clara,Intel Technology Journal Q2 ’98.

[22] HSL and HSV, Wikipedia,http://en.wikipedia.org/wiki/HSL_and_HSV

[23] Yang Ming-Hsuan,David J. K.,Narendra A., Detecting Faces in Images: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24(1):34-58, 2002/01

[24] Hjelmås E.,Low B. K., Face Detection: A Survey, Computer Vision and Image Understanding 83,Pages. 236-274, 2001.

[25] V. Govindaraju, Locating human faces in photographs, Int. J. Comput. Vision, Vol. 19(2), Page.129-146, 1996.

[26] H. Zabrodsky, S. Peleg, and D. Avnir. Symmetry as a contin- uous feature. IEEE Trans. PAMI, Vol. 17(12), Pages. 1154–1166, 1995.

[27] Paul Viola and Michael J. Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, Computer Vision and Pattern Recognition, Vol. 1, Pages 8-14, 2001.

[28] B. Scassellati, Eye Finding via Face Detection for a Foveated, Active Vision System, Proc. 15th Nat’l Conf. Artificial Intelligence, 1998.

[29] Paul Viola and Michael J. Jones, Robust Real-Time Face Detection, International journal of computer vision, Volume 57, Number 2, Pages 137-154, 2001; revised 2004.

[30] Nefian A., Hayes M., Hidden Markov Models for Face Recognition, IEEE International Conference on Acoustics, Speech and Signals Processings, Seattle, Washington, Pages. 2721-2724, 1998.

[31] Kearns M., Valiant L. G., Cryptographic Limitations on Learning Boolean Formulae and Finite Automata, Journal of the ACM, Vol. 41(1), Pages. 67-95, 1994.

[32] Zhao Nan, Face Detection Based on AdaBoost, Undergraduate thesis of Peking University, 2005.

[33] F.G. Meyer, P. Bouthemy, Region-Based Tracking Using Affine Motion Models in Long Image Sequences, *CVGIP: Image Understanding*, Vol. 60(2), Pages. 119-140, 1994.

[34] Liang Wang, Tieniu Tan, Weiming Hu. Face Tracking Using Motion-Guided Dynamic Template Matching. *The 5th Asian Conference on Computer Vision*, Pages. 23-25, 2002.

[35] G. Tsechpenakis, K. Rapantzikos, N. Tsapatsoulis, S. Kollias, A snake model for object tracking in natural sequences, Vol. 19(3), Pages. 219-238, 2004.

[36] Geng Pei, Su Xiaolong, CamShift Moving Object Tracking,School of Computer Science and Technology, China University of Mining and Technology,2011

[37] Gary Bradski, Adrian Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 1st edition (October 1, 2008), Pages. 338-340

[38] Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk, Online learning of robust object detectors during unstable tracking, 3rd On-line Learning for Computer Vision Workshop, Kyoto, Japan, IEEE CS, 2009.

[39] Zdenek Kalal, Components of TLD, http://info.ee.surrey.ac.uk/Personal/Z.Kalal/Publications/2010_cvpr_demo_poster.pdf

[40]Z. Kalal, K. Mikolajczyk, and J. Matas, Face-TLD: Tracking-Learning-Detection Applied to Faces, International Conference on Image Processing, 2010.

[41] Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk, Forward-Backward Error: Automatic Detection of Tracking Failures, International Conference on Pattern Recognition, 23-26 August, 2010, Istambul, Turkey

[42] Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk, P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints,23rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR, June 13-18, San Francisco, CA, USA, 2010

[43] Arthurv, C++ implementation of TLD, https://github.com/arthurv/OpenTLD, 2011