Department of Computer Science and Engineering

The Chinese University of Hong Kong

2006-2007 Final Year Project Report (1st Term)

A Generic Real-Time Facial Expression Modelling System

Group: LYU0603

Supervised by

Prof. Michael R. Lyu

Students:

Cheung Ka Shun (05521661)

Wong Chi Kin (05524554)

Table of Contents

Chapter 1: Introduction

1.1 Background Information

Image and Video based face coordinate recognition system is a system which tries to recognize human face coordinates such as eyes and nose using a gallery or a video (the video can be live recording if the computation speed is high enough) instead of using any mechanical apparatus such as Infrared sensor. In the other words, it is a computation-driven technology for automatically identifying a human face position from an image or a video. In contrast, infrared sensor based application can only recognize the position of human limbs and head (commonly applied in game industry) but cannot precisely recognize the facial expressions due to colour missing. The major mechanism behind image and video based face coordinate recognition system is trying to compare between the face database and the collected picture from image source (a frame in video perspective) so as to figure out the matched pattern between them.

There exists several kinds of application for image and video based face coordinate recognition system. Security system with eye iris recognition system can differentiate the user from which authentication level who was granted. It is much more secure and convenience than the traditional password based secure system. Another typical application is an accessibility tool which helps the disabled person. The application recognizes the eye-ball movement such that disabled person can control the computer by the movement of eyeballs or head.

1.2 Project Motivation

Nowadays, the utilization of the face recognition technology becomes more common and widely used in different area of information technology and entertainment industry. The applied level also spread from business level to user-level. The major reason is that the processing power of the electronic devices is much more improved than few years ago, and the cost of such devices reduced. End-user requires more interesting and attractive effect than before. The following are two examples which applied face recognition system.

Canon PowerShot SD900 includes face detect system feature which can automatically focus (AF) on human face. It facilitates user on focusing.

Fig.1.1 Canon PowerShot SD900

Logitech QuickCam Pro5000 supports face detection and cartoon simulation.

Fig.1.2 Logitech QuickCam Pro 5000 and its video effect

Video source (e.g. from webcam) was commonly used in net-meeting software such as Windows Live Messenger. This kind of applications is able to browse the real time video capture source from who is net-meeting with. To enrich the functionality and entertainment of such kind of applications, we decided to make a more interesting and innovative video output can be presented to the other side.

High computation power computer system has become more available to end-user. The resolution, frame rate and video quality (image with less noise) supported by webcam have improved in recent years. As the limitation caused by the hardware equipment has reduced, we would like to build an application which is able to detect human facial expression and to model the detected expressions into a 3-D model.

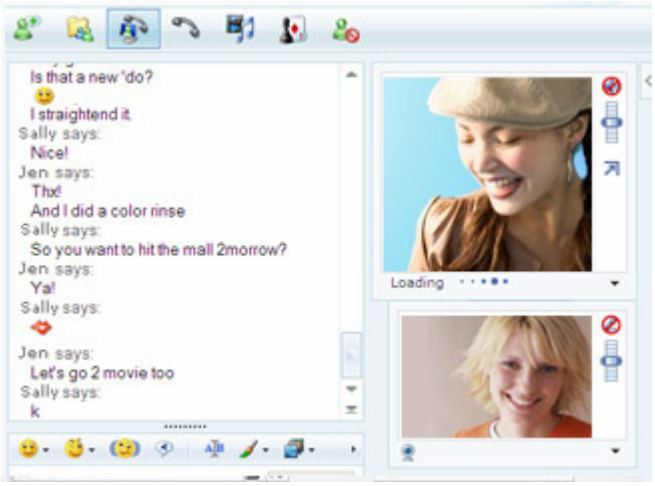

Fig.1.3 Using net-meeting features in Windows Live Messenger

Windows Live Messenger has become more popular than previous year, and there are many ICQ users migrating to Windows Live Messenger. The major reason is that Windows Live Messenger provides many interesting emotion pictures in its chatting environment. Windows Live Messenger also allows user to import self-defined emotion pictures so as to enrich the chatting environment. It is obviously that users are interested in some luxuriant features on top of basic functionality given by the software.

We think that the net-meeting can be further enhanced, like the market selling point in Windows Live Messenger chatting environment. Our enhancement is to provide a more interesting video output in net-meeting.

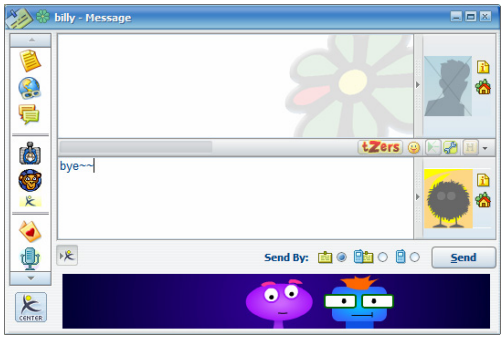

Fig.1.4. ICQ chatting environment (only supports pre-defined emotion pictures)

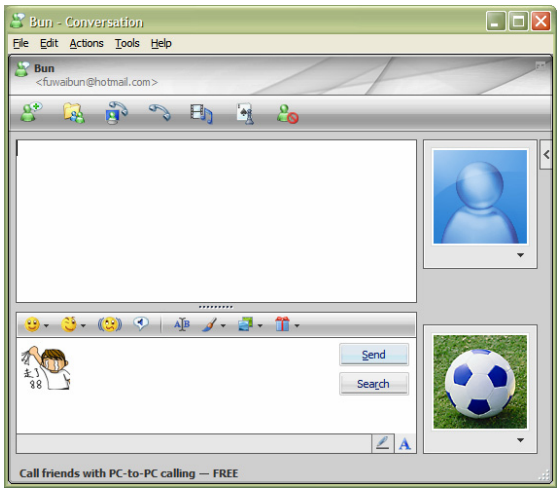

Fig.1.5. Windows Live Messenger chatting environment which supports self-defined emotion pictures.

1.3 Project Objective

The primary goal of Generic Real-Time Facial Expression Modelling System (GRTFEMS) is to enrich and enhance the functionality of video source from a web-cam or video clip so that end-user can finally enjoy an interesting and dynamically generating video effect on net-meeting applications or computer games.

Due to the fact that this solution is implemented in software based approach, users are not required to pay extra cost on specific equipment. In other words, the application is able to work on different kinds of hardware as to achieve our generic goal. Our system can also recognise different faces and draw an appropriate model. It supports different faces from different people, which is also generic.

GRTFEMS acts like a filter of the video output which can be embedded into current net-meeting software. When the user is net-meeting with his or her friends, he or she can use a virtual character to replace and display to the other side instead of displaying the realistic environment.

1.4 Project Scope

This part describes our project scope. In other words, it defines what the project is going to be developed and to which extent. It is useful for scheduling our tasks so that we can work in a more efficient way.

This project is divided into two major parts. The first part is human facial expressions detection. It detects several varieties of facial expressions such as eyes blinking, mouth opening and head movement.

The second part of the project is facial expressions modelling. After collecting the facial information and analyzing the information in part one, we can model a 2-D or 3-D character which representing the actual movement of the human.

1.5 Project Equipment

In our project, there are two dedicated PCs for us to develop the application. The major configurations of our PCs and peripheral are listed as following:

CPU	Intel P4 3.0Ghz Processor
Motherboard	ASUS P5LD2
Memory Size	1.0GB
Display card	NVIDIA GeForce 6600
Video Capture Device	(i) Logitech Webcam Pro4000 (ii) Panasonic NV-GS180 DV Camera (2.3M Pixel, 3CCD, FireWire compatible)

We test our system on both video capture devices. Panasonic DV Camera usually has a better performance than Logitech Webcam Pro4000.

On the other hand, we use KIOSK as our testing environment as the lighting condition is balanced.

Fig.1.6. KIOSK, which has a balanced lighting environment

Chapter 2: Development Environment

2.1 Programming Platform

This part introduces the programming platform we used to develop GRTFEMS. They are important because it also affect the running platform of our finalized system.

There is an important tool we have used in our project, which is GraphEdit. It is a visual tool for building filter graph (a graph which comprised of DirectShow filters). With the help of GraphEdit, we are able to experiment with filters we build in a convenient way because it provides some fundamental features like start or pause playing the streaming.

2.1.1 Microsoft Windows

Our system is developed under Microsoft Windows XP Professional Edition. We chose Windows XP as our operating system because we have chosen Microsoft DirectX as our programming APIs and it is only available in Windows environment.

2.1.2 Microsoft Visual C++

Microsoft Visual C++ is an Integrated Development Environment (IDE) for programming language engineered by Microsoft. It provides many powerful tools and features for program development such as keywords colouring and IntelliSense (a form for automated completion feature). The debugger embedded is easy-to-use and powerful.

As programming language C++ is more compatible with Microsoft DirectShow and Direct3D than Visual Basic because Visual Basic can only access a small subset of DirectShow API. Therefore, the IDE we chosen must support C++. It is the reason why we chose Microsoft Visual C++ as our programming platform.

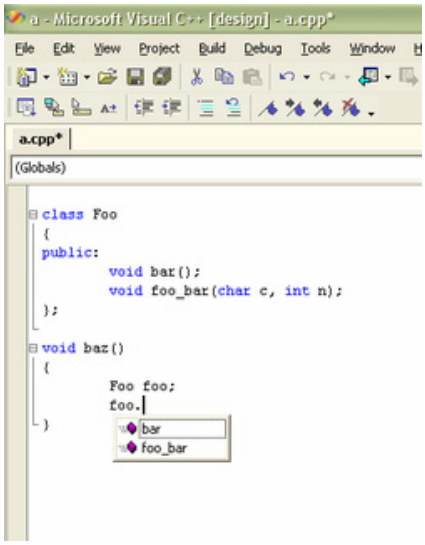

Fig.2.1 The above figure is an illustration of IntelliSense

2.1.3 GraphEdit

GraphEdit is a tool which visually builds and tests Microsoft DirectShow filters (filter graph). It supports drag and drop feature so that we can drag a move clip into the windows.

Figure 2.2 shows a filter graph displayed visually. The filter closer to the left is upstream.

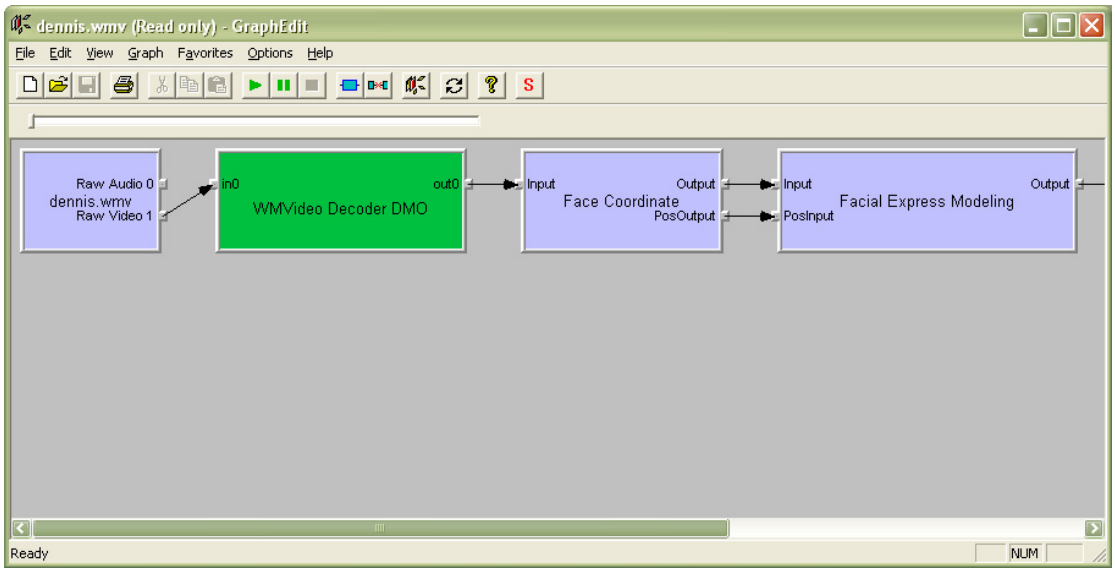

Fig.2.2 User interface of GraphEdit

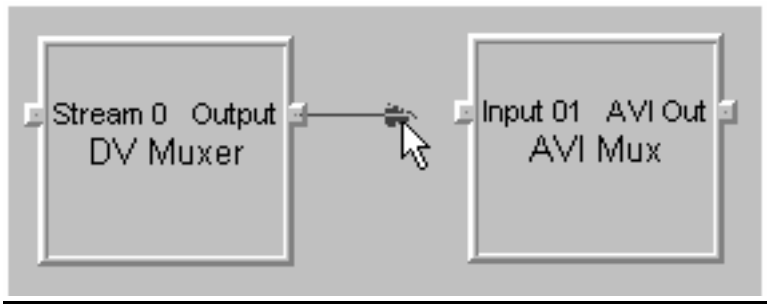

Fig.2.3 To connect the filters, we only have to drag an arrow which represents a stream between two filters

After we built a filter graph by GraphEdit, we can start to play the streaming of the filter graph by simply clicking on the play button. In figure 2.4 shows playing a movie clip (in mpg format) using filter graph.

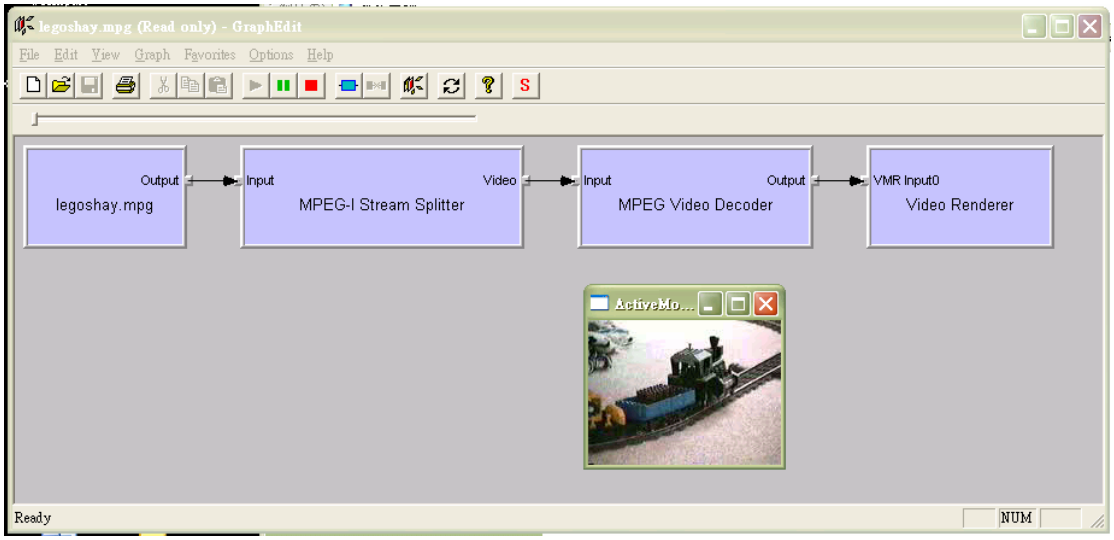

Fig.2.4 Playing a movie clip (.mpg) using GraphEdit*

Actually, GraphEdit is invoking the functions Run, Pause and Stop in IMediaControl interface to perform Playing, Pausing and Stopping of the streaming.

2.2 Programming API

In our project, we adopted C++ as our programming language, Microsoft Visual C++ as our programming environment with Microsoft DirectShow and Microsoft Direct3D as the programming API. Due to the fact that they are all developed by Microsoft Corporation, the compatibility between them is very good.

The following sections will further introduce the features of the programming platforms and state the reasons why we adopted them in our system. It also shows our study and learning on them.

2.1.1 Microsoft DirectShow

Microsoft DirectShow is an architecture which is applied for streaming media, see figure 6. It provides a mechanism for video source capture and then playback as a multimedia streams. It supports wide variety of media formats. Different video capture devices such as webcam and DV camera have their own video output format. With the help of DirectShow filters, we can use different video source in our project without modifying the code as to adapt the type compatibility. For example, it can support Audio-Video Interleaved (AVI) and Windows Media Video (WMV).

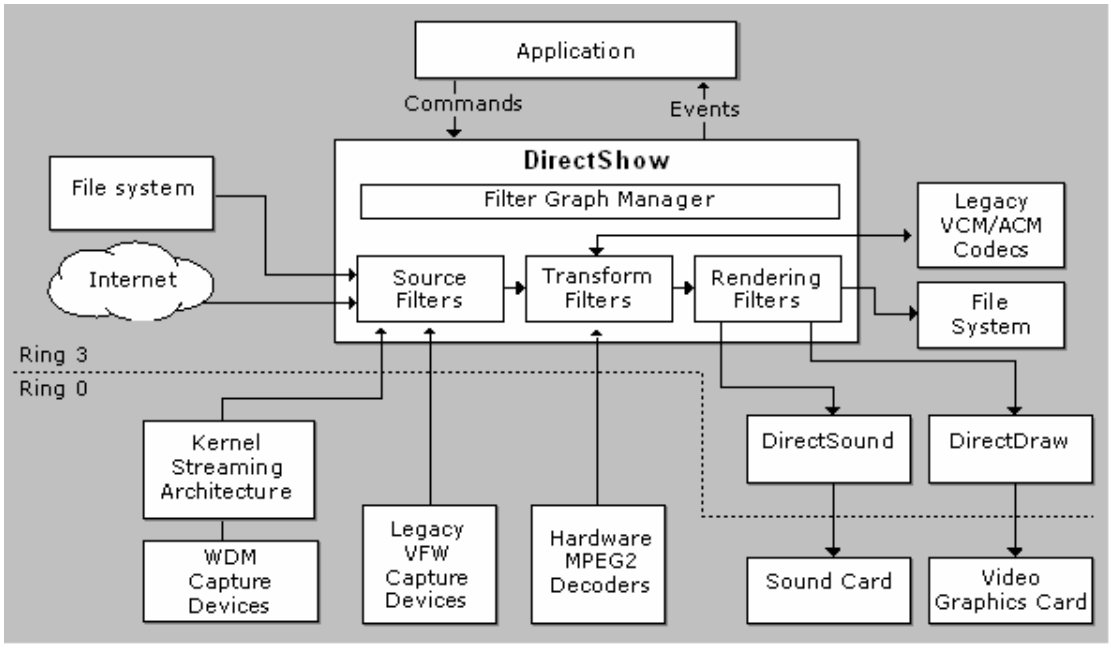

Fig.2.5. Architecture of DirectShow

DirectShow allows us to access underlying stream control architecture for applications that require custom solutions. It simplifies the task of creating digital media applications by isolating applications from the complexities of data transports, hardware differences, and synchronization. The good modularizing model helps developer has a better management on the system and separation of concerns.

Since Microsoft DirectShow uses modular architecture, the data streaming processing is done by COM (Component Object Model) object, the COM responsible for data streaming is called filter. There are some standard filters already provided by DirectShow such as AVI Splitter.

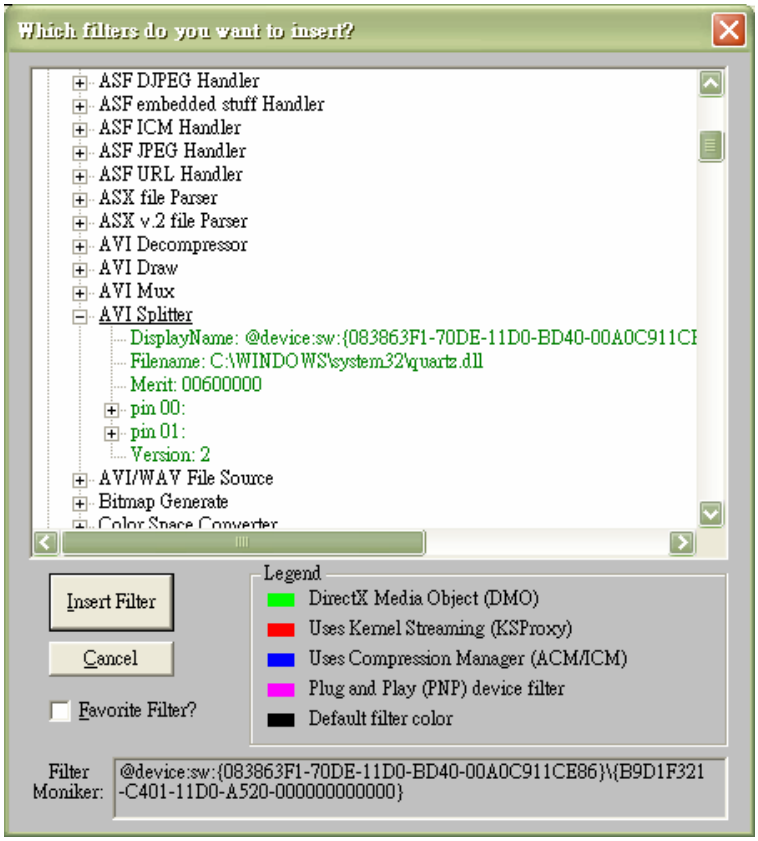

Fig.2.6. There are many standard DirectShow filters provided for streaming processing

As DirectShow filters is comprised to form a streaming, a set of connected filters is called a filter graph, as shown in figure 8.

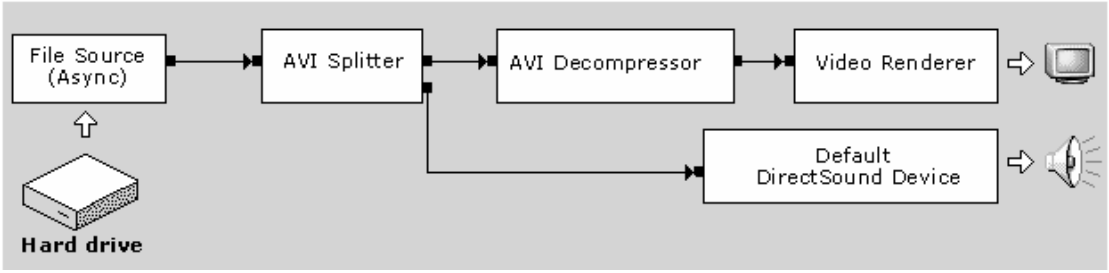

Fig.2.7 An example for DirectShow filter graph

In a DirectShow application, it is required that the application has to manage the data flow by itself. There is a COM object, Filter Graph Manager which is a high-level API for controlling data stream. The relationship between the application, filter graph manager and filter graph are illustrated in figure 9.

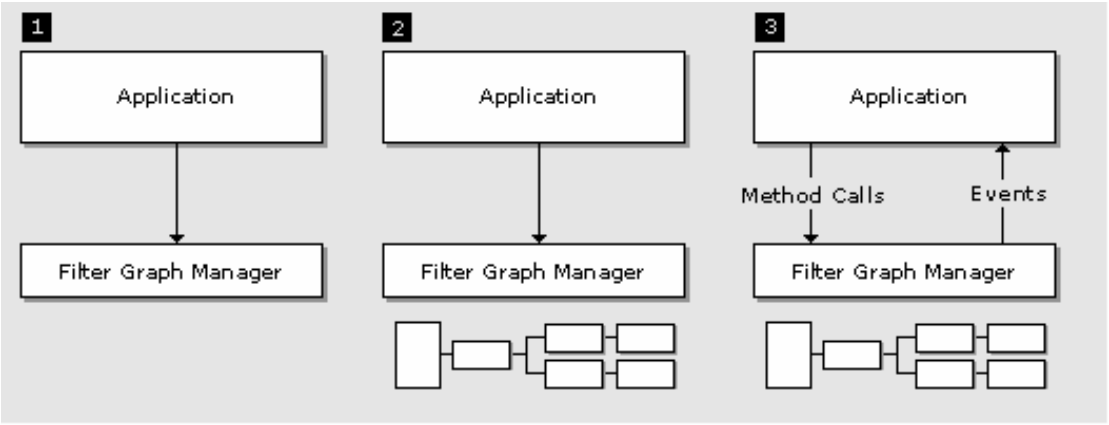

Fig.2.8 Relationship of application, filter graph manager and filter graph

There are totally five types of DirectShow filters. They are source filter, transform filter, renderer filter, splitter filter and mux filter.

Our project is trying to build a transform filter. Transform filter is an intermediate component in a video stream. It takes an input stream from the upstream filter, and then processes the data. After that, the filter will create an output dream to the downstream filter. In Generic Real-Time Facial Expression Modelling System, the filter we built is named as Facial Express Modelling filter. The functionality of this filter is to analyze the human facial expression in a frame of the video stream. After the facial expression result is obtained, it will create a Direct3D device which is responsible for drawing appropriate picture representing human expressions.

Due to the fact that DirectShow is an excellent video streaming processing API, and it provides many useful functional APIs for video processing, we adopted it in our project.

2.2.2 Microsoft Direct3D

Microsoft Direct3D is a part of DirectX API and only be available in Microsoft Windows platform. Direct3D provides an API to developers so that they can render three dimensional graphics using graphical acceleration device if it is available.

Microsoft Direct3D and Microsoft DirectShow are also under Microsoft DirectX software development kit. The programming techniques required are similar. One of the examples is that, the DirectX developer should have basic knowledge on COM object. If we are familiar with using and writing a COM object, it will be beneficial on programming with both APIs.

In Direct3D, we must be familiar with two systems which are featured in Direct3D in order to draw or model object in the Direct3D world. They are 3-D coordinate system and texture coordinate system. We will have more details later on.

2.2.3 FaceCoordinate filter

FaceCoordinate filter is provided by VIEWLab, which is authorized to be adopted in our project.

In our project, we have used FaceCoordinate filter as our interface. We had studied on FaceCoordinate filter so that we can utilize the output streamed out from this filter.

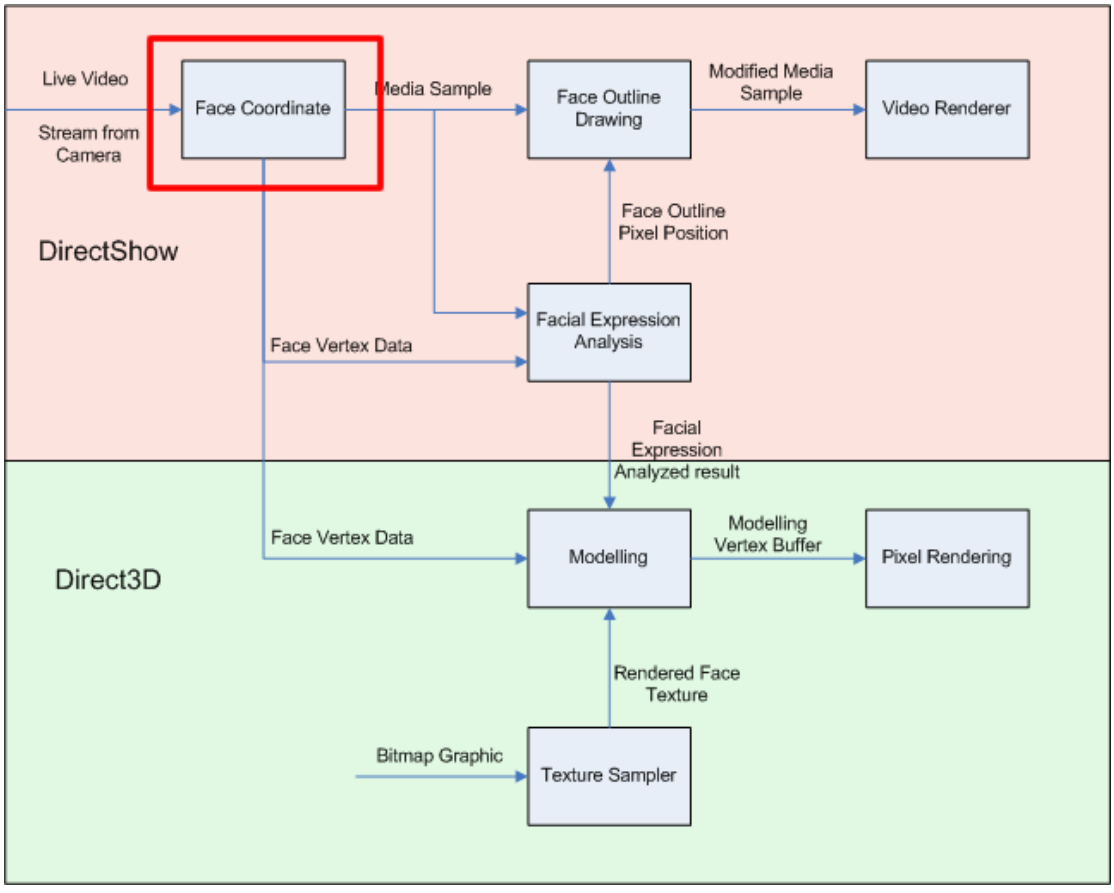

Fig.2.9 The relationship between Face Coordinate filter with our system

The major function of FaceCoordinate filter is to trace human face and then provide the coordinates of the face as an output. In this perspective, a face is being segmented with 100 vertices, which is shown in figure 2.10. A number (0-99) is given to each vertex which is an index for retrieving the particular vertex.

The detection of the face is done by using Active Appearance Models (AAMs) which can match statistical models of appearance to new images.

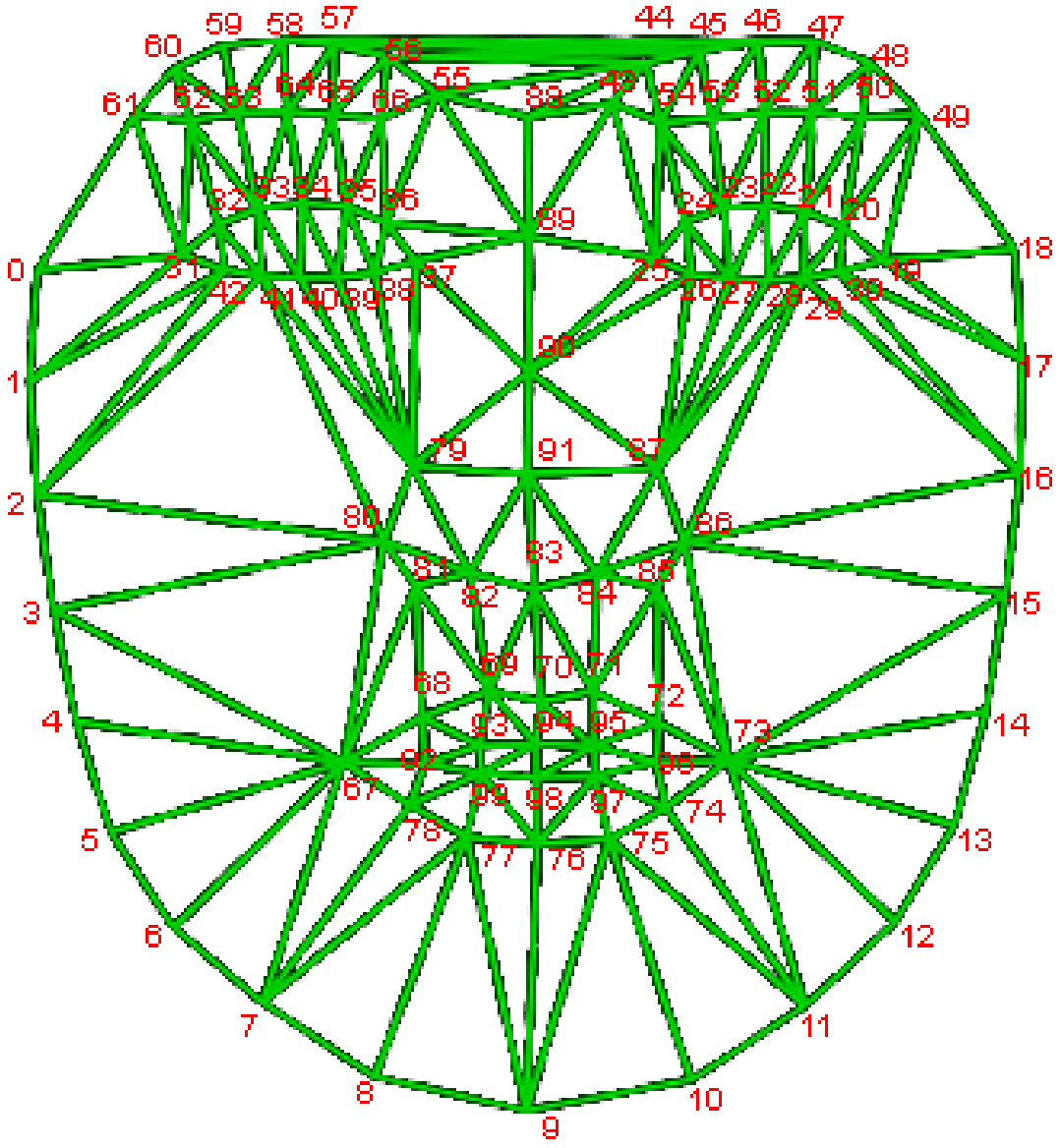

Fig.2.10. Human face mesh which was divided into 100 points and labeled with 0-99

Chapter 3 Implementation

3.1 Introduction

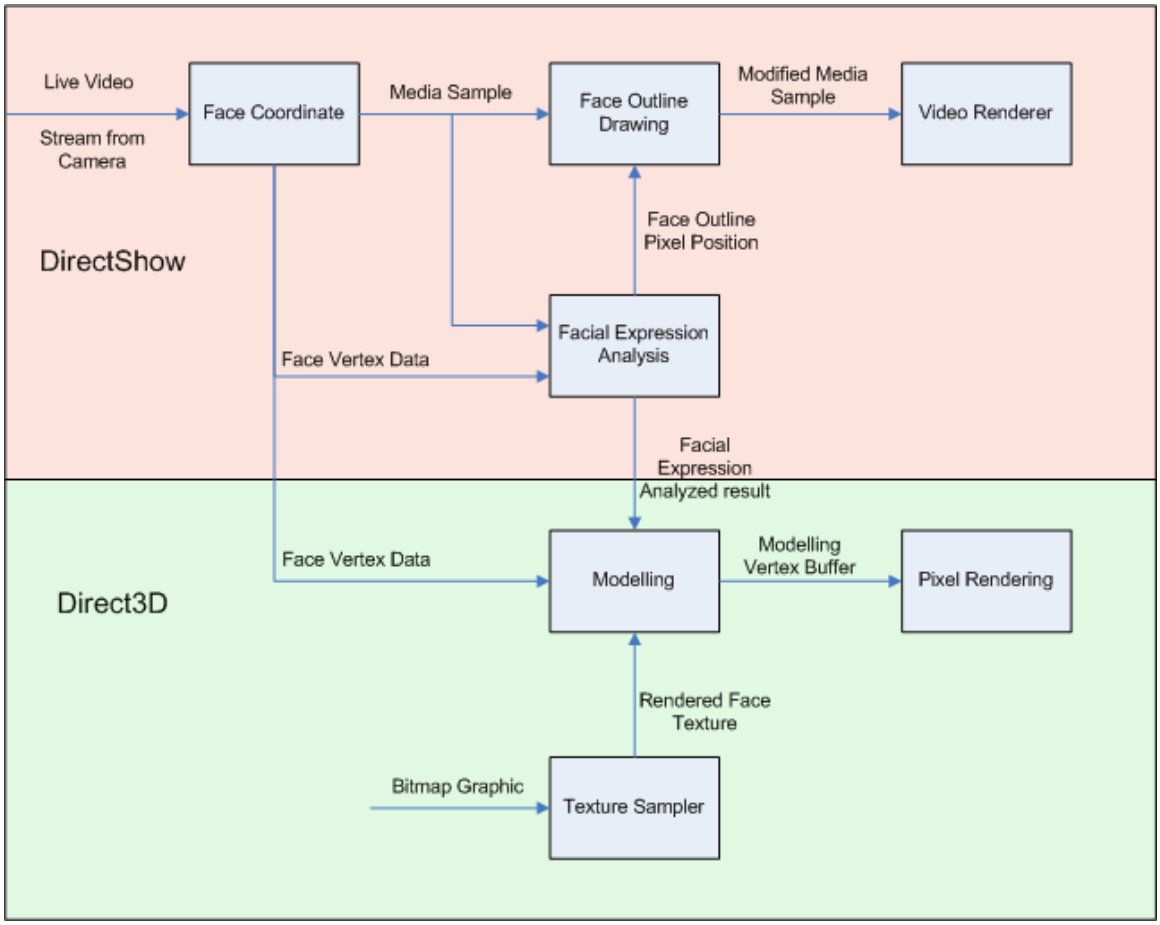

Fig.3.1 Overview of module in the system

There are number of modules under implementation to identify the facial expression and perform modelling on the facial expression. We are currently working on Module "Face Outline Drawing", Module "Facial Expression Analysis", in DirectShow area and Module "Texture Sampler" Module "Facial Expression Modelling" in Direct3D area.

Our system highly relies on the Face Coordinate Filter since the output provided from the filter is one of the important data in our analysis. We make an assumption that Face Coordinate Filter is usually working properly in our system. That is, the face mesh can be fitted accurately.

3.2 Project Modules

3.2.1 Module "Face Outline Drawing"

3.2.1.1 Introduction

In order to make a good prediction of the facial expression from the video source, we need to ensure the FaceCoordinate filter is working as accurate as possible. In order words, we want the face mesh is as close as possible to the human face area. Since our project concentration area is on the facial expression modelling, this module is not going to change the coding or algorithm presented in the FaceCoordinate filter. In this module, what we want to do is to add some indications in the video stream so that the user can manually adjust the face mesh fitting.

3.2.1.2 Basic concepts

In Figure 3.2, we can see that how it would look like when the face mesh is drawing above the human face. However, it is not easy to observe the changes in the face, especially both eyes which are small comparable to the whole face mesh. In addition, the vision effect is no good for the end user.

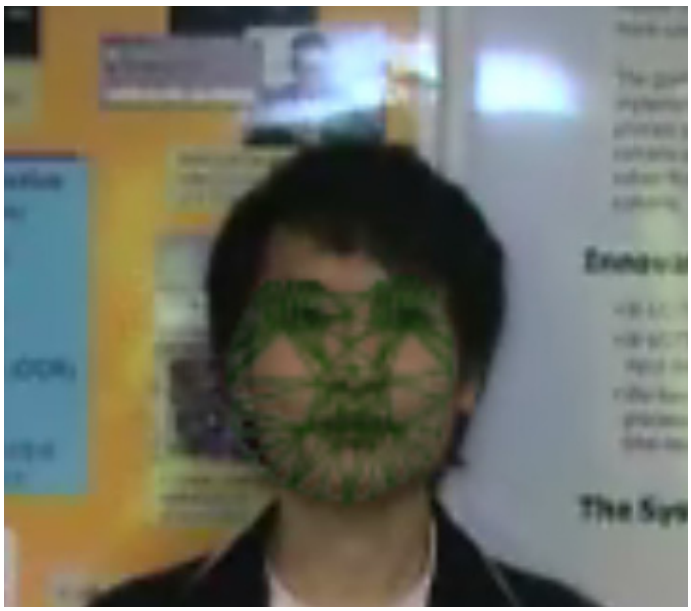

Fig.3.2 The face mesh fitting on the human face (eye opened).

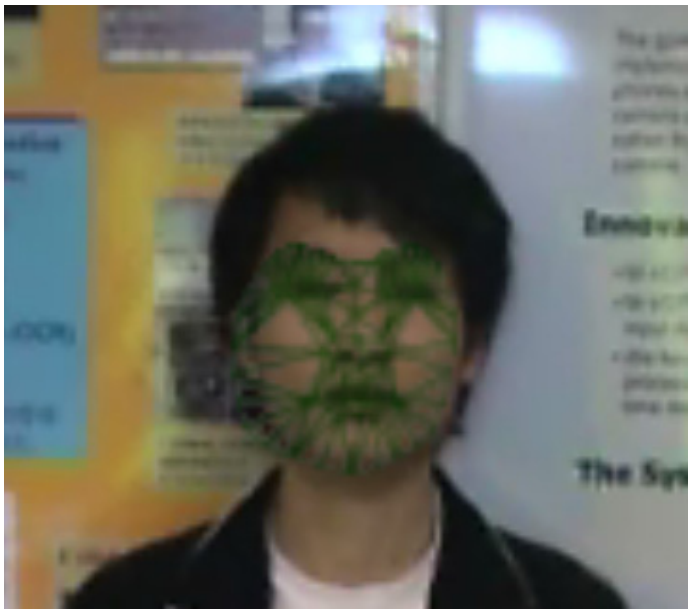

Fig.3.3 The face mesh fitting on the human face (eye closed).

We decide to make it simple to represent the face mesh by showing only the face mesh vertices and the result is as follow.

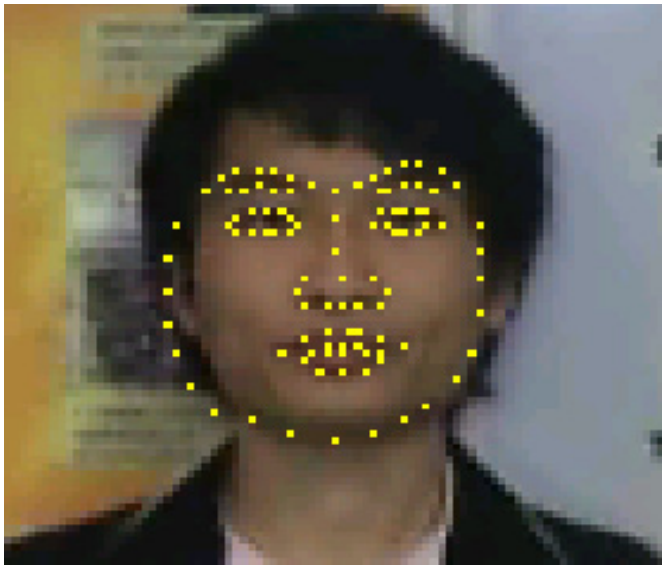

Fig.3.4 Shows all the face mesh vertices on the face.

To make it more simple and easy to use, we further reduce the number of vertex showing on the face to make it just enough to represent the face mesh.

Here is our final version of the simple face mesh.

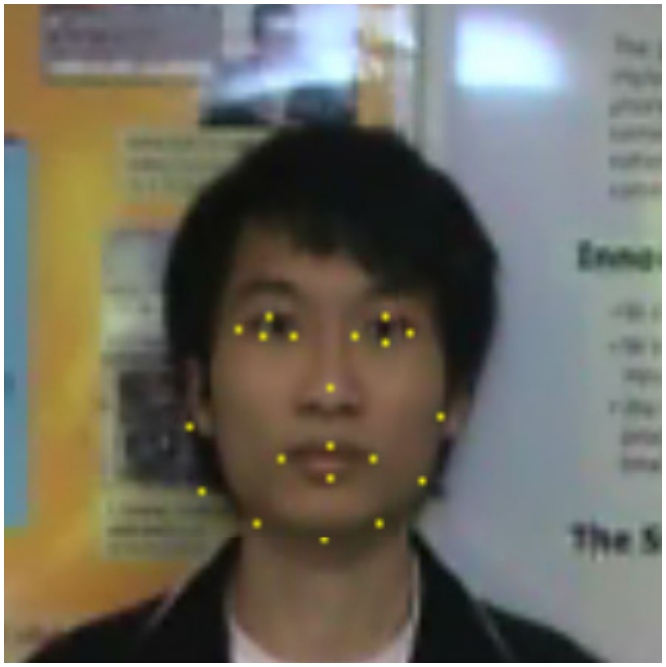

Fig.3.5 The simple face mesh (eye opened)

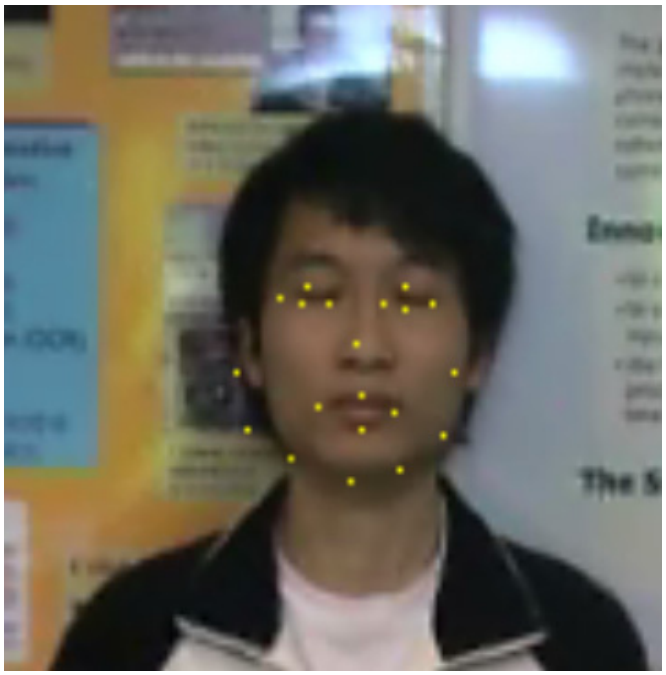

Fig3.5 The simple face mesh (eye closed)

3.2.1.3 Implementation

The first step in this module is to calibrate those face mesh coordinates to pixel coordinate. According to the implementation of the Face Coordinates Filter, the face detect system will provide a hundred points on the face area representing the different face components such as eye, mouth, nose, so and so forth. Those face mesh coordinates is represented in a 2D coordinate system with floating point number. It is, in fact, a 200 data array with the first 100 entries of x-coordinate followed by another 100 entries of y-coordinate. Performing calibration on those face mesh coordinates and re-structure the data format so that we can find out which pixel we would like to mark in the next step. The new x and y coordinate of a point after calibration is given by:

$$\forall iP(i,x) = \lfloor F(i) + 0.5 \rfloor$$
$$\forall iP(i,y) = \lfloor F(i+100) + 0.5 \rfloor \quad \text{where} \quad 0 \le i \le 100$$

We are then getting the frame buffer memory, which are simply arrays of bytes, from the media sample.

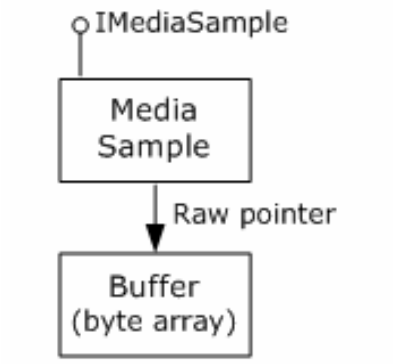

Fig.3.6 Retrieves the pointer to the buffer's memory

According to the DirectX SDK, the bytes placed in the frame buffer is in bottom-up orientation. It means that the buffer start with the bottom row of pixels, followed by the next row up, and so forth. The face mesh coordinate, however, is referring the top-left corner as (0, 0). We need to reverse count the buffer to find out the calibrated face mesh vertices and then alter the pixel value.

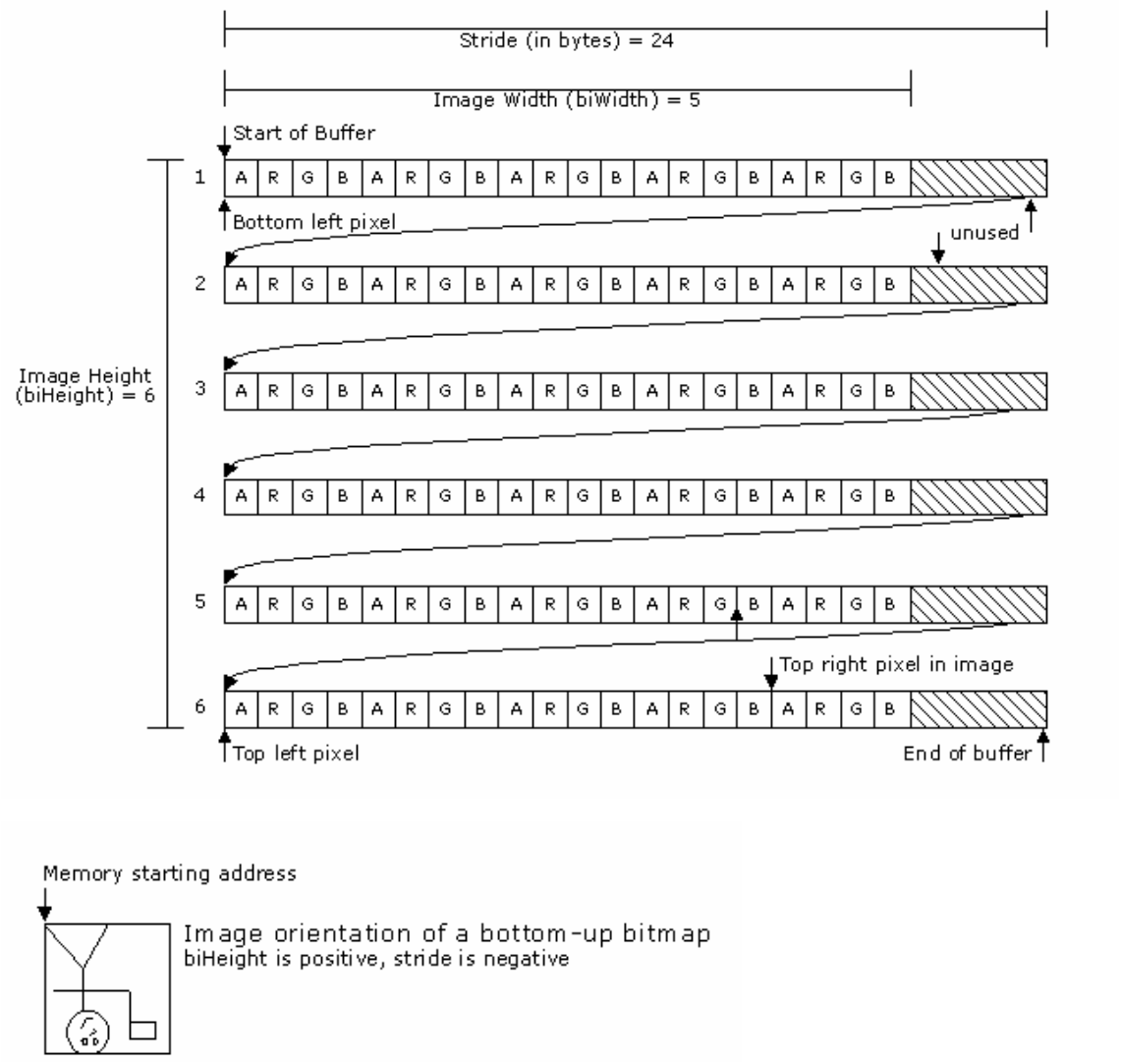

Fig.3.7 Physical layouts of bytes in a 5 x 6 bottom-up ARGB buffer

In our system, we choose yellow to mark out the point in the face. The new colour value of the particular pixel is given by:

$$\begin{aligned} P(i,R) &= 255 \\ P(i,G) &= 255 \qquad \forall i \in FaceOutlineVertex \\ P(i,B) &= 0 \end{aligned}$$

3.2.1.4 Conclusion

After the iteration of altering the pixel, we will get the result in Figure 3.7. Note that this step should be done after the analysis of the facial expression module. It is because this step is directly changing the video frame pixel value which is the important raw data in the analysis process.

3.2.2 Module "Facial Expression Analysis"

3.2.2.1 Introduction

This is one of the major components in our system. The other one is the Modelling module which will be shown in details later on. The critical area in this module is how to distinguish different facial expressions from the video stream source. Since we already have the face coordinate value, the nature way is simply by comparing each value of the coordinates from two consecutive video frames and then predicting the movement of the face components. It is, however, not quite work in most of the time.

3.2.2.2 Detect by coordinate system

3.2.2.2.1 Introduction

Our first goal is to detect the eye blinking and mouth opening by using the coordinates retrieved from the FaceCoordinate filter. We tried to look at the difference between a pair of vertex such as (33, 41) and (34, 40). In the measurement, we need to record a reference distance so as to determine the opening or closing of the face component. The reference distance is recorded at the beginning of the video. The user can also re-initiate the reference distance by clicking the button on the property page of the filter. This step is important because this system can be used to adapt for different people.

A movie is recorded to analyze the variation of the coordinate. The movie includes the motion of eye blinking and mouth opening. We have collected the value of the face mesh coordinate from consecutive video frames. As we want to statistically analyze the coordinate variations, we have plotted graphs as follow. In the graphs, the x-axis is the frame number, which can be imagined as a timeline of the movie. The opening or closing of the face components is highlighted by a square frame.

As shown in figure 3.8, we used three pairs of vertex (33, 41), (34, 40), (35, 39) to compare their distance with the following formula:

$$\Phi = | V_1(y) - V_2(y) |$$

where V(y) is the y-coordinate of the face mesh vertex

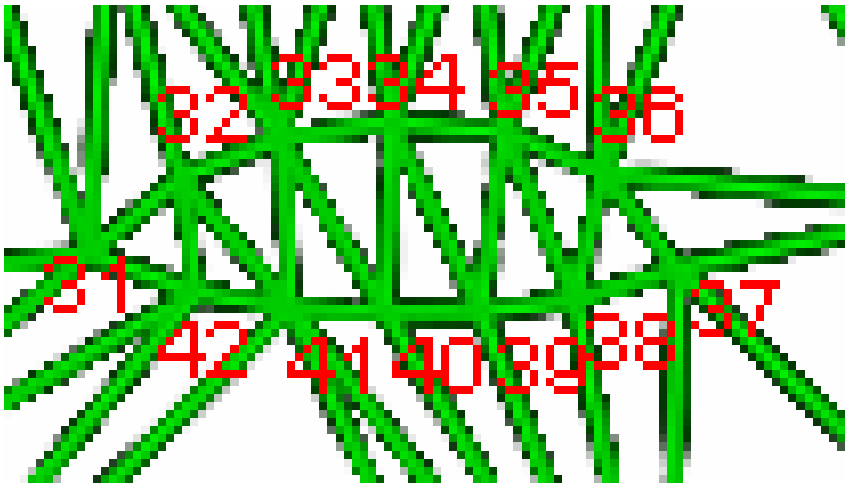

Fig.3.8 Part of face mesh showing left eye

The statistical result is presented in the following figures.

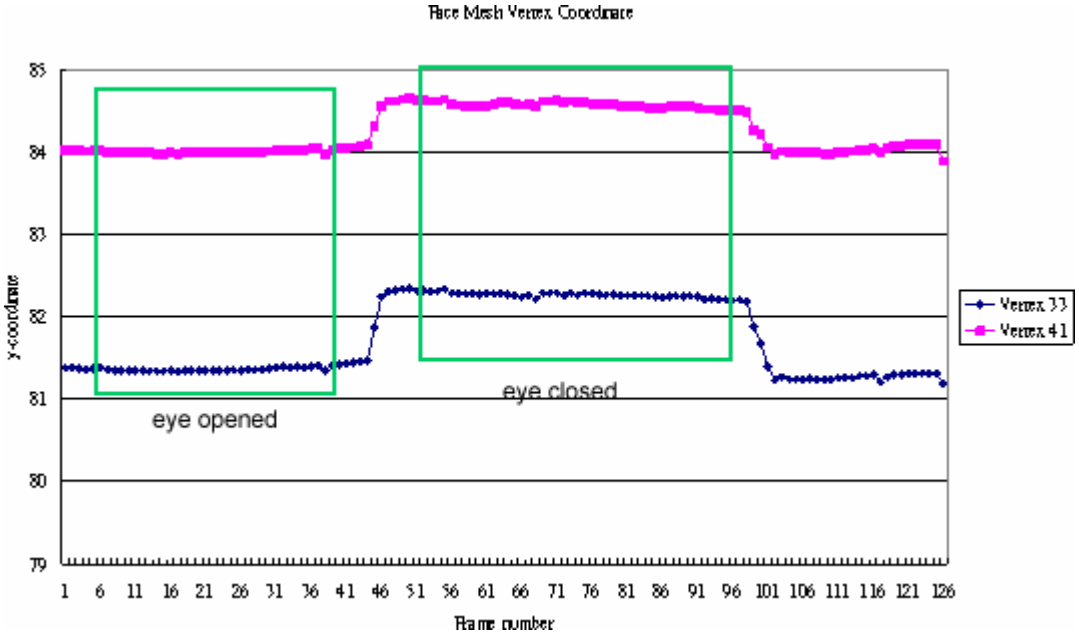

Fig.3.9a The relationships between coordinates of vertex 33 and vertex 41 and timeline (frame number)

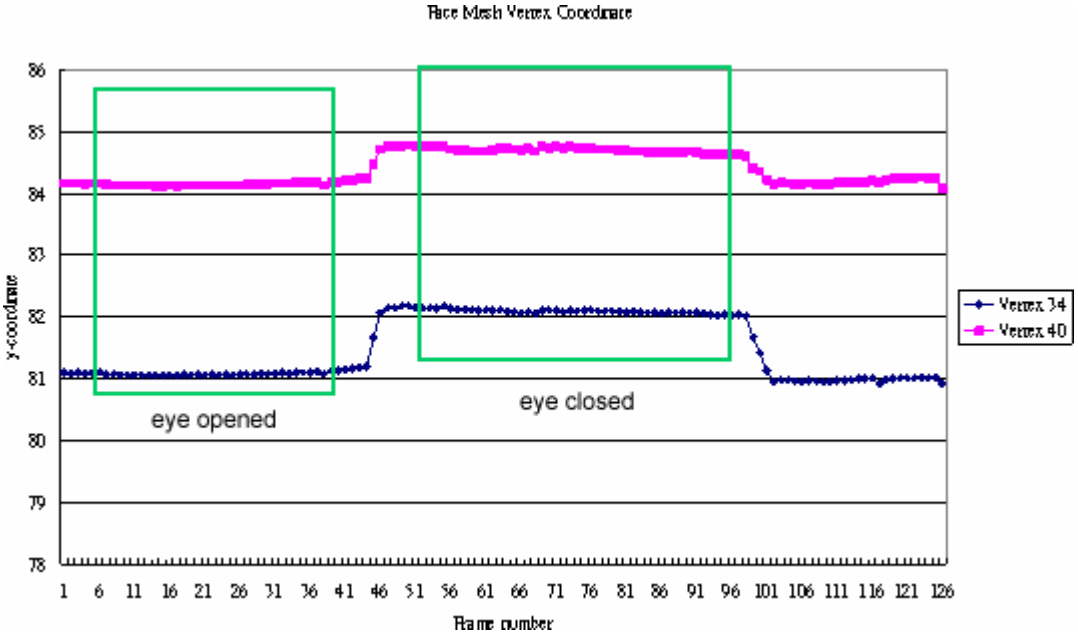

Fig.3.9b The relationships between coordinates of vertex 34 and vertex 40 and timeline (frame number)

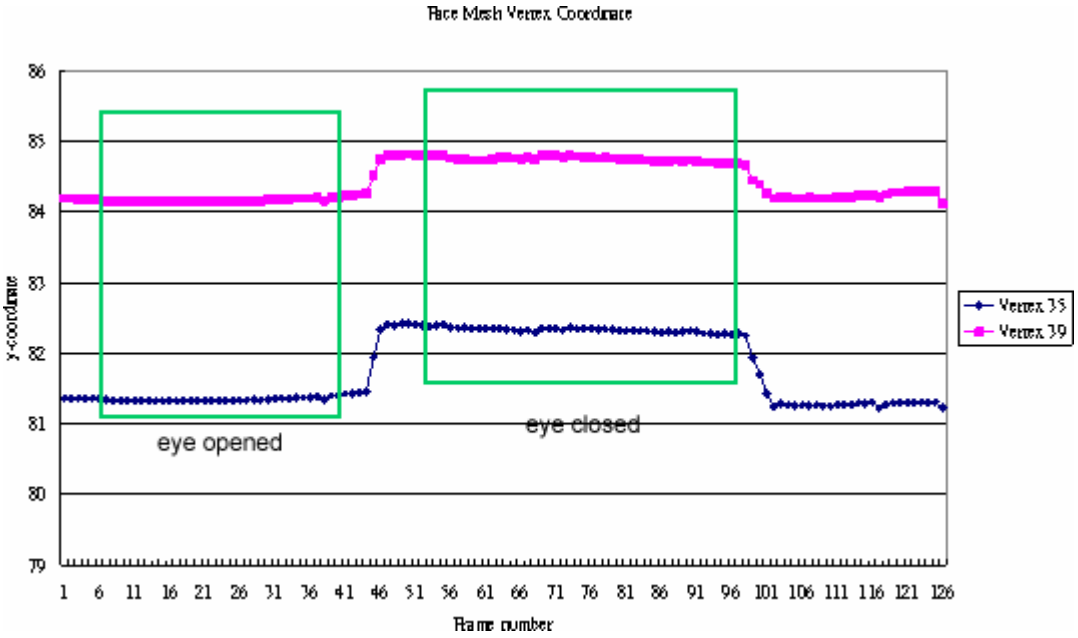

Fig.3.9c The relationships between coordinates of vertex 35 and vertex 39 and timeline (frame number)

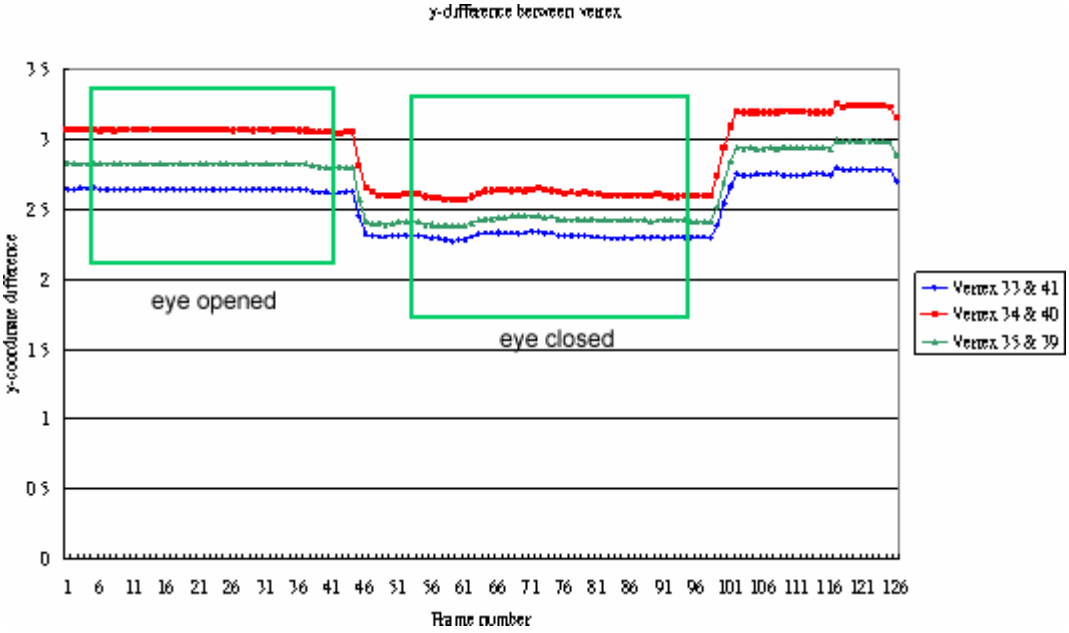

Fig.3.9d The change of y-coordinate difference of 3 vertex pairs when use is opening mouth

Next, we will show the statistical result of the different calculated pairs for mouth opening or closing detection. As shown in figure 3.10, we used totally 6 pairs of vertex to compare their difference. The outer mouth vertex pairs are (69, 77), (70, 76), (71, 75), while the inner mouth vertex pairs are (93, 99), (94, 98), (95, 97).

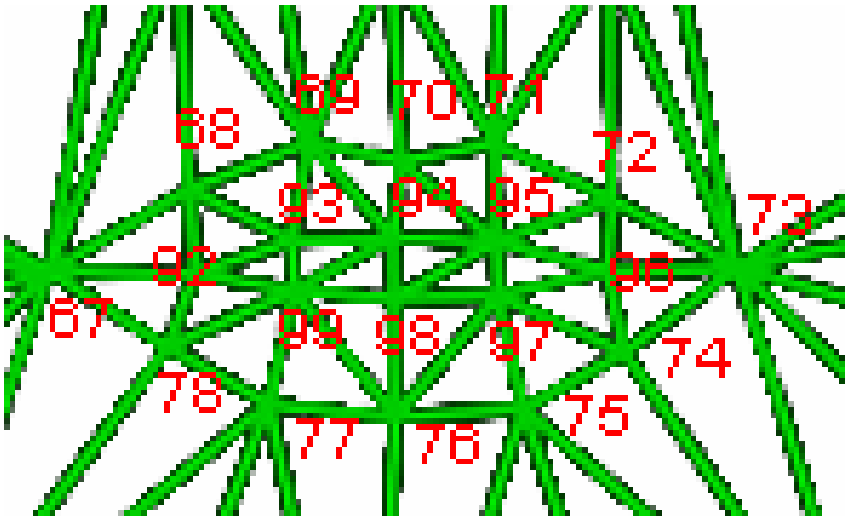

Fig.3.10 Part of face mesh showing the lips

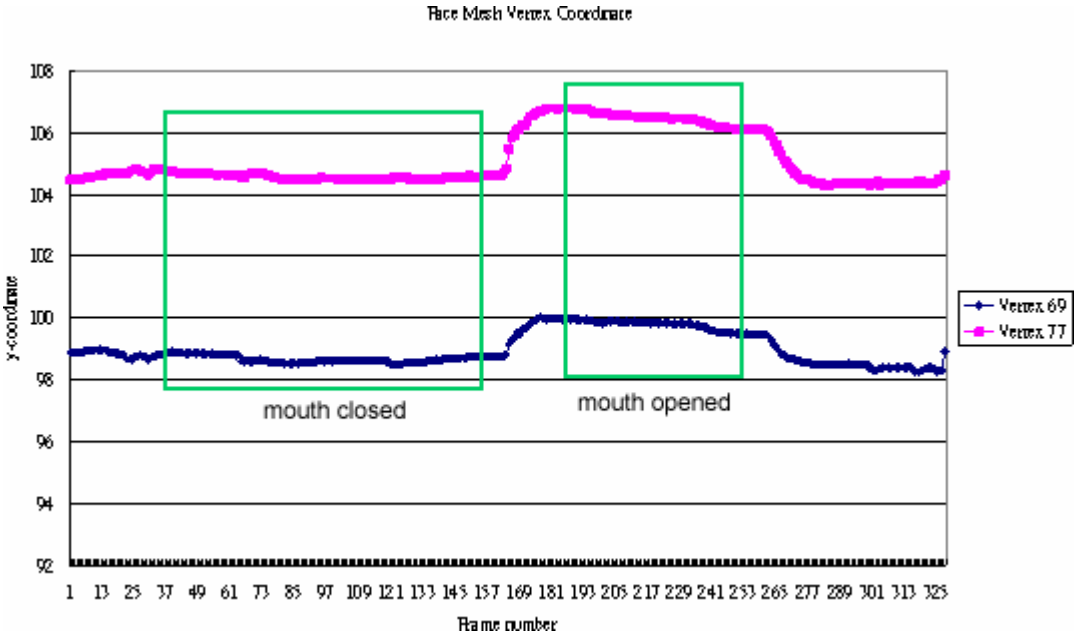

Fig.3.11a The relationships between coordinates of vertex 69 and vertex 77 and timeline (frame number)

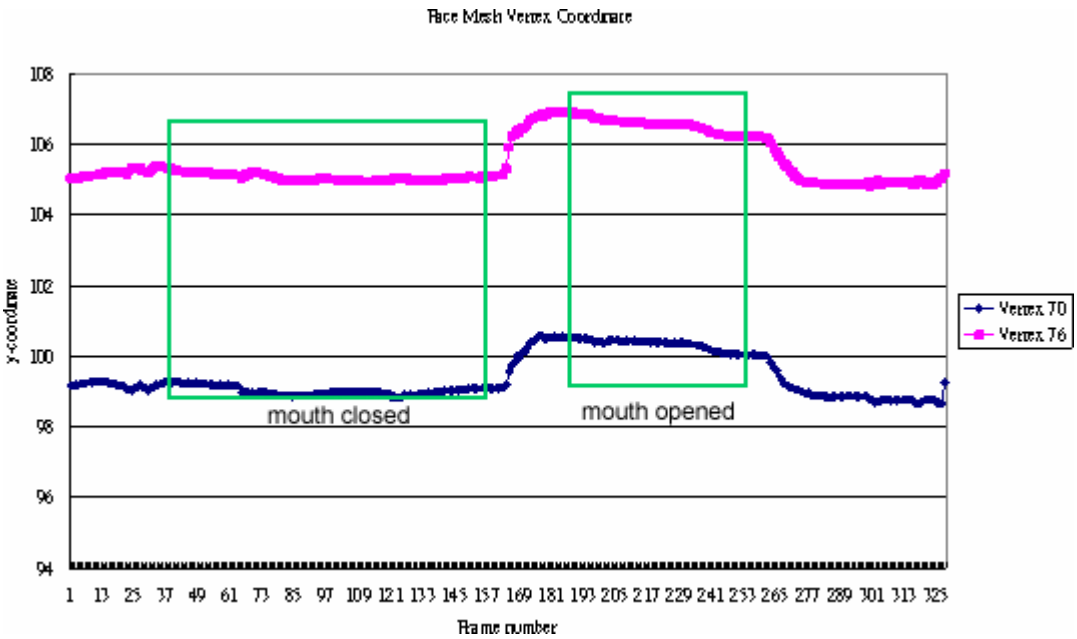

Fig.3.11b The relationships between coordinates of vertex 70 and vertex 76 and timeline (frame number)

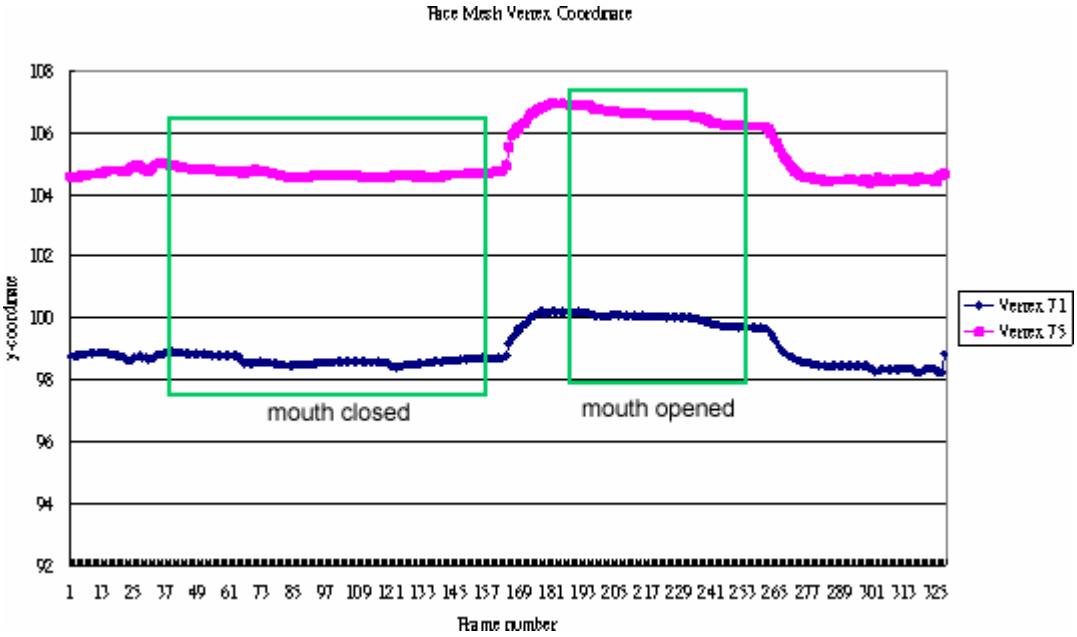

Fig.3.11c The relationships between coordinates of vertex 71 and vertex 75 and timeline (frame number)

Fig.3.11d The change of y-coordinate difference of 3 vertex pairs when use is opening mouth

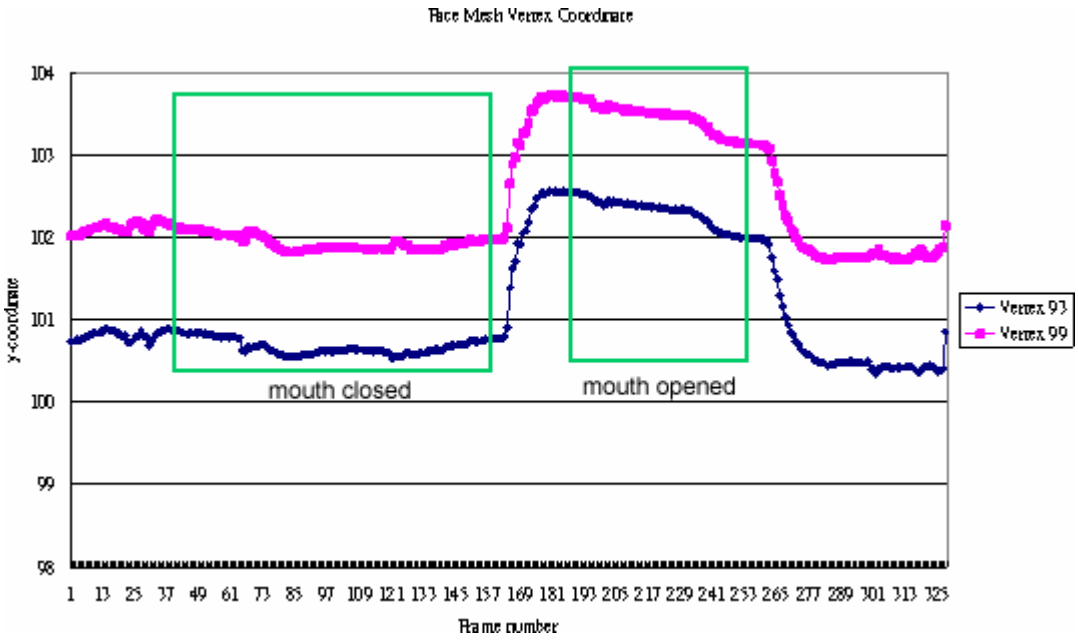

Fig.3.12a The relationships between coordinates of vertex 93 and vertex 99 and timeline (frame number)

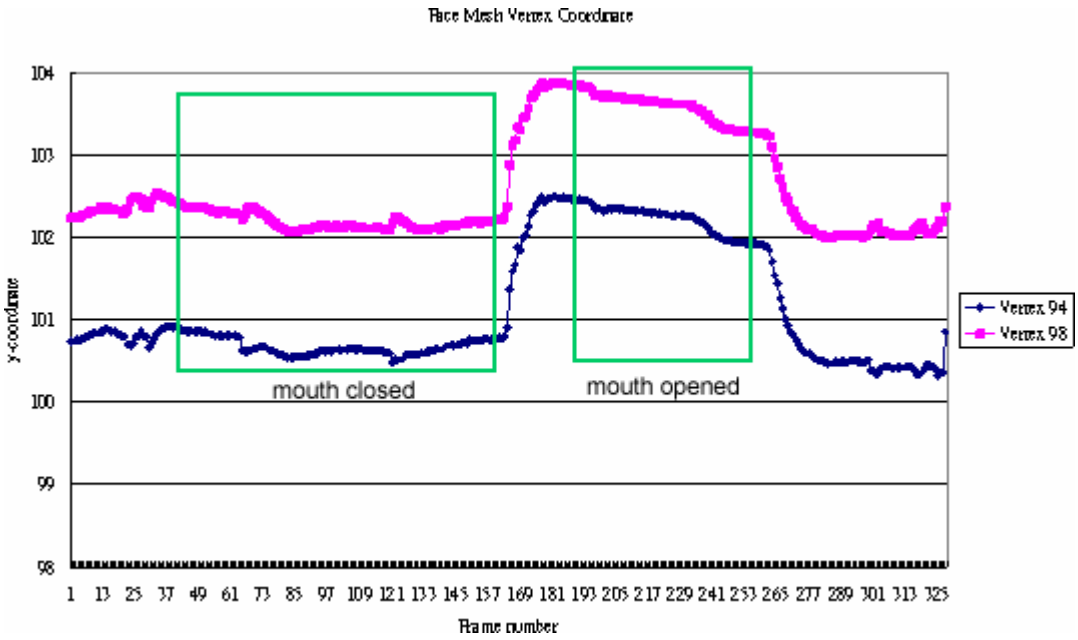

Fig.3.12b The relationships between coordinates of vertex 94 and vertex 98 and timeline (frame number)

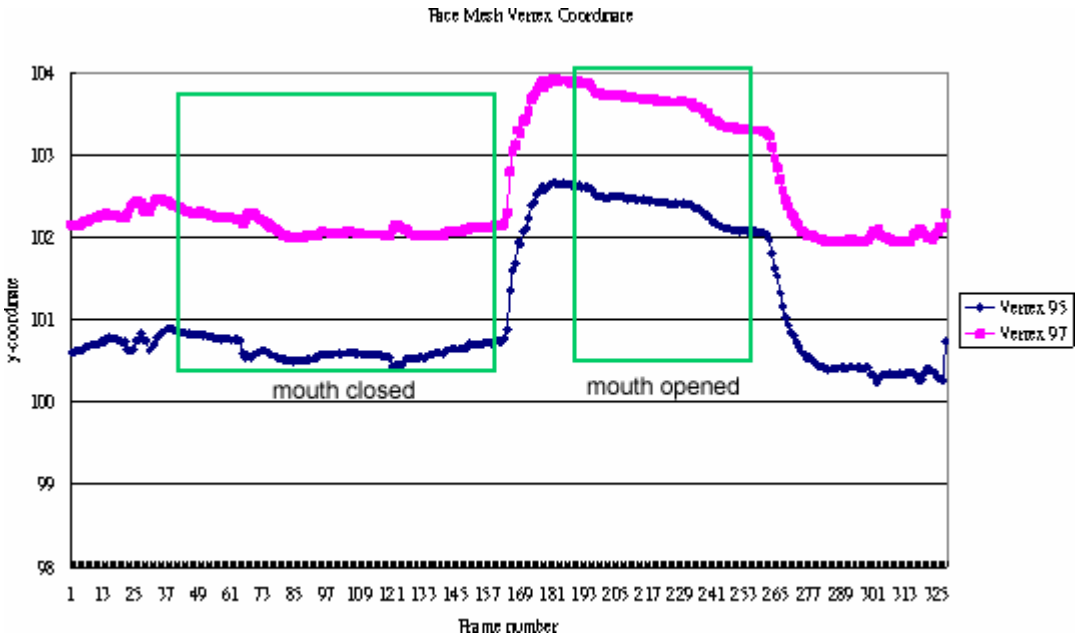

Fig.3.12c The relationships between coordinates of vertex 95 and vertex 97 and timeline (frame number)

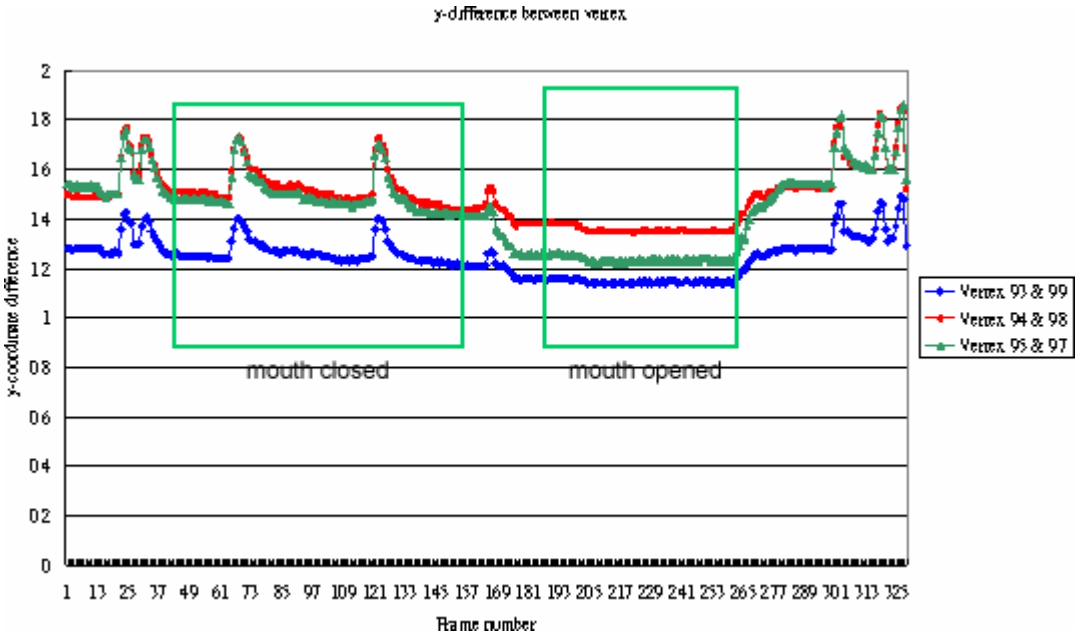

Fig.3.12d The change of y-coordinate difference of 3 vertex pairs when use is opening mouth

3.2.2.2.2 Interpretation

We can see that the average difference before opening mouth and closing mouth, or opening eye or closing eye, is just within 1 unit. This range is too small to distinguish the opening or closing of the face components.

There are many factors that can affect the difference obtained. If the user gets closer to the camera, the difference of the face components is increased because the human face is being magnified (nature of the camera). While the user gets far away from the camera, the difference of the face components is decreased because the human face is being minified. We have considered a solution that we would like to compare the ratio of y-coordinate difference to other y-coordinate difference of face component such as the y difference of the whole face, which has replaced using absolute value.

Another factor is that, if the user tries to move his or her head quickly, the mesh will fluctuate and the difference measured will not be precise. The reason of fluctuation is due to the fact that the frame rate is not high enough to allow Face Coordinate filter to detect the face.

3.2.2.2.3 Conclusion

We would like to conclude that, by using the coordinate system to determine whether the face components is opening or closing will result a high error rate. Nevertheless, it is a simple way to have a rough prediction of the movement and we will use it as a reference data.

Here, we propose three different approaches to identify the facial expression. Before showing the details, we first clarify the facial expression we are going to implement. The 100 face mesh coordinates we got from the FaceCoordinate filter is concentrated on the eye, nose, mouth and the face outline. Our study follows this as well and is going to determine the variation of the mouth, eye and the whole face.

To begin with, we segment the human face into 172 pieces of irregular triangle based on the 100 face mesh coordinates.

3.2.2.3 Competitive area ratio

3.2.2.3.1 Introduction

We have observed that when user opens his mouth, the triangles under the mouth are getting smaller. At the same time, the triangles marked as red in the figure 3.13 is getting larger. We want to compare these two sets of area to see the ratio between them. However, the triangles are irregular and we only have the coordinates of three vertices in each triangle. To calculate to area of the triangle, we have used the following approach.

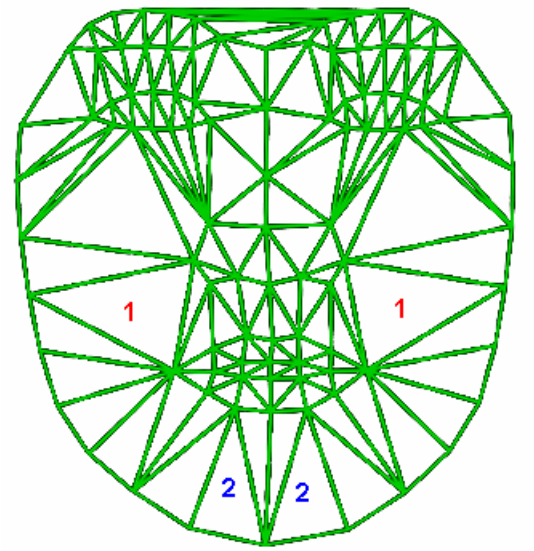

Fig.3.13 Face mesh marked by 1 and 2, which indicates triangle gong to be compared.

3.2.2.3.2 Interpretation

Consider an irregular triangle with vertices of arbitrary coordinate $(x_1, y_1), (x_2, y_2), (x_3, y_3)$

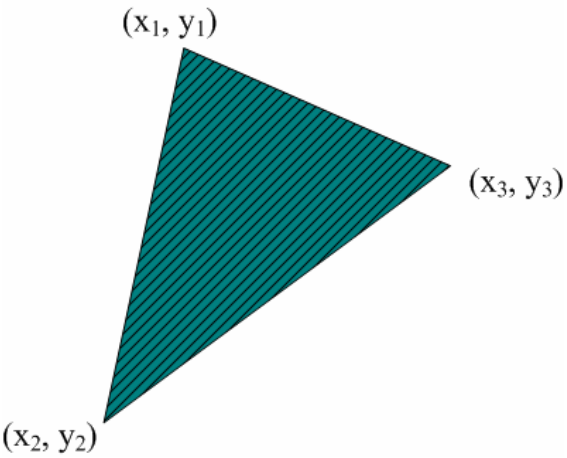

Fig.3.14a A triangle example showing how area is being calculated.

By applying the Cramer's Rule, we get

$$A = (\frac{1}{2})\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$ where A is the area of the triangle

The determinant of the matrix is

$$D = x_1 y_1 + x_2 y_3 + x_3 y_1 - x_1 y_3 - x_2 y_1 - x_3 y_2$$

It can be factored to:

$$D = (x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3)$$

So we can finally get the area of the triangle by

$$\frac{1}{2} D = \frac{1}{2}(x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3)$$

By comparing the ratio of triangles 1 and blue triangles 2 indicated in figure 3.13, we can know the mouth is opening when the ratio is larger than threshold.

After collecting the coordinate's information we can do the area calculation using the above formula. After that, we will record relationship of the area ratio between triangles marked by 1 and triangles marked by 2, with the frame number.

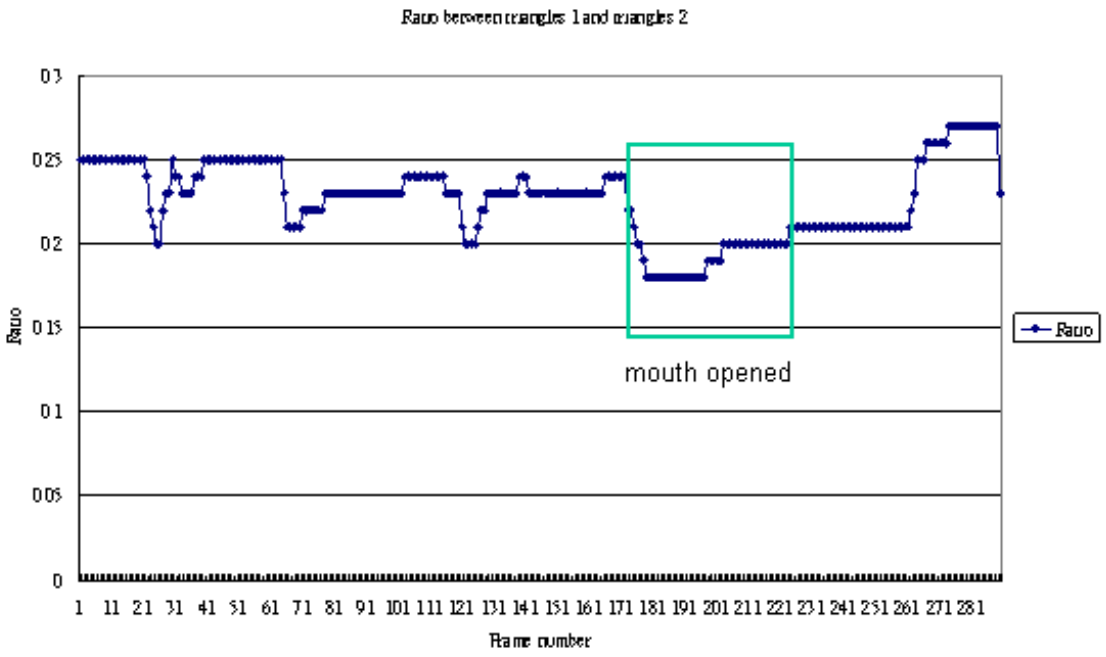

Fig.3.14b The ratio analysis between triangles

The threshold is statically obtained after sampling few movie clips. In figure 3.14b, it is obvious that the ratio drops under 0.2 of the mouth is opening.

3.2.2.3.3 Conclusion

Competitive area ratio so far is an acceptable algorithm to find out the facial expressions. However, we must work out experiments as much as possible to obtain the changes of the human faces.

Also, we realize that different people would have different face area ratio. We try to make this system becomes more generic. That is, allow most people to play with this system. We should do initialization before user tries using this system. The initialization mainly record normalize original ratio before doing any facial expressions, and record the new ratio if user does different kinds of facial expressions.

3.2.2.4 Horizontal eye-balance

We want to measure whether the head is inclined. An example of head inclination is shown in figure 3.15.

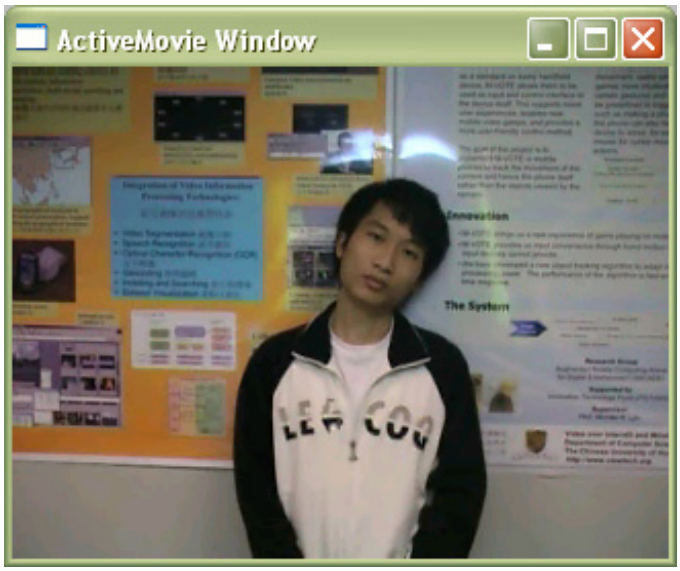

Fig.3.15 User's head is inclined

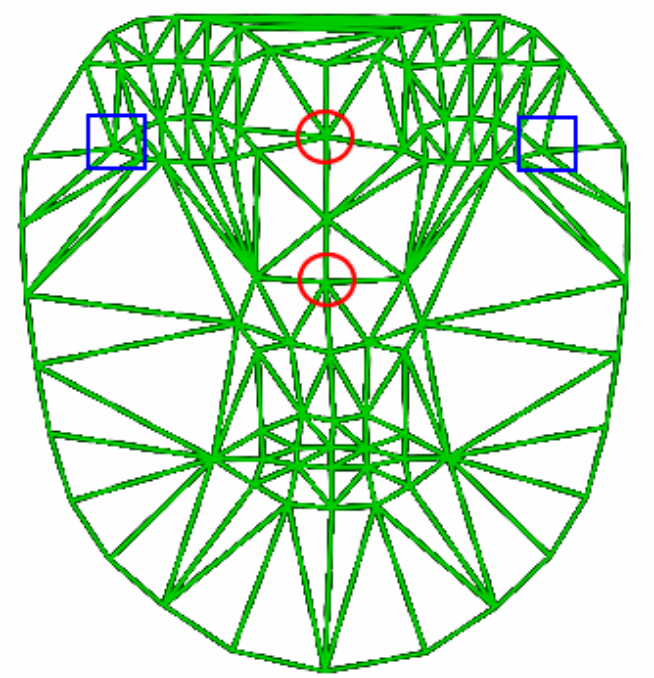

Fig.3.16 Face mesh with circles and squares to indicate the referenced vertices. Vertices surrounded by Circle: (89, 91), Vertices surrounded by Square (19, 31).

The first approach we used is using the angle between the line connected by the vertices in circles shown in figure 3.16 and the horizontal axis.

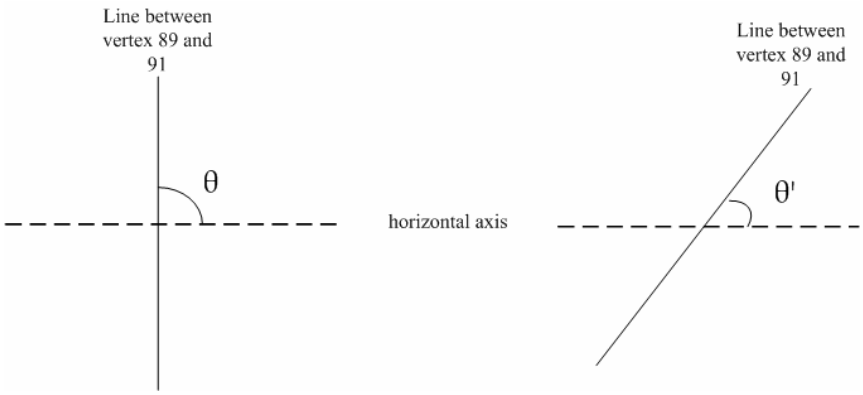

Fig.3.17. Calculation the angle using vertex (89, 91)

By comparing the angel measured, we can determine whether the head is inclined. However, this approach is not accurate if the user tries to turn his head to the left or to the right. Figure 3.18 shows the sampled line. Thus, the accuracy is poor by using this approach.

Fig.3.18 An example of inclination of the line without head inclination (head turning)

To avoid the above problem, we used another two referenced vertices (bounded by square in figure 3.16) to measure the angle between the line and the horizontal axis.

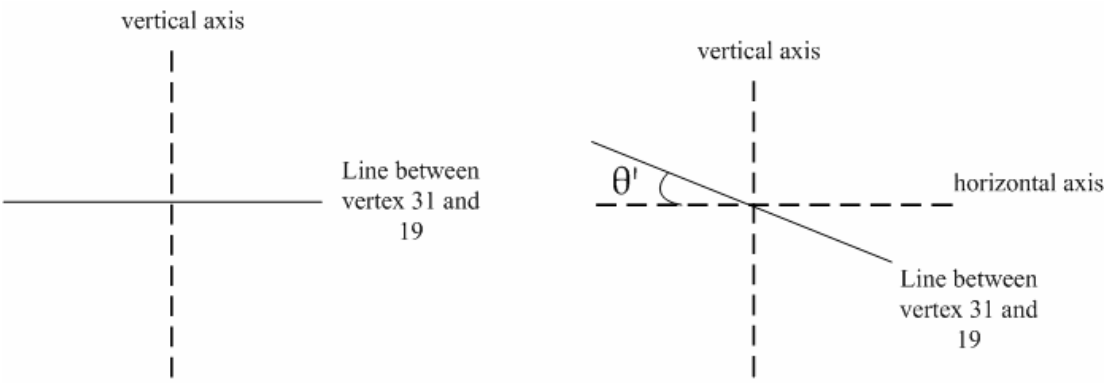

Fig.3.19 Calculation the angle using vertex (19, 31)

We use the following formula to calculate the angle,

$$\theta' = a\tan\left(\frac{|y_1 - y_2|}{|x_1 - x_2|}\right)$$

If $y_1 < y_2$, it indicates head is inclined to right

Else if $y_1 > y_2$, it indicates the head is inclined to left.

3.2.2.5 Nearest-colour convergence

3.2.2.5.1 Introduction

In the following figure, we want to find out whether the eye is closed or not. To start with, we consider the red area of the eye and make an assumption that the iris is nearly black in colour.

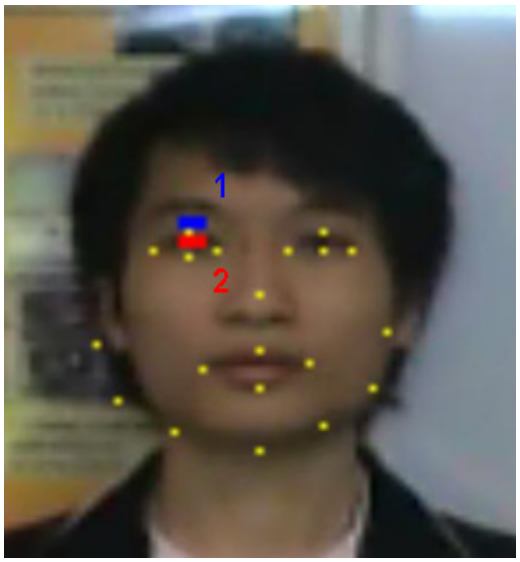

Fig.3.20 The sampling area of collection colour

3.2.2.5.2 Interpretation

As the video source is stored in RGB format, we treat the pixel colour into three different channel and store the value of each channel within the red area (indicated by 2). Then, we compute the average value in each channel so as to minimize the variation of the colour value. The average value is given by:

$$r_1 = \frac{\sum_{i=1}^{k}(i,R)}{k}$$

$$g_1 = \frac{\sum_{i=1}^{k}(i,G)}{k}$$

$$b_1 = \frac{\sum_{i=1}^{k}(i,B)}{k}$$

We can think of this result as if producing a new pixel with colour value (r_1, g_1, b_1). Besides, we would have a threshold colour value which indicates the colour should be contained in an opened eye. We, then, map the colour value into a 3-D coordinate system in such a way that r = x-coordinate, g = y-coordinate, b = z-coordinate. Using this coordinate geometry, we can find out the difference between the new pixel colour and the threshold colour by finding their distance in the 3-D space. The difference is given by:

$$diff. = \sqrt{(r_1 - r_\Phi)^2 + (g_1 - g_\Phi)^2 + (b_1 - b_\Phi)^2}$$

and the difference would be accepted if it has the following property:

$$0 \le diff \le \Phi$$

3.2.2.5.2 Conclusion

However, light intensity is different under different environment. Besides, different face may have different colour iris. To achieve a better accuracy, we have modified the threshold to become a colour sampling from the blue area (indicated by 1). This area is taken for analysis as they would have the nearest colour when the eye is closed. The new threshold value can be computed directly by applying those equations in the above and the other equation would be held.

3.2.2.6 Conclusion on face analysis algorithm

With the above three different approaches, our system can now detect three kinds of facial expression. To detect each facial expression, we may use more than one approach to maximize the accuracy.

There exists many factors affect the result obtained from the face analysis algorithm. Thus, we apply different algorithm as to compensate the weakness of the other algorithm.

We still try to think about whether some other new algorithm can detect a more interest facial expression or optimize the accuracy of our system. To achieve this, we need to do many statistic researches so as to find out the pattern of the data records.

3.2.3 Module "Texture Sampler"

3.2.3.1 Introduction

In the rest of our system, these two modules are implemented so as to reflect the facial expression result generated from the previous module. We do this by making use of the Direct3D texture map capability.

3.2.3.2 Basic concepts

3.2.3.2.1 Texture Mapping

Texture behaves like bitmap graphic which uses two-dimensional array to store the colour value. In texture perspective, each colour value is called a textual element (Texel). Similar to a pixel in bitmap graphic, each Texel has its own address as a position indicator. There are two components, which can be imagined as a column number (u) and a row number (v).

Texture coordinates are used in texture space. The coordinates are representing the point relatively to the point (0, 0) in the texture. Before a texture is applied to a primitive in 3-D environment, the Texel addresses must be translated into screen coordinates or pixel locations.

Microsoft Direct3D uses inverse mapping to map the texels directly to the screen space. It is done by finding corresponding Texel position in each screen space pixel. The colour around the Texel position is being sampled to the screen space.

A generic addressing scheme is applied to Texel address as to provide a uniform address range for all textures. Under this addressing scheme, the vertex resides in the range, 0.0 to 1.0, inclusively. In our project, we normalize the coordinates in bitmap to this addressing scheme by dividing the x and y position of the pixel by the width and height of the bitmap graphic respectively.

In figure 3.21 we can see that with the same texture address (0.5, 1.0), we sampling different coordinate in different texture. That is, (2, 4) in texture 1 while (3, 6) in texture 2.

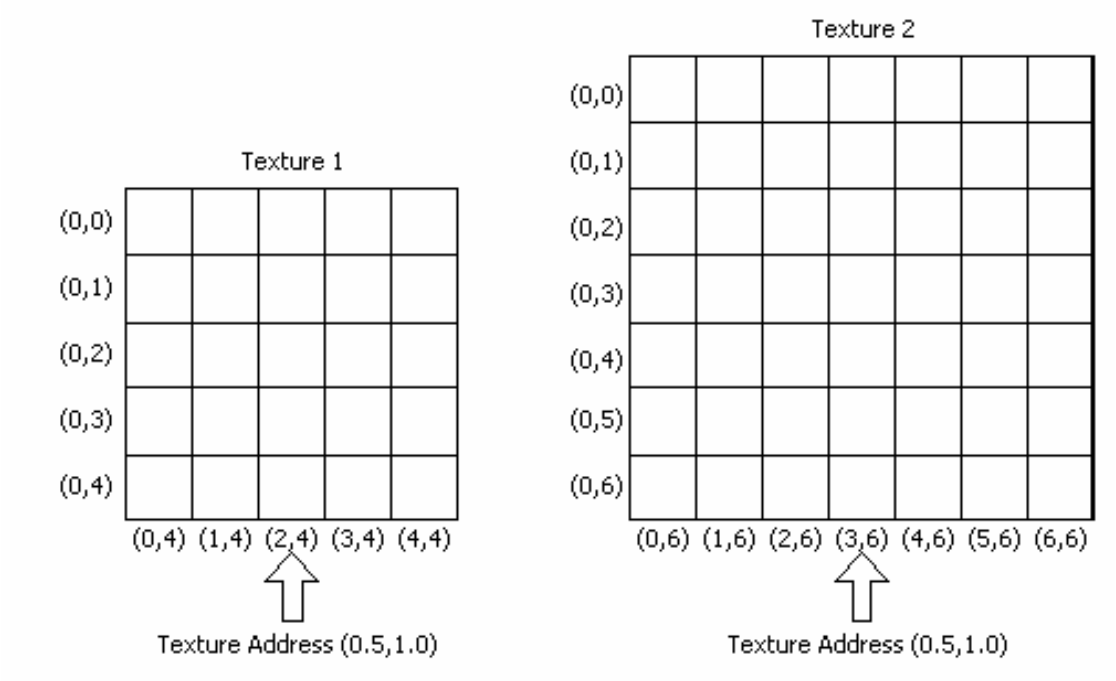

Fig.3.21. Above picture shows different textures address are pointed by same texture address

When the texture is going to map onto the primitive surface, it uses the addresses of four corners in a pixel to map with the 3-D primitive in object space. The pixel will be distorted since due the shape of primitive and the viewing angle.

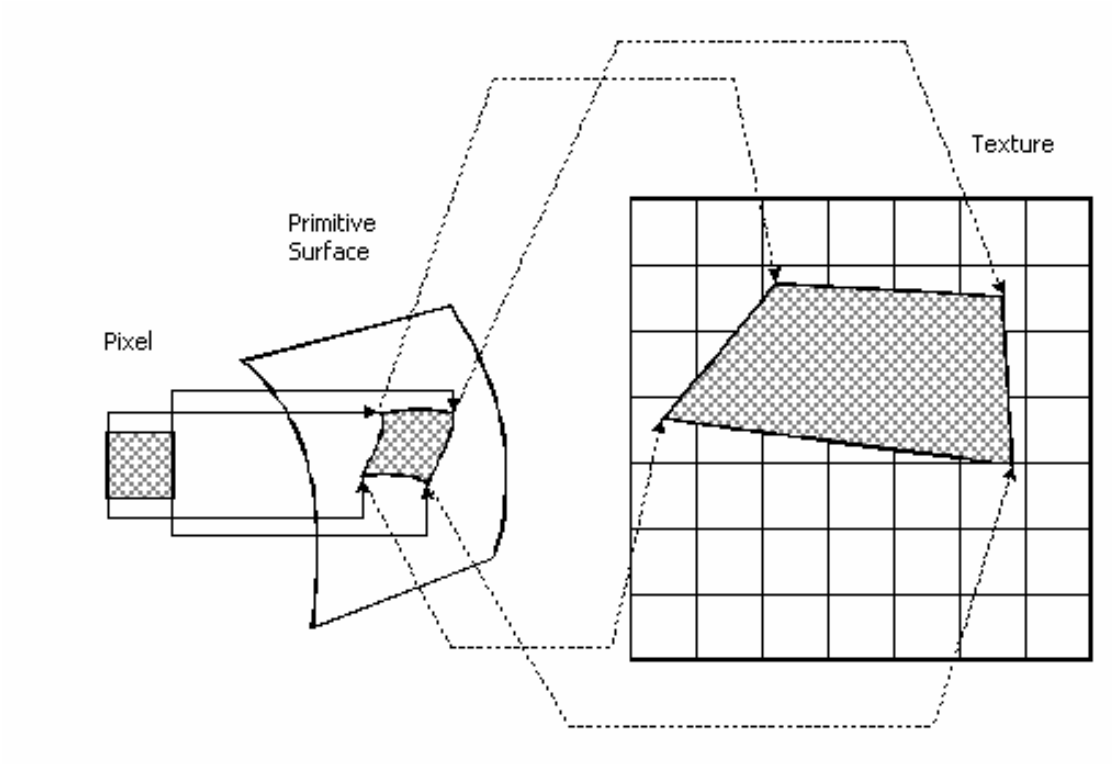

Fig.3.22. The distortion of a pixel when it was pasted on a primitive surface

The computation of pixel colour is done by collecting texels colour in the area which the pixel maps. There are a few texture filtering methods which determine how Direct3D use the texture to create a colour for a pixel.

The texture is being magnified or minified when texture filtering is being performed. There could be two possible results. That are, more than one pixel is being mapped into one Texel, or one Texel is being mapped into one pixel. However, these effects will make the image become blurry or aliased. To avoid such situation, blending must be done on Texel colours before we apply it to pixel. We are using the Direct3D default option, nearest-point sampling to simplify the filter operation.

3.2.3.2.2 3-D primitive

Texture only provides a kind of material on the surface. It likes a colour panel where we can choose a colour to be displayed. But how the shapes of object to be draw on the screen depending on what primitive is used.

We often refer 3-D primitives as a polygon in which there are at least three vertices. A triangle is the most common primitives and is used in our system.

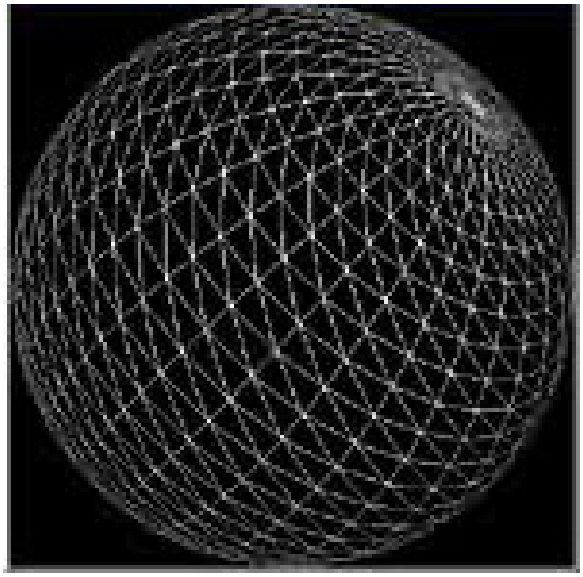

Fig.3.23 A wire frame sphere created by triangle unit

3.2.3.2.3 Vertex buffer and index buffer

Vertex buffer objects enable applications to directly access the memory allocated for vertex data which is used to represent the coordinate we want to draw on the scene. For example if we want to draw a triangle, we need to provide the 3 vertices coordinate to the Direct3D device.

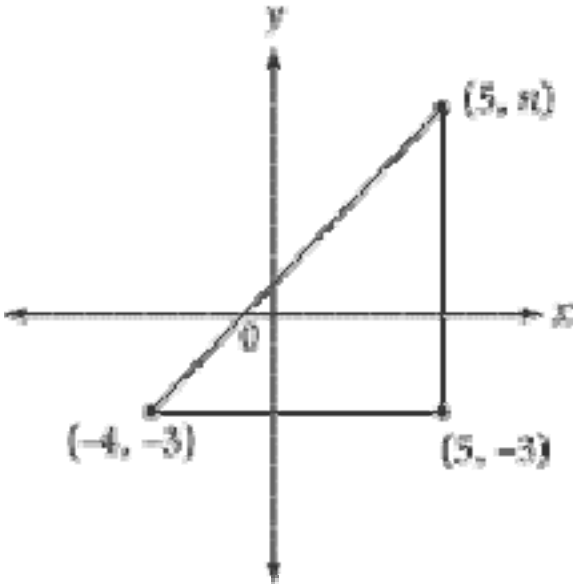

Fig.3.24 drawing a triangle using three vertices

The face mesh contains different size of triangle and each vertex is used to represent more than one triangle. There are many common vertices in the mesh.

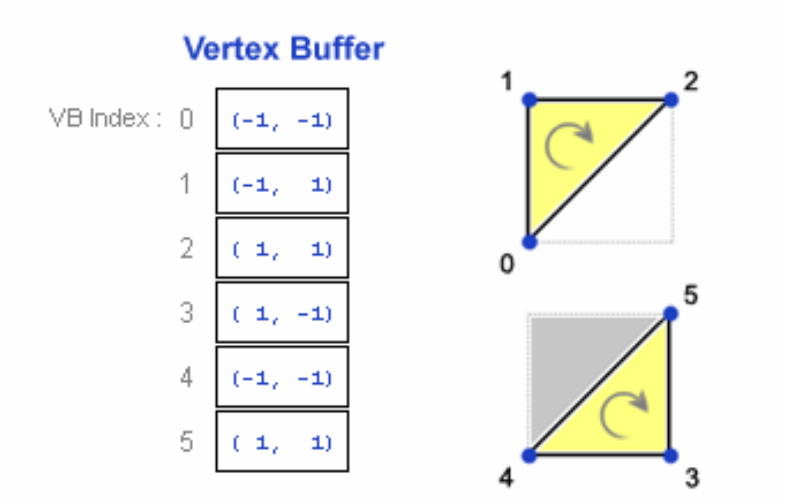

Fig.3.25 using vertex buffer to represent a square

Therefore, if we represent each triangle vertices separately, there would be a few hundred of data in the vertex buffer. In order to avoid duplicate data, we introduce the index buffer to compress the vertex buffer. Another important advantage it takes is to allow the display adapter to store those vertices in a vertex cache. The display adapter can fetch those recently used vertex data from the cache which enhance the performance greatly. Here is an example,

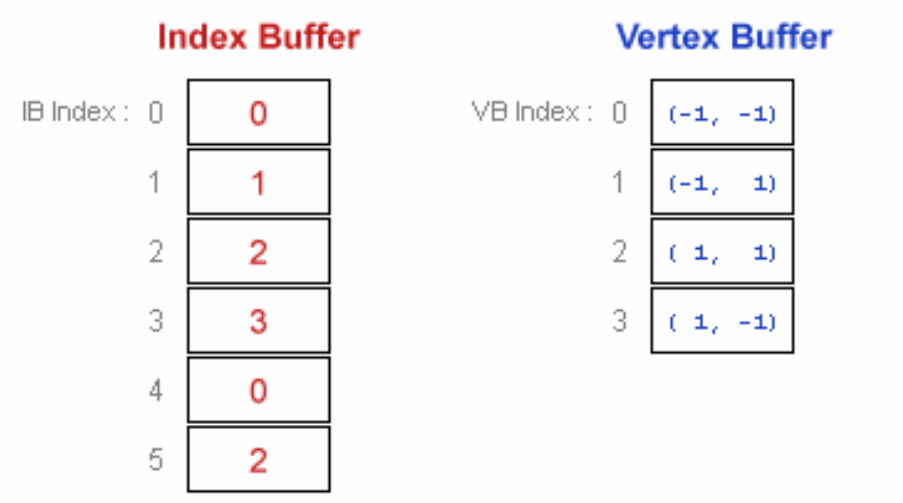

Fig.3.26 using index buffer to represent the square

3.2.3.3. Implementation

Pre-production of texture

Our approach is using a predefined image to associate with one facial expression and a set of images would be called a character model. A pre-production process is done in this purpose. We collect a few image of the same character and manually modify them to represent the different facial expression and their combination. There are currently two character models to be used in the system.

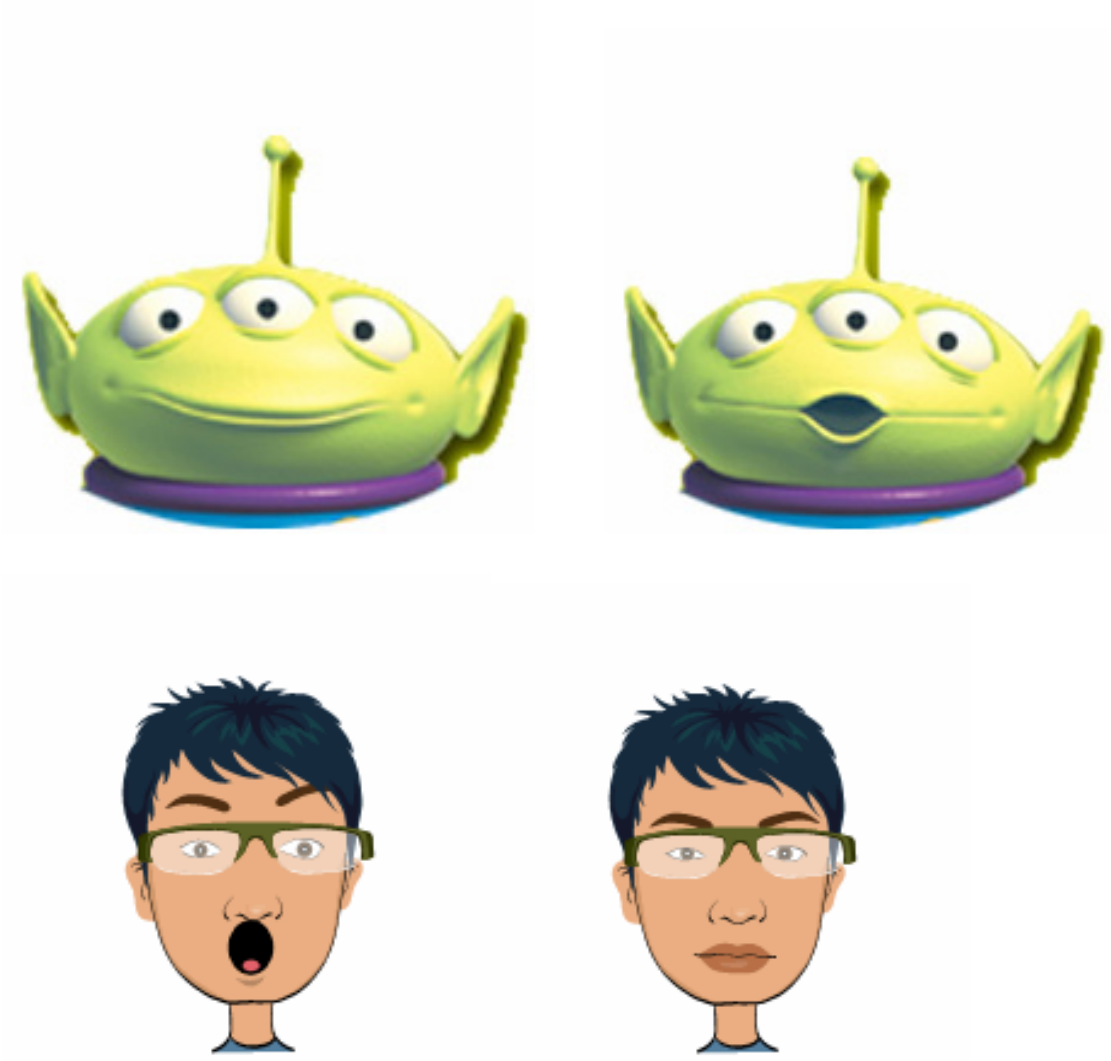

Fig.3.27 Some examples of facial expression in the two character models

Loading the texture

The next stage is to create the texture from those file-based images. For each image, we need to create a corresponding texture map. In order to have a smooth running when switching the character model during execution of the application, this step should be done before the application starting to run.

Mapping object coordinate to Texel coordinates

After this initialization, we will have our own texture resource database and is ready to be used in the system. The next step is to define a modelling vertex data to map the face mesh coordinates. The modelling vertex data would contain 100 modelling vertices in order to have a 1 to 1 mapping of the face mesh coordinates. The modelling vertex data will be further used by the next module to render the character. In each vertex, we need to have two set of coordinates, one is object coordinate used to draw geometric shapes in the 3D scene, and the other one is texel coordinates used to supply our own colour, i.e., the loaded texture in this time. Each texel has a unique address in the texture and we have manually mapped each modelling vertex to a specified texel.

Prepare the index buffer

We then reduce the size of buffer to be loaded into the display adapter by specifying an index buffer which has the same number of entry with the number of triangle in the mesh.

Each entry would contain three numbers and each number represents the corresponding index number in the vertex buffer. The index buffer can be thought as a list of triangle.

Conclusion

After the mapping, we would now have those 100 modelling vertices filled with Texel value and the index buffer filled with the list of triangle vertex.

3.2.4 Module "Facial Expression Modelling"

3.2.4.1 Introduction

Up to this module, everything we need to render the face object is already here. With respect to the facial expression analyzed result, we set the corresponding texture that the Microsoft Direct3D device will be used to render.

3.2.4.2 Basic concepts

Viewports and clipping

We render our object in the 3-D scene but what we finally see on the screen is only a projected 2-D rectangle. This is so called the viewports. To specify the viewports, we need a viewport's camera and define the x, y, z axis direction. Our system is using the right-handed Cartesian Coordinates, that is, the positive z-axis is pointing out of the screen.

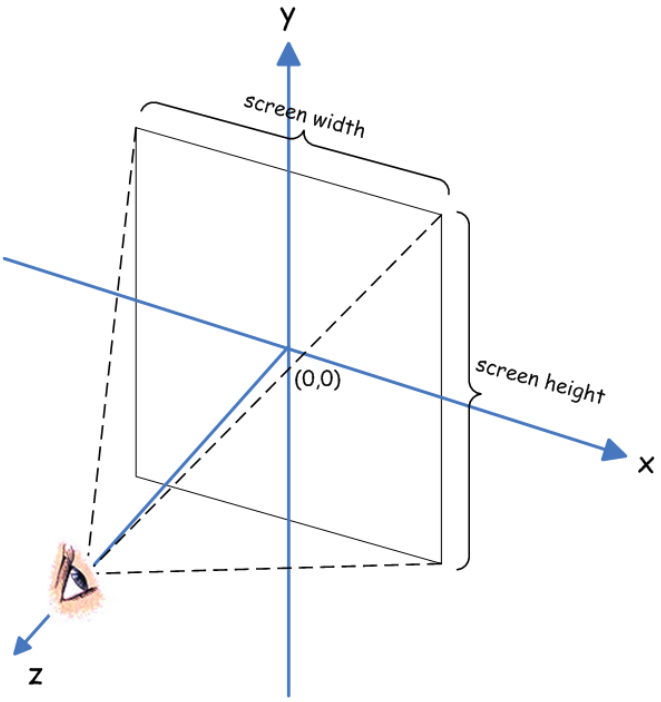

Fig.3.28 a simple viewport's example

Relative to the viewport's camera, we define a viewing frustum which is a 3-D volume in a scene. This volume affects how our object will be projected on the scene. This is done by specifying the fov (field of view) and by the distances of the front and back clipping planes.

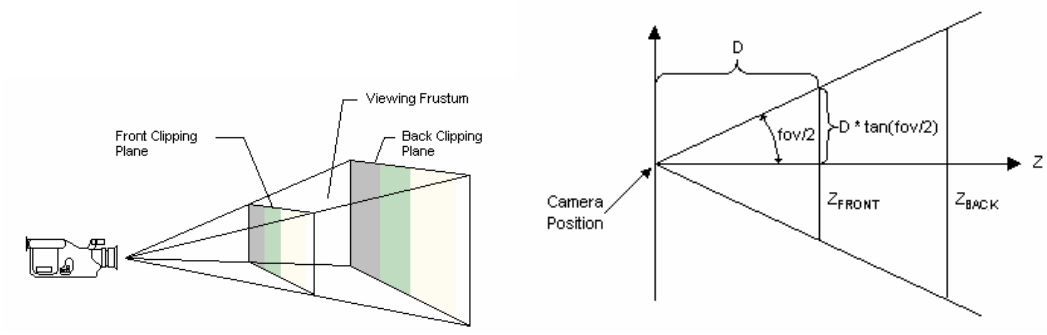

Fig.3.29 Define a viewing frustum using the fov and distance between the front and back clipping planes.

3.2.4.3 Implementation

To reflect the real-time movement of the face, we first need to update the object coordinates in the modelling vertex data from the face mesh coordinates. According to our own view port, the central position on the output screen is (0, 0). This configuration is needed in order to dynamically change the viewport dimension without shifting the origin. The face mesh coordinate, however, is referring the lower left corner as (0, 0).

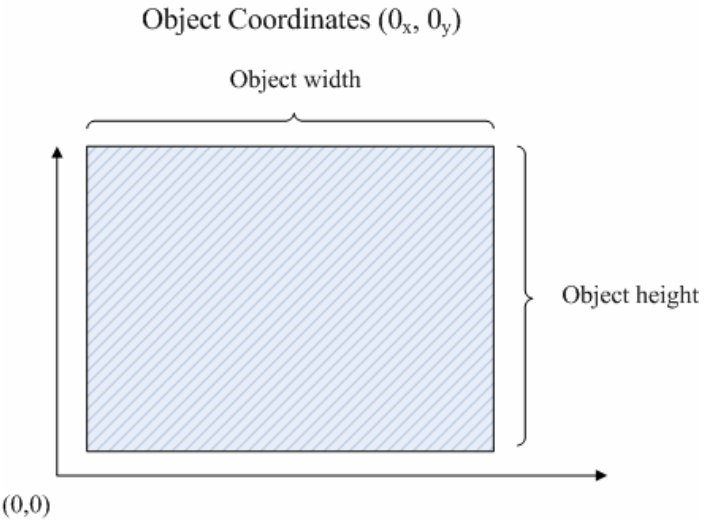

Fig.3.30 Object coordinates geometry

We need to shift the face mesh coordinates into the right position so that the whole rendered object will be seen in our screen. We can, then, fill up the vertex buffer which is our stream source with this complete modelling vertex data. The scene is finally ready to be rendered. We can then use the Direct3D device to render the index buffer.

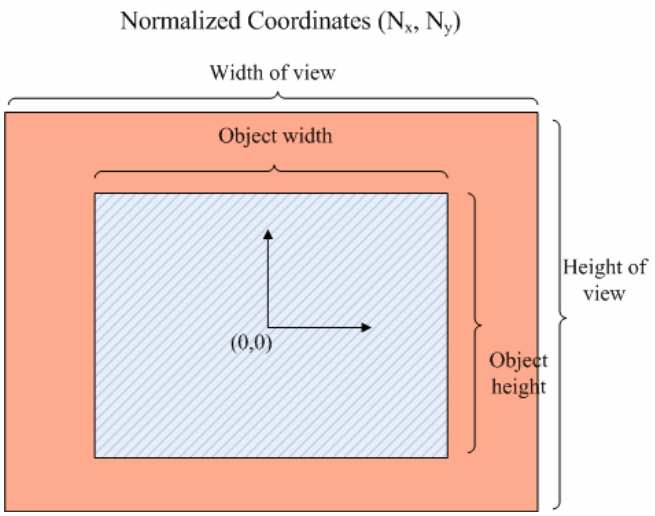

Fig.3.31 Normalized coordinates geometry.

Conclusion

After those shifting and rendering processes, we should able to see the output presented on two different screens. The ActiveMovie Window shows the modified video stream output. User can see himself/herself with adding the indications and check out whether the face mesh is fitted above their face. Once the mesh can detect the face, the character model will be shown on the Display View screen.

Fig.3.32 Normalized coordinates geometry.

Figure 3.32 An ActiveMovie Window showing the user (left) and the Display View showing the character model (right).

The facial expression image is now rendered under the modelling vertex data, therefore even in the same facial expression, the texture image and the rendered target would have different performance. Here is an example.

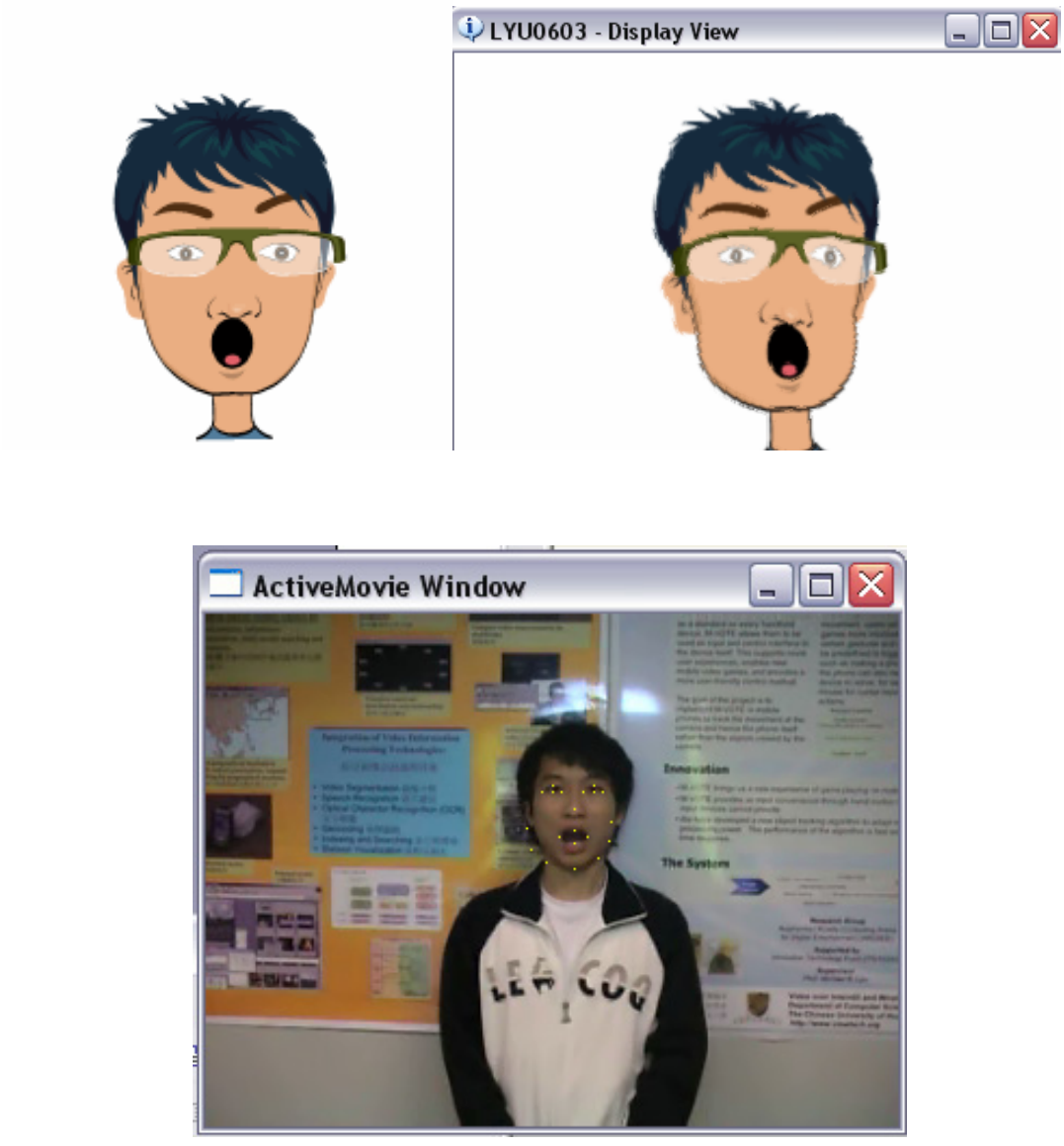

Fig.3.33 The original image file (top-left), the rendered output (top-right), and the current facial expression on the human face (bottom).

Chapter 4: Difficulties

This part stated the difficulties we faced in this project. We will also talk about how we eliminate or minimize the difficulties.

The most difficult problem in our project is to make the system becomes generic. It is known that implementation of this project will become easier if we use the same user and same environment for testing. Thus, to make the system more generic, we must collect many sample data and carry out analysis, which is time consuming.

The next difficulty is in finding out the pattern of colour or coordinates as to detect the facial analysis. To analyze the data, we have to collect many sample data under different environment. The light condition can affect the colour being retrieved. The camera quality can also affect the result because the colour recovery system is different.

Our future development is limited by the coordinates provided by the Face Coordinate filters. It is obviously that if there are more vertices in the face mesh, the accuracy of facial expressions can be improved. However, the more vertices will lead to a traditional problem, a sacrifice of computation cycle.

Chapter 5: Conclusion

We would like to conclude by summarizing what we have done and what we have learnt in semester one.

Firstly, we have defined our project motivation and objective, so we have a scope for learning reference materials at the earlier stage. We also have evaluated the possibility of the project. The difficulties should be controlled so that we can finish the project in a year.

Secondly, we have got familiarized with DirectShow and Direct3D APIs. They are quite difficult to learn at the starting stage because the lack of tutorial books. Also, we have studies on the Face Coordinate filter, which is an essential API for us to retrieve the coordinates of the face vertices.

Thirdly, we have designed and tried different algorithms and approaches to analyze the facial expressions. The statistics we have made is useful for mining the face information with respect to the facial expressions. However, it is quite time consuming.

Additionally, we have drawn a model to represent the user accordance to the facial expressions detected before.

Last but not least, we have provided different facial texture for user so that the output can be more interesting.

Chapter 6: Future Work

In the coming semester, we could improve our system by thinking some new algorithms to achieve the goal of facial expressions detection. The accuracy is important that we should put more afford on it.

Now we model the virtual character by using a 2-D texture. We want to improve it to a 3-D model so that the character is more interesting and attractive.

We also want to embed this system into existing net-meeting software. However, we have to face a problem that we have two output dialogs, which are original video with face outline, and the video modelled.

Chapter 7: Acknowledgement

First, we would like to express our heartfelt thank to our project supervisor, Professor Michael R. Lyu. Professor Lyu has given us many useful advices on the project. He is considerate of our workload and reminds us the importance of good scheduling. He also cares about our resources in doing this project so that we always have enough resources to work on the project.

Second, we would like to thank for our project manager, Mr. Edward Yau, who has given us many innovative ideas in our project.

In addition, we would also like to thank for research staff, Un Tsz Lung, who helped us on the problem of implementation on the project.

Last but not least, we must thank for the author of Face Coordinate filter, Zhu Jian Ke.

Chapter 8: Reference

[1] Jianke Zhu, Steven C.H. Hoi, Edward Yau and Michael R. Lyu, "Automatic 3D Face Modeling Using 2D Active Appearance Models". Proceedings of the 13th Pacific Conference on Computer Graphics and Applications, Macau, China, October 12-14, 2005

[2] Jianke Zhu, Steven C.H. Hoi, Edward Yau and Michael R. Lyu, "Real-Time Non-Rigid Shape Recovery via Active Appearance Models for Augmented Reality". Proceedings 9th European Conference on Computer Vision (ECCV2006), Graz, Austria, May 7 - 13, 2006.

[3] Yuwen Wu, Hong Liu, and Hongbin Zha, "Modeling facial expression space for recognition". Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on 2-6 Aug. 2005 Page(s):1968 – 1973.

[4] Hongcheng Wang; and Ahuja, N., "Facial expression decomposition". Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on 13-16 Oct. 2003 Page(s):958 - 965 vol.2.

[5] Yeasin M., Bullot B. and Sharma R., "Recognition of facial expressions and measurement of levels of interest from video". Multimedia, IEEE Transactions on Volume 8, Issue 3, June 2006 Page(s):500 – 508.

[6] Aleksic, P.S. and Katsaggelos, A.K., "Automatic facial expression recognition using facial animation parameters and multistream HMMs". Information Forensics and Security, IEEE Transactions on Volume 1, Issue 1, March 2006 Page(s):3 – 11.

[7] Chan-Su Lee and Elgammal, A., "Nonlinear Shape and Appearance Models for Facial Expression Analysis and Synthesis". Pattern Recognition, 2006. ICPR 2006. 18th International Conference on Volume 1, 20-24 Aug. 2006 Page(s):497 – 502.

[8] Chandrasiri, N.P., Naemura, T. and Harashima, H., "Interactive analysis and synthesis of facial expressions based on personal facial expression

space". Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on 17-19 May 2004 Page(s):105 – 110.

[9] Yabuki, N., Matsuda, Y., Fukui, Y. and Miki, S., "Region detection using color similarity". Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on Volume 4, 30 May-2 June 1999 Page(s):98 - 101 vol.4.

[10] Uysal, M. and Yarman-Vural, F.T., "A fast color quantization algorithm using a set of one dimensional color intervals". Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on Volume 1, 4-7 Oct. 1998 Page(s):191 - 195 vol.1.

[11] Chew Keong Tan and Ghanbari, M., "Using non-linear diffusion and motion information for video segmentation". Image Processing. 2002. Proceedings. 2002 International Conference on Volume 2, 22-25 Sept. 2002 Page(s): II-769 - II-772 vol.2.

[12] Mojsilovic, A. and Jianying Hu, "A method for color content matching of images". Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on Volume 2, 30 July-2 Aug. 2000 Page(s):649 - 652 vol.2.

[13] Jian Cheng, Drue, S., Hartmann, G. and Thiem, J., "Efficient detection and extraction of color objects from complex scenes". Pattern Recognition, 2000. Proceedings. 15th International Conference on Volume 1, 3-7 Sept. 2000 Page(s):668 - 671 vol.1.

[14] Unsang Park and Anil K. Jain, "3D Face Reconstruction from Stereo Video". Computer and Robot Vision, 2006. The 3rd Canadian Conference on 07-09 June 2006 Page(s):41 – 41.

[15] Yuankui Hu, Ying Zheng and Zengfu Wang, "Reconstruction of 3D face from a single 2D image for face recognition". Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on 15-16 Oct. 2005 Page(s):217 – 222.

[16] Ansari, A.-N. and Abdel-Mottaleb, M., “3D face modeling using two views and a generic face model with application to 3D face recognition”. Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003. 21-22 July 2003 Page(s):37 – 44.

[17] KyoungHo Choi and Jenq-Neng Hwang, “A real-time system for automatic creation of 3D face models from a video sequence”. Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on Volume 2, 2002 Page(s):2121 – 2124.

[18] Rozinaj, G. and Mistral, F.-L., “Facial features detection for 3D face modeling”. Industrial Technology, 2003 IEEE International Conference on Volume 2, 10-12 Dec. 2003 Page(s):951 - 954 Vol.2.

[19] Rudomin, I., Bojorquez, A. and Cuevas, H., “Statistical generation of 3D facial animation models”. Shape Modeling International, 2002. Proceedings 17-22 May 2002 Page(s):219 – 226.

[20] Sako, H. and Smith, A.V.W., “Real-time facial expression recognition based on features' positions and dimensions”. Pattern Recognition, 1996, Proceedings of the 13th International Conference on Volume 3, 25-29 Aug. 1996 Page(s):643 - 648 vol.3.

[21] Kanade, T., Cohn, J.F. and Yingli Tian, “Comprehensive database for facial expression analysis”. Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on 28-30 March 2000 Page(s):46 – 53.

[22] Youngsuk Shin, Shin Sooyong Lee, Chansup Chung and Yillbyung Lee, “Facial expression recognition based on two-dimensional structure of emotion”. Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on Volume 2, 21-25 Aug. 2000 Page(s):1372 - 1379 vol.2.

[23] Gokturk, S.B., Bouguet, J.-Y., Tomasi, C. and Girod, B., “Model-based face tracking for view-independent facial expression recognition”. Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on 20-21 May 2002 Page(s):272 – 278.

[24] Abboud, B. and Davoine, F., “Bilinear factorisation for facial expression analysis and synthesis”, Vision, Image and Signal Processing, IEE Proceedings-Volume 152, Issue 3, 3 June 2005 Page(s):327 – 333.

[25] Ma, L., Xiao, Y., Khorasani, K.and Ward, R.K., “A new facial expression recognition technique using 2D DCT and k-means algorithm”. Image Processing, 2004. ICIP '04. 2004 International Conference on Volume 2, 24-27 Oct. 2004 Page(s):1269 - 1272 Vol.2.

[26] Xiaoxu Zhou, Xiangsheng Huang, Bin Xu and Yangsheng Wang, “Real-time facial expression recognition based on boosted embedded hidden Markov model”. Image and Graphics, 2004. Proceedings. Third International Conference on 18-20 Dec. 2004 Page(s):290 – 293.

[27] Zhan Yong-zhao, Ye Jing-fu, Niu De-jiao and Cao Peng, “Facial expression recognition based on Gabor wavelet transformation and elastic templates matching”. Image and Graphics, 2004. Proceedings. Third International Conference on 18-20 Dec. 2004 Page(s):254 – 257.

[28] Sung Uk Jung, Do Hyoung Kim, Kwang Ho An and Myung Jin Chung, “Efficient rectangle feature extraction for real-time facial expression recognition based on AdaBoost”. Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on 2-6 Aug. 2005 Page(s):1941 – 1946.

[29] Jaewon Sung, Sangjae Lee and Daijin Kim, “A Real-Time Facial Expression Recognition using the STAAM”. Pattern Recognition, 2006. ICPR 2006. 18th International Conference on Volume 1, 20-24 Aug. 2006 Page(s):275 – 278.

[30] Abboud, B. and Davoine, F., “Appearance factorization based facial expression recognition and synthesis”. Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on Volume 4, 23-26 Aug. 2004 Page(s):163 - 166 Vol.4.