LYU1201

香港中文大學
The Chinese University of Hong Kong

MicroXP

Department of Computer

Science & Engineering

3D DANCE HEAD USING KINECT AND 3D PROJECTOR

Student: Chan Wai Yeung (1155005589)

Lai Tai Shing (1155005604)

Supervised by Prof. LYU Rung Tsong Michael

Abstract

We are going to design and implement an algorithm to achieve Dance Head effect using Microsoft Kinect Camera. Since there is no similar product all over the word at this moment, it is hoped that our FYP product could provide entertainment for the public and people would have fun on playing this. In this project, we have firstly studied the Kinect SDK. However, we decide to use OpenNI and OpenCV with Kinect to achieve the goal of our project. We have tried many methods for extracting the heads. We have done several experiments in order to find out the best ways to perform Dance Heads. Lastly, we use the surface normal combined with depth value to extract the head and superimpose to the Dance Head video.

Chapter 1 : Introduction

Motivation
Background
Objective
Runtime environment

Chapter 2 : Kinect

Introduction
Specification
Kinect SDK

Chapter 3 : Open Source Library

OpenNI introduction
Overview of OpenNI
OpenCV introduction
Overview of OpenCV

Chapter 4 : Face Detection

Face Detection in Kinect SDK
Face Detection in OpenCV
Comparison between OpenCV and Kinect SDK

Chapter 5 : Design and Implementation

Milestone 1
Milestone 2
Milestone 3

Chapter 6 : Conclusion and further work

Conclusion

Difficulties encountered

Current limitation

Further work

Chapter 7 : Acknowledgement

Chapter 8 : Reference

Chapter 1 : Introduction

Motivation

While surfing the Internet, we have found that there are many video called “Dance Heads”. Those videos are extremely funny and interesting. Dance Heads is an innovative entertainment for all ages of people. Many people were addicted to this entertainment when Dance Heads was invented.

Figure 1.1 Dance Heads

The principle of dance heads is not difficult to work out. The participant’s heads are superimposed to the bodies of some professional and beautiful dancers with theanimated backgrounds. At that moment, the participants just need to sing or swing their heads. Watching the video, you would find that the clothes of the dancers could be changed according to the background music. The video would show that the participants are dancing even the players do not know how to dance. However, the participants are just stand and swing their head only. They actually do not need to dance at that time!

However, there are some limitations to the participants. First of all, while capturing their heads, they need to stand up and wear colored clothes. The colored clothes should be the same color as the background and it should cover the participant’s neck.

Figure 1.2 People playing Dance Heads

When we watch the video, what comes to our mind is how to implement the Dance Heads without the above limitations. We also want to do something funny in the project. We have noticed that XBOX are now gaining popular all over the world. We are then inspired by the XBOX 360 Kinect. We come up the idea of using Kinect to capture the head instead of using the camera. We think this project would be interesting and funny so that we decide to do the Dance Heads using Kinect.

We hope that we would do the dancing head effect using Kinect without the limitations mentioned above. Meanwhile, we hope people would enjoy playing this Dance Heads application.

Figure 1.3 Kinect

Background

Searching “Dance Head” inYoutube, it is not difficult to find out some video about “Dance Heads”. What is “Dance Heads”? “Dance Heads” is a video that participants’ head is extracted and then superimposed to a dancers with an animated background and music. Dance Heads was found in 2003. It was established by a company which is called Dance Head Inc. The company then applied for the trade mark for the Dance Heads. Apart from Dance Heads, the company have also launched other applications called Sport Heads, Super Heads and DH Recording. The principle of these applications is quite similar. Heads are extracted and superimposed into different bodies.

Figure 1.4 Logo of the company

In the past, these video became extremely popular. Lots of companies, including Coca-Cola, Disney to name but a few, made their advertisement using the Dance Heads effect. Without a doubt, Dance Heads was a great success. It also provided a funny entertainment for the public. Dance heads had also used in many fairs, festivals, social events and amusement parks etc.

How to make Dance Heads video? The participants have to wear clothes with the same colour as the background. The camera captures the heads of them and removes the other part of the bodies. The participants then choose the favourite music and stand in front of a rail. A Dance heads Video is finally made.

Figure 1.5 Dance Heads

According to the Dance Heads specification, Dance Heads requires 10’ x 15’ (Width x Deep) space. It could set up outside but a shaded area or a tent is needed. The maximum numbers of participants are 3. However, this interactive entertainment is quite expensive.

Objective

In our final year project, we are going to study the KINECT, design and implement an algorithm to achieve Dance Heads effect using KINECT camera.

We have to find the best way to perform Dance Head effect using the Kinect camera. We should extract the participants' heads using the Kinect and render the Dance Head effect.

There are several objectives in our final year project.

- Study the Kinect SDK or other software used in our project.
- Design and implement an algorithm to extract the "Head" part using KINECTcamera.
- Rendering the Dance Head effect into 3D Video or viewing through 3D projector.

Runtime environment

For our development, we have decided Ubuntu as the platform. Since we have found that there are some limitations using the Kinect SDK, we have to use OpenNI with OpenCV instead of Kinect SDK. But we still to use the Kinect camera as the major component to capture the body and the face of participants.

Figure 1.6 Ubuntu

We have to use C++ as programming language, and develop the program in Ubuntu. Several graphs could be made finally. We mainly focus on the depth graph, as we are interested in the depth value of the participants' faces. The most important thing is how to extract the head well to perform Dance Head effect.

Ubuntu is a Linux based operating system. The latest version is 12.10.Many software packages are component of Ubuntu. Therefore, we could integrate different open-source libraries into our FYP under the Ubuntu system. We have used OpenNI and OpenCV and these two open-source software are also be compatible in Ubuntu.

Chapter 2 :Kinect

Introduction

Kinect was first launched at 2010. In 2012, Microsoft had launched Kinect for Windows operating system. The name of Kinect comes from the word “kinetic” and “connection”. Kinect Sensor is very popular all over the world and becomes the fastest selling consumer electronicsdevice at 2011.

Kinect is developed by Microsoft, and mainly used for the equipment of Xbox 360 and Windows. Kinect is a motion sensing input device. By using Kinect, the playersdo not need to use the remote control anymore. They could interact and control the Xbox 360 by their gestures and movement of body.

At June 2011, Microsoft released Kinect software development kit for Windows 7. Developers could study Kinect SDK and develop program in C++, C# or Visual Basic.

Figure 2.1 Kinect

Nowadays, Xbox 360 with Kinect is one of the most popular electronic game devices all over the world. During playing the video game, the players could also do exercise. The faith that playing video game is not healthy would be exploded.

Specification

Kinect device have RGB camera, 3D depth sensor and multi-array microphone with a motorized pivot at the base. Using the Kinect, full body 3D motion could be captured. Apart from this, it could also recognize the face and voice.

Figure 2.2 Kinect

RGB camera

Kinect camera is an RGB camera with resolution 640*480 in 32-bit colour. This camera works at a 30Hz frame rate and it could recognise face and movement of the body. The camera could detect at most 6 players, however, only two active players are detected for the joint recognition. For each player, twenty joints of the body could be tracked.
The working range of Kinect sensor is from 1.2 to 3.5 meters. The horizontal field of view is 57° wide. Therefore, the maximum range scanned is 3.8 meters wide. Meanwhile, the vertical field of view is 43°. At the same time, the vertical pivot allows the sensor to move up and down at most 27° in either direction.

3D Depth sensor

Kinect could see in 3D in lighting conditions. The sensor range would be adjusted atomically by software. The depth sensor is made up of an infra-red projector and a monochrome CMOS sensor. The depth sensor has a 320*400 resolution with 16-bit sensitivity. Similarly to the RGB camera, the video captured is in 30Hz frame rate.

Multi-array microphone

The microphone could suppress the noise, localize the acoustic source and recognize the voice. It consists of a 4-array microphone capsules and processes sound in 16-bit audio at a rate of 16 KHz.

Kinect Port

The Kinect port is a Microsoft proprietary connector. The connector supply power and data communication for the Kinect sensor.

Kinect Specification	
Sensor	color and depth-sensing lenses
Data Rate - RGB Camera	640×480 pixels / 32 bit color @ 30 frames/sec
Data Rate - Depth Sensor:	320×240 pixels / 16 bit grey scale @ 30 frames/sec
Sensor range	1.2m – 3.5m
Field of View-Horizontal	57 degrees (1.3m - 3.8m)
Field of View - Vertical	43 degrees
Tilt motor for sensor adjustment in the vertical plane	Adjustment ± 27 degrees
Skeletal Tracking System	tracks up to 6 people, including 2 active playersTracks 20 joints per active player, 33ms response time
Microphone array	4 mic cells
Data Rate	16-bit audio @ 16 kHz
Audio Systems	Live party chat and in-game voice chat
Echo cancellation system	Enhancesvoice input Speech recognition in multiple voice environments
Power Supply	Supplied by Kinect port or AC power adaptor
Data Connection	Supplied by Kinect port or USB port (with AC adaptor)

Kinect SDK

Software development kit (SDK) of Kinect for Windows sensor is launched by Microsoft. Using the SDK, developer could use C++/C# or Visual Basic to develop application by their own.

Several versions of Kinect for Windows SDK had been released.The SDK includes drivers for using the Kinect for Windows sensor on a computer which runs Windows 8, Windows 7, or Windows Embedded Standard 7. Apart from this, it includes APIs and device interfaces. The latest version of the SDK is 1.6 and is released October 2012. On the other hand, the toolkit includes source code samples, for example, Kinect Studio, Face Tracking SDK etc. It is useful for the developer since it would help developing applications by using the Kinect for Windows SDK. The latest version of toolkit is 1.6 and is updated October 2012.

The face tracking SDK is released in 1.5 SDK and developer toolkit special features. The face tracking SDK is useful for our final year project.

1.6 of the SDK and the Developer Toolkit
Windows 8 Support
Visual Studio 2012 Support
Accelerometer Data APIs
Extended Depth Data
Color Camera Setting APIs
More Control over Decoding
New Coordinate Space Conversion APIs
German Language Pack for Speech Recognition
Infrared Emitter Control API
Introducing New Samples!
Kinect Studio 1.6.0
The Infrared Stream Is Exposed in the API
Support for Virtual Machines

Hardware Requirements

- Windows 7, Windows 8, Windows Embedded Standard 7
- 32 bit (x86) or 64 bit (x64) processor
- Dual-core 2.66-GHz or faster processor
- Dedicated USB 2.0 bus
- 2 GB RAM
- A Kinect for Windows sensor
- Graphics card that supports DirectX 9.0c

Software requirements

- Visual Studio 2010 or Visual Studio 2012
- .NET Framework 4 or .NET Framework 4.5

Supported Operating Systems

- Windows 7
- Windows 8
- Windows Embedded Standard 7

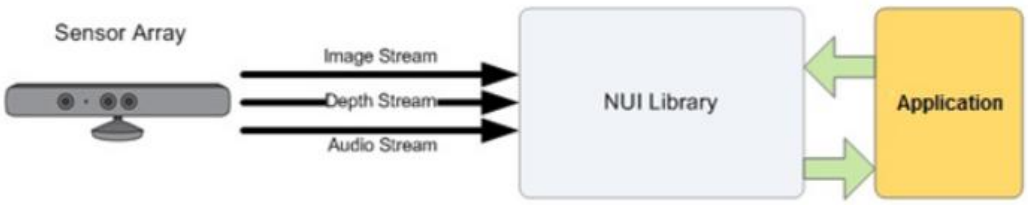


Figure 2.3 Kinect for Windows structure

Hardware and Software Interaction with an Application

The SDK gives software library to help developers use the form of Kinect-based input like image, depth and audio. The Kinect and the software library interact with the application.

The Natural User Interface (NUI) is the core of the Kinect for Windows API. Through the NUI, developer would get the sensor data such as audio, depth and image stream in application.

There are two methods for getting the image frames, polling and event models. The polling model is used to read data frames. The event model supports the ability to use those data streams with more accuracy and flexibility.

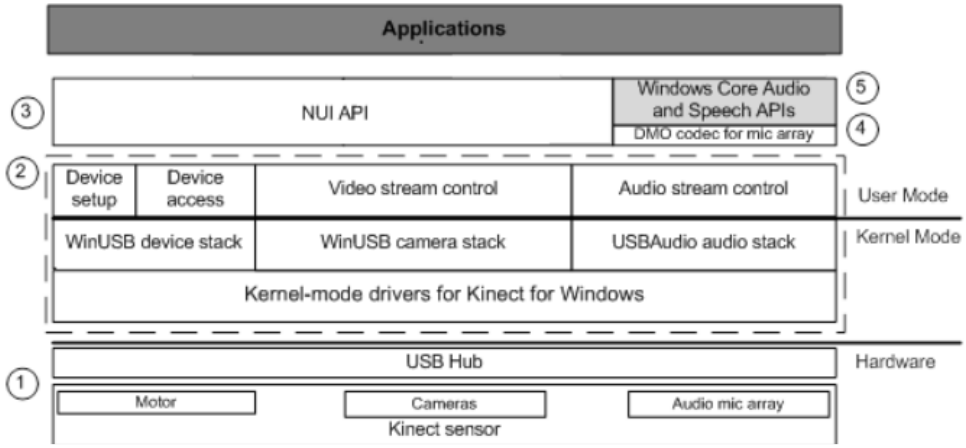


Figure 2.4 SDK Architecture

Chapter 3 : Open Source Library OpenNI

Introduction

The OpenNI organization is a non-profit organization. The organization is designated to promote the compatibility and interoperability of Natural Interaction (NI) devices, applications and middleware.

The organization was formed in November 2010. There are many members of this organization. PrimeSense, which is the company behind the technology used in the Kinect, is one of the members.

Natural Interaction (NI) is a concept that human-device interaction based on human senses. For example, hand gesture and body movement are one type of human-device of NI. They could use their hand gesture and body movement to control something in game or other devices.

OpenNI is an open source and a cross platform framework which provides the interface for physical devices such as sensors and software components. The framework defines application programming interface (API) for developing application using natural interaction.

OpenNI support several platforms such as Windows XP and later, for 32-bit only, Linux Ubuntu 10.10 and later, for x86 and MacOS and some of the embedded system.

Overview

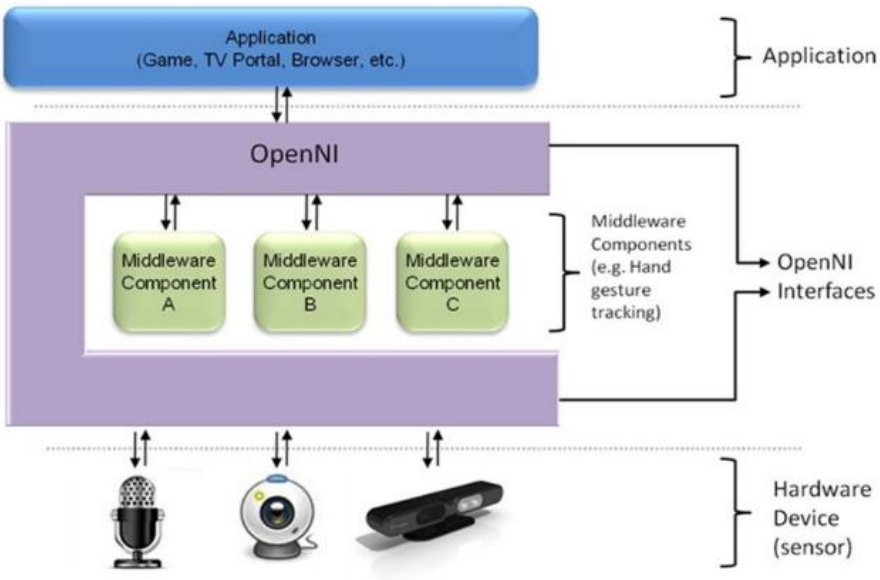


Figure 3.1 Three-layered view of the OpenNI Concept

From the graph, the top represents the software implements natural interaction (NI) applications on OpenNI. The middle represents OpenNI which provides communication interfaces. The interfaces interact with the sensors and middleware components and analyze the data from the sensor. The bottom represents the hardware devices.

OpenCV

Introduction

OpenCV (**Open** Source **C**omputer **V**ision) is a library of programming functions for real time computer vision. It is developed by Intel and supported by Willow Garage and Itseez. It is open source software and used around the world. Similar to OpenNI, the OpenCV library could be used in different platforms. It is mainly used for the image processing.

OpenCV was first launched in 1999, it is intend to improve the CPU-intensive applications such as 3D displays wall and real-time ray tracing. And then, the first alpha version was released in 2000. Another five beta versions were released in 2001 and 2005. In 2006, the version 1.0 was finally released. Two years later, 1.1 version of OpenCVwas released.

In 2009, OpenCV was released. There are some major changes. The changes include C ++ interface, new functions and have a better implementation for multi-core system. Nowadays, for every six months, a new version would be released. The latest one is 2.4.3 released on November 2012.

OpenCV run on several platforms. For example, it mainly runs on Android, MacOS, Windows and Linux. OpenCV is written in C++ and its primary interface is C++. However, it remains C interface. Now, there are full interface in Java, Phython and Matlab. Wrappers in other languages such as C#, have also been developed.

Overview

Many applications are provided by OpenCV. The followings are some examples2D and 3D feature toolkits, face recognition, object identification, motion tracking, gesture recognition and Human-computer interaction etc.

We have to include several libraries when using OpenCV. For the library cv, it is used for image processing, object or face detection and camera calibration etc. Simple GUI, image/video I/O is created when the HighGUI library is included.MLL is something about the classifiers and clustering algorithms. CXCORE is used to perform simple operation, matrix algebra and math functions to name but a few. IPP is the optimized code for CPUs.

Chapter 4 :Face Detection

To develop Dance Heads application, it is important for us to extract the head part. After that, the head is superimposed to the Dance Heads video. We have found that there are two main methods to detect the face. The first one is Kinect SDK and the other one is provided by OpenCV. We would talk about these two methods on the following.

Face Detection in Kinect SDK

Since Kinect SDK V1.5, Microsoft added a face tracking component. Our most interest is in this application since we would like to use the face to achieve our final year project, Dance Heads.

For the face tracking, there are many features. To start with, the face tracking includes tracking the face position, orientation and the face features in real time. Moreover, a 3D mesh of face, including the mouth and eyebrow is animated. Last but not least, it would track several faces at the same time.

Face tracking

Face tracking SDK is used for the developer to develop applicationabout face detection. The face tracking SDK engine analyzes the input from the Kinect camera. It then estimates the head pose and facial expressions. Finally, the information would drive NUI or used in other application.

Technical Specifications

Input image

Face Tracking SDK allows Kinect depth and color images as input. The tracking quality would be affected by the image quality of input frames. For example the darker condition would be harder for tracking the face. Surprisingly, larger faces are easily tracked than smaller faces.

Coordinate system

Face tracking SDK use Kinect coordinate system to output 3D tracking results. The camera's optical center is the origin. Y axis is pointing up while Z axis is pointing towards the user. The measurement units are meters for translation while the measurement units are degrees for rotation angles.Finally, the computed 3D mesh has coordinates which place it over the player's face.

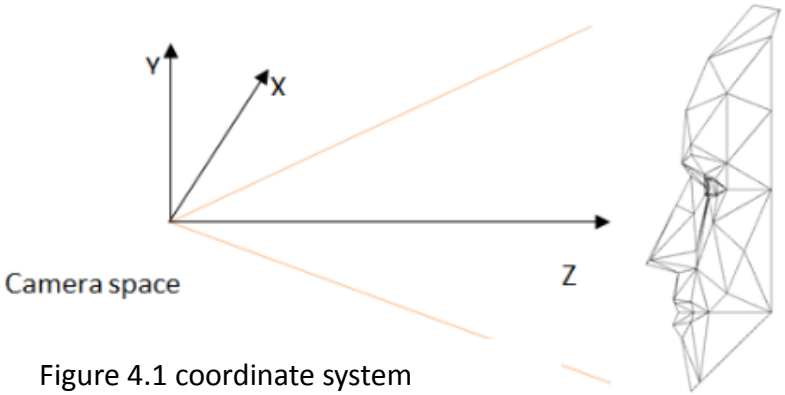


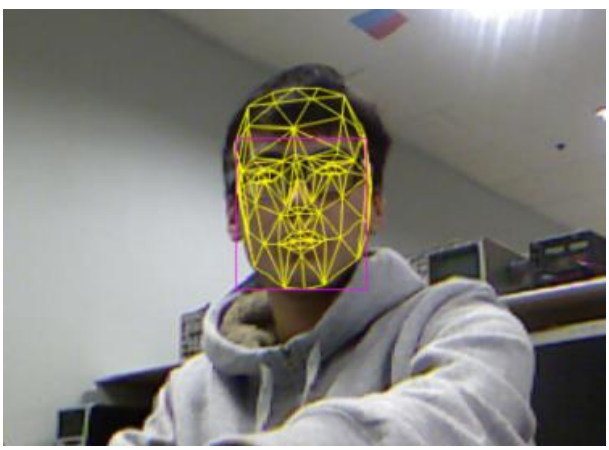
Figure 4.1 coordinate system

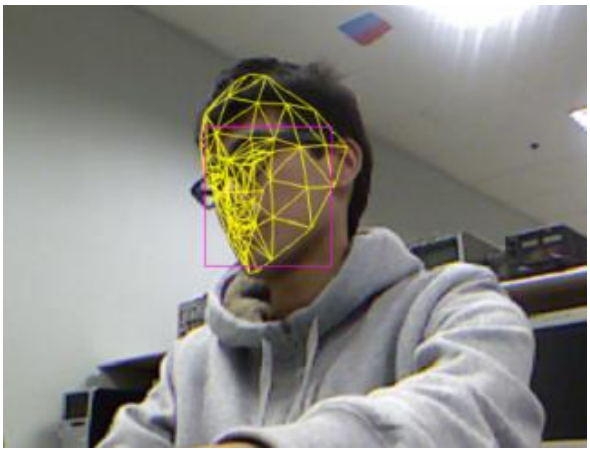
Face Tracking Outputs

The output of the Face Tracking engine contains the following information about a tracked user:

- Tracking status
- 2D points
- 3D head pose
- Animations Units

We are mainly interested in 2D points for tracking the face in our final year project. The Face Tracking SDK tracks the 87 2D points on the face while 13 points that aren't shown on the face. The 13 points include the center of eye, the corner of mouth and the center of noise. All of the tracked points are returned in an array. They are defined in the coordinate space of the RGB image with 640* 480 resolution.





Face detection in OpenCV

There are many researches talking about how to detect face correctly. OnepCV has a method to detect face using AdaBoost Learning with Haar-Like features provided by Viola & Jones.

How to detect the face? We have to first give a lot of sample which is related to face. Using AdaBoost learning algorithm, some representative sample is chosen. These samples are called Haar-Like Features.Before using the Haar-Like features to identify objects, we have to create some specific rectangular blocks data base, for example, face features classifiers. Harr classifiers includerectangular block with 2 -3 black areas.

A classifier trained with few hundred sample of a particular object such as face.

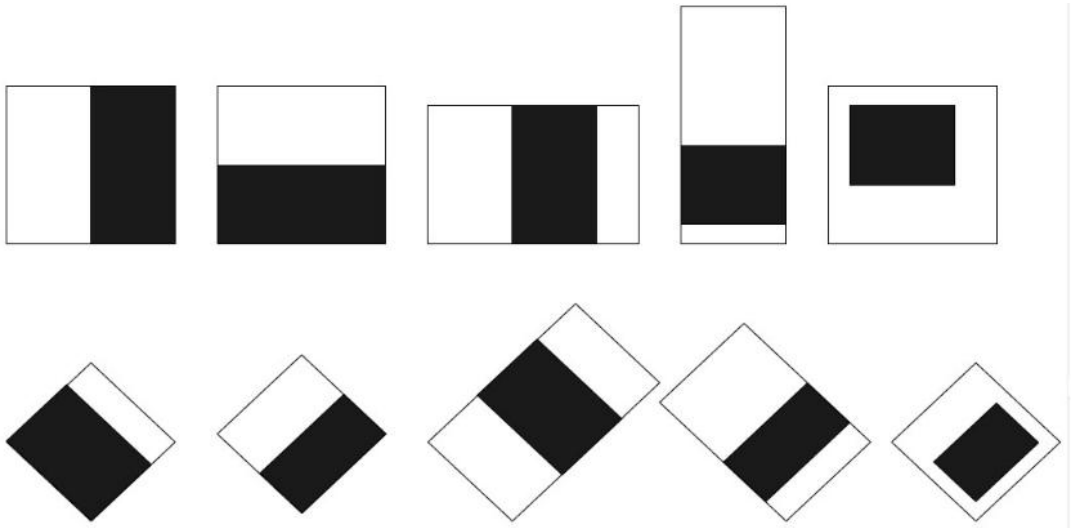


Figure 4.4 Rectangular block used to create classifier

After a classifier is trained, it can be applied to input image. The classifier outputs 1 if the region is likely to be a face while the classifier output 0 if it is not a face.
The word “cascade” means that the resultant classifier consists of several classifiers stages that are used subsequently to the interested region until at the stage that the case is rejected or all the stages are passed successfully.
The most important is that we have to use the haarcascade_frontalface_alt.xml. That means we use the classifier of the front face.Using some function in OpenCV, a rectangle or a circle would be drawn if face is detected in image or video.

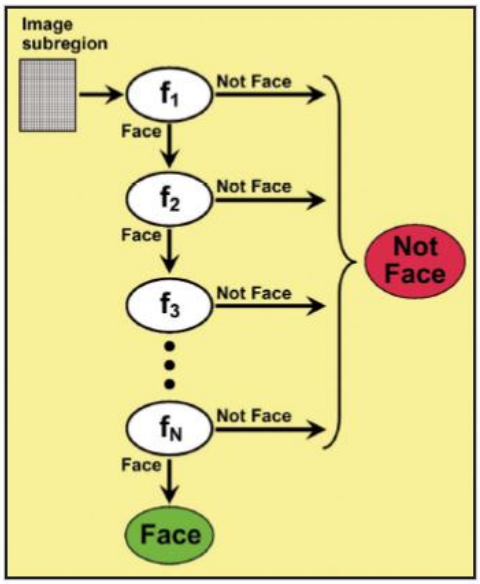


Figure 4.5 Cascade classifier stage to detect face

Figure 4.6 Detected face

Comparison between Kinect SDK and OpenCV

Kinect SDK	OpenCV
Work on Windows 7	Work on many platforms such as Linux, MacOS, Andriod
Based on 3D data	Based on 2D image
Programming language: C# / C++ / Visual Basic	Programming language: C / C++ / python/ Matlab
Based on face tracking SDK and COM interfaces provided by Microsoft	Based on Cascade Haar-Like features
Result image: 3D image with mesh	Result image: 2D image
Multi-face would be tracked	Multi-face would be tracked
Skeleton tracking	Skeleton tracking

From the above table, we would find that there is a great difference between Kinect for Windows and OpenCV. However, they would perform the same operation, for example, face detecting using different method. These two face detections are quite useful in our application.

We have to understand the principle behind Kinect for windows and OpenCV so that we would use it better. After understanding it thoroughly, we have to choose a better way for our final year project, Dance Heads.

Chapter5:Design and Implementation

To do our final year project Dance Heads using Kinect, we have to understand the background of Dance Heads. After surfing the Internet, we have found that a company treat Dance Heads as business for people. It is an entertainment for the public and also an interesting advertisement for company promotion.

After studying the Kinect SDK, we found that face tracking SDK is a powerful tool which could detect the face using Kinect. Without a doubt, it would be useful in our project, Dance Heads. However, there are lot of limitation using Kinect for windows. The developing environment must be Windows 7 or above.

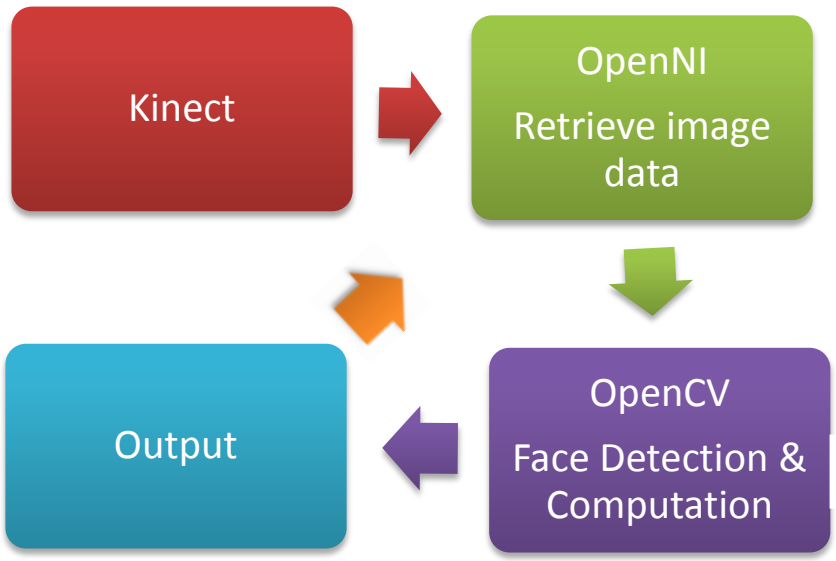


Figure 5.1 Work flow

At the very first beginning, we have done a research on Dance Heads. It is found that there is no Dance Heads which is using Kinect. To start with, as mentioned before, we have to compare between Kinect SDK and OpenCV and consider the limitation of both of them. Finally, we choose using OpenNI with OpenCV rather than Kinect SDK.

After receiving the data, we use OpenCV for our computation. The most important part is the face detection in OpenCV. Using the cascade Harr features classifier, we would detect the face. Finally, we have to consider how to cut the head well from the data that we have.

In this semester, we have done several things. To summarize, we would divide our work in to 3 millstones. On the following, we would discuss those 3 millstones one by one in details.

Millstone 1

We use C++ as our programming language. OpenNI is a great platform that we could get the data from Kinect.

We have to include the useful header file. The main programming language in OpenNI is C, but we have to include the <XnCppWrapper.h>header file. We would use the format of C++ to use OpenNI. Other than that, we also have to include the other useful header files.

Fristly, we have to familiar with the depth and colour map. We try to handle the depth data with RGB image data. We find a certain range at depth data. At that specific range, the object would be shown. For the other depth, the background is set to black.

Next, we have to initial the context in order to use the releated modules and other data in OpnenNI.After initializing the context, it is important to create the production node. We can use depth generator in OpenNi to get the depth data. Hence, we would use tha data to achieve wehat we want.And we use a loop to

keep tracking the newest data. In addtion, there should be error detection in the main program. Ther program should exit if there is any error.

To conclude, we would develop a program that using Kinect and OpneNI. We would use the depth value collected by Kinect to achieve some specific purpose. By doing so, we would get familiar with how to programming in Kinect and OpenNI. The most important thing is that we would practise and get familiar using the depth data of the object.

Milestone 2

After playing with the depth value in OpenNI, we have move to the next step. The objective of our final year project is to develop a application which is about Dance Heads. Therefore, we are interested in the part of head.In this milestone, we have a try on the face detection in OpenCV. Apart from this, we would integrate the face detection into milstone so that we are more familiar with depth value and the face detection.

We then have to use face detection provided by OpenCV in our OpenNI program. Since OpenCV provides some drawing and computation functions. It is very power tool in our program without any doubt. It is important that we have to include many OpenCV header files. We have to include the OpenCV and OpenCV2 libraries. The most common libraries in OpenCV are cv.h, highgui.h. the cv.h includes image processing and vision algorithms while the highgui.h includes, GUI, image and Video I/O.

After including the useful libraries, in milestone 2, we have to integrate the face detection in our programming. To perform the face dectection, we have to add a function called detectAndDraw and definte the variable and configue the path of xml files.

On the image map, a circule with a cetain radius is drawn on the face of the image.

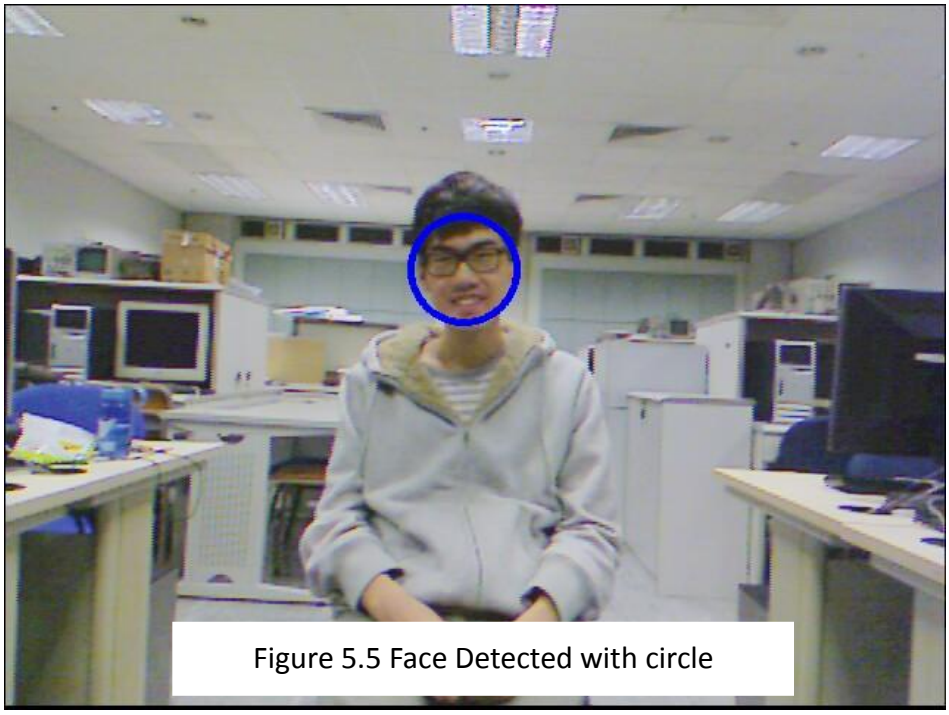


Figure 5.5 Face Detected with circle

At the same time, we combine it with what we have done in milestone 1. Now, face is detected on the image graph. For the depth graph, we use the depth value of the face. We then set the range of the depth between

depth of head+20cm <depth of head< depth of head-5cm

The image between that certain depth value would be showed. However, besides the face, the other things which are the same depth value of the face would also be shown.

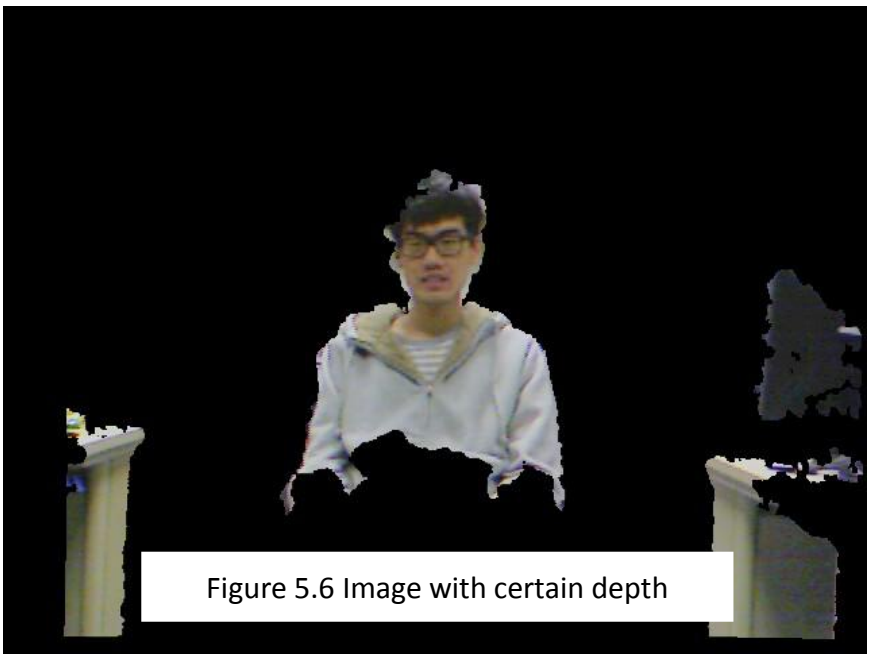


Figure 5.6 Image with certain depth

Figure 5.7 Mask of the image

Connected component

Some of the other objects at that specified depth would also be shown. However, our interest region is only the head part. We have to exclude the other objects within the range of the head's depth. We have to use the concept of the connected component. The objects which are not connected to the face would be exclude from the result image. Hence, the other objects with the same depth value of the head would not be shown finally. Connected component is a useful method to improve our output image. We would exclude the other things in our result so that only the player is shown.

```
//set point(x,y) is connected if it is masked in src and not yet set
//and check the points surrounding point(x,y) recursively

void toConnected(Mat& src, Mat& dst, int x, int y) {
    if (src.at<uchar > (y, x) == 255 && dst.at<uchar > (y, x) != 255) {
        dst.at<uchar > (y, x) = 255;
        for (int i = -1; i <= 1; i++)
            for (int j = -1; j <= 1; j++)
                if (i != 0 || j != 0)
                    if (x + i >= 0 && y + j >= 0 && x + i < src.cols && y + j < src.rows)
                        if (src.at<uchar > (y + j, x + i) == 255)
                            toConnected(src, dst, x + i, y + j);
```

```
//calculate the connected part of the image
Mat maskWithConnect = Mat(cDepthImg.rows, cDepthImg.cols, CV_8UC1, Scalar(0));
toConnected(maskWithBoundingWindow, maskWithConnect, centerX, centerY);
//namedWindow("maskWithConnect");
//imshow("maskWithConnect", maskWithConnect);
```

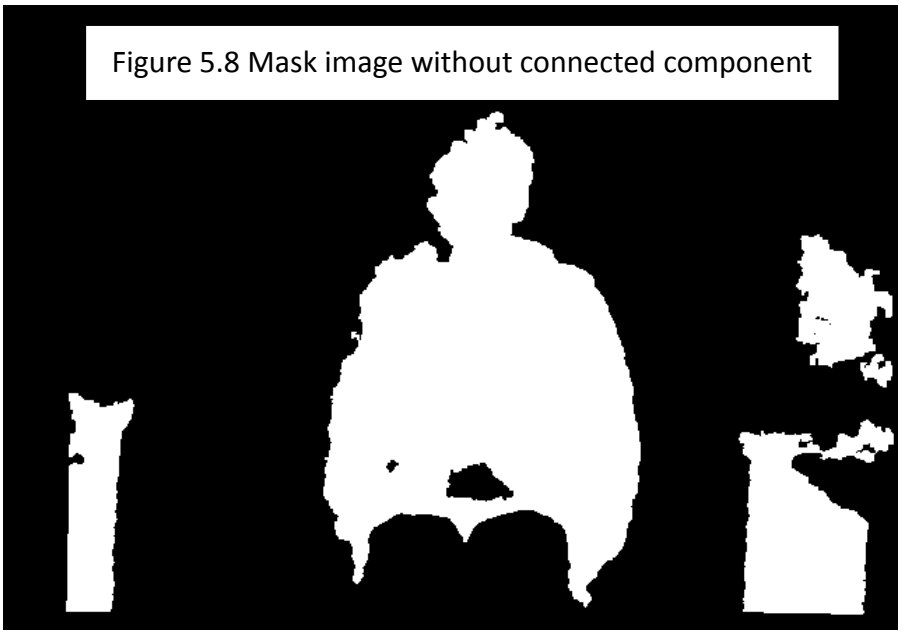


Figure 5.8 Mask image without connected component

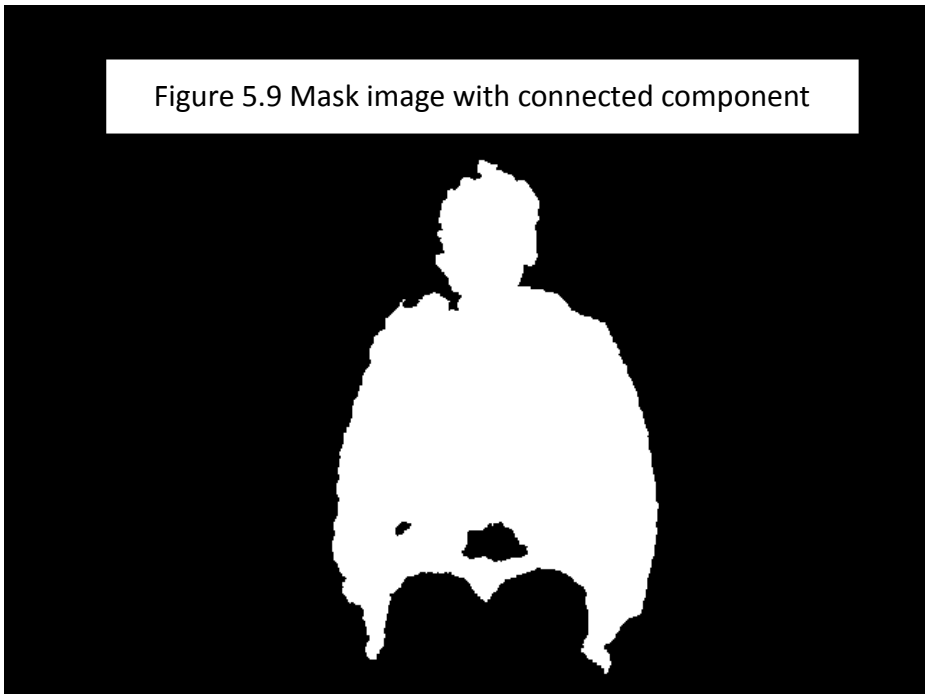


Figure 5.9 Mask image with connected component

From the above two graph, if the concept of connected component is applied, only the object connected to face would be shown. The other thngs would not be shown at all.

Filled holes in graph

Another problem is that there are many black holes in the outcome results. The black holes would certainly affect the image. Doubtless, these would be an imperfection to our application. The black holes are come from the noise of the infra-red of the Kinect sensor. The infra-red of the Kinect is easily interfere with the background noise so that there are many black holes in the image captured by Kincet.

```
//fill the hole caused by the imperfect result of KINECT
Mat maskWithFilledHoles = maskWithConnect.clone();
Mat holes = maskWithConnect.clone();
floodFill(holes, Point(0, 0), Scalar(255));
for (int i = 0; i < maskWithConnect.rows; i++) {
    for (int j = 0; j < maskWithConnect.cols; j++) {
        if (holes.at<uchar > (i, j) == 0)
            maskWithFilledHoles.at<uchar > (i, j) = 255;
    }
}
```

Figure 5.10 Image without holes

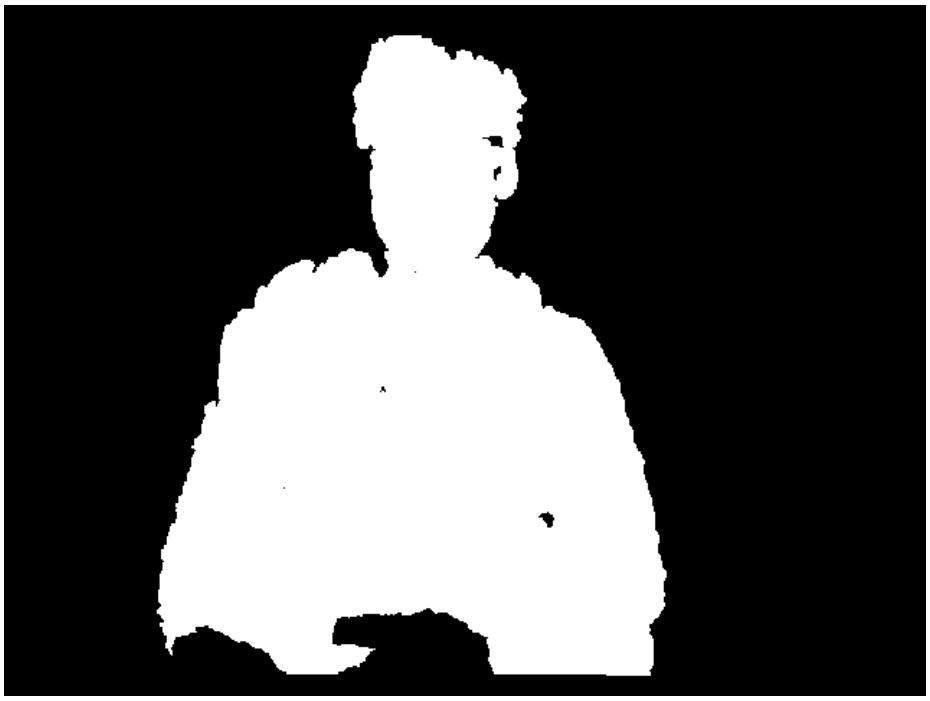


Figure 5.11 Image with holes

Figure 5.12 Holes filled image

Milestone 3

In the first two milestones, we get a concept of the depth value and face detection. In this milestone, we have to extract perfectly of the head of the player.

Figure 5.13 Ways of extracting face

We have tried several methods for extracting the face.

1. Chop the face by the raduis of circle
2. Cann Edge detection
3. Surface normal
4. Surface normal with depth value

In our main program in milestone 3, we used several ways to extract the face and try to find the best one which is suitable for making the Dance Heads application. We have to create several functions and graph in order to show the result.

Chop the face by the raduis of circle

In this milestone, after the classifier identify the face, a circle is draw on the image graph. The radius is set to be a certain ratio to the radius face detection. It is obvious that the image inside of the circle is our main interest. Therefore, in the depth map, we just show that region. The result is including the face and the lower part of the neck.

Using the code as mentioned before, face is detected. The face detection function is provided by OpenCV. It uses the cascade classifier to detect the face of the player.

```
void faceDetect(Mat& src) {
    Mat img;
    src.copyTo(img);

    //load cascade
    CascadeClassifier cascade;
    cascade.load("./haarcascade_frontalface_alt.xml");

    int i = 0;
    vector<Rect> faces;
    const static Scalar color = CV_RGB(0, 0, 255);

    //resize the image to speed up the detection
    Mat gray, smallImg(cvRound(img.rows / SCALE), cvRound(img.cols / SCALE), CV_8UC1);
    cvtColor(img, gray, CV_BGR2GRAY);
    resize(gray, smallImg, smallImg.size(), 0, 0, INTER_LINEAR);

    //normalize the histogram of the image
    equalizeHist(smallImg, smallImg);

    cascade.detectMultiScale(smallImg, faces, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));
    for (vector<Rect>::const_iterator r = faces.begin(); r != faces.end(); /*r++,*/ i++) {
        Point center;
        int radius;
        center.x = cvRound((r->x + r->width * 0.5) * SCALE);
        center.y = cvRound((r->y + r->height * 0.5) * SCALE);
        radius = cvRound((r->width + r->height)*0.25 * SCALE);
        circle(img, center, radius, color, 3, 8, 0);

        //save the result to global
        centerX = center.x;
        centerY = center.y;
        _radius = radius;

        //constrain only detect one face
        r = faces.end();
    }

    //namedWindow("result");
    //imshow("result", img);
}
```

Based on the centre of the face, the face is extracted by a certain value of radius. However, the lower part of the neck including the clothe of the player is extracted.

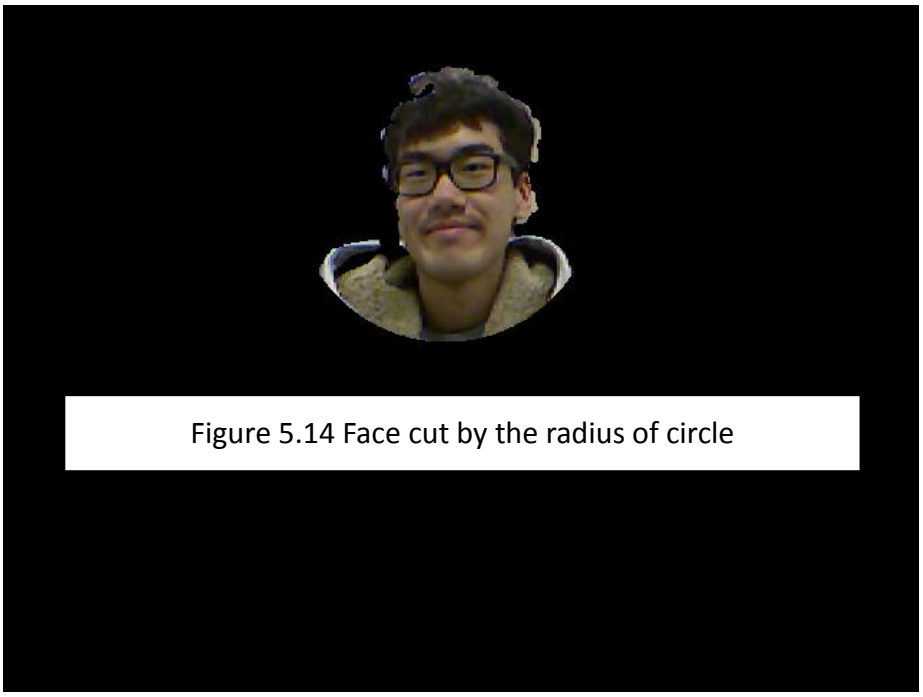


Figure 5.14 Face cut by the radius of circle

Edge detection

Since the above method does not cut the face very well, we have tried another way for extracting the face that is cutting by edge. We use the function of Canny in OpenCV to crete the edge result of the image.

Canny function provided by OpenCV. It is used to implement Canny edge detector which is an algorithm to detect the edges of the image. The algorithm is used to lower the error rate of detecting edge. Besides, it is good localization and minimal response.

The Canny edge detector use the upper and lower thresholods for checking the

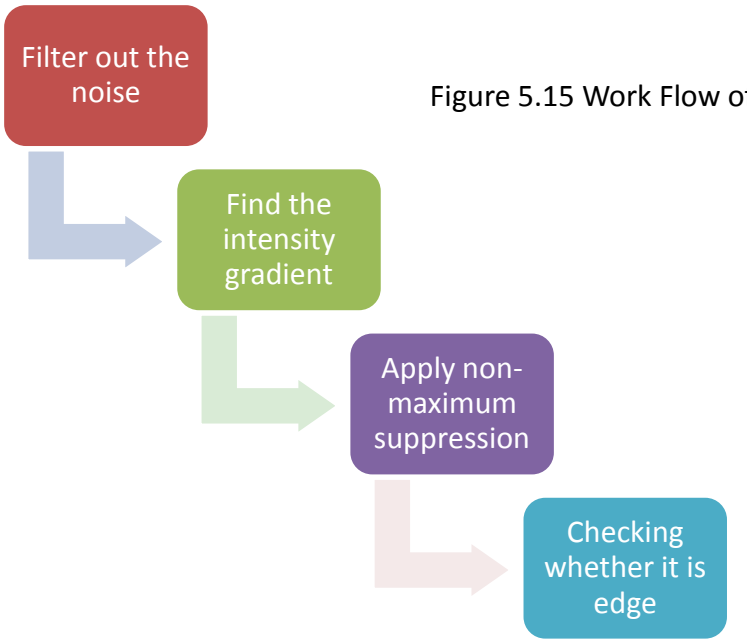


Figure 5.15 Work Flow of canny edge detector

edge of the image. If the pixel gradient is larger than the upper threshold, the piexl is meant to be edge. Meanwhile, if the pixel gradient is smaller than the lower, it is not edge. If the pixel garident is between the upper and lower thresholds, it would be edge if the connected pixel is larger the upper thresholds.

Finally, the edge image is created. However, after studing the edge image, we found that it is not possible to extract the head part of the player. It is because the there are lots of edges and the gardient changes is not obvious. It is extremely

difficult for us to cut the head part using those edge. To conclude,the way using the edge to cut the face is not possible at all. We have to think about another method to achieve the aim of our project.

Figure 5.16 Depth edge detection

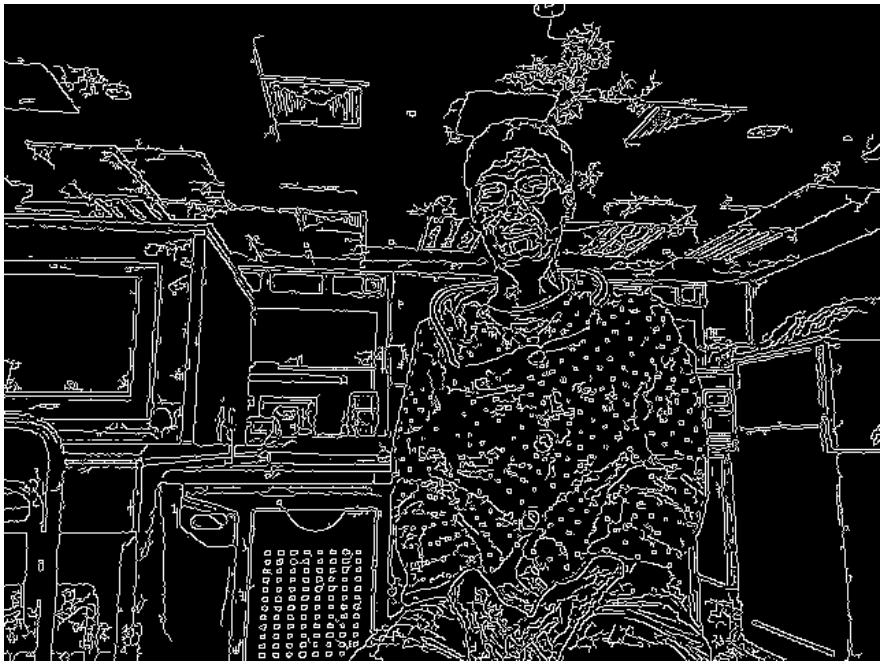


Figure 5.17 Color edge detection

Identify by the face colour

We have tried another method to extract the head part perfectly. We have set a certain colour range before. Next, we use the colour range for detecting the face. If the objects with that colour, it would be shown in the result image. We use three different type of the colour types such as HSV, RGB and YCbCr. The theory of these three are quite similar while the implementation are different between them. However, this method is not suitable for our project. Since there are some drawbacks, we would not apply this method in our project for extracting the heads. There are mainly two limitation on this method: the colour of the objects, race of pepople. For this method, we use the skin colour to filter out other objects. Since some of the objects would have the same or similar color to the skin, the results would definelty affected. The objects with the same colour of the skin other than the face would be shown. Besides, the race of people is the another problem. People's faces are varied with different race. It is impossible for us to use their skin colour for extracting the heads. By serious consideration, it is not a feasible method for our project.

Surface **Normal**

Since we cannot extract the head part using the edge part, we decide to use another method, surface normal. We firstly think that we have to cut the face using the chin as the margin. Therefore, we have to find the gradient difference between the chin and the neck. Using the depth valus and some computation in OpenCV, the surface normal of the image would be shown in the result.

What is surface normal?

In 3D geometry, surface normal is a vector at a point that perpendicular to the tangent plane to that surface at that point. The vector cross product of two edges is the surface normal for a convex polygon. It is obvious that the normal to a surface is not unique so that there would be a great difference even the gradient change is very small.

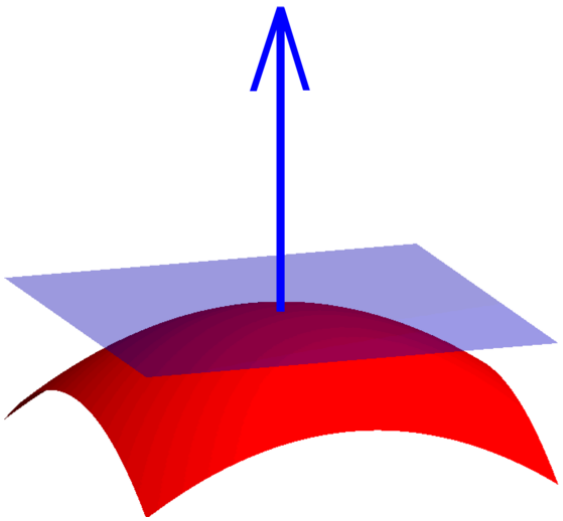


Figure 5.18 Surface normal

The cross product is calculated by this equation

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}.$$

And then it expands as

$$\mathbf{a} \times \mathbf{b} = \mathbf{i}a_2b_3 + \mathbf{j}a_3b_1 + \mathbf{k}a_1b_2 - \mathbf{i}a_3b_2 - \mathbf{j}a_1b_3 - \mathbf{k}a_2b_1.$$

The result of the cross product is

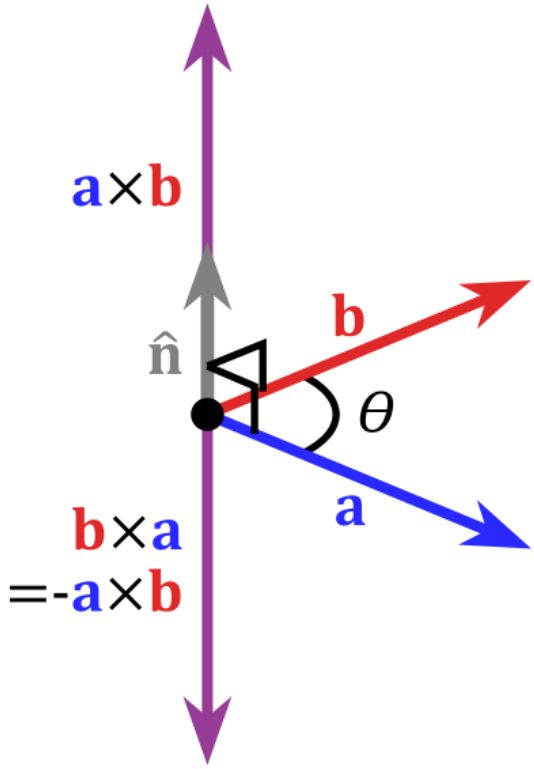


Figure 5.19 Result of cross product

However, we found that the result of the surface normal is not the perfect one. It is because the noise of the surface normal image is quite large. When we tried to extract the face by the above methods, there are a lots of difficulties we faced.

```
//d_mid is the depth value of the point that you want to calculate the normal
//d1 is the point above d_mid
//d2 is the point at the right of d_mid
//d3 is the point below d_mid
//d4 is the point at the left of d_mid
//length is the length between mid and other point
//we define the depth image as a matrix of 3d vector
//each element in the matrix is (0,0,depth)
//so that we could combine the vectors to form a surface
//and calculate the normal the by computer the cross product

Vec3f surfaceNormal(int d_mid, int d_top, int d_right, int d_bottom, int d_left, int length) {

    Vec3f normal = Vec3f(0, 0, 0);
    Vec3f v_top = Vec3f(0, 0, 0);
    Vec3f v_right = Vec3f(0, 0, 0);
    Vec3f v_bottom = Vec3f(0, 0, 0);
    Vec3f v_left = Vec3f(0, 0, 0);

    if (d_mid != 0) {
        //transform the depth values to vector pointing to top
        if (d_top != 0) {
            v_top[0] = 0;
            v_top[1] = length;
            v_top[2] = d_top - d_mid;
        }
```

To create surface normal, we need to find all the vector first. After that, the cross product of these few vectors are calculated. Next, sum them together to get the surface normal.

```
        //calculate the cross product of non-zero vectors and sum them together
        if (d_top != 0 && d_right != 0)
            normal += v_right.cross(v_top);
        if (d_right != 0 && d_bottom != 0)
            normal += v_bottom.cross(v_right);
        if (d_bottom != 0 && d_left != 0)
            normal += v_left.cross(v_bottom);
        if (d_top != 0 && d_left != 0)
            normal += v_top.cross(v_left);
    }

    //normalize the vector
    normal /= norm(normal);
    return normal;
}
```

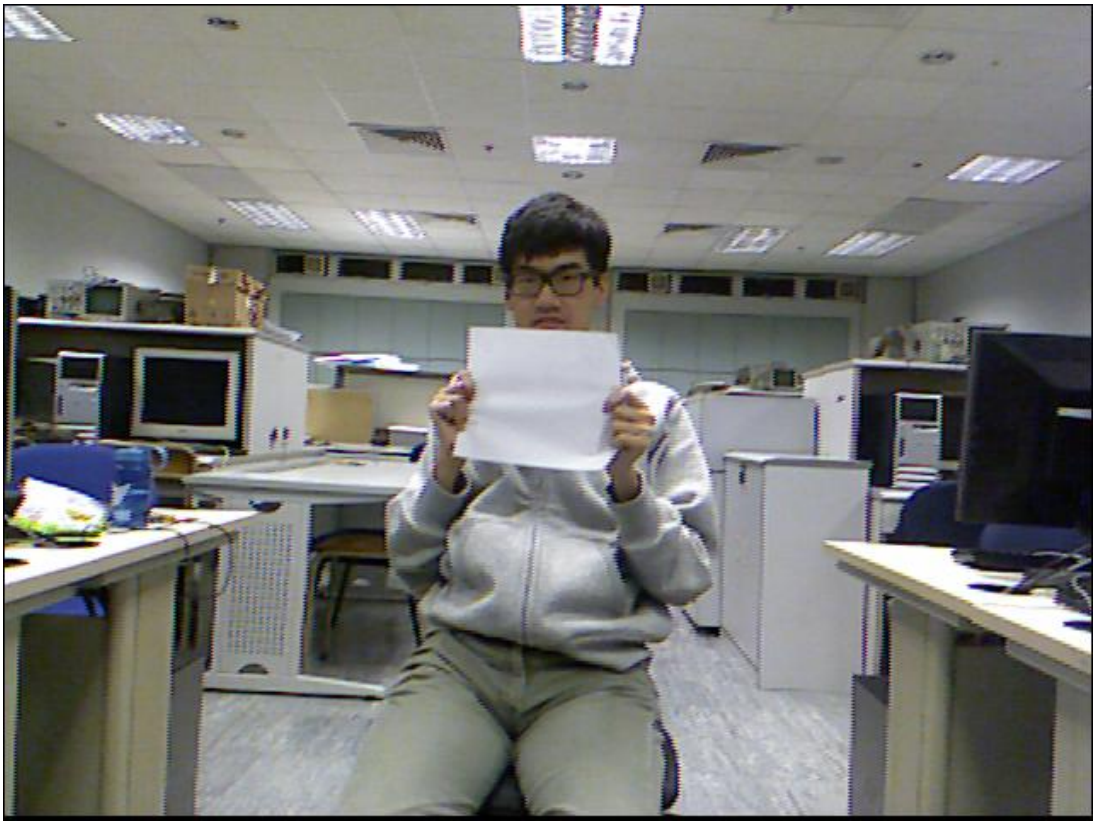
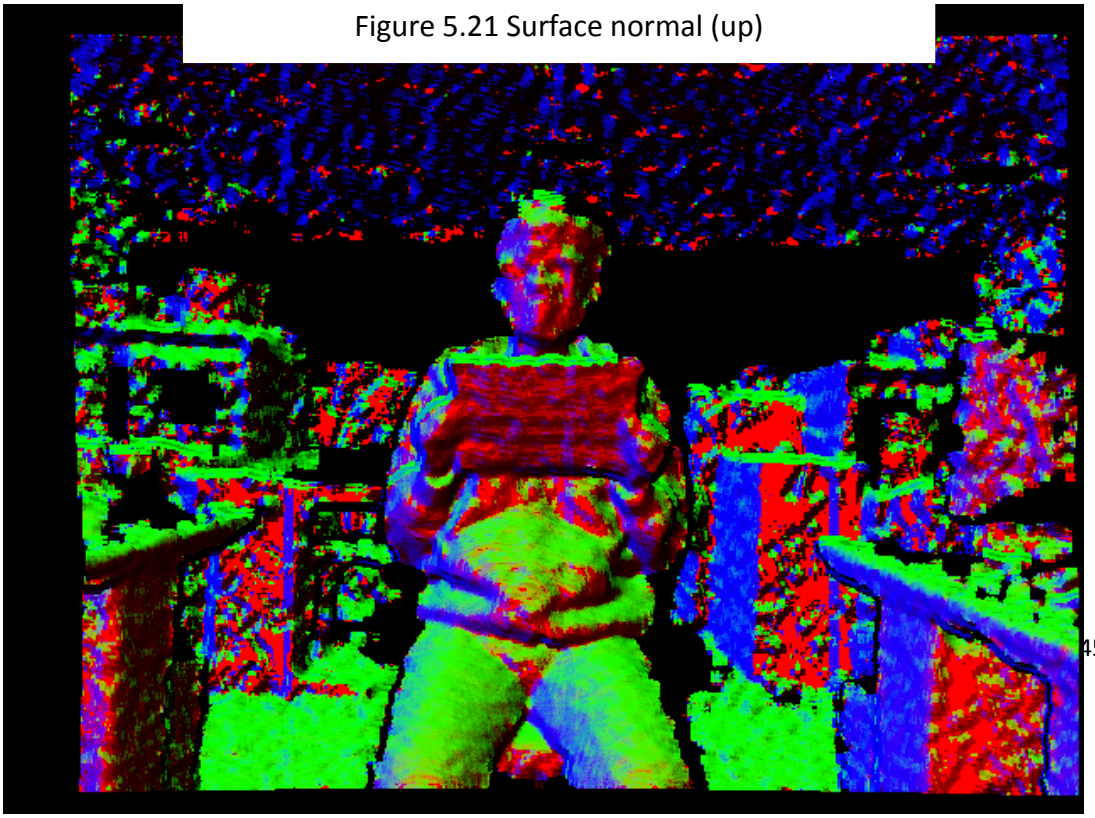


Figure 5.21 Surface normal (up)



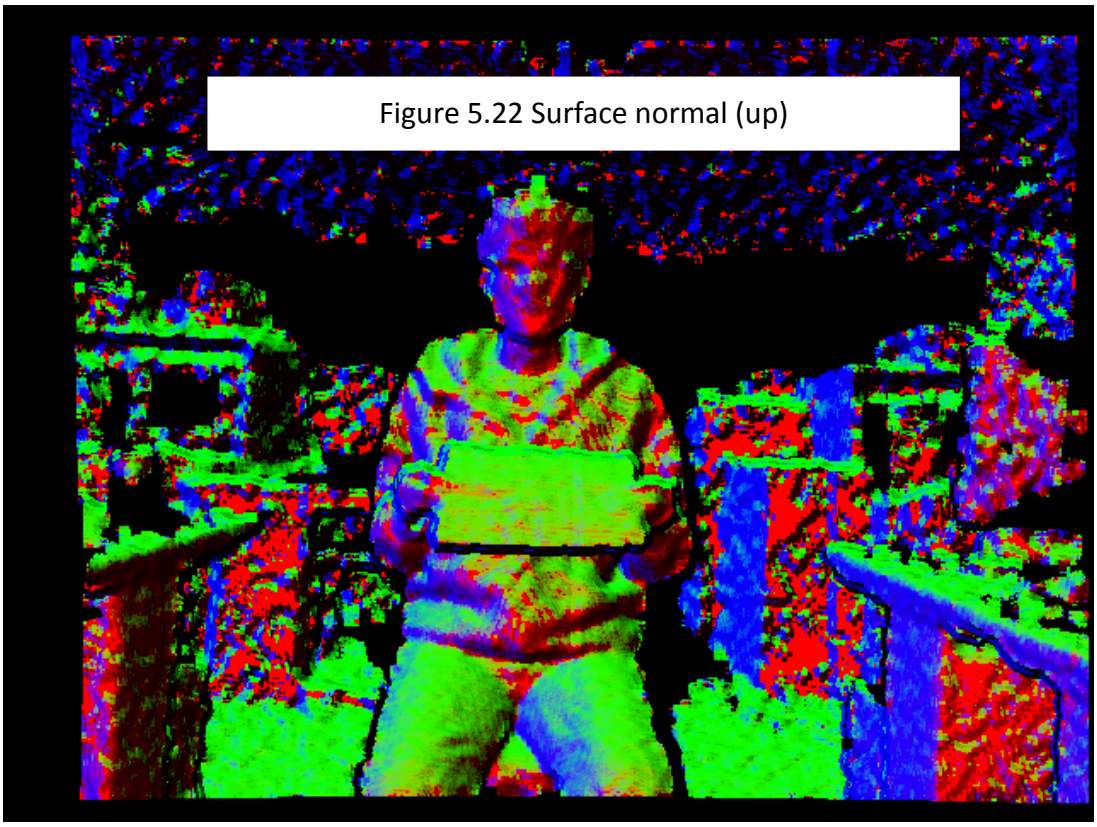


Figure 5.22 Surface normal (up)

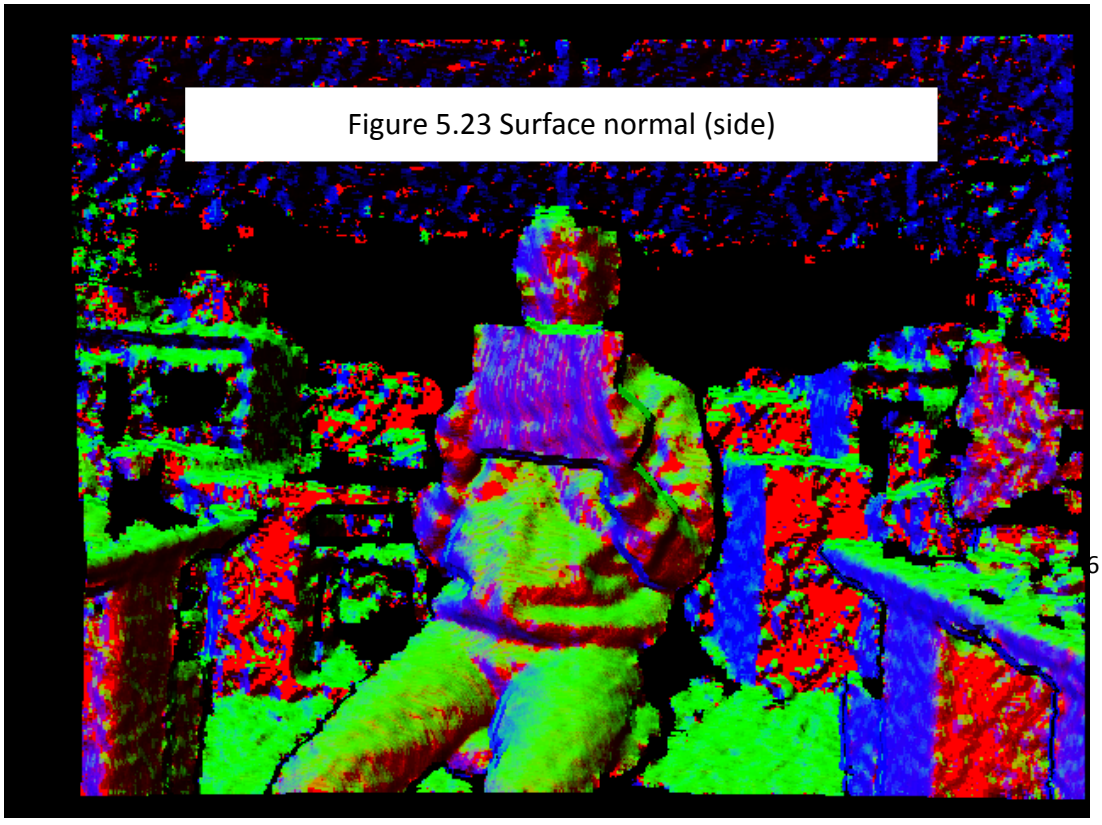


Figure 5.23 Surface normal (side)

Using the surface normal combined with depth value, it is obvious that it is the best way for us to extracted the head very well.

```
//calculate the surface normal by the depth value
Mat surfaceNorm = Mat(cDepthImg.rows, cDepthImg.cols, CV_32FC3, Scalar(0, 0, 0));
for (int i = 3; i < maskWithFilledHoles.rows - 3; i++) {
    for (int j = 3; j < maskWithFilledHoles.cols - 3; j++)
        if (maskWithFilledHoles.at<uchar > (i, j) == 255)
            surfaceNorm.at<Vec3f > (i, j) = surfaceNormal(cDepthImg.at<short>(i, j),
            cDepthImg.at<short>(i + 3, j), cDepthImg.at<short>(i, j + 3), cDepthImg.at<short>(i - 3, j),
            cDepthImg.at<short>(i, j - 3), 3);
}
//namedWindow("surfaceNorm");
//imshow("surfaceNorm", surfaceNorm);

//change the surface normal to mask by the SURFACE_NORM_ACCEPTENCE
Mat surfaceNormalMask = Mat(cDepthImg.rows, cDepthImg.cols, CV_8UC1, Scalar(255));
for (int i = centerY + CHIN_DETECH_RATIO * _radius; i < surfaceNormalMask.rows; i++) {
    for (int j = 0; j < surfaceNormalMask.cols; j++)
        if (surfaceNorm.at<Vec3f > (i, j)[1] < SURFACE_NORM_ACCEPTENCE ||
            surfaceNorm.at<Vec3f > (i, j)[0] < SURFACE_NORM_ACCEPTENCE) {
            surfaceNormalMask.at<uchar > (i, j) = 0;
        }
}
//namedWindow("surfaceNormalMask");
//imshow("surfaceNormalMask", surfaceNormalMask);

//tune the surface normal mask with respective depth value
for (int i = 0; i < surfaceNormalMask.rows; i++)
    for (int j = 0; j < surfaceNormalMask.cols; j++)
        if (surfaceNormalMask.at<uchar > (i, j) == 0 && cDepthImg.at<short> (i, j) < centerDepth +
            CHIN_DEPTH_BOUND && cDepthImg.at<short> (i, j) > centerDepth - CHIN_DEPTH_BOUND)
            surfaceNormalMask.at<uchar > (i, j) = 255;
//namedWindow("surfaceNormalMask2");
//imshow("surfaceNormalMask2", surfaceNormalMask);
```

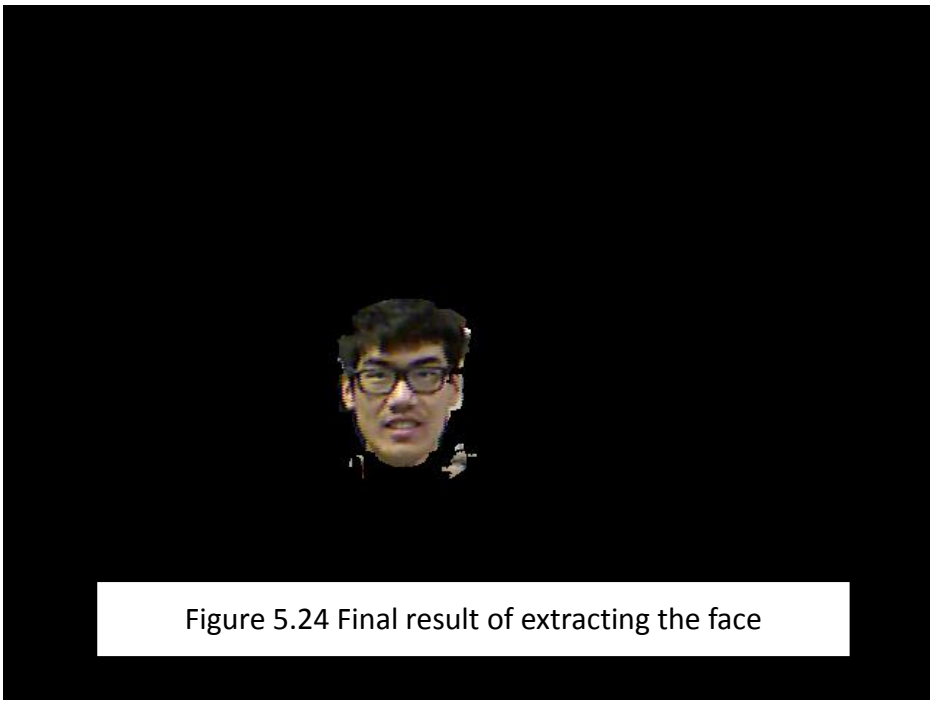


Figure 5.24 Final result of extracting the face

After extracting the face, we employ the head to the video. We have loaded a video first. Next, the extracted head is superimposed to the video. However, the frame rate is very low.

```
//check whether a video is loaded
if (argc == 2) {
    cap.open(argv[1]);
    if (!cap.isOpened())
        exit(0);
}

//if a video is load copy the head to the video
if (cap.isOpened()) {
    Mat frame;
    cap >> frame;
    Mat frame2;
    resize(frame, frame2, Size2i(640, 480));
    cBGRImg.copyTo(frame2, maskWithSurfaceNormalMask);
    namedWindow("frame");
    imshow("frame", frame2);
}
```

Figure 5.25 Dance Heads

To summarize what we have done in this milestone, we have drawn an UML diagram of our program. Finally, we have extrated the face by the surface normal combined with the depth value. Last but not least, the head is superimposed on the video succefully. However, the frame rate of the video is quite.

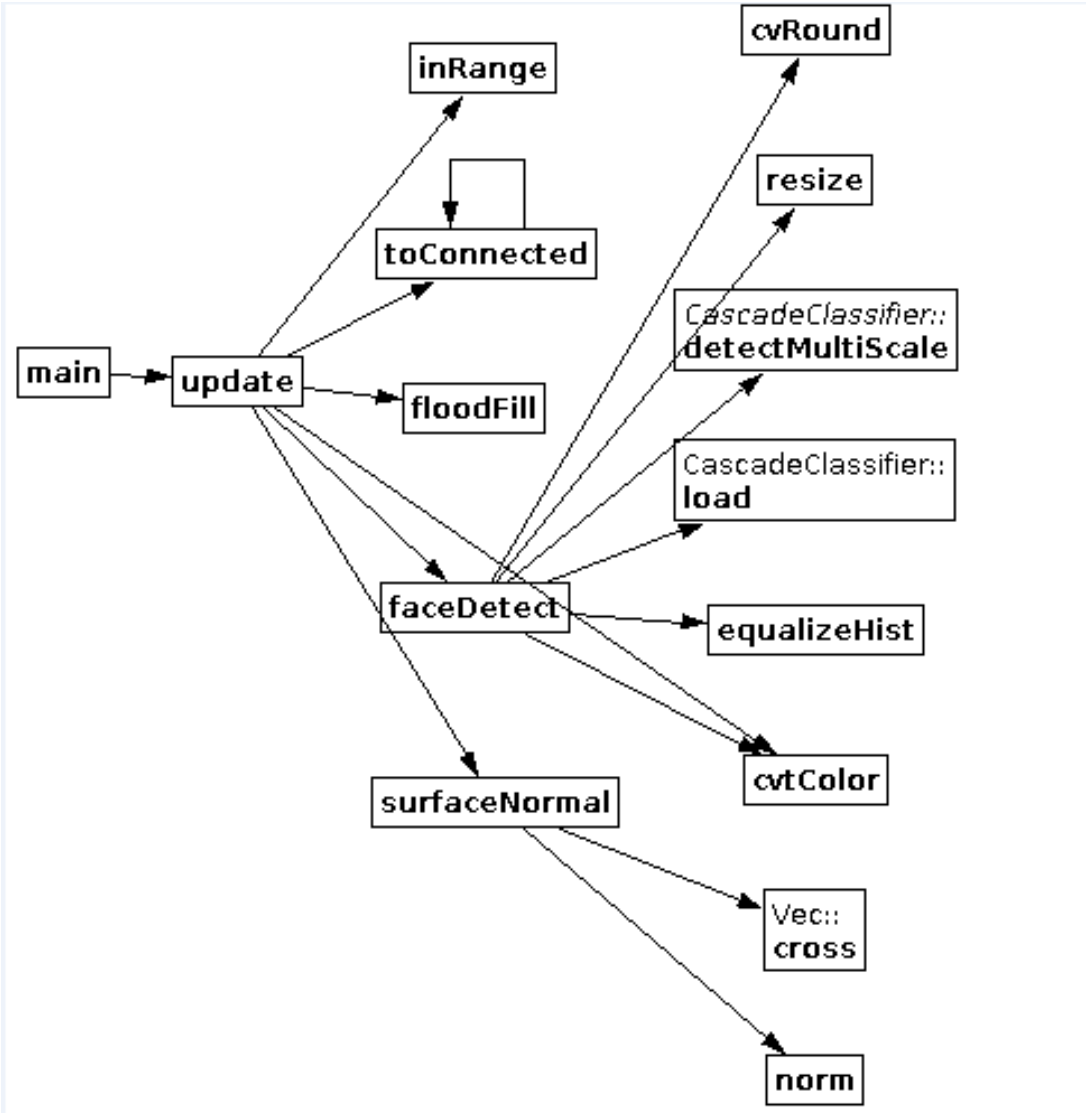


Figure 5.26 UML Diagram

Chapter 6 : Conclusion

To conclude, it is all known that Dance Heads is an interesting game. Meanwhile, Kinect is gaining more and more popular all over the world. In our objective, we have to use Kinect to achieve the similar application of Dance Heads.

Our first step is to study the SDK of Kinect for Windows an OpenNI. As SDK of Kinect for Windows provides special functions which is face tracking. Doubtless, face tracking SDK is quite powerful to track the face. However, the program should be developed in OpenCV. Due to this limitation, we have studied the other softwares, OpenNI and OpenCV. We have to integrate OpenCV and OpenNI with the Kinect. Finally, we develop program using the Kinect under the help of OpenCV and OpenNI.

The second step is to extract the head by using different methos.For OpenCV, there is a function which uses cascade classifier to detect the face of the image. Then we just cut the face by a circle with certain radius. However, the result is poor.

Another way we used is edge detection. We want to find the edge of the chin but this method is not the success one. It is because the noise of the image is high so that it is difficult to find out the chin.

Using the surface normal seems to a better way to achevie the objective.Yet, it is obvious that extracting the heads is not en easy task. We have tried several methods including edge detection, face detection in OpenCV and the surface normal. Sadly ,the result is not very well.

However, when we try to use the surface normal combined with the depth value. We would extract the head better than before. Therefore, we use that head and superimposed it on to the video. But we find theat the frame rate is low.

Admittedly, OpenCV is a powerful library that provide us many useful functions and computation. We would also use it to calculate the surface normal and display the image. it certainly is a useful software on computer graphic.

In this semester, we have learned a lot ranging from Kinect to OpenCV and OpenNI. It is very interesting doing something about image processing. We are happy that we have a chance of using Kinect in our final year project.

Difficulties encountered

Lack of knowledge in image processing

While we are doing this project, we have encounted many difficulties. First of all, we are not familiar with image processing. Since both of us come from Computer Engineering, we have little chance to do some programmings about image processing. This is our first time using Kinect, OpenCV and OpenNI. At the beginning, we have to spend lots of time to study the sample in OpenNI and OpenCV.

Difficult to find the best way for extracting the head

In this semster, we have tried out many methods to extract the head part. We have tried to use the edge method, face detection provided by OpenCV, surface normal and skin colour method. It is obvious that using the surface normal is temporarily the best way for extracting the head. However, we do not think that it is the best way to do so.

Limited resources and documentation

Moreover, there is only a few application developed by the others using Kinect, OpenNI and OpenCV. Hence, there is little documentation, reference and example that we would study and follow. In addition, we decide to OpenNI instead of the SDK of Kinect for Windows. We have to study the working principle of OpenNI. We have to integrate those softwares into our application.

Current limitation

Kinect limitation

There are some limitation on Kinect. For example, there may be some error when detecting some black objects such as hair, glasses etc. Since the black objects would absorb the infra-red, there would be errors in the result image.

Face detection

At this moment, for our program, only one face is detected. If two players use it at the same time, the result image would flush all the time. Hence, it is not stable for detecting more than one face. This is one of problem that we encounter in semseter one. For the application of the Dance Heads, three or more faces should be detected so that the application would be more fun.

Side view problem

Since we use the face detection provided by OpenCV, we judge the face by the front face only. If player swings his head, side view of the face would not be detected. Therefore, the head part of the image would not be superimposed to the Dance Heads. We have to find out a way to solve the problem.

Low frame rate

After extracting the face, we have found that the output frame rate is low . It may cause by the large quantity of calculating such as the calculating of surface normal.

Further work

Improvement of the output image

It seems that we have not finished our project in this semster as we still not merge the head very well to the video. Apart from this, we have to improve the quality of

the output image. Now, noise is quite large in the result. We have to do suppress the noise of the result so that the result would be better afterwards. We have to make an interesting application for the public to play so we have to ensure that there would be no error and no limitation when playing Dance Heads using our program.

Improve the frame rate

To improve the efficiency of our program, we will try to reduce the complexity and resize before doing complex calculation.

Implement advance features to Dance Heads

After extracting the head of the user, we have to employ it on video. We have to think about the other special features for our Dance Heads program. Therefore, more and more would enjoy it. For example, we try to use a 3D projector to play 3D Dance Heads using Kinect.

Point Cloud

During working on our project using Kinect, we have found a new technology called Point Cloud. A point cloud is a set of vertices in 3D coordinate system. A number of points of the object are collected by the device which would collect the 3D data. Point cloud means the set of points that measured by the 3D device. After that, the point cloud would be used for another purpose such as animation, visualization, rendering etc.

Point Cloud Library is an open source framework for 2D/3D image and point cloud processing. The library includes surface reconstruction, model filtering,

segmentation and feature estimation. The most useful function in PCL is recognizing the object in the world based on geometric appearance of the objects. It is hoped that Point Cloud would be useful for our project. We decide to give a try on the Point Cloud.

Chapter 7 :Acknowledgement

We would like to cherish this opportunity to thank our supervisor, Professor Michael Lyu who gives us great support and advice for our project. He also reminds us how to make a good presentation.

In addition, we would like to thank Mr.Edward Yan in VIEW Lab. He always provides some valuable advice and technical support when we faced problem.

We are not sure that our project would be success or not in future, but we are quite confident to overcome the difficulties with the help of Professor Lyu and Mr.Edward.

Chapter 8 : References

[1] Kurt Demaagd, Anthoy Oliver, Nathan Oostendorp&Katherine Scott “Practical Computer Vision with SimpleCV”

[2] Gary Bradski& Adrian Kaebler “Learning OpenCV”

[3] Kinect for Windows :

http://www.microsoft.com/en-us/kinectforwindows/

[4] Microsoft Developer Network

http://msdn.microsoft.com/en-us/default.aspx

[5]OpenCV

http://opencv.org/

[6]OpenNI

http://openni.org/

[7] Face Detection in OpenCV

http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html

[8] Haar-like features

http://en.wikipedia.org/wiki/Haar-like_features

[9] Face Recognizer in OpenCV

http://docs.opencv.org/modules/contrib/doc/facerec/index.html?highlight=face

[10] Using Kinect and OpenNI compatible depth sensors

http://docs.opencv.org/trunk/doc/user_guide/ug_highgui.html#using-kinect-and-other-openni-compatible-depth-sensors

[11] Cascade classifier

http://docs.opencv.org/trunk/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html#cascade-classifier

[12]OpenNI history

http://en.wikipedia.org/wiki/OpenNI

[13] Getting started with Kinect for Windows SDK

http://msdn.microsoft.com/en-us/library/hh855354.aspx#feedback

[14] Face tracking

http://msdn.microsoft.com/en-us/library/jj130970.aspx

[15] Dance Heads

http://danceheads.com/

[16] Opencv 2.4.3

http://fossies.org/dox/OpenCV-2.4.3/objdetect_8hpp.html

[17] Yong KokChing, Anton SatriaPrabuwono, RizaSulaiman “Visitor face tracking system using OpenCV library

[18]OpenNI Programmer Guide

http://openni.org/Documentation/ProgrammerGuide.html