LYU 1201

香港中文大學
The Chinese University of Hong Kong

Department of Computer Science

& Engineering

3D DANCE HEAD USING KINECT AND 3D PROJECTOR

Student: Chan Wai Yeung (1155005589)

Supervised by Prof. LYU Rung Tsong Michael

Abstract

Kinect is a device used mostly for motion sensing game in Xbox. More and more people get addicted in Kinect. In this project, we would like to develop an interesting application about Dance Heads.We are going to design and implement an algorithm to achieve Dance Heads using Kinect Camera. 3D projector is used in order to project out the 3D Head part of the player. Our application could include the following features: extracted the head part and superimpose to the video, use point cloud to reconstruct the image captured and exchange two player's faces.

Content

Chapter 1: Introduction

Motivation

There are many video called “Dance Heads” when surfing the Internet such as Youtube. Those videos are extremely funny and many people are interested. It draws a lot of attention. At that time, many people were addicted to this entertainment when Dance Heads was invented. Doubtlessly, Dance Heads is an innovative entertainment for all ages of people.

Figure 1.1 Dance Heads

The principle of Dance Heads is not difficult to work out. The player’s heads are extracted and then superimposed to the bodies of some professional dancers with animated background. The players just need to swing their heads at that moment. According to the background music, the clothes of the dancers would be changed. It is not necessary for the player know how to dance. They only need to stand and swing their head part. Through the videos, the players seem to dancing at that time.

Unfortunately, there are so many limitations of making Dance Heads videos. To start with, the players need to stand up and wear coloured clothes when capturing their heads part. The most important is the coloured clothes should be same colour to the backgrounds. Also, the coloured clothes should cover the player's neck as shown in the figure.

Figure 1.2 Making Dance

While we are watching the Dance Heads video, what comes to our mind is how to implement the Dance Heads without the above limitations. At the same time, we have noticed that XBOX are now gaining popular all over the world. We are then inspired by the XBOX 360 Kinect.After that, we come up the idea of using Kinect to capture the head instead of using camera.

Figure 1.3 Kinect

Figure 1.4 3D projector

Nowadays, 3D technology is extremely popular. Many movies such as Avatar, Iron man 3 would use 3D technology in order to attract more people to watch the movies. Therefore, after capturing and extracting the head part of the players, we use a 3D projector to present the image captured by the Kinect.

Background

Searching “Dance Head” in the Internet, it is not difficult to find out some videos about “Dance Heads”. What is “Dance Heads”? “Dance Heads” is a video that player’sheadsareextracted and then superimposedto dancers with an animated background and music. Dance Heads was found in 2003. It was established by a company that is called Dance Head Inc. This 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 then superimposed to different bodies.

Figure 1.5 Logo of Dance

These video were popular in the past. Many company, for example, Coca-Cola, Disney to name but a few, made their advertisement using the Dance Heads videos. 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.

It is interesting of making Dance Heads videos. The players have to wear clothes with the same colour as the background. The camera then captures their heads and removes the other part of their bodies. After that, the players choose their favourite songs and stand in front of the rail. Finally, Dance Heads videos are formed.

Figure 1.6 Make of Dance Heads video

As shown in Dance Heads specification, it requires 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 players are three. Sadly, the cost of this interactive entertainment is very expensive.

For the 3D Vision, there are several ways to present the 3D image. There are mainly two types. The easiest way to distinguish is whether it needs glasses or not. Firstly, to show the 3D image, the users have to wear glasses such as active or passive glasses. If no glasses are needed, some technologies are required to process the 3D image. However, the cost of it is extremely expensive. Hence, we would find that most of 3D technology in our daily life required us wearing glasses, for example, 3D television or watching film.

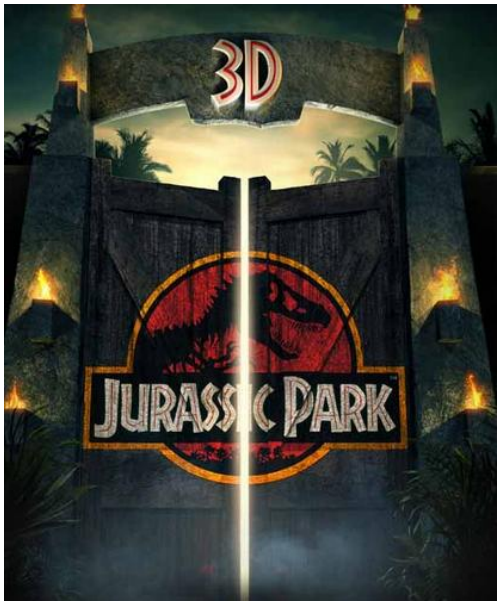


Figure 1.7 Poster of 3D film

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 extract the player's heads using Kinect camera to captured image. After, we should use different software and libraries to render the image into 3D graphic. 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.
- Study Point Cloud library used for reconstruct the 3D point image.
- Make some special features for the application like exchange two players' faces.

Runtime environment

Firstly, we use Ubuntu as the main platform. As we in semester one found that there are some limitations using Kinect SDK, we use OpenNI and OpenCV instead of Kinect SDK. However we still use Kinect camera as the major component to capture the bodies and faces of the players.

In the project, we use C++ as the programming language and the program is developed in Ubuntu in semester one and Window in semester two respectively. For semester two, we have to develop in Windows because the 3D Vision Pro projector is compatible for Windows not for Ubuntu.

Several graphs are made finally. We are interested the players' faces only. So, the depth value of the players' faces is used. The most difficult part is how to extract the head part well in order to perform perfectly in Dance Heads effect.

Ubuntu is a Linux based operating system. The latest version is 12.10.Many software packages are components of Ubuntu. Therefore, we could integrate different open-source libraries into our final year project under the Ubuntu system. We have used OpenNI and OpenCV and these two open-source software are also compatible in Ubuntu. Besides, Point Cloud Library is used to construct the 3D image for the use of 3D projector. Point Cloud Library could run on different operating system such as Linux, Windows and Mac.

Cmake is then used to control the software complication process using complier independent configuration files and simple platform.

Figure 1.8 Logo of CMake

Chapter 2: Kinect

Introduction

The name of Kinect comes from the word “kinetic” and “connection”. Kinect was first launched at 2010. In 2012, Microsoft had launched Kinect for Windows operating system. Kinect is 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. it 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 device

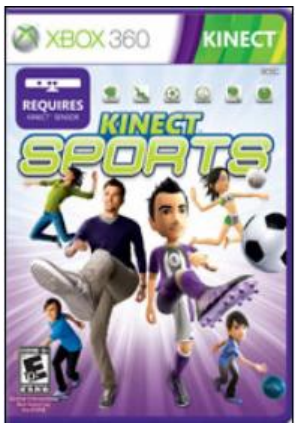


Figure 2.2
Game for Xbox Kinect

Nowadays, XBox 360 with Kinect is one of the most popular electronic game devices around 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

Figure 2.3 Kinect structure

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.

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 players, tracks 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)

Figure 2.4 Table of Kinect specifications

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 1.7 and is released March 2013. 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 1.7 and is updated March 2013.

The face tracking SDK is released in 1.5 SDK and developer toolkit special features. And it 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
Infrared Emitter Control API
Introducing New Samples!
Kinect Studio 1.6.0
The Infrared Stream Is Exposed in the API
Support for Virtual Machines

Figure 2.5 Table of 1.6V SDK and developer toolkit

Microsoft also launches the latest update at March 2013. It makes the Kinect SDK more powerful. The latest SDK offer new tools, samples and features for the developers developing smart application. Kinect interactions are involved in it. It also improves the control of Windows Presentation Foundation which is easier for developers to build applications. In addition, grip recognition and physical interaction cone are the new features of the latest SDK.

Now, it is possible for developers to program Kinect for Windows applications which would reconstruct high-quality, three-dimensional renderings of people and objects in real time.

1.7 of the SDK and the Developer Toolkit
Real-time, GPU-assisted 3-D object and scene reconstruction by using the Kinect for Windows sensor
Ability to infer relative sensor position and orientation from a 3-D scene for augmented reality application
Advanced algorithms that are powerful enough for large sensor movements and scene changes during scanning
DirectX11 compatible graphics cards supported
Non-real time CPU mode for non-interactive rate scenarios
Kinect Fusion Studio and samples demonstrate 3-D scanning capabilities
AMD Radeon 7950 and NVidia GTX560 have been validated to run at interactive rates

Figure 2.6 Table of 1.7V SDK and developer toolkit

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

Hardware and Software Interaction with an Application

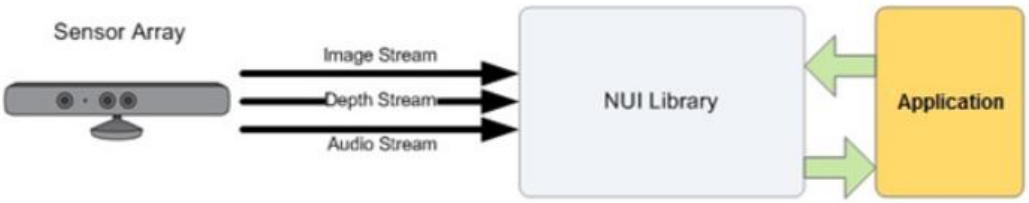


Figure 2.7 Kinect for Windows structure

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.

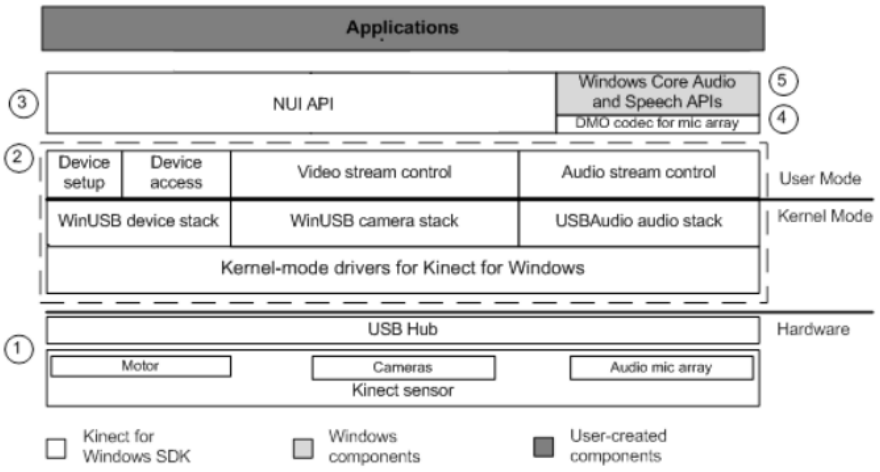


Figure 2.8 SDK Architecture

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 supports 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

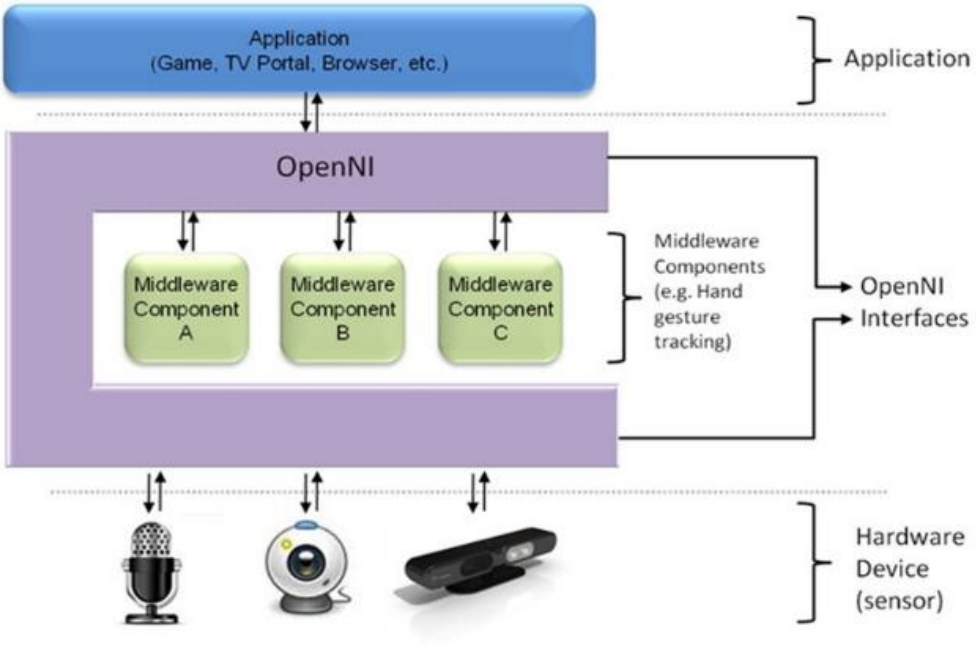


Figure 3.1 Three-layered view of the OpenNI Concept

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 2.4.5 and released on April 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 etc. Wrappers in other languages have also been developed.

Overview

Many applications are provided by OpenCV. The followings are some examples, 2D 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.

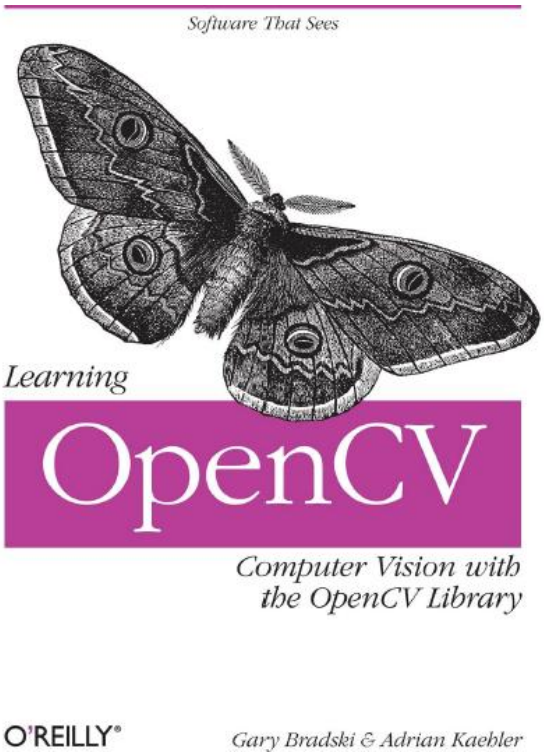


Figure 3.2 Cover of reference book

CMake

Introduction

CMake is an open source and cross platform build system using compiler independent method. It is used to support the applications that composed of different libraries. CMake stands for cross platform make. It is also used for the use in conjunction with native build environment like Microsoft Visual Studio. It only requires C++ compiler on its own build system.

CMake was first developed in 1999 and used for the need of cross platform build environment. After, it was implemented in 2000 and developed in 2001. Afterwards, CMake was still developed and improved.

CMake is compatible to project which has several libraries which have different directories. Moreover, CMake could work with projects which require executables to be created before generating code to be compiled for the final application.

The build process with CMake has two stages. Firstly, standard build files are created from configuration files. After that, the platform's native build tools are used for the actual building. For the build project, there is a CMakeLists.txt file which controls the build process. When there are many built-in rules for compiling the software libraries and executables, there are also provisions for custom build rules.

Visualization Toolkit (VTK)

Introduction

Visualization Toolkit is open source and cross platform software system for image processing, visualization and 3D computer graphic.It runs on Linux, Windows, and Mac platforms. It consists of C++ library and many interpreted interface layers such as Java, Python. VTK supports different visualization algorithms, for example, scalar, vector, tensor, texture, and volumetric methods. Furthermore, it supports modeling techniques like mesh smoothing, cutting, implicit modeling etc.

VTK has an extensive information visualization framework. It also has a suite of 3D interaction widgets and supports parallel processing. Last, it integrates with various databases on GUI toolkits such as Qt and Tk. The latest version of VTK is 5.10.1. Point Cloud makes use of the VTK library for rendering for range image and 2D operating.

Point Cloud Library

Introduction

Point Cloud library

(PCL) is large scale, open project for2D and 3D image processing. PCL framework consists of several advanced algorithms, for example surface reconstruction, filtering, registration, model segmentation and fitting. They are used for different application such as filtering outliers from noisy data, connecting 3D point cloud together, extracting key points, recognizing object based on geometric appearance, creating surfaces from point cloud and visualizing.

PCL is free for research and commercial use. It is open source software. Besides, it is cross platform and compiled on different operating system such as Windows, Linux,MacOS, Android and IIOS. PCL is divided into a series of smaller code libraries which could be complied separately so that it is easier for the users developing programs. The modularity is important for distributing PCL on platforms with reduced computational or size constraints

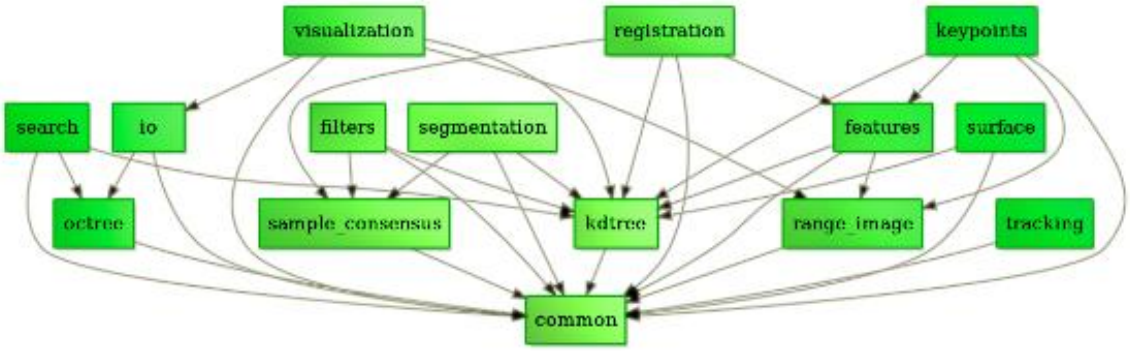


Figure 3.3 Library depencence

PCL is developed by many scientists and engineers from different organization and university around the world. Meanwhile, it is financially support by many companies all over the world such as Toyota, Google, NVidia to name but a few.

Overview

Point Cloud is a data structure used to represent s wide range of multi-dimensional points. It is commonly used for representing 3 dimensional data. In the 3D point cloud, these points represent X, Y, and Z geometric coordinates in order to show the sampled surface.

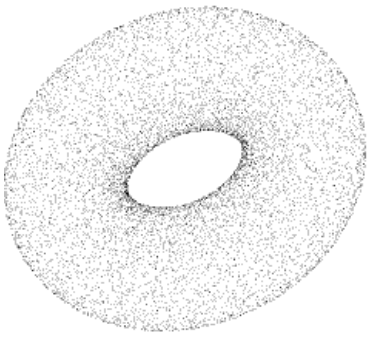


Figure3.4 Example of point cloud

The most interesting is a point cloud iscompatible to a number of hardware sensors. The hardware sensors include 3D scanner, stereo camera or generated from computer program. Point cloud library also supports OpenNI 3D interfaces and hence could acquire and process data from different devices like Microsoft Kinect or PrimeSensor 3D camera.

Point Cloud library is spilt into a number of modules. The followings are the most important modules.

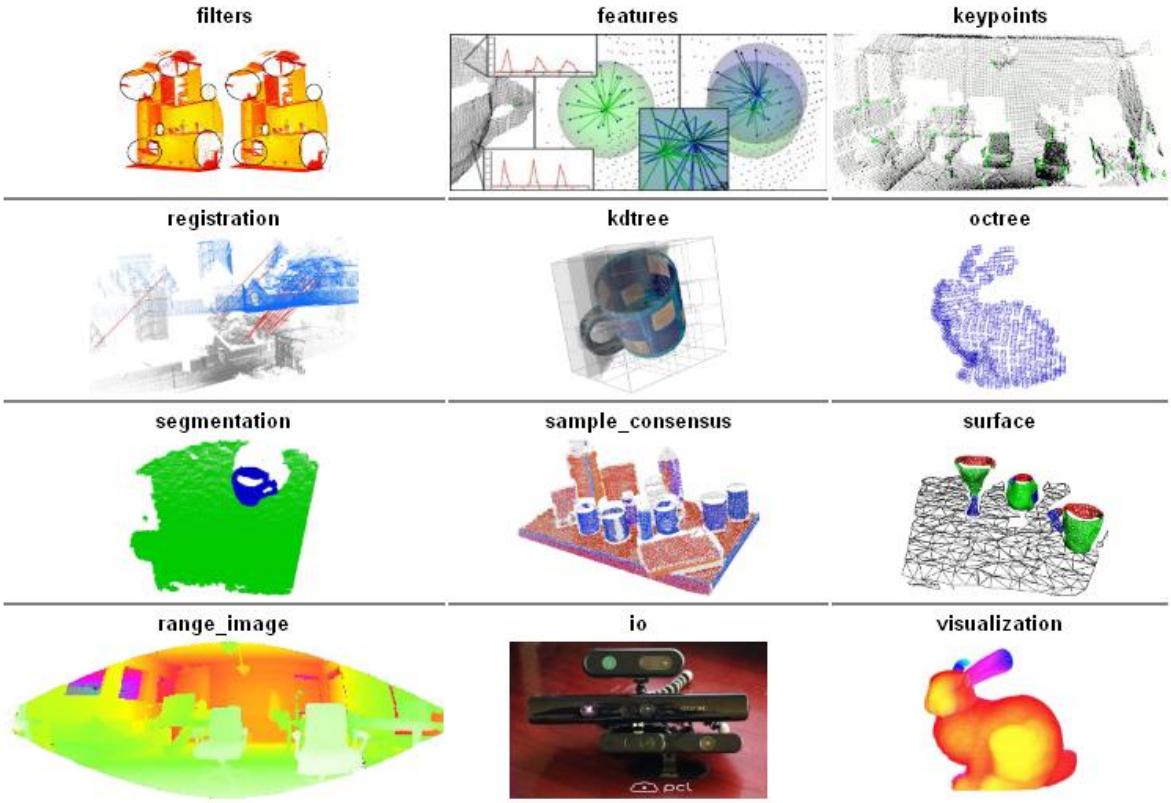


Figure 3.5 Major 9 modules in PCL

For our final year project, we are mostly focus on the io and visualization modules. We would discuss the module of visualization and io in details since we mainly use this module in our program.

Module Visualization in PCL

Visualization modules are used to phototype and visualize the results of algorithms operating on 3D point cloud data. It is similar to OpenCV'shighgui routines in displaying 2D image and forming 2D shapes. Moreover, it offers several methods for setting and rendering visual properties, for example, colour, size etc. it also provides method for drawing 3D shapes from sets of point.

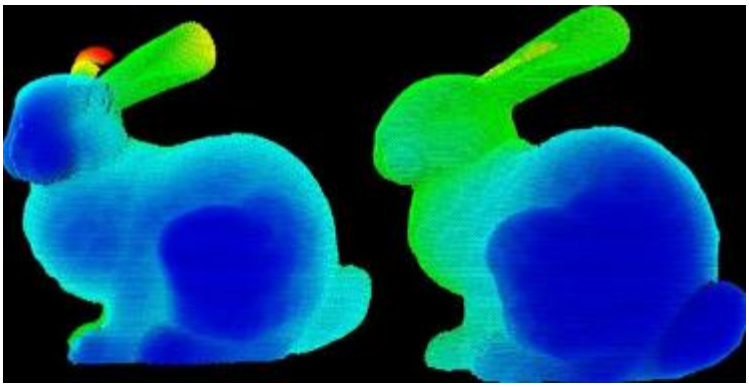


Figure 3.6 Example of visualization output

These two methods are extremely useful in our project since we have to reconstruct the head image in 3D shapes. The packages make use of VTK library as mentioned before for rendering 3D image and 2D operation.

Module IO in PCL

The io library consists of many classes and function for writing and reading the point cloud data and capturing point clouds from sensing devices. The OpenNI grabber framework in PCL is useful for our project. PCL with compatible cameras would use the data source to generate 3D point clouds.

Figure 3.7 Kinect device

The OpenNI grabber interface is very powerful and makes the connection to OpenNI compatible cameras in our program code.

Chapter 4: Three-dimensional (3D) display technology

Introduction

Since we want to use 3D projector in our project, we would like to make a research of 3D display technology. We focus on the 3D vision published by NVIDIA. Let's explain two term, 3D and stereoscopy. The term 3D means three dimensional while stereoscopy means stereo vision. However, most of people say 3D is 3D display nowadays. In fact, stereoscopy is the most accurate term. Now, several companies do research 3D technology and release many products such as NVIDIA (3D Vision), Sony (PlayStation 3) to name but a few.

As people eyes could have depth perception when seeing the object, people would see the 3D vision of that object. Each person has two eyes; there is binocular parallax when seeing the objects and then the brain would merge two images so that stereoscopy vision would be formed. The size, light, material would also affect the 3D judgment of people.

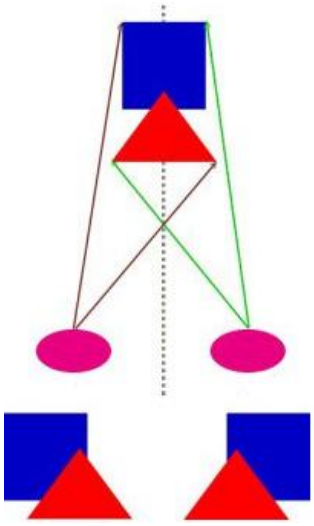


Figure 4.1 Principle of stereoscopy

Overview

There are many ways to show the images are in 3D. All these ways are divided into two main categories. The first category is that the users need to wear glasses. For the other category, obviously, the users do not need to wear glasses.

With glasses	**Without glasses**
Active glasses	Spatial-multiplexed
Passive glasses	Time-multiplexed
Anaglyph	Depth-fused multi-layer
Polarizer	Holographic
Wavelength multiplex	Volumetric
Head mount display	

Figure 4.2 Methods of forming stereoscopy

For different methods, they may have their advantages and disadvantages.

For our project, we would discuss a few of the above methods.

3D with active glasses

For this moment, the most common way to show 3D image is using glasses. There are mainly two types of glasses: active and passive. To start with, active glasses are also called shutter glasses.

The principle of this is based on the frequency of the screen and the screen would alternatelydisplaythe left-eyeand right-eyeimages. At the same time, the glasses would shield left and right alternately like the figure shown.

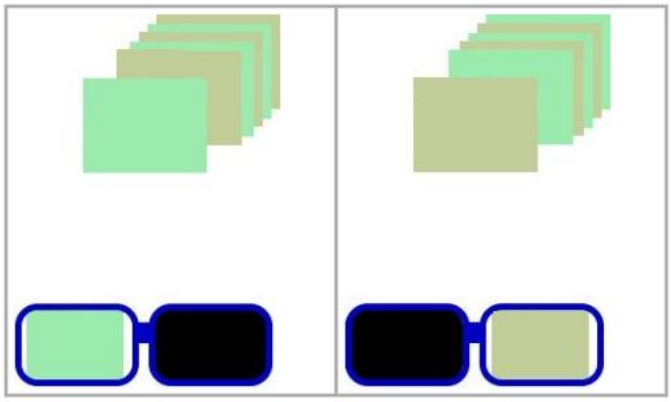


Figure 4.3 Principle of stereoscopy with active glasses

Figure 4.4 Active glasses andsynchronization signal transmitter

As human eyes could temporary store the images that they see, eyes could see different image. Hence people would sense the images are three-dimensional. Nowadays, most of the glasses are made up of liquid crystal. Apart from the glasses, the projector should be 120Hz. It is important for the glasses synchronous with the screen. For the wireless glasses, there should be synchronizationsignaltransmitter in order to show the 3D images.

3D with passive glasses

The passive glasses include anaglyph, polarize and wavelength multiplex. The principle of it is similar to the active glasses. Many people have heard about "red-blue (green)" glasses as shown in the figure. For example, people use colored glasses. Only green information for left eye while information for right eye. Finally, merge them together. 3D images would be shown. The cost of this method is the cheapest.

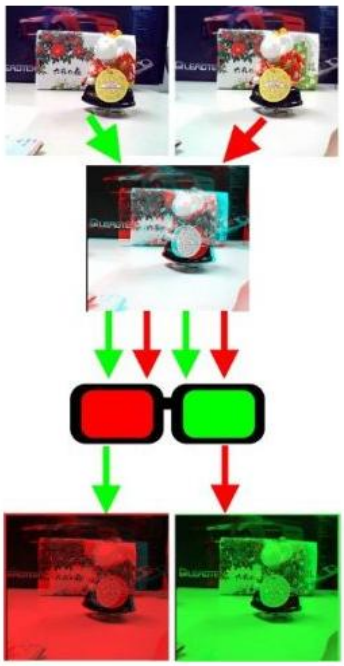


Figure 4.5 Principle of red-green glasses

As the figure shown, the left-eye imageofthescreen are presented to the light vibrating in thelightvibratingin a horizontal direction while the right-eye images are presented to the light vibrating in the vertical direction.As longastheleft eye of the glassesputahorizontallinepolarizer, onlyimageofthehorizontal vibration would be shown. Similarly, averticallinepolarizer is put on right eye. Then, only image of vertical vibration would be shown. High frequency projector is needed for showing the 3D images. A polarizing film should be used in front

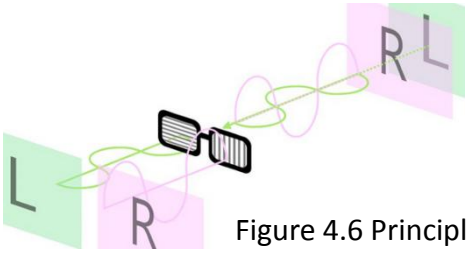


Figure 4.6 Principle of stereoscopy with passive glasses

of the projector.

3D Vision with NVIDIA

NVIDIA is a great company that releases display cards and components for 3D display system. 3D Vision Pro provides active shutter glasses and robust radio-based control hub with 120Hz projector and panels.

Minimum system requirements
Supported NVIDIA Quadro or GeForce graphics board
Quadro based solutions
DirectX stereo support available on Windows 7 or Windows 8only
Intel Core 2 Duo or AMD Athlon X2 CPU or higher
1GB of system memory. (2GB is recommended)
100 MB free disk space
3D Vision-Ready Display or Projector

Figure 4.7 Requirements of NVIDIA 3D Vision Pro

However, NVIDIA Quadro Graphics Boards and 3D Vision Pro also support different stereoscopic projectors and display devices which are not 3D Vision

Ready. Some of the display cards could only run on Windows, while some of them just operate on Linux.

Chapter 5: 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 methods to detect the face. The first one is Kinect SDK and the other one is face detection provided by OpenCV. We would discuss 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 face detection 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 analyses 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.

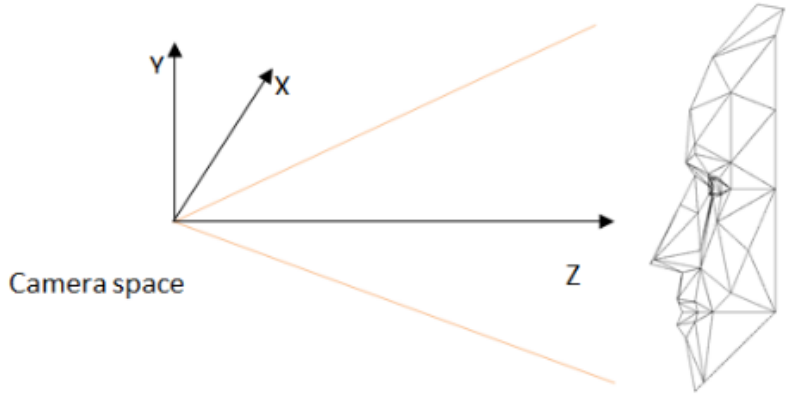


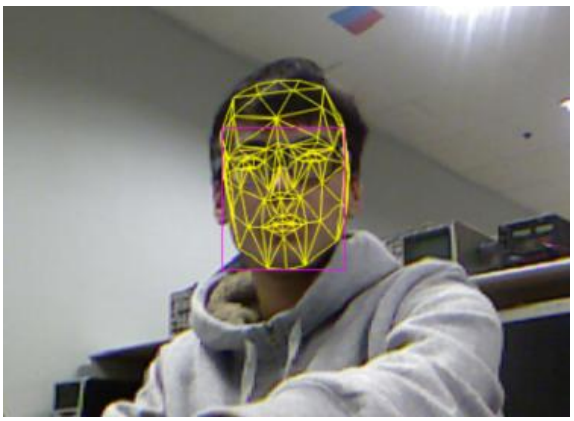
Figure 5.1 Coordinate system in Kinect SDK

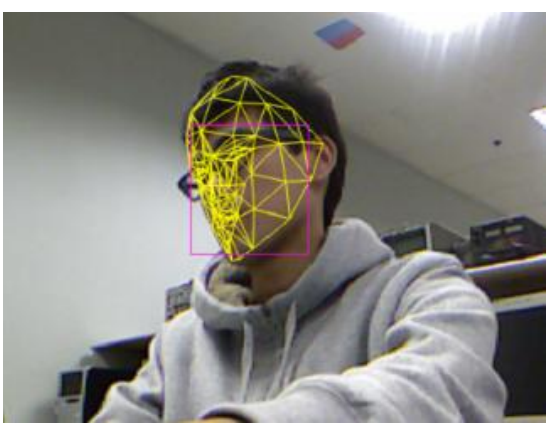
Face tracking outputs

Theoutput 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 resolutions.





of detected face

Figure 5.3 Side view of detected face

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 samplesare 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.

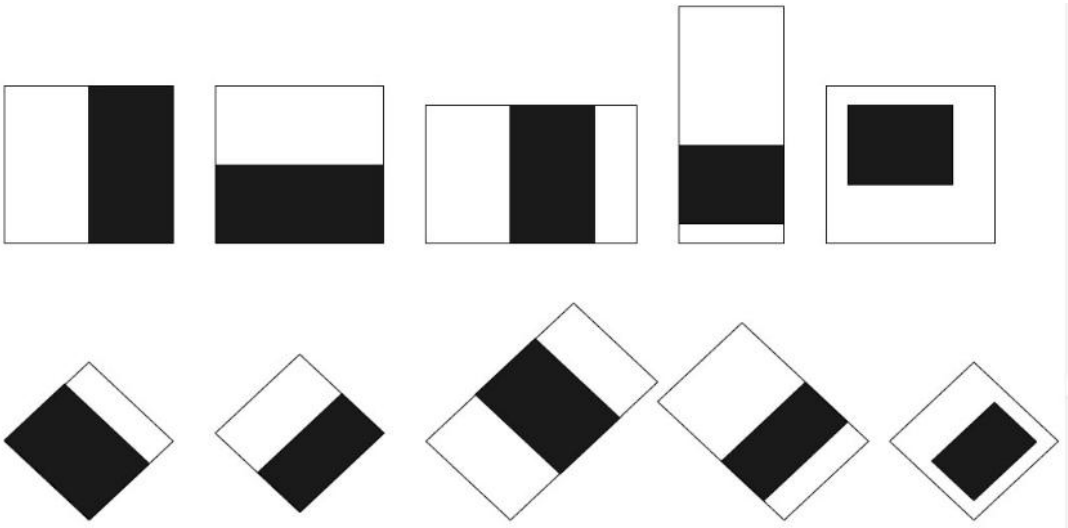
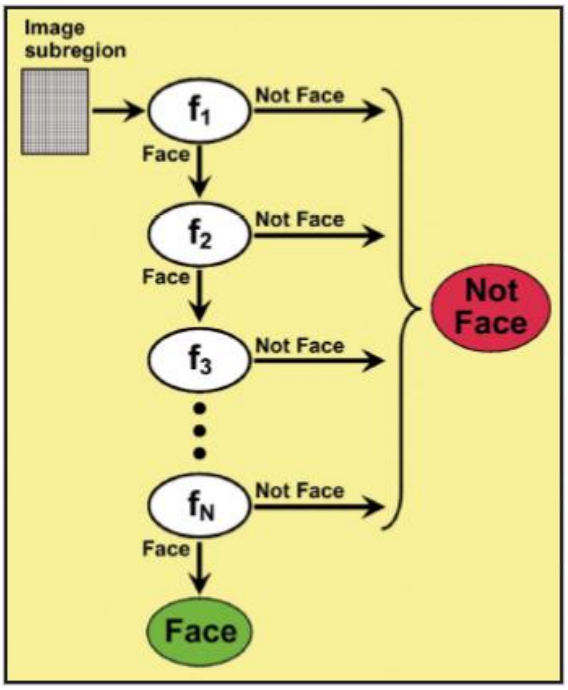


Figure 5.4 Rectangular block used to create classifier

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

Figure 5.5 Cascade classifier stage to detect face

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 5.6 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

Figure 5.7 Table of Comparison

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.

Chapter 6: Design and implementation

To do our final year project Dance Heads using Kinect, we firstly 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.

Therefore, in the semester one, we use Linux as the operating system. We make good use of OpenNI to retrieve the images data captured by Kinect. After, using face detection provided by OpenCV to get the face of the player. Finally, we use the face and then superimposed into the video. Dance Heads video is shown.

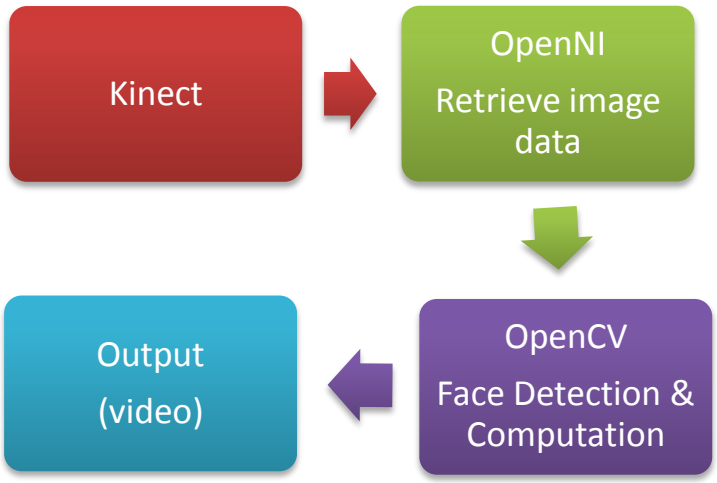


Figure 6.1 work flow in 1st semester

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 the first semester, we have done several things. We have tried out many methods for extracting the heads perfectly.These methods include face detection provided by OpenCV, edge detection, surface normal and surface normal with the depth. Besides, we have to make sure the quality of the images. Hence, we have to do something such as filling the holes in the images. Finally, the head is superimposed to the video. We would discuss those methods one by one in details later.

However, for our project in semester one, there are some limitations and problem. In our application, the video could only available for one user, it could only detect one face each time. The frame rate of the video is extremely low. The application could not detect the side view of the player face.

In the second semester, in order to project out the 3D images using NVIDIA 3D vision Pro, we use Windows as the operating system. The reason why we choose Windows is that the display card only supports the Windows operating system, not Linux. After we had found out the most suitable way to extract in semester one, we pass the data to the point cloud library. Point cloud library is used to reconstruct the images using the data collected before. The images are made up of many points.

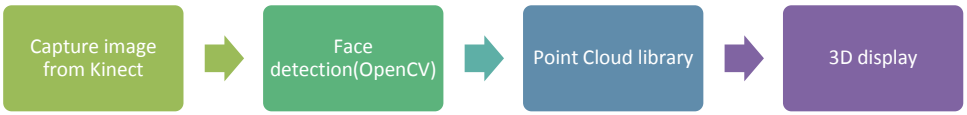


Figure 6.2 Work flow of 2nd semester

At this semester, it is important for us to solve the problems we found in the first semester such as the number of face detected and also the quality of the images. We lastly develop a program that it could detect more than one user faces each time. Special features are also integrated into our program. For our final application, it would detect two players' faces. Interestingly, the faces could be exchanged by clicking the button. Apart from this, the face of the player would also be enlarged.

There are 5 stages in our final year project. In each stage, we make an achievement. On the following, we discuss what we have done in each stage.

Stage 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.

Figure 6.3 Result image

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, thereshould be error detection in the main program. Ther program should exit if there is any error.

To sum up, 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.

Stage 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.

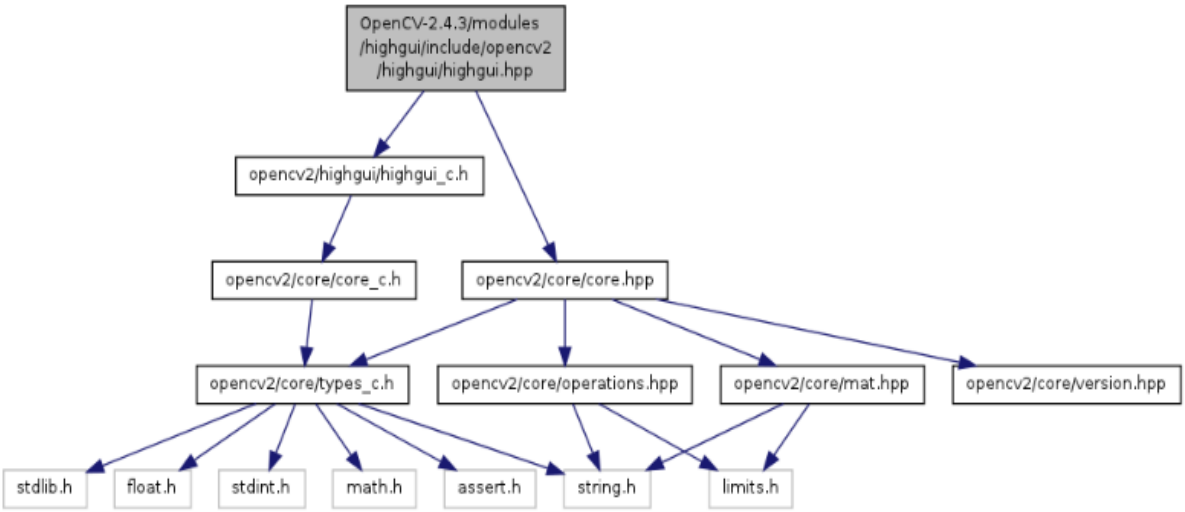


Figure 6.4 Example of library dependence

After including the useful libraries, in stage 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 6.5 Face Detected with circle

At the same time, we combine it with what we have done in stage 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.

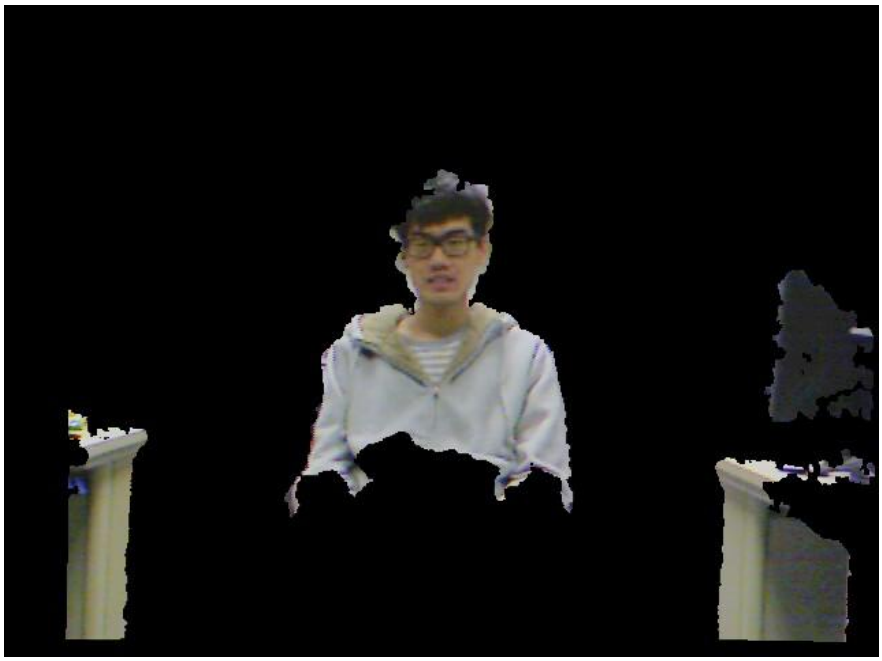


Figure 6.6 Image with certain depth

Figure 6.7 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.

```
//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);


//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);


    }
}
```

Figure 6.8 Mask image without connected component

Figure 6.9 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

```
//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;
    }
}
```

image captured by Kincet.

Figure 6.10 Image without holes

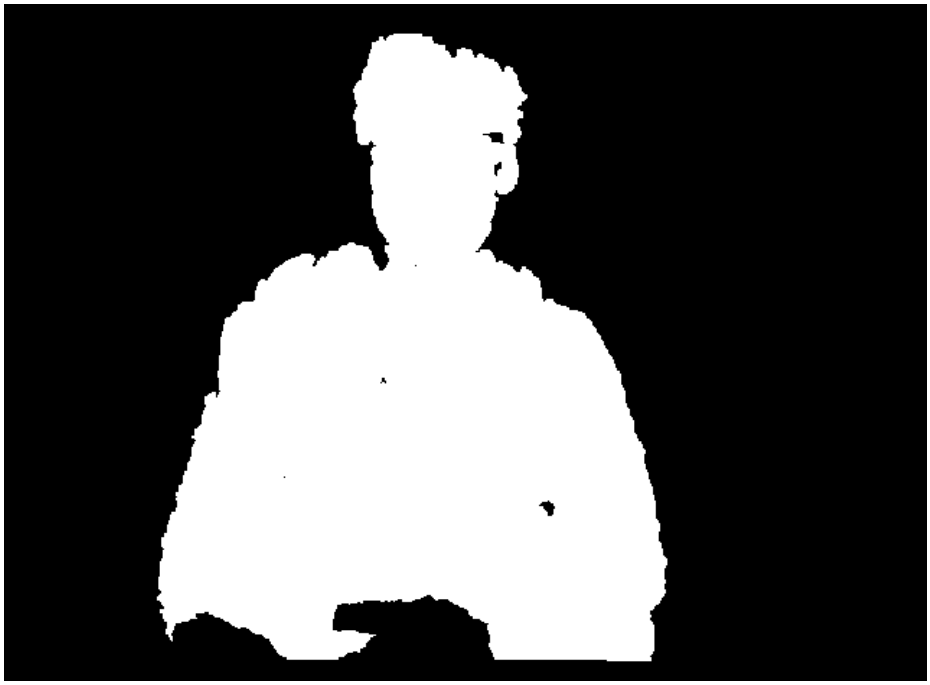


Figure 6.11 Mask image with holes

Figure 6.12 Mask image with holes filled

Stage 3

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

Figure 6.13 Ways of extracting face

We have tried several methods for extracting the face.

1. Chop the face by the raduis of circle

2. Canny edge detection

3. Surface normal

4. Surface normal with depth value

In our main program in stage 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 graphs in order to show the results.

Chop the face by the raduis of circle

In this stage, 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.

```
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);
}
```

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.

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.

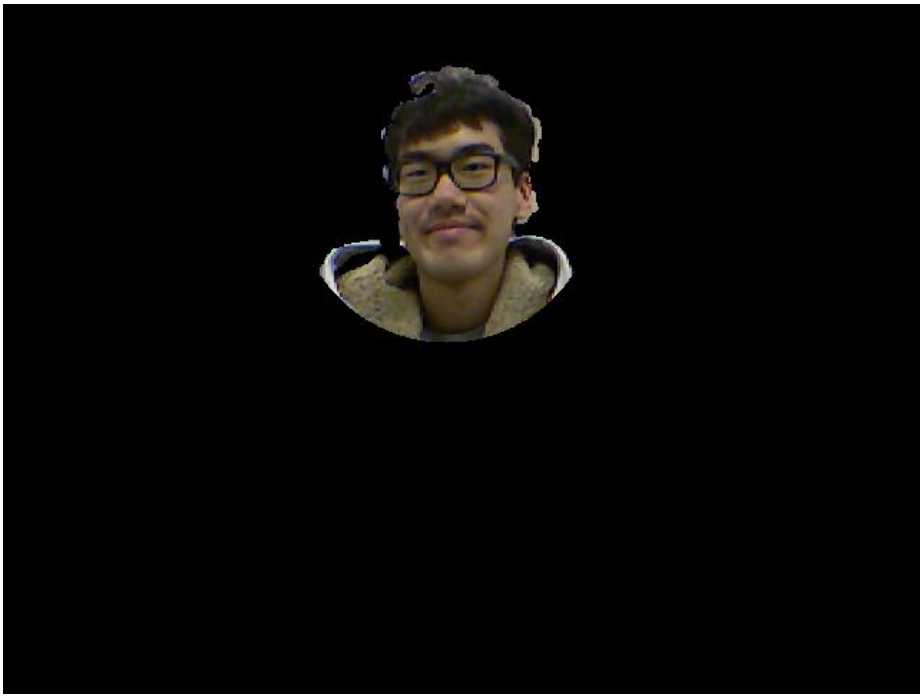


Figure 6.14 Face cut by the radius of circle

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.

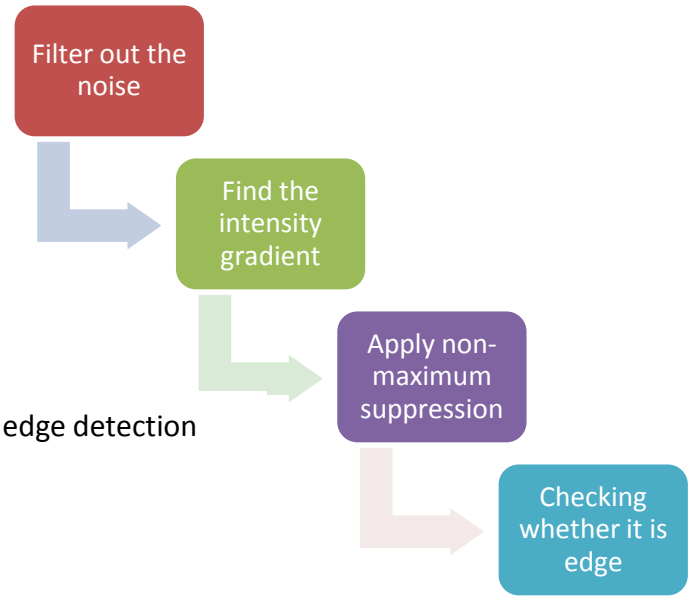


Figure 6.15 Flow of edge detection

The Canny edge detector use the upper and lower thresholods for checking the 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 6.16 Depth edge detection

Figure 6.17 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.

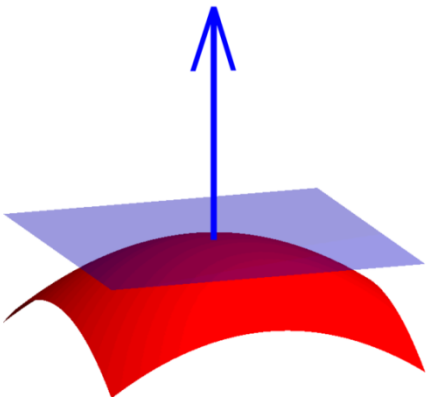


Figure 6.18 Surface normal

The cross product is calculated by this equation

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

The result of the cross product is

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

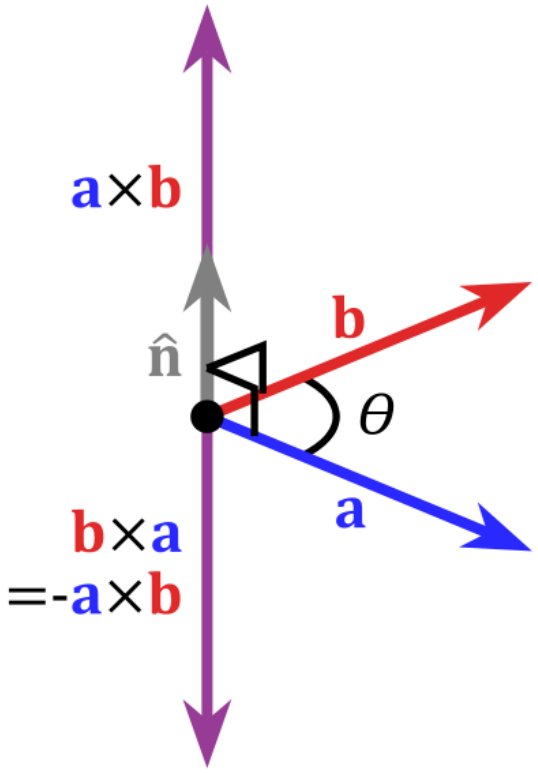


Figure 5.19 Result of cross product

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;
}
```

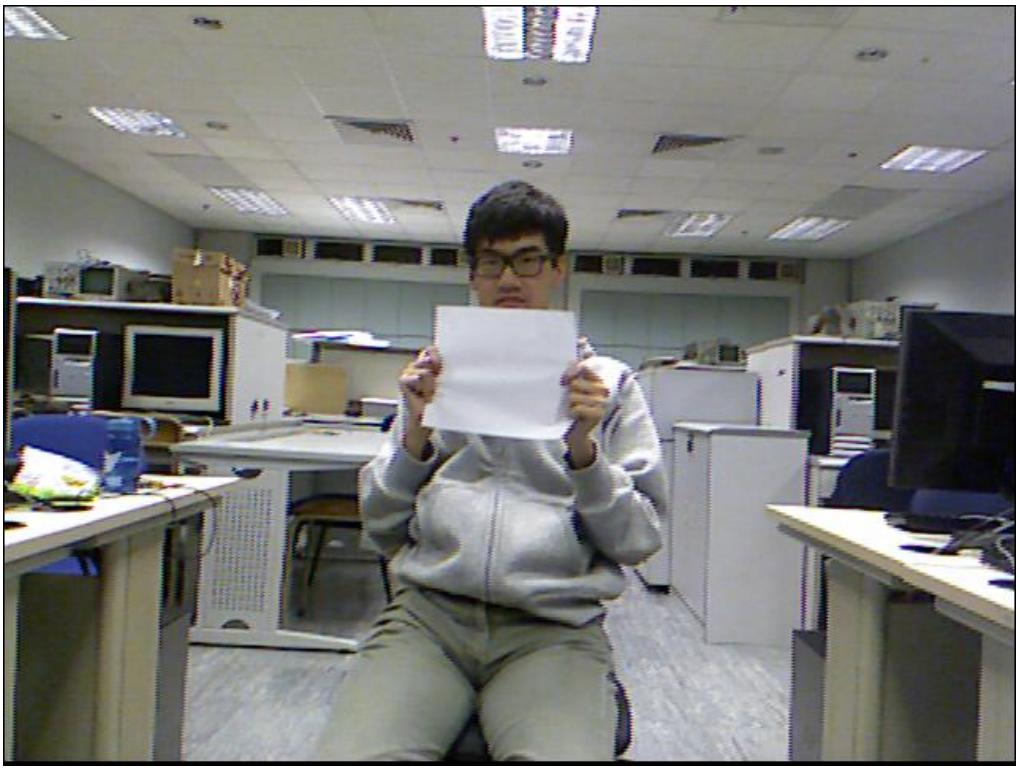


Figure 6.20 Color image to test the surface normal

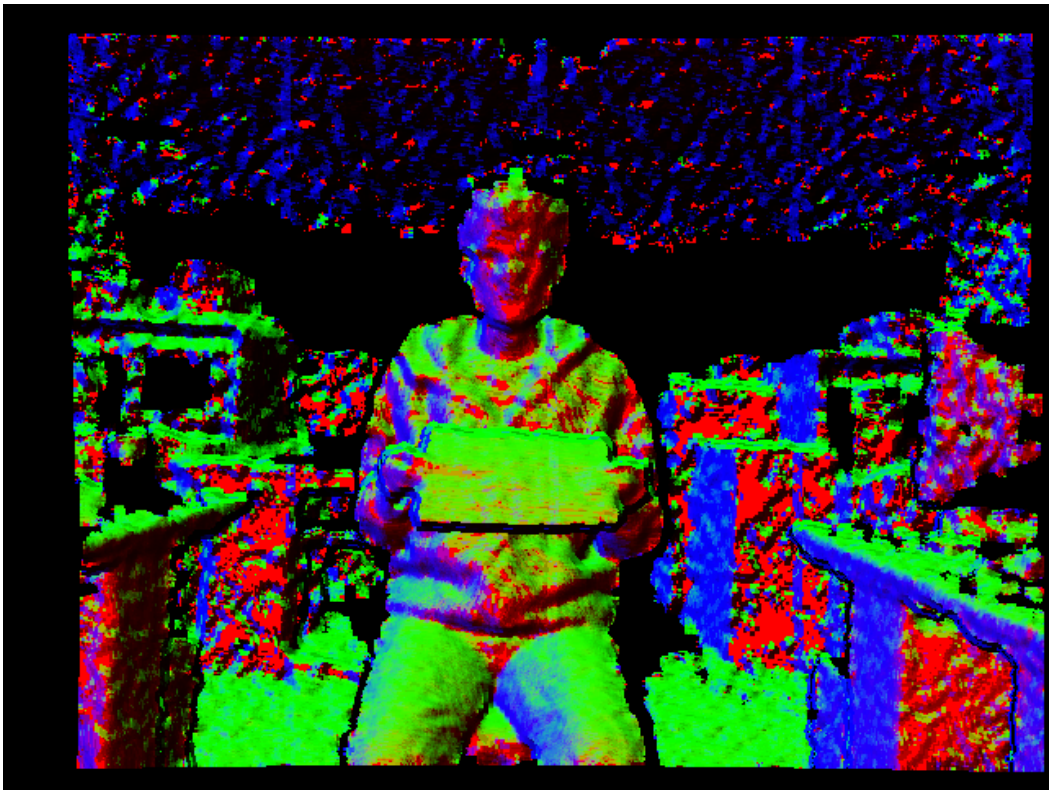


Figure 6.21 Surface normal (down)

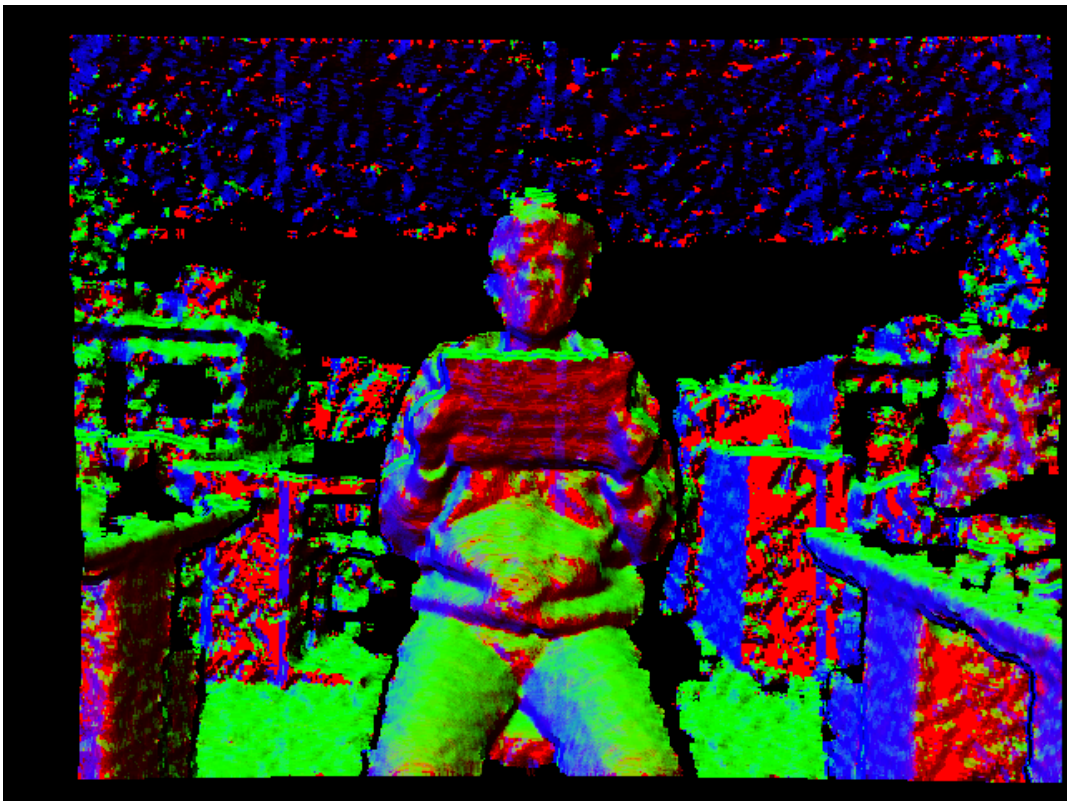


Figure 6.22 Surface normal (up)

Figure 6.23 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);
```

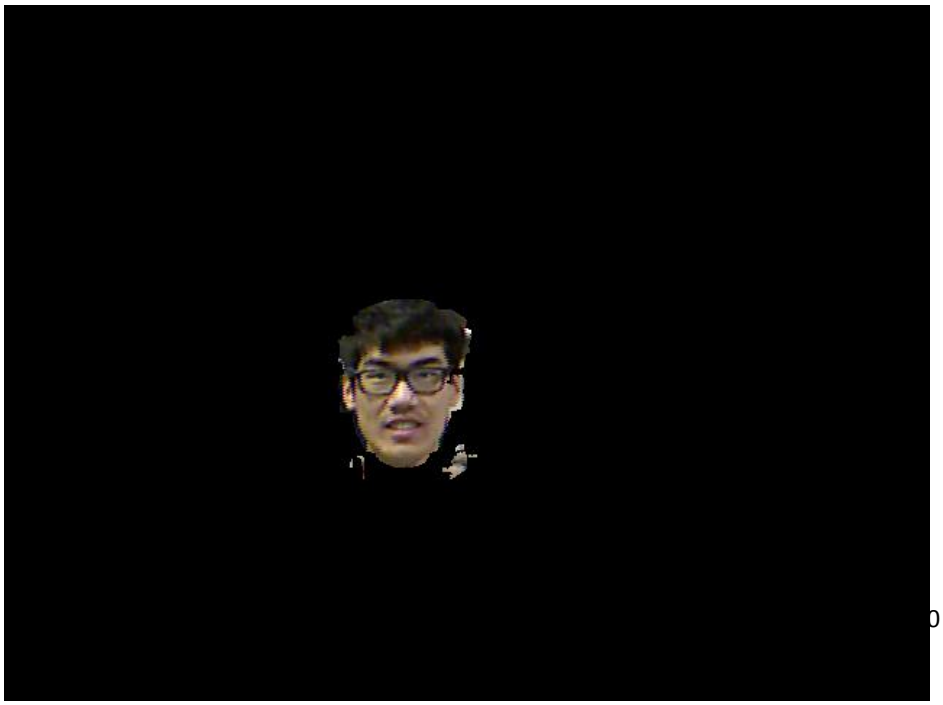


Figure 6.24 Final result of extracting the face

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 6.25 Dance heads video

To summarize what we have done in this stage, 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.

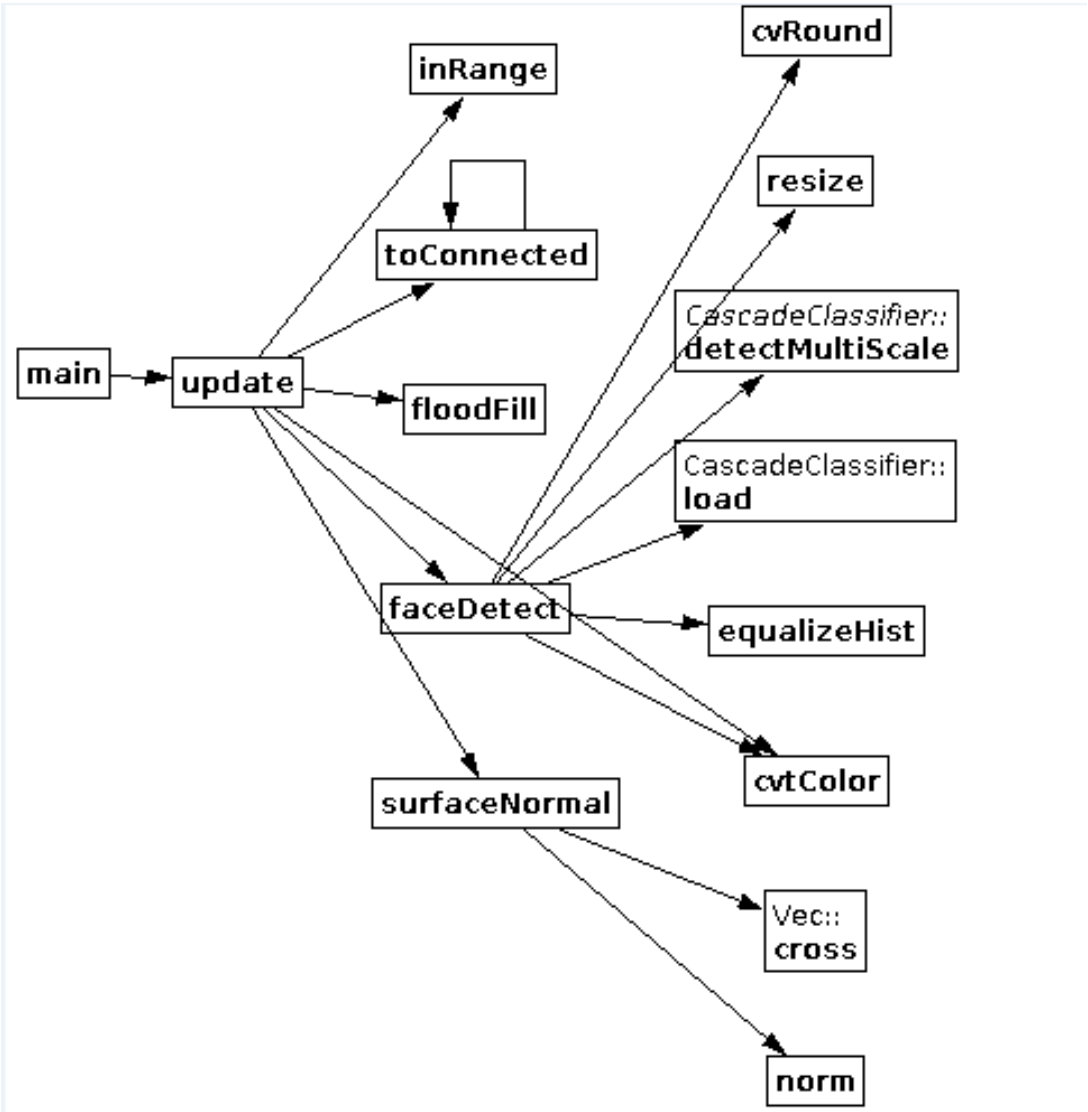


Figure 6.26 UML Diagram

Stage 4

As we have found out the better way for extracting the player's head and the head is superimposed to the video in the previous stage, it is necessary to pass the image to the 3D projector so that 3D images are form and present out.

In this stage, an open source library, Point Cloud Library, is used. Point Cloud is a set of vertices in 3D coordinate system. It means that the set of the points which are measured by the device, Kinect in our project. The 3D data collected is further used for the rendering, animation or visualization, for example.

First of all, we have to understand what point cloud is and how to use point cloud library. PCL provides several modules. After reading and understand the specification of each module, we think that module io and visualization are useful for our application. Before using the class in module visualization and module io, it is important for us to understand the data structure and functions in that module.

After understanding the PCL, it is important for us to improve what we have done before. Since the result of semester one is not perfect, we have to find ways to solve the problem that we encountered before. Low frame rate, the number of detected faces, computation time are the major problems.

Module io

This module provides many classes and function tosave and load writing point cloud library data (PCD) files. Also, it is used to capture the point clouds from different sensing devices.

Two major functions are used. These functions are pcl::io::savePCDFileASCIIandpcl::io::loadPCDFile. To use these two functions, it is important for including the header file, pcl/io/pcd_io.h

pcl::io::savePCDFileASCII

It is used for saving point cloud data to PCD file with specific given cloud format. Point Cloud structures use float values to store RGB data. Using ASCII to store a float, it could introduce variations to smallest bits and alter the data. Then the fix involves switching RGB data to be stored as a packet integer.

pcl::io::loadPCDFile

It is used to load PCD file.

We use these two functions for loading and saving the background for our application.

In the io module, there are a few of classes related to the OpenNI devices, for example, Primesense PSDK, Microsoft Kinect etc. These classes are called***openni_wrapper::OpenNIDevice*** and ***openni_wrapper::DeviceKinect***. In each class, there are many public and protected types and member functions.

However, we have not used these two classes for our application. On the following, we would discuss the reason why we do not choose ***openni_wrapper*** in our project.

To use ***openni_wrapper,***the***pcl/io/openni_camera/openni_device.h*** or ***pcl/io/openni_camera/openni_device_kinect.h*** should be included.

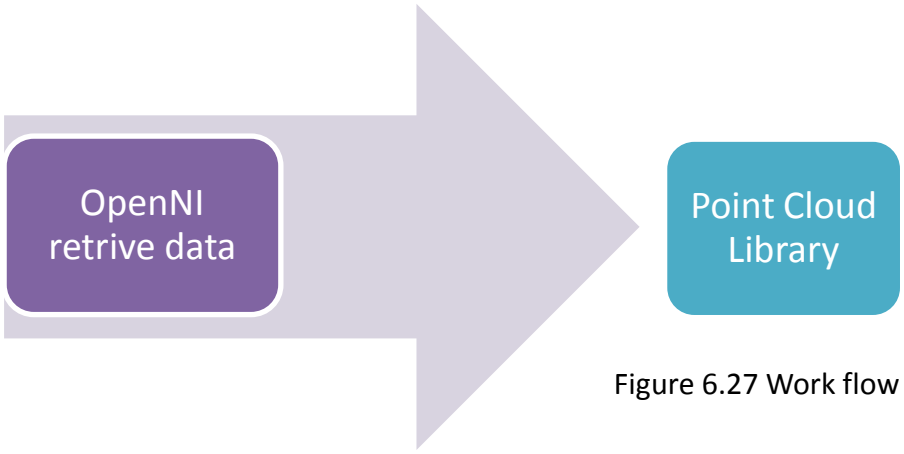


Figure 6.27 Work flow in openni_wrapper

If we use this, after OpenNI retrieve the data by the devices like Kinect, the image data would immediately pass into the PCL. It is not possible for us to use and compute the data retrieved by the OpenNI devices. We only would do computation and use the image data after passing into PLC. By serious consideration, we do not use ***openni_wrapper*** in our project.

Instead of using ***openni_wrapper***, we use the module visualization and io to achieve what we wanted.

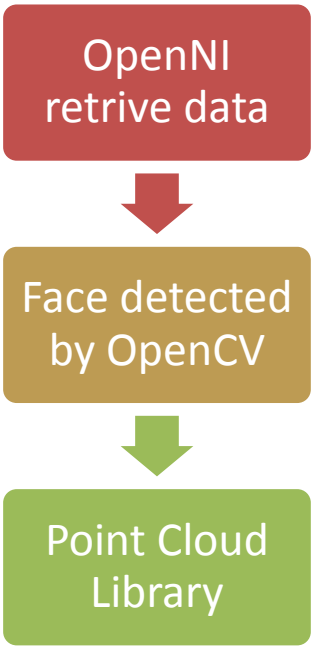


Figure 6.28 Work flow in our program

After capturing the image by OpenNI with Kinect device, the image is then passed to the OpenCVpart.Hence,faces are detected using OpenCV face detection. As the result of semester one, we use the surface normal for extracting the head part. It is because we are satisfied with the result of this approach to extract the head of the player. Finally, the data is passed into the Point Cloud Library for further use and reconstruction of 3D image.

Module visualization

Module visualization provides several classes and functions for rendering or drawing 3D shapes on the screen. It is used for operating the 3D point cloud data. In our application, we use the simple point cloud visualization class. We have to include *pcl/visualization/cloud_viewer.h* and pcl/point_types.h. There are a few public types and different public member functions.

pcl::visualization::CloudViewer::CloudViewer

It would construct a cloud viewer with a window.

pcl::PointXYZRGBStruct

This is a point structure to represent xyz coordinates and RGB. There are several public member functions as shown below.

Public Member Functions		
PointXYZRGB (const_PointXYZRGB&p)		
PointXYZRGB ()		
PointXYZRGB (uint8_t _r, uint8_t _g, uint8_t _b)		
Eigen::Vector3i	getRGBVector3i ()	
const Eigen::Vector3i	getRGBVector3i () const	
Eigen::Vector4i	getRGBVector4i ()	
const Eigen::Vector4i	getRGBVector4i () const	

Result

Firstly, we try out the sample code provided by the official website of PCL.

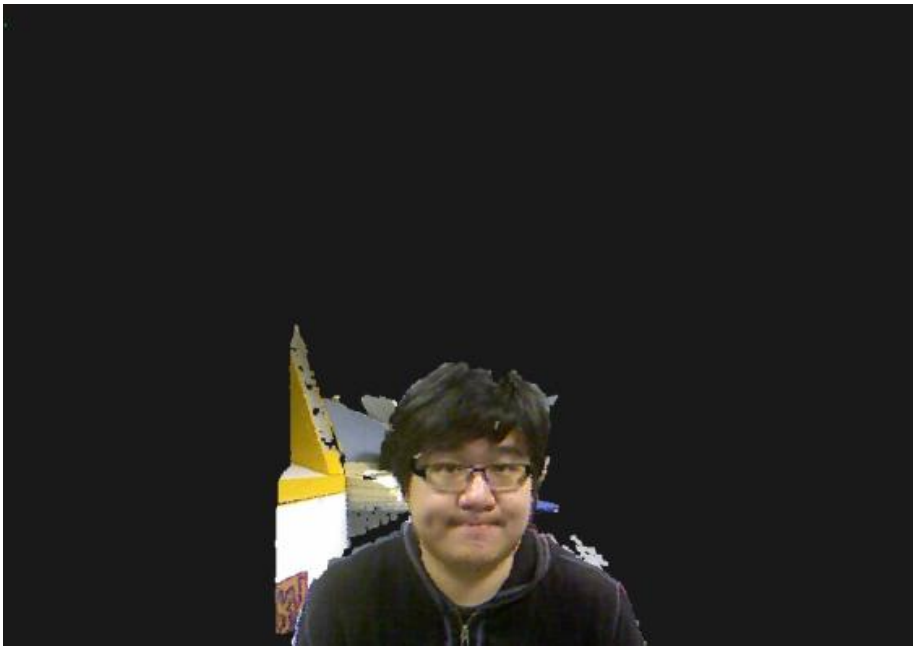


Figure 6.29 Point cloud image (front view)

In this stage, we have hand-on experience using PCL in our program. After fully understanding of PCL, we try to use it in our program and make some graphic for our trial. Module io and visualization are mainly used in our program. The following are the results that we firstly to use PCL with Kinect.

```
pcl::visualization::CloudViewer viewer("testing");
pcl::PointCloud<pcl::PointXYZRGB>::Ptr bg(new pcl::PointCloud<pcl::PointXYZRGB>);
pcl::PointCloud<pcl::PointXYZRGB>::Ptr view(new pcl::PointCloud<pcl::PointXYZRGB>);
```

Figure 6.30 Point cloud image (side view)

Due to the limitations such as the number of face detected in the previous stages, we need to find way in order to let the program detect more than one heads. We consider the situation that two players play our application. Hence, the results in semester should be improved and multi-faces should be detected. As a result, more special features would be integrated into the application so that it would be funny.

From previous results, it is not perfect for extracting the head part of the player using surface normal. If the edge is not cutting very well, we would use the nearest point of the face to estimate better edge for extracting the head.

After extracting the face, we have found that the output frame rate is low.At the same time, we try to tune adjust the scaling factor. In our program, we set ***SCALE 1.1.*** Therefore, the image captured would be reduced to 1.1 times to the original image. By doing so, the speed of face detection would be increased. Hence, the frame rate would certainly be raised.

Stage 5

After having experience in Point cloud Library, we make some special features on our application. We have made several features in our application.

- Load image to the background
- Swap faces between two players
- Enlarge or reduce the face size by clicking buttons
- Move forward or backward to the image of the players
- Blur function to the image

In semester two, we spend most of our time to implement the above features in our problem. It is hoped that our application would be fun for the players. Afterwards, we discuss the above features one by one in details.

Before implementing all the features, we have to enable the keyboard in order to make the change to the results. By typing different specified buttons, a specified feature would be shown to the output.

Using the ***pcl::visualization::CloudViewer*** Class, there is a function called ***pcl::visualization::CloudViewer::registerKeyboardCallback***. Keyboard event is set.

```
if(event.getKeySym()=="k" && event.keyDown ()){
    save = 1;
}
else if(event.getKeySym()=="l" && event.keyDown ()){
    cout << "background loading" << endl;
    if (pcl::io::loadPCDFile<pcl::PointXYZRGB> ("bg.pcd", *bg) == -1) //* load the file
    {
        cout << "Couldn't load background file" << endl;
    }
    else
    {
        cout << "background loaded" << endl;
        load = 1;
    }
}
else if(event.getKeySym()=="v" && event.keyDown ()){
    if (second == 0){
        cout << "second head not found" << endl;
    }
    else{
        if (swapped == 0){
            swapped = 1;
            cout << "head swapped" << endl;
        } else
        {
            swapped = 0;
            cout << "head reset" << endl;
        }
    }
```

By pressing different buttons, different features could be shown.

Load image to the background

As mentioned in the previous stage, we use the module io and visualization to develop our application.

```
if(save == 1){
    cout << "background saving" << endl;
    pcl::io::savePCDFileASCII ("bg.pcd", *view);
    cout << "background saved" << endl;
    save = 0;
}
```

```
else if(event.getKeySym()=="l" && event.keyDown ()){
    cout << "background loading" << endl;
    if (pcl::io::loadPCDFile<pcl::PointXYZRGB> ("bg.pcd", *bg) == -1) //* load the file
    {
        cout << "Couldn't load background file" << endl;
    }
    else
    {
        cout << "background loaded" << endl;
        load = 1;
    }
```

We use the module io to store and load the background. After capturing the background, it changes to PCD file. To save the background image, type “k”. Similarly, when we want to load the background that we stored before, type ”l”. Therefore, using the io module, we would store and load the background easily.

After loading the background, the player could choose whether show the background or not. By typing the “b”, the background would be shown.

```
else if(event.getKeySym()=="b" && event.keyDown ()){
   if (load == 0)
       cout << "background is not loaded" << endl;

   else{
       if (cbg == 0){
           cbg = 1;
           cout << "background changed" << endl;
       } else
       {
           cbg = 0;
           cout << "background reset" << endl;
       }
   }
```

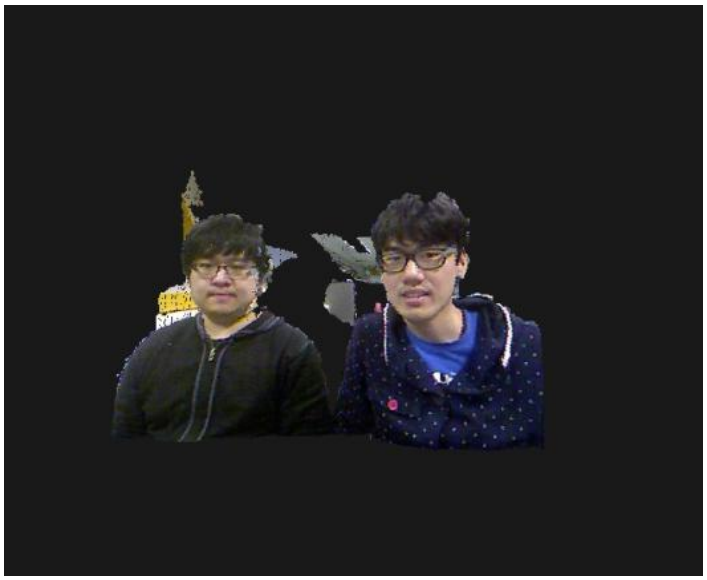


Figure 6.31 Image without background

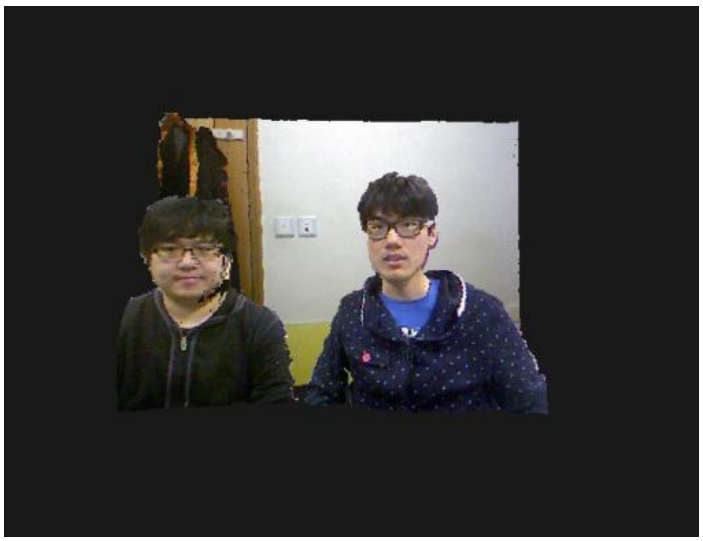


Figure 6.32 Image with background

Swap faces between two players

If the Kinect detects two players' faces, special function would be shown. Two players' faces are swapped between them. That means player 1 face is changed to player 2 face while player 2 face is changed to player 1 face.

However, if the Kinect cannot detect two players' faces, this special function cannot be performed. At the same time, a command window is generated in order to show whether Kinect detect the players' faces. Buffer is set to store the face data for swapping the faces.

```
if (depthValue!=0)                // if depthValue is not NaN
{
    if (cbg == 0 || load ==0){
        if (second == 1 && swapped == 1){
            if(headmask1.at<uchar > (i, j) == 255){
                // Find 3D position respect to rgb frame:
                newPoint.z = depthValue - centerDepth + centerDepth2;
                newPoint.x = (j - centerX + centerX2) ;
                newPoint.y = (i - centerY + centerY2) ;
                newPoint.r = rgbImage.at<cv::Vec3b>(i,j)[2];
                newPoint.g = rgbImage.at<cv::Vec3b>(i,j)[1];
                newPoint.b = rgbImage.at<cv::Vec3b>(i,j)[0];
                if (newPoint.r!=0 || newPoint.g!=0 || newPoint.b!=0) {
                    outputPointcloud->push_back(newPoint);
                    //cout << "in" << endl;
                }
            } else if (headmask2.at<uchar > (i, j) == 255){
                // Find 3D position respect to rgb frame:
                newPoint.z = depthValue + centerDepth - centerDepth2;
                newPoint.x = (j + centerX - centerX2) ;
                newPoint.y = (i + centerY - centerY2) ;
                newPoint.r = rgbImage.at<cv::Vec3b>(i,j)[2];
                newPoint.g = rgbImage.at<cv::Vec3b>(i,j)[1];
                newPoint.b = rgbImage.at<cv::Vec3b>(i,j)[0];
                if (newPoint.r!=0 || newPoint.g!=0 || newPoint.b!=0) {
                    outputPointcloud->push_back(newPoint);
                }
```

Figure 6.33 Image without swapping the faces

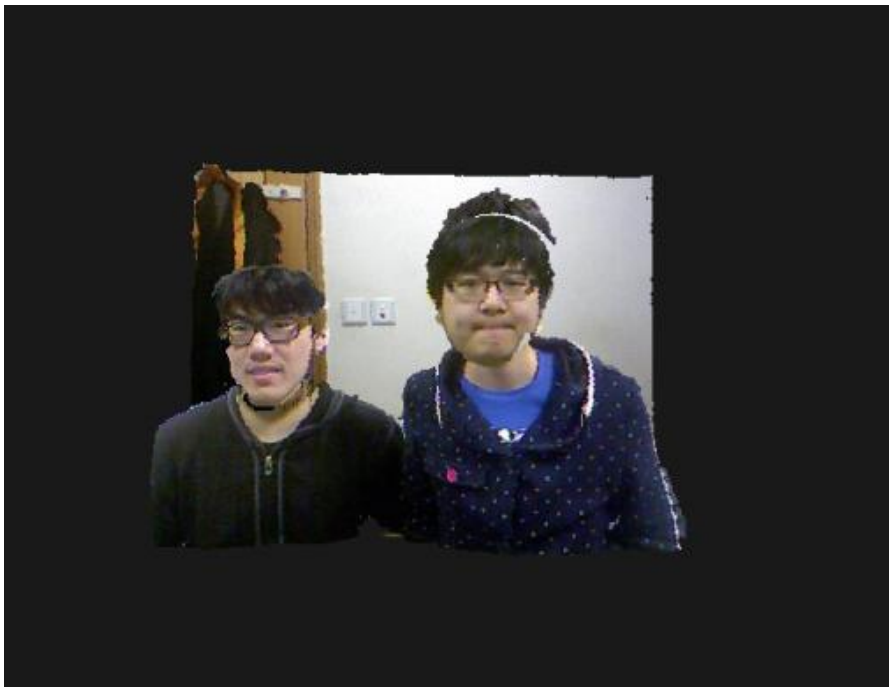


Figure 6.34 Image with swapping the faces

Enlarge or reduce the face size by clicking buttons

By suitable computation, the size of players' faces would be enlarged or reduced. The effect could be shown when the players press the button designated buttons including "," "." ">" "<".

```
else if(event.getKeySym()=="comma" && event.keyDown ()){
    resize1+=0.1;
        cout << "resize1:" << resize1 << endl;
}
else if(event.getKeySym()=="period" && event.keyDown ()){
    resize1-=0.1;
        cout << "resize1:" << resize1 << endl;
}
else if(event.getKeySym()=="less" && event.keyDown ()){
    resize2+=0.1;
        cout << "resize2:" << resize2 << endl;
}
else if(event.getKeySym()=="greater" && event.keyDown ()){
        resize2-=0.1;
        cout << "resize2:" << resize2 << endl;
}
```

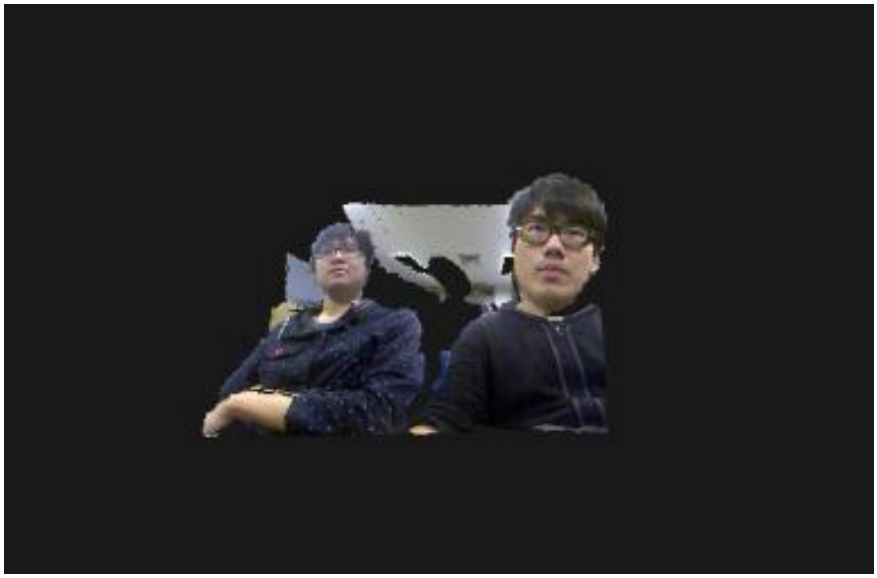


Figure 6.35 Image with swapping enlarged or reduced size of faces

Move forward or backward to the image of the players

```
}else if(event.getKeySym()=="bracketleft" && event.keyDown ()){
    dis1+=100;
        cout << "dis1:" << dis1 << endl;;
}
else if(event.getKeySym()=="bracketright" && event.keyDown ()){
            dis1-=100;
        cout << "dis1:" << dis1 << endl;
}
else if(event.getKeySym()=="braceleft" && event.keyDown ()){
    dis2+=100;
        cout << "dis2:" << dis2 << endl;
}
else if(event.getKeySym()=="braceright" && event.keyDown ()){
            dis2-=100;
        cout << "dis2:" << dis2 << endl;
}
```

Figure 6.36 Players move to the certain distance

From the result, the positions of the players are changed around +/- 100. The players move forward or backward by clicking the designated buttons.

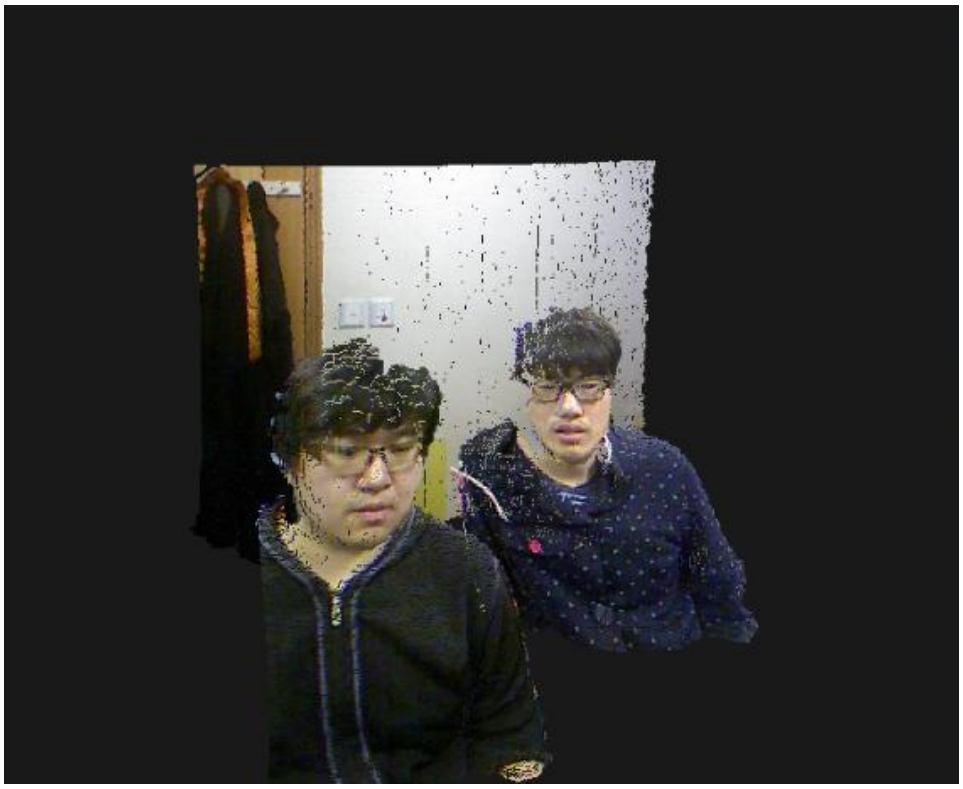


Figure 6.38 Player move forward and backward

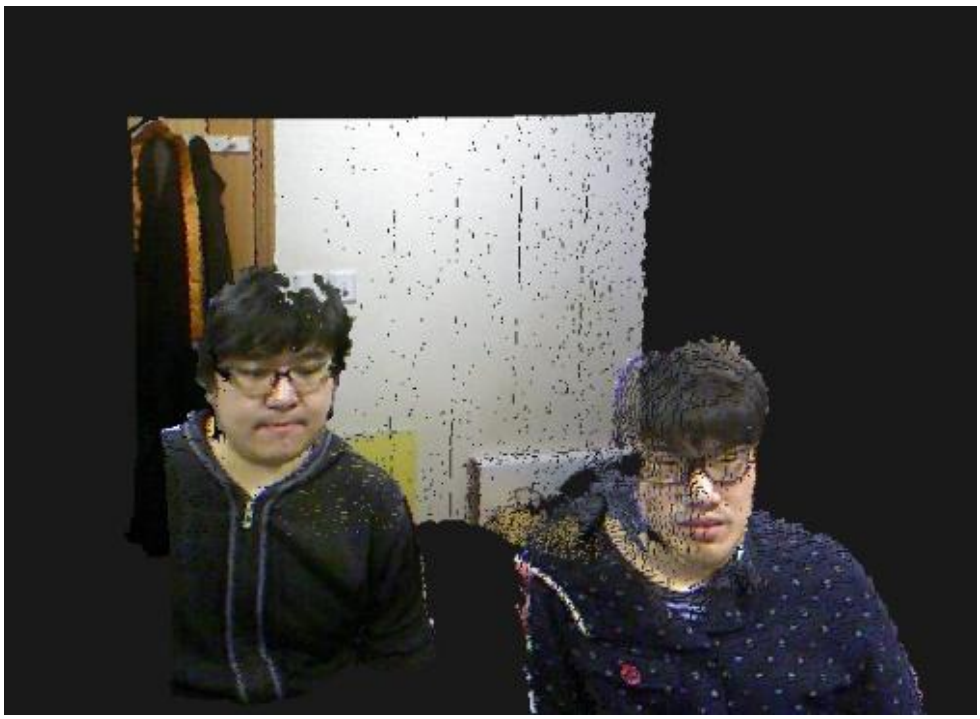


Figure 6.39 Players move forward and backward

Blur function

The results we obtained are not perfect. There are so many black points on the player face. Besides, the edge of the faces is not smooth. There is a broken line in the players' heads. We finally use the blur function to smooth the edge and the image would be improved.

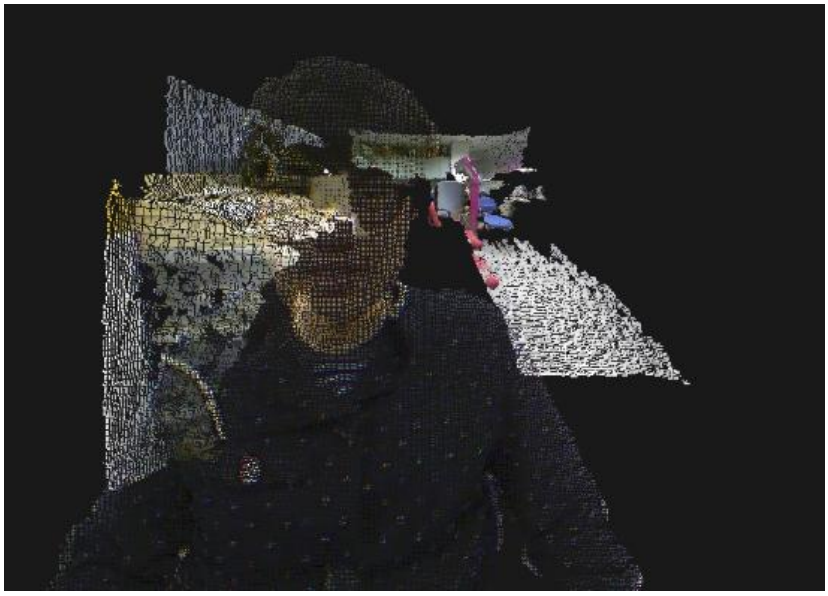


Figure 6.40 Image with more points

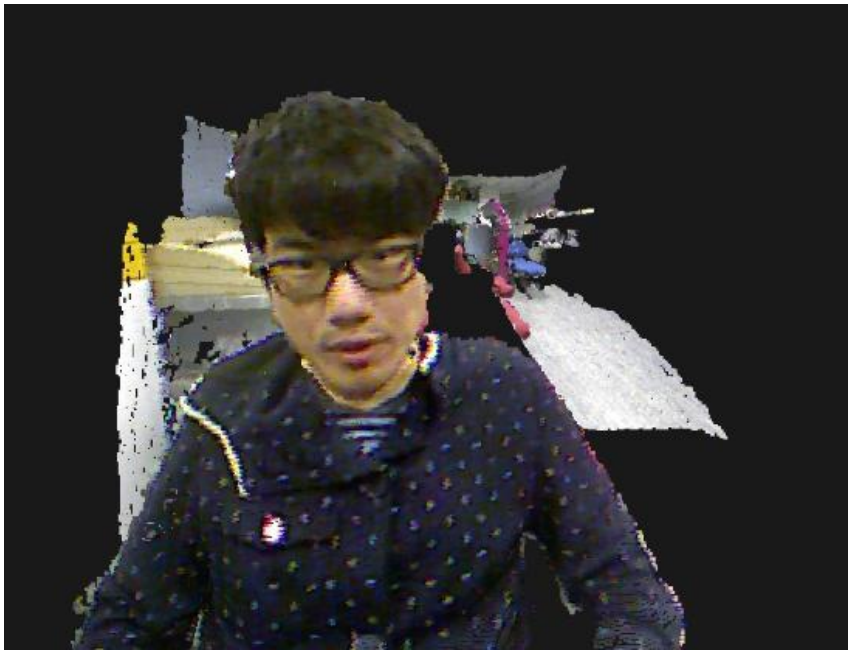


Figure 6.41 Image with less points

3D display (red-blue)

Figure 6.42 Image with 3D display (red-blue)

Keyboard	Features
L	Load the background
K	Save the background
B	Change the background
+	Increase resolution
-	Reduce resolution
[/]	Move player 1 forward and backward
{ / }	Move player 2 forward and backward
, / .	Enlarge and reduce the size of the player 1 face
< / >	Enlarge and reduce the size of the player 2 face

Chapter 7: Contribution of work

Summer

In last academic year, I and my partner, Lai Tai Shing, chose Dance Head with KINECT as our final year project. After that, we start to think about what we problem we need to face and handle. Since it base on Kinect, we started to study the official and unofficial library for Kinect, Kinect SDK and OpenNI.

After a brief study, I found that Kinect SDK is powerful in gesture control. However, it is not very related to our topic and it is only be used in window 7. So, we decide to use the OpenNI library.

After a deeper study, I discover that OnenNI is lack of image processing. So, I tried to combine OpenNI and OpenCV together in order to fulfill our demand.

Semester 1

After a thorough of OpenCV and OpenNI, I set the target of semester 1, recognize the head and divide it for the background and the body. We started to test different method of face detection and find the way to cut off the head from the chin.

However, we had tried a couple of method to find the way to cut off the head but still cannot find a way to do so. After receive the advice from View Lab team, I tried to find the chin by calculate the surface normal of the face andsucceed to cut out the head finally.

Moreover, I also implement connected component to cut the people from background for better result and future uses.

Semester 2

At the beginning of semester 2, I reviewed the program to modify the program to support multi-face detection and optimized the cutting edge of the head. After that, I tried to merge the 2D image and the depth value into a 3D image. As a result, I started to study point cloud library and tried to merge the previous result and point cloud library.

In order to make the program more interesting, I added some effects to the application in order to make the program more interesting after succeed merge them together,

Conclusion

In this final year project, I had tried a lot to finish a whole project. I had met a lot of difficulties. I tried a lot of method to solve it and learnt a lot in image processing where I never learnt before. And I have a great cooperation with my partner to solve the difficulties we met.

Chapter 8: Conclusion

To conclude, we have achieved several goals in our final year project.

- Study the Kinect SDK and other open source library such as OpenNI, OpenCVused in our project.
- Design and implement an algorithm to extract the “Head” part using Kinect camera.
- Rendering the image captured by Kinect.
- Study Point Cloud library used for reconstruct the 3D point image.
- Design some special features for the application like swap two players’ faces, enlarge or reduce the size of head etc
- Load the image to the background.

In the first term, we mainly focused on studying the Kinect SDK. But we di not use it finally, instead of it, we use OpenNI to retrieve the image data. Meanwhile, Kinect is our major device for our project. We also investigated how to program using OpenNI in Linux operating system. In addition, we have tried several ways to extract the head based on the face detection provided in OpenCV.

To demonstrate the progress, we make video which the player's head is superimposed to the video. Hence, Dance Head effect is achieved.

In the second term, we tried to use Point Cloud Library in our application. After retrieving data through OpenNI with Kinect camera, PCL is used to reconstruct the 3D image data for further use. In our application, we also design different special features such as swapping and enlarging the faces. In order to fulfil the requirement of the 3D projector and display card, we have chosen Windows as our platform.

It is interesting to use Kinect and PCL to develop application. PCL is a powerful library that it could generate 3D image data. To make the application interesting, several special features are designed.

Chapter 9: Progress and Difficulties

Period	Progress
Summer2012	Research on topic Refine on topic
September 2012	Study Kinect SDK Study OpenNI and OpenCV Try face detection
October 2012	Start program with OpenNI Find different ways for extracting head
November 2012	Modify the program Find the best way to extract head Prepare first term report
December 2012	Final testing to the result Evaluate the result
January 2013	Research on 3D display Study Point Cloud Library
February 2013	Code with PCL Design features on our application
March 2013	Implement the special features Test all the features
April 2013	Refine the program Prepare second term report

Difficulties and challenges

Lack of knowledge in image processing

In this project, ther are many images and data we need to handle. However ,both of us do not have experience on image processing. Doubltless, it is quite difficult for us to start our project. Moreover, this is our first time to use the open source library such as Point Cloud Library, OpenNI, OpenCV. Therefore, we spent most of our time to study the source library.

Limited resources and documentation

For our project, there is no similar application developed by other using Kinect, OpenNI, OpenCV and PCL. We have to fully understand about the documentation, reference and sample code in order to code out program. Also, we have to intergate the open source librayr together so that our goals would be achieved finally.

Side view problem

To detect face, harr cascade method in OpenCV is used. It is not possible to detect the side view of the face. The problem only detects the front face of the players. If the players swing their heads, their heads would not be detected. Hence, the swap function could not be run.

Kinect limitation

There are some limitations on Kinect. There may be some errors 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.These would affect the results of extracing the head.

Quality of the output image

From the results, we find that the quality of the output image is imperfect. For example, there are some points such as the glasses that cannot be detected. Apart from this, many black points are shown on the image. We have to use the blur function as mentioned before to decrease the resolution so that a better image is obtained.

Testing SDK and use suitable libraries

At the beginning, we test the Kinect SDK. It is quite difficult for us since we have no idea and experience in using this. Firstly, it is very complicated when we study the SDK and understand how it works. Finally, we have chosen OpenNI, OpenCV and Point Cloud Library in our application. Through the process, we tried to compile sample code and fully understand it. At the same time, we have to solve many linking problem about integrating different libraries.

Low frame rate

Due to the low frame rate in first term, we adjust the scale factor to 1.1. The speed of face detection is faster. The frame rate is better than before, however, we are still not satisfied at all.

Evaluation

We have some review and reflection of our application. We summarize them on the following. Time is limited for our project so we have solved all problems that we encountered.

Improve quality of the output image

As mentioned before, quality of the image is not very well. It is important to find a solution. Then the results of our program would be better.

Face detection for more players

As our program could detect up to two players now, the program would be enhanced. Therefore, more players would join to our application each time.

User interface

To make our user-friendly, we would design a better interface for the user to attract more people playing it. Furthermore, we would design our program for different platforms, for example, develop for Android/IOS application.

More special features

To make the application more interesting, we should design more and more features using the image data. For example, move the head to other position according to the players.

Improve frame rate

As we have adjusted the scale factor in order to speed up the speed of face detection. Although the frame rate is better compared to the semester one result, the frame rate is still low that we do not satisfied at all. The frame rate should be improved so that the application would run smoothly.

Display with 3D projector

We develop our program in Windows platform in order to fulfil the requirement of the 3D projector and display card. However, time is limited for our project. We spend most of time in reading the document about PCL and designing the special features. We have no time to use and have a try on the 3D projector.

Chapter 10: 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. Besides, he reminds us how to make a good presentation and report.

In addition, we would like to thank the research team Mr.Edward Yau and Mr. Un tze Lung in VIEW Lab. He always provides some valuable advice and technical support when we faced problem. They also give us lots of brilliant ideas and facilities for our project. Without their support, this project would not be so successful.

Thanks again that they give us opportunity to do this project. We certainly learn a lot from them through the project process.

Chapter 11: 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

http://fossies.org/dox/OpenCV-2.4.4/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

[19] Point Cloud Library

http://pointclouds.org/

[20] NVIDIA 3D Vision Pro

http://www.nvidia.com/object/3d-vision-professional-users.html

[21] 3D Computer Graphic

. http://en.wikipedia.org/wiki/3D_computer_graphics

[22] Image processing

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