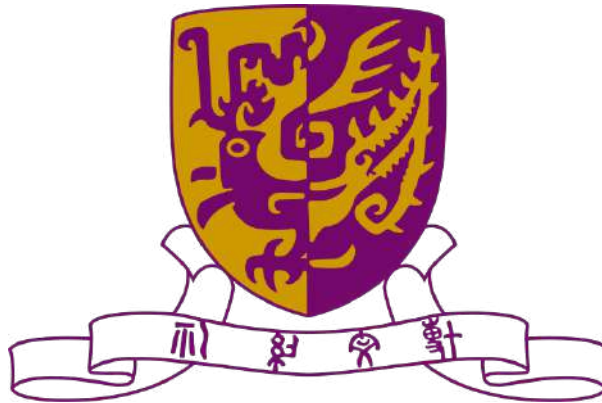


Department of Computer Science and Engineering  
The Chinese University of Hong Kong



2011-2012  
Final Year Project Report (2<sup>nd</sup> Term)

**LYU1104**  
**iSurveillance – Interactive Surveillance Video Query Apps for iPad**

Wong Ka Yan  
1009601432  
kywong9@cse.cuhk.edu.hk

Supervised by  
Prof. Michael R. Lyu

# Abstract

Our Final Year Project is called iSurveillance. iSurveillance is an mobile application developed for iOS platform. Our product aims at being an auxiliary tool for security guards; to construct some function modules to assist their daily jobs, make their jobs become easier.

This report covers our study and progress during the development. It begins with an introduction about our project; to have a brief understanding of our project background; and then it will explain the motivation behinds our idea --- like where do our FYP idea come from?

A case study is followed after that. A case study is carried out as to know more about security works and have a better understanding of the nature of this field.

After that, we would have a system review; to introduce what features our product has and the mechanism behinds. A system architecture overview will also be given to explain how our system is construed during the design phase.

And for the user interface implementation, chapter 5 is to report how our UI was designed and what concerns have been taken into consideration in between.

Last but not least, we are going to share what difficulties we encountered during the development and how did we solve it. And finally to look ahead of the future, to see what could be further improved if we still have time for our project.

# Table of Contents

<b>ABSTRACT .....</b>	<b>1</b>
<b>TABLE OF CONTENTS .....</b>	<b>2</b>
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>4</b>
1.1 OVERVIEW.....	4
1.2 BACKGROUND .....	5
1.3 STUDYING ON CURRENT SURVEILLANCE SYSTEM .....	6
1.3.1 CURRENT SYSTEM .....	6
1.3.2 PROBLEMS .....	8
1.4 PROPOSED SYSTEM.....	9
1.4.1 ADVANTAGE.....	11
1.4.2 SIMULATED EXAMPLE.....	12
1.4.3 DEVELOPMENT PLATFORM AND DEVICE.....	13
<b>CHAPTER 2 CASE STUDY ON CURRENT SURVEILLANCE SYSTEM.....</b>	<b>19</b>
2.1 OVERVIEW.....	19
2.2 MOTOROLA'S SOLUTION .....	20
2.2.1 ARCHITECTURE.....	21
2.2.2 COMPARISON.....	23
<b>CHAPTER 3 SYSTEM REVIEW.....</b>	<b>30</b>
3.1 OVERVIEW.....	30
3.2 IMPLEMENTATION .....	32
3.3 API STUDY .....	49
3.3.1 CORE-PLOT .....	51
3.3.2 JSON.....	57
3.3.3 ZXING .....	59
<b>CHAPTER 4 SYSTEM DESIGN .....</b>	<b>63</b>
4.1 OVERVIEW.....	63
4.2 CONVENTION .....	64
4.3 SYSTEM ARCHITECTURE DESIGN .....	65
4.3.1 SYSTEM BLOCK DIAGRAM .....	65
4.3.2 SYSTEM COMPONENT DIAGRAM .....	82
4.3.3 SYSTEM SEQUENCE DIAGRAM.....	83
<b>CHAPTER 5 USER INTERFACE DESIGN .....</b>	<b>84</b>
5.1 OVERVIEW.....	84
5.2 USER INTERFACE DESIGN PROCESS .....	85
5.2.1 CREATING A DESIGN DOCUMENT .....	85
5.2.2 DIVING INTO THE CODE OTHER DETAILS CONCERNS .....	87
5.2.3 TUNING PROCESS.....	93
5.3 USER INTERFACE DESIGN TOOLS.....	94
5.4 BOOKS REFERENCE .....	95
5.5 CHAPTER SUMMARY .....	97

<b>CHAPTER 6 LIMITATIONS AND DIFFICULTIES .....</b>	<b>98</b>
<b>6.1 OVERVIEW.....</b>	<b>98</b>
<b>6.2 MODULE PART - RECORDED VIDEO PLAYBACK MODULE &amp; MULTIPLE-CAM TRACKING MODULE.....</b>	<b>99</b>
<b>6.3 BACKEND LIMITATIONS.....</b>	<b>108</b>
<b>6.4 FRONTEND LIMITATIONS .....</b>	<b>109</b>
<b>CHAPTER 7 ROOM FOR IMPROVEMENT .....</b>	<b>114</b>
<b>7.1 OVERVIEW.....</b>	<b>114</b>
<b>7.2 ROI MODULE.....</b>	<b>115</b>
<b>7.3 CHATBOX MODULE .....</b>	<b>116</b>
<b>7.4 LIVE-STREAM VIDEO MODULE WITH RTSP .....</b>	<b>118</b>
<b>7.5 DRAG AND DROP IMAGE INQUIRE MODULE .....</b>	<b>119</b>
<b>CHAPTER 8 CONCLUSION .....</b>	<b>120</b>
<b>CHAPTER 9 ACKNOWLEDGMENT .....</b>	<b>121</b>
<b>CHAPTER 10 CONTRIBUTION OF WORK.....</b>	<b>122</b>
<b>CHAPTER 11 REFERENCES .....</b>	<b>123</b>

# Chapter 1 Introduction

## 1.1 Overview

In this chapter, we are going to see the importance of safeguarding, how does it play a crucial role among our daily life; and after that we will have a brief study on the current surveillance systems to discuss the deficiencies and limitations current systems have. We then turn to propose our FYP idea, discuss how our proposed system is better than the current one. We would use a simulated example to explain how does our system work better and mechanism behinds. Finally, we will talk about the reason why we decide to develop our application on iOS platform and the advantages behind it.

## 1.2 Background

### The Importance of Safeguarding

When it comes to a building management, safeguarding must come to a great concern and play a crucial role among it. As a matter of fact, security control is of paramount importance and become indispensable to our daily life; not only does it protect our property and wealth, but also give us a place of serenity (where we can no longer worry about thieves and burglars) so that we can focus on our work.

And now, you can see that how important security is for to us; the real question we concern here is - the one who take all those responsibilities. However, in most reality case among our daily life, all these important work fall into one hand only - our security guard, making our security guards bear a great burden. They are in a great responsibility of safeguarding our wealth and life and hence have to be very careful at every single moment. Unfortunately the fact is, most of the securing work is monotonous and mechanical (like watching a surveillance video over a front desk 12 hours like), it is not easy for them to sharpen their vigilance all time long and they maybe overlook the details when there is a burglary or theft really happen. Hence there is a potential security breach we can't risk.

With today's advanced Information technology, you can see that I.T. has already been deployed in various fields as an auxiliary tool to lift down mankind workload. So, when it comes to security field; we are asking the same question, how can we make a great use of I.T. to help a security guard? What technology can we use to facilitate the work for security personnel? This comes to our FYP topic - iSurveillance; a project aims to develop a mobile application so as to facilitate the work of a security guard. After all, we believe that the ultimate I.T. goal is to make a better life for human.

## 1.3 Studying on Current Surveillance System

In the following, we are going to investigate the current surveillance system to have a grasp of how does current systems operate and to see what are the disadvantages within.

### 1.3.1 Current System

#### What does the Current System look like?

The most common and traditional way is 24-hours CCTV-Monitoring (manly based) and a fixed patrol schedule. For that, the security company would hire a security personal to sit over on a front desk for monitoring the surveillance cam during his/her shift (which usually at least 10 hours long). As we have discussed before, this method is very inefficient; cause most of the surveillance video is monotonous and boring, they can't heighten their vigilance all day long and easy to overlook and not notice the details when there is really a burglary or theft happen.



*Figure 1.1 Traditional Security Control Room*

For a better security company (which is usually at a bigger scale, like a security team for an entire business building, a mall or a casino), they would have a better framework of labor division and adopt some software programs for their security control. For instance, the security framework is divided into frontend patrolling team and backend controlling team. The frontend security personnel are mainly for patrolling while the backend security control central is for CCTV-Monitoring and communication.



**Figure 1.2 Professional Security Control Room**

However, this kind of security deployment always involves lots of resources and money, not that every building can afford to. And still, this method is still in a passive way. When there is really a burglary or theft happen; a prompt and decisive action is required, a division framework like this may result in a communication problem; the communication may not give a clear and decisive order to the fronted patrolling in time.



### 1.3.2 Problems

#### **Deficiency of Current Systems**

In a nutshell, we can see that there are some deficiencies from the current systems.

For a small-scale and traditional security company, the way that they adopt a 24-hours (manly based) CCTV-surveillance is not efficient enough; safety guards may easy to relax their vigilance due to the mechanicalness of video watching and hence poses a security beach.

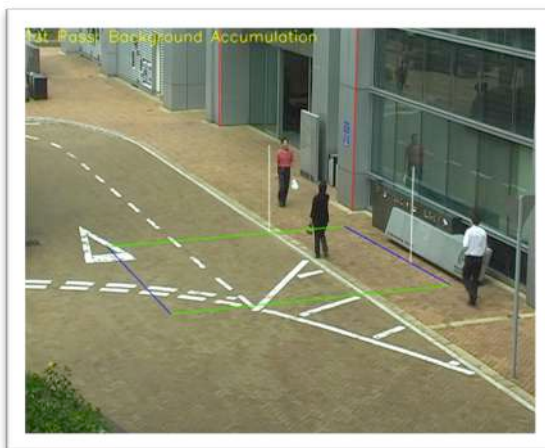
And, for the second security system mentioned above, the security deployment that a bigger security company adopted always involves lots of resources and money; not that every building can afford to. It is so hard to popularize. And still it is in a passive way; when there is really a burglary or theft happen; a prompt and decisive action cannot be achieved in a short time, because there is a potential communication problem between two ends.

## 1.4 Proposed System

### What will we Do and How?

Given the deficiencies of current systems we have investigated, we are going to see if there is a way we can do to improve the situation.

Fortunately, View Lab (The VIEW Technologies Laboratory), which is partly founded by our FYP professor Michael R. Lyu, has developed a backend visual computer program which analyze a recorded video based on the photo chromatography and algorithm input to compute and analyze the data required, for instance, to calculate how many people has occurred at a particular point within the video.



**Figure 1.3 Backend Program**

And, we thought it would be a good idea to co-operate with this backend system and put it into practice. So, we decide that our FYP aim is to construct a frontend software application that fully works with this visual program so as to make it useful to a security company.

And after the characteristic and properties of security works have been taken into consideration, we decided to orientate our FYP topic into a mobile application because of its' popularity, mobility, ease of usage and user-friendliness; it all fits to the field of security works. In other word, it is absolutely suitable for a security guard to use it.

For a first glance, we would like our product to have the following functions:

1. Live streaming playback
2. Video analysis (for searching) with time line and optimized thumbnail
3. Region of Interest for a quick video searching
4. Multiple cams for target tacking
5. Query searching by an Image

The details of the functions mentioned above will be described in the next chapter.

### 1.4.1 Advantage

Then, we are about to evaluate the benefit of our proposed system and give a simulated example as a demonstration for a better understanding.

#### **Why need us?**

As discussed before, it is clear that in the most reality case, whenever there is a security alert, a prompt and decisive action has to been taken at a very soon time. And with our product functions facilitating the work, a security guard can easily get what he/she wants and hence make a quick and right decision in time. For instance, when there is a security alert, the security personal will be immediately noticed through our system, and he/she can then do a quick video searching through our application to find out the suspect and the relevant information; what is more, to even facilitate the video searching process, he/she can do a Region of Interest search to further narrow the range down.

In the other aspect, when it comes to the cost, it is certainly cheaper than the big-scale security deployment we have mentioned above.

## 1.4.2 Simulated Example

For a bettering understanding of our system, please take a look of the following example.

Imagine you are a security guard and there has just happened a burglary case; you are informed through our software application. After you have arrived to the event location, you can then use our application to list out how many cams that are nearby and may capture the image you want. You can then take a quick playback from the interested camera and do a video analyzing program. The video analyzing program will give you back a scatter chart and a thumbnail array that representing how many people has appeared and the time they appeared during the specific period.

So, you can make use of those results to do a primary searching. Like, you can tap on the thumbnail of the pedestrians and the relevant surveillance video will be shown automatically on the video player.

After you have spotted out the suspect or you want a more refined search, you can undergo “Region of Interest searching”; you just have to draw a square box on the screen to indicate your interested area, and our system will then filter the result and give back the only required information you want.

Furthermore, if you want for an assistance or share the information to your colleague, you can use our communication box and just drag the thumbnail to the message box.

### 1.4.3 Development Platform and Device

#### Why Mobile Platform?

We decided to orientate our FYP topic into a mobile platform because we have to consider the nature of securing work.

Given the characteristic and properties of security works, our application have to be popular, portable, easy to use, user-friendly and with lots of API and libraries supported.

We decided to develop our software application on a mobile platform because we think that the current mobile platform is mature enough and fits all the requirements we needs.



*Figure 1.4 iPad2*

## Why iOS?

iOS is a popular mobile operating system nowadays and even create the iProducts Mania. According to the Nielsen survey, Apple owns the second largest market-share on the smart phone market, following the Android developed by Google Inc. And the number is expected to be further increased in the future.

The statistic mentioned above doesn't include iPad and other iProducts. Therefore the total number of iOS distribution is much higher than the statistic.

Besides, when it comes to iOS' development environment, there are also many good things to talk about.

The development environment of iOS is very mature. The IDE, Xcode has been developed for a long time since the releasing of Mac OS X; and the base frameworks, like Cocoa Touch, Cocoa Animation, Cocoa Media, Cocoa Quartz are very useful and powerful; easy to use and implement. Besides, the documentation supported by Apple Inc. is well beyond sufficient. All the frameworks and technical details are documented on the paper; some of them are even recorded in a video.

Moreover, what attracted us most is, there are a lot of 3rd party libraries that have been developed to support iOS. Most of them are open-source, that means, it is free of charge.



*Figure 1.5 iOS5 Logo*

## Why iPad?

Choosing iPad as our FYP platform is not a rush decision, we have taken a holistic and comprehensive consideration for each options of choice. The following table summarizes the comparison.

Gizmodo's Guide to Upcoming Slates						
	Apple iPad	HP Slate	JooJoo	Notion Ink Adam	Dell Mini 5	Archos 7 Android
Screen	9.7-inch 4:3 IPS LCD (1024x768)	10-inch LCD	12-inch LCD (1366x768)	10-inch transfective Pixel Qi LCD (1024x600)	5-inch LCD	7-inch LCD (800x480)
OS	iPhone OS 3.2	Windows 7	JooJoo OS	Android	Android	Android
Browser	Mobile Safari	Whichever you want	WebKit-based	Android browser	Android browser	Android browser
Adobe Flash Support	No	Yes	Yes	No	No	No
Ebook Client	Yes	Yes	No	Yes	Yes	Yes
Apps	App Store, with new iPad apps	HP TouchSmart, Windows apps	Web-based, more like shortcuts	Android Market	Android Market	Android Market
Multitasking	No	Yes	No	Yes	Yes	Yes
Multitouch	Yes	Yes	Yes	Yes	Yes	Yes
Camera	No	Yes	Yes, 1.3MP	Yes, 3MP	5MP; MAYBE second front-facing cam	Yes
I/O	30-pin; camera-only USB add-on	USB	USB	USB, HDMI	USB	USB
Processor	1GHz Apple A4	MAYBE: Intel z540 Atom 1.86GHz*	1.6GHz Atom with Nvidia Ion	Nvidia Tegra 2 (up to 1GHz)	Qualcomm 1GHz Snapdragon	MAYBE: ARM CortexTM-A8 600MHz
Storage	16, 32 or 64GB	MAYBE: 32GB, 64GB*	4GB	16 or 32GB	Unknown	8GB
Expandable Memory	Optional SD dongle MAY add storage	Not according to pics	Probably not	Yes, SD	Dual microSD slots, possibly internal	Yes, MicroSD
Wireless	Wi-Fi, 3G (HSPA/GSM), Bluetooth	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth	Wi-Fi, 3G (HSPA/GSM), Bluetooth	Wi-Fi, 3G (HSPA/GSM), Bluetooth	Wi-Fi, Bluetooth
Weight	1.5 pounds	Unknown	2.4 pounds	1.7 pounds	Unknown	Unknown
Battery Life	10 hours, 1 month standby	Unknown	5 hours	16 hours web, 8 video, 48 standby	Unknown	Unknown
Availability	April (Wi-Fi), May (3G)	Sometime in 2010	February-March	Spring	Unknown	March
Price	\$500 to \$830	MAYBE: \$500 to \$600*	\$500	Unknown	Unknown	\$240

\*HP Slate specs from tipster... Thanks!

Figure 1.6 Comparison between iPad and other devices

Based on the popularity over the current market, we finally ruled out other options and left Android and iOS for final consideration.



## **Why Not Android?**

Android is not always designed for the device and too free and liberal for development, in most case, the Android device you see on the web page may or may not have been designed for that particular hardware.

The best example to elaborate this maybe, the Android tablets that you can buy on current market today may not always follow the standard specification. What does it mean, take a tablet as an example, despite Google recommends that tablet should be run on a particular OS version, let say (Android version 3), you can still find that tablets may run on different version with varying degrees of usability, because different telecommunications companies have different target aim. Since Android is an open-source project, there is no tight restriction over it. Sometimes, some tight restrictions are good for development in an overall and a long time aspect.

In comparison, iOS is hardware specific on its release data. You quickly know which devices are compatible, and which ones are not, as recommended by Apple.

So, choosing iOS may leave us less concern over the OS compatibility. We can ensure that if we follow the development guidance Apple Inc. suggested, we can always achieve the best performance for the application need no to consider the particular OS system case by case.



***Figure 1.7 iPhone vs Android***

## **Multi-Touch Technology**

It is simple. As Apple's slogan said:

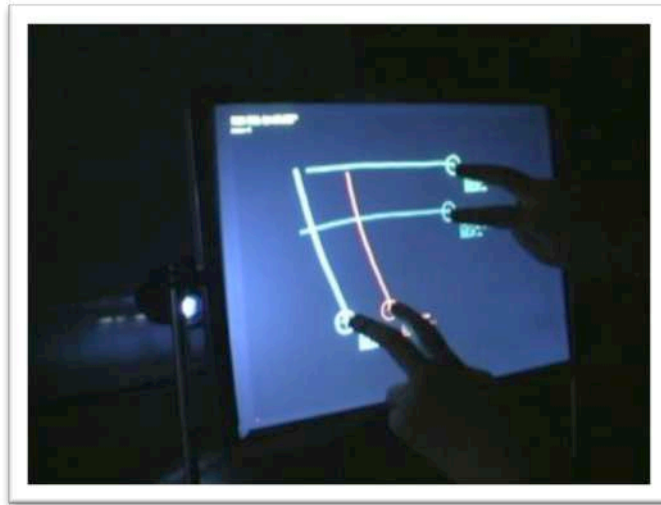
***“Everything's at your fingertips.”***

With iPad, you can use your fingers to do everything. And thanks to advanced and mature Multi-Touch technology, everything is easier and much faster. When a security guard touch the display, it senses them using electrical fields and instantly transforms the taps, swipes, pinches, and flicks into lifelike actions. So, it is easy and effective.

And we believe that such gesture would be very useful and can facilitate the work of a security personal, and as a result to achieve user-friendliness.



***Figure 1.8 iPad Multi-Touch Gesture***



*Figure 1.9 iPad Multi-Touch Gesture*

Such interactive multi-touch technology can bring the user an interactive experience. User will be more rejoice to use it and feel much more comfortable.

# Chapter 2 Case Study on Current Surveillance System

## 2.1 Overview

Our FYP topic is about securing. So, it is essential for us to know more about security works. To have a better understanding on what the nature of securing is, what would it look like and how can we make use of I.T. to assist their work. All of these questions are very important, we have to figure it out before implementing our FYP; only that can make our FYP be practical and achieve the goal we want at the first place. So, given that there are some security companies doing similar things on the current market, it would be a good idea to have a case study about them.

And our target is Motorola's Solution.



*Figure 2.1 Motorola Solutions Logo*

## 2.2 Motorola's Solution

### What Motorola's Solution is?

Motorola's solution is a consulting and technology company that one of its many businesses is security consulting, and its' video surveillance solutions is very famous.

### Why our case study is Motorola's Solution?

First of all, Motorola security has over 65 years experience and always plays the leading role in the security field. It is undoubtedly that this company is professional and experienced in security work.

Also, Motorola's video surveillance solution deploys lots of advanced technologies, we can say that this company always keep up to the modest technology and use it in the security field. Like, they transform video security systems from the passivity (just simple video surveillance monitoring) to a more active, collaborative and intelligent way. It needs the nature of security; from being passive to active.

And we think that, what this company has done is very similar to our FYP idea; we may take this opportunity to study their experience and expertise, to take a reference from these. We sure that a case-study likes that would help our FYP a lot, it must be a good lesson.

## 2.2.1 Architecture

It is comprehensive, holistic, advanced and end-to-end.

Basically, Motorola security architecture includes applications at the backend office, devices in the field and a broadband transport networks.

And combining all of these components plus the components introduced below, it can provide an advanced video security solution, which will be more than just simple video surveillance. Again, his security architecture and ultimate goals are very similar to our FYP idea, the only difference is - the technology/resources involved and development cost are far beyond our FYP.

### **Video Back Office:**

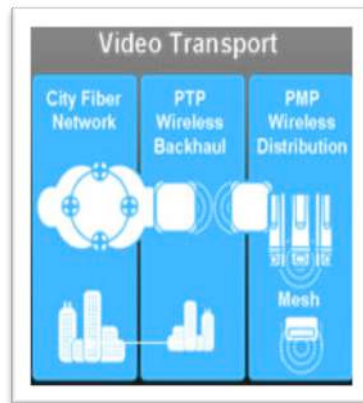
One of the components, to integrate different disparate systems and gathers information from multiple cams. So it aims for easy access and cost-effective storage.



**Figure 2.2 Video Back Office**

### Video Transport:

Optimally distributes and shares video content over wire line and wireless transport networks to and from the video back office.



*Figure 2.3 Video Transport*

### Video Edge:

Like our FYP, in this case, Motorola's remote computing devices, like PDA, cell phone.



*Figure 2.4 Video Edge*

## 2.2.2 Comparison

Comparison between Motorola's Security System and our FYP

Now, we are going to take a look of its features components and to see how are these features similar to our FYP design.

### **1. Both have Video Analytics Engine**

The whole surveillance system is more than just capture what's happening at given locations, but also to interpret the information of the captured video using computers' artificial intelligence. That means, using the pre-determined criteria as a parameter data; with the help of sophisticated algorithms, the system can easily to customize the captured video we want. And it is obvious to say, such video analytics enables faster problem recognition, and can achieve to an effective problem solving.



## ALPR VS our Backend Video-Analyzing program

## Motorola's ALPR

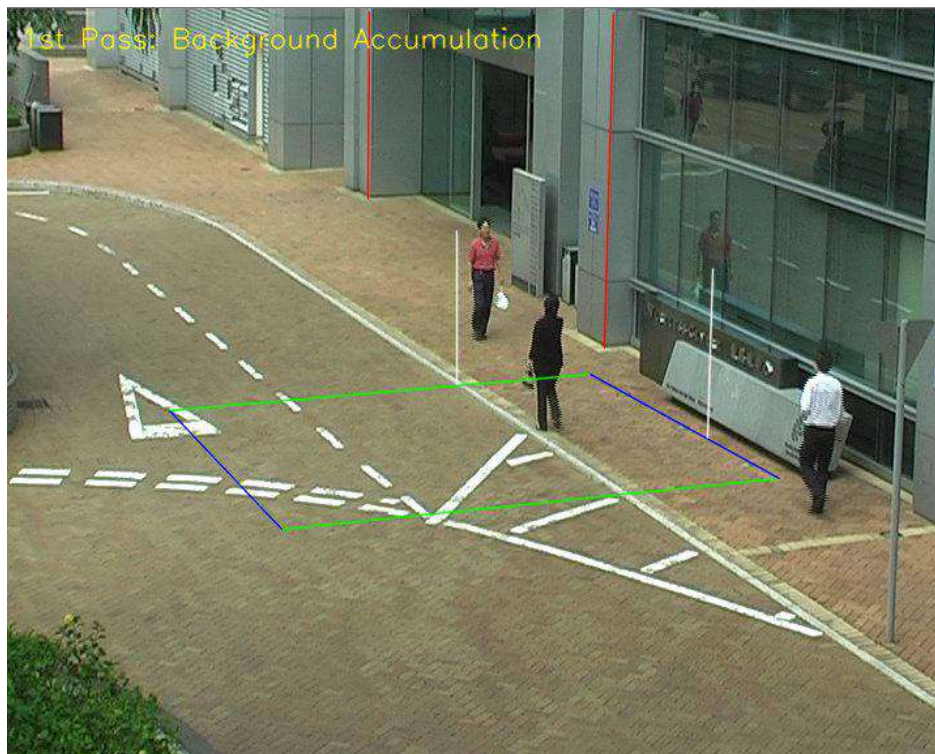
Automatic License Plate Recognition (ALPR) delivers the ability to read vehicle license plates and check them against an installed database for rapid identity verification. The license plate recognition system has been used to locate stolen or wanted vehicles and identify parking-ticket scofflaws.



**Figure 2.5 ALPR**

### **Our backend video-analyzing program**

A visual computer program, which analyzes a recorded video, based on the photo chromatography and algorithm input to compute and analyze the data required, for instance, to calculate how many people has occurred at a particular point within the video.



**Figure 2.6 Backend System**

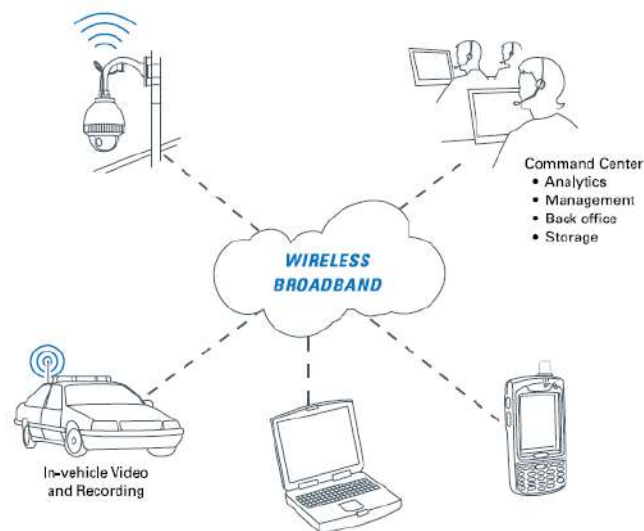
## **2. Both have real time inquire functions**

### **Motorola's Wireless Network**

His system able a quick security-inquire though wireless network they own.



***Figure 2.7 Motorola's Wireless Network***



**Figure 2.8 Wireless Broadband**

### **Our Frontend Application**

If our user want for an assistance or share the information to his/her colleague, he/she can use our image query module and just drag the suspect's thumbnail to the module. (Please refer to the later chapter; there will be more details about it)

## **What did we gain?**

There are many to learn after we have studied the Motorola's surveillance system and we can't write them down here. But some issues are particularly useful and valued.

### **Quality Vs. affordability**

Video quality and bandwidth affordability is two conflicting problems. How can we balance between them and hence maximum the application performance? This issue never came to our mind before we did our case study. And the following table list out the trade off.

Option	Result/Implication
<b>Increase Bandwidth</b>	Reduces cost but provides less detail
<b>Reduce Frame Rate</b>	Reduces cost but results in choppy motion
<b>User Digital Compression Techniques</b>	Essential to all modern digital surveillance solutions to help balance quality and bandwidth needs

## **What Resolution should we use?**

Resolution defines the size and sharpness of the picture. Most of the time, we don't need a very sharp image. Because, for most time, a security guard only need to monitor a video to see if there is anyone pass by and they don't need to recognize whom it is. They just need to know if there really is someone's there and be able to dispatch a response team. So, in such case, a high resolution filming is not required.

However, for the case like traffic monitoring or investigating a burglary, high resolution is need. Suspect's faces should be recognizable and license plates should be readable. So, it becomes a problem to decide what resolution should a cam use.



**Figure 2.9 Resolutions Algorithm**

# Chapter 3 System Review

## 3.1 Overview

There are seven main function modules in our FYP application. We are going to introduce them in this chapter.

### **TimeLine Module**

A module that represents the data (data like how many passengers pass by and when) into a bar/scatter chart; when the user taps the points on the chart, it communicates with the video player and jump that playback time.

### **Video Playback/Streaming Module**

A video player for playing the surveillance video

### **Multiple-Cam Tracking Module**

An auxiliary tool module, system will gather the all cameras' information; user can then switch cam to cam more easily

### **Image Query Module**

One of the inquiry module; user can drag a thumbnail image into the designated area, and an automatic search will be done then.

### **ROI Module**

One of the inquiry modules; user can draw a rectangle over the video screen to indicate the area he/she interested for an advanced search

### **QR Code Check-in Module**

QR Code Check-in Module is used for the security guard to check in fast and easily.

### **Enhanced ROI Module**

Enhanced ROI Module is an extended class from original ROI Module. It provides a searching function with more refined and accurate filtering rules.

After that, we would talk about what APIs have we used and have a quick study on them.



## 3.2 Implementation

There are seven modules in our system, they are:

1. TimeLine Module
2. Video Playback/Streaming Module
3. Multiple-Cam Tracking Module
4. Image Query Module
5. ROI Module
6. QR Code Check-in Module
7. Enhanced ROI Module

## Main Screen

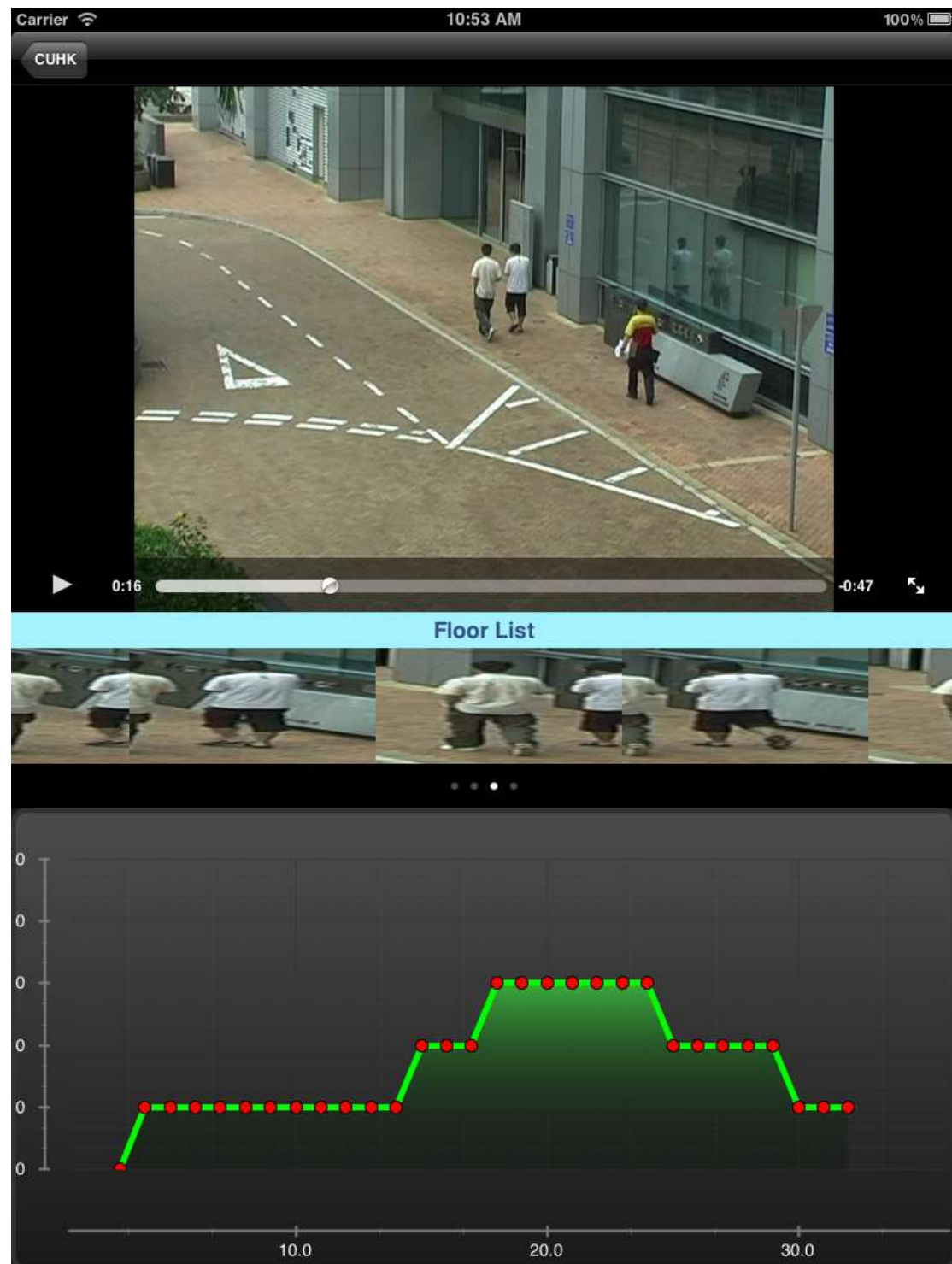
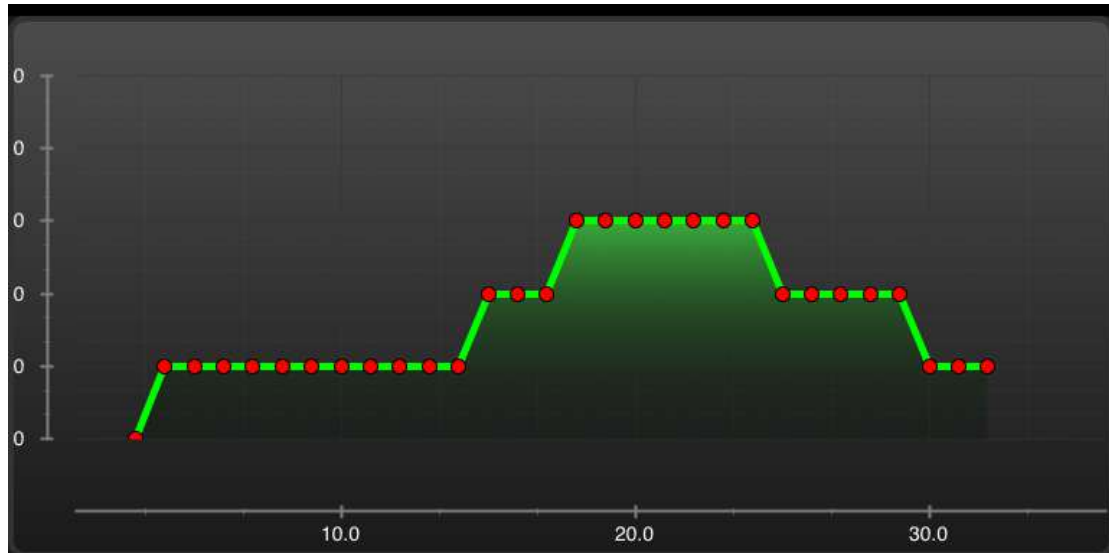


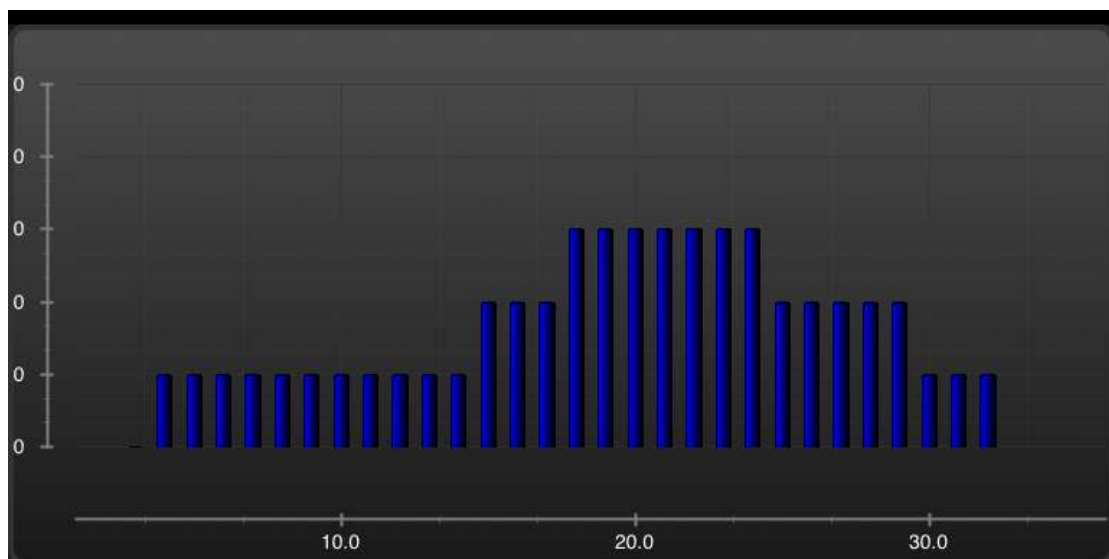
Figure 3.1 Main Screen

The following is the description of functions in each module.

### **TimeLine Module**



*Figure 3.2 TimeLine Module (Line Chart)*



*Figure 3.3 TimeLine Module (Bar Chart)*

Timeline Module is a representing module. It represents the data (data like number of passengers passed by and time) in a bar/scatter chart-manner; each point on the chart represents the corresponding amount of passengers recorded at that time. Moreover, it integrates with Core-plot API and the Playback module, when the user taps the points on the chart, it sends a message to the video player and player will jump into the corresponding playing time directly.

## Video Playback/Streaming Module



*Figure 3.4 Video Playback/Streaming Module*

Video Playback/Streaming Module is to play back the recorded and analyzed video stored in the back-end server. The video is to show what actually happened right back right there. HTTP streaming protocol is used for video streaming for instantaneous playback and user convenience; so the video could be instantaneously available without loading it all.

## Multiple-Cam Tracking Module



Figure 3.5 Multiple-Cam Tracking Module

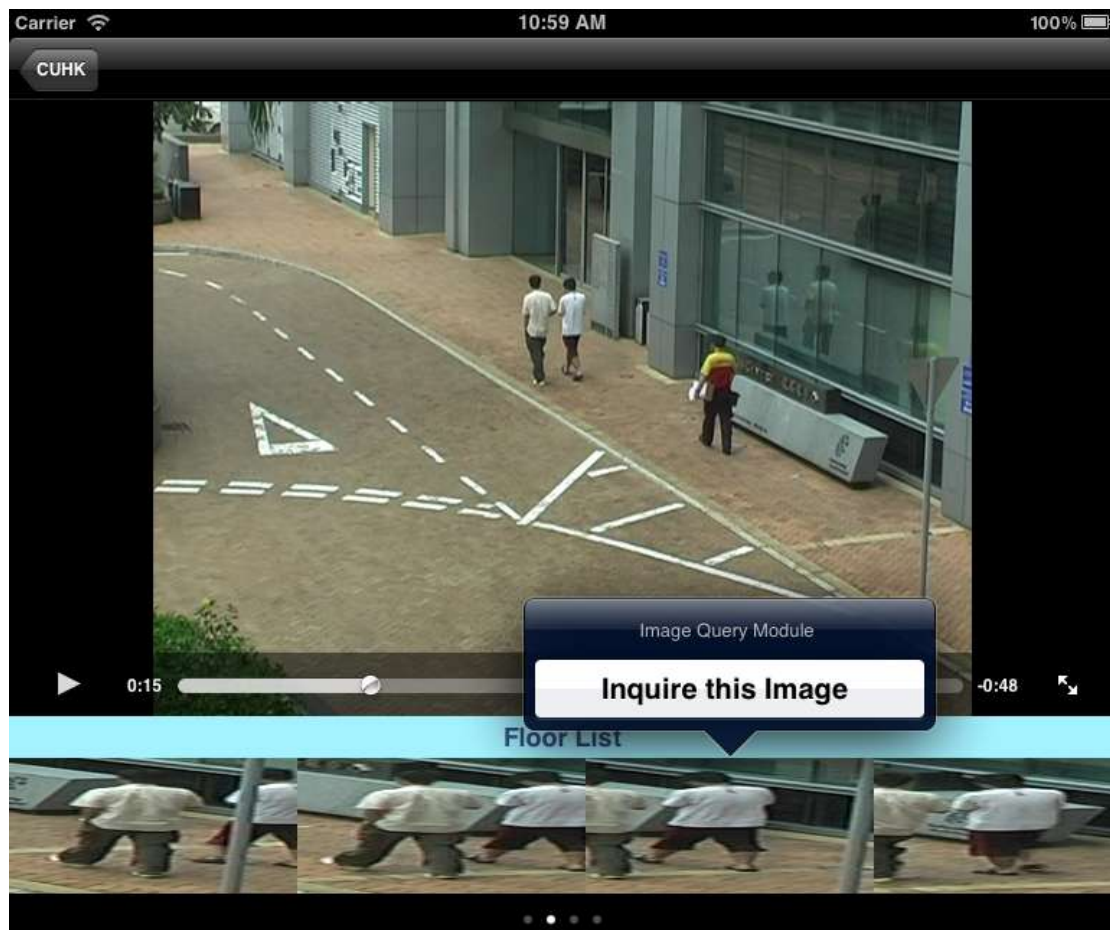


Multiple-Cam Tracking Module is an auxiliary tool module and used to monitor several CCTVs at one time; we developed this module because we discovered that security guards always need to keep a watchful eye on different sites and sometimes they even need to switch between cam and cam for navigation; so there we come this module to solve the problem.

Multiple-Cam Tracking Module consists of multiple live CCTV in a page and the page is generated dynamically according to the JSON file stipulated. The JSON file stores the cam information such as number of cams, cam's neighborhood relationship, cam's display coordination and size (the details of this part will be explained in the later chapter).

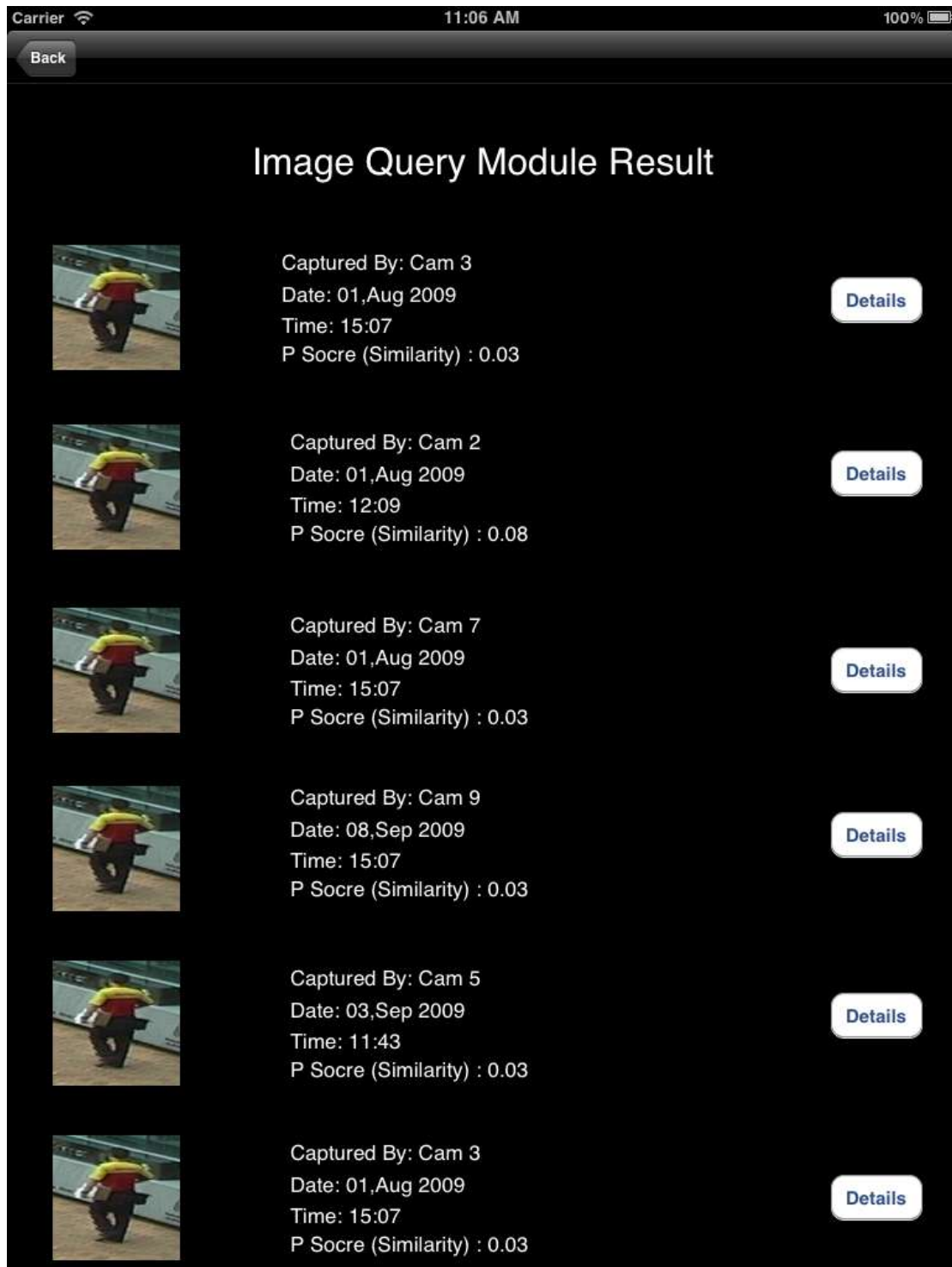
We make use of the iOS gesture to facilitate the cam-switching navigation; we integrate our module with iOS' gesture recognizer's framework. Users can easily swipe up or down to switch the floor they are viewing, and left or right to change into neighbor cam.

## Image Query Module



*Figure 3.6 Image Query Module*





*Figure 3.7 Image Query Module with Result*

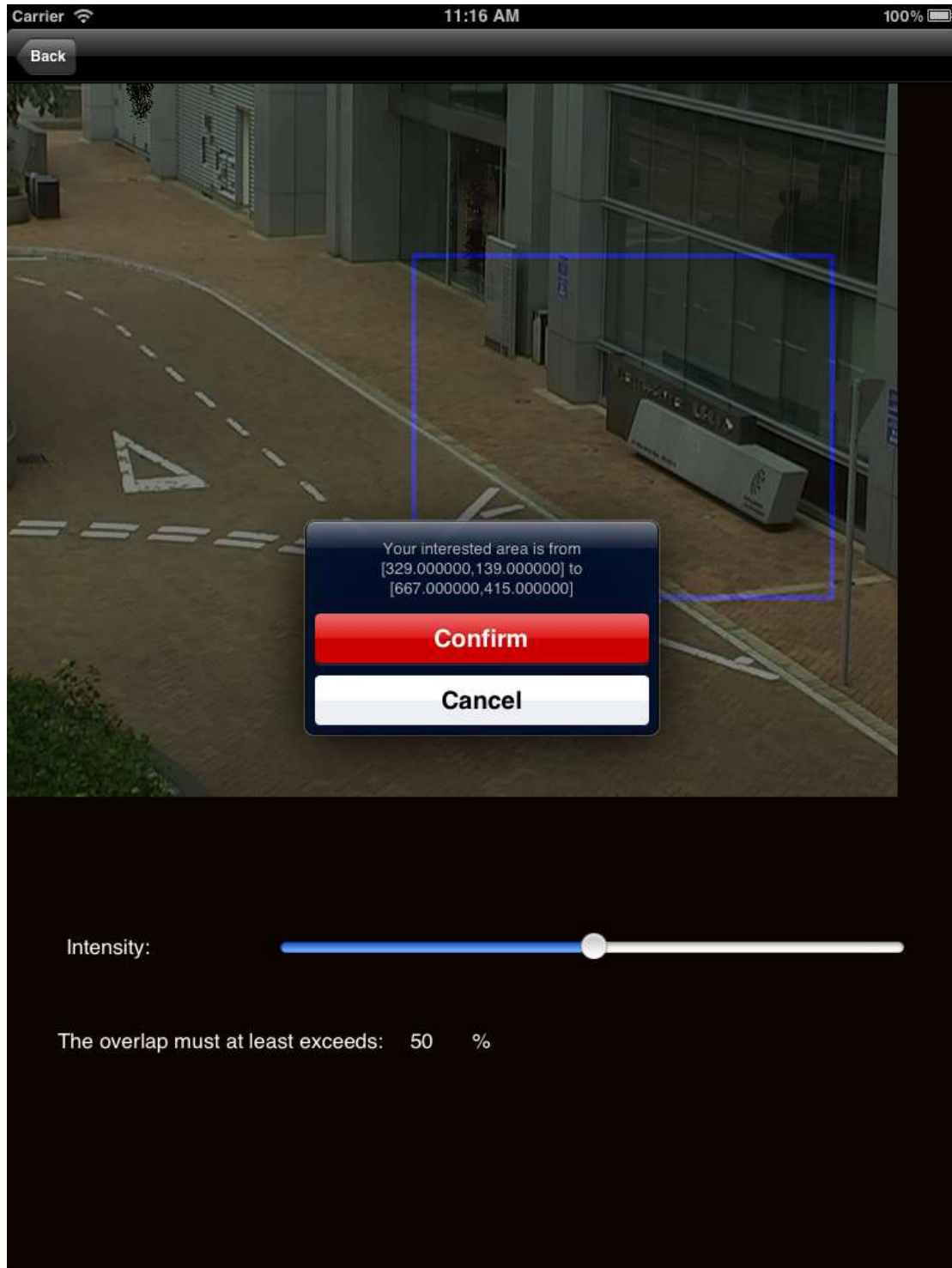
Image Query Module is an image-based data-searching module. When there is suspicious person in the video, user can activate the Image Query Module; Image Query Module will send the image ID to the back-end system for relevant-image searching; the server will return all the relevant-image captured in others cameras. Our Module will then interpret the result and represent to the user. To conclude, Image Query Module is an intermediate platform communicating with the back-end server for the imaged-based searching.

## ROI Module

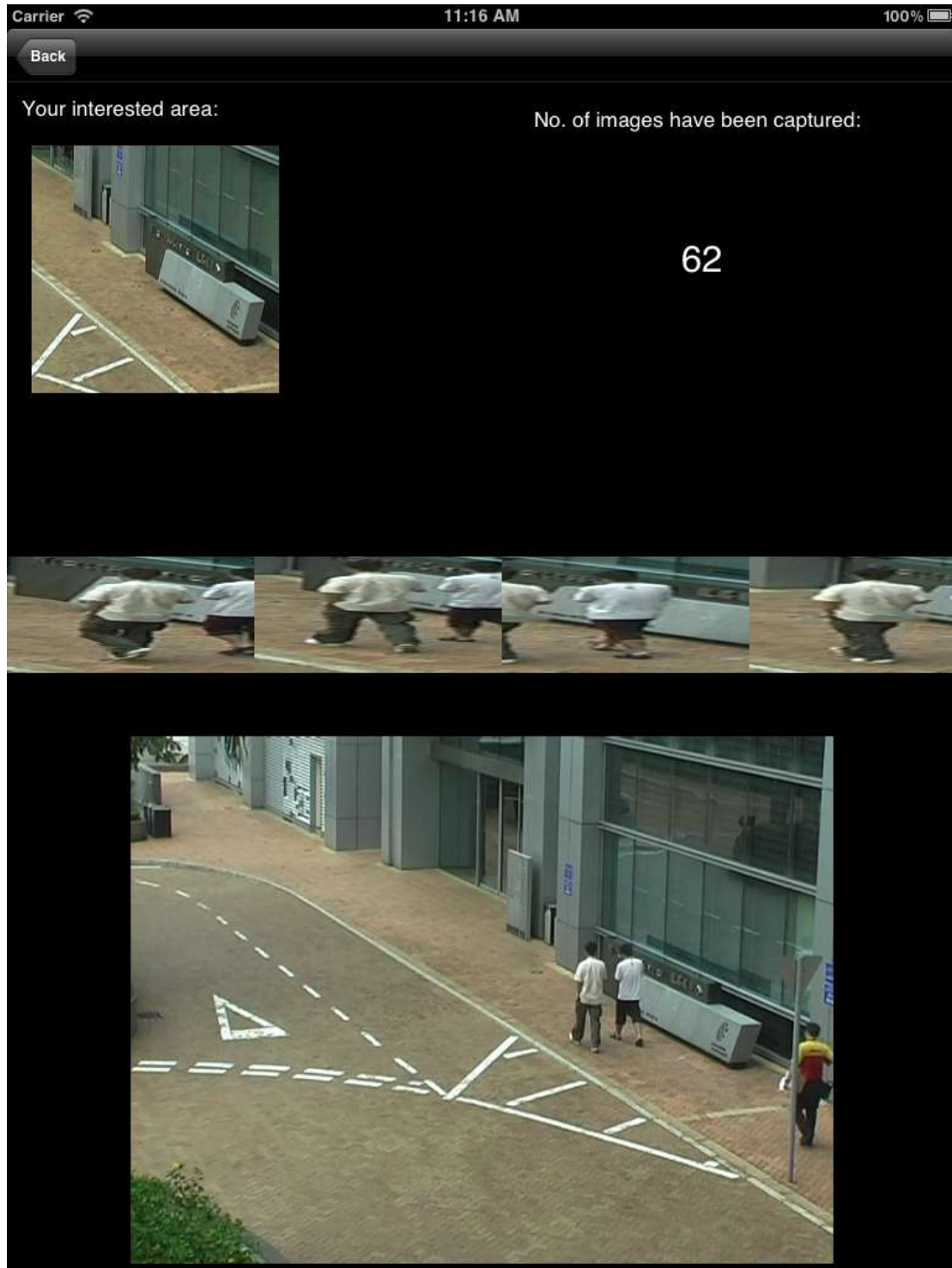


*Figure 3.8 ROI Module*

Region of Interest (ROI) Module is a coordination-based data-searching module. Users can easily indicate their interested area by drawing a rectangle on the street-view. Then ROI Module would calculate the corresponding the coordination and request for that JSON file from the server. After that, it would interpret the returned JSON file and undergo a series of ROI area-related algorithm to find what data “fit-into” the requirement and put them into result list. Finally it will then represent the result to the user by showing the corresponding thumbnail captured.



*Figure3.9 After Cropping*



*Figure 3.10 After ROI*

## QR Code Check-in Module

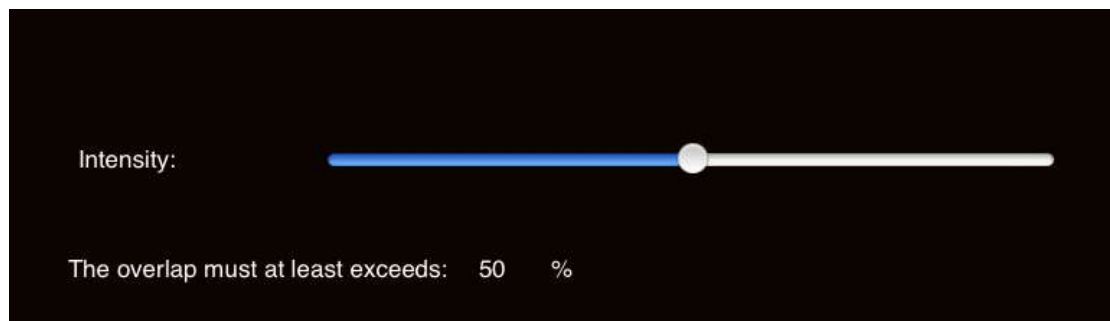


*Figure 3.11 QR Code*

QR Code Check-in Module is used for the security guard to check in fast and easily. We make use of the advantages of QR Code to achieve the target causes QR code can provide a fast readability and large storage capacity compared to other methods.

When a security guard is on patrol and arrives at a site, he/ she can just easily open the QR Code Module and face the camera to the QR Code to check-in and the record will be automatically recorded at the server side along with the ID and the time.

## **Enhanced ROI Module**



**Figure 3.12 Enhanced ROI Module**

Enhanced ROI Module is an extended class from original ROI Module. Enhanced Region of Interest (EROI) provides a searching function with more refined and accurate filtering rules.

Intensity slide-bar here is to indicate the overlapping percentage a user preferred when he/she is doing the EROI searching. This value will be used as a threshold value for judging whether the data should put into the result list: only when the image sharing the overlapped area with interested area exceeds the indicated intensity value would counts as one of the result.



Intensity here means the overlapped area between two objects. As showed in the figure below, the pedestrians' coordination rectangle overlaps with the interested area, (the arrow pointed area). This area will be used as a threshold value for judging whether this pedestrian is "inside" the indicated interested area drawn by the user.



*Figure 3.13 Overlapping Area*

### 3.3 API Study

In our FYP, frameworks of Core-Plot, JSON and ZXing were used.



**Figure 3.14 Core-Plot Logo**

Core-Plot was chosen for plotting graph. In the TimeLine Module, the time that pedestrians passed by are collected in a data array; and those data are used to plot a bar chart and a scatter chart.



**Figure 3.15 JSON Logo**

JSON was used as a way of data interchange between the server side and the client side.



*Figure 3.16 ZXing*

ZXing was used for QR-Code Check in Module. ZXing is an API-Kit developed by ZXing Team that allow a user to scan 2-D graphical barcodes with the camera in their iOS devices.

### 3.3.1 Core-Plot

#### What Core-Plot is?

Core-Plot is a third party open-source library developed for iOS enhancement. It is a plotting framework for Mac OS X and iOS. It provides 2D visualization of data, and is tightly integrated with Apple technologies like Core Animation, Core Data, and Cocoa Bindings.

#### Structure

```
@interface CPTGraph : CPTBorderedLayer
{
    @private
    CPTPlotAreaFrame *plotAreaFrame;
    NSMutableArray *plots;
    NSMutableArray *plotSpaces;
    NSString *title;
    CPTTextStyle *titleTextStyle;
    CPTRectAnchor titlePlotAreaFrameAnchor;
    CGPoint titleDisplacement;
    CPTLayerAnnotation *titleAnnotation;
    CPTLegend *legend;
    CPTLayerAnnotation *legendAnnotation;
    CPTRectAnchor legendAnchor;
    CGPoint legendDisplacement;
}

@property (nonatomic, readwrite, copy) NSString *title;
@property (nonatomic, readwrite, copy) CPTTextStyle *titleTextStyle;
@property (nonatomic, readwrite, assign) CGPoint titleDisplacement;
@property (nonatomic, readwrite, assign) CPTRectAnchor titlePlotAreaFrameAnchor;
@property (nonatomic, readwrite, retain) CPTAxisSet *axisSet;
@property (nonatomic, readwrite, retain) CPTPlotAreaFrame *plotAreaFrame;
@property (nonatomic, readonly, retain) CPTPlotSpace *defaultPlotSpace;
@property (nonatomic, readwrite, retain) NSArray *topDownLayerOrder;
@property (nonatomic, readwrite, retain) CPTLegend *legend;
@property (nonatomic, readwrite, assign) CPTRectAnchor legendAnchor;
@property (nonatomic, readwrite, assign) CGPoint legendDisplacement;

-(void)reloadData;
-(void)reloadDataIfNeeded;

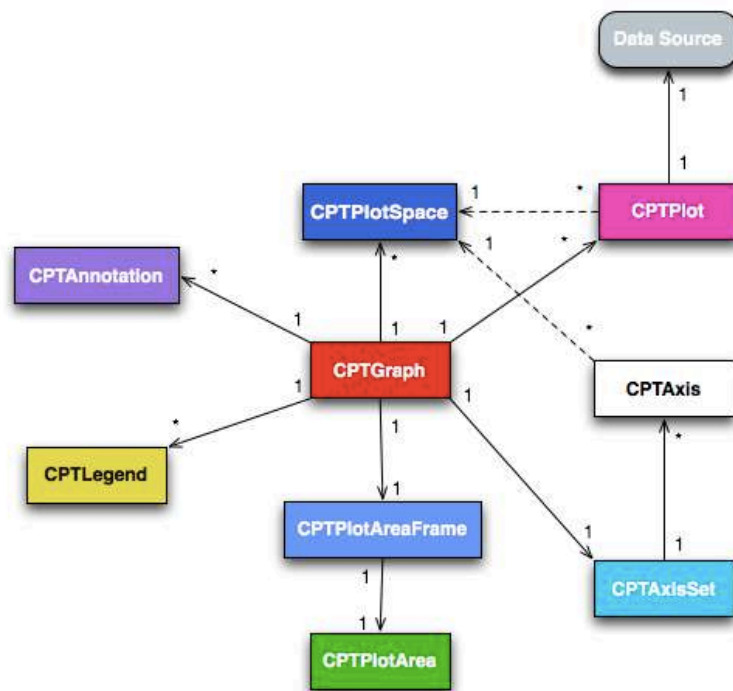
-(NSArray *)allPlots;
-(CPTPlot *)plotAtIndex:(NSUInteger)index;
-(CPTPlot *)plotWithIdentifier:(id <NSCopying>)identifier;
```

```
-(void)addPlot:(CPTPlot *)plot;
-(void)addPlot:(CPTPlot *)plot toPlotSpace:(CPTPlotSpace *)space;
-(void)removePlot:(CPTPlot *)plot;
-(void)removePlotWithIdentifier:(id <NSCopying>)identifier;
-(void)insertPlot:(CPTPlot *)plot atIndex:(NSUInteger)index;
-(void)insertPlot:(CPTPlot *)plot atIndex:(NSUInteger)index
intoPlotSpace:(CPTPlotSpace *)space;

-(NSArray *)allPlotSpaces;
-(CPTPlotSpace *)plotSpaceAtIndex:(NSUInteger)index;
-(CPTPlotSpace *)plotSpaceWithIdentifier:(id <NSCopying>)identifier;
-(void)addPlotSpace:(CPTPlotSpace *)space;
-(void)removePlotSpace:(CPTPlotSpace *)plotSpace;
-(void)applyTheme:(CPTTheme *)theme;

@end
```

## Class Diagram



*Figure 3.17 Core-Plot Class Diagram*

## Objects and Layers

This diagram shows run time relationships between objects (right) together with layers in the Core Animation layer tree (left). Color-coding shows the correspondence between objects and their corresponding layers.

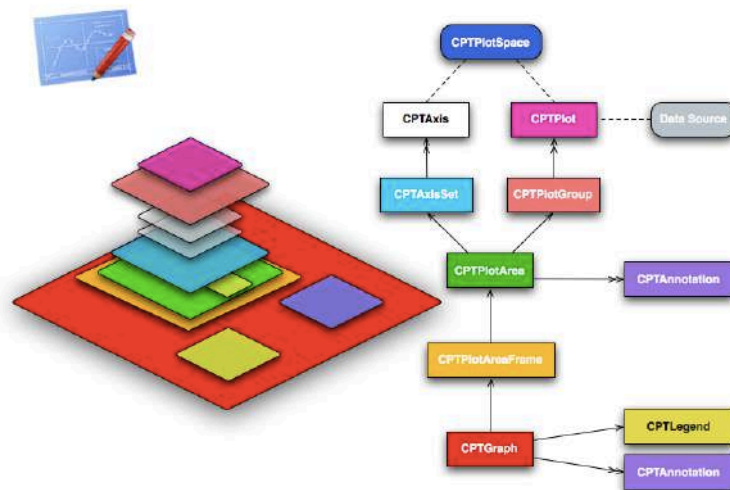


Figure 3.18 Object and Layers

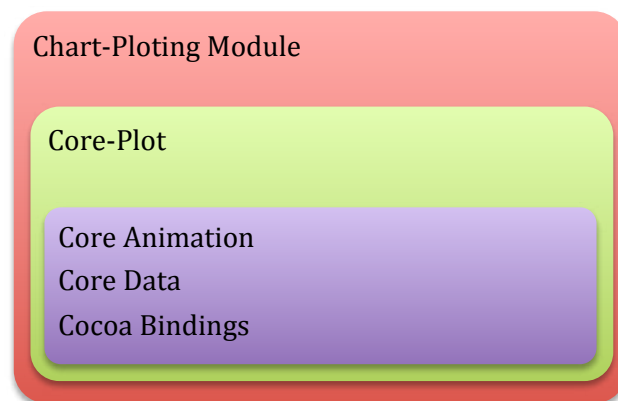
## Porting Library to iOS

A step-by-step approach has been used to port the library; First it is being ported to Mac OS X then porting to iOS.

A step-by-step porting strategy is used because Core-Plot is a very complicated and large-scale library, we want to have a better understanding first before implement it on iOS.

The first step is quite straightforward, cause Mac OS Lion has lots of OS function to co-operate with Core-Plot. It is not so hard.

However, the second move didn't go so well at first. Because iOS has relatively limited libraries call to work with (when compared to Mac OS), so we have to limit our design to "fit in". For example, it is okay for Core-Plot to generate the plot dynamically without saving the data in an array; however this is not the same on iOS. We have to do modify our code, to save the data first.



**Figure 3.19 Module Diagram**



## **Limitation**

Core-Plot is limited to iOS specification. Thanks to the strict specification and configuration Apple Inc. set for iOS. Core-Plot can't provide a dynamic linked Library ( a DLL file ), only a static library is provided instead. So, if you want to use Core-Plot, you have to include the whole package into your Xcode and set up the system configuration for compiling and releasing. And all those work are very complicated and bothering.

### 3.3.2 JSON

#### **Why JSON?**

There is another text notation that has all of the advantages of XML, but is much better suited to data-interchange. That notation is JavaScript Object Notation (JSON).

#### **Why not XML?**

XML is not well suited to data-interchange, much as a wrench is not well suited to driving nails. It carries a lot of baggage, and it doesn't match the data model of most programming languages. When most programmers saw XML for the first time, they were shocked at how ugly and inefficient it was. It turns out that that first reaction was the correct one.

Let's compare XML and JSON on the attributes that the XML community considers important.

#### **Simplicity**

JSON is much simpler than XML. JSON has a much smaller grammar and maps more directly onto the data structures used in modern programming languages.

**Extensibility**

JSON is not extensible because it does not need to be. JSON is not a document markup language, so it is not necessary to define new tags or attributes to represent data in it.

**Interoperability**

JSON has the same interoperability potential as XML.

**Openness**

JSON is at least as open as XML, perhaps more so because it is not in the center of corporate/political standardization struggles.

### 3.3.3 ZXing

#### **What ZXing is?**

ZXing is an API-Kit developed by ZXing Team that allow a user to scan 2-D graphical barcodes with the camera in their iOS devices.

With the features ZXing provided, QR Code can be used to store and retrieve web addresses, geographical coordinates, and text ...etc.

#### **What QR Code is?**

QR Code (abbreviated from Quick Response Code) is the trademark for a type of matrix barcode (or two-dimensional code) first designed for the automotive industry. More recently, the system has become popular outside of the industry due to its fast readability and large storage capacity compared to standard UPC barcodes. The code consists of black modules arranged in a square pattern on a white background. The information encoded can be made up of four standardized kinds ("modes") of data (numeric, alphanumeric, byte/binary, Kanji), or through supported extensions, virtually any kind of data.



*Figure 3.20 QR Code*

## Why ZXing?

First of all, it is one of the most popular libraries in current smartphone market, it already has over a quarter million ratings and more than 50 million downloads. Moreover, ZXing is a very mature and well-known third-party library, it is safe and easy to develop our module using ZXing.

## Structure

```
/**
 * Copyright 2009 Jeff Verkoeyen
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#include <UIKit/UIKit.h>
#include <AudioToolbox/AudioToolbox.h>
#import <AVFoundation/AVFoundation.h>
#include "Decoder.h"
#include "parsedResults/ParsedResult.h"
#include "OverlayView.h"

@protocol ZXingDelegate;

#if !TARGET_IPHONE_SIMULATOR
#define HAS_AVFF 1
#endif

@interface ZXingWidgetController : UIViewController<DecoderDelegate,
                                CancelDelegate,
                                UINavigationControllerDelegate

#if HAS_AVFF
                                ,
                                AVCaptureVideoDataOutputSampleBufferDelegate
#endif

> {

    NSMutableSet *readers;
    ParsedResult *result;
    OverlayView *overlayView;
    SystemSoundID beepSound;
    BOOL showCancel;
    NSURL *soundToPlay;
    id<ZXingDelegate> delegate;
    BOOL wasCancelled;
    BOOL oneDMode;
#if HAS_AVFF
    AVCaptureSession *captureSession;
    AVCaptureVideoPreviewLayer *prevLayer;
#endif
    BOOL decoding;
    BOOL isStatusBarHidden;
}

#if HAS_AVFF
@property (nonatomic, retain) AVCaptureSession *captureSession;
@property (nonatomic, retain) AVCaptureVideoPreviewLayer *prevLayer;
```

```
#endif
@property (nonatomic, retain ) NSSet *readers;
@property (nonatomic, assign) id<ZXingDelegate> delegate;
@property (nonatomic, retain) NSURL *soundToPlay;
@property (nonatomic, retain) ParsedResult *result;
@property (nonatomic, retain) OverlayView *overlayView;

- (id)initWithDelegate:(id<ZXingDelegate>)delegate showCancel:(BOOL)shouldShowCancel
OneDMode:(BOOL)shouldUseOneDMode;

- (BOOL)fixedFocus;
- (void)setTorch:(BOOL)status;
- (BOOL)torchIsOn;

@end

@protocol ZXingDelegate
- (void)zxingController:(ZXingWidgetController*)controller didScanResult:(NSString
*)result;
- (void)zxingControllerDidCancel:(ZXingWidgetController*)controller;
@end
```

## Class Diagram

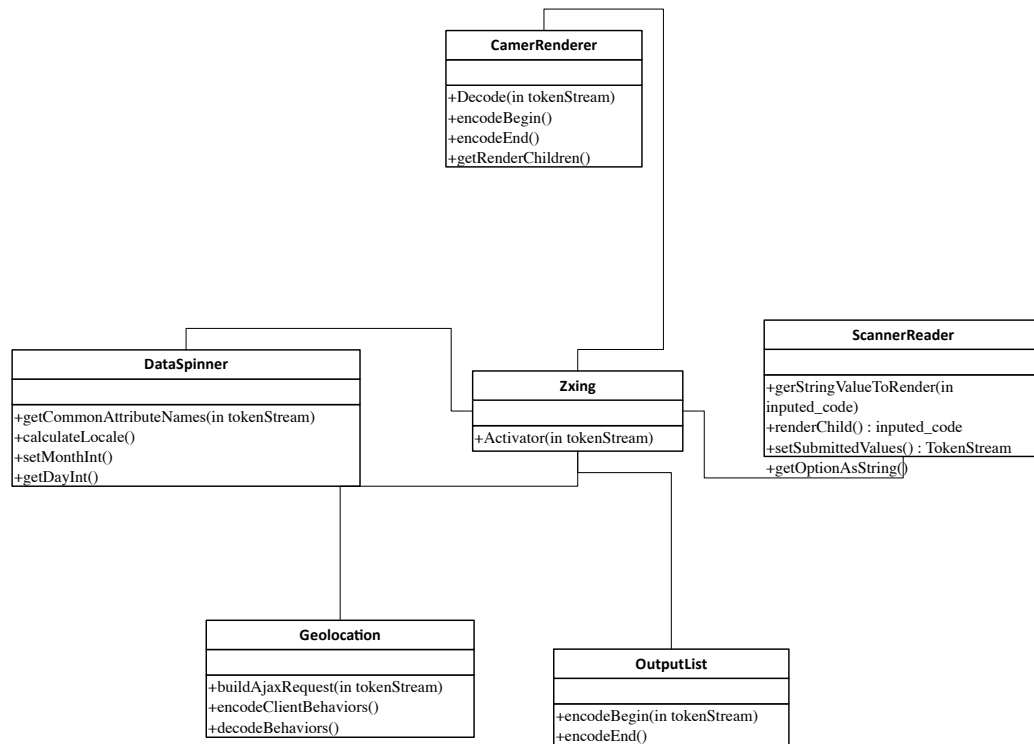


Figure 3.23 ZXing Class Diagram

# Chapter 4 System Design

## 4.1 Overview

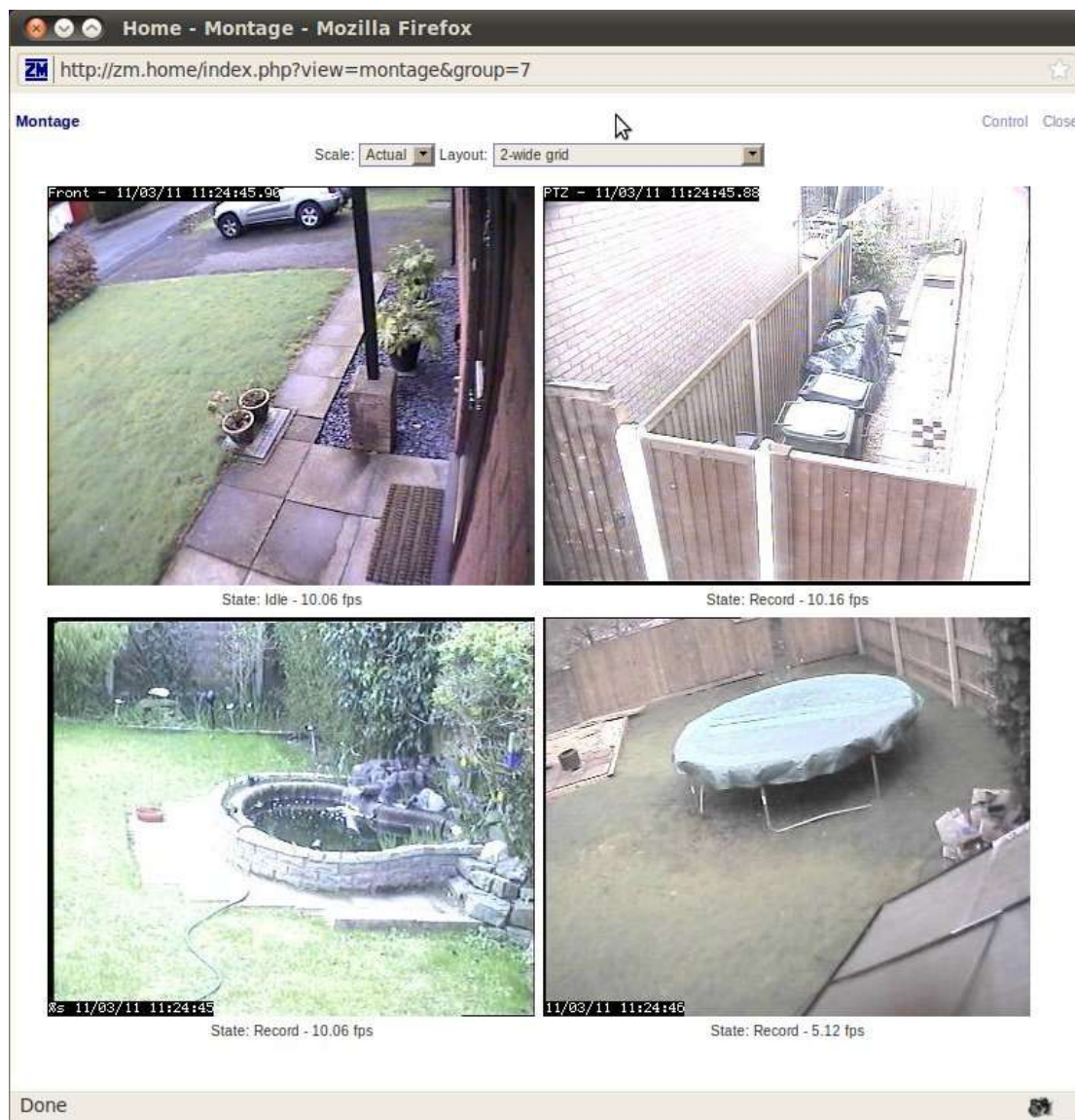
In this chapter, we mainly focus on the design issues; on to explain how our FYP system is design and why do we design it in such way. We would go through the whole explanation process through the aid of different system diagram and modules descriptions.

For different modules our FYP consists of, we would talk about how did it be implemented and the principle behinds. What is more, we then turn our focus to the server side; have a look on the design issue about it.

Last but not least, at the end of this chapter, two System Component diagrams are shown to have an overall grasp on how our system would work in a well-ground and comprehensive way.



## 4.2 Convention

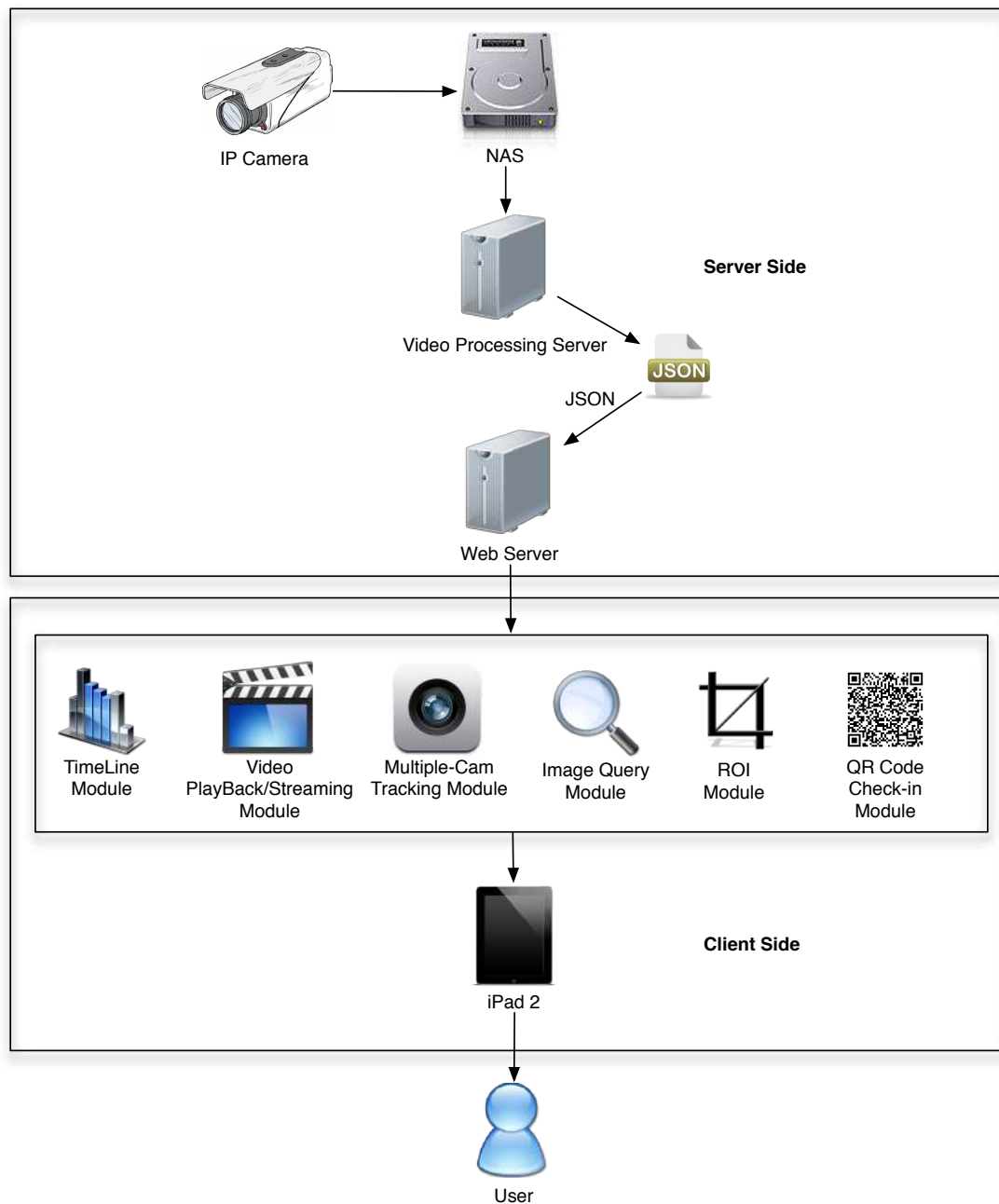


**Figure 4.1 Modern Security System**

Traditionally, most of the surveillance system only provides a screen monitoring for the security guard, and the function of it is only can show the past video, incase there is a crime happen, it is very hard to find a suspicious people in the video.

## 4.3 System Architecture Design

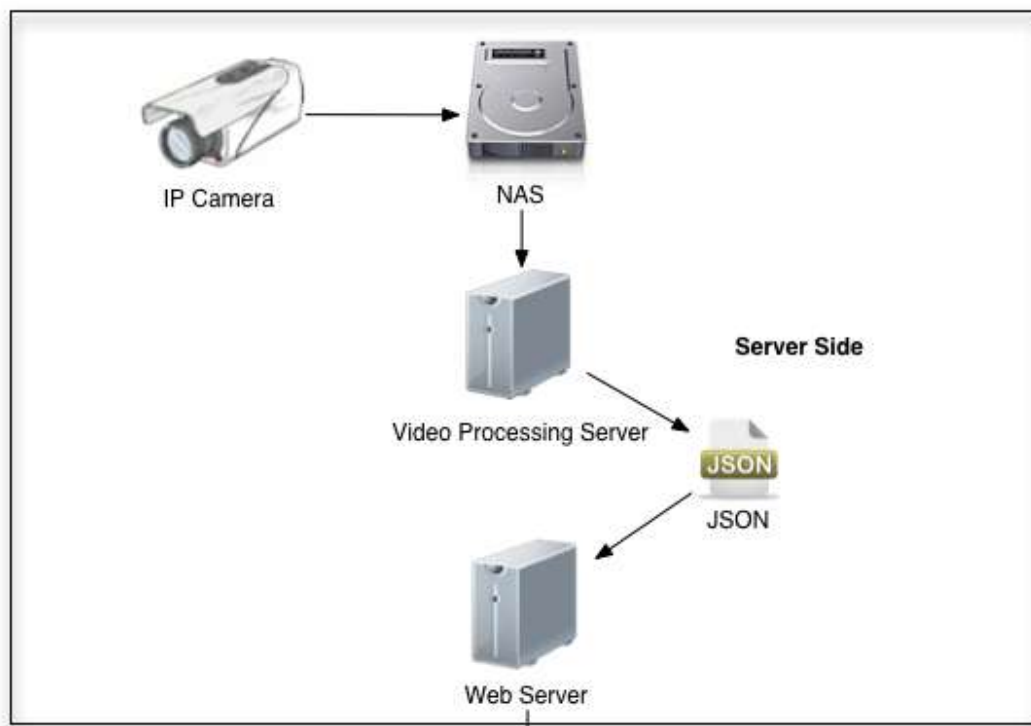
### 4.3.1 System Block Diagram



**Figure 4.2 System Block Diagram**

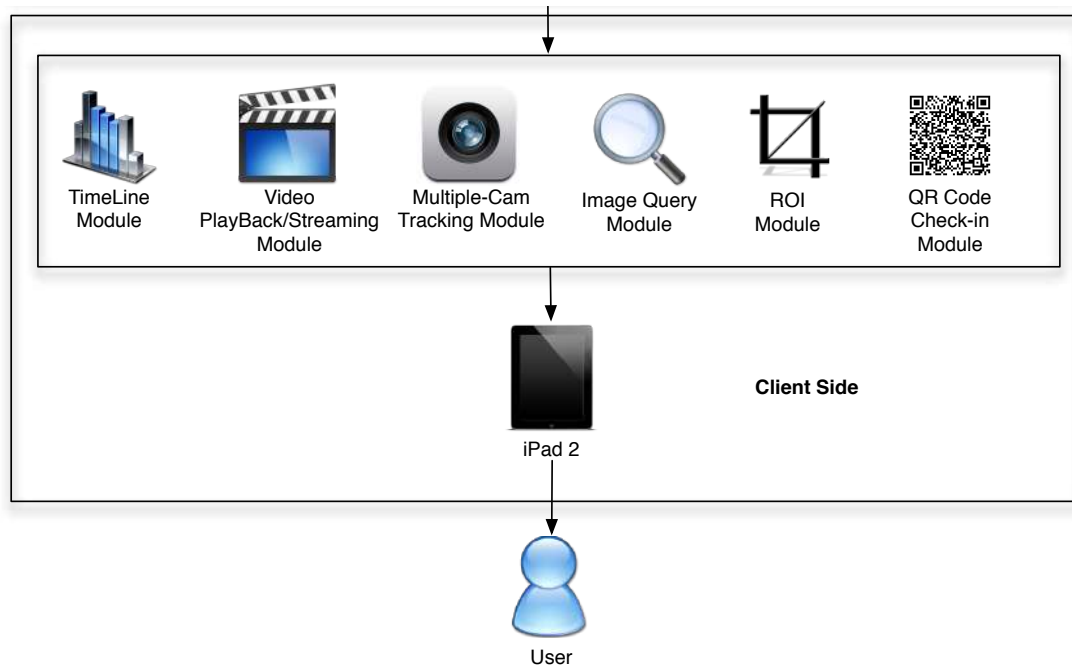
The whole system can mainly be divided into two parts: Back-End video analyzing system and the front-end iOS UI platform.

In the server side, after the IP Cameras recorded the video and stored in the backend system storage. Video Processing Server Module would then retrieve that video file for analyzing and calculation, a JSON file containing useful information would then be generated and sent to Web Server. Web Server is just basically an Apache-based platform for handling the HTTP request between iOS devices and the Video Processing Server Module.



**Figure 4.3 Server Side**

While in the client side, as mentioned before, our project composes of seven function modules. The details of each module would now state below.



**Figure 4.4 Client Side**

## Description of each module

### TimeLine Module

TimeLine Module
-(void) load
-(void) connect_CorePlot
-(void) parseJSON
-(IBAction) tap

*Figure 4.5 TimeLine Module*

The Timeline Module consists of four methods. The first one is load, which is used to initial all the instance variables, class variables and all the environment settings. The second one is connect\_CorePlot, which is used to plot the timeline graph so that it can display the information passed by the JSON file, and we have chosen the Core-Plot framework for drawing the graph. More about Core-Plot framework will be discussed in the next chapter. The third one is parseJSON, which is used to parse the JSON file that is downloaded from the Web Server, and the JSON file stored all the necessary data about the video. The fourth one is tap, which is an event handler, when the user touches the timeline module, it will trigger the “tap” function, which will bring the movie to the appropriate time.

## **Video Playback/Streaming Module**

<b>Video Playback/Streaming Module</b>
<b>-(void) load</b> <b>-(void) parseJSON</b> <b>-(IBAction) play</b>

***Figure 4.6 Video Playback/Streaming Module***

The Video Playback/Streaming Module consists of three methods. The first one is load, which is used to initial all the instance variables, class variables and all the environment settings. The second one is parseJSON, which is used to parse the JSON file that is downloaded from the Web Server, and the JSON file stored all the necessary data about the video. The third one is play, which is an event handler, when the user touches the Video Playback/Steaming Module, it will trigger the “play” function, which will play the video.

## **Multiple Cameras Module**

<b>Multiple-Cam Tracking Module</b>
- (void) load - (void) parseJSON - (IBAction) up - (IBAction) down - (IBAction) left - (IBAction) right

***Figure 4.7 Multiple-Cam Tracking Module***

The Multiple-Cam Tracking Module consists of six methods. The first one is load, which is used to initial all the instance variables, class variables and all the environment settings. The second one is parseJSON, which is used to parse the JSON file that is downloaded from the Web Server, and the JSON file stored all the necessary data about the video. The third, fourth, fifth and sixth are up, down, left and right which is an event handler, when the user touches the Multiple-Cam Tracking Module, it will trigger the “up” or “down” or “left” or “right” function, which will jump to the corresponding camera.

## Image Query Module

Image Query Module
-(void) load -(void) parseJSON -(IBAction) submit

**Figure 4.8 Image Query Module**

The Image Query Module consists of three methods. The first one is load, which is used to initial the entire instance variables, class variables and all the environment settings. The second one is parseJSON, which is used to parse the JSON file that is downloaded from the Web Server, and the JSON file stored all the necessary data about the video. The third one is submit which is an event handler, when the user touches the Image Query Module, it will trigger the “submit” function, which will submit the image to the server, and then the server will search all the cameras see whether the image is inside it.



## **ROI Module**

<b>ROI Module</b>
-(void) load -(void) parseJSON -(IBAction) crop

*Figure 4.9 ROI Module*

The ROI Module is consists of three methods. The first one is load, which is used to initial the entire instance variables, class variables and all the environment settings. The second one is parseJSON, which is used to parse the JSON file that is downloaded from the Web Server, and the JSON file stored all the necessary data about the video. The third one is crop, which is an event handler, when the user touches the ROI Module, it will trigger the “crop” function, which will crop the image and send it to the server, and then the server return the corresponding area information back to the application.

## **Enhanced ROI Module**

Enhanced ROI Module
<ul style="list-style-type: none"><li>-(void) load</li><li>-(void) parseJSON</li><li>-(void) intensity_calculation</li><li>-(void) store_Result</li><li>-(IBAction) crop</li><li>-(IBAction) check</li></ul>

***Figure 4.10 Enhanced ROI Module***

Enhanced ROI Module is an extended class from original ROI Module. Enhanced Region of Interest (EROI) provides a searching function with more refined and accurate filtering rules.

EROI share the basic coding structure with ROI Module, so most of them are the same.

The function `intensity_calculation()` is used for binding-box area coverage calculation. `intensity_calculation()` read the value from the intensity slide-bar and use that value for binding-box coverage calculation. For instance, it calculates and compares the coordination of the stated area and the coordination of the image captured in server slide, to see how much overlapping area two object share. If the overlapped area exceeds the required threshold value, `intensity_calculation()` would call `store_result()` to store the data into the result list.

## QR Code Check- in Module

QR Code Check-in Module
<ul style="list-style-type: none"><li>-(void) load</li><li>-(void) call_ZXing</li><li>-(void) didScanResult</li><li>-(void) didCancel</li><li>-(void) store_Result</li><li>-(IBAction) check_in</li></ul>

*Figure 4.11 QR Code Check-in Module*

QR Code Check-in Module is used for the security guard to check in fast and easily.

Call\_Zxing() is a method we developed to connect with the ZXing API module. When the user push the Check-in button, Call\_Zxing() will be activated and send a message to the Zxingcontroller for preparation; after that it will pass the control to the ZXing module.

Zxingcontroller: didScanResult:( ) is a function call provided from the ZXing API Kit, it deal with many iOS Core foundations like camera-module, animation module .... etc.

Zxingcontroller: didScanResult:( ) is used to decode the QRCode that the camera is facing and send back the decoded result to the calling object.

## **Web Server**

Web Server here is just basically an Apache-based platform for handling the HTTP request from iOS devices; serving a communication bridge between the mobile devices and the Video Processing Server Module.

The main role of the web server is to store the JSON file generated by the Video Processing Server. And, as mentioned above, it handles HTTP requests from different function modules of our FYP.

## **Why Apache?**



*Figure 4.12 Apache*

When coming to decide which type of web-server we should use, we come straightforward.

### **Popularity**

Apache HTTP Server is chosen because it is one of the most popular HTTP server software in use; According to the Apache Software Foundation , in March 2012 Apache 57.46% websites are using Apache and engaging 65.24% of the top servers market across all domains

**Mature, Stable and High Performance**

Apache supports a variety of features, and there are a lot well-developed libraries for us to use. For instance, authentication modules including `mod_access`, `mod_auth` and `mod_digest` could be very useful to our project.

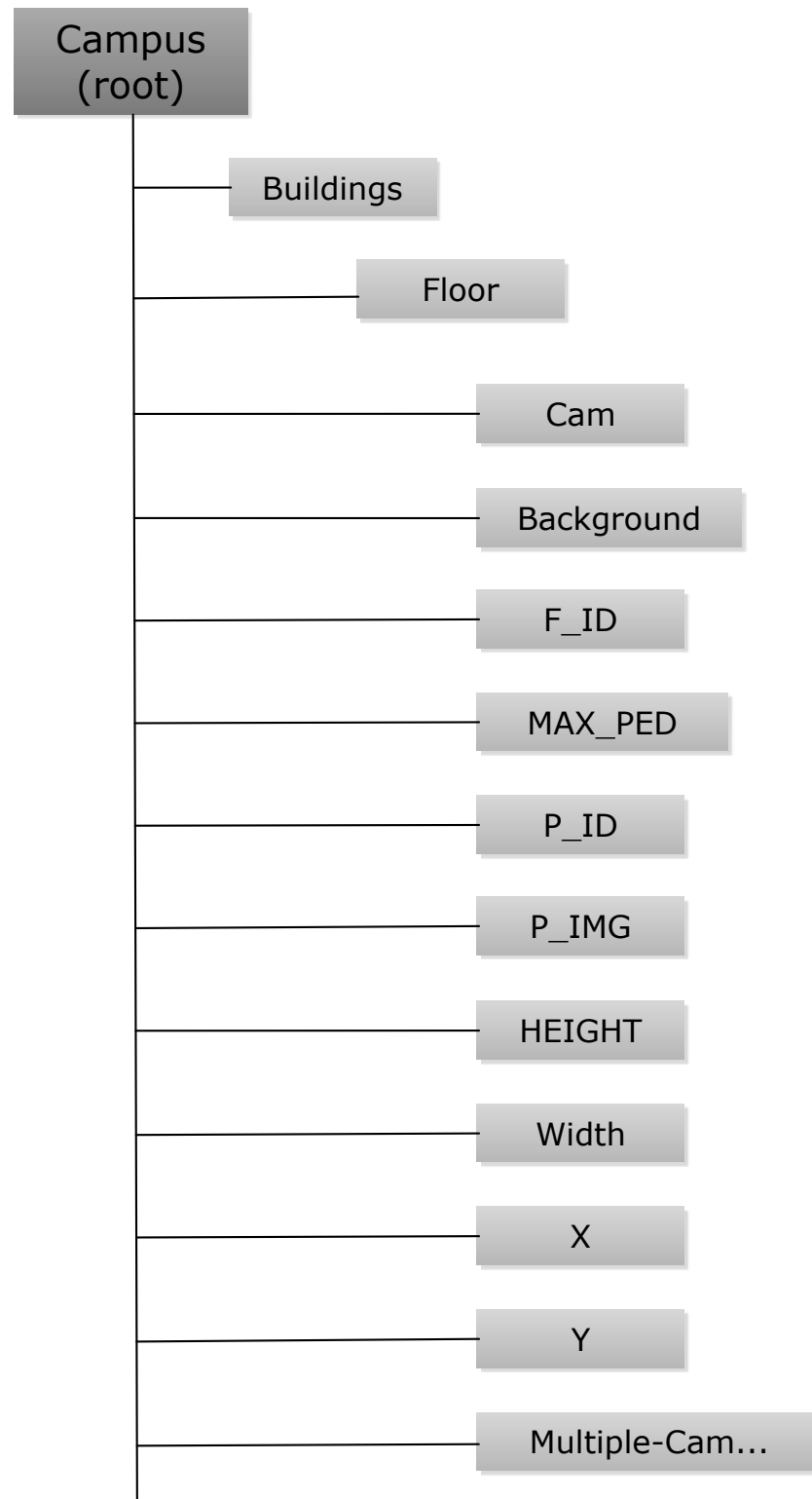
When it comes to subject of performance, Apache is very good at this aspect. Apache is hailed as one the most "high-performance" web servers; Apache provides a variety of Multiprocessing Modules (MPMs). MPMs make the web server can run in a process-based or hybrid mode; and this feature contributes a lot in Apache's high performance.

## **Video Processing Server**

The server will run a visual computer program, which analyzes a recorded video, based on the photo chromatography and algorithm input to compute and analyze the data required, for instance, to calculate how many people has occurred at a particular point within the video. After that, it will out output a JSON file send to the web server.

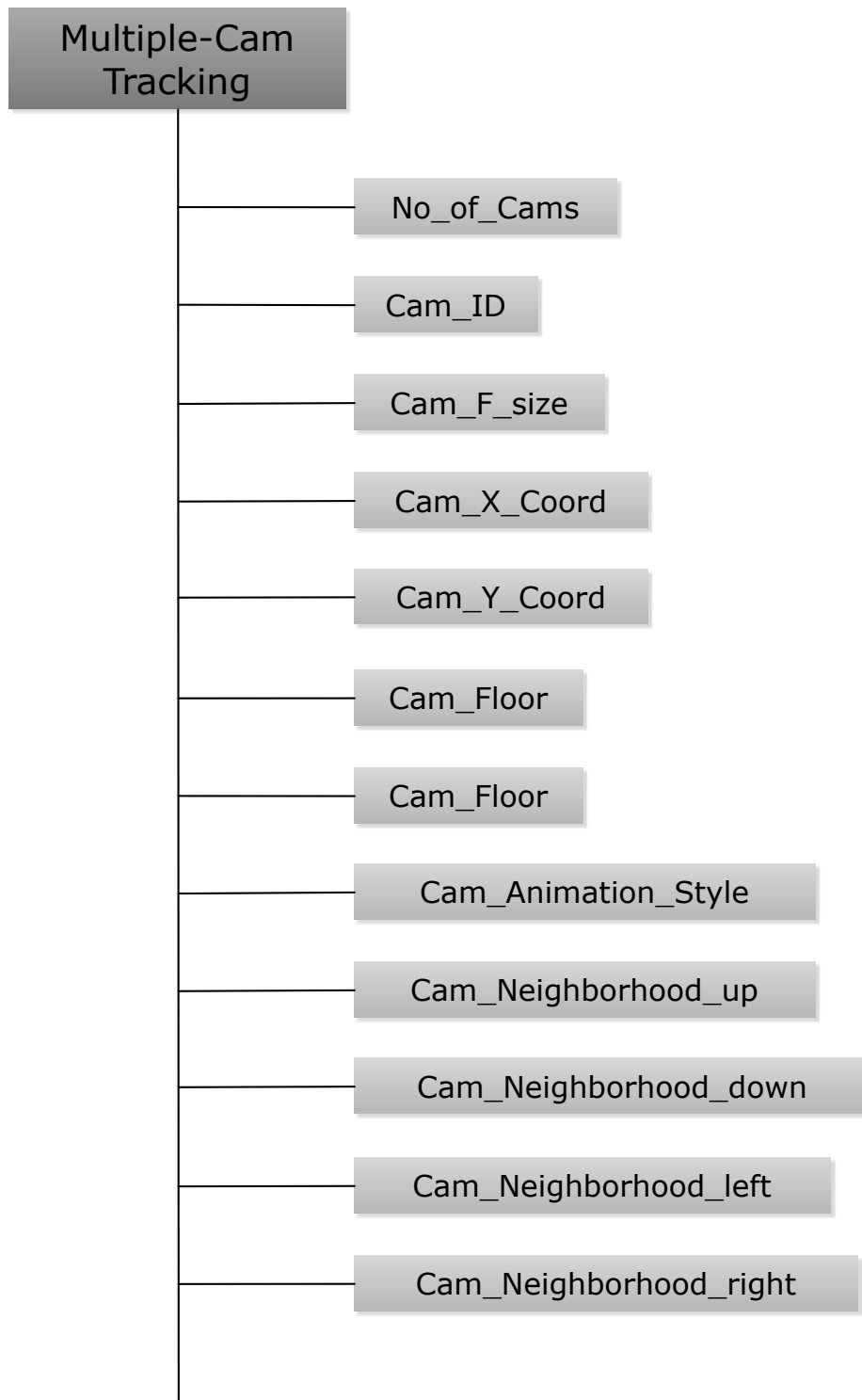
The output JSON has a simple structure shown as follows, where the branch represents the level of the node.

The following is the structure of JSON file.



*Figure 4.13 JSON Structure Diagram*

### JSON Structure Diagram – Multiple Cam Part



*Figure 4.14 JSON Structure Diagram*



This part of the JSON file stored the cam information such as number of cams, cam's neighborhood relationship, cam's display coordination and size etc.

This JSON file defines how the page should display and how each cam frames should position in Multiple-Cam page. Multiple-Cam Module would interpret this information and use them to generate the page dynamically.

As you can see in the above figure:

**Number\_of\_cams**

defines the total number of cameras the building has.

**Cam\_id**

is a variable in order to identify the cameras.

**Cam\_farme\_size**

state how large the CCTV should be showed on the Multiple Cams Moduel.

**Cam\_x\_coordination**

states the starting x coordination of the cam frame.

**Cam\_y\_coordination**

states the starting y coordination of the cam frame.

**Cam\_floor**

mentions the floor the camera belongs.

**Cam\_neighborhood\_up**

stipulates the neighbor cam, which cam should the module switch to when the user swiping up the cam frame.

**Cam\_neighborhood\_down**

stipulates the neighbor cam, which cam should the module change to when the user swiping down the cam frame.

**Cam\_neighborhood\_left**

stipulates the neighbor cam, which cam should the module change to when the user swiping left the cam frame.

**Cam\_neighborhood\_right**

stipulates the neighbor cam, which cam should the module change to when the user swiping right the cam frame.

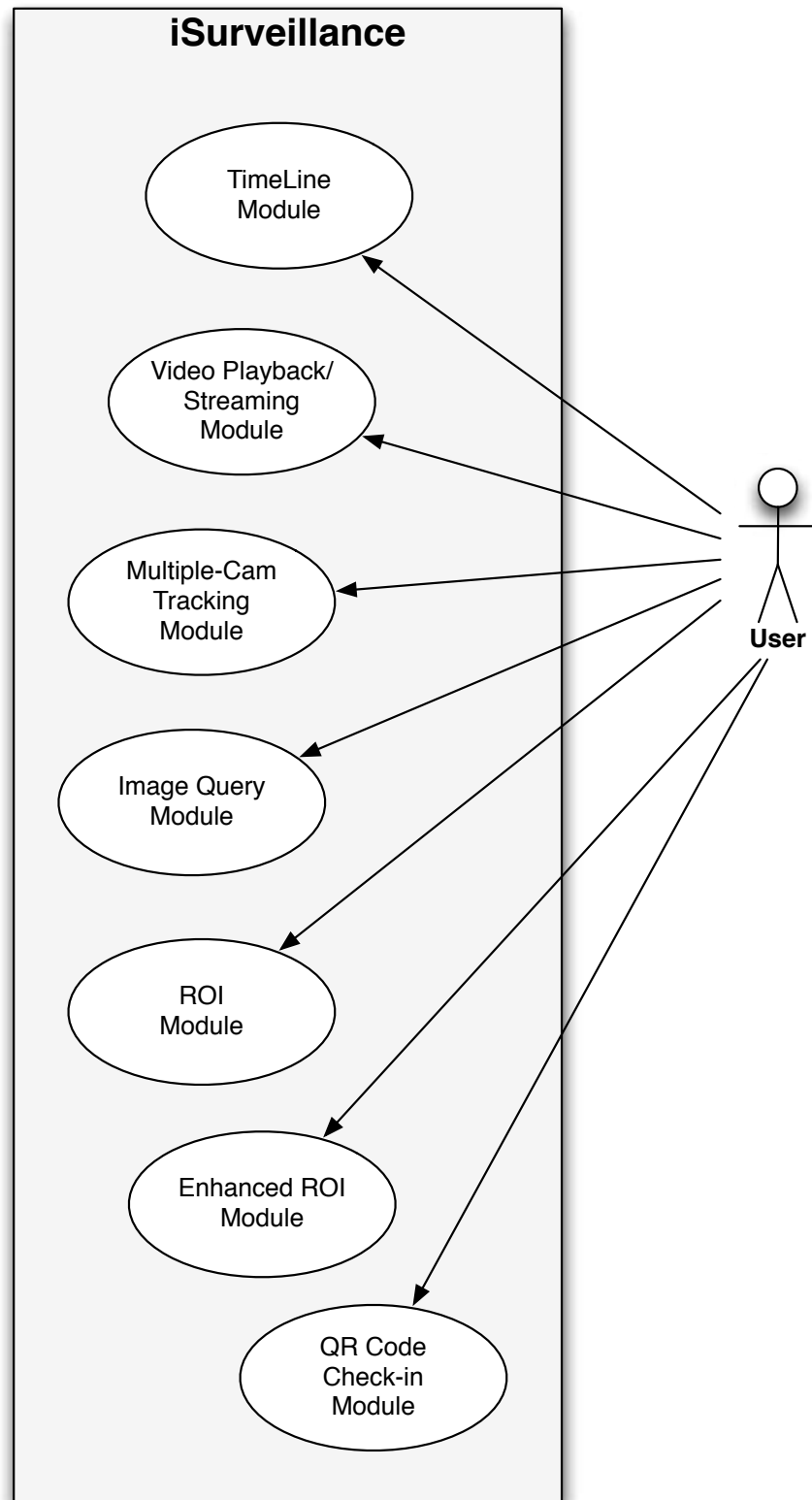
**Cam\_reminder\_notice**

states if there is any reminder notice.

**Cam\_animation\_style**

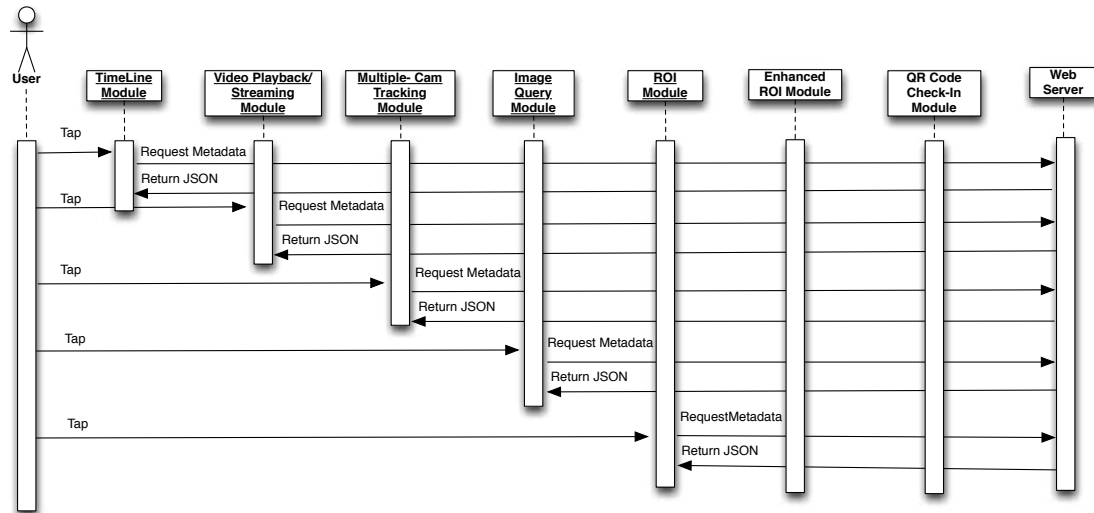
states what animation style the cam should use.

### 4.3.2 System Component Diagram



*Figure 4.15 System Component Diagram*

### 4.3.3 System Sequence Diagram



**Figure 4.16 System Sequence Diagram**

# Chapter 5 User Interface Design

## 5.1 Overview

In this chapter, we are going to report how did our user-interface be designed, what concerns have been taken into consideration and what software have been used during UI design process. We would also talk about where do we gain the UI design knowledge and then briefly describe our whole design process.

## 5.2 User Interface Design Process

### 5.2.1 Creating a Design Document

At first, like most of application developers, we were reluctant write a design document and leap directly to writing code without any sort of specification of how our application will look and behave.

However, after we have read the UI-Design books introduced above, all of them claim that it is of a extreme importance for writing a design specification before you do your coding.

As a matter of fact, we turned out and found that we can really benefit from having a specification early in the FYP project. Jotting down the behaviors of the application into documentation before you do anything, it does help!

Because it will force you to think about the complicated interactions and intricate relationship between different modules and components, so it prevent you from falling into a dark hole - which is a situation that you are difficult to make a correction because you have to dig out thousands of lines of code for just a one tiny modification.

In a nutshell, we all believe that a good design specification is the backbone of our application and will help us to make vital decisions quickly and cheaply, before they become too expensive to fix later.

So, we have jotted down a list that records the advantages and disadvantages, the various strengths and weaknesses of every feature and how a decision would affect it. These lists helped us to cross out the obvious poor choices before we ever started to do any implementation.

## 5.2.2 Diving into the Code other Details Concerns

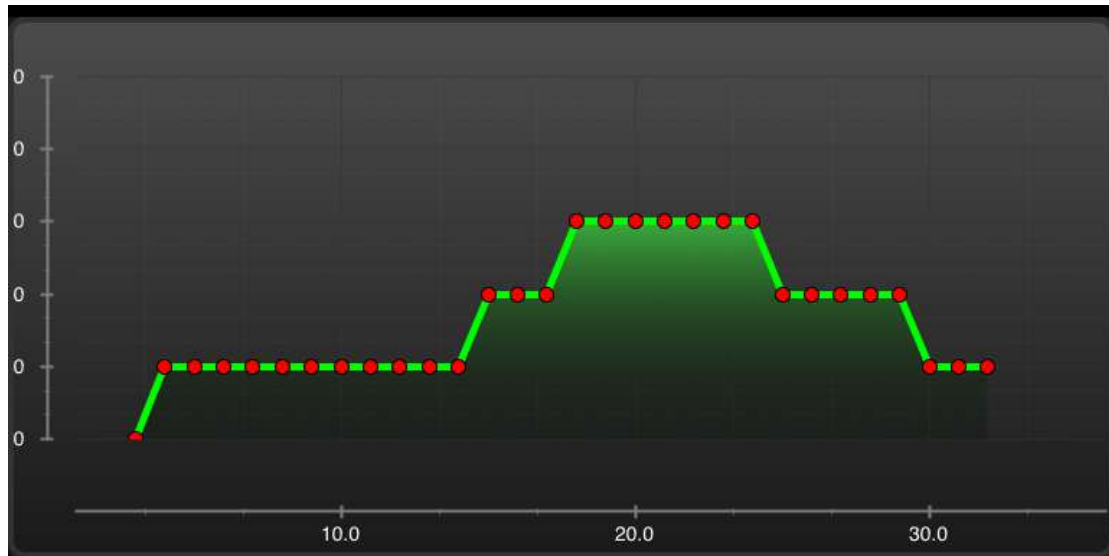


Figure 5.1 TimeLine Module

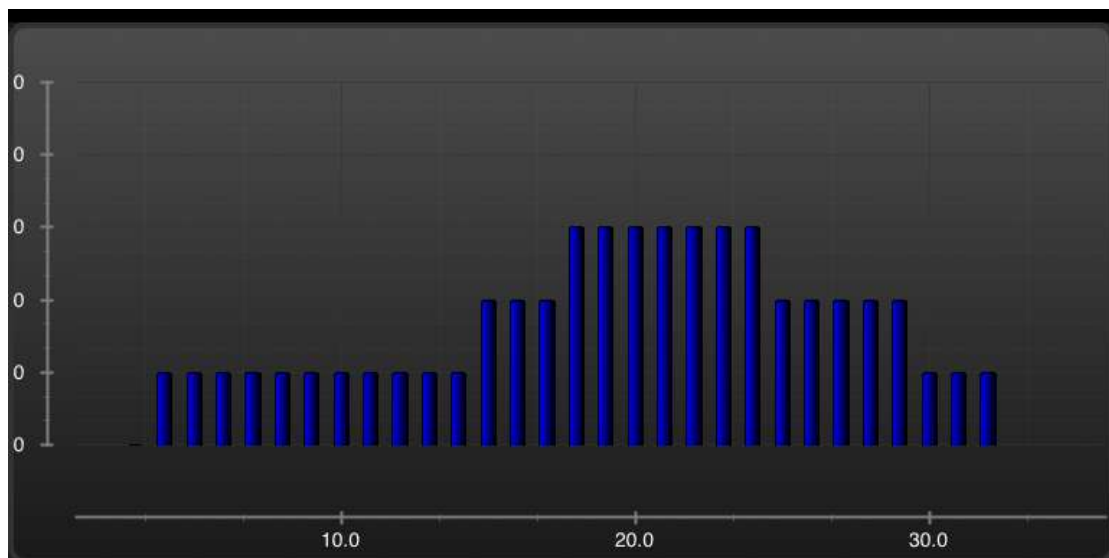


Figure 5.2 TimeLine Module

We first study on how large our plot should be for displaying on the screen, what kinds of plot should we use to achieve the best representation. Pie chart or scatter chat or bar chart? Luckily, using Core-Plot make plot generating easier;



Core-Plot has different API for generating different kinds of plot, so we tried them all. Finally, scatter chart or bar charts are used because we think they can give a best data representation.



**Figure 5.3 Thumbnail Module**

This part is about the resolution problem as we have previously mentioned on chapter 2. What resolution should we use? For a thumbnail like this, not a high resolution is required; the thumbnail here is just to display the basic shape of the image. However, when a security guard is investigating a burglary, high resolution is needed. Suspect's faces should be recognizable and readable when the guard taps the thumbnail images. So, it becomes a problem on the UI design issue, when there is a tap on the thumbnail, what should we do? To pop up a new window to show a sharper image or just refresh the image at the same location with a higher resolution one? And these are the details we have to concern. Again, it maybe trivial but means a lot in the point of view of user experience.

16:9



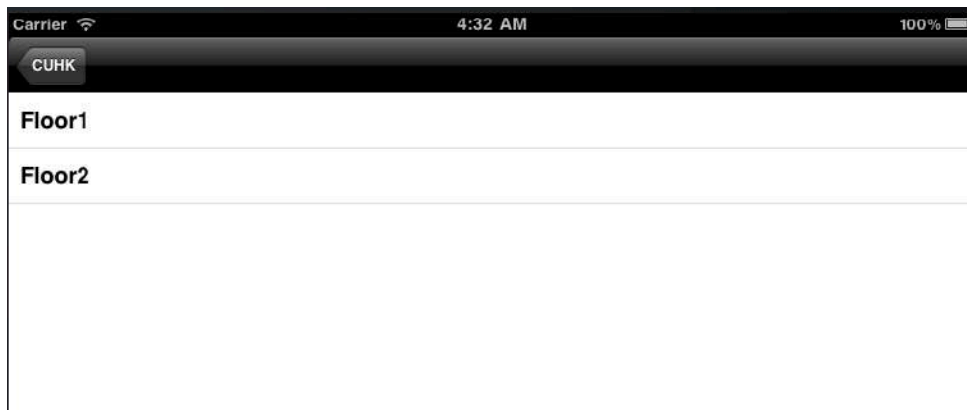
*Figure 5.4 Video Player*

For a video player, how large should it be come to our first mind? We want it as larger as it can be. However, we have to balance different factors involved; we have to consider the resolution and the ratio of the video source. We tried our best to make the video player in a ratio of 16:9 large so that it can fully utilize its area without wasting.

### **Format of the video source**

Different format would have different properties and performance; we have to take care of what format should our video playback files be. For example, an AVI format video would have a very high resolution for viewing, but the size would be too large when it comes to data transmission; while mp4 format is very suitable for mobile devices, however Apple's Media framework doesn't support it. And finally, m4v is chosen.

As we know, m4v is very suitable and optimized under iOS platform; it would be one of our best choices. Apple development frameworks are all compatible with m4v format and have lots of resource to support this format. We think it is a safer choice and would save us a lot of time.



*Figure 5.5 Floor List*

This part comes straightforward. We know that UITableView structure should be used for sure.

UITableView is a means for displaying and editing hierarchical lists of information. In fact, UITableView is a data presenting technique developed by Apple and provided in iOS. It is basic and simple, but useful and effective. Managing the Editing of Table Cells in UITableView is very developer-friendly. For example: to Inserting, Deleting, selecting, and Moving a Row in the table list, lots of functions is provided:

- beginUpdates
- endUpdates
- insertRowsAtIndexPaths:withRowAnimation:
- deleteRowsAtIndexPaths:withRowAnimation:
- moveRowAtIndexPath:toIndexPath:
- insertSections:withRowAnimation:
- deleteSections:withRowAnimation:
- moveSection:toSection:

## Cascade Design of Floor List



*Figure 5.6 Cascade Floor List*

We have hidden the floor list in the main page causes we know every single piece of space on the iPad is very precious. When we are designing the page, we have a principle --- to keep things simple and easy. So, we want use the space effectively and efficiently. We decided to hide the floor list on the left and make a “Floor Plan button” in the middle of the main page instead.

When the user tap on the “Floor Plan button”, the floor list will eject from the left showing the floor options for the user to choose.



*Figure 5.7 Cascade Floor List*

It turns out that not only can this design save us the space, but also increase the interaction of that page. When the user tap on the button, an animation effect will show and the floor list emerges. This animation definitely increases the page's user-interaction and make the page much more "exciting".

### 5.2.3 Tuning Process

As we all know, touch screen interface is one of the most unique and prominent properties iPad has. And using the touch-based gestures requires a thoughtful tuning so that an application can “feels right”. We have to take an extra care of many tiny things, for example, we have to think about how large the button we should use, not too large to waste the screen space and not too small to hinder the touch-sensitivity.

Also, we have to consider the size and the color of each component so that they can work in a perfect harmony.

## 5.3 User Interface Design Tools

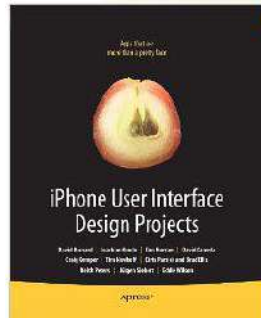
We have used a fairly large list of developer, media, and graphics applications to design and build our application, the following is the list:

- Photoshop
- Illustrator
- Xcode
- iPhone Simulator
- iPhoto

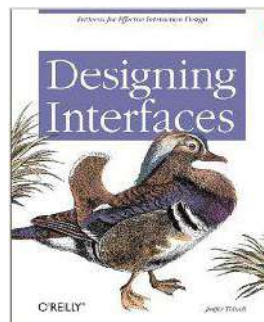


## 5.4 Books Reference

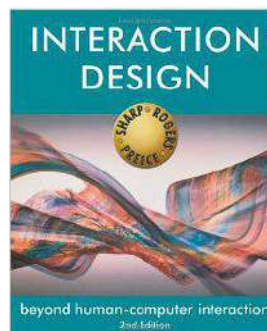
### Where we learnt how to make a Good Design



*iPhone User Interface Design Projects by Mark*



*Designing Interfaces: Patterns for Effective Interaction Design by Jenifer Tidwell*



*Interaction Design: Beyond Human-Computer Interaction by Helen Sharp*



After read the above reference books, we can conclude some principles for good User Interface Design.

1. Consistency
2. Follow the standards
3. Listen to your stakeholders
4. Explain the rules
5. User interface items is important
6. Screen size is important
7. Make labels clear
8. Understand the UI widgets
9. Color appropriately
10. Follow the contrast rule
11. Align all the fields effectively
12. Accept users to make mistake
13. Use data appropriately
14. Indubitable design
15. Don't create complicate design
16. Take an evolutionary approach

## 5.5 Chapter Summary

Putting all of our ideas together to build and develop our FYP product is really hard and involves tonnes of work.

Not only did we need to care about each and every component on our screen, but also need to consider which kind of animation, presentation-methodology should we adopt in order to make every component in our apps work in perfect harmony.

And we always think hard about the user's expectations and their preconceived notions; to put ourselves into their shoes, like, what do the user expects to see and what do the user would like to have.

Sometimes, we just have to push the boundaries of the technology we are developing with, for instance, we have overwrote the basic page-scrolling design framework Apple provided and changed it into a more "fancy" side-scrolling page.

However, sometimes when there was really a difficulty and having considered that the time is limited, we just had to throw away the idea we love and start over with a simpler method. Like, we supposed to have a three-finger swipe function; however, it is so hard for us to finish that within a limited time, so we turned out to resort to the original page-swipe function Apple provided.

To conclude, to us, a best UI design is not only about how good your interface-design is, but also how you can manage your time and limited resource you have.

# Chapter 6 Limitations and Difficulties

## 6.1 Overview

In this chapter, we are going to see what the difficulties we have encountered during the development phase and how do to tackle with that.

The difficulties we encountered are generally from three aspects: Module part, backend system and frontend application. And the problems can be further classified as two type --- software problem and hardware issue.

We would talk through the difficulties one by one and to discuss how did we find a way to get out of the morass we fell into. What strategy did we use? And what choice did we make in order to keep things fine.

## 6.2 Module part - Recorded Video Playback Module & Multiple-Cam Tracking Module

### **RTSP is not supported by iOS**

RTSP is not officially supported by the iOS development kit and this becomes a very big issue to us. As mentioned before, the whole backend system is constructed using C++ under the Microsoft Windows System. All the video analyzing module and transfer protocol are targeted as RTSP and RMVB format. So, when we want whose module operating on iPad properly, something concrete have to be done.

### **RTSP problem on Video Playback Module**

For the Recorded Video Playback Module (which has decided to use M4V format), problems here are simple and easy. In this module, we are only required to play the CCTV video and no computation is need. So the easiest and straightest forward way would be changing the backend video format to M4V. We change all the RMVB video files to M4V format in the backend storage; so all the CCTV video would be available for viewing.

## **RTSP problem on Multiple-Cam Tracking Module**

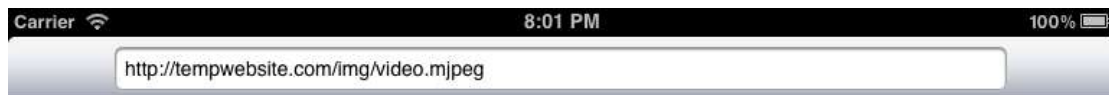
In Multiple-Cam Tracking Module, iOS not supporting RMVB format poses a very great challenge to us. For a live-streaming module, either HTTP or RTSP streaming protocol should be used for video transmission; however, as stated before, the backend system is constructed under the Microsoft Windows System and all the video related modules are using RTSP. That means, if we want to have a live streaming from the IP Cameras, we either have to reconstruct the back-end system or adopt another format the IP Cameras supports. And we chose the later.

## **Motion-JPEG Image View for iOS**

After we have looked up to the IP Cameras' specification, we found that the IP cameras which our system using supports another video stream protocol called Motion-JPEG. Motion-JPEG is basically a constantly updating stream of JPEG-compressed bitmap files and quite common on live-video streaming.

## **Implementation of Motion JPEG on iOS --- Using MotionJpegImageView**

After we decide to use this format, we have done our research on how to use this MJPEG on our FYP. Luckily, it is quite simple. You can just drop in a UIWebView in your code and point it to the IP camera's URL, and then you can get what you want, You can see the live CCTV on that UIWebView.



*Figure 6.1 UIWebView*

However, this method does not perform well, and it cannot make sizing Motion-JPEG streams easily. In a nutshell, this method may be simple and straight forward, but performs very slow and engages lots of resources. So, we looked up to the Internet to find if there is a better option; and then we discovered MotionJpegImageView.



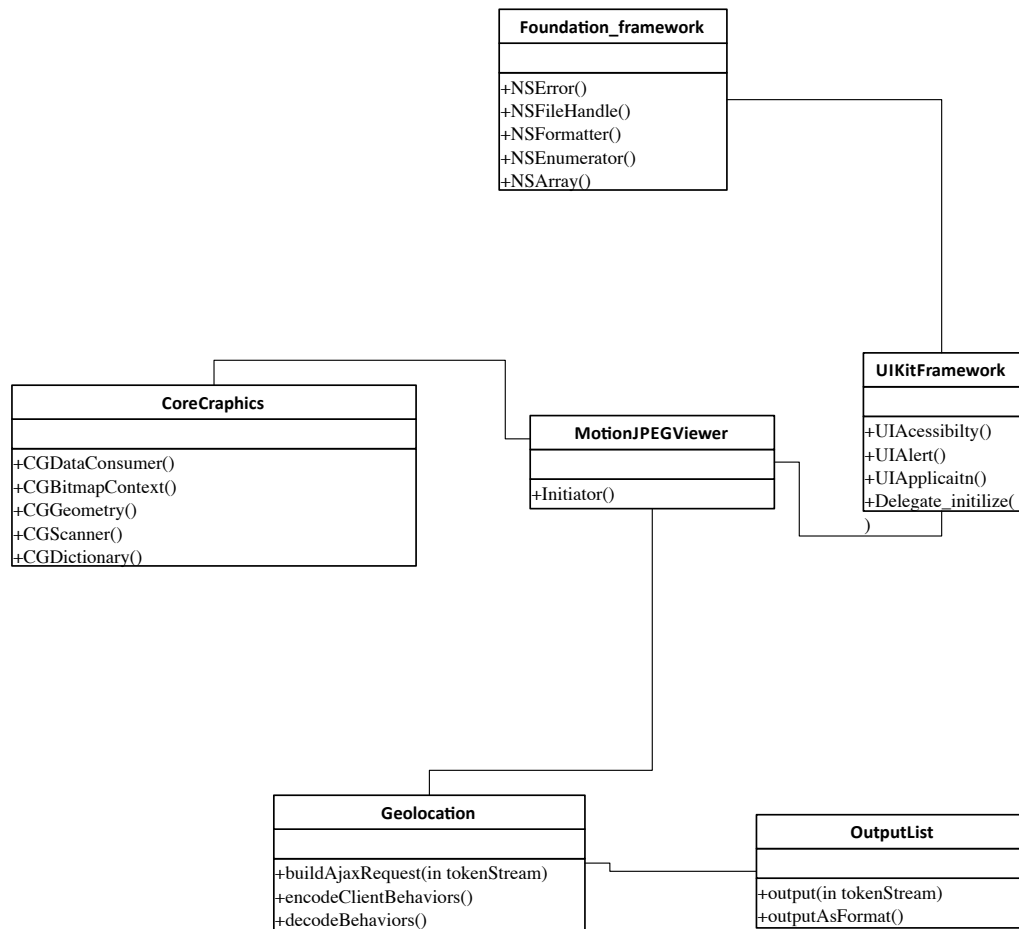
Figure 6.2 MotionJpegImageView

## **What is MotionJpegImageView**

MotionJpegImageView is a third-party framework, which provides a simple interface for creating and displaying Motion JPEG streams in an iOS application. As a matter of fact, MJPEG Steaming is just an action of constantly updating JPEG image files. So, the principle of this MotionJpegImageView is to let the UIImageView Object creates a connection to the source and receives picture frames one at a time over the network. And when it receives each frame, MotionJpegImageView will hydrate the streaming data into an UIImage object, and then sets its inherited image property.



## Class Diagram



**Figure 6.3 MotionJpegImageView Class Diagram**

## **Comparison between simple Method and Using MotionJpegImageView**



*Figure 6.4 Comparison*

P.S. (both of the streams are running at full Retina Display resolution — accommodating two 640×480 streams onscreen at once)

This part is to compare the difference between using a simple UIImageView and MotionJpegImageView for MJPEG streaming.

At the top of the figure is using a classic UIImageView stream; while the bottom is using MotionJpegImageView. It is clear to see that the latter one displays a more or less smooth video stream while the former one struggles to update the picture seconds by seconds.

So, when compared with using UIImageView, we can conclude that MotionJpegImageView is much better due to the following :

1. Much improved frame rate. MotionJpegImageView can have 7x to 8x speed improvement when compared with UIImageView .
2. Easy handling of MJPEG streams. MotionJpegImageView provides lots of library call and API functions for controlling of MJPEG streams.
3. Simple resizing. MotionJpegImageView will automatically sizes content to fit the screen while UIImageView resizing can be tricky and difficult .
4. Convenient play/pause/stop/clear options. MotionJpegImageView provides lots of library call and API functions for playing or pausing the stream quickly and easily.

## **Unsatisfied performance of EROI**

We have implemented Enhanced ROI Module in semester two. As introduced in Chapter 3, EROI is an extended class from original ROI Module that provides a searching function with more refined and accurate filtering rules.

Unfortunately, after we have constructed the EROI and tested the result, the performance is far from satisfaction.

Supposedly, the EROI module would take the value in the Intensity slide-bar as a threshold value for judging whether the data should put into the result list: only when the image sharing the overlapped area with interested area exceeds the indicated intensity value would counts as one of the result. And in fact, this part went well, our module can read the value from the Intensity slide-bar.

However, when it comes to the binding-box area coverage calculation, the result is not good and far from satisfaction. The final result list is so “ugly”. Even if we manage to calculate the overlapping area between two objects, it still can't improve the accuracy of the ROI searching.

We have consulted the problem to Mr. Edward Yau. Edward is a Research Laboratory Officer from ViewLab. And he is one of the members from the development team; he is familiar with the backend system, so we think that we can use some of his expertise.

It is concluded that the unsatisfied performance of EROI may be due to the inaccurate raw data backend system provided. There is already some error from the raw data and using those data for computation may result in a larger area. And, at the end, due to the very unsatisfactory result of EROI, we decided to shut the module down.

## 6.3 Backend Limitations

### Live Stream video analyzing is not supported

The video analyzing backend system developed by ViewLab only supported a recorded video at the moment; an analysis for a live-stream video is not supported yet. So, if we want to expand our features into a live-analyzing; it would be a problem. And, due to the fact that we are not the development team of the video analyzing program, we just co-operate with it; we can do nothing about it. We just have to wait for the system's breakthrough at the future.

### Developed in different platform

The video analyzing backend system was developed under Linux system using programming language C++ while our application is constructed by Objective C for iOS usage. A different platform can't communicate directly, so instead, we use a http request and a JSON file as communication. Our application send a http request for data-query and system return a json file as a result.

## 6.4 Frontend Limitations

### Limited resources and background knowledge


This is our first time to write an iOS apps; before that, we have no background knowledge and experience. We started from zero; it would be our great problem. We don't know if our idea is tangible and feasible; can our idea really put into practice? We don't know what limitation of iOS and objective c is. So, we have to self-study for a very long time to learn better what the advantages and disadvantages iOS and objective C have before starting coding.

And, what is more, all the data are stored in ViewLab's network. To access and retrieve those data, CSE-VPN has to be established first. However, CSE-VPN is not yet supported on iPad. And it became a bottleneck if we want to multi-task our FYP; to solve the problem --- we can only try to make use of the iPad simulator provided by XCode, but it is much time-taking and inconvenient.

### **Limited computational power of iPad**

iPad is just a mobile device and its' computation power is still very limited (even the continuous growth of mobile device's technology ) when compared to a real desktop computer. Says, even we use the latest model iPad2, it only have 1G CPU and 512 MB memory. So, we have put a lot of effort on optimization: we have modified the codes, adopt different protocols that suits iOS most, for example, we change our video format from rmvb to m4v after we know iOS perform better for a .m4v video. All these modifications are just to make sure that our application can run fine under the iOS environment.



	iPad	iPad 2
<b>Display</b>	9.7-inch LED-backlit LCD screen	9.7-inch LED-backlit LCD screen
<b>Resolution</b>	1024 x 768 pixel resolution	1024 x 768 pixel resolution
<b>Operating System</b>	iOS 4.3	iOS 4.3
<b>Processor</b>	1 GHz Apple A4 processor	1GHz dual-core Apple A5 processor
<b>Wi-Fi</b>	Yes	Yes
<b>3G Connectivity</b>	Only Wi-Fi + 3G models	Only Wi-Fi + 3G models
<b>Bluetooth</b>	Yes, v2.1 + EDR	Yes, v2.1 + EDR
<b>Camera (Rear)</b>	No	Yes, still camera with 5x digital zoom HD (720p) video recording up to 30 frames per second with audio
<b>Camera (Front)</b>	No	Yes, VGA-quality still camera VGA video recording up to 30 frames per second with audio
<b>HDMI Video Out</b>	No	Yes, (1080p) via optional accessory
<b>Storage</b>	16, 32 and 64 GB	16, 32 and 64 GB
<b>Expandable Storage</b>	No	No
<b>Battery</b>	25 watt hour rechargeable lithium-polymer battery	25 watt hour rechargeable lithium-polymer battery
<b>Weight</b>	Wi-Fi: 1.5 pounds (680 g) Wi-Fi + 3G: 1.6 pounds (730 g)	Wi-Fi: 1.33 pounds (601 g) Wi-Fi + 3G: 1.35 pounds (613 g)
<b>Size</b>	Height: 9.56 in (243 mm) Width: 7.47 in (190 mm) Depth: 0.5 in (13 mm)	Height: 9.50 inches (241 mm) Width: 7.31 inches (185.7 mm) Depth: 0.34 inch (8.8 mm)
<b>Other features</b>	Multi-touch Headset controls Proximity sensor Ambient light sensors Accelerometer Magnetometer	Multi-touch Headset controls Proximity sensor Ambient light sensors Accelerometer Magnetometer Three-axis Gyro Photo and video geotagging over Wi-Fi

**Figure 6.5 iPad Specification**



## Limitations on Objective C/ iOS

One of the intimidating things about Objective C (a programming language used in iOS) is its' memory management. Unlike some language such as Java or C++ which provides an automatic garbage collection scheme; Objective C does not have support these kind of method; rather it is by a primitive reference counting technique.

However, the primitive reference counting iOS used is prone to a memory leak problem. Objective C beginner like us, are easy to forget to retain or release an object when the program is complicated and in large-scale; thus a memory leak always happen and affect the application performance.

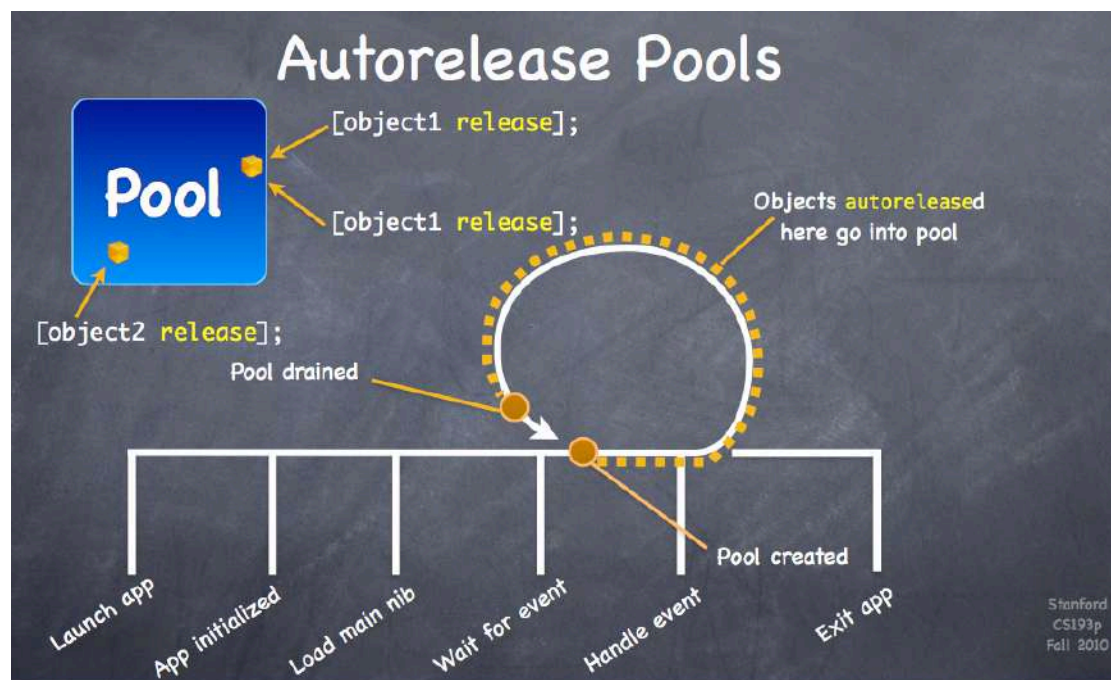


Figure 6.6 iOS Memory Management (Stanford University CS193P)

## **Insufficient Libraries and Documentation for 3rd party frameworks**

Agreed, there are lots of 3rd party frameworks /projects are available on the Internet and most of them are useful to our FYP project. However, everything comes with its flaws, many of 3rd party frameworks are lack of guidelines and tutorial so a lot of time is spent to understand how to manipulate those frameworks and how to “fuse” them into our own products. It is really time taking; like, we have to figure out on our own how to install and use applications without sabotaging our data and hardware, system configuration.



*Figure 6.7 Open Source*

# Chapter 7 Room for Improvement

## 7.1 Overview

After a year hardworking, we are glad that we have finished our Final Year Project and are quite satisfied with the work we have done so far. However, due to the limited time and manpower resource, our modules are not prefect. And if there is other time provided for us, we think we can further enhance our product and improve its performance.

In this chapter, we would take an overview on which modules we think it could be further enhanced and modified; and then explain how could we do work on that.

In general, there are four features modules still have some rooms for improvement. They are Region of Interest, ChatBox, Live-Stream Video (with RTSP) and Drag-and-drop modules.

We will also to see the possibility to adopt more APIs / third party libraries. To see which open-source libraries we are interested in, how can they further enhance our system and discuss the possibility to “fuse” them into our product.

## 7.2 ROI Module

How can we further modify and enhance the ROI?

### **Circle the suspect out, no need to draw a rectangle**

Now, our client able to draw a rectangle on the video screen to indicate his interested area for ROI video searching; this function is good and we want to keep it. However; drawing a rectangle is not user-friendly enough and maybe a little bit time-taking, So, instead of using a formal rectangle drawing as area indication, would it be a good idea to use other method to implement the same function. For example, instead of drawing a formal rectangle, it would be nice for a user to just circle the suspect out using their finger. This may be just a little modification, but we believe that it means a lot when it comes to user-friendliness.

However, this idea may seem easy, but there is a potential challenge. As far as we know, rectangle is the best shape for graphic-coordination calculation, a less error would be resulted if a rectangle is used. However, if a circle is drew by a finger, the shape may be informal and “ugly”, hard to define the area it intends to mean, hence pose a difficulty in calculation on graphic-coordination.

### **Multiple ROI at a time**

Our current product only supports one ROI search at a time, no multiple searches are allowed. We think that it would be nice if we can extend the function so it can support multiple ROI searches simultaneously at one time.

## 7.3 ChatBox Module

We haven't implemented our ChatBox Module in our FYP; but we think it would be a great idea if our product has a chatting module for communication. ChatBox module looks like a WhatsApp or Facebook chatting system. It is complicated and hard to implement. For this reason, we think that it is a good idea to absorb an open-source API called three20.

### **Three20**

Three20 is an open source Objective-C library constructed in Objective-C. Three20 is very popular and have been used by dozens of well-known brands in the App Store, like Facebook, Posterous, Pulse etc....

Three20 provides many powerful view controllers such as the Launcher, chat room module that fully fits our need. Moreover, the best thing about Three20 is the way it constructed; Three20's library is modular, you can choose extract what elements you want and include in your app, need not to integrate the whole codes. This design method is what we called 'extensions' from the community and is very popular nowadays.



*Figure 7.1 Three20 Logo*



Figure 7.2 WhatsApp

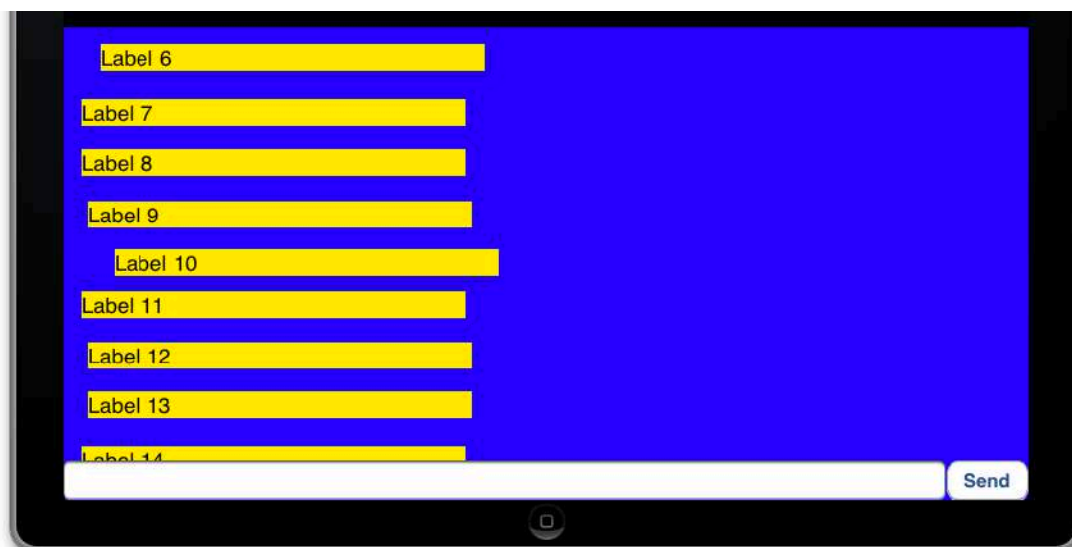


Figure 7.3 ChatBox Area

## 7.4 Live-Stream Video Module WITH RTSP

We haven't implemented our Live-Stream Module with RTSP in our FYP, because we found out that RTSP is not officially supported by iOS. However, we would still like to implement this module. It is because, if RTSP can be used for video-streaming, it would reduce a lot of work on the server side, and hence the whole performance would be much better.

And luckily, after digging on the Internet, we found that there are some third-party libraries available supporting video-streaming using RTSP on iOS. Even though it is very complicated and may be time taking, it would be a good idea to give a try.

### **What RTSP is?**

The Real Time Streaming Protocol (RTSP) is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Clients of media servers issue VCR-like commands, such as play and pause, to facilitate real-time control of playback of media files from the server.

As mentioned before, the whole backend system is constructed using C++ under the Microsoft Windows System. All the video analyzing module and transfer protocol are targeted as RTSP and RMVB format.

## 7.5 Drag and Drop Image Inquire Module

In the Image Query Module; now, if a user want to query a suspect, he/she have to “long-press” the image in order to activate the Module Window. But we think it would be a good idea if we implement a more user-friendly and convenient way for Image-Query activation. What if a user can just Drag and Drop those images? Is it much “fancier” and user-friendlier?

So, we decide to do like this:

User can drag a thumbnail image into the designated area, and an automatic search will be done then. And for this module, we think DragKit would be a great option.

### What DragKit is?

DragKit is a way to make iPhone/iPad support the application of open source libraries drag operation. Just click the icon to pull the icon twice, you can see the drag of the instructions. Many examples of applications, one is to pull a song like album. Users can also drag buttons and toolbars.

Mobile device applications and desktop applications is generally the biggest difference is that you can present their intuitive operation interface, the user will not have the time and patience to read the instructions, then simple interface it is very important. This is to introduce the iPhone support drag operation DragKit.



*Figure 7.4 dragKit Logo*



## Chapter 8 Conclusion

Now, my partner, and I have finished the FYP and we are so glad about this. The following is my conclusion for the whole FYP, to say what I feel for this final year project and what I have learnt through it. In term of the division of labor, cause Gavin think that I am a person with a greater prudence and discretion and I believe Gavin may have a better experience in programming. So, I decide to do the JSON file part and the basic framework; while Gavin is responsible for the function modules.

For my part, designing a JSON file is easy. But designing an effective and efficient JSON file architecture could be very troublesome and sophisticated, lots of factors have to be taken into consideration when design the format as I want my JSON file can perform well and fast. I have looked up some books to help me on this design problem. Reading books like how to make a fast and effective data exchange using the least resource and CPU-time. Because all of these require lots of technically knowledge and concepts and I am glad that I have a chance of learning it. Besides the actual work, I think the “soft-problem” part is very meaningful to me. In my point of view, what I have to do for our FYP looks like a real business. We have to write a frontend application to co-operate with the backend system, to design and implement on our own selves, we have to consider many things like user-friendliness, performance optimization and even time-management; it all looks like a real business. And I believe what we learned is very practical and invaluable for our future.

Finally, now, all the major modules and the basic framework have implemented successfully and we are satisfied with the overall result. However, due to the limited time we have, our work is not perfectly perfect at the moment. And if there is more time provided for us, we think we can further enhance our product to make it more “fantastic”.

## Chapter 9 Acknowledgment

We would like to take this rare opportunity to express our thanks to Prof. Michael Lyu, our supervisor of the final year project. Prof. Lyu gave us lots of invaluable suggestions and consultation over the past semester; the guidance he gave played a crucial role to our FYP.

Also, we would like to extend our sincere thanks to Mr. Edward Yau. Mr. Yau is a Research Laboratory Officer from ViewLab. And since he is one of the members of Visual-analyzing program development team, he is familiar with the backend system. Mr. Yau gave his expertise over our FYP design, share his view on how our FYP should be implemented to fully co-operate with the backend.

Last but not least, we are grateful to the ample supply ViewLab and Prof. Lyu provided. We are very thankful to have a latest mode iPad2 for our application development; and really appreciate for the another coming iPad2 that will be provided to us in the next semester; after Prof. Lyu noticed our FYP process is bottlenecked by the limited resource.

## Chapter 10 Contribution of Work

Item	Alex	Gavin
<b>Background Research</b>		
Background Analysis	O	
Case Study		O
Consultancy Seeking		O
<b>Modules Implementation</b>		
JSON Design	O	
Map Kit Module	O	
TimeLine Module	O	
Thumbnail Module	O	
Multiple-Cam Tracking Module		O
Image Query Module		O
ROI Module		O
EROI Module		O
QR Code Module		O
<b>Others</b>		
Libraries Importation		O
Graphic Design	O	
<b>Report Editing</b>		
Introduction	O	
Case Study		O
System Review	O	
System Design		O
User Interface Design	O	
Limitation and Difficulties	O	O
Rooms of Improvement		
Conclusion	O	O

# Chapter 11 References

- [1] <http://niw.at/articles/2009/03/14/using-opencv-on-iphone/en>
- [2] <http://news.hk.msn.com/local/article.aspx?cp-documentid=5188546>
- [3] <http://www.radcommuk.co.uk/video-monitor-wall-solutions/>
- [4] Mark - *iPhone User Interface Design Projects*
- [5] Jenifer Tidwell - *Designing Interfaces: Patterns for Effective Interaction Design*
- [6] Helen Sharp - *Interaction Design: Beyond Human-Computer Interaction*
- [7] <http://wiki.cse.cuhk.edu.hk/wikis/viewtech/laboratory/viewlab>
- [8] <http://www.apple.com/ipad/>
- [9] <http://www.apple.com/hk/ios/>
- [10] <http://searchmobilecomputing.techtarget.com/report/Mobile-operating-systems-Which-mobile-device-platform-fits-your-strategy>
- [11] <http://www.uniforce.com.hk/?content=product&lang=en>
- [12] <http://www.geovision.com.tw/english/index.asp>
- [13] <http://www.2mcctv.com/>
- [14] <http://www.cctvcamerapros.com/>
- [15] <http://www.surveillance-video.com/comsys.html>
- [16] <http://installingsecurity.com/surveillance-system.html>
- [17] <http://www.videosurveillance.com/>
- [18] <http://igadgetlife.com/information/10-reasons-why-ios-beats-android/>
- [19] <http://macdailynews.com/2011/08/05/why-ios-development-is-winning/>
- [20] <http://gigaom.com/apple/a-strong-argument-for-why-ios-development-is-winning/>
- [21] <http://www.gottabemobile.com/2011/06/07/why-ios-5-isnt-enough-to-make-me-switch-from-android/>
- [22] <http://www.ibtimes.com/articles/231724/20111015/apple-iphone-4s-ios-5-smartphone-os-best-gesture-reasons-features-touch-universal-inbox.htm>
- [23] <http://www.wintechmobiles.com/iphone/why-ios-5-is-better-than-android-gingerbread-2-3-3/>
- [24] <http://www.android.com/>

- [25] <http://successbeginstoday.org/wordpress/2011/10/multi-touch-gestures-on-the-ipad-with-ios5/>
- [26] <http://www.mobileburn.com/definition.jsp?term=multi-touch>
- [27] <http://www.multitouchtechnology.com/>
- [28] <http://www.hendonpub.com/resources/articlearchive/details.aspx?ID=2079>
- [29] [http://www.canopy-wireless-solutions.com/Solutions/Documents/video\\_surveillance\\_brochur.pdf](http://www.canopy-wireless-solutions.com/Solutions/Documents/video_surveillance_brochur.pdf)
- [30] <http://www.procom2way.com/video-surveillance.htm>
- [31] <http://www.json.org/>
- [32] <http://en.wikipedia.org/wiki/JSON>
- [33] <http://www.xfront.com/json/index.html>
- [34] <http://code.google.com/p/core-plot/>
- [35] <http://www.switchonthecode.com/tutorials/using-core-plot-in-an-iphone-application>
- [36] <http://www.e-string.com/content/simple-graph-using-core-plot>
- [37] <http://troybrant.net/blog/2010/01/set-the-zoom-level-of-an-mkmapview/>
- [38] [http://developer.apple.com/library/IOs/#documentation/MapKit/Reference/MKMapView\\_Class/MKMapView/MKMapView.html](http://developer.apple.com/library/IOs/#documentation/MapKit/Reference/MKMapView_Class/MKMapView/MKMapView.html)
- [39] <http://blog.vayen.ch/2011/02/01/tips-for-better-mkmapview-performance/>
- [40] <http://developer.apple.com/devcenter/ios/index.action>
- [41] <http://uxdesign.smashingmagazine.com/2009/01/19/12-useful-techniques-for-good-user-interface-design-in-web-applications/>
- [42] <http://toastytech.com/guis/uirant.html>
- [43] <http://www.stanford.edu/class/cs193p/cgi-bin/drupal/>
- [44] [http://en.wikipedia.org/wiki/Streaming\\_media](http://en.wikipedia.org/wiki/Streaming_media)
- [45] <http://sourceforge.net/>
- [46] [http://www.motorola.com/Business/US-EN/Business+Solutions/Product+Solutions/Incident+Command+Systems/Wireless+Video+Surveillance\\_US-EN](http://www.motorola.com/Business/US-EN/Business+Solutions/Product+Solutions/Incident+Command+Systems/Wireless+Video+Surveillance_US-EN)
- [47] "QR Code features". Denso-Wave. Retrieved 3 October 2011.
- [48] Borko Furht (2011). Handbook of Augmented Reality. Springer. p. 341.
- [49] "QR Code — About 2D Code". Denso-Wave. Retrieved 3 October 2011.

- [50] "QR Code Standardization". QR Code.com. Denso-wave.com. Retrieved 23 April 2009.
- [51] "AIM Global Online Store". Aimglobal.org. Retrieved 23 April 2009.
- [52] "Synchronization with Native Applications". NTT DoCoMo. Retrieved 17 February 2009.
- [53] "Barcode Contents". zxing – A rough guide to standard encoding of information in barcodes. Retrieved 17 February 2009.
- [54] <http://www.mobilemarketer.com/cms/news/software-technology/11930.html>
- [55] <http://blogs.vancouversun.com/2012/01/04/tescos-cool-qr-code-advertising-campaign/>
- [56] "Google Chart Tools".
- [57] "QR Code Readers for iPhone, Android, Blackberry and Windows Phone 7".
- [58] "Geo Tagged QR Codes". Retrieved 27 October 2011.
- [59] "Nokia Europe – Nokia N80 – Support".
- [60] "package overview for mbarcode". Maemo.org. Retrieved 28 July 2010.
- [61] <http://www.physorg.com/news/2010-11-phone-friendly-codes-ads.html>
- [62] <http://www.internetretailing.net/2011/08/14m-americans-scanned-qr-and-bar-codes-with-their-mobiles-in-june-2011/>
- [63] "Korea's Tesco reinvents grocery shopping with QR-Code "stores"".
- [64] "Korea's Tesco reinvents
- [65] "Version and Maximum capacity table". Denso-Wave.
- [66] "QR Droid". Google. 19 August 11. Retrieved 5 September 11.
- [67] "Encrypted QR Codes". QR Droid. 24 October 11. Retrieved 5 September 11.
- [68] Orli Sharaby (18 October 2010). "Form Meets Function: Extreme Makeover QR Code Edition". Retrieved 29 July 2011.
- [69] Hamilton Chan (18 April 2011). "HOW TO: Make Your QR Codes More Beautiful". Retrieved 29 July 2011.
- [70] "QR Code.com". Denso-wave.com. 6 November 2003. Retrieved 23 April 2009.

- [71] "UK QR Code Trademark".
- [72] "EU QR Code Trademark".
- [73] "US QR Code Trademark".
- [74] "US alternative QR CODE Trademark".
- [75] a b "Jargon Watch", Wired 20 (1): 22, Jan 2012.
- [76] "Malicious Images: What's a QR Code". SANS Technology Institute. 3 August 11. Retrieved 31 August 11.
- [77] "Barcode Scanner". Google. 1 June 11. Retrieved 31 August 11.
- [78] "QR Droid". Google. 19 August 11. Retrieved 31 August 11.
- [79] "ScanLife Barcode Reader". Google. 24 May 11. Retrieved 31 August 11.
- [80] "Consumer Alert: QR Code Safety". Better Business Bureau. 23 June 11. Retrieved 31 August 11.
- [81] "AVG Cautions: Beware of Malicious QR Codes". PC World. 28 June 11. Retrieved 31 August 11.
- [82] "EvilQR – When QRCode goes bad". AppSec-Labs Blog. 14 August 2011. Retrieved 31 August 2011.
- [83] "QR Codes: A Recipe for a Mobile Malware Tsunami". Cyveillance, Inc. 20 October 2010. Retrieved 31 August 2011.
- [84] QR Codes hold up to 2.9 KB whereas the smallest known computer virus is about one-tenth that size "The Smallest Virus I Could Manage". Virus Labs and Distribution. 1995. Retrieved 31 August 2011.
- [85] "Beware of Malicious QR Codes". ABC. 8 June 2011. Retrieved 31 August 2011.
- [86] [http://www.pelco.com/sites/global/en/products/video-management-solutions/range-presentation.page?c\\_filepath=/templatedata/Offer\\_Presentation/3\\_Range\\_Datasheet/data/en/shared/video\\_management\\_solutions/net5301\\_tc.xml](http://www.pelco.com/sites/global/en/products/video-management-solutions/range-presentation.page?c_filepath=/templatedata/Offer_Presentation/3_Range_Datasheet/data/en/shared/video_management_solutions/net5301_tc.xml)
- [87] <http://www.fileformat.info/format/bmp/spec/b7c72ebab8064da48ae5ed0c053c67a4/view.htm>
- [88] <http://www.matroska.org/technical/specs/codecid/index.html>
- [89] Apple - Developer - Graphics & Imaging Overview". Retrieved 2007-02-12.

- [90] "Apple - Developer - Graphics & Imaging Reference". Retrieved 2007-04-17.
- [91] "Apple - Developer - Quartz Programming Guide for QuickDraw Developers: Drawing Destinations". Retrieved 2007-02-12.[dead link]
- [91] "Apple - Info - Docs - About the Mac OS X 10.4.3 Update (Delta)". Retrieved 2007-09-20.
- [93] "Apple - Developer - Graphics & Imaging Overview". Retrieved 2007-02-12.
- [94] "Mac OS X DP4 Inside Quartz". Retrieved 2011-09-07.
- [95] Paquette, Mike. "Why Apple didn't use X for the window system". Retrieved 2006-12-23
- [96] "About the Apache HTTP Server Project". Apache Software Foundation. Retrieved 2008-06-25.
- [97] "Apache HTTP Server". Ohloh.net.
- [98] Netcraft Market Share for Top Servers Across All Domains August 1995 - November 2009
- [99] "February 2009 Web Server Survey". Netcraft. Retrieved 2009-03-29.
- [100] [https://secure1.securityspace.com/s\\_survey/data/man.200907/apacheos.html](https://secure1.securityspace.com/s_survey/data/man.200907/apacheos.html)
- [101] "March 2012 Web Server Survey". Netcraft. Retrieved 2012-03-05.
- [102] "Why the name 'Apache?'". HTTPd Frequently Asked Questions.
- [103] Apache MPM worker
- [104] [http://people.apache.org/~jim/presos/ACNA11/Apache\\_httpd\\_cloud.pdf](http://people.apache.org/~jim/presos/ACNA11/Apache_httpd_cloud.pdf)  
Apache httpd 2.4
- [105] OpenBSD however may ultimately replace Apache with Nginx, a 2-clause BSD licensed web server.