Acknowledgement

We would like to express our gratitude to our project supervisor, Professor KING Kuo Chin, Irwin who has given us many useful advices on the project. Professor KING consistently provides us with guidance and leads us to the right direction with his invaluable suggestion.

We would also like to thank him and the department of Computer Science and Engineering for encouraging us to join the Apple World Wide Developer Conference 2009 (WWDC2009); and Apple Inc. for providing this great opportunity for us. The journey for WWDC2009 has given us a precious experience to meet other iPhone developers from all over the world and inspirations for our project.

Index

1 Introduction

1.1 Overview

Owing to the rapid development on microprocessor technology, smartphone is getting more popular nowadays. The iPhone is an Internet and multimedia enabled smartphone designed and marketed by Apple Inc. The first generation iPhone was introduced in January 2007 and Time magazine named it the Invention of the Year in 2007. Within 3 years from the first iPhone released, Apple (17.8%) remains third place behind Nokia (39.7%) and RIMs Blackberry (20.6%) in the current market share for smartphone in third quarter 2009. Although iPhone wasn't the first smartphone or even touch-phone, it offers an amazing user-interface in a well-designed handset that gives inspiration to the trend of smartphone design.

Until the second quarter of 2009, a total of 33.75 million iPhones have been sold. This huge selling figure has proved the success of iPhone and the great market for developing iPhone application. There are many useful and handy features in iPhone, like the multi-touch interface, accelerometer, maps and GPS and the intelligent keyboard. All these awesome features greatly enhance the creativity of application design in mobile phone.

With the very convenient application purchasing platform, iTune App Store, and users can easily browse and download their desired applications onto their iPhone or iPod Touch. According to the latest sales figure released in November 2009, 57 million iPhones and iPod Touches were sold in the last two years. There are now over 1 million applications available in the App Store and up till November, there are over 2 billion app downloads from the App Store. It has brought much profit to Apple Inc. and the developers. Such a huge market in for iPhone application definitely encourages the iPhone application development.

1.2 Motivation

Since iPhone application market is ongoing expanding every single day, it provides considerable opportunities for developers, ranging from large-scaled company to individuals, to develop innovative mobile applications. It would be a fascinating challenge for us to develop a stunning and user-friendly application to enhance the functionality of iPhone.

Comparing iPhone with other mobile devices, there is still some space for improvement. Instances are better add more features that iPhone doesn't have face and smile detection for camera. Therefore, we would like to develop an iPhone application that can strengthen the functionality of camera in iPhone.

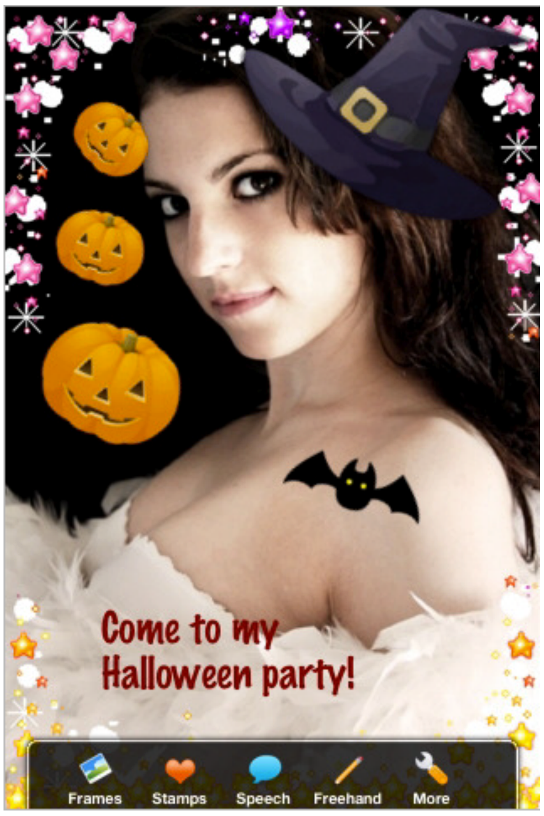
1.3 iPhone Application Review

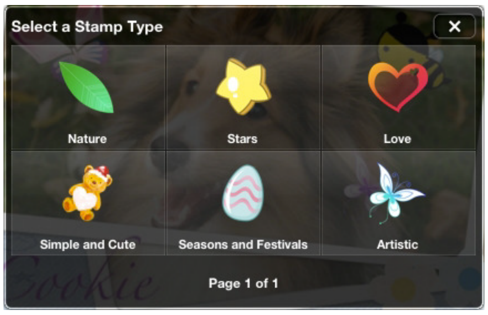
In the early stage of designing the iPhone application, we have compared three different popular photo applications released from the iTunes App Store. Via the comparison of the strengths and weaknesses between different applications and the ways they are implemented, we managed to design a better one that can include the strength of them and solve their weaknesses at the same time.

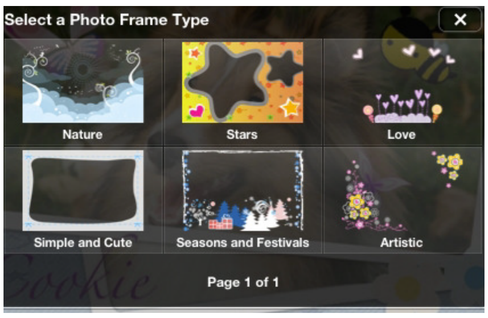
The reviewed photo applications are Puri!, PhotoFunia and Polarize.

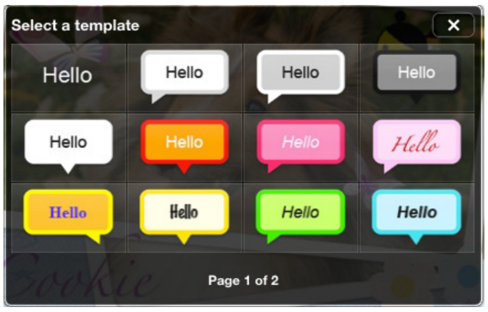
1.3.1 Puri! By ThinkBulbs Ltd.

It is a photo sticker application that provides users with high-quality collection of frames and stamps to decorate their photos into "purikura". It is very convenient to use and all user needs to do is a simple tap on the touch screen. User can also draw lines and add speech bubbles onto the photo. However, as the artwork provided required a lot of memory space, thus the file size of the application exceeds 10MB, which the application can only be downloaded via wifi connection or computer.





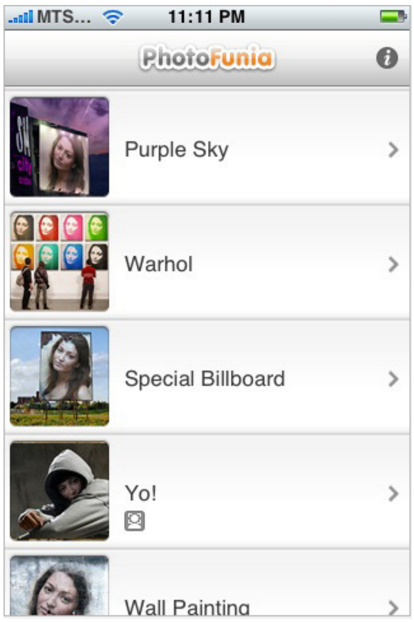


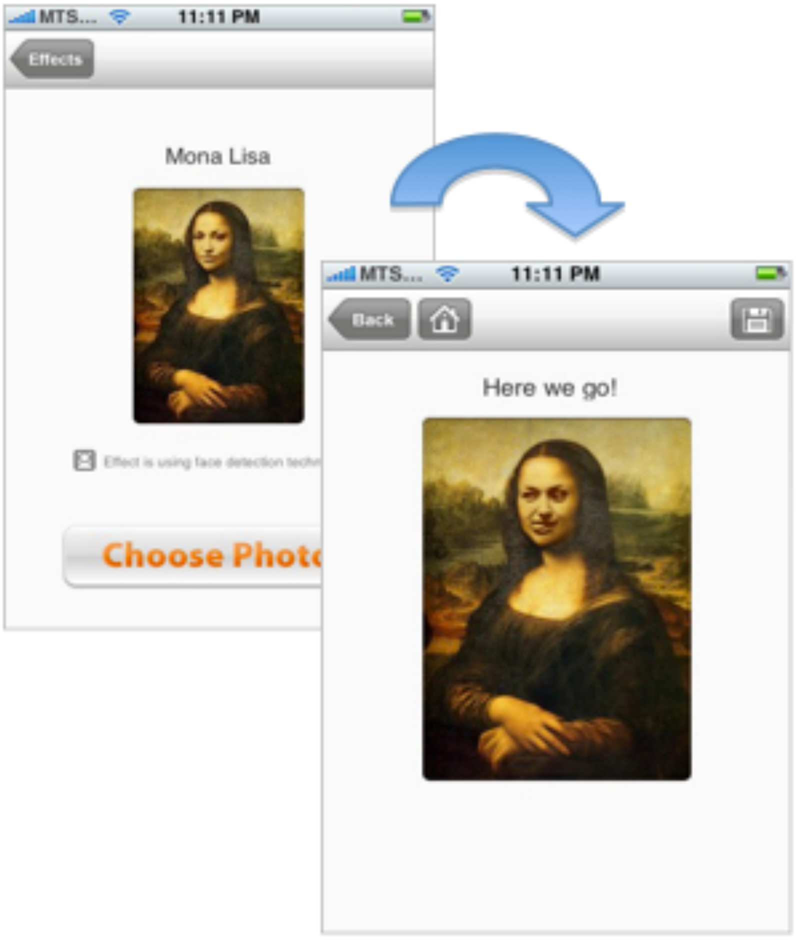


1.3.2 PhotoFunia by Alexey Ivanov

It is an online photo-editing tool. Users can select the effect they like from over 100 different effects, then pick a picture from Camera Roll or take one with iPhone's built-in camera. The application will then upload the photo to the server, the server will automatically do the face identification in the photo, add the effect selected to create a funny face photo montages.

Though there are over 100 themes that user can choose from, it also requires user to wait for some time in order to load the different effects from the server. The face detection in this application is done by the server, so it needs the use of Internet for sending and downloading the photo to and from the server, otherwise, it cannot perform the face detection and add the according effect in the photo. In addition, the face identification technology in this application can only detect single face.

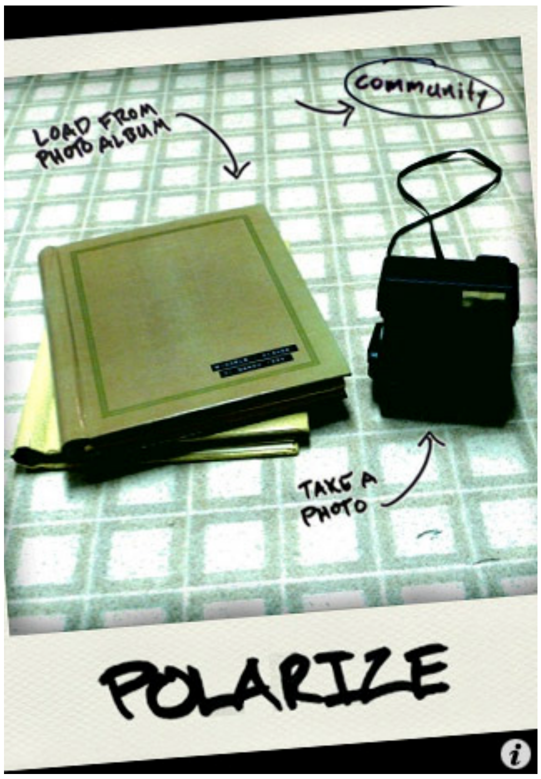


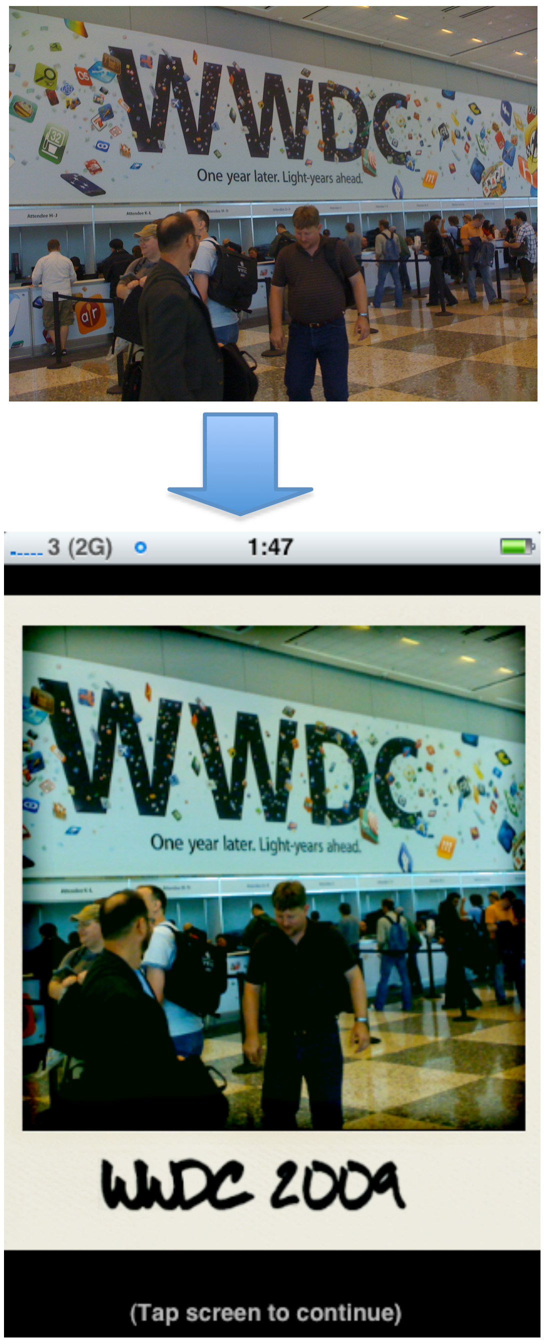


1.3.3 Polarize by Christopher Comair

Polarize allows users to add a funky retro look onto their photos. User can select photo from the photo album or take one immediately. The application will then do a series of pixel-by-pixel modifications to the photo; make it like a real Polaroid. In addition, there is a tagging function that users can write at the bottom of their photos with a custom hand written font.

User can also check out the Polarize Flickr group which is a community of users all enjoying this application. There user can also have direct access to the developers, for comments, feedbacks, suggestions.




WWDC
One year later. Light-years ahead.
3 (2G)
1:47
WWDC
One year later. Light-years ahead.
WWDC 2009
(Tap screen to continue)

2 Application Design

2.1 Ideas Brainstorming

Via comparison between different photo applications, altogether four draft ideas came up in our mind at the very beginning of the design phase.

2.1.1 Photo Booth application – include all the awesome features of Photo Booth from Mac Book

- It uses a built-in video camera to take pictures with a certain special effect, such as Bulge or Coloured Pencil
- It can also take video clips with special background, like roller coaster, space…etc
- The effects are chosen before you take the picture, and the preview shows the effect already applied before you save the photo or video.

2.1.2 Be the Chinese Hero!

- A photo application that can detect your face and map your face to some famous Chinese hero's body, provided with the history background of that hero. User can have better understanding of Chinese history at the same time.
- The face detection technology is developed inside the application, unlike PhotoFunia, it doesn't need the use of internet.
- You can either select your hero by yourself OR the system can find the hero that looks the most familiar with you
- You can upload your photo in the Facebook, send it to your friend via blue tooth, print it via some local photo printing store

2.1.3 Party Games – includes lots of micro-games like Warioware in Wii

- A puzzle game focusing on micro-games, which are short games that generally less than 5 seconds long
- Most games present instructions in the form of a verb and quickly drop the player into the situation where they must perform said verb
- This game allows great creativity by design numerous mini games. Many unique iPhone features can be used in the game design.

2.1.4 Ghost Hunter – Come and Save your friend!

- Use the built-in iPhone camera to take a snap of your friends
- Ghost will appear around your friends in your iPhone, you have to kill the ghosts by different actions like tapping the screen, shaking the phone, sliding the energy bar …etc
- It can utilize the features provided, like 3GS built-in camera, accelerometer, multi-touch… etc
- There is no such game that uses the camera to play game in the App Store yet. It would be a brand new idea in designing game in iPhone.

2.2 Final Design

Among the ideas that we have brainstormed, they share some common features, like using the built-in video camera and face detection. Our target is to develop an application that is original, meanwhile, entertaining. Our final design is to develop a photo application, not only includes all the nice features provided from the popular photo application, but develop an even better one.

2.2.1 Specification of iPhone photographic application - SNAP!

Introduction

The well-developed iPhone, undoubtedly, is the user-friendliest mobile device ever existed. To make iPhone to be a more comprehensive device, a new iPhone photo application, SNAP!, is going to be introduced a real-time face tracking camera to all iPhone users. SNAP! is a photo editing tool that can give user a fun experience. With the auto face-tracking camera, user can easily take nice snap shots with their friends at anywhere anytime! Besides, user can also select the photo from the album or take one by using the built-in iPhone camera. With the face identification technology, user can add photo editing effect easily and create funny face photomontages.

Apart from the automatic face detection features, to trim up your photos, SNAP! also allow user to decorate their photo with different frames or stamps. After user finishing the decoration, they can publish their photo to facebook by uploading it or share it to their friends through email.

Objective

- To allow user to enjoy and having fun when taking photos
- To allow user to create funny photo edit photos and add some effect on it
- To allow user to share photos on Facebook or through email

Target Features:

Elementary features:

- Load image from Photo Library
- Save image into Photo Library
- In-app photo taking
- Scaling and Cropping of the photo
- Static face detection on Photo

Advanced features:

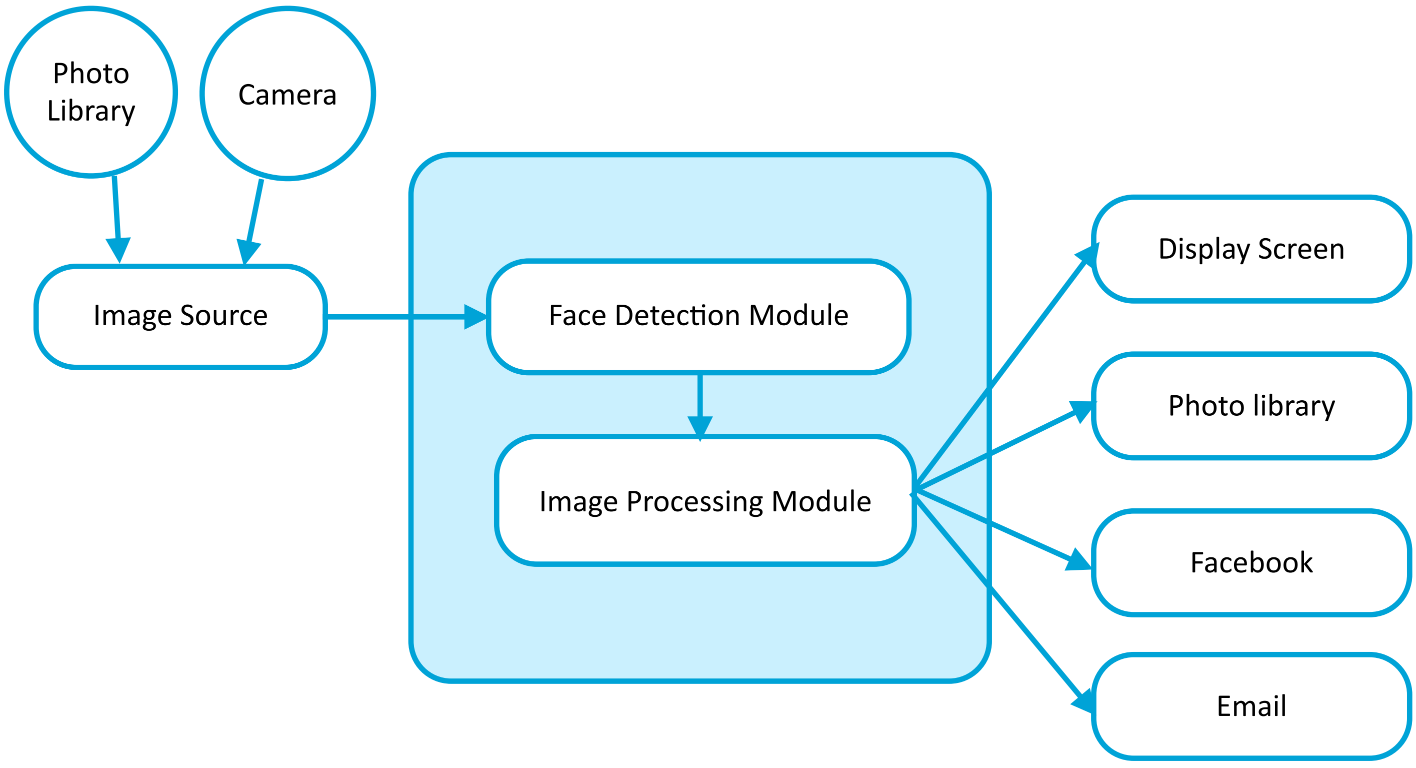
- Face detection on static photo (Single Face/Multi Faces)
- Real-time face tracking camera
- Image Layering for adding stamps, frames and text boxes
- Filtering
- Export to facebook
- Send the photo through emails

2.3 Comparison between the similar photo applications and our target application, SNAP!

Features / Applications	Puri!	PhotoFunia	Polarize	SNAP!
Adding Frame	✔	✔		✔
Adding Stamps	✔			✔
Adding Text	✔			✔
Face Detection (on server)		✔		
Face Detection (on iPhone)				✔
Real Time Face Tracking Camera				✔
Pixel-by-pixel Filtering			✔	✔
In-app Photo Taking		✔	✔	✔
Save photo to album	✔	✔	✔	✔
Export to Facebook	✔			✔
Email Photo	✔			✔

2.4 System Architecture

Photo is passed from the photo library or the camera as the image source, the image is then passed to the program. First, the image will undergo the face detection process to locate the face on the photo. After that, the image will be passed to the image processor to edit the image effect on the photo. The processed image will then be passed to the screen for display. Furthermore, it will also be passed to the photo library to save the new image. The processed image can also be exported to Facebook and email.



2.5 Modules design

2.5.1 Face Detection Module

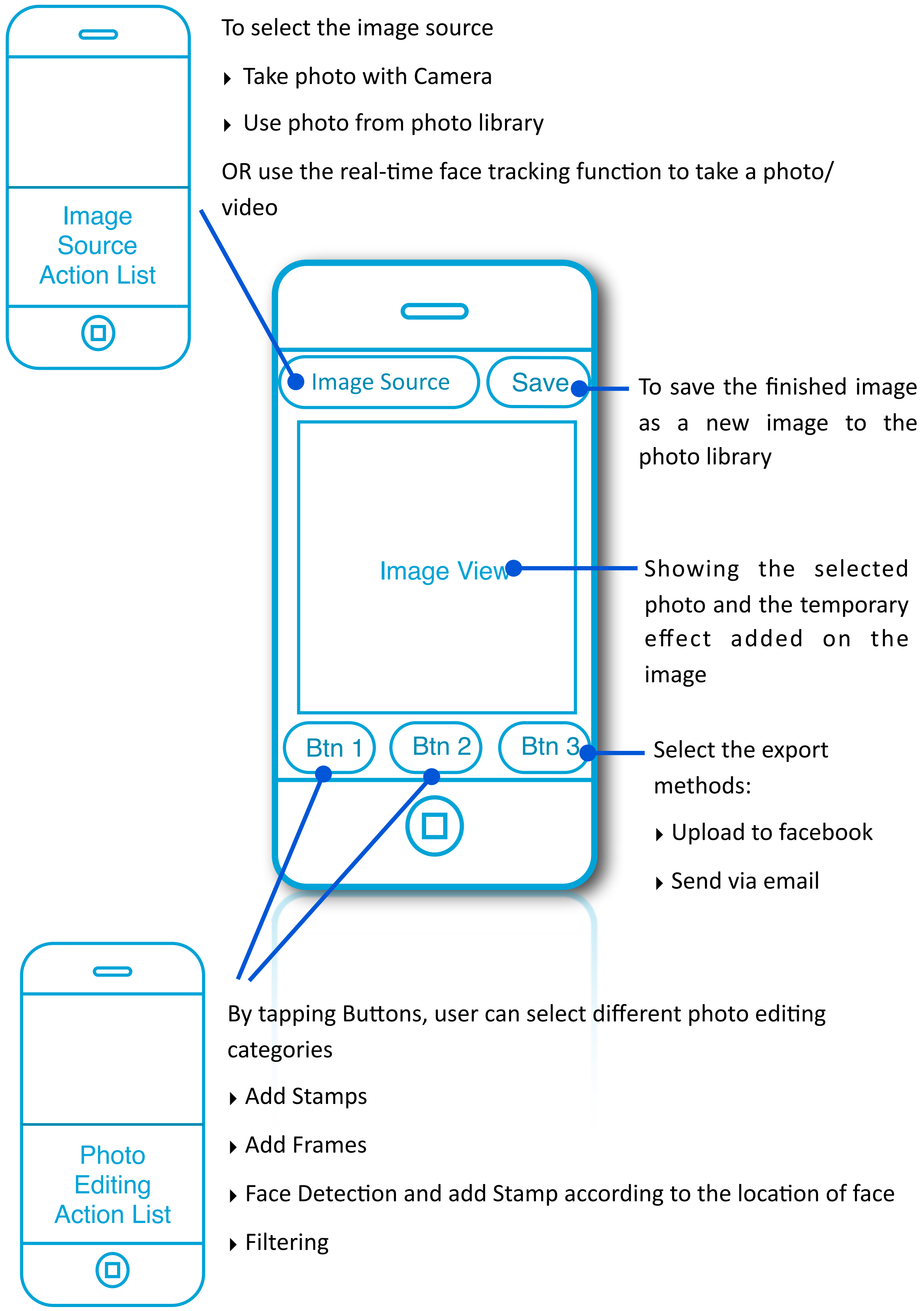
In this module, the objective for this module is to detect face in the photo. It will first do some preprocessing jobs like resizing the photo and turning the image into grey-scale. Then it will start the face detection process, which is locating the face or faces within an image. It will pass the image and the location of the faces to the image processing module for further process.

2.5.2 Image Processing Module

This module handles all the image processing work, including image layering and filtering. Image layering can allow the user to add the stamps, frames on the photo by overlay the stamps image and the photo. It is also responsible for the image filtering feature. It includes colour correction such as brightness and contrast adjustments or colour translation to a different colour space.

After editing the photo, it's responsible for passing the processed image to the screen for display, photo library for saving the image, Facebook for publishing it and e-mail system to send it via e-mail.

2.6 User-Interface Design



To select the image source

- Take photo with Camera
- Use photo from photo library

OR use the real-time face tracking function to take a photo/ video

To save the finished image as a new image to the photo library

Showing the selected photo and the temporary effect added on the image

Select the export methods:

- Upload to facebook
- Send via email

By tapping Buttons, user can select different photo editing categories

- Add Stamps
- Add Frames
- Face Detection and add Stamp according to the location of face
- Filtering

2.7 Project Schedule

Month	Week	
September	Week 1	- Review on the top iPhone applications available on iTunes App Store - Marketing research on the iPhone applications
	Week 2	- Ideas Brainstorming
	Week 3	- Draft of the application Design
	Week 4	- Deliver the detailed specification of the design - Implement the user-interface
October	Week 5	- Implement simple image editing feature - Get familiar with the theory and algorithm of face detection
	Week 6	- Study OpenCV on PC - Research about OpenCV on Mac - Implement face detection feature on static image
	Week 7	- Implement face detection feature on static image
	Week 8	- Implement face detection feature on static image
November	Week 9	- Testing on iPhone simulator
	Week 10	- Import the program to iPhone - Testing on iPhone (both iPhone 3GS and 3G) - Implement image layering feature
	Week 11	- Documentation - Refinement and Amendment of program
	Week 12	- Documentation - Icon design
December	Week 13	- Review on the mid term progress of our project - Schedule the further tasks in the next semester

		September				October				November				December			
Design Phase	Review and Market Research	🟦															
	Application Design		🟦	🟦	🟦												
	User-Interface Design			🟦													
	Icon Design												🟦				
Implementation Phase	User-Interface Implementation				🟪												
	Simple photo editing feature implementation					🟪											
	Face detection feature implementation					🟪	🟪	🟪	🟪	🟪							
	Image Layering feature implementation										🟪						
	Editing icon												🟪				
Testing Phase	Testing on iPhone simulator									🟥	🟥	🟥	🟥				
	Testing on iPhone 3GS and 3G										🟥	🟥	🟥				
	Refinement and Adjustment											🟥	🟥				
Report	Documentation											🟨	🟨				
	Review on progress													🟨			

3. Implementation Phase

To start the implementation phase, we have listed all the key elements and technologies that required in our application and separate them into 3 parts. Including user-interface creation, face detection and image layering.

As the program is developed on iPhone, the programming language will be Objective C. And the developing platform will be Xcode on the Mac OS.

For the first part of the implementation phase, the program started with implementing the user-interface of the framework mainly for the ease of testing the program.

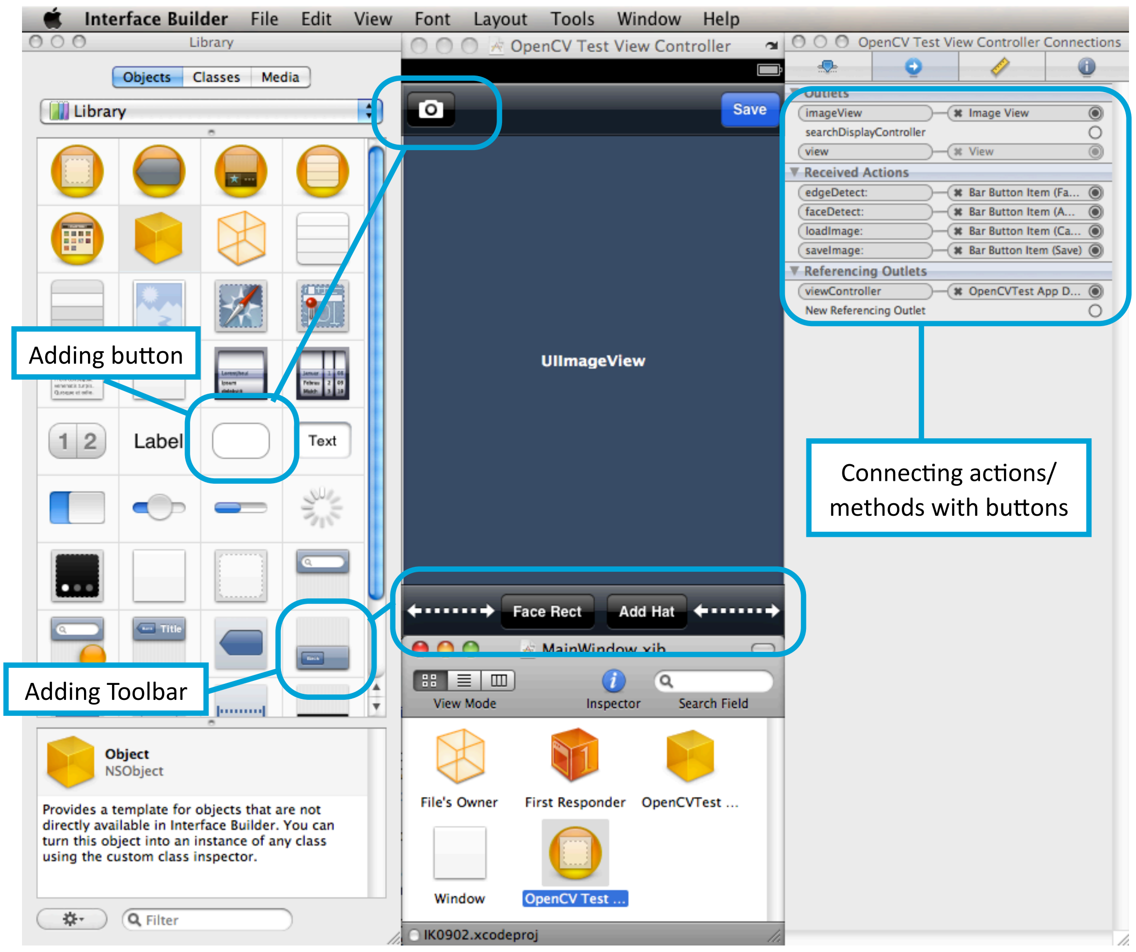
Afterwards, the core part of the application, face detection, is the next step in the implementation stage. In this part, we tried to implement the face detection function into the program with the help of OpenCV library's face detection technology. Thus, we plugged the OpenCV library into the Xcode project and make use of the library functions for the face detection. With the help of the library, the face detection works smoothly.

Finally, with the detected face(s), the program can edit the image source with specified face location(s), i.e. adding hat automatically, mark out the faces in the image, etc.

3.1 Part 1: User-Interface Creation

There are several methods to create the user-interface, as this is the first time for us to develop an iPhone application, we decided to use the "Interface Builder" and the "UIKit" provided from the development tools and the iPhone SDK.

"Interface Builder" is an application for designing and testing user-interface. We can use Interface Builder to create user-interfaces that follow the Mac OS X human-interface guidelines by dragging user-interface elements from a palette of predefined controls and dropping them into the window or view they are configuring.



Object
NSObject

Provides a template for objects that are not directly available in Interface Builder. You can turn this object into an instance of any class using the custom class inspector.

imageView — Image View
searchDisplayController
view — View
edgeDetect: — Bar Button Item (Fa...
faceDetect: — Bar Button Item (A...
loadImage: — Bar Button Item (Ca...
saveImage: — Bar Button Item (Save)
viewController — OpenCVTest App D...
New Referencing Outlet

"UIKit" provides the classes needed to construct and manage an application's user-interface for iPhone and iPod touch. It provides an application object, event handling, drawing model, windows, views, and controls specifically designed for a touch screen user-interface. Figure 3.1.1 illustrates the classes in this framework.

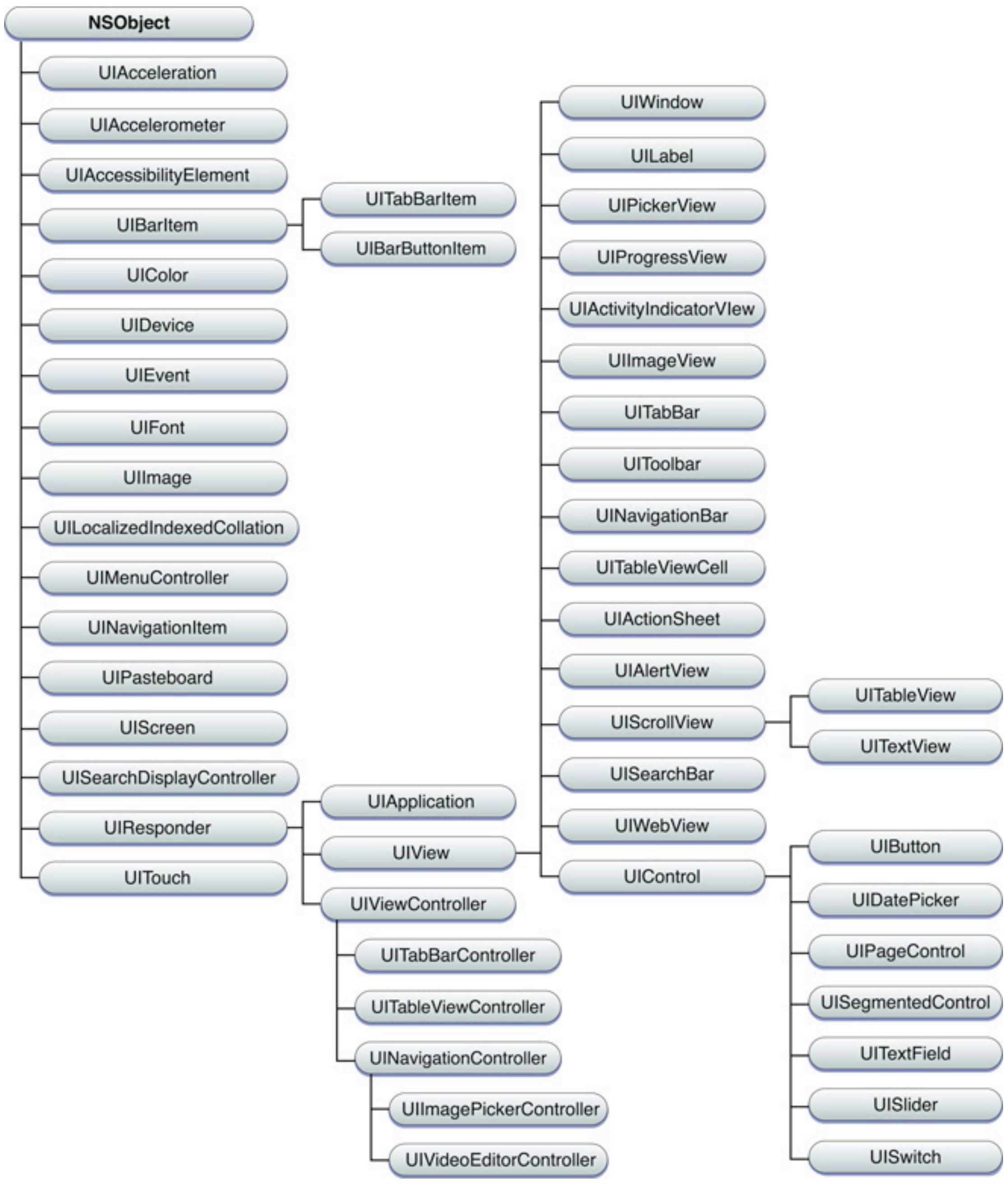


Figure 3.1.1 UIKit class hierarchy

With the help of "Interface Builder" and "UIKit", we built the user-interface as the followings:

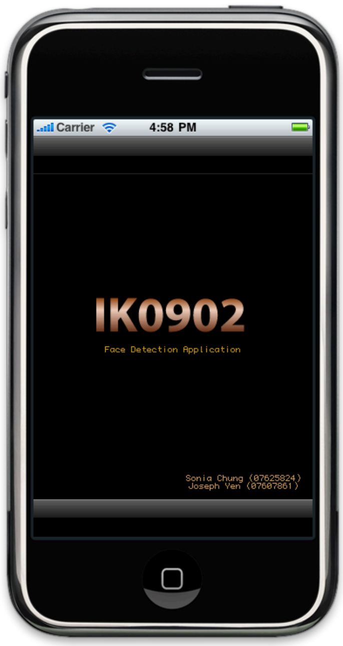


Figure 3.1.2 Loading page

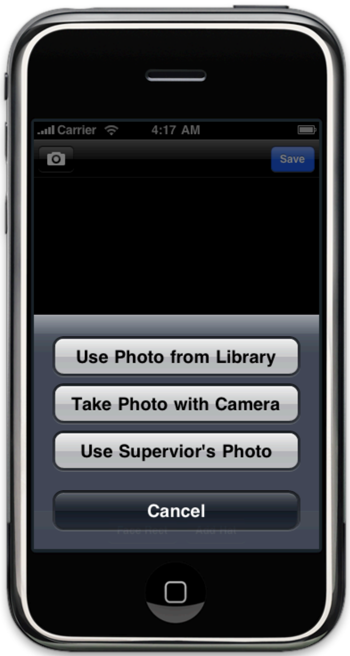


Figure 3.1.3 Source choosing page

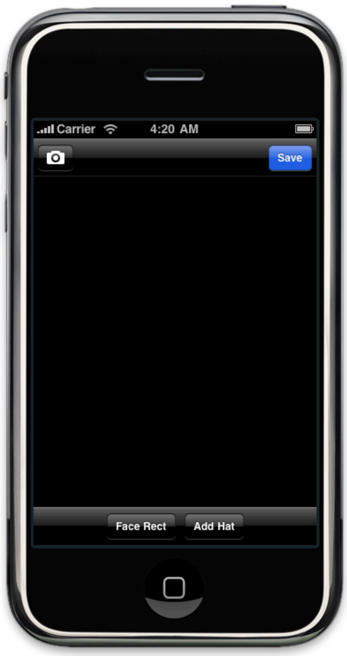


Figure 3.1.4 Image processing page

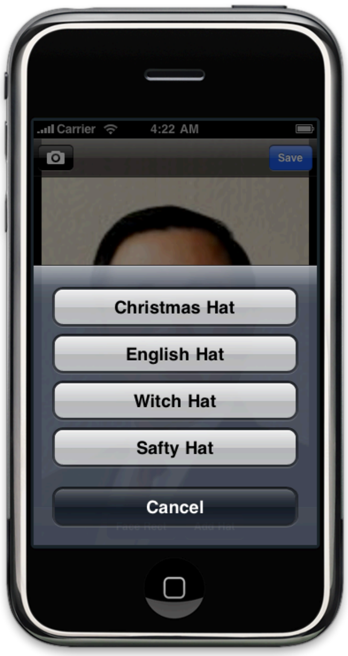


Figure 3.1.5 Action choosing page

3.2 Part 2: Face Detection

In the start of implementing face detection phase, as we can only test the program at the iPhone simulator on the Mac. We did not try any coding for invoking the iPhone camera. Instead of using the iPhone camera, we decided to use the iPhone photo library as the image source for the implementation phase until we can test our program on the iPhone.

We plugged OpenCV library's face detection technology for the application. In order to get the basic idea about how the face detection operates, we spent two weeks to study the OpenCV library before implementing the codes for the application. The algorithm is shown in figure 3.2.1.

Also, there are some assumption for the face detection. To identify the face in an image, it is necessary that the face on the photo is front facing, well lit, not blurred and not covered by anything. A passport photo is a good example. Preferably, the photo's background should be neutral.

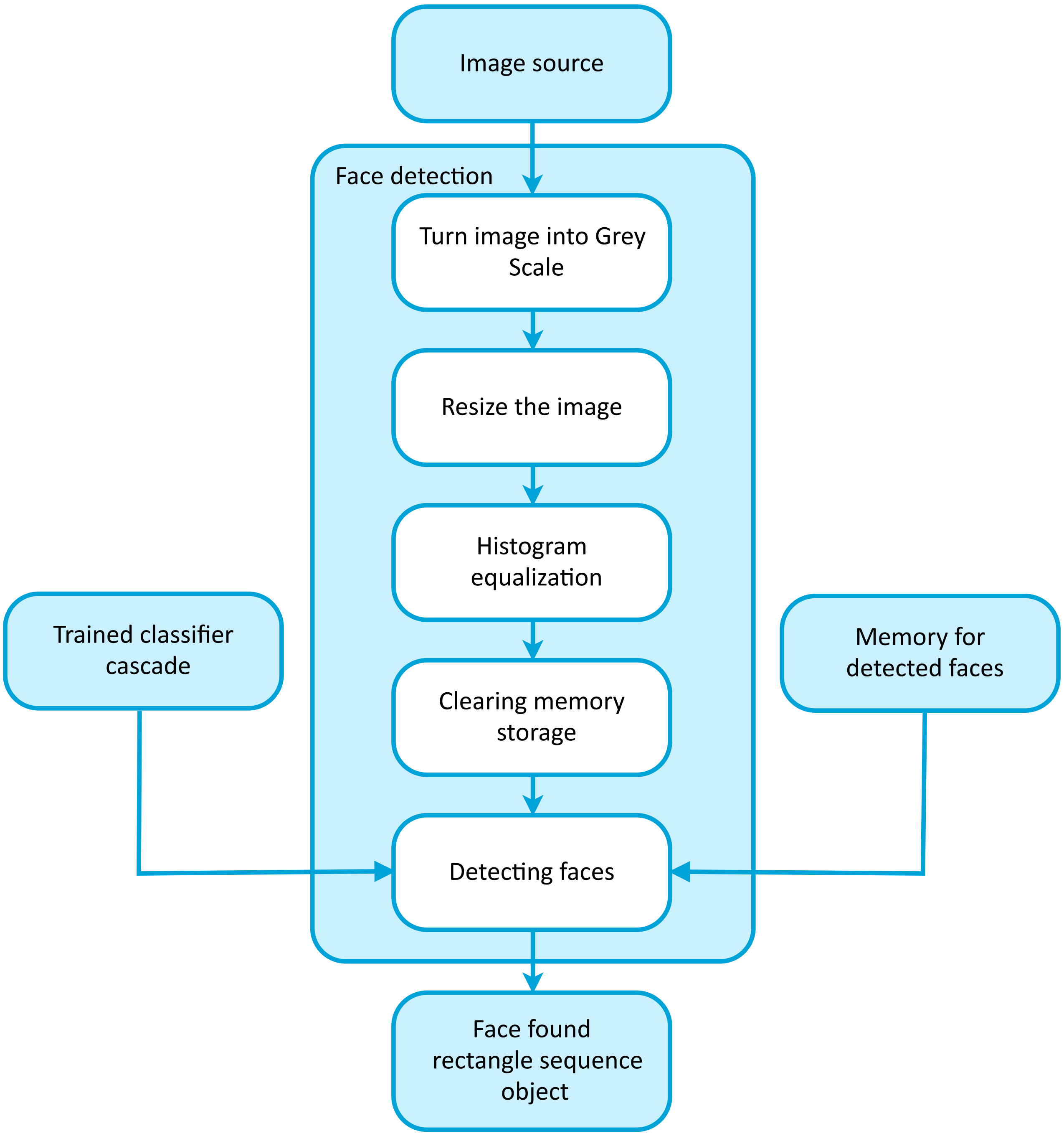


Image source
Face detection
Turn image into Grey Scale
Resize the image
Histogram equalization
Clearing memory storage
Detecting faces
Trained classifier cascade
Memory for detected faces
Face found rectangle sequence object

Figure 3.2.1 The algorithm for face detection.

At the very beginning of face detection phase, we need to load a previously trained classifier cascade for the future use by the face detection part.

The classifier cascade is a Haar classifier, the Viola-Jones detector, which builds a boosted rejection cascade. The classifier cascade is pre-trained with thousands of object examples and tens of thousands of non-object examples as training data set. The classifier uses the threshold of the sums and differences of rectangular regions of data produced by any feature detector, which may include the Haar case of rectangles of grey-scale image values.

With the Viola-Jones rejection cascade, which boosted the classifier, the weak classifiers that it boosts in each node are decision trees that often are only one level deep “decision stumps”. A decision stump allows just one decision of the following form: “Is the value *v* of a particular feature *f* above or below some threshold *t*”; then, a “yes” indicates face and a “no” indicates no face:

$$f_i = \begin{cases} +1 & v_i \geq t_i \\ -1 & v_i < t_i \end{cases}$$

The Viola-Jones classifier employs AdaBoost at each node in the cascade to learn a high detection rate at the cost of low rejection rate multi-stump classifier at each node of the cascade. This algorithm incorporates several features.

- It uses Haar-like input features: a threshold applied to sums and differences of rectangular image regions.
- Its integral image technique enables rapid computation of the value of rectangular regions or such regions rotated 45 degrees. This data structure is used to accelerate computation of the Haar-like input feature.
- It uses statistical boosting to create binary (face - not face) classification nodes characterized by high detection and weak rejection.
- It organizes the weak classifier nodes of a rejection cascade.

The Haar-like features used by the classifier are shown in Figure 3.2.2. At all scales, these features form the "raw material" that will be used by the boosted classifiers. They are rapidly computed from the integral image representing the original greyscale image.

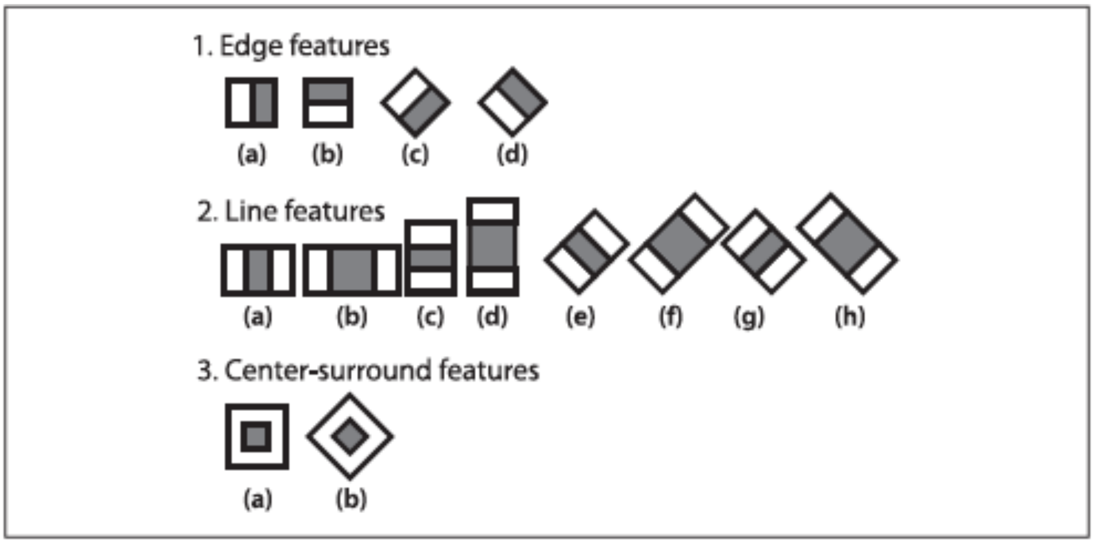


Figure 3.2.2 Haar-like features from the OpenCV source distribution (the rectangular and rotated regions are easily calculated from the integral image): the light region is interpreted as "add that area" and the dark region as "subtract that area"

Also, each classifier group is boosted into nodes of a rejection cascade as shown in Figure 3.2.3. In the figure each of the nodes F_i holds the boosted cascade of groups of decision stumps which trained on the Haar-like features from faces and non-faces examples. Moreover, the nodes' order is in a ascending order in the level of complexity. Therefore, the computations are minimized with rejecting easy regions of the image. Furthermore, each node is tuned to have a very high detection rate. For example, when training on faces, almost 99.9% of the faces are found with about 50% of the non-faces are incorrectly "found" at each node. However, this is acceptable because using for example 20 nodes will provide a face detection rate of 0.999^{20} is approximately 98% and the false positive rate of 0.5^{20} is approximately 0.0001% only.

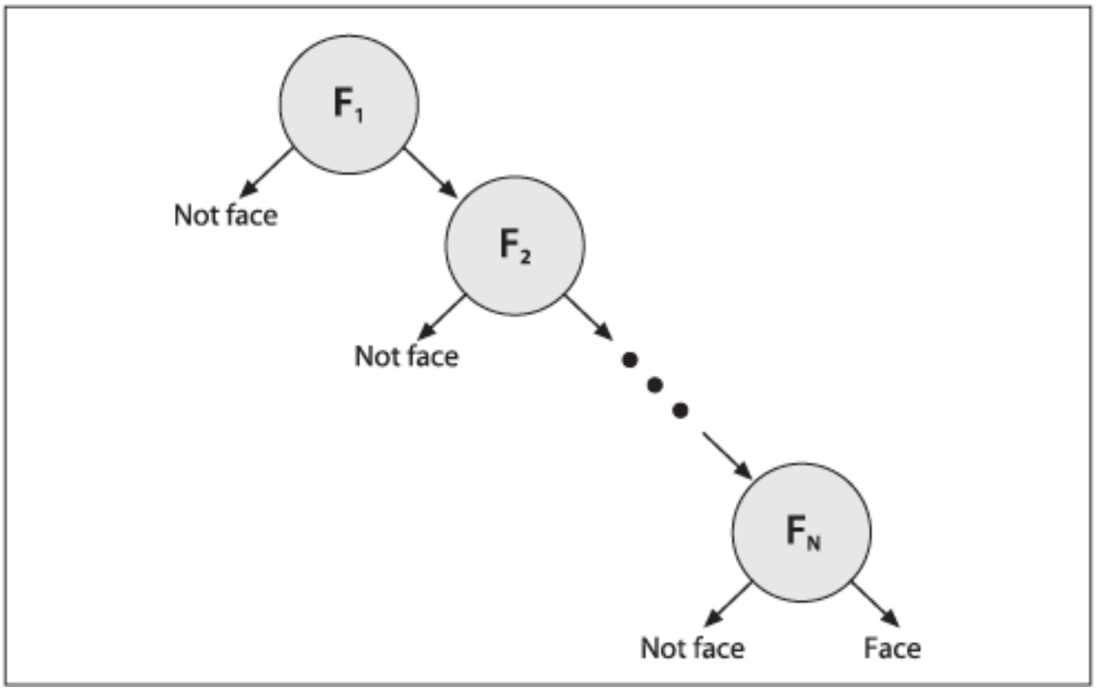


Figure 3.2.3 The rejection cascade used in the Viola-Jones classifier.

Afterwards, with the pre-trained classifier cascade, the face detection part start.

1) Converting the image source into grey-scale image.
    - As the face detection part need to calculate with the integral image which using light region and dark region for calculation. To simplify the calculation, it is better to use a grey-scale image instead of colour image.
2) Resizing the grey-scale image.
    - This action can speed up the further face detection steps with a smaller and simpler image.
3) Equalizing histogram.
    - As the integral image features are based on differences of rectangle regions. There is a need for balancing the brightness values for preventing the differences being skewed by overall lighting or exposure of the images.
4) Clearing memory storage.
    - Due to the classifier returns found object rectangles as a sequence object CVSeq, the global storage which use for returning need to be cleared.

5) Detecting faces.
    - The actual face detection process will be taken place.
    - First, the resized grey-scale image, the pre-trained classifier cascade and the memory storage for detected faces will be inputed for the face detection part.
    - The scale_factor parameter determines the size of a jump between each scale; the higher the value, the faster the computation time but at the cost of possible missed detection.
    - The min_neighbors parameter controls the prevention of false detection. This is due to the actual face location in an image will normally get multiple detection within the same area because the scales and surrounding pixels often indicate a face; the higher the value, the more detections are needed for the final decision of a face in a single location.
    - The flags parameter has four settings:
        - CV_HAAR_DO_CANNY_PRUNING

          The classifier will skip the flat regions (i.e. the regions with no lines).
        - CV_HAAR_SCALE_IMAGE

          The algorithm will scale the image rather than the detector.
        - CV_HAAR_FIND_BIGGEST_OBJECT

          This will only return the largest face found in the image.
        - CV_HAAR_DO_ROUGH_SEARCH

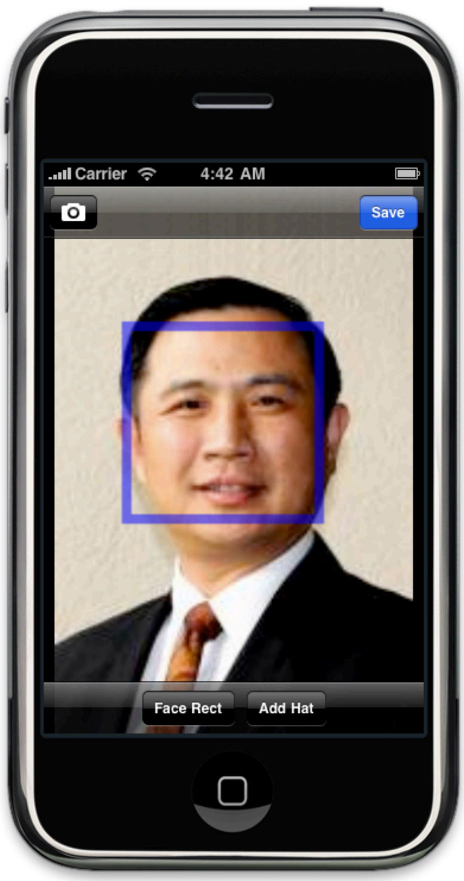
          This will return the first face found in the image immediately.
    - The min_size parameter will determine the smallest region which to be search for a face. The larger the value, the faster the computation yet, it also leads to the cost of missing small faces.
6) After the detecting faces, a sequence of detected faces will be obtained from the face detector. With the sequence of detected faces, the image can be used for further image process.

3.3 Part 3: Image Layering

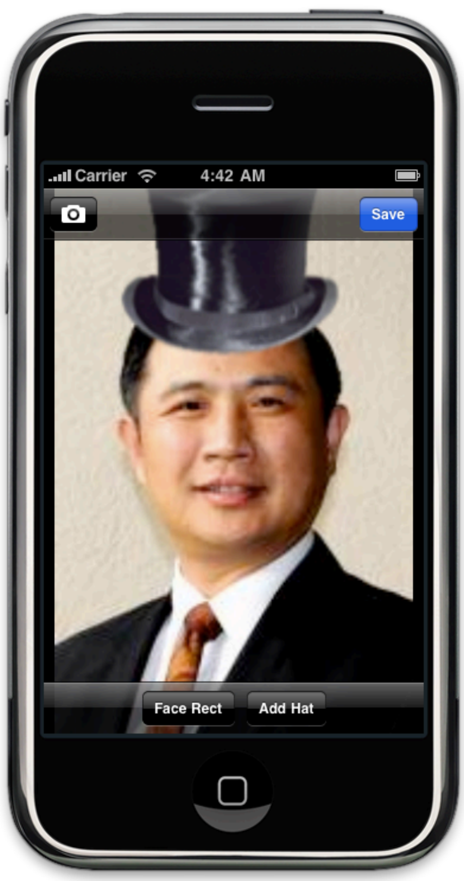
After the program found all the faces out, the face sequence can be used for further image processing. Like, adding image on the present image, cropping out the face to place on a different image.

At the present stage, the image after detecting faces can add different hat on the detected faces, or adding a rectangle to point out the location of the faces.

As there will be a sequence of faces can be obtained from the face detection part, the image processing will use the sequence's positions and sizes of the faces for adding rectangle on the face, or adding hat on the top of the face by drawing extra image on the image source.



or



3.4 Problem Encountered

During the implementation phase, there were several problem encountered.

1) The iPhone Processing Power

The processing power is a critical problem encountered during the implementation. The processing power of the simulator and the iPhone 3G S and the iPhone 3G have a significant difference.

By using different devices, the speed of the face detection process varies. It is hard to determine the threshold which is suitable for both iPhone 3G and 3G S.

2) The licensing problem

Apple's developer license is needed for importing our program to the real iPhone devices for testing. In the beginning of the implementation phase, we did not have the developer license. Therefore, we can only test the program on the iPhone simulator of the Mac OS, which is a simulated device without the camera function. This affects a little bit at the beginning phase, but we managed to adjust our schedule and worked on other features instead of the camera function. Thus, our program was decided to detect faces on static images in the first semester. Thanks to the generous support from our department, the license problem was solved in mid November. We can test the program on the real device to get a better understanding about the actual performance.

4 Testing and Evaluation

The objective of the testing phase is to test the limitation of the face detection feature and the performance on different devices. Through out the testing, we can have better understanding about the program and how well it performs, thus detect software failures so that defects may be uncovered and corrected.

Phase 1 testing is mainly focusing on the limitation of the face detection feature. We have tested the maximum rotation angle of the faces, both horizontally and vertically. The minimum size of face and the photo it can detect.

Phrase 2 testing is used to test the performance, accuracy and consistency of our project. We can do refinement based on how well it performs.

4.1 Data Description

To have a better control of consistency, we decided to use a single face photo that the face detection test succeed, then duplicate it to create multiple faces in one image. We can therefore control the number of faces in the testing image easily and ensure that the face should be detected at the same time.

4.2 Methodology

Phase 1

To test the maximum horizontal rotation angle of the face the program can detect, we have use a front facing photo then rotate the photo horizontally by 5 degrees per time to produce the testing photos required. After that, the testing photos are imported to the simulator and iPhone to test the maximum rotation angle.

To test the maximum vertical rotation angle, the testing photos are captured from a video. Since we found difficulty to perform the vertical rotation of the photo by using photo editing tool and the effect is different between rotating the photo and rotating the face, we decided to take a video of a person rotating his head and then capture the frames to be the testing photos. The testing photos are imported to the simulator and device to test the maximum rotation angle. Similarly, to test the smallest size of faces can be used; the photo is adjusted into different sizes for testing.

Phase 2

- Performance

  Program has been imported to the iPhone simulator, iPhone 3G S and iPhone 3G to test the time required for each machine. The less time it needed, the better performance it has.

- Accuracy

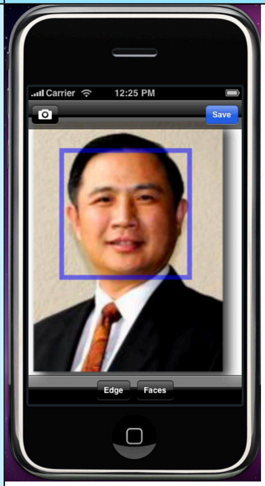
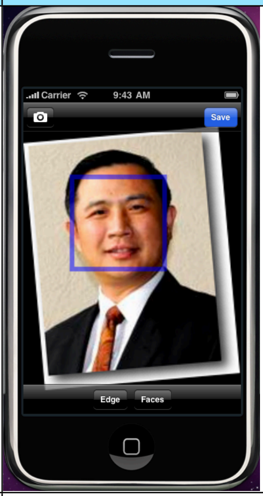
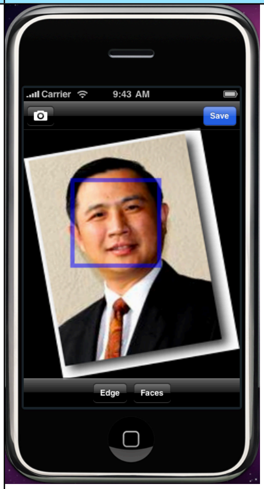
  Under the constraint of the program found in the phase 1 testing, photos of multiple faces are used to test the accuracy of the program to check whether the program can detect all the faces that should be detected.

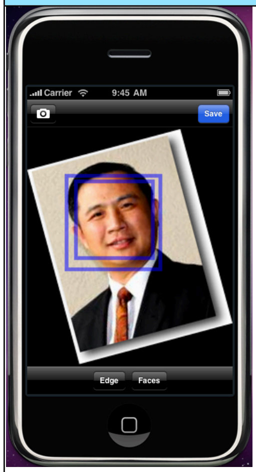
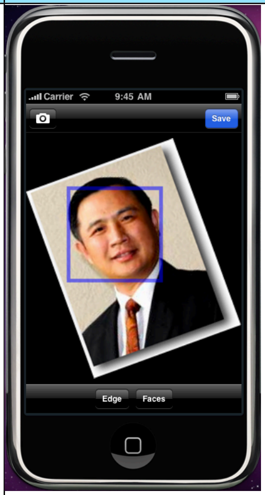
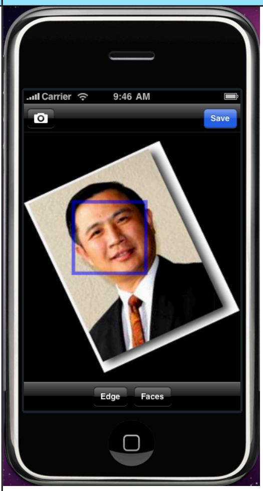
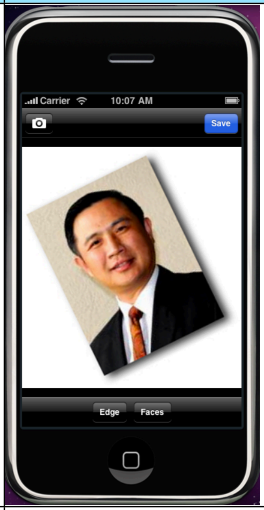
- Consistency

  Each set of testing photos are tested for at least 3 times to see if the results are the same. The runtime for each devices is also tested to see if the processing time differs.

4.3 Evaluation and Results

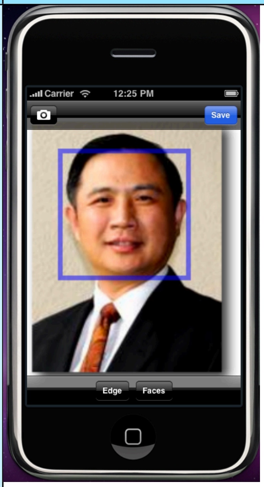
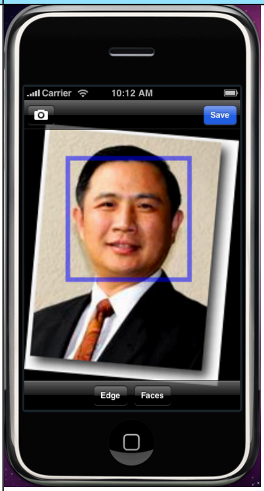
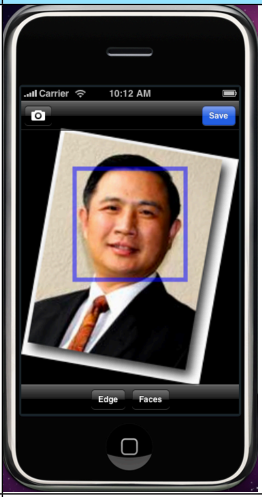
4.3.1 Phase 1 - Horizontal rotation angle of the face

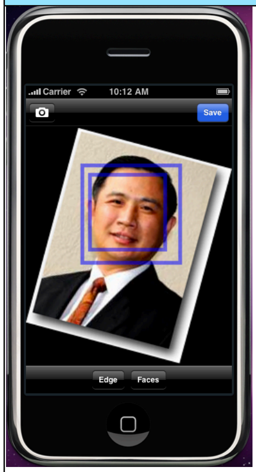
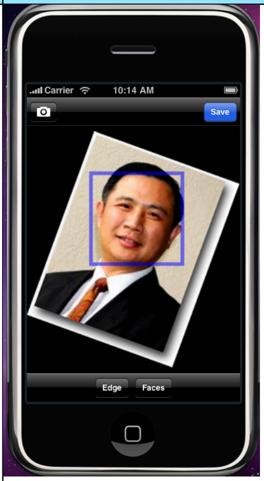
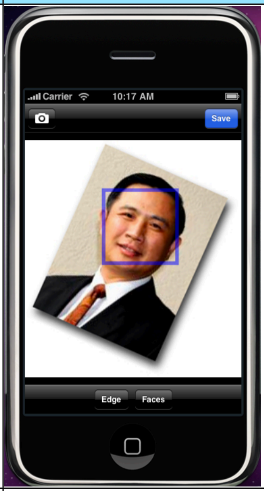
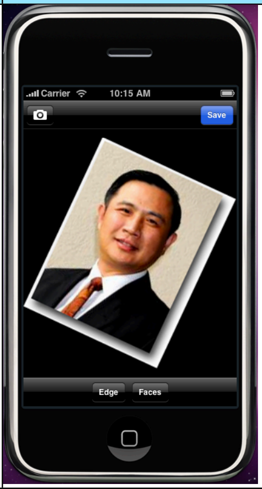
Rotation degree	0^o	Left 5^o	Left 10^o
			
Result	Succeed	Succeed	Succeed

Left 15^o	Left 20^o	Left 25^o	Left 26^o
			
Succeed	Succeed	Succeed	Fail

From the above testing, it shows that the maximum rotation angle to the left side is about 25^o. Rotating 26^o of the face causes failure in the testing.

Similarly, we perform the same testing to the right side.

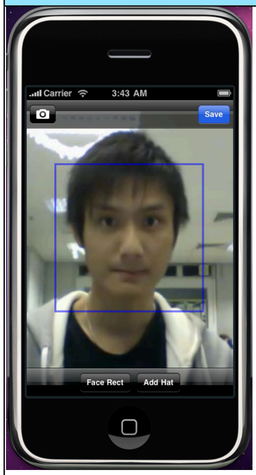
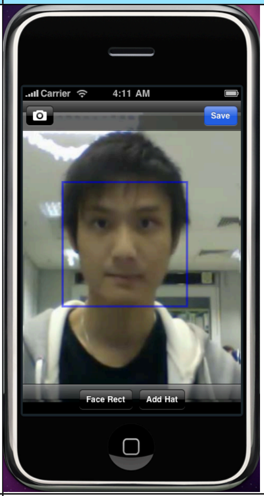
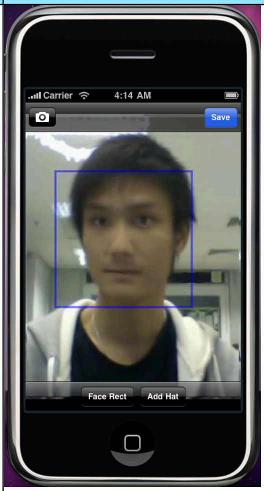
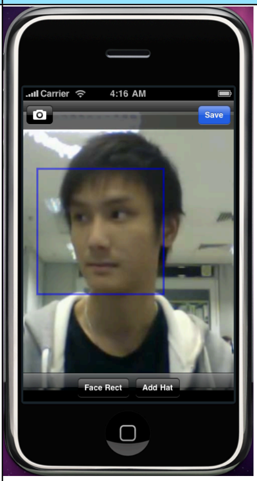
Rotation degree	0°	Right 5°	Right 10°
			
Result	Succeed	Succeed	Succeed

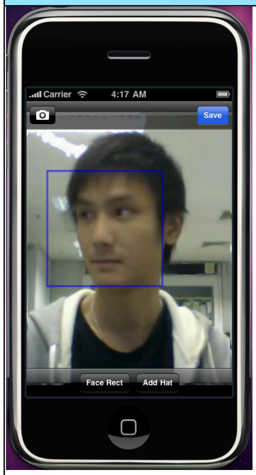
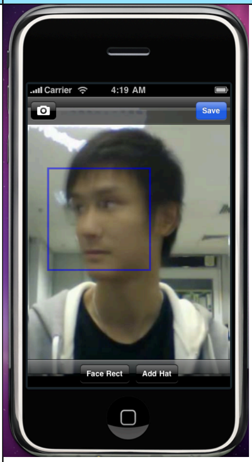
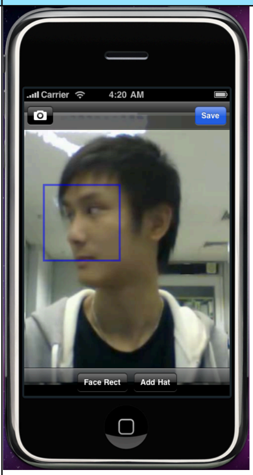
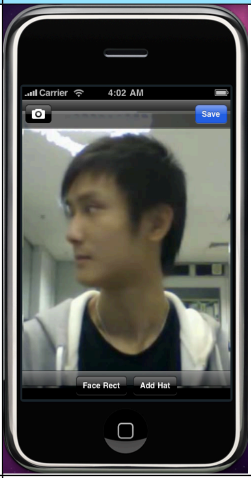
Right 15°	Right 20°	Right 24°	Right 25°
			
Succeed	Succeed	Succeed	Fail

The result for the right side is similar to the one on the left side; the maximum rotation angle to the right is about 24°.

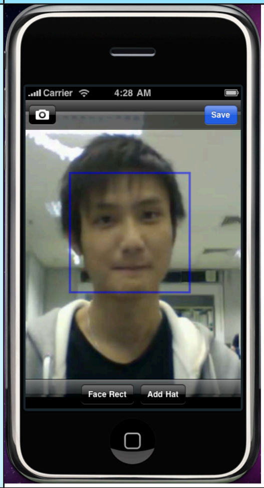
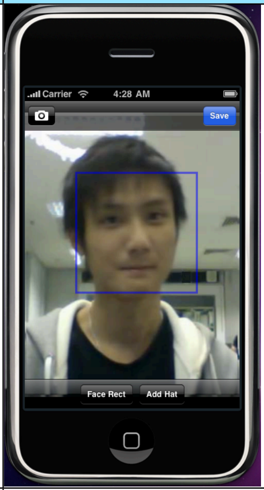
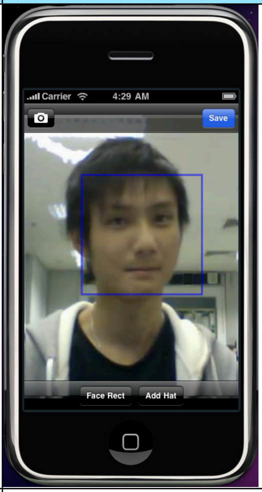
4.3.2 Phase 1 - Vertical rotation angle of the face

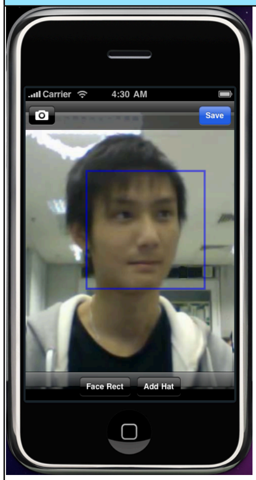
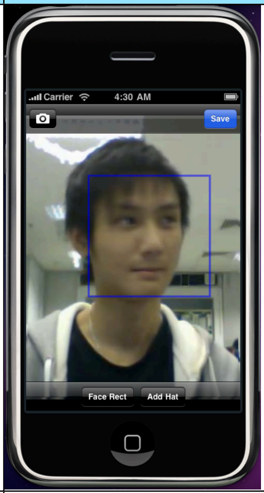
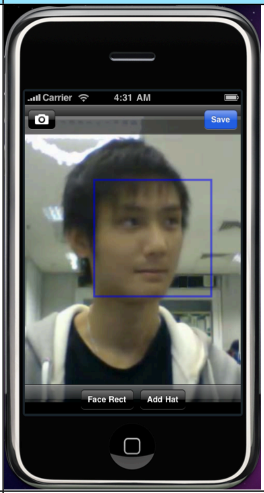
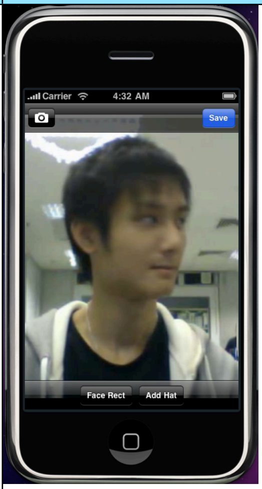
Left Rotation

0º	Left 10º	Left 20º	Left 30º
			
Succeed	Succeed	Succeed	Succeed

Left 40º	Left 50º	Left 60º	Left 70º
			
Succeed	Succeed	Succeed	Fail

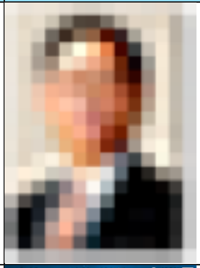
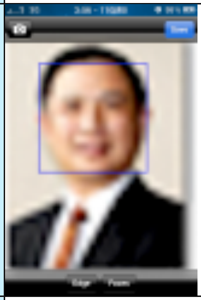
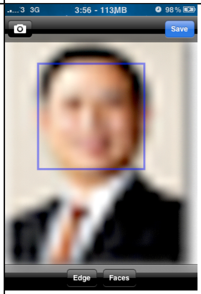
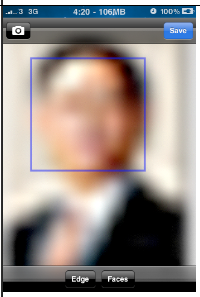
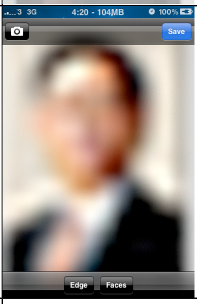
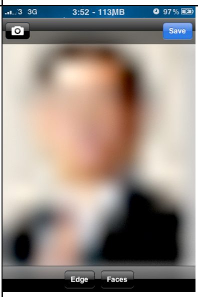
Right Rotation

Rotation degree	0°	Right 10°	Right 20°
			
Result	Succeed	Succeed	Succeed

Right 30°	Right 40°	Right 50°	Right 60°
			
Succeed	Succeed	Succeed	Fail

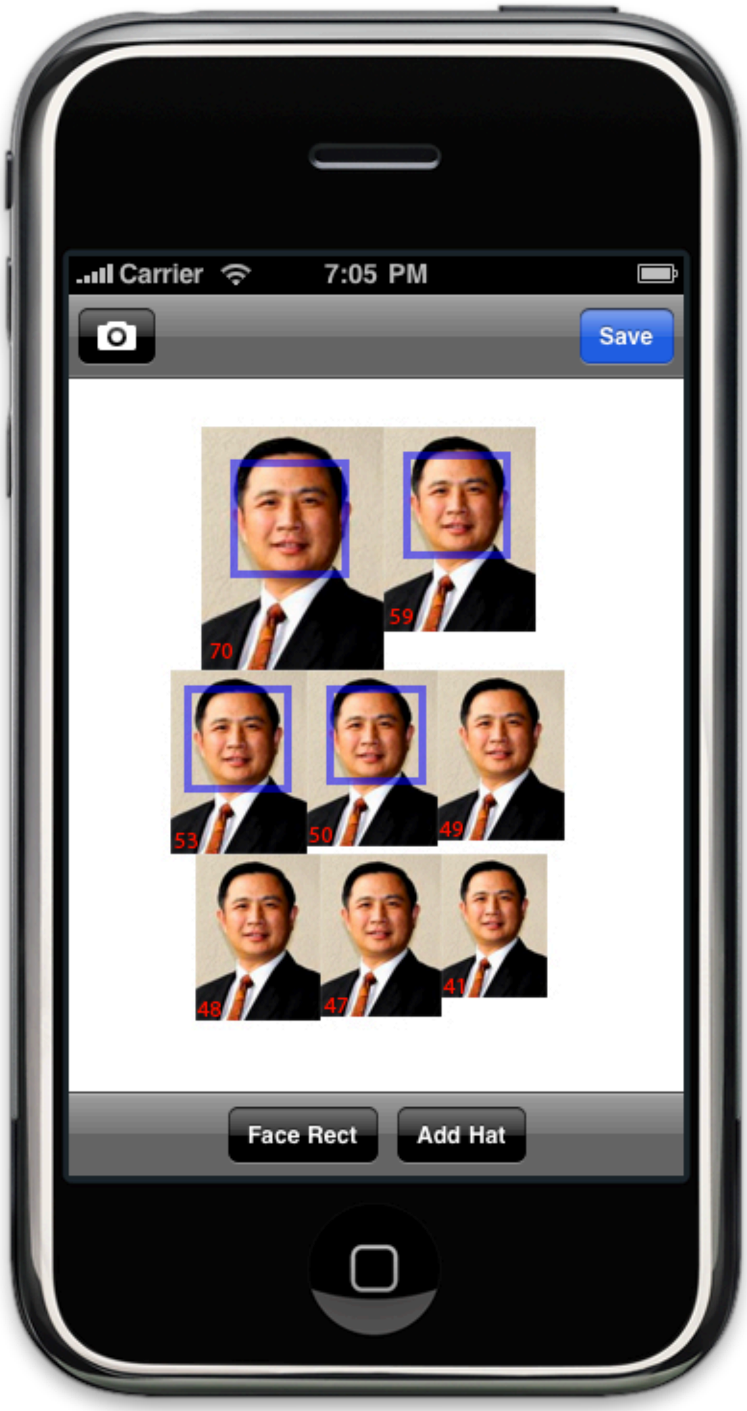
For the left rotation, the maximum rotation angle is about 60° while the one for right rotation is 50°. Since the testing photos are captured from a video, there may be imprecision due to the inconsistent movement of the face in the video.

4.3.3 Phrase 1 – Smallest size of photo

File Size	30 x 38	20 x 25	15 x 20	14 x 19	10 x 13
Photo					
In iPhone					
Result	Succeed	Succeed	Succeed	Fail	Fail

In the program, it will automatically resize the photo. Therefore, if the photo size is really small, like those on above, the application will still resize it to a bigger size before performing the face detection on the photo. Therefore, we found that the photo size affects only when the face in the resized photo is too blurred so that the face can't be identified after resizing the photo by the system. In this testing, the smallest size of photo is about 15x20.

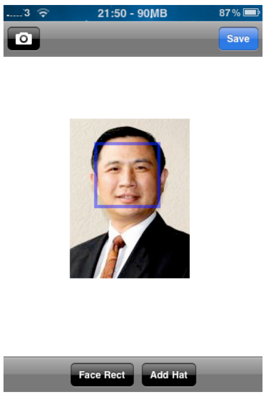
4.3.4 Smallest size of face in photo



The tested result for the smallest size of the face that can be detected is 50x50pixels.

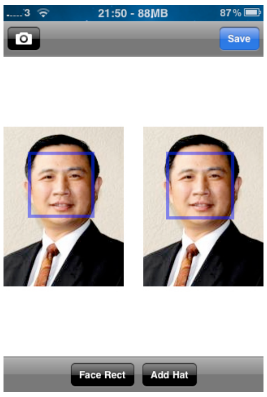
4.3.5 Phase 2 – Performance

Single Face Photo (1 Face)



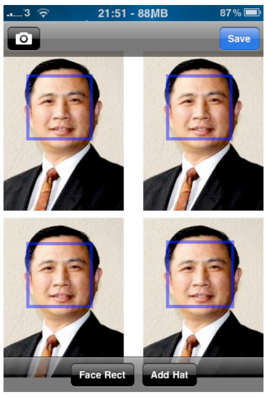
Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	<1 sec	1.5 sec	4 sec

Multiple Faces Photo (2 Faces)

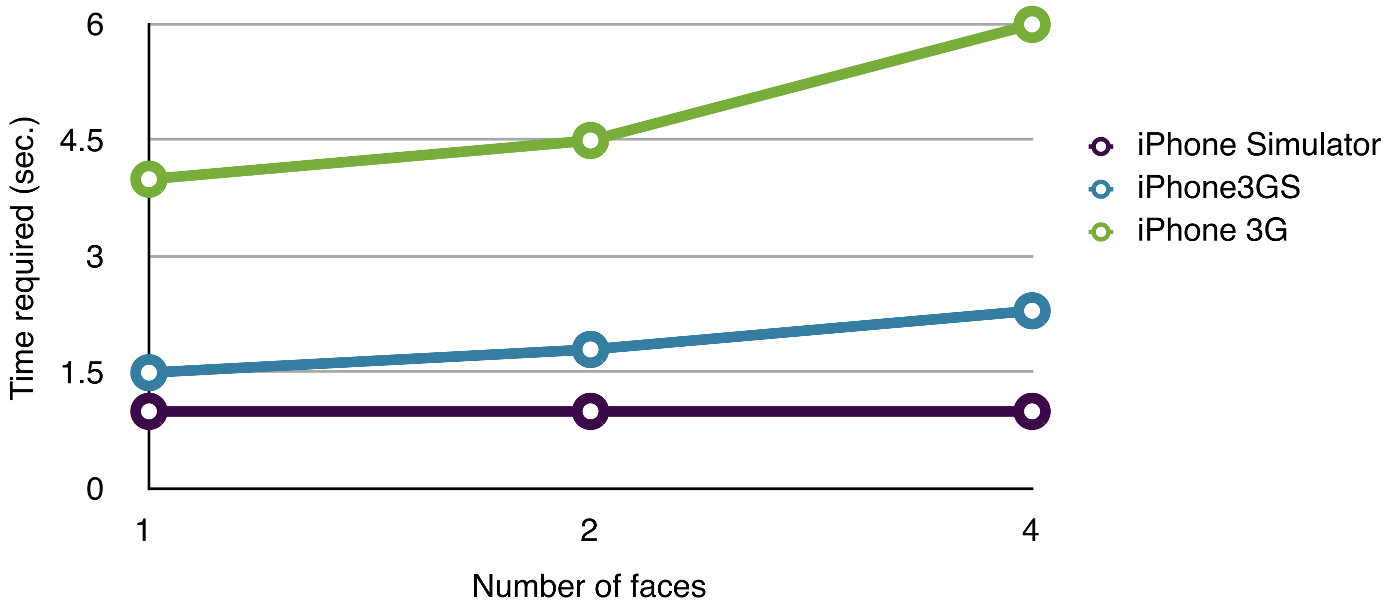


Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	<1 sec	1.8 sec	4.5 sec

Multiple Faces Photo (4 Faces)

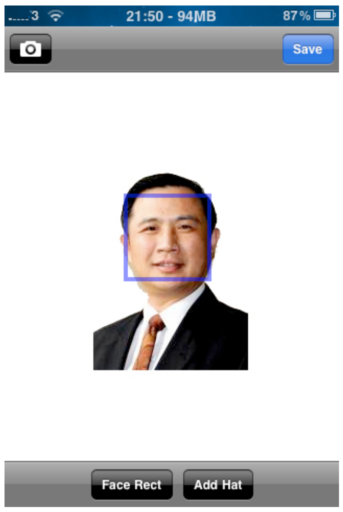


Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	<1 sec	2.3 sec	6 sec



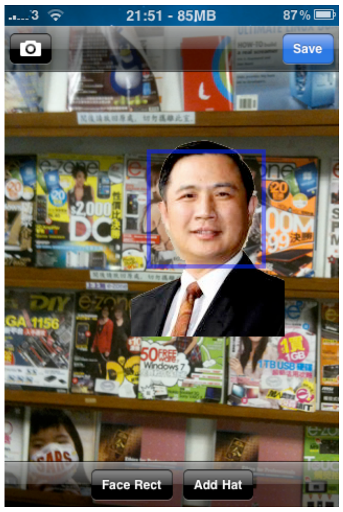
From the above graph, it shows the time required for detecting different number of faces for different devices. Comparing with iPhone 3G S, iPhone 3G needs 2 times the time required for iPhone 3GS to detect faces. The time required increase slightly with the number of faces detected.

Photo with simple background

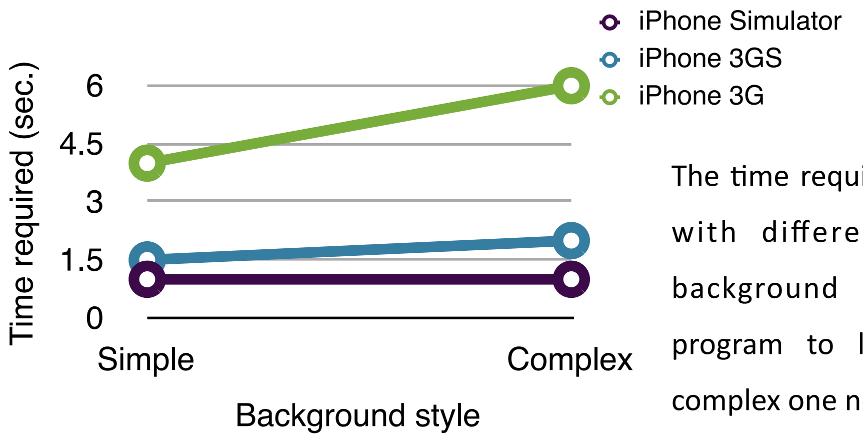


Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	< 1 sec	1.5 sec	4 sec

Photo with complicated background

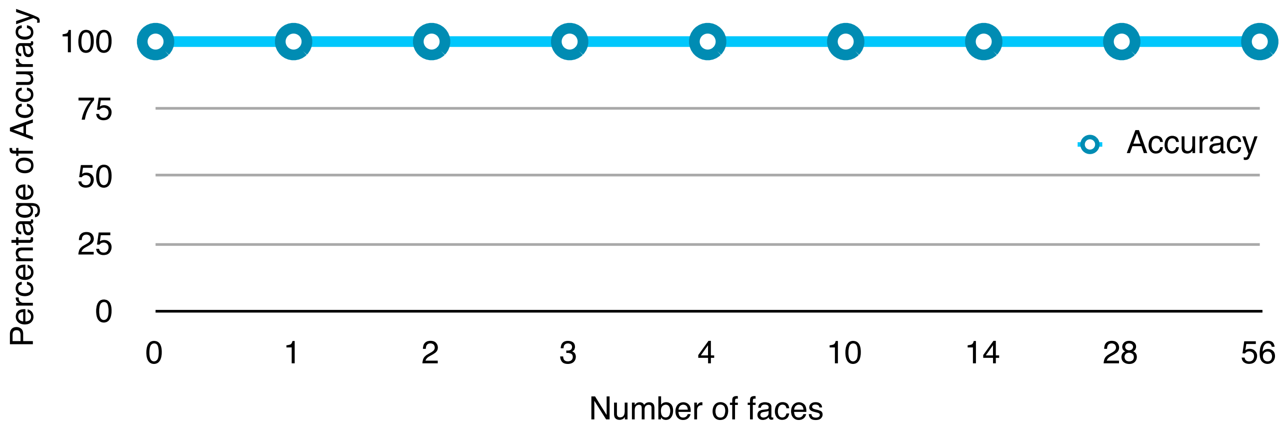


Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	< 1 sec	2 sec	6 sec

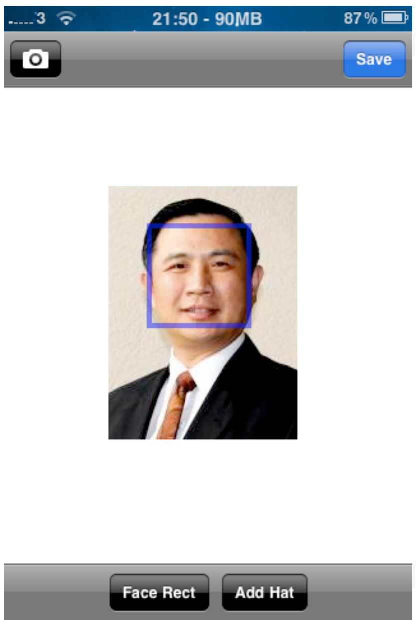


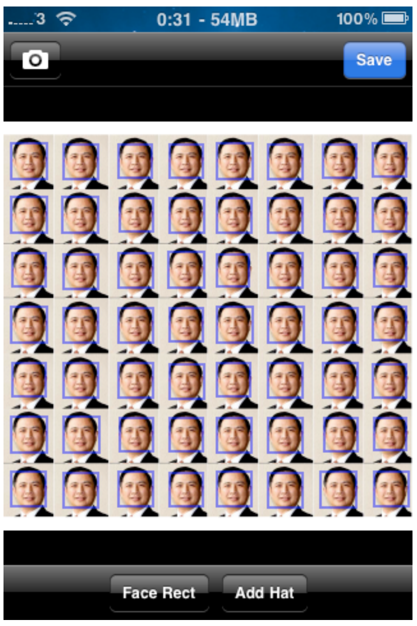
The time required for face detection varies with different backgrounds. Simple background needs less time for the program to locate the face while the complex one needs more time.

4.3.6 Phase 2 – Accuracy



Above graph shows the percentage of accuracy of detecting the according number of faces in a single image. Up to 56 faces were tested and the percentage of accuracy remains 100% .





4.3.7 Phase 2 – Consistency

The program produces the same results for detecting the same set of photos in various time. The processing time for the each device to detect the same set of photos is also approximately the same.

4.4 Analysis and Discussion

In phase 1 testing, we found the maximum rotation angle of the face the program can be detected, both horizontally and vertically. If the tilting angle exceeds the limitation, there is a high probability that the program cannot detect the face. Moreover, we also test the smallest photo size it can be used. Since the program will resize the photo to a optimal size, the image will be distorted if the image size is too small. The program will then identify the location of the face in the photo. If the face is distorted too much, the program cannot detect it. Similarly, in a standard photo with the size of 320x480, we have tested the smallest size of the face that can be detected. All the limitation found in the testing is listed in the table below.

	Constraint
Maximum Horizontal Rotation Angle of Face	Left: 25^o Right: 24^o
Maximum Vertical Rotation Angle of Face	Left: 60^o Right: 50^o
Smallest photo size	15 x 20 pixels
Smallest face size in a standard photo	50 x 50 pixels

Phase 2 testing is more focus on the performance of our program. Since the processing power is different for different devices, it requires longer time for lower processing power machine. iPhone 3GS performs much faster than 3G, yet, it still takes 1.5 second to detect a single face photo. Since 1.5 seconds still means a long processing time if we would like to develop a real time face tracking camera.Therefore, refinement is required for further development.

Besides the difference in processing power of the devices, there are some factors affecting the performance of the face detecting process. They are as follows:

1. The number of faces in the image. It needs longer time for detecting more faces.
2. The background of the image affects the performance. The more complicated the background is, the longer time it needs for detecting the faces.

From the accuracy test, we tested it with an image of 50 faces, it still works well without any failure. In normal situation, there will not be more than 50 faces in a snap shot. Therefore, we can assume the program can function properly for our further development.

5.Conclusion

5.1 Accomplishment

In the first semester, we have successfully implemented the basic photo application that can detect multi faces and allow user to add different hats onto the image. During the development progress, we have learnt a lot about how to use the interface builder, Objective-C programming language and Xcode to implement an iPhone application. Meanwhile, we also managed to understand the algorithm of object detection of OpenCV.

5.2 General Conclusion

Developing an iPhone application as a final year project not only enables us to get to know more about the latest technology in the society, but also nurture us to be equipped with self-learning ability for the ever-changing world.

The iPhone application market is ongoing growing every day, more and more innovative applications are available in the App Store. To be more competitive, it simulates us to endeavor to do the best we can and include more creative features in our project. We understand the importance of self-learning, thus we have taken the initiative to explore more. We have learnt about the Objective-C programming language and face detection algorithm through a lot of readings. Since the technology is continuous changing with each passing day, we have also done a lot of research through the Internet to keep us updated with the current trend.

5.3 Further development

By the end of this semester, we have done a review on our progress and plan for our next stage. With the technique we have learnt in the last semester, we believe that we can develop a photo application that is more user-friendly in the near future. More challenging features will be added to our current application. We will be more focus on developing the following aspects in the coming semester.

- Performance Refinement

  To have a better performance, we have to increase the accuracy, meanwhile, shorten the time for face detection feature.

- Real-time face tracking camera

  This function enables user to take a photo or video with a real time face-tracking feature. This function can be enhanced to build an auto face-focusing camera, but we still need to improve the performance for face detection before implementing this feature.

- Photo Editing Features

  - More stamps and frames will be included
  - Filter the photo with different colours
  - Allow colour translation or correction

- Export to Facebook

  To allow user to share their decorated photo to their friends, user can use this feature to publish the photo on Facebook.

- Sending Photo through Email

  To allow user to share their decorated photo to their friends by sending an e-mail.

6 References

- Dave Mark & Jeff LaMarche (2009). Beginning iPhone 3 Development: Exploring the iPhone SDK. Berkeley, C.A.: Apress.
- Mark Dalrymple & Scott Knaster (2009). Learn Objective-C on the Mac. Berkeley, C.A.: Apress.
- Gary Bradski & Adrain Kaehler (2008). Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, C.A.: O'Reilly Media, Inc.
- Apple Inc. (2009). Interface Builder. Available: http://developer.apple.com/tools/interfacebuilder.html. Last accessed 1 Dec 2009.
- Apple Inc.. (2009). UIKit Framework Reference. Available: http://developer.apple.com/iphone/library/documentation/uikit/reference/uikit_framework/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006955-CH1-SW1. Last accessed 1 Dec 2009.
- Apple Inc.. (2009). Apple Reports Second Quarter Results. Available: http://www.apple.com/pr/library/2009/04/22results.html. Last accessed 1 Dec 2009.
- TXT4EVER. (2009). iphone Sales Figures 2009: and their impact on mobile marketing. Available: http://www.txt4ever.com/news/iphonesales-111109.php. Last accessed 1 Dec 2009.
- PhotoFunia. (2007). PhotoFunia. Available: http://www.photofunia.com/. Last accessed 1 Dec 2009.
- gnozu. (2009). iPhone Apps - Polarize. Available: http://www.apptism.com/apps/polarize. Last accessed 1 Dec 2009.
- Mac Geek. (2004). Objective-C Beginner's Guide. Available: http://www.otierney.net/objective-c.html. Last accessed 1 Dec 2009.

7 Appendix

7.1 Application Description Page

IK0902

SNAP!

Category: Photography
Updated 2 December 2009
Current Version: 1 (iPhone OS 3.1.2 Tested)
Publisher: Department of CSE, CUHK
© Department of CSE, CUHK
3.2MB

APPLICATION DESCRIPTION
Want to take a nice snap shot with your family and friends?
Want to decorate it with fancy hats?
SNAP! is here for you!

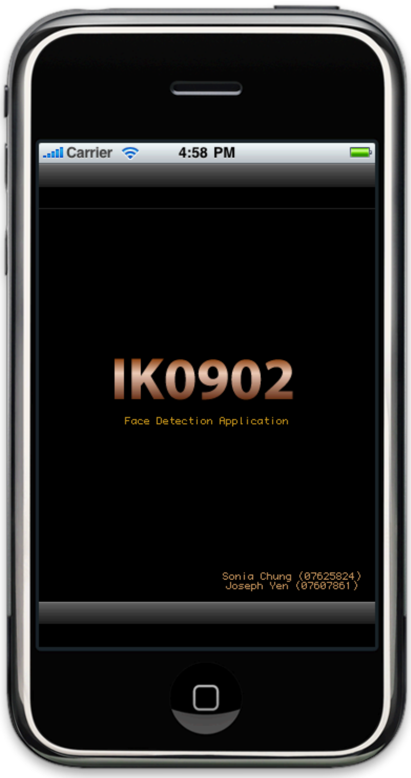
SNAP! allows you to take great picture with your iPhone camera, decorate it automatically with different hats by using our advanced face detection technology. It can detect multi faces in your photo. The interface is simple and easy to use. It is a perfect tool for you to have fun in taking pictures.
So, come and use SNAP! to capture the precious moments with your family and friends!

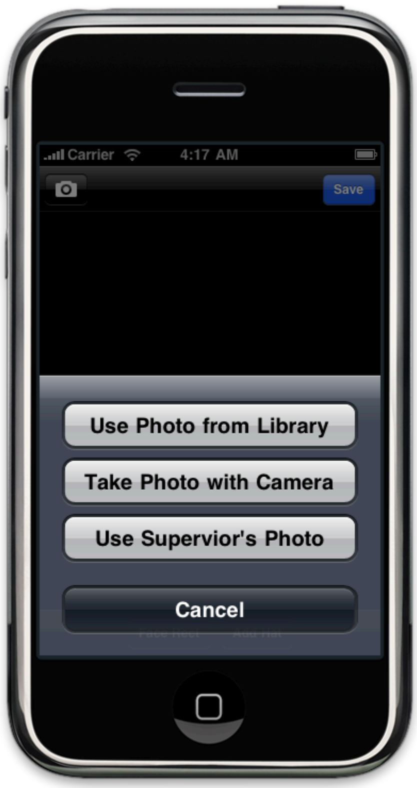
Tags: sticker photos, purikura, face detection,

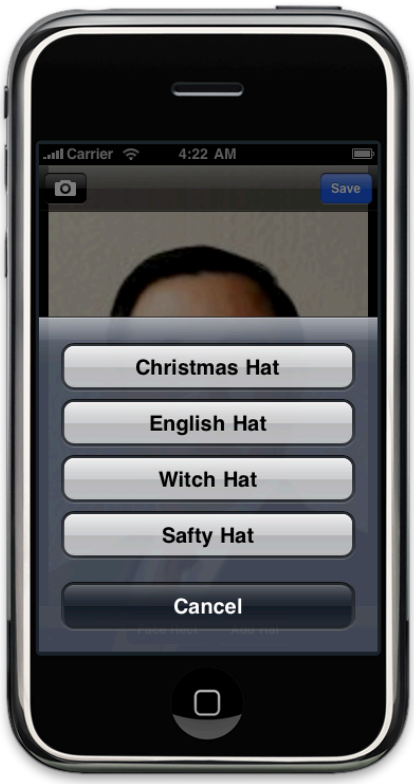
LANGUAGES:
English

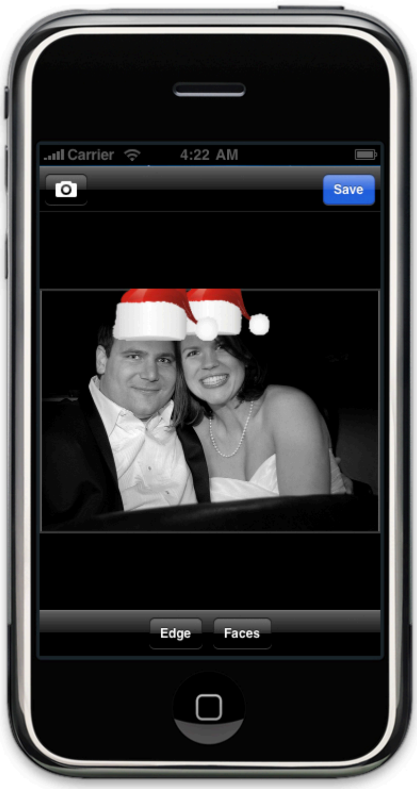
REQUIREMENTS:
Compatible with iPhone and iPod touch
Requires iPhone OS 3.0 or later

IK0902 Website ➔


Carrier
4:58 PM
IK0902
Face Detection Application
Sonia Chung (07625824)
Joseph Yen (07607861)


Carrier
4:17 AM
Save
Use Photo from Library
Take Photo with Camera
Use Supervior's Photo
Cancel


Carrier
4:22 AM
Save
Christmas Hat
English Hat
Witch Hat
Safty Hat
Cancel


Carrier
4:22 AM
Save
Edge
Faces