Final Year Project
2009 - 2010

GAMING ON IPHONE
GROUP: IK0902
SUPERVISOR: PROF. KING KUO CHIN, IRWIN
STUDENT: CHUNG WAN YING, SONIA (07625824)

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

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 provided us with guidance and led us to the correct direction with his invaluable suggestion. The Final Year Project is a fascinating challenge to us, as it required us to learn independently and actively, meanwhile, allowed us to explore to the real iPhone industry that equipped us better for the future.

Table of Content

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, iTunes 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 185,000 applications available in the App Store and up till April, 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 an innovative and user-friendly application to enhance the functionality of iPhone.

Apart from that, the recent introduction of the iPad has brought a new device to all iPhone developers to develop on. iPad is a large, high-resolution LED-backlit IPS display that allows incredibly responsive Multi-Touch and it's thin and light enough to take anywhere. Comparing to iPhone, it allows better touch movement and display and thus greatly increases the range of creativity for designing the applications. Since the SDK of iPad is very similar to that of iPhone, therefore, developers can submit one application design for both iPhone and iPad that have a larger pool of users.

2. Design Phrase

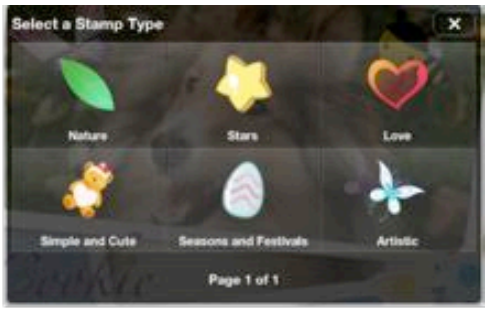
2.1 Semester 1

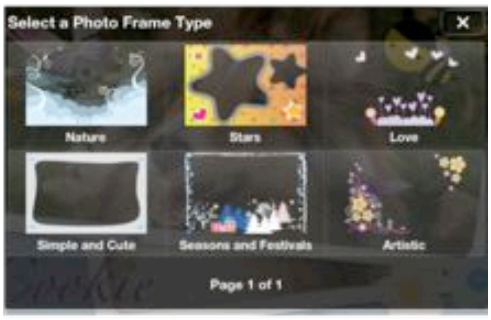
To begin with, we decided to start our project by building a simple photography application. In the early stage of designing the simple iPhone application, we have compared three different popular photo applications released from the iTunes App Store. Via the comparison of the strengths and weaknesses between different applications and the ways they are implemented, we managed to develop our first simple application.

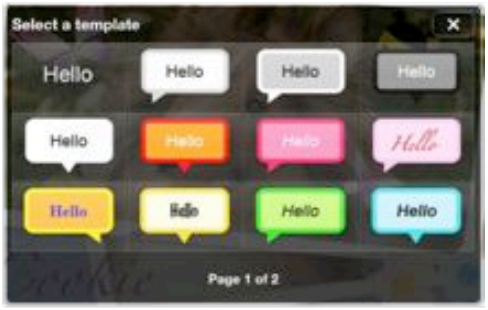
The reviewed photo applications are Puri!, PhotoFunia and Polarize.

2.1.1 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.



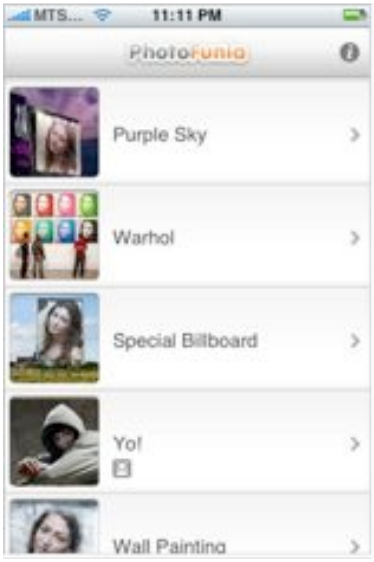


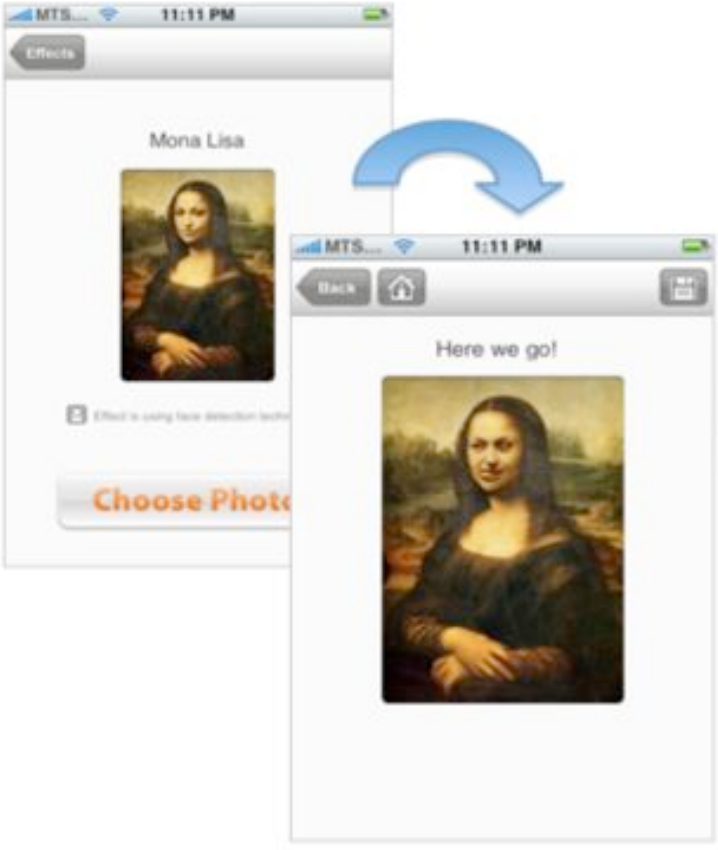


2.1.2 PhotoFunia by Alexey Ivanov

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.





2.1.3 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.

2.2 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.

2.2.1 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 Colored 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.

2.2.2 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

2.2.3 Party Games

- 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.

2.2.4 Ghost Hunter

- 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.

2.3 Final Design of SNAP!

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.

2.3.1 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

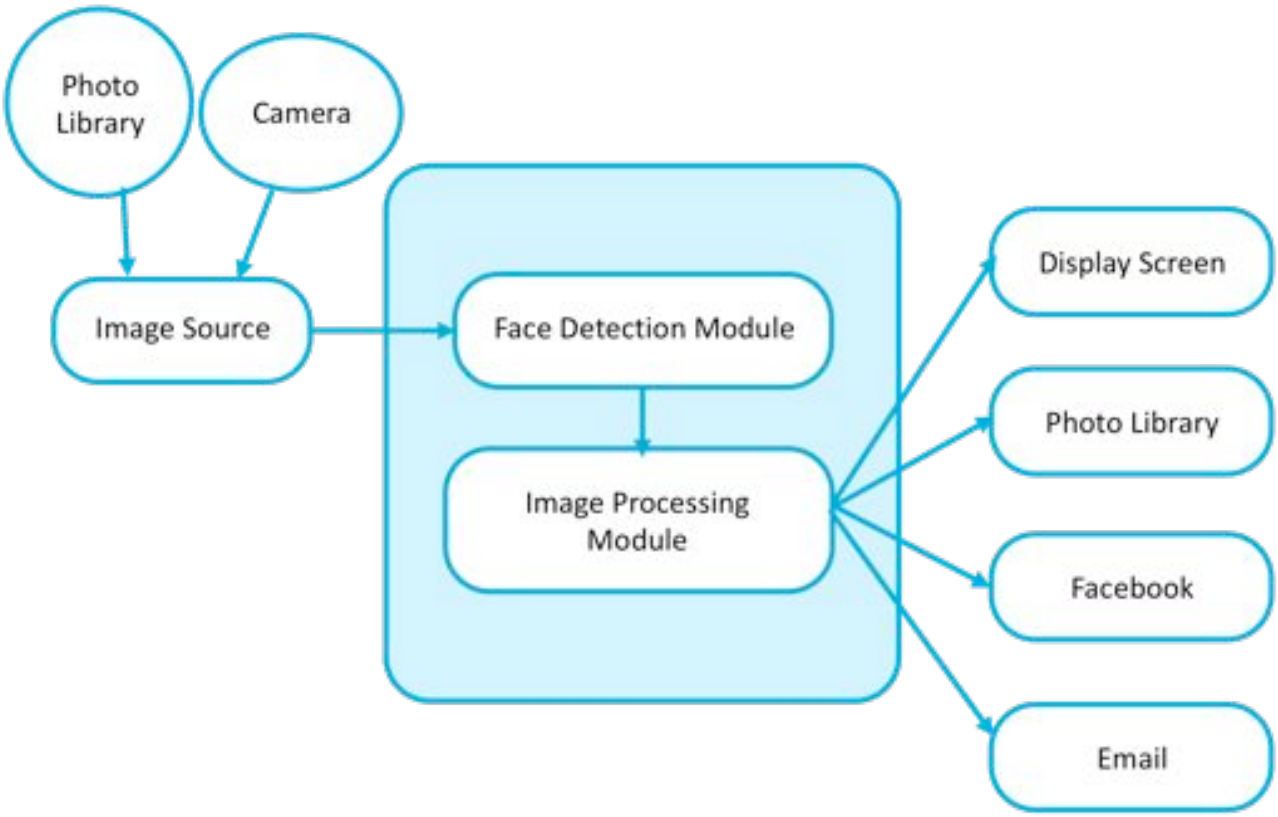
- Face detection on static photo (Single Face/Multi Faces)
- Real-time face tracking camera
- Image Layering for adding stamps, frames and text boxes
- Filtering
- Export to facebook
- Send the photo through emails

2.4 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	✔			✔

2.5 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.



2.6 Modules design

2.6.1 Face Detection Module

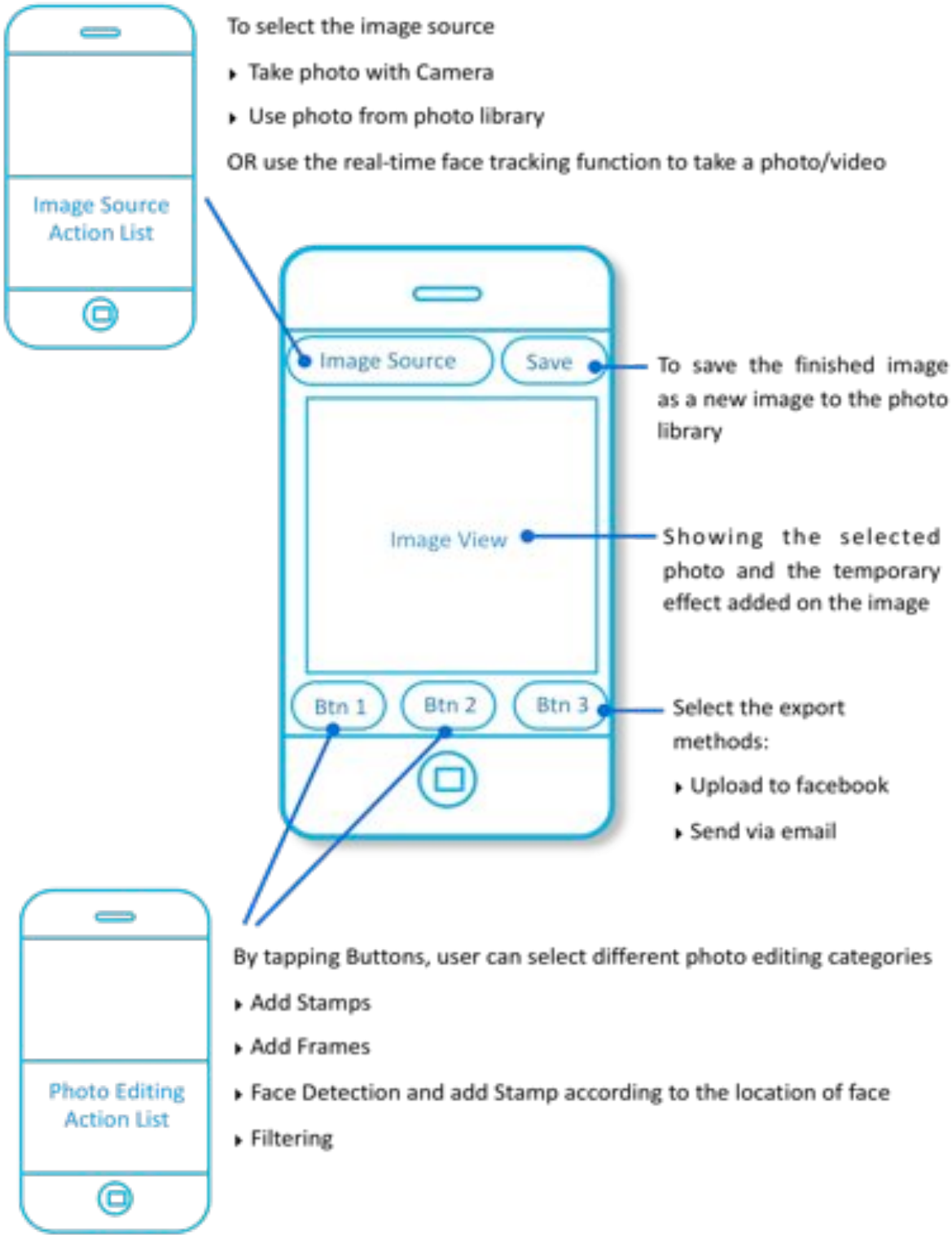
In this module, the objective for this module is to detect face in the photo. It will first do some preprocessing jobs like resizing the photo and turning the image into grey-scale. Then it will start the face detection process, which is locating the face or faces within an image. It will pass the image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

2.6.2 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.

2.7 User Interface Design



2.8 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																

2.9 Summary of First Term

By the end of semester one, we managed to developed SNAP! It is a photo application that allows users to pick photo and add different style of hats onto their photos. User can also saved the edited photo into the library in the end. Although it is not a completed version, it still had some significant feature. We learned a lot from the implementation process and it actually give us a whole picture of the development process from design to implementation, from testing to enhancement and refinement.

Features implemented

- **Load image from the library**
- **In-app photo taking by camera**
- **Face detection (Multi-faces)**
- **Add frame onto all the face location**
- **Add different images (hat) onto the photo**
- **Save image to the photo library**

1. Loading Page

2. The application will pop up an action list that allow user to select the image source, either from photo library, take one by camera or use the sample photo

3. The application will pop us an action list that allows users to select different hats e.g. Christmas hat, English hat, Witch hat, Safety hat

4. The application will detect the faces automatically and add the selected hat onto the photo. The photo can be saved by tapping the "save" button. User can also change the image source by tapping the camera button.

2.10 iPhone Application Overview

As mentioned in the previous pages, we completed a simple application that can detect the face location and add different hats accordingly in the semester one. In this semester, we decided to step up our application to next level by enhancing it to a virtual makeover application.

Before amending the design of our iPhone application, we had meeting with our supervisor and we all agreed that it is a good idea to enhance it to a makeover application. To have a whole picture of virtual makeover application and the user interface, we spent some time to look into three different popular virtual makeover 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 makeover applications are Hair Makeover, Photo Makeover and iMakeup Unlimited – Never Forgot every makeup brand and color you own!

2.10.1 Hair MakeOver by Touch Apps

It is a photography application that supports 14 languages and allows user to find his/her best hairstyle. First, user can pick a photo from library or take new with camera, try out a wide range of hairstyles for both women and men, adjust his/her photo to suit the hairstyle and find out his/her favourite hairstyle. The final preview can be saved and shared with his/her friends through facebook.

Though there are a lot of hairstyles to choose from, yet, the effect is not as good as it mentioned. It takes time to change the styles and indeed hard to adjust the photo to fit the hairstyle.

2.10.2 Photo Makeover by Reallusion Inc.

Photo Makeover is a photography application that allow user to take a snap shot and then edit it with cosmetic magic to change facial features and adjust facial expression with just a few swipes. User can edit the general shape and size of the facial expression to create full and dramatic makeovers. There are altogether five main features including a library of pre-made facial expression templates created by analyzing thousands of faces, 5 selectable facial regions each with move, rotate and size controls, instant pro photo editing results with one-touch templates, automatic color balancing, face detection, import/export photos to facebook.

2.10.3 iMakeup Unlimited – Never forgot every makeup brand and color you own! by Dash Technologies

iMakeup Unlimited allows user to save all the makeup or cosmetic brand and color inside the iPhone. Instead of saving just one lipstick or just one mascara, now user can have unlimited lipsticks brand, color and notes. It also allows user to share her favourite makeup or cosmetics via email.

However, this application has only very few features, after all, it is only a note-taking application with email feature.

3. Application Design

3.1Ideas Brainstorming

Via comparison between different makeover applications, altogether 2 draft ideas came up in our mind at the very beginning of the design phase.

3.1.1 The Hat Shop

It has a wide range of hats for users to choose from ranging from graduation hat to safety hat, witchy hat to English hat

It can also take video clips with both the hat and a special background, like roller coaster, space…etc

The effects are chosen before the user takes the picture, and the preview shows the effect already applied before the user saves the photo or video.

3.1.2 Welcome to Salon

It has a wide range of hairstyles for users to choose from ranging from long hair to short hair, straight hair to curly hair.

This can help user to have a preview of the desired hairstyle to see if it suits you.

The effects are chosen before the user takes the picture, and the preview shows the effect already applied before the user saves the photo or video.

3.1.3 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

3.2 Final Design

Among the ideas that we have brainstormed, we can build adopt the idea of building the application, “The Hat Shop”. All we need to do is to replace the other than working on a easier approach, we decided to pick up this great opportunity and challenge to make a makeover application. Like last semester, our target is still to develop an application that is original, meanwhile, entertaining. In this semester, our final design is to develop a virtual makeover application that not only includes useful and handy features, but also give users an awesome and fun experience.

3.2.1 Sepcification of iPhone makeover application – FaceLook

Introduction

The well-developed iPhone, undoubtedly, is the user-friendliest mobile device ever existed. To make iPhone to be a more comprehensive device, a virtual makeover application, FaceLook, is going to be introduced to all iPhone users. FaceLook is a photo editing tool that can give user a total fun experience. With the auto face-detection, eye-detection and mouth-detection technology, user can easily apply different kinds of makeup onto their face to create a cool look and save the look to their photo library. For instance, user can first pick a photo from the library or take one by camera. User can simply tap the lipstick button and pick her favourite lipstick colour on the colour palette. The system will then automatically detect the lip location on the photo and user can further adjust the size and postion of the lipstick applied if needed. User can also select the color on the color palette to change the lipstick color and the preview will be shown immediately. The eye shadow button and blush button works similarly like the lipstick button. The eye shadow button and the blush button can detect the eye and the face location respectively. User can saved all the product saved to a note with the finished photo in the photo library in the end.

It is an excellent application for all the girls that want to find out her best pick before going to the drug store to get her new lipstick, eye shadow palette or blush. Unlike other makeup application, all the colours used in our application are picked from the famous cosmetic brand, Bobbi Brown Cosmetics. You can actually buy the product according to the list saved! Isn't it so convenient?

For the boys, it would be a fun application for you too! You can take a picture of your buddy and try to put makeup onto his face! Can you imagine how your friend looks like after putting on the eye shadow? It would be an absolute fun experience!

Objective

- To allow user to enjoy and have fun
- To allow user to try different makeup on before buying the actual products
- To allow user to share the photo and the used product information through email

Target User Groups

- Females of all range
- Teenagers aged 15-25

Target Features

Elementary features

- Load image from Photo Library
- Save image into Photo Library
- In-app photo taking
- Face detection on static photo
- Create various color palettes

Advanced features

- Eye detection on static photo
- Mouth detection on static photo
- Image layering that allows movement and resize by touch
- Draw shapes on the image by using control points
- Blending the image

3.3 Comparison between the similar photo applications and our target application, FaceLook

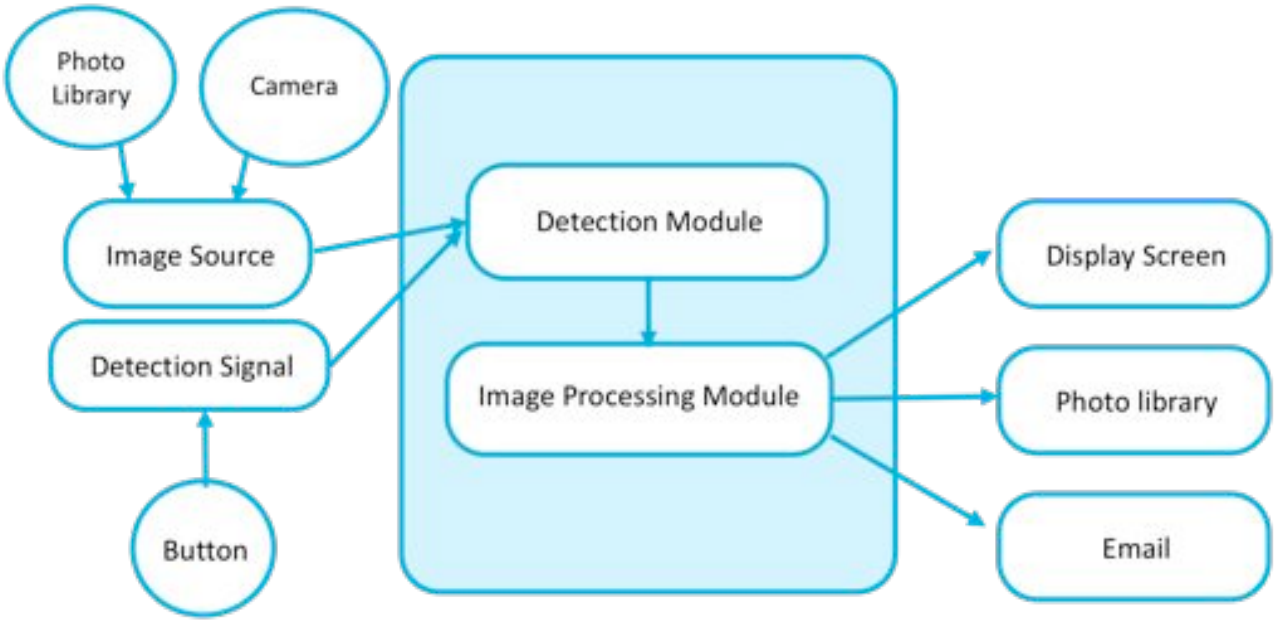
Features / Applications	**Hair Makeover**	**Photo Makeover**	**iMakeup Unlimited**	**FaceLook**
In-app Photo Taking	✔	✔		✔
Load image from library	✔	✔		✔
Save image to library	✔	✔		✔
Face Detection (on iPhone)		✔		✔
Eye Detection (on iPhone)				✔
Mouth Detection (on iPhone)				✔
Image Layering (allow touch movement, resizing image)	✔			✔
Blending the image				✔
Note-taking			✔	✔
Export to facebook	✔	✔		
Export to email	✔		✔	✔

3.4 System Architecture

Photo is selected from the photo library or the camera as the image source and then passed to the main program. Different button click will generate different detection signal for the detection modules. For instance, the lipstick button click will generate a lip detection signal to the detection module. When the detection module receives the detection signal, the image will undergo the detection process to locate the position for face, eyes, lip. Then, the image and the respective location information will be passed to the image-processing module to edit the effect on the photo. The processed image will then be passed to the screen for display.

Furthermore, different color button will pass the color signal to the image-processing module for changing the color of lipstick, eye shadow or blush. The amended image will be passed to the screen for display immediately.

The processed image can be passed to the photo library to save as a new image or exported to email as attachment.



3.5 Modules design

3.5.1 Detection Module

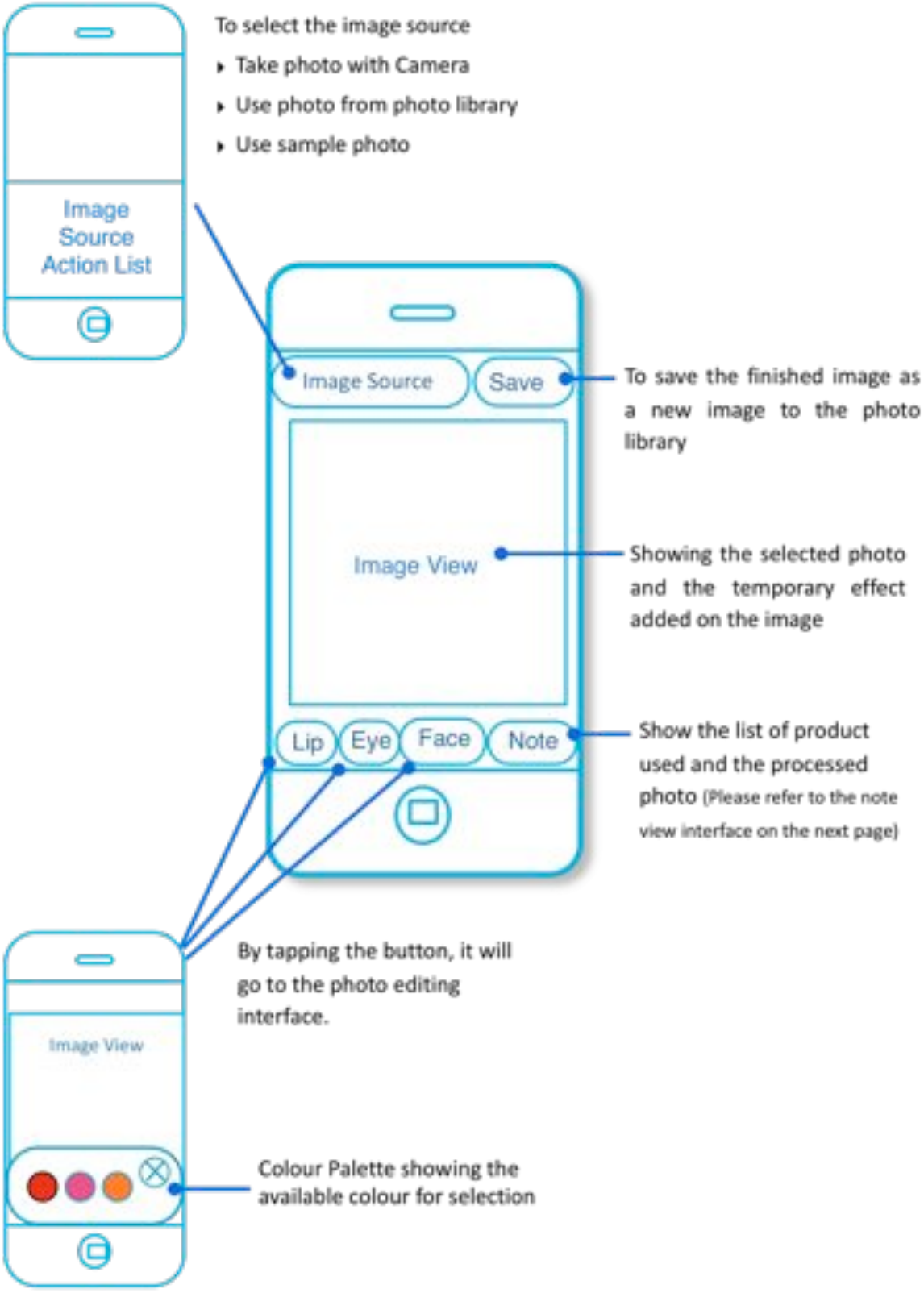
In this module, the objective for this module is to detect the location for lip, eyes, and the face in the photo. It will first do some preprocessing jobs like resizing the photo and turning the image into grey-scale. Then it will start the detection process, which is locating the face, lip or eyes within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

3.5.2 Image Processing Module

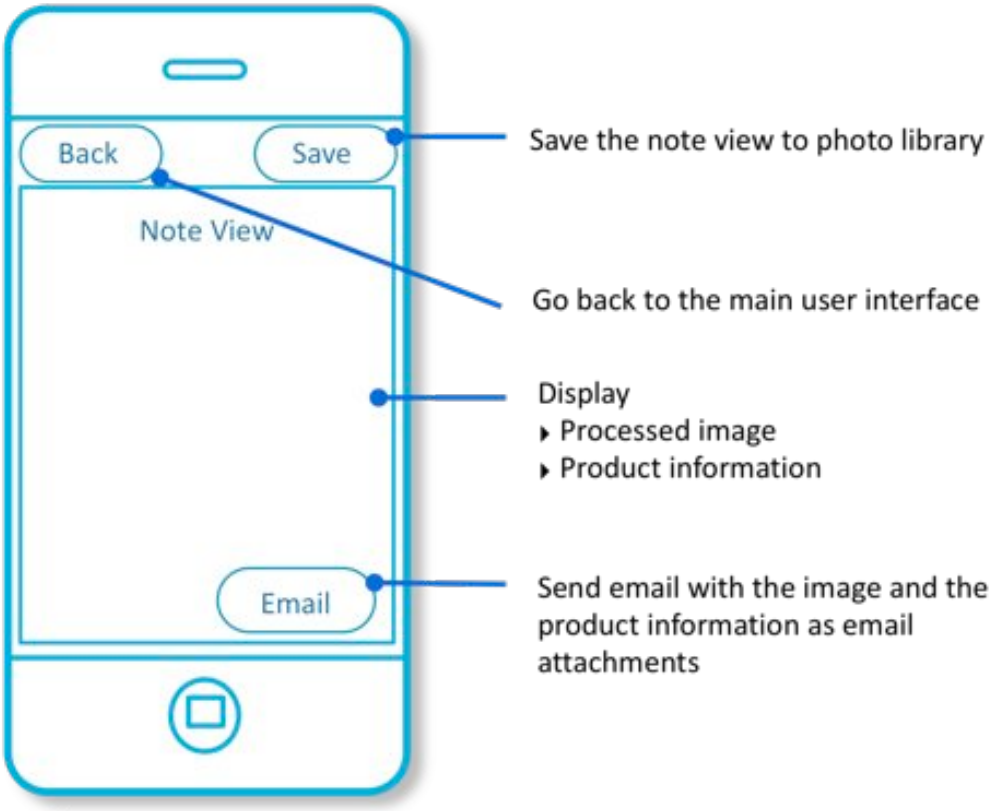
This module handles all the image processing work, including image layering and the touch handling. After getting the location of the facial feature from the detection module and the color information from user, this module will overlay a semi-transparent image layer on top of the photo to act like the applied cosmetics. The color of the image layer will change according to user required. The size and shape will be change according to the control points.

After editing the photo, it is responsible for passing the processed image to the screen for display, photo library for saving the image, email system as an email attachment.

3.6 FaceLook – User Interface Design



Note View Interface



3.7 Project Schedule

Month	Week	Assigned Work
January	Week 1	- Review on the progress of semester one - Review on the top iPhone application available on iTunes App Store
	Week 2	- Ideas Brainstorming
	Week 3	- Draft of the Application Design - Draft of the User Interface
February	Week 4	- Update the new project specification - Implementation the image layering
	Week 5	- Implementation the touch movement - Get familiar with the theory and algorithm of eye detection and mouth detection
	Week 6	- Implementation the eye detection feature - Implementation the mouth detection feature
	Week 7	- Implementation the eye detection feature - Implementation the mouth detection feature
March	Week 8	- Implementation the eye detection feature - Implementation the mouth detection feature
	Week 9	- Implementation the eye detection feature - Implementation the mouth detection feature
	Week 10	- Testing on iPhone 3G and 3GS - Debugging
	Week 11	- Testing on iPhone 3G and 3GS - Debugging - User interface
	Week 12	- Debugging - Combining all features - User interface - Business Plan
April	Week 13	- Documentation - Refinement and Amendment of program - Business Plan
	Week 14	- Documentation - Refinement and Amendment of program - Business Plan

		January			Febrary				March					April	
Design Phase	Review on progress of 1st Term														
	Review on makeover applications														
	User-Interface Design														
	Application Design														
	Application Specification														
Implementation Phase	User-Interface Implementation														
	Combine Feature														
	Face detection feature implementation														
	Image Layering feature implementation														
	Debugging														
Testing Phase	Testing on iPhone simulator														
	Testing on iPhone 3GS and 3G														
	Refinement and Adjustment														
Report	Documentation														
	Business Plan														

4. Implementation Phase

Since we have implemented SNAP! in the last semester, we have already had some idea how to use the Interface Builder and Xcode. We found the interface builder is simple and easy to use, so we continued to use Interface Builder to build FaceLook in this semester. Face detection used in SNAP! is also useful for the implementation of FaceLook. We therefore will apply what we have learned in semester one to build our application.

The programming language used will be Objective C. And the developing platform will mostly be Xcode and Interface Builder on the Mac OS.

In the beginning of the implementation phase, we divided it into three main parts: user interface creation, OpenCV detection (include face detection, mouth detection and eye detection) and image layering.

4.1 Part 1: User Interface Creation

By referring the experience in semester one while implementing SNAP!, we found that Interface Builder is quite a good choice for creating the user interface in a simple way which need not much code implementation have to be done. All the elements in the Interface Builder are standard which users can easily adapt to. With the help of the iPhone SDK API, UIKit, most of the user interface is created by using the Interface Builder.

"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.

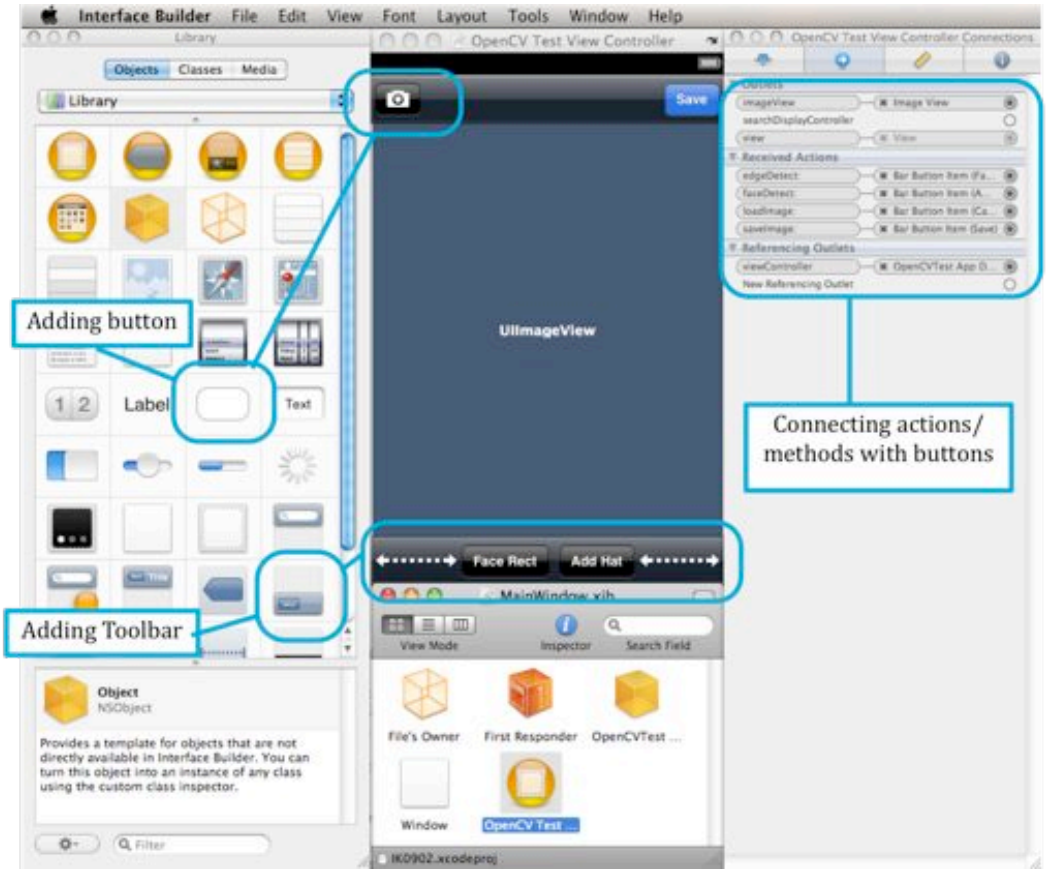


Figure 3.1.1 Basic view of Interface Builder

"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.

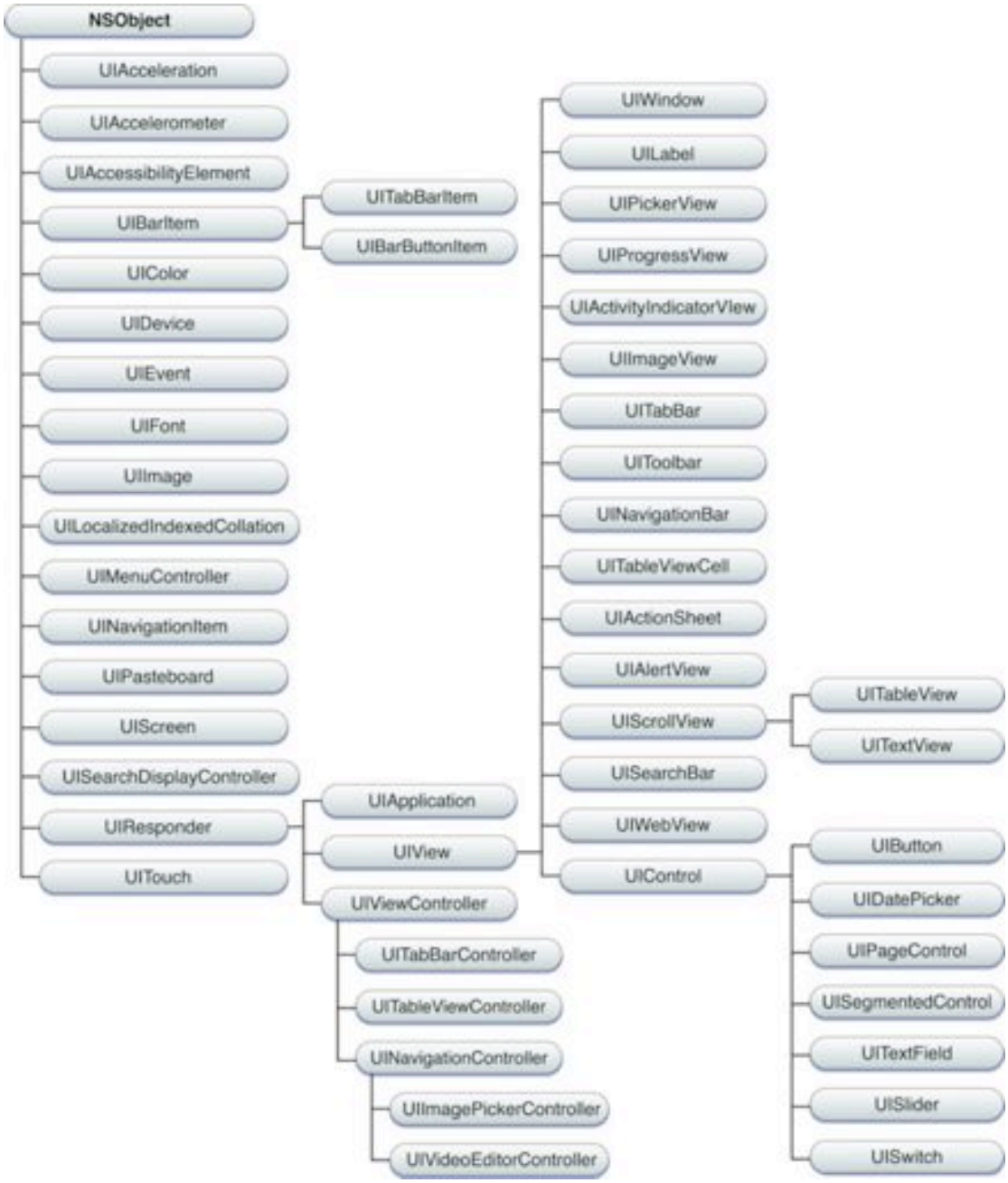
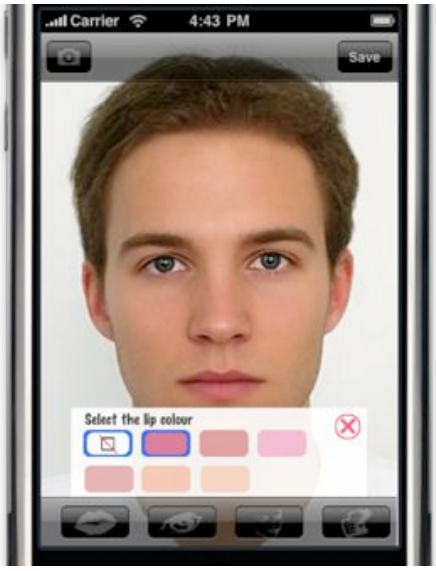
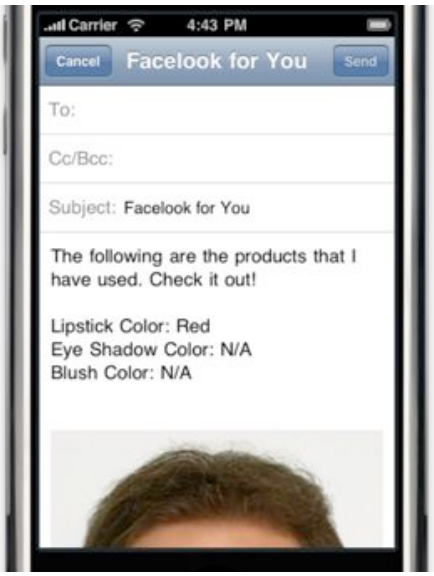


Figure 3.1.2 UIKit class hierarchy

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





4.2 Part 2: OpenCV Detection

To start implementing the detection features, we have to get familiar with the algorithm and the detection theory of the OpenCV library. The algorithm is shown in figure 3.2.1.

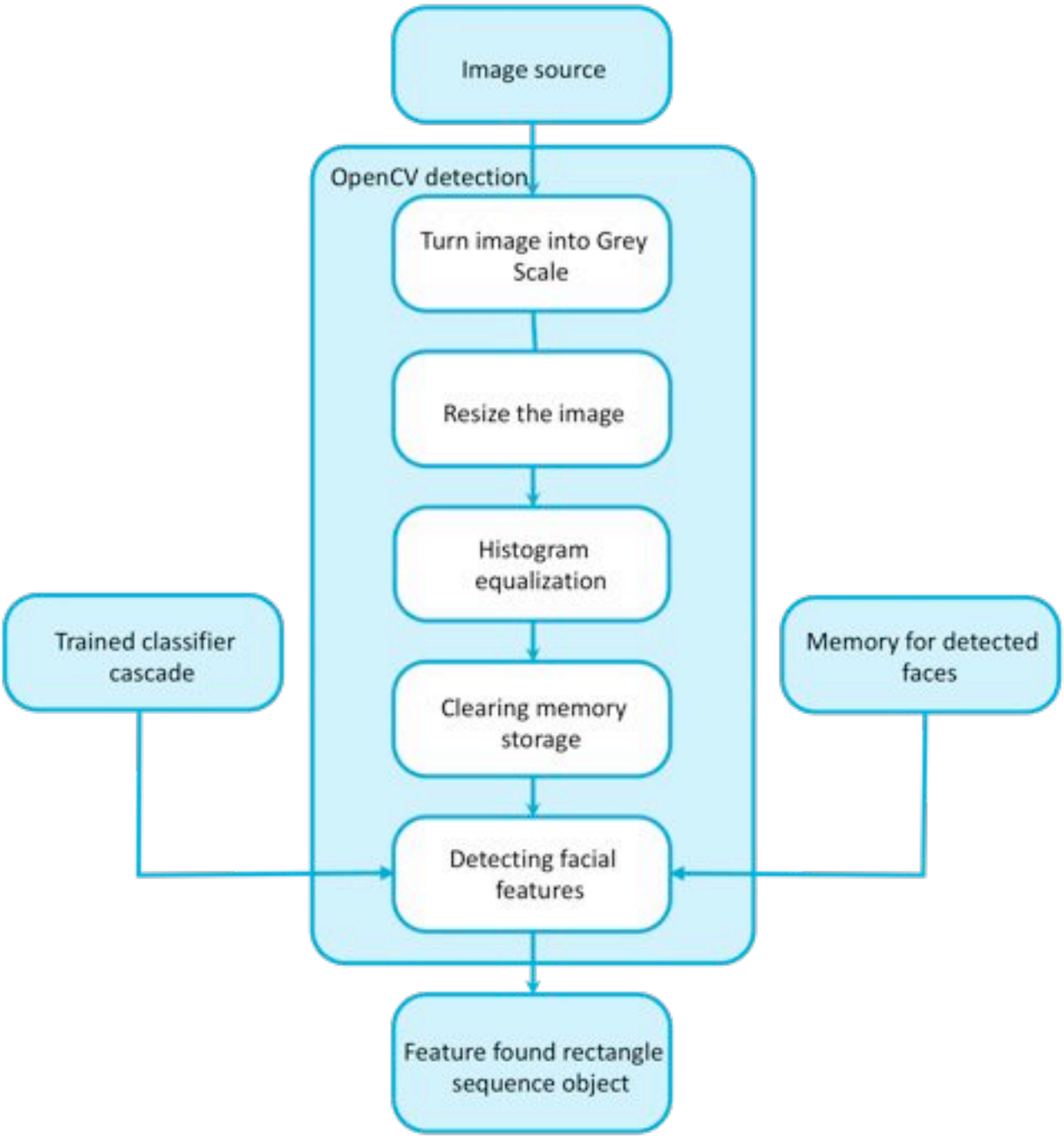


Figure 3.2.1 The algorithm for OpenCV detection

At the very beginning of detection phase, we need to load a previously trained classifier cascade for the future use by the detection part. And where the detection of mouth and eyes are working similarly.

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.

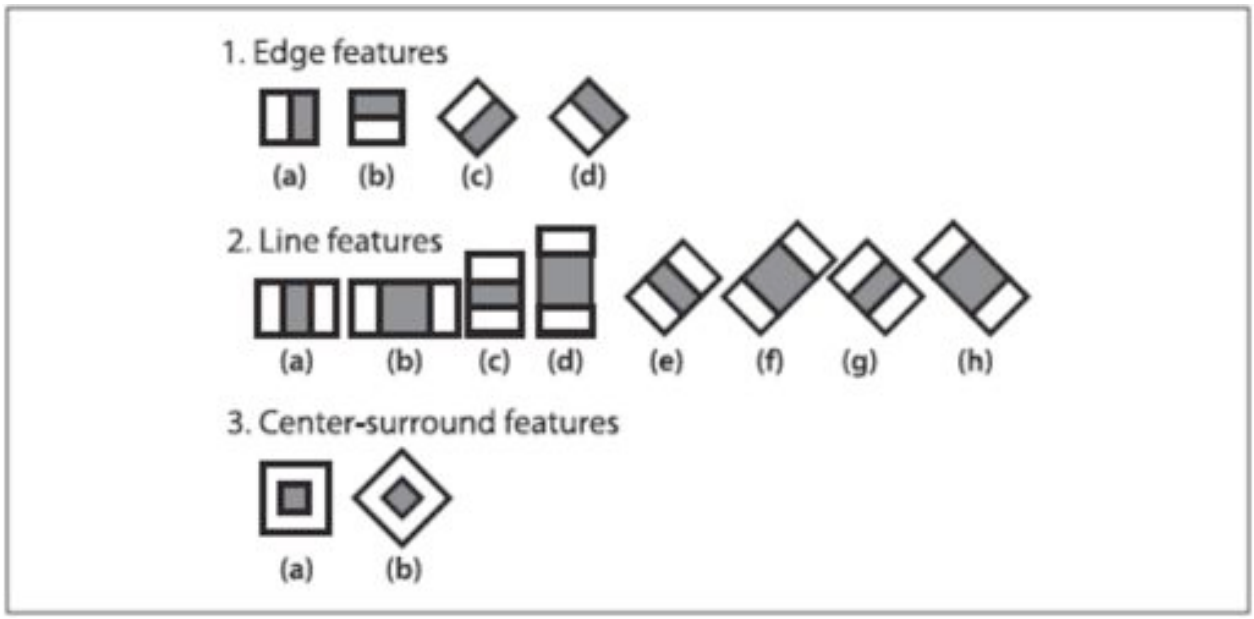


Figure 3.2.2 Haar-like features from the OpenCV source distribution (the rectangular and rotated regions are easily calculated from the integral image): the light region is interpreted as "add that area" and the dark region as "subtract that area"

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.

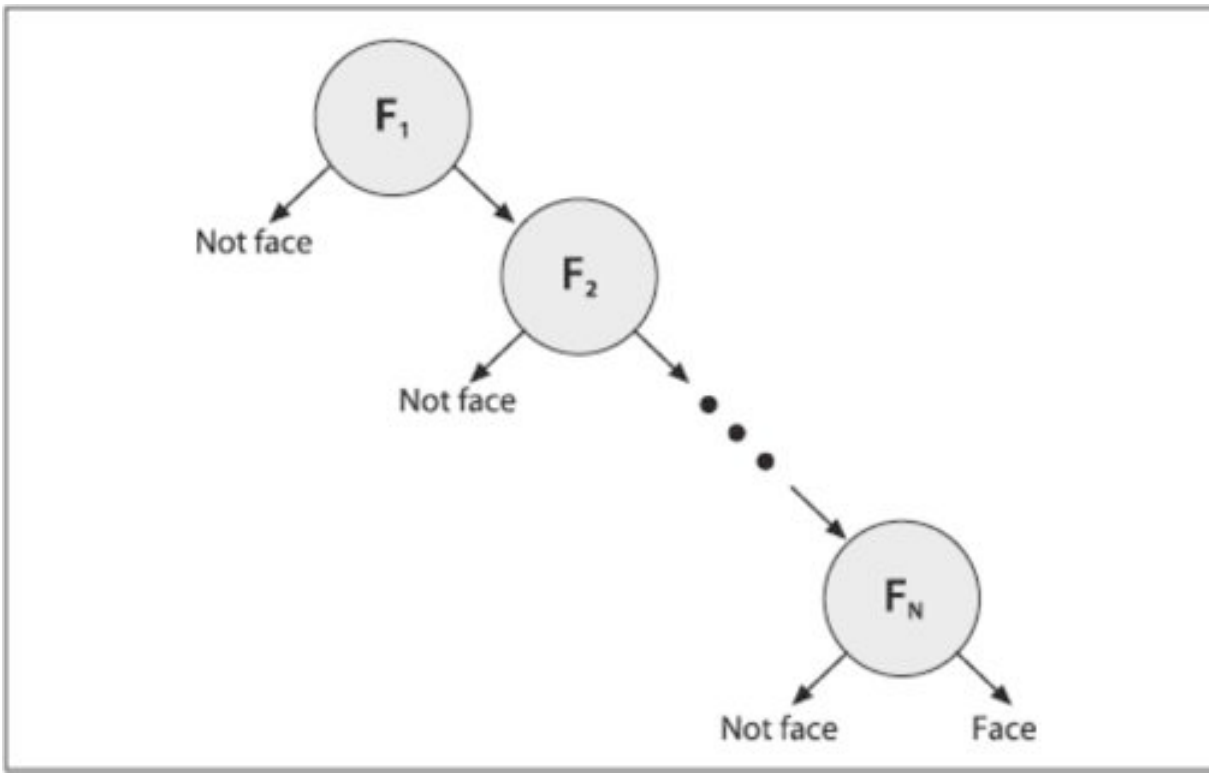


Figure 3.2.3 The rejection cascade used in the Viola-Jones classifier.

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

4.2.1.1 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.

4.2.1.2 Resizing the grey image.

This action can speed up the further face detection steps with a smaller and simpler image.

4.2.1.3 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.

4.2.1.4 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.

4.2.1.5 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.

4.2.1.6 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.

There are some assumptions for the OpenCV detection in order to increase the accuracy. To identify the facial features in an image, it is necessary that the face on the photo is facing to the front, well lit, not blurred and not covered by anything. Passport photo is a good example for the detection. Preferably, the photo's background should be neutral.

By learning the past experience from SNAP!, we found that, detection function takes the longest processing time of all the actions in the application. Although face detection has a high accuracy, the accuracy for eye detection and mouth detection is not as high as face detection. This is out of our expectation that it takes a lot of time to adjust the threshold. The main concerns about detection facial features would be the low accuracy in eye and mouth detection and long processing time problem.

4.3 Part 3: Image Layering

After the program found out the position of facial feature(s), the feature sequence can be used for further image processing. For instances, adding image(s) on top of the detected area, cropping out the face and place on a different image like PhotoFunia... etc.

For SNAP!, one semester earlier, all the overlay images (like, hat) that will be added to the image are directly drawn onto the presence image. After the overlay image is drawn, it can't be moved around the view or deleted from the view. Indeed, there isn't any image view to store the overlay image.

After amendment, all the overlay images are stored as the subviews of the image. In short, all the overlay images are the children of the main image, whenever the main image has any changes, the overlay images will then follow the change accordingly. Also, all the overlay images are independent of each other in such a way that any change of one overlay image will not affect the other overlay image. This provided users the flexibility for adjusting the detected position and size of the overlay image.

At the present stage, the detection module help to find out the position of the facial features . By using the image overlay method, an overlay image can be added to the photo at the position of the specific facial features, meanwhile, changing the color of the overlay image according to the color chosen in the color palette. This can allow user to have a preview of how she looks after applying certain colors onto her face in a few seconds, rather than spending 15 minutes to apply the real makeup onto her face.

4.4 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, the iPhone 3G S and the iPhone 3G have a significant difference.

Operating in different devices, the speed of the detection process varies. Therefore, it is hard to determine the detection threshold that is suitable for both iPhone 3G and 3G S. Moreover, it also hinders the development process as it requires us to build in the real devices for actual performance.

2) **Low accuracy for eye and mouth detection**

As the accuracy for eye and mouth detection is relatively low comparing to that of face detection. The position obtained after eye or mouth detection may not be accurately shown due to the detection failure.

To solve this problem, we spent time on tuning the threshold of the eye and mouth detection. Yet, the result is not significant. Therefore, we decided to add some control points for fine tuning the overlay image in order to suit the main photo.

3)**Color blending**

For the time being, the color blending for the overlay image to the target image is just a simple color filling with transparent effect. With the present color filling mode, it is not really merge into the target image. This makes the overlay image like a color mask rather than cosmetic.

To improve the color applied, we have to apply some graphical technique to enhance the effect. For example, blurring on edges of the image will induce a gradient effect.

4)**Too small control points**

Although we used control points to allow user to manually adjust and modify the size of the makeup applied, it is a little bit hard for users to move around the control points. This is because the user's finger may cover a portion of the screen and make the tuning become difficult and less user friendly. Comparing to iPhone, iPad will have a better control in this case.

We may use a magnifier glass to resolve this problem. Whenever the user is going to fine tune the overlay image, a magnifier glass will pop out above the specific control points. This help users to see the portion of the screen that is being cover by his own finger and thus allow easy adjustment of the control points.

5. 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 facial feature detection. 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.

Phase 3 testing is used to test the accuracy and the performance of the eye detection and mouth detection.

5.1 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.

5.2 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.

Phase2

- 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.

Phase 3

- Accuracy

  15 front-faced photos of people from different races are used for testing the accuracy of eye detection and mouth detection separately. The eye detection succeeds if it can detect the correct positions of two eyes. The mouth detection succeeds if it can detect the correct position of the mouth.

- Performance

  A photo is passed to the iPhone simulator, iPhone 3G S and iPhone 3G to test for the performance. The time needed for eye detection and mouth detection on each device is recorded separately. To compare the result, we also record the time required for detecting the eye and the mouth at the same time.

5.3 Evaluation and Results

5.3.1 Phase 1 – Horizontal rotation angle of the face

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

Left 15°	Left 20°	Left 25°	Left 26°
			
Succeed	Succeed	Succeed	Fail

From the above testing, it shows that the maximum rotation angle to the left side is about 25°. Rotating 26° 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°.

5.3.2 Phase 1 – Vertical rotation angle of the face

Left Rotation

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

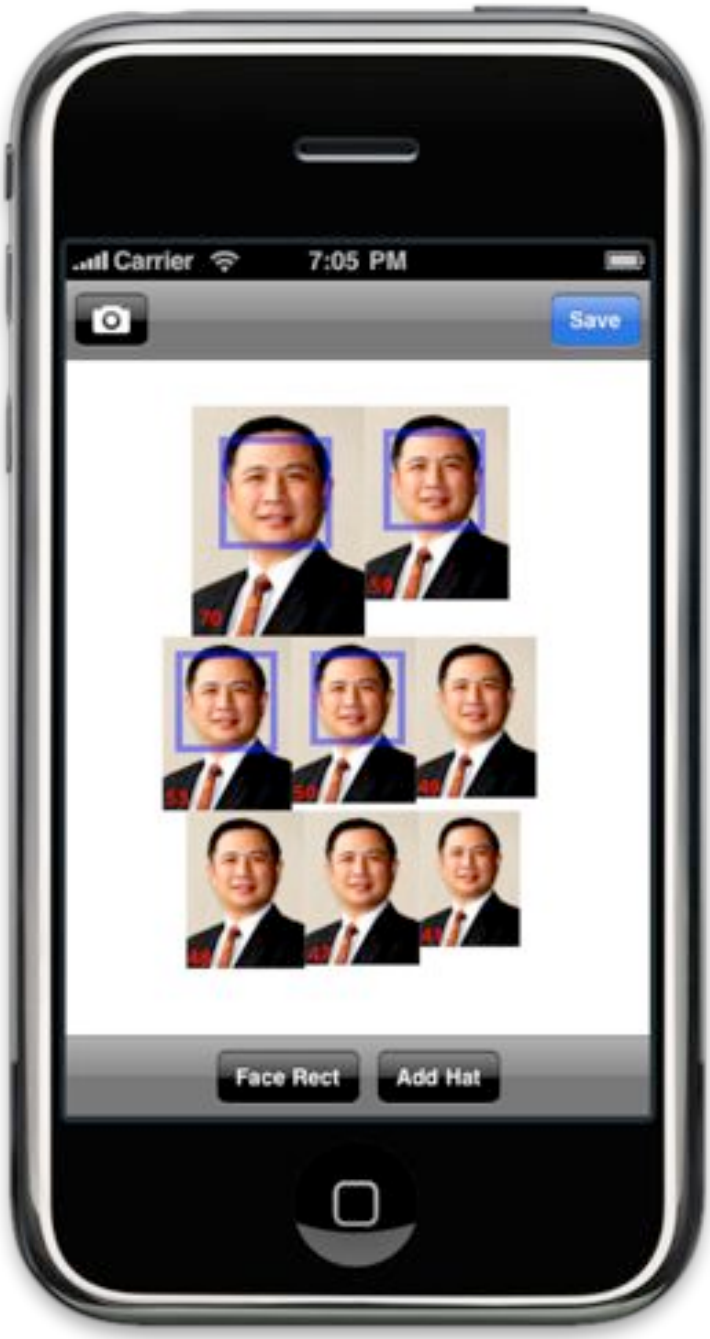
For the left rotation, the maximum rotation angle is about 60° while the one for right rotation is 50°. Since the testing photos are captured from a video, there may be imprecision due to the inconsistent movement of the face in the video.

5.3.3 Phase 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.

5.3.4 Smallest size of face in photo



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

5.3.5 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

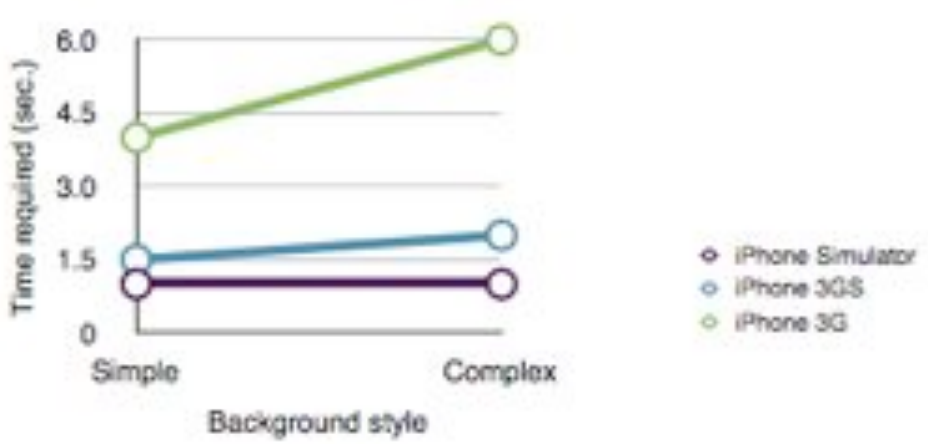
From the above graph, it shows the time required for detecting different number of faces for different devices. Comparing with iPhone 3G S, iPhone 3G needs 2 times the time required for iPhone 3GS to detect faces. The time required increase slightly with the number of faces detected.

Photo with simple background

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

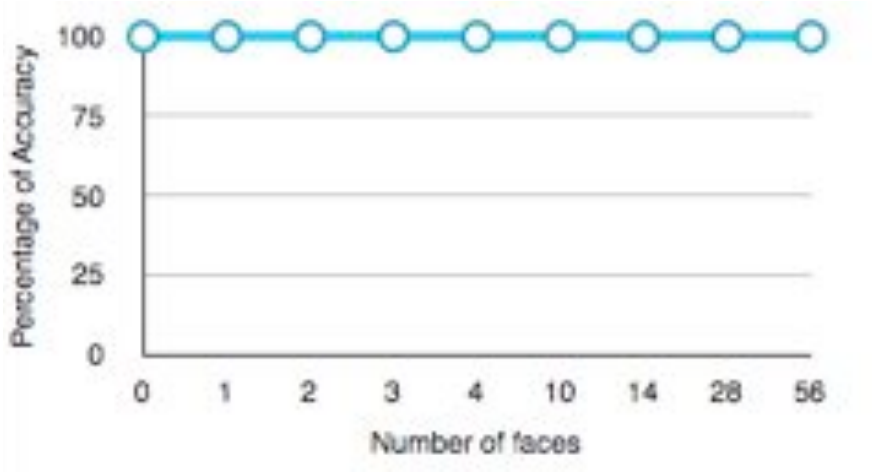
Photo with complicated background

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.

5.3.6 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% .

5.3.7 Phase 2 – Consistency

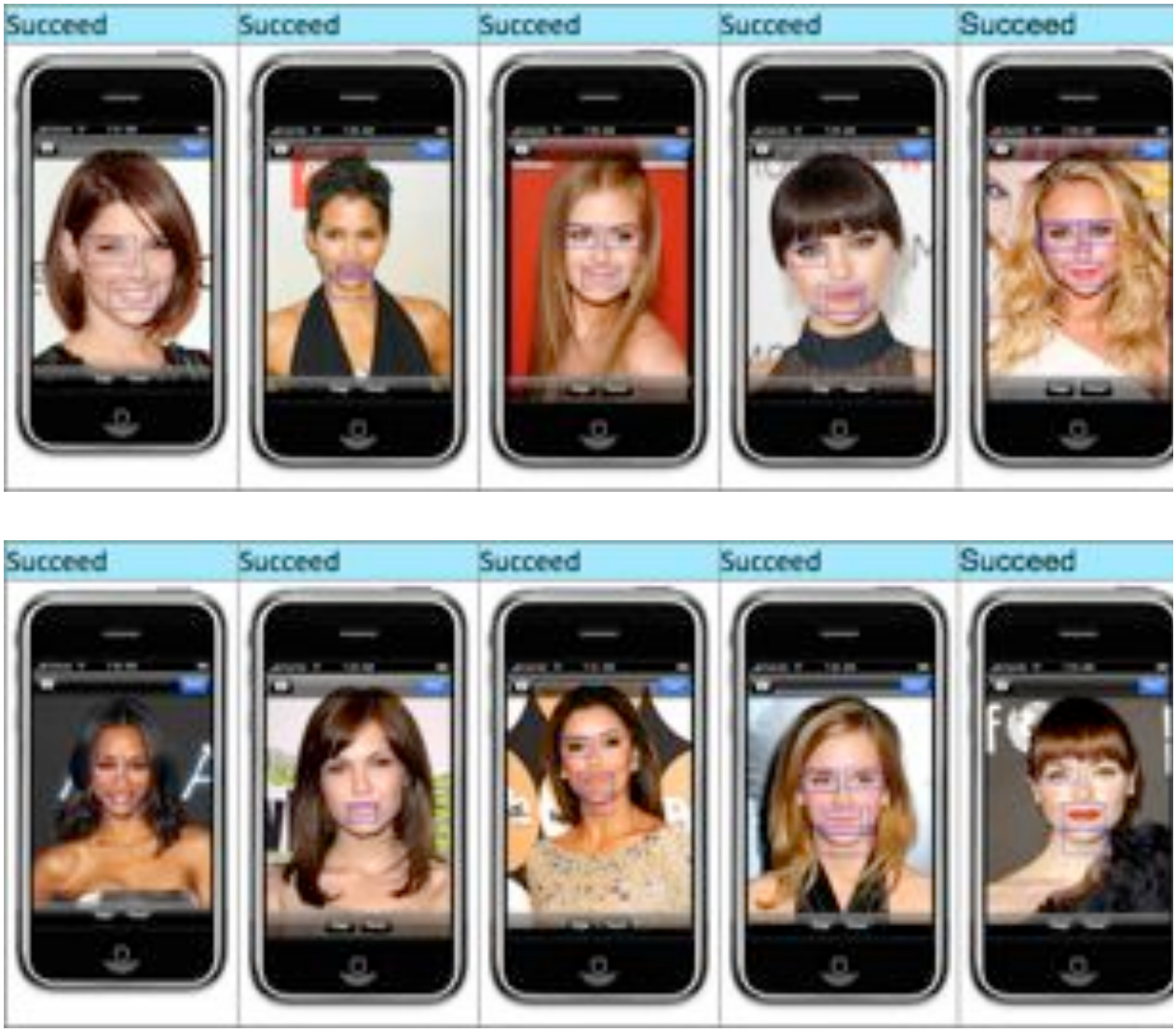
The program produces the same results for detecting the same set of photos in various time. The processing time for the each device to detect the same set of photos is also approximately the same.

5.3.8 Phase 3 – Accuracy

Eye Detection

As shown from above, the system can detect 8 photos correctly out of 15 photos. The accuracy of the eye detection is therefore 53%.

Mouth Detection



The system can detect the mouth location for all 15 photos correctly. The accuracy is therefore 100%. Yet, the system also detect the location of eye as the mouth location for 11 photos among 15 photos.

5.3.9 Phase 3 – Performance

Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	0.5 sec	3 sec	7 sec

Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	0.5 sec	3 sec	6.5 sec

Machine	iPhone simulator	iPhone 3GS	iPhone 3G
Time required	1 sec	6 sec	13 sec

5.4 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° Right: 24°
Maximum Vertical Rotation Angle of Face	Left: 60° Right: 50°
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 detection 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.

Phase 3 testing is used to test on the accuracy and the performance of the eye detection and mouth detection. Before the testing, we expect the rate of accuracy will be similar to face detection, which is high. However, the accuracy test for eye detection is only 53%, which is much lower than that of face detection. Although the system can find all the mouth locations, we found that the system will treat the eye location as the mouth location for over 73% of photos. Therefore, if we only count those that can detect only mouth location correctly, the accuracy will be just about 27%, which is very low. Since the accuracy for both eye and mouth is not as good as expected, it obviously needs adjustment of the threshold.

The performance test shows the time needed for different devices to perform the eye detection and mouth detection. We found that the processed time is acceptable for iPhone 3G S, but the time doubled for iPhone 3G, which is about 6-7 seconds. The time needed to perform both eye-detection and mouth detection together is basically the sum of the time required for detecting eye and mouth separately. Therefore, there is no advantage to perform the eye and mouth detection together in the application, especially when the iPhone 3GS needs 13 seconds which is totally intolerable.

5.5 Job Division

5.5.1 Distribution of Work

Work		Joseph Ye	Sonia Chung
Design Phrase	Research on applications	✔	✔
	Specification(+System Architecture)	✔	✔
	Interface Design		✔
Implementation	User Interface		✔
	Face Detection	✔	✔
	Eye Detection	✔	
	Mouth Detection	✔	
	Image Layering		✔
	Control Point for drawing	✔	
	Touch movement	✔	
	Export to Email		✔
	Application Refinement	✔	✔
Testing	Phase 1 testing - Face detection range	✔	✔
	Phase 2 testing - Performance, Accuracy, Consistency	✔	✔
	Phase 3 testing - Eye & Mouth Detection Range	✔	
	Phase 4 testing - Application Testing		✔
	Analysis Testing Data		✔
Documentation	Mid-term Report	✔	✔
	Final Report	✔	✔
Others	Icon Design - SNAP!	✔	✔
	Icon Design - FaceLook	✔	✔
	Cover page Design	✔	✔

5.5.2 Personal Reflection on the Final Year Project – Sonia

Final Year Project has always been a great challenge to me. Since I am not a fast learner in programming, it is hard for me to learn a new programming language all by myself in a very short time. Luckily, there are developer forums in the internet that the developers from all over the world shared a lot of useful tutorials, ideas and precious experience in developing iPhone applications. Moreover, joining the WWDC last summer is an awesome experience. Via the talks and lab sessions, I got a lot of inspirations from the talented developers.
My partner, Joseph, has always been supportive and he explained a lot when I have doubts in coding. While Joseph was more concentrated to work on the detection modules which is the major part of our application, I was responsible to design, implement the side parts like the interface, email system and image overlay. I enjoyed working with Joseph as we can share the work evenly and help each other whenever needed.

As mentioned above, I was responsible for designing the interface. At the beginning, I didn't realize how important the user interface means to an application until our professor consistently emphasized the importance of a user interface. Just look at Apple's products, good interface led Apple to a consistent success. User interface is what the user direct interacts with, it straightly affects how the user feels about your application. I tried hard to create a simple user interface that is easy to use and cause as less confusion as it can. I chose to build the interface by using the Interface Builder as it had all the standard elements of an interface like the toolbar, round button, image view...etc. It is easy to start with and it gives the user the standard layout which is easy to use. Yet, as all the things in the Interface Builder is standard, it doesn't allow much variation. Therefore, it would be better for game developers to create the user interface by UIKit, Quartz and OpenGL ES as it allows a more interesting interface. I also helped to make the image overlay and email feature

of the system. It is a fun practice that I had to learn all the codes from the tutorials online and go through a lot of "try and error" process before succeed. I felt so satisfied when I got it done.

I also took part in doing the testings and analysis. We separated the testing into four phases. Phase 1 testing is mainly focus on the limitation of the face detection feature and Phase 2 testing is used to test the performance accuracy and consistency of SNAP!. Phase 3 testing is used to test about the accuracy and performance of eye and mouth detection. The testing process is very important to the application development as it provides solid data for analyzing the application and reveals the areas that need improvement and refinement.

Comparing to the first semester, we may not be able to implement the whole thing in a very short period of time. This is because we spent quite a lot of time to rebuild the whole system and solve the low accuracy of eye and mouth detection. I think it is easy for us to implement an application like the Hair Makeover, as we have already done the SNAP! in the first semester. All we need to do is just changing the hats to wigs. Yet, we really want to learn more through the FYP and we endeavored to struggle for the best we can. That's why we chose to take the challenge and change our original application to a virtual makeover application. Admittedly, we failed to consider the fact that we have to complete the whole thing in about 3 months time. I would take FYP as a good lesson as I understand that I should aim high, meanwhile, consider other factors involved. Anyways, I do enjoy working on my final year project and I would really love to see it on the App Store!

5.5.3 Personal Reflection on the Final Year Project – Joseph

The final year project gave me a great change to have an experience to go through the whole application development process. We started our application with just some ideas from brainstorming. In the design phase, we started to think of our design of the application seriously and document the specification, system architecture, modules design... etc. We started to build our first application by reading books, watching tutorials on YouTube. It is a new and fun experience when we saw the application simulate in the iPhone simulator at the very first time. We then explored to something more difficult, which is the facial feature detection, implemented the detection, tested and finished it - SNAP! It is actually very impressing to see the application done, even though it is just a small milestone in our development process. It would be awesome when we can get the application upload to the App Store. I really enjoy very much in joining the whole development process.

Through the implementation process, I learned skills in programming in Objective C language and using the iPhone developing platform, Xcode & Interface Builder. In the implementation of the detection module, I also used OpenCV for detecting the face, eyes and mouth locations in the photo. To have a better understanding of the detection process, we have read the books to learn about the theory and algorithm within the detection process. Apart from that, we also discussed and shared our progress with other FYP groups. This learning experience equipped me to learn independently and actively.

I enjoy working with my partner, Sonia, as we are both good at different areas and we can share the work respectively. Not only we can shoulder different aspect of duties, we also have very different ideas and thought which allows a lot of inspirations during discussion. This is a very good chance for me to learn about the whole process of developing a program with teamwork's. And I am looking forward to see my App is on the App store.

6. Conclusion

6.1 Accomplishment

In the first semester, we successfully implemented the basic photo application, SNAP!, that can detect multi faces and allow user to add different hats onto the image. During the development progress, we 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.

In the second semester, we have used what we learnt in the first semester to build our second application, FaceLook, which can allow user to apply virtual makeup onto his/her photo to create a makeover. The processed photo and product information can be exported to email as attachments. During the implementation, we got a better understanding how to use the Interface Builder to create a better user interface. We also learnt to apply the detection algorithm to detect the eye and mouth in our application and perform the image layering.

6.2 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. It provides us such a fascinating opportunity to explore in the iPhone industry and go through the whole application development process.

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 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.

6.3 Further Development

By the end of the development period, we have done a review on our progress and there are still rooms for improvement. We may include the following feature or enhance the application and release a updated version of the application in soon future.

- Performance Refinement

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

- Adding magnifying glass

  Since the control points are too small to move around, by adding the magnifying glass feature, it will magnify the selected area for user to have a clearer view.

- 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.

- Color Blending

  To improve the effect of the makeup applied, we have to apply some graphical technique to enhance the effect. For example, blurring on edges of the image will induce a gradient effect which will make the overlay layer looks more realistic.

7. 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 Apr 2010.
- 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 5 Apr 2009.
- Hair Makeover. (2009). Hair Makeover. Available: http://www.hairmakeoverapp.com/. Last accessed 20 Apr 2010.
- Reallusion. (2010). Reallusion. Available: http://iphone.reallusion.com/. Last accessed 20 Apr 2010.
- Eric Slivka (April 8, 2010). "Apple's iPhone OS 4.0 Media Event: 'Sneak Peek Into the Future'". MacRumors. Retrieved April 8, 2010.

8. Appendix

8.1 Application Cover Page

8.1.1 FaceLook

FaceLook

Category: Photography

Updated 20 April 2010

Current Version: 1.0

(iPhone OS 3.1.2 Tested)

Publisher: Department of CSE, CUHK

3.2 MB

APPLICATION DESCRIPTION

Have you ever thought that you can try ALL colours of lipstick, eye-shadow, and blush? Have you ever thought about your look after putting on makeup?

FaceLook is here for you!

FaceLook allows you to apply virtual makeup onto your photo. After selecting a front-faced photo, all you have to do is a simple button tap of the facial feature and the colour. Our application will automatically detect the according location and put the makeup for you. You can also adjust the size and shape by your own! The interface is simple and easy to use. It would be a perfect tool for you to have fun in taking pictures!

Let's try to use FaceLook to create some fun makeover!

Tags: makeover, face detection, eye detection, mouth detection

LANGUAGES:

English

Requirements:

Compatible with iPhone and iPod Touch

Requires iPhone OS 3.0 or later

Screenshot of FaceLook

8.1.2 SNAP!

SNAP!

Category: Photography

Updated 2 December 2009

Current Version: 1.0

(iPhone OS 3.1.2 Tested)

Publisher Department of CSE, CUHK

3.2 MB

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 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

Screenshot of SNAP!