IMAGE-BASED OBJECT SEARCH ON ANDROID

By

Ting Lay Then

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

JAN 2014

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Image-Based Object Search On Android

Academic Session: Jan 2014

I ____________________TING LAY THEN_________________________

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes

Verified by,

_________________________ _________________________

(Author's signature) (Supervisor's signature)

Address:
37, Prsn Rapat Baru 9
Taman Song Choon
31350 Ipoh, Perak.

____
Supervisor's name

Date: _____________________ **Date**: ____________________

IMAGE-BASED OBJECT SEARCH ON ANDROID

By

Ting Lay Then

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Perak Campus)

JAN 2014

DECLARATION OF ORIGINALITY

I declare that this report entitled "**IMAGE-BASED OBJECT SEARCH ON ANDROID**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	__________________________
Name	:	__________________________
Date	:	__________________________

ACKNOWLEDGEMENT

I would like to give my thanks to my supervisor, Mr Tou Jing Yi for giving guidance and direction through my research on FYP. The project provided me an opportunity to engage into computer vision field. Throughout this FYP, I able to learn more than what my course is being offered, much independent, and enhanced my problem solving skill on tackling all the technical questions I faced.

Furthermore, I would like to thanks Ms Yap Seok Gee for providing me very valuable advice and suggestion when I struggling for my FYP. I appreciated the time she spent on listened to the academic problems I had faced. Million thanks for her.

I would also like to thank the opportunity given by Christoph Göeth during my industrial training. The project assigned by him had greatly improved my knowledge on developing a mobile application with computer vision technique. Nevertheless, I appreciated the helps from my colleagues, Kristjan and Dusan, who helped me along when developing the application. Sincere thanks to them for giving me such an excellent intern experience.

Nevertheless sincere thanks to all my friends who support me during the FYP development. Last but not least, I would like to say million thanks to my parent for their dedication and many years of supports. Their love, support, encouragement, and patience to me will be the greatest treasure for my life.

ABSTRACT

Digital image processing refers to process digital images by means of computer. However, most of the current digital image processing applications are done on computer platform which installed with high performance hardware. It is inconvenience for end user bringing their laptop to use those applications ubiquity.

This project developed an Android application which allows users to **use different images to search for object** via computer vision technique. Two images are needed, namely template image and search space image. Search space image is compared with template image to match for the object. The objective of this project is to search for object that supporting scale and rotation invariant including partially occlusion scenario under 10s. The targeted object can be any colour and shape.

It uses **compare colour histogram technique** to reduce search area. Area of search space that having significant colour differences with the template image will be filter off.

Next, the filtered result will go for **direct pixel comparison**. The system read the template image's middle column pixel value. The middle column is compared with selected 5 columns of search space image. All of the 5 columns are nearby neighbour of the centre pixel. The returned result is compare again for its rows value.

Finally, the marked potential area compared again with its neighbouring pixels to ensure high accuracy of result achieved. Given matching object is found, it will mark colour boxes on the search space image. Otherwise, it will prompt error message to the user.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

OS	Operating System
PC	Personal Computer
OpenCV	Open Source Computer Vision Library
NDK	Native Development Kit
JJIL	Jon’s Java Image Library
SDK	Software Development Kit
FAST	Features From Accelerated Segment Test
SURF	Speeded Up Robust Feature
SIFT	Scale Invariant Feature Transform
RAD	Rapid Application Development
IDE	Integrated Development Environment
ROI	Region of Interest

Chapter 1 Introduction

1.1 Introduction

Compare to conventional mobile phone, smartphone has installed with a mobile operating system (OS) that has advanced computing capability and connectivity. There are varieties of mobile OS available in today market such as Android, iOS, and etc. Historically, computer vision or digital image processing only preformed on computer environment. Due to advancement in technology, digital image processing can be performed on smartphones.

Digital images can be created through drawing with graphical software or snapping picture with camera. Digital images are a numerical representation of a two-dimensional image where it consists of pixel of each block. Other than providing visual appearance, nowadays developers make use of the information stored on each pixel and integration of computer algorithms for digital image processing.

Unfortunately, smartphone platform have a limited resources which does not generate processing speed as fast as system unit. Currently object detection and object matching in mobile devices is in very limited domain. Changes in scale, orientation, or position could significantly affect the matching process. Moreover, reduce the efficiency and effectiveness of the result.

In this project, the author intends to develop an application on Android based smartphones which able to search for an object location based on the comparison from two images. The first input image will be the object source. It will search for the object location from the second input image.

1.2 Problem Statement

In order to obtain fast and high accuracy result in object matching, it is mostly accomplished using system unit. However, it is unlikely the end users will carry their personal computer (PC) in and out to perform object searching. Regrettably, smartphone devices have limited resources where the performance cannot compete with system unit. This is because smartphone devices is powered by portable battery and does not utilize high performance hardware.

As a result, those algorithms which provide robust object searching and matching result are computation expensive when execute on smartphones. Therefore, searching of object with Android smartphones is relatively slow due to most of the high quality algorithms require high computation time.

1.3 Motivation

Image processing on Android is a new field that had emerged few years back. The motivation of this project is to enable object searching can be performed on mobile devices. Although the existing research able to carry the functionality mentioned above but the processing speed and result is not ideal. To generate high accuracy result, it requires high robust algorithm which take a long time to process on slow processing power devices. On the other hand, fast processing method might result in low accuracy result.

Moreover, current Google Play Store only published application which only detect for certain product only. This is because object matching requires much complex algorithm, as object’s colour might change dramatically from pixel to pixel. Integrating the functionality into smartphone devices allow the users to perform object searching anywhere and anytime conveniently.

1.4 Project Objectives

The application developed from this project is to allow users to search for an object by using digital image processing technique. The final application must have following feature:

<u>To search for a targeted object within an image based on scale and rotational invariance</u>

Differences in object scale and orientation will greatly affect the searching result. This project aim to search for the targeted object based on two different images. In most of the scenario the input image and the data set having different scale and orientation, human eyes can tells both objects are same but not computer vision. However, the object must be the same surface. It would not take in account for the object side view or back view.

<u>To search for a targeted object within an image which able to handle partial occluded scenario</u>

If the given object is partially occluded, then the machine will lost that particular region of data. In this project, it aims to detect the targeted object given some of the information is not available to the machine. However, if the object is highly occluded it is likely fail to mark that area as object position.

<u>To implement an Android based image processing application which uses OpenCV library</u>

It is well known that Android is mostly run in Java environment. However, the complete product of this application should be able to run C++ code in Android environment that supporting Open Source Computer Vision Library (OpenCV) library.

<u>To optimize an instance of object matching process under 10s by using Android device with 2.3GHz</u>

Lastly, this project should be able to perform object matching based on two input images less than 10 seconds by using Android device with 2.3GHz. The timer is started after both images have been confirmed and search process start. The final deliverable should be able to generate fast and accurate result.

1.5 Project Scope

Object detection and matching application that runs on Android devices are to ease the users on object searching by using handheld device. The system will be implemented by utilizing computer vision library which is OpenCV. Meanwhile, the user interface will be implemented by the existing library provided by Android itself. In order to perform object detection and matching successfully, the application must able to read at least two images.

- **<u>Use the template image to detect targeted object</u>**

The image can be either snap with the device's camera or choose from image gallery that stored the image. Next, image is manually cropped by user before register as template image. This is to significantly reduce unwanted area and extract the object features. It is designed in a way that the application will only detect the copped image as one object on every single execution.

- **<u>Perform object matching by comparing two images</u>**

The second image will be snap by the user. This image is used to search for the object location. Different approaches will be tested on the object matching and the ideal solution which is fast and with acceptable accuracy will be implemented on the final deliverable product. It is consider the targeted object might be scale into different size, different orientation, or both.

- **Object matching with template matching technique**

  As the user crop out the image in the first image, it will store as the template image (T). The template image (T) will be compared with search space image (I). However, template matching are not invariant to scale and orientation changes. Thus, it will scale and rotate the template explicitly.

- **Object matching with colour histogram matching technique**

  Pixel-by-pixel matching in template matching is very inefficient method. Thus, the template image (T), it will extract the colour histogram and compare with search space image (I) colour histogram. The HSV colour space is the chosen colour mode. The hue component of it is necessary for characterizing the object intent to search for.

- **Direct pixels comparison**

  The filtered result in search space image (I) is directly compared with the pixel's value in template image (T). The direct comparison is to significantly reduce false alarm result that might having similar colour distribution but different outlook with template image (T).

In a nutshell, the application will mark on the founded similar object with a coloured box. Otherwise, prompt alert box if object is not found.

1.6 Impact, significance and contribution

This project is targeted on end users from all range, where it can range from youngster to golden citizen. It mainly helps the user to overcome the problem of having object searching in complex environment with the aid from Android devices instead of using PC platform. The proposed method enables object searching on handheld devices become a reality. This application benefits the users by giving the object location on the image itself.

The main contribution of this project is to enhance the efficiency of image-based target object searching on mobile devices. The efficiency mentioned above is applicable to both speed and accuracy. This application will be new to the market as current apps do not provide such functionality.

The result of this project can turn into beneficial application. Although finding a book in the library sound relatively simple, it only needs to get the call number from library database which will lead to the correct bookshelf and narrow down searching range. However for various reasons, some patron might spend a long time to find their desire book. It might due to unknown book conditions or binding. With this application, user may snap the image found on the library database, when reached to the bookshelf user may snap another photo and perform the searching.

1.7 Background Information

1.7.1 What are Digital Image Processing and Computer Vision?

A digital image may be defined as a discrete representation of data possessing both spatial (layout) and intensity (colour) information. A digital image is composed of a finite number of elements, each having a particular location and value. Generally, the term *pixel is* used to denote the elements of a digital image.

The field of digital image processing refers to processing digital images by means of a digital computer. It can process an image, include storing, transmitting and represent it for autonomous machine perception. Other than that, it may be does some function such as smoothing, sharpening, contrasting, or stretching on the digital image. The purpose is to extract image's information by the machine or improve the image quality so human can interpret it better.

The ultimate goal of computer vision is to use computers to emulate human vision, includes learning and able to make judgment and action based on input visual. However, unlike human eyes, computer does not born with a brain. From the science aspect, computer vision is related to artificial systems that extract information from images. Computer vision is the superclass of image processing and uses image processing algorithm to extract images information such as what objects presented in the image.

1.7.2 What is OpenCV?

OpenCV is an open source library for developing computer vision applications and machine learning software. Everyone allows using, distributing, and adapting it either for academic or commercializing application under legal licensing. It has C, C++, Python and Java interfaces that support for numerous OS which are Windows, Linux, Mac OS, and Android. OpenCV was designed for computational efficiency and programmed natively in C++, the library can take advantages of parallel processing.

The library stored more than 2500 optimized algorithms, inclusive of a set of computer vision and machine learning algorithms. These algorithms can be used to extract information from images, such as detect and recognize faces, identify objects, object recognition, real time tracking moving objects and etc. The library is used extensively in organization, research groups, and governmental department.

1.7.3 What is mobile application and how does it relate to Android?

Mobile applications or in short mobile apps are a software application programmed to run on smartphones, tablet computer and other handheld devices. Originally, mobile apps were offered for general productivity and information retrieval, such as email and contacts management. Due to market demand and expansion on the availability of developer tools, mobile application not just meant for general usage. Developers can use available tools to build games, turn it into Global Positioning System and etc. Normally those applications are available through application distribution platforms, such as Google Play and operated by the owner of the mobile OS. Mobile apps can be downloaded from Internet for free or with certain charges (OnGuardOnline, 2011).

One of the famous mobile OS in today market is Android. Android is a Linux based OS that designed for touchscreen mobile devices. It is an open source platform. Any developers allow modifying the source code and redesign the “look”. Holes and bugs in the OS can be quickly found and patched. In 2007, Android was introduced by using only Java to build the application. Any operating system installed with java and Android SDK environment able to build mobile apps that run on Android. In 2008, the first Android based smartphone with version 1.0 was officially launched to the market. Currently the latest Android version is 4.2 (Android, 2013).

1.7.4 Why using OpenCV in Android?

While most of the Android applications are written in Java, in 2009 Android Native Development Kit (NDK) was announced. Android NDK allows developers to implement part of the apps using native-code languages such as C/C++. OpenCV is a library of programming functions mainly focused on computer vision. The launched of Android NDK enables the developers to use optimized OpenCV code in Android. This is because OpenCV is written natively with C++ (OpenCV, 2013).

There are other library can be used for image processing in Android, OpenCV is chosen due to the efficiency and rich documentation. One of the image processing libraries for Android is JavaCV, which is a wrapper for OpenCV. JavaCV contains almost all the function that OpenCV has, but it does not have any rich documentation to find the functions for your project.

Jon's Java Image Library (JJIL) is a Java image processing library. The reason why it is not chosen in this project is because it hasn't been updated since 2008 and only works on very old version of Android. Other than that, the library only provided limited function. Those sophisticated function such as object matching is not available (JJIL, 2008).

Chapter 2 Literature Review

In this session, firstly it will discuss about the development kits available for Android. In addition, studies on colour model and object matching technique that related to image processing field before designing the methodology and implementation commence.

2.1 Mobile Application Development in Android

Mobile application development is the process of developing application for resources constrained devices, such as smartphones. Each smartphone is installed with a mobile OS. Different mobile OS has their respective development environment. The development environment is for the developers to write, test and implement applications into targeted platform environment.

During the planning phase of developing an application, the developers must consider on several aspects. It should account on the length array of screen sizes, hardware specifications and the minimum firmware version in order to run the application. The developed application in Android environment can be compress into an .apk file for user to install it. These applications can be downloaded from various mobile software distribution platforms. Google Play Store is the one of the popular market for Android users to download applications.

There are various programming model for Android. In 2007, Android was launched by only using Java as the application programming language. The Android Software Development Kit (SDK) supported in Java contains tool for debugging, testing, and utilities that are required to develop an application. In 2009, Android NDK was introduced. It helps the developers to implements part of the app using native-code languages such as C++. In later year, it introduced Renderscript for Android 3.0 and above. These three models can be used to develop applications of similar functionality.

The comparison was performed by running applications that represent these three models. The application is called Balls. It receives input from sensors, conducts physics computations and draws resulted graphs on the screen. It simulates the movement of hundreds of balls according to the gravity and bounce among them. On screen touching will force the ball move in the ordered direction (Sams, 2011).

Same as an Android game, it requires physical and graphical computation. Figure 2.1 shows the comparison result among the three programming model. The number of bodies represents the number of balls, while FPS is frame per second. The SDK drops its performance drastically with more bodies, while the NDK and Renderscript have almost linear performance degrations with more bodies.

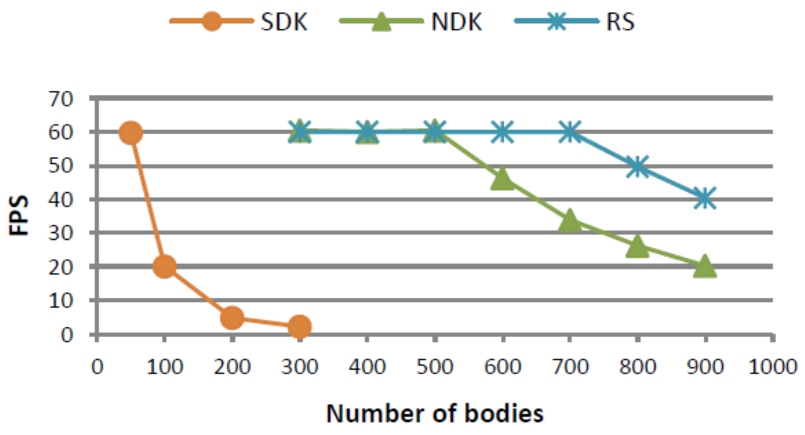


Figure 2.1 Comparison of three programming model in Android 3.2 Tablet *(*Qian *et al.*, 2012)

<u>Strengths</u>

There are pros and cons among these three models. The encapsulation in Android SDK provides high level of security at language level. It is easy to develop application by using the Android SDK. The portability in Android SDK allows developers continue their work in any platform, as long the machine pre-installed with the Java runtime. The mojor difference between NDK and Renderscript is due to the threading model. Renderscirpt is implicitly multi-threaded.

<u>Weakness</u>

Android SDK generates low performance compared to NDK and Renderscript. Although NDK outperform SDK, it has issue when move to different microarchitecture. C++ does not have the strong security feature. Overall, Renderscript has the best performance, but it only support for Android 3.0 or above. Also, the library extensibility in Renderscript is limited.

There is a restriction in Android based mobile application development. If the developers wanted to use the Android NDK or Renderscript model, it must run together with Android SDK. Thus, the ideal development kit for Android application is SDK. Other than that, using NDK or Renderscript will increase the application complexity.

<u>Method to resolve the limitation</u>

Thus, that paper proposed a unified programming model that alleviates the issues in current models by combining of the existing models. It introduces vector type in Java layer, and defines runtime to eliminate the data copying across JNI (Java native interface) as RenderScript does. The new model compiles NDK C++ code into intermediate representation as RenderScript does. In this way, developers do not need to write separate code with RenderScript API but keep the benefits of RenderScript (Qian *et al*., 2012).

2.2 Colour models in digital images

Colour is a powerful descriptor that often simplifies object detection and extraction from a scenario. Due to human perception on colours, colours are seen as variable combinations of the *primary colours*, which are red (R), green (G), blue (B). These primary colours can be mixed to produces *secondary colours*, includes magenta (red mix with blue), cyan (green mix with blue), and yellow (green plus red). Mixing the primary colours with secondary colours in correct intensities results white colour. Figure 2.2 illustrated how primary colours and its sum up resulting in secondary colours and white. Moreover absent of the light will resulting in black.

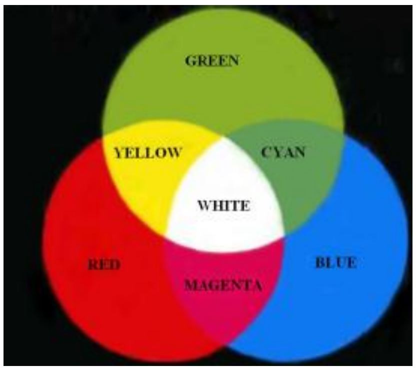


Figure 2.2 Primary and secondary colours of light (Gonzalez & Woods, 2010)

There are 3 characteristics used to differentiate one colour from another. Firstly will be brightness, it means the amount of intensity if a colour. Secondly it the hue level, it is an attribute that describe the pure colour for example pure red, yellow, blue and etc. Lastly is saturation, it is the value of the degree of the pure colour is diluted by the white light. For example, when added white into pure colour read, it will become pink colour. The value is basically the measurement of brightness.

The purpose of a colour model is to facilitate the specification of colours in standardize way. Each colour is specified to their respective coordinate and subspace within that system and represented by a single point. There are various colour models available in digital image processing field, each having different design and serve for different purpose (Gonzalez & Woods, 2010).

2.2.1 RGB Colour Model

In RGB model, each colour represents in its primary spectral components which are red, green, and blue. It is based on the Cartesian coordinate system. Different colours are point on or inside the cube. Figure 2.3 illustrates the RGB colour space of interest in cube dimension.

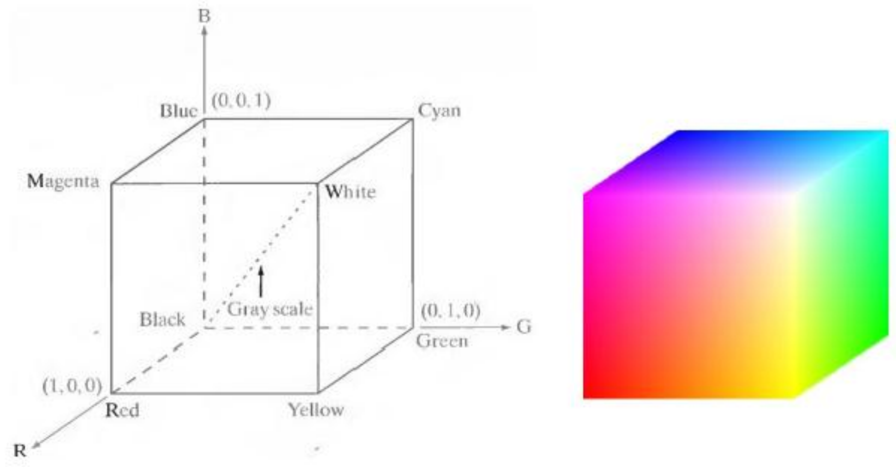


Figure 2.3 RGB Colour Model (Gonzalez & Woods, 2010)

<u>Strength</u>

Some of the application prefers to use RGB colour model instead of others. It uses for sensing, representing, and displaying images in electronic systems, such as televisions and computer. This is because RGB model is constructed based in human perception of colours as mentioned above. It directly reflects the physical properties of "true colour" in each pixel displays. This colour model is supported by most graphics tool and visual display devices.

Weaknesses

However, RGB colour model is not a friendly system to describe the changes hue and saturation. In this model, a colour is described by specifying the intensity levels of red, green, and blue. Unfortunately, human does not refer to the colour of an object by stating the percentage of each of the primary colours. Furthermore, no one look at colour images as being composed by the 3 primary images that combines and form a single image.

2.2.2 HSV Colour Model

HSV colour model are more natural and intuitive to the way of human perceive and interpret colour. In, HSV, H stand for Hue, S stand for saturation and V stand for value. Hue is an attribute that describe the pureness of colour for example pure red. Saturation is the value of the degree of the pure colour is diluted by the white light. The value is the measurement of brightness. Figure 2.4 illustrates the colour space of HSV colour model.

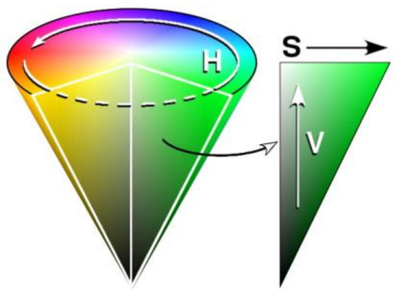


Figure 2.4 HSV Colour Model (Gonzalez & Woods, 2010)

Strength

With the cylindrical-coordinate representations, it will be much easier to describe colour with. The relationship between tones around the colour circle is easily identified. For example, changes from pure red to magenta can be done by shifting the hue slider. Meanwhile, the developers can easily define the colour to be darker or lighter by simply shift the saturation slider until hits the sweet spot. It is an ideal tool for developing image processing algorithms based on colour descriptions that are natural and intuitive to humans.

Weaknesses

The HSV colour model ignores the complexity of colour appearance. Consequently, the trade-off is the computation speed. A more sophisticated model would have been computationally expensive. Due to the cylindrical-coordinate representations, in order to specify a colour precisely requires stating the HSV values and the characteristics of RGB space used.

2.2.3 Discussion

Although are more than 2 colour models available in image processing field, but the models discussed above are leading models in image processing and widely used in most of the applications. However, there is no best colour model available in image processing field. Depend on the project requirements, the choice of colour model are different.

2.3 Review on techniques for object searching

Object detection is a task of searching a given object in an image or video. In computer vision, the task for searching an object that is varies in viewpoints, sizes, or orientation is a challenge. For decade, numerous techniques had been proposed for object detection.

2.3.1 Image smoothing

Even though this method does not provide the functionality of object detection but it is critical on the process of searching an object. Blurring (smoothing) is a straightforward and frequently used image processing method. This is to create an approximating function that attempts to capture important patterns in the data, while filtering out noise or other fine-scale structures.

To perform blurring operation, a filter will be applied to the image. It resembling viewing of an image through a translucent screen, distinctly different from the blurring effect produced by an out-of-focus lens or shadow of an object under illumination (Figure 2.5).

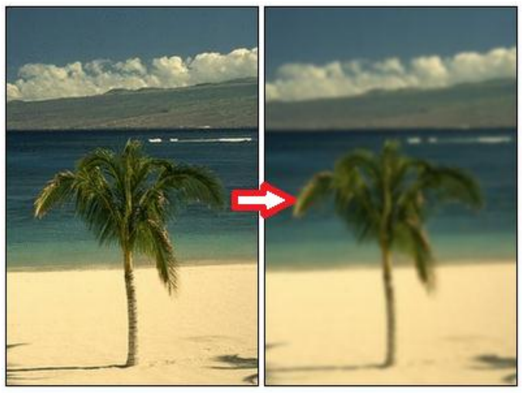


Figure 2.5 Effect of Gaussian Blur (Radhakrishna, 2008)

There are various methods available in image blurring. Gaussian Blur blurs an image using Gaussian filter. Filtering a static image with a 2D Gaussian can be implemented by using two 1D Gaussian, where less calculation is require and hence, faster.

Strength

Refer to Eq. (2.1), Gaussian blur technique computes weighted averages in which nearby pixels are assigned a larger weight than ones further away. Gaussian filter is separable filters that filter can the done on the image horizontally followed by vertically or the opposite way.

$$G(x) = Ae^{-x^2/2\sigma^2} \tag{2.1}$$

The normalizing coefficient A is chosen such that the different weights sum to one. The σ (sigma) value which controls the width of the resulting Gaussian function. The greater value, the flatter the function will be. With OpenCV, users allow setting the σ value and filter size that best suit own preference filter effect.

Weakness

Although it is a useful filter but is not the fastest filter. Using 2D Gaussian Blur in single pass, added the feature that masked-out pixels that exceed the threshold setting in the input image are excluded from blurring operation. This operation cannot be incorporated into two passes blurring process (Sachs, 2013).

2.3.2 Template matching

Template matching is an approach that finding similar areas of an image that match to the template image. In this technique, it only requires 2 images, which will serve as the template image (patch image) and source image (to find a match to template image). The ultimate goal is to search for highest matching area.

To identify the matching, the source image is compare by template image by sliding it. It means moving the template image one pixel at a time, from left to right, up to down.

At each coordinate, a metric is calculated to represent how similar the matching is at that coordinate (Figure 2.6).

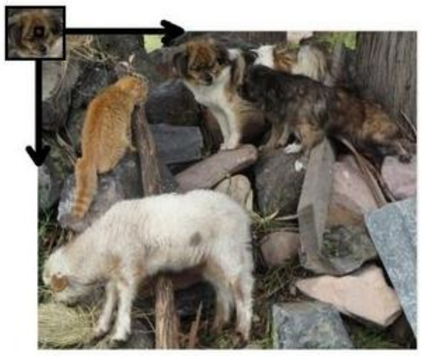


Figure 2.6 Template matching (OpenCV, 2013)

Template matching provided by OpenCV is not invariant to scale and orientation changes. Given the object in source image was rotated by 90 degrees, this technique would never find it. By writing brute force algorithms would that generate all possible rotation and scale that help us to find the matching (OpenCV, 2013).

However, moving pixel-by-pixel at a time would significant take more time compare to moving the image in a larger pixel distance. Moreover, if the developer rotate the image degree-by-degree and scale the size pixel-by-pixel, it will guarantee high accuracy result but this technique would be extremely computational expensive and result in long processing time. Thus, under normal circumstances a template is normally rotate by 10 °to 20 °, depend on the user setting.

2.3.3 Histogram matching

Images are composed of pixels of different values which include brightness. The distribution of pixels values across the image constitutes an important characteristic of the image. Histograms can be used to characterize the image's content and detecting object. Histogram is a table show the collected counts of each pixel value in an image. Obviously, summing up the total entries of a histogram will obtain the total number of pixels in an image. Histogram can keep count on not only gradients but others image features such as colour intensities.

By looking at an image area showing a particular object, then the histogram of this area can be assume as the probability that a given pixel belongs to this object. One of the features in histogram matching is able to find an object. Suppose the users do not know the location of the object to be found, the histogram which count the colour values keep track the number of pair of certain pixels that occur at certain areas. If starting from the initial location and iteratively move around, at last it should be possible to find the exact object location.

In the scenario of using colour histogram to search for object, it is highly recommended to use HSV colour model. By using the hue component, H, it allows us to characterizing the object we searching for. Other than that, it can remove the V, value of intensity. This allows us to handle with large variations of illumination. For example, everyone having different skin colour but the pure colour is remaining the same.

Unfortunately, this method is not suitable for monochrome images. Many other pixels in the image might share the same intensity. The preceding result could be disappointing. Thus, it is highly encourage using colour information for better performance (Laganière, 2011).

2.4 Study on existing solution implemented on smartphone platform

According to a research done by (Ismail *et al.*, 2012), comparison between Features from Accelerated Segment Test (FAST), Speeded Up Robust Feature (SURF), and Scale Invariant Feature Transform (SIFT) methods was studied to analyze their performance. Those algorithms were tested on the Samsung GalaxyS Android smartphone. The analysis result was concluded respect to the algorithm efficiency, quality, and robustness of the object detection.

One of the measurements is record number of key point detected in each object. The higher numbers of key point recorded require more points need to be matched compare to those having less key point. To measure the robustness of the algorithms test, instead of having the key point on every frame as the reference point, it used in normal location and illumination. The key point allocation allows them to observe repeatability difference between the key point in normal or different illumination, and orientation (Figure 2.7).

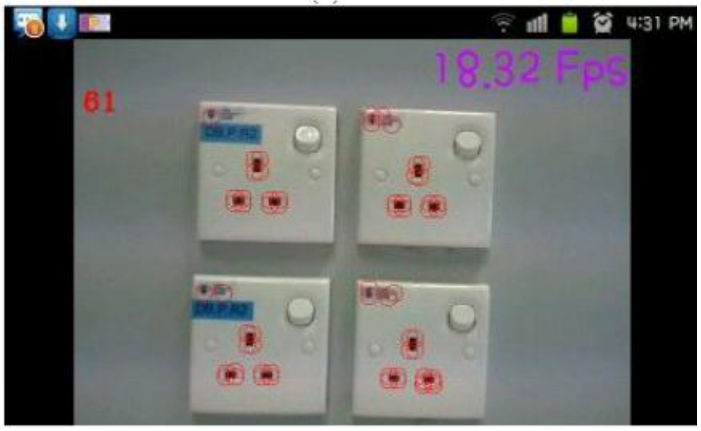


Figure 2.7 Real-time Object Detection in recording number of key points *(*Ismail *et al.*, 2012)

Algorithm like SIFT is able to detect and matches object where it may vary in size, rotation and even it is partially occluded. However, SIFT uses the descriptor of a high-dimensional vector, so the computation time for descriptor generator and feature matching are relatively high. SURF has similar performance to SIFT but run much faster than it. However, both having complex processing solution, thus it is not ideal to run in mobile devices.

Although FAST has the best performance, but the response toward change in orientation and illumination change is significantly low compared to others. The experiment was tested in PC environment. The size of image for each data set is set to 277X156. Figure 2.8 shows the sample data set where the top right is data set 1, next to it is data set 2, and so forth. As shown in Table 2.1, the time spent for SIFT and SURT are relatively high. The experiment was done with 8 differences data set.

Figure 2.8 Data Set (Jeong & Moon, 2011)

Table 2.1 Result of Performance Comparison on a Desktop PC (Jeong & Moon, 2011)

Time (ms), points	**Data set**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
SIFT	296.39 (245)	229.95 (194)	210.01 (170)	204.09 (160)	278.83 (233)	236.39 (186)	317.04 (276)	173.95 (111)
SURF	28.64 (181)	30.28 (202)	20.06 (133)	19.41 (110)	21.01 (118)	18.53 (116)	24.80 (171)	22.34 (132)
FAST	0.90 (230)	0.86 (246)	1.20 (599)	0.67 (112)	0.79 (128)	0.98 (334)	1.20 (517)	1.03 (483)

When using FAST corner detector, data for object recognition is created based on corner information. Under such circumstance, object recognition is more challenging because the extracted features vary in terms of the edges and occlusion. To extract features required for object recognition using FAST algorithm it must be combined with machine learning techniques for object recognition. This will create extra tasks to the process (Jeong & Moon, 2011).

SIFT and SURF are good methods which yield high quality result, however they are too computationally intensive to use on smartphone platforms. Thus, both methods are less suitable in resource limited devices. Meanwhile due to patent issues, both methods are categorized into non-free module and not included in the OpenCV for Android package. Although there is solution to build non-free module for the Android project but the final deliverable is incompatible with Android OpenCV Manager and it cannot be distributed on Google Play Market (OpenCV, 2013).

2.5 Review on similar application

One of the Android applications was reviewed in this study, named as "*Google Goggles*". This application can be downloaded from the Google Play Store market. This application was developed with some of image processing technique. Although the application is not the same product as what will be developed in this project but the techniques of the object recognition and searching can be used for references.

"*Google Goggles*" is an application that supports image-based searching. It let the user capture or import a photo from gallery. It will identify what the object is using image processing technique. Currently it support objects such as books, landmarks, logos, contact info, artwork, businesses, products, barcodes, and plain text only (Raphael, 2009).

Edges in an image are usually robust to changes in lighting. One of the edge detection technique applied is Canny Edge Detection, where it able to detect edges in a template and image. The edges are used to compare with the dataset. Another algorithm used for objet detection is SIFT. Firstly, it extracts the key points of objects from the set of reference images and stored in a database. An object is recognized from the input image. The feature of the object is individually compares to the database and finding matching candidate based on Euclidean distance of their feature vectors (Handa, 2011).

<u>Strengths</u>

When the user captures an image, the application will break it down into object-based signature. The analysis returns the result and goes through a set of rules created by developers. It compares those signatures against every object it can be found in its huge online database. It also tries to find text in the image using optical character recognition to obtain better idea of what the object might be. Within second, it generates the result ordered by rank (Schaap, 2011). Figure 2.9 shows an example of how "*Google Goggles*" is being used.

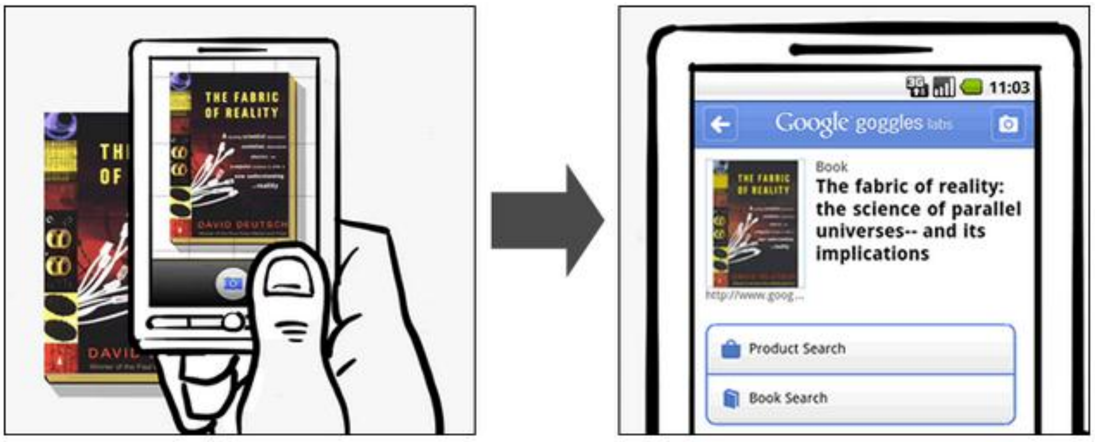


Figure 2.9 Sample objects searching on *Google Goggles* (Google, 2011)

Weaknesses

Google Goggles is still in its experimental phase and not all results are accurate. Meanwhile, if the object is partially occluded, it might fail to detect the object. In addition, object searching in Google Goggles is mainly depended on the dataset. For example, the database does not contain object features of pets. In this case, images of animal generate zero returned result by using Google Goggles.

In some cases, it might return the result before we snap a photo. This is because of the integrated GPS and compass functionality. Thus, this application does not fully rely on image processing functionality.

Method to resolve the limitation

In order to recognize more objects, from time to time the developers need to add the extracted image feature to the database. Currently, Google is working to allow the system able to recognize different plants and leaves. It aid people curiosity for those wish to avoid toxic plants, and botanists and environmentalists searching for rare plants. However, this application must have Internet connection to perform searching.

Chapter 3 Methodology

3.1 Methodology

An appropriate methodology ensures the project can be successfully carried out. Methodology provides general principles that guide the developer to complete the project. In this session, decision on which methodology should be applied in this project will be discussed.

Rapid application development (RAD) categories are a better approach compare to the other methodologies. Prototyping methodologies is one of the RAD, it will be the chosen methodology to develop the application. A prototype can be developed for the review session which can overcome the flaw in traditional methodologies. Analysis, design, and implementation phases can be done concurrently to build the first version. Users able to review the prototype before implement the final system into the environment. Comments and feedback provided by the users will take it as the guideline to make changes on the system (Figure 3.1).

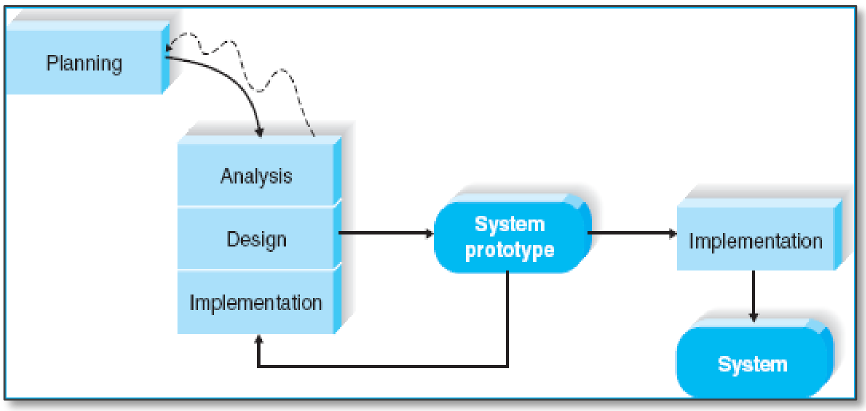


Figure 3.1 System Development Life Cycles (Hoffer *et al.*, 2011)

Prototyping development has the ability to re-analyse, re-design, and re-implementation on the previous prototype provides refinement to the system. The process continues in a cycle until the prototype provides enough functionality to be installed and be used in the environment. This is the main reasons why it has been chosen as the main methodology for this project (Hoffer *et al.*, 2011).

This project is best suit to develop in prototyping methodology. Traditional methodologies will have critical issue on going backward after that particular phase completed. For instance, the waterfall methodology will proceed accordingly start from the planning phase to implementation phase.

In each phases, it must first present to stakeholders for approval. After approved, only then it can proceed to next phase. Although it is possible to do backward, but any changes of requirements after the analysis phase will affect the system development life cycle and it is extremely difficult. The key deliverables for each phase are very long, hence waterfall methodology is not practical to use in short time scheduled project. Thus, traditional methodologies will not be chosen.

3.2 Timeline

This project was begun in November 2012 and end in April 2014. Although the timeframe for the project is long but during Jan 2013 to May 2013 less focus is pay to the project development.

Phase 1 Planning

In November 2012, the first phase of the project will be started with requirement gathering and feasibility study of the project. It defines the problem statement, project scope and objectives. The process will be continued by plan the project milestone and respective timeframe for each milestone.

Phase 2 Analyses, Design, and Implementation

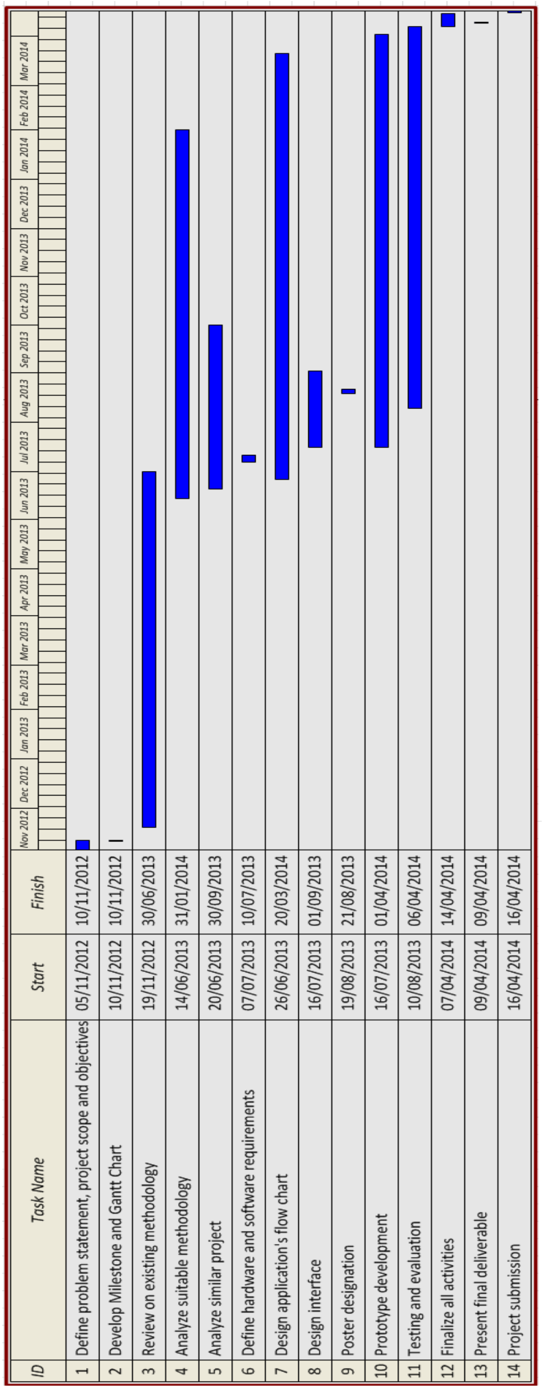
After the planning phase is completed, it proceeds with the analysis phase. It focus on determine the suitable methodology and technique that able to achieve the objectives. Similar project will be reviewed to grab some ideas on the technique used and interface design.

Next, it the design phase. Based on what had analysed previously, it design activity diagrams for the system flow. In addition, it designs on how the application interface should looks like.

During the initiation of implementation phase, it will develop the first prototype. Testing and evaluation will be done to determine whether the implemented technique suitable to use or not. As mentioned above, this project used prototyping methodology. Based on the result, it re-analysed, re-design and implement an improved prototype version.

Phase 3 Final implementation

The project will go through numerous re-analyse of methodology used, re-design the system flow until the developed prototype had achieved all the objectives. Lastly, numerous test samples will be carried out to test the efficiency and usability of the application. In a nutshell, Figure 3.2 shows the planned timeline for each milestone.



ID	Task Name	Start	Finish
1	Define problem statement, project scope and objectives	05/11/2012	10/11/2012
2	Develop Milestone and Gantt Chart	10/11/2012	10/11/2012
3	Review on existing methodology	19/11/2012	30/06/2013
4	Analyze suitable methodology	14/06/2013	31/01/2014
5	Analyze similar project	20/06/2013	30/09/2013
6	Define hardware and software requirements	07/07/2013	10/07/2013
7	Design application's flow chart	26/06/2013	20/03/2014
8	Design interface	16/07/2013	01/09/2013
9	Poster designation	19/08/2013	21/08/2013
10	Prototype development	16/07/2013	01/04/2014
11	Testing and evaluation	10/08/2013	06/04/2014
12	Finalize all activities	07/04/2014	14/04/2014
13	Present final deliverable	09/04/2014	09/04/2014
14	Project submission	16/04/2014	16/04/2014

Figure 3.2 Project Gantt chart

Chapter 4 Implementation

4.1 Application Overview

Figure 4.1 shows the steps need to be taken in order to search for the targeted object.

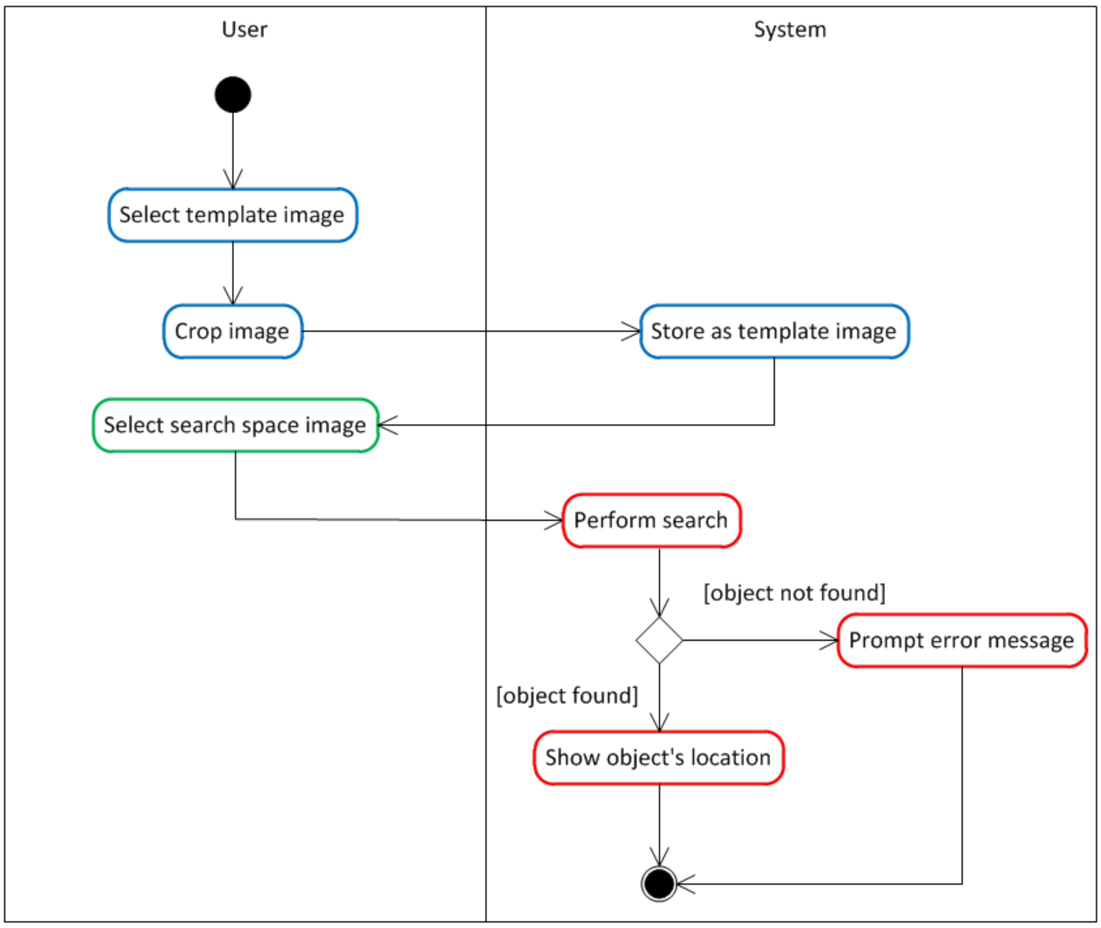


Figure 4.1 Application Overall Process Flow

4.2 Step 1 – Select template image

This process is to read the source object which is to be searched in the searching process. Users can select the template image via snapping with the device’s camera or choose from phone memory. Next, user has to manually crop the selected image. This is to significantly reduce unwanted area. The cropped area is always in square dimension. To ensure best performance achieved, the cropped area is recommended within the object. The cropped area will be stored as the template image. Figure 4.2 shows an example of user cropping the image and stores it as template image.

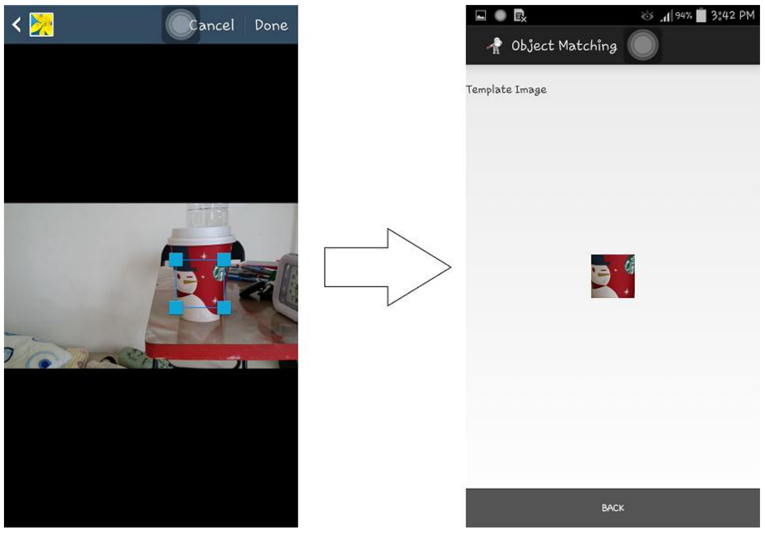


Figure 4.2 User cropping and storing the template image

4.3 Step 2 – Select search space image

This process is to read the image where the searching process takes place to locate the object. Users can select the search space image via snapping with the device’s camera or choose from phone memory.

4.4 Step 3 – Perform search

This is the process where the system will compare the template image and search space image to find the object location. The backend process is illustrates in figure 4.3. Given the object is found, it will draw coloured square boxes on the search space image, otherwise prompt error message.

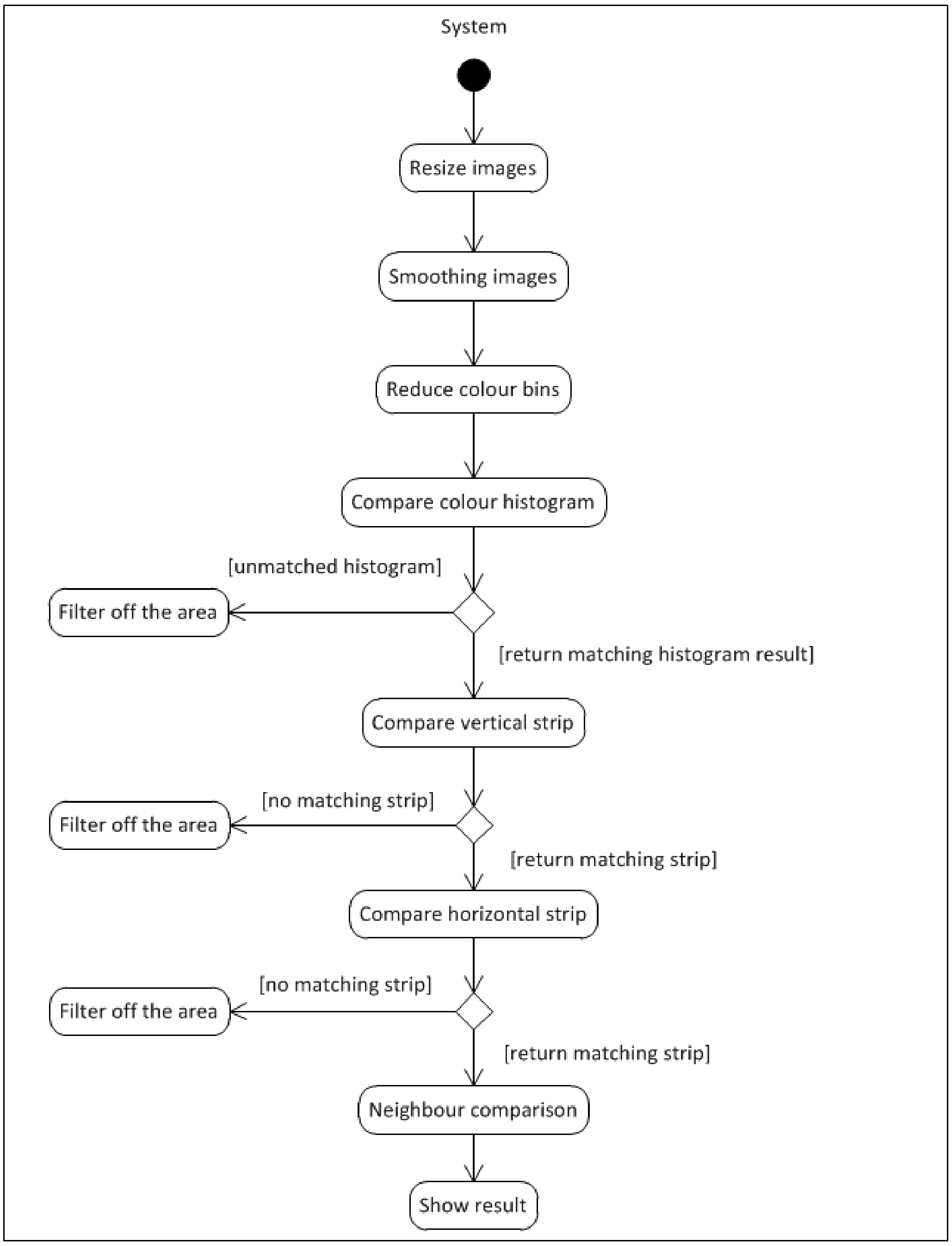


Figure 4.3 Object searching backend process

4.4.1 Step 3.1 – Resize images

First of all, both template image and search space image are read in RGB format. Both images are resized for best searching performance by using *resize ()* function provided by OpenCV library. Template image is resized into 160×160 dimensions.

On the other hand, if the search space width or height is greater than 1000 pixels, then it will be resized with following method:

```
Input   : Image $_s$ , where $_s$ representing search space image
Output : Image $_{out}$

For every dimensions [x, y] ≥ 1000 in Image $_s$
                    Scale factor  = 0.8
                    Image $_{out}$ = ((Scale factor × x), (Scale factor × y))
```

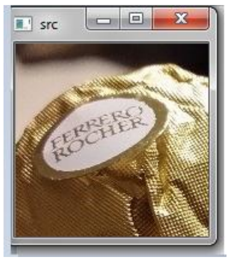
4.4.2 Step 3.2 – Smoothing images

Next, both resized images will be blurred with *medianBlur ()* function provided by OpenCV library. The median filter run through each element of the image and replaces each pixel with the median of its neighbouring pixels. Each channels of the input image is processed independently, by setting the difference aperture value, the template image show slightly differences.

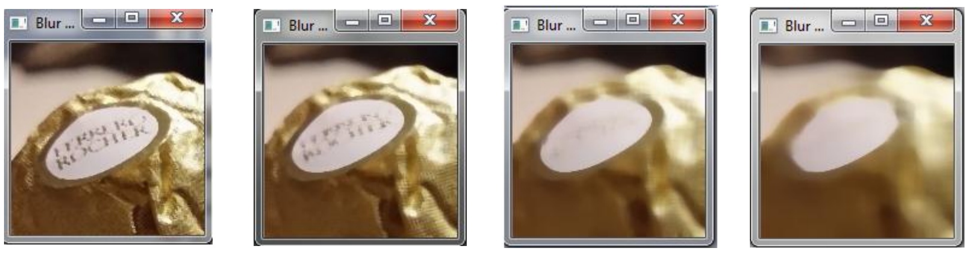
Justification

Experiment is conducted to test which aperture value best suit this project.

When aperture value is to 5, most of the details still well preserved in the resulted image. However corners are not well-preserved by median filter and tend to be blurred to a degree proportional to the size of the median filter. When the aperture value is set to 3, most of the details are remain; this might impact on the colour comparison when too much of details are taking into account. On the other hand, when the aperture value is set to 11 and 21, most of the significant details are gone and greatly replaced by its neighbouring pixels (Figure 4.4).



Original Template



(a) k = 3 (b) k = 5 (c) k = 11 (d) k = 21

Figure 4.4 Resulted image by applying Median Blur, where k = aperture value

Conclusion: The selected aperture linear size is 5.

4.4.3 Step 3.3 – Reduce colour bins

By default the images are read in RGB and having a total number of 256 colour bins. Both images are converted into HSV colour model by using *cvtColor ()* function provided by the OpenCV library. In order to reduce the complexity of analysis, reduce the total number of colours will generate better performance. The total colour bins is reduce to 64.

The colour reduction function is compute as follow:

```
Input   : Image $_s$ , where $_s$ representing search space image
Output : Image $_{out}$

For every pixels [x, y] in Image $_s$
        IF Image $_s$ [x, y] > 1
            Image $_{out}$ [x, y] = Image $_s$ [x, y] / 4
        ELSE Do nothing
```

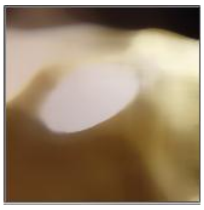
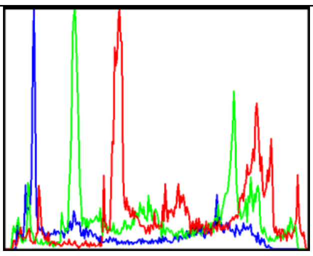
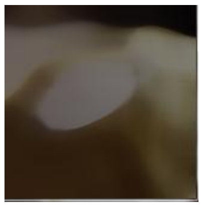
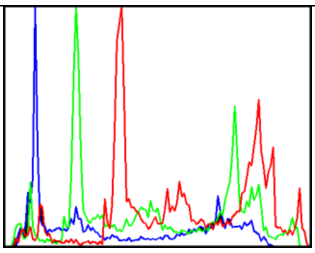
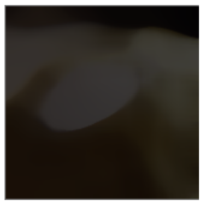
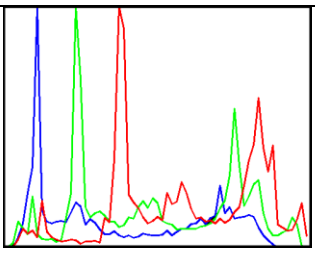
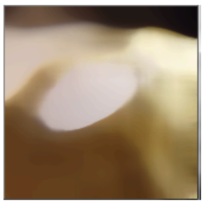
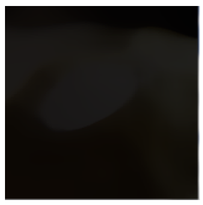
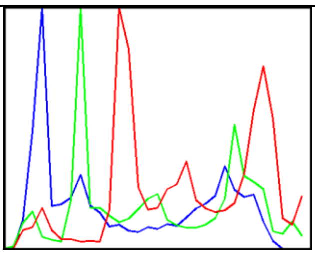
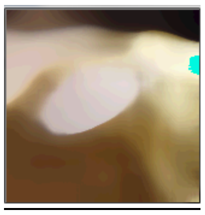
Justification

Experiment is conducted to test which total number of colour bins best suit this project.

Based on the resulted histogram, although the reduction is the colour details can be clearly observed but the overall histogram pattern is preserved. When the total colour bins are reduce to 128 and 64, there is no significant differences when revert back to the original total bins number. However when the total colour bins is reduce to 32, there are noises found in the reverted image (Table 4.1).

Conclusion: A total number colour bin of 64 is being selected.

Table 4.1 Colour bins reduction comparison result

Total number of colour bins	Resulted Image	Resulted Histogram	Revert to 256 colour bins
256			
128			
64			
32			

4.4.4 Step 3.4 – Compare colour histogram

Both reduced colour bins images are compared with its colour histogram. The comparison is performed for 7 times by applying different scale factors on search space image. Again, *resize ()* function provided by OpenCV library is used. The applied scale factors are 1.3, 1.0, 0.75, 0.45, 0.25, 0.15, and 0.05 respectively. This is to ensure the object can be detected regardless of it scale changes.

The template image is resized into the dimensions of 40×40 and a circular mask is set with a diameter of 40. The circular region will be region of interest (ROI) to be compared with the colour histogram of search space image (Figure 4.5). Thus, the object's orientation would not affect the output of the histogram comparison.

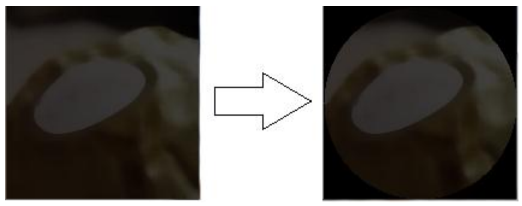


Figure 4.5 Template image with circular mask

The template image will slide through the search space image at 10 pixels at a time, from left to right, top to bottom. At each position, the colour histogram is compared to determine it similarity. The histograms are compared with following method.

```
Input   :
    a.  Histogram t , where t representing template histogram
    b.  Histogram s , where s representing search space histogram
Output : Result $_r$

From Bins, B $\in$ [3 to 54] in Histogram $_t$ and Histogram $_s$
                IF Bins difference in Histogram $_t$ and Histogram $_s$ Less than 90%
                        THEN increase the counter value
IF the Total Counter is more than 50%
        THEN mark as possible object area
```

For each possible area ∈ {P_1, P_2, P_3 … P_n}, it is demand for further analysis (discussed in Section 4.4.5, page 40). If the comparison found zero matching histogram among these two images, then the searching process will be terminated and prompt error message. Although OpenCV library itself provided *compareHist()* function for histogram comparison but it is not used in this project

Provided OpenCV method is not used.

The *compareHist()* functions provided in OpenCV are computation expensive.

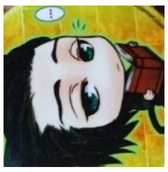
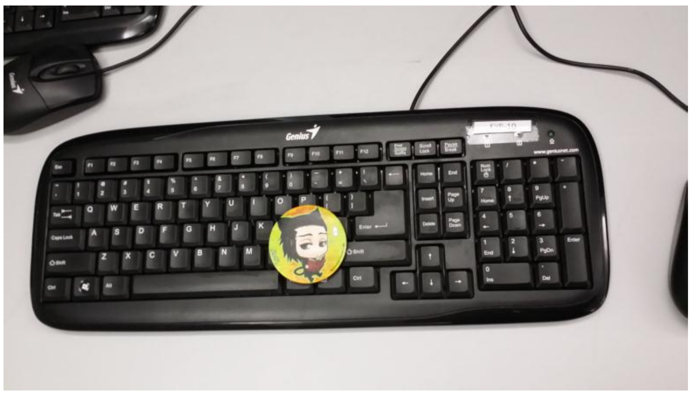
Following listed the 4 methods available in OpenCV *compareHist ()* function:

1. Intersection
2. Chi-square

The returned result with method (1) Intersection and (2) Chi-square are from 0 to infinite. As a result, these two methods is not a suitable as it is difficult to define the similarity percentage.

3. Bhattacharyya
4. Correlation

Method (3) Bhattacharyya and (4) Correlation is compared with the following result:

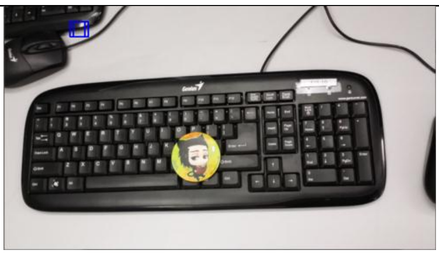



Template image Search Space image

Figure 4.6 Sample data for histogram comparison

Returned Result for matching probability of 60% or above.

Table 4.2 Returned result from histogram comparison

Method	Result	Failure Reason
Bhattacharyya		Wrong position detected
Correlation		No result returned

Both returned false result, as a result object is failed to be detected in the next filtering stage.

Conclusion: Non-of the OpenCV *compareHist()* function is chosen.

4.4.5 Step 3.5 – Compare vertical strips

In reference to Step 3.4, the system marked a set of possible area = {P_1, P_2, P_3 … P_n}. In order to find the exact object match among Pn, each pixel in the middle vertical strip of the template image is compared with middle column of search space image and its neighbouring columns where the distance is within the range of -4 to +4 (Figure 4.7).

In each searching area, the template will be rotated by 10 ° by using the *rotate()* function provided in OpenCV library and compared for a complete template rotation. In case 3 out of 5 columns hit the matching percentage which is more than 34% then it will marked as potential area and check for the next area. Otherwise, area failed to meet the criteria will be filtered off.

If the comparison returned zero match case, then the searching process will be terminated and prompt error message.

1	254	62	12	84	62	61	85	64
23	254	62	12	84	62	48	8	64
16	254	12	65	84	62	41	5	64
15	84	62	25	24	63	61	185	64
4	24	72	12	74	62	51	85	64
1	54	62	12	86	65	61	185	64
233	62	195	185	49	60	100	55	25
65	103	230	222	135	54	50	65	16
9	215	52	164	200	101	132	32	85

Middle column

1	254	62	12	84	62	61	85	64
23	254	62	12	84	62	48	8	64
16	254	12	65	84	62	41	5	64
15	84	62	25	24	63	61	185	64
4	24	72	12	74	62	51	85	64
1	54	62	12	86	65	61	185	64
233	62	195	185	49	60	100	55	25
65	103	230	222	135	54	50	65	16
9	215	52	164	200	101	132	32	85

Range of columns to be compared

Figure 4.7 Columns being compared

The column comparison function is compute as follow:

```
Input:
    c.  S ∈ {$P_1$, $P_2$, $P_3$ … $P_n$}, where $P_n$ are returned from Step 3.4
    d.  Template image (t)
    e.  Search space image (s)
Output:     C ∈ {$C_1$, $C_2$, $C_3$ … $C_n$}

For every S = {$P_1$, $P_2$, $P_3$ … $P_n$}
    1.  Get middle column of t
    2.  Get column of s by -4 with it middle column position
    3.  IF pixel difference between column of s and column of t is $\leq 3$ THEN increase
        matching percentage and counter
    4.  For selected 5 columns of s
            4.1 IF matching percentage is $\geq 34\%$ THEN go to (4.2)
                ELSE IF the template orientation is not 350° THEN rotate the template
                by 10° and repeat (1)
            4.2 IF counter $\geq 3$ THEN store into $C_n$
                ELSE repeat (3) by +2 to the column of s
    5.  IF the template orientation is not 350° THEN rotate the template by 10° and
        repeat (1)
```

4.4.6 Step 3.6 – Compare horizontal strips

In reference to Step 3.5, the system marked a set of possible area = {C1, C2, C3 … Cn}. In order to further reduce false positive result among Cn, each pixel in the middle horizontal strip of the template image is compared with middle row of search space image and its neighbouring columns where the distance is within the range of -4 to +4 (Figure 4.8).

Same technique in applied as mentioned in Section 4.4.5 (page 40). In each searching area, the template will be rotated by 10 ° by using the *rotate()* function provided in OpenCV library and compared for a complete template rotation. In case 3 out of 5 columns hit the matching percentage which is more than 34% then it will marked as potential area and check for the next area. Otherwise, area failed to meet the criteria will be filtered off. If the comparison returned zero match case, then the searching process will be terminated and prompt error message.

If the comparison returned zero match case, then the searching process will be terminated and prompt error message. Otherwise marked a set of possible area = {R_1, R_2, R_3 … R_n}.

1	254	62	12	84	62	61	85	64
23	254	62	12	84	62	48	8	64
16	254	12	65	84	62	41	5	64
15	84	62	25	24	63	61	185	64
4	**24**	**72**	**12**	**74**	**62**	**51**	**85**	**64**
1	54	62	12	86	65	61	185	64
233	62	195	185	49	60	100	55	25
65	103	230	222	135	54	50	65	16
9	215	52	164	200	101	132	32	85

Middle row

1	**254**	**62**	**12**	**84**	**62**	**61**	**85**	**64**
23	254	62	12	84	62	48	8	64
16	**254**	**12**	**65**	**84**	**62**	**41**	**5**	**64**
15	84	62	25	24	63	61	185	64
4	**24**	**72**	**12**	**74**	**62**	**51**	**85**	**64**
1	54	62	12	86	65	61	185	64
233	**62**	**195**	**185**	**49**	**60**	**100**	**55**	**25**
65	103	230	222	135	54	50	65	16
9	**215**	**52**	**164**	**200**	**101**	**132**	**32**	**85**

Range of rows to be compared

Figure 4.8 Rows being compared

The row comparison function is compute as follow:

```
Input:
    a. C ∈ {$C_1$, $C_2$, $C_3$ … $C_n$}, where $C_n$ are result returned from Step 3.5
    b. Template image (t)
    c. Search space image (s)
Output:         R ∈ {$R_1$, $R_2$, $R_3$, … $R_n$}

For every C ∈ {$C_1$, $C_2$, $C_3$ … $C_n$}
    1. Get middle row of t
    2. Get row of s by -4 with it middle row position
    3. IF pixel difference between column of s and column of t is $\leq 3$ THEN increase
       matching percentage and counter
    4. For selected 5 columns of s
           4.1 IF matching percentage is $\geq 34\%$ THEN go to (4.2)
               ELSE IF the template orientation is not 350 °THEN rotate the template
               by 10 °and repeat (1) by +2 to the row of s
           4.2 IF counter $\geq 3$ THEN store into $R_n$
               ELSE repeat (3)
    5. IF the template orientation is not 350 °THEN rotate the template by 10 °and
       repeat (1)
```

4.4.7 Step 3.7 – Neighbour comparison

In reference to Step 3.6, the system marked a set of possible area which having similar pixels values horizontally and vertically = {R_1, R_2, R_3 … R_n}. To ensure the application marked the correct target object, the neighbour pixels (Figure 4.9) of the potential area are compared.

If the comparison failed to meet 50% of similar pixels distribution in template image, then it will filter off that area. Areas that successfully fulfil the requirement will be marked as object position and return the result.

1	254	62	12	84	62	61	85	64
23	254	62	12	84	62	48	8	64
16	254	12	65	84	62	41	5	64
15	84	62	25	24	63	61	185	64
4	24	72	12	74	62	51	85	64
1	54	62	12	86	65	61	185	64
233	62	195	185	49	60	100	55	25
65	103	230	222	135	54	50	65	16
9	215	52	164	200	101	132	32	85

Returned Middle point

1	254	62	12	84	62	61	85	64
23	254	62	12	84	62	48	8	64
16	254	12	65	84	62	41	5	64
15	84	62	25	24	63	61	185	64
4	24	72	12	74	62	51	85	64
1	54	62	12	86	65	61	185	64
233	62	195	185	49	60	100	55	25
65	103	230	222	135	54	50	65	16
9	215	52	164	200	101	132	32	85

Neighbours to be compared

Figure 4.9 Neighbours being compared

The neighbour comparison function is compute as follow:

```
Input:
    a.  R ∈ {$R_1$, $R_2$, $R_3$, … $R_n$}, where $R_n$ are result returned from Step 3.6
    b.  Template image (t)
    c.  Search space image (s)
Output:         N ∈ {$N_1$, $N_2$, $N_3$, … $N_n$}

For every R ∈ {$R_1$, $R_2$, $R_3$, … $R_n$}
    1.  Get returned middle row and column of $R_n$
    2.  Compare the neighbour pixels with template image
    3.  IF pixel difference between column of s and column of t is $\le 3$ THEN increase
        the matching percentage by 1%
    4.  IF matching percentage is $\ge 50\%$ THEN mark as object position, $N_n$
                ELSE filter off the area
    5.  Return the marked position, $N_n$
```

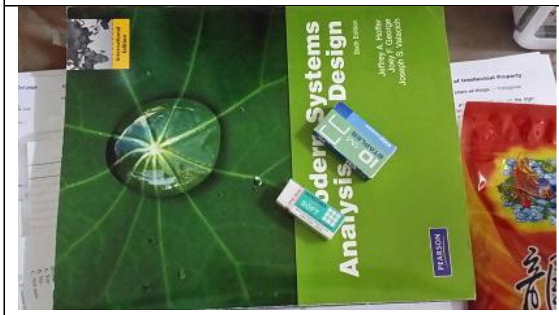
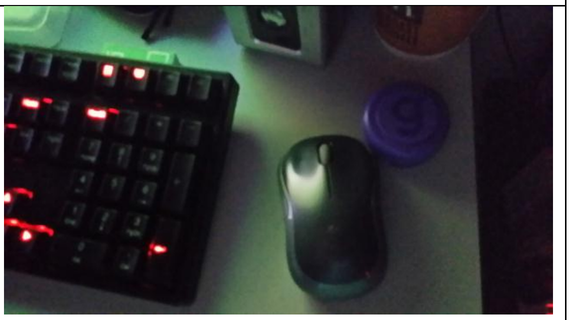
Chapter 5 Experiments and Result

This chapter demonstrates the experiments have been performed during the research of the project by using all the methods as described in Chapter 4.

5.1 Data Evaluation

This research project is targeted to search for object via images. Every image is captured under different environment. Some of the environment is challenging, such as the targeted object being partially occluded, night scene, instable search space due to hand shake when capture and etc. that might affect the accuracy. To access the performance and accuracy on searching result, 50 images are captures with Samsung Galaxy Note 3 with a dimension of 4128×2322 that containing targeted objects which will be the test data for this project. Table 5.1 shows 6 out of 50 from the sample data set.

Table 5.1 Sample Data

The data set are categorized with following characteristic:

1. Scaled or rotated or both target object
2. Target object occlusion condition
3. Lightning conditions
4. Type of background

5.2 Experiment stages

This project featured three major techniques which are comparing colour histogram, vertical strip, and horizontal strip. Each stage significantly reduces the search area. In order to test out the effectiveness, it is necessary to access them sequentially. Therefore there will be three experiment phases are conducted for analysis purpose.

5.2.1 Experiment stage 1 – Comparing colour histogram

This test is conducted by using a window size of 40×40 and sliding 10 pixels at a time. In each test, the returned result on marked area is closely analyzed. Set of different scenario have been selected to conduct the experiment.

The scenarios are:

1. Low light with same background colour with targeted object
2. Low light with different background colour with targeted object
3. Day light with same background colour with targeted object
4. Day light with different background colour with targeted object

Table 5.2 Summarized result from colour histogram comparison

Lightning conditions	**Type of background**	**Total Tested Image**	**True positive detection**	**False positive detection**	**True negative detection**	**False negative detection**
Low light	Same colour with targeted object	2	50%	100%	100%	50%
Low light	Different colour with targeted object	2	50%	100%	100%	50%
Day light	Same colour with targeted object	12	100%	75%	0%	0%
Day light	Different colour with targeted object	34	91.18%	82.35%	29.4%	8.82%
Summary		**50**	**90%**	**76%**	**28%**	**10%**

Table 5.3 Sample marked area as potential object position - 1 (Colour histogram comparison)

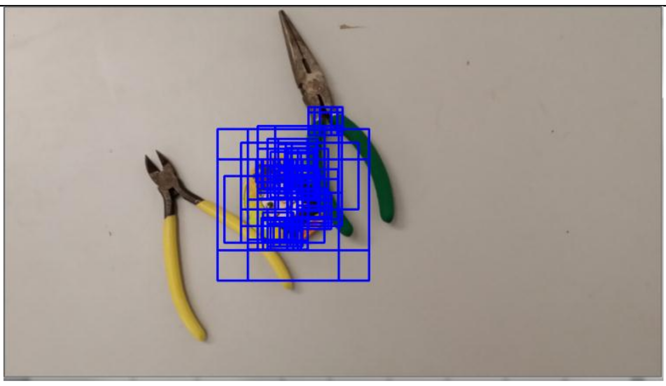
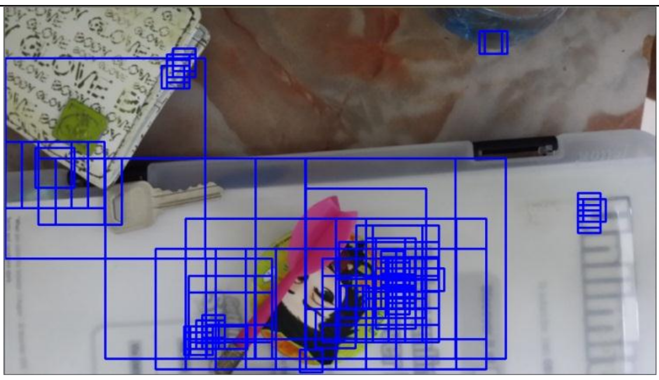
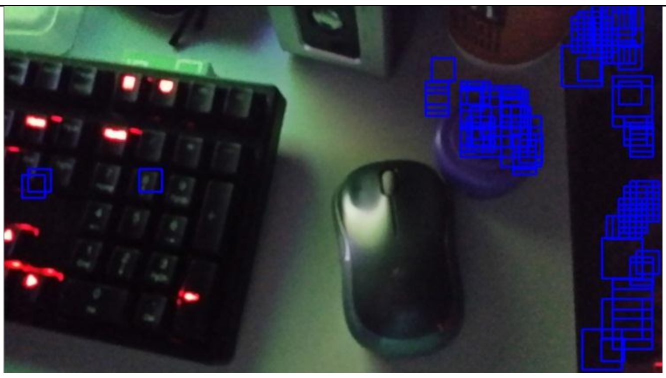
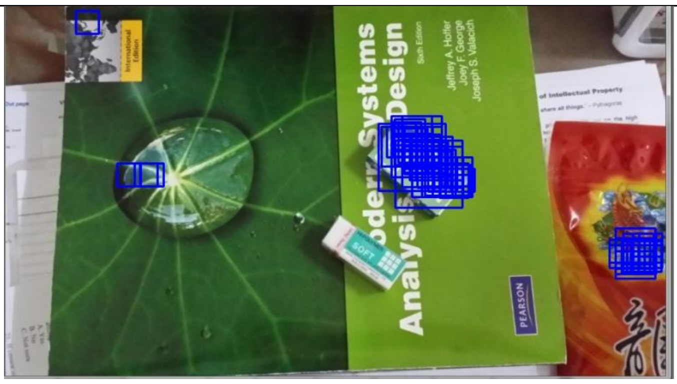
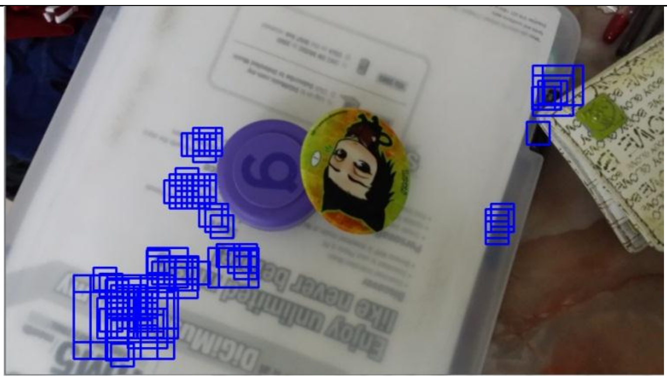
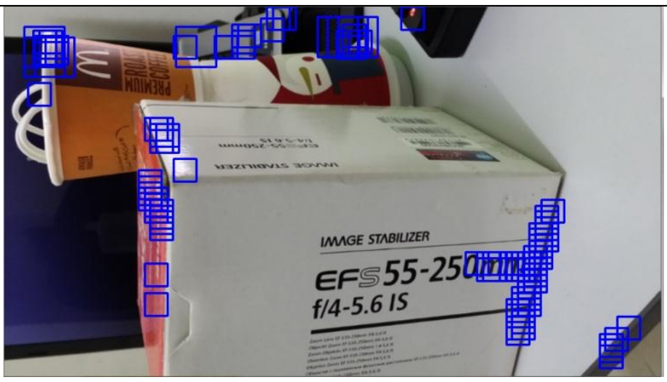
Template image	**Search Space image**	**Return true positive detection**
		✓
		✓
		✓

Table 5.4 Sample marked area as potential object position - 2 (Colour histogram comparison)

Template image	**Search Space image**	**Return true positive detection**
		✓
		✗ Reason: Object is slanted and too dark
		✗ Reason: Object is highly occluded

5.2.2 Experiment stage 2 – Comparing vertical and horizontal strip

This test is conducted by using the result returned from Experiment stage 1. Given the same template and data set, the vertical and horizontal strips are compared via its HSV colour model with it Hue value only.

Table 5.5 Summarized result from direct pixel comparison

Lightning conditions	**Type of background**	**Total Tested Image**	**True positive detection**	**False positive detection**	**True negative detection**	**False negative detection**
Low light	Same colour with targeted object	2	50%	50%	100%	50%
Low light	Different colour with targeted object	2	50%	100%	100%	50%
Day light	Same colour with targeted object	12	83.33%	66.67%	58.33%	16.67%
Day light	Different colour with targeted object	34	85.29%	76.47%	44.11%	14.71%
Summary		**50**	**82%**	**76%**	**52%**	**28%**

Table 5.6 Sample marked area as potential object position - 1 (Direct pixel comparison)

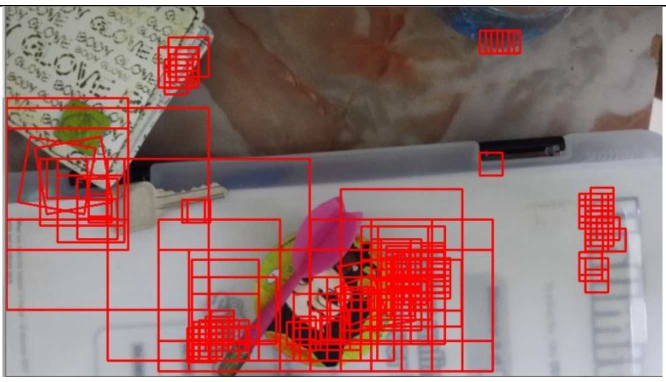
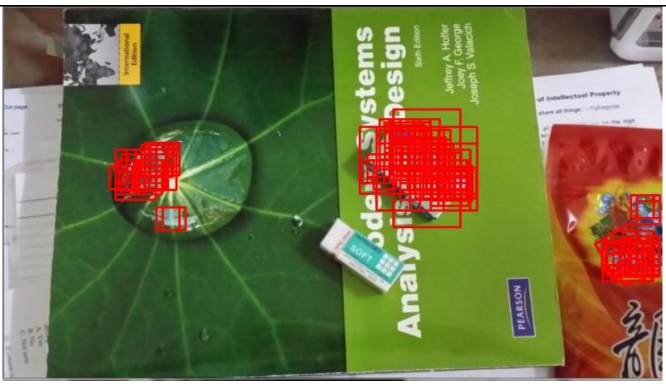
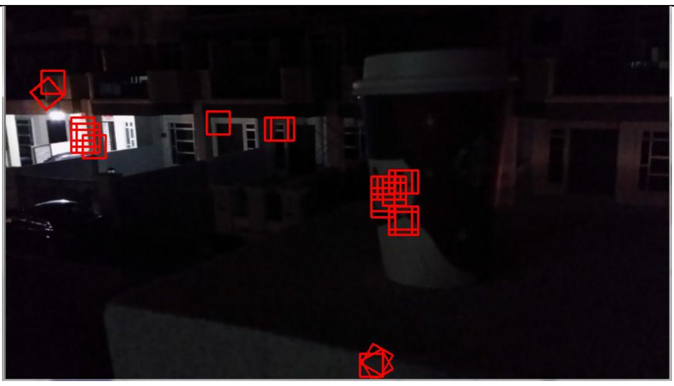
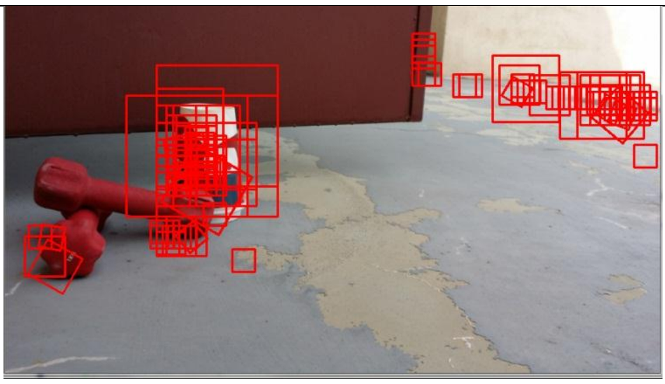
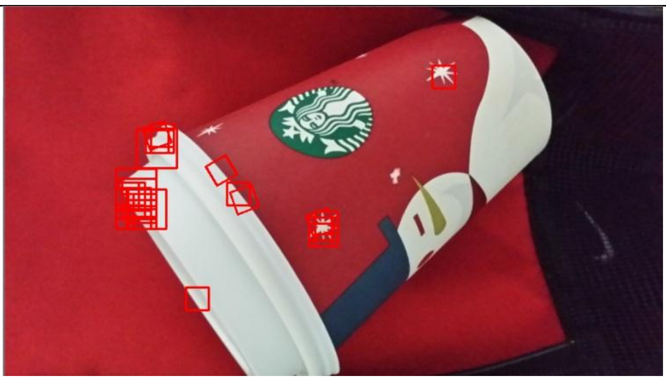
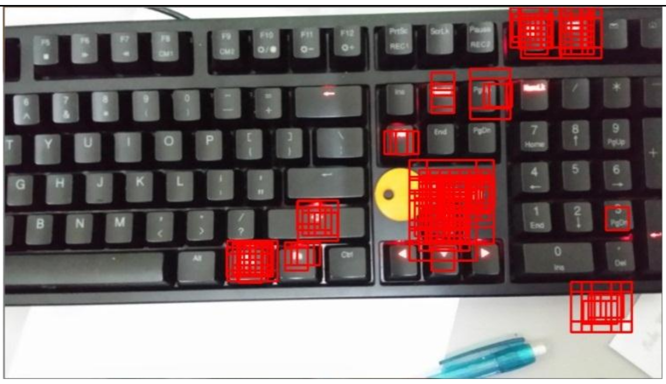
Template image	**Search Space image**	**Return true positive detection**
		✓
		✓
		✓

Table 5.7 Sample marked area as potential object position - 2 (Direct pixel comparison)

Template image	Search Space image	Return true positive detection
		✓
		✗ Reason: Object is too near
		✗ Reason: Object is too bright

5.2.3 Experiment stage 3 – Neighbour comparison

This test is conducted by using the result returned from Experiment stage 3. The neighboring pixels of the marked area center point will be compared with the neighboring pixels of template image center point. The marked area which does not filter off will be accounted as the **final result**.

Table 5.8 Summarized result from neighbor comparison

Lightning conditions	**Type of background**	**Total Tested Image**	**True positive detection**	**False positive detection**	**True negative detection**	**False negative detection**
Low light	Same colour with targeted object	2	50%	50%	100%	50%
Low light	Different colour with targeted object	2	50%	100%	100%	50%
Day light	Same colour with targeted object	12	75%	41.67%	75%	25%
Day light	Different colour with targeted object	34	85.29%	14.71%	82.35%	14.71%
Summary		**50**	**80%**	**26%**	**82%**	**20%**

Table 5.9 Sample marked area as potential object position - 1 (Final Result)

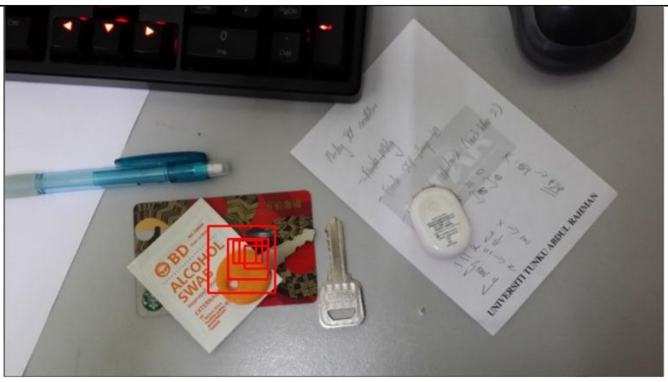
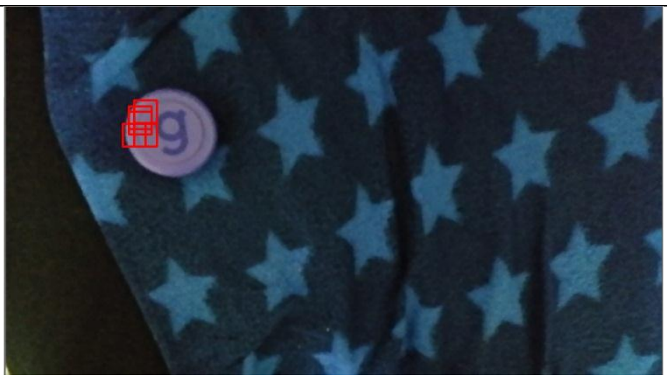
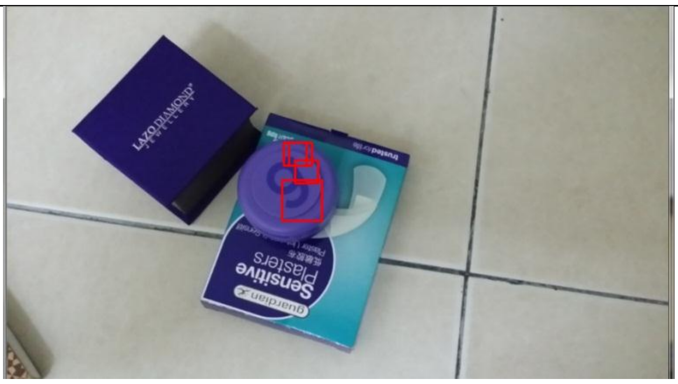
Template image	**Search Space image**	**Return true positive detection**
		✓
		✓
		✓

Table 5.10 Sample marked area as potential object position - 2 (Final Result)

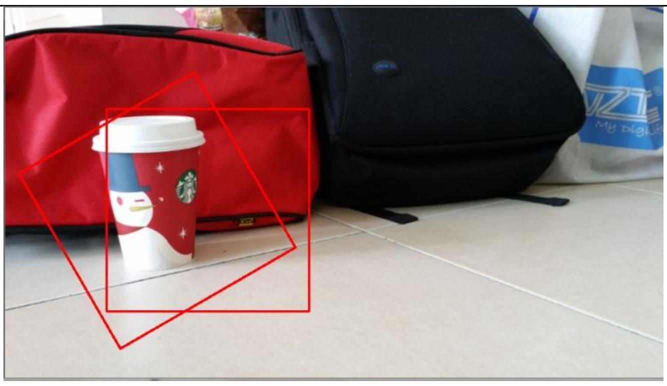
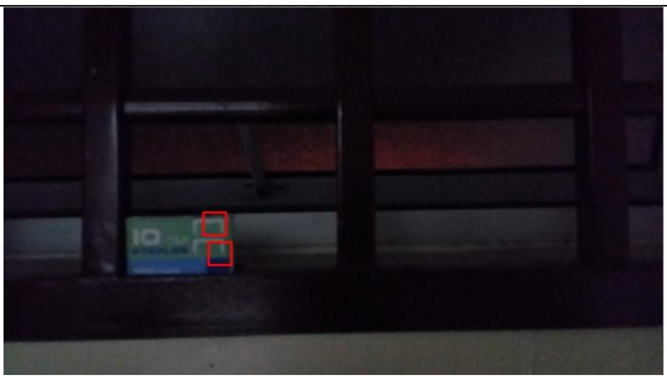
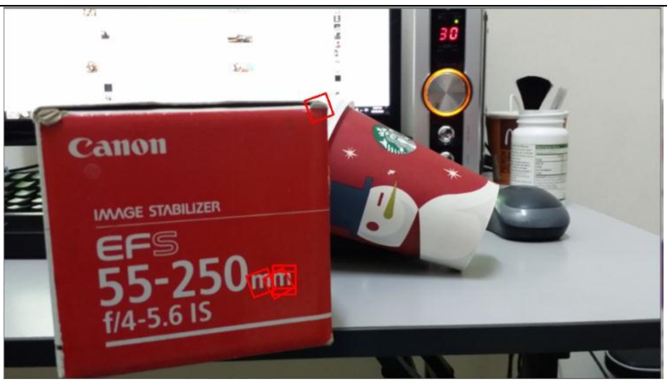
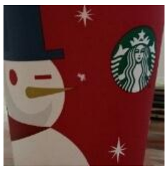
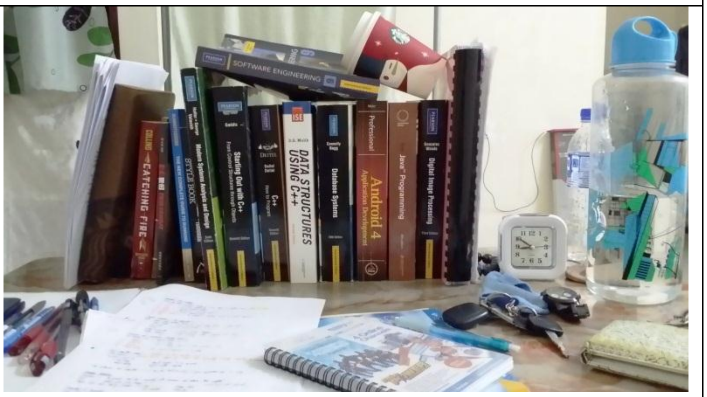
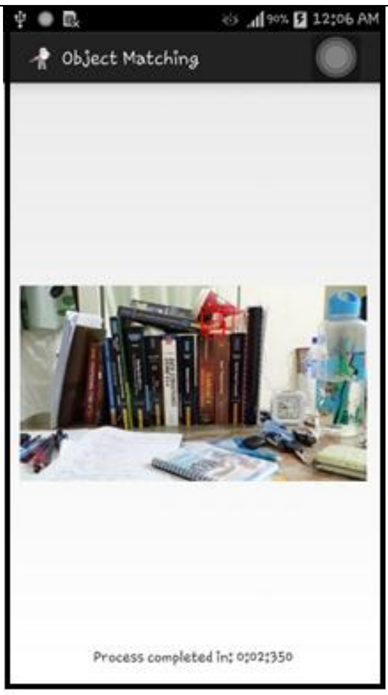
Template image	**Search Space image**	**Return true positive detection**
		✓
		✗ Reason: The neighbour pixels having similar colour distribution
		✗ Reason: Both object having similar colour distribution

Table 5.8 shows the sample result of final deliverable on Samsung Note 3 device by using the proposed method.

Table 5.11 Sample Phone Result

Template image	Search space image
	
Result	
 Object Matching Process completed in: 0:02:350	

5.3 Summary

In order to achieve higher accuracy of searching result, the lightning conditions must be sufficient for the object colour to be read. It has been found that the application worked satisfactory in searching the object with a precision rate at 75% and negative predictive value at 80% by using the captured data set.

In contrast, some limitations are found within the proposed method. Lightning conditions and the environment colour would greatly impact on the searching result. Meanwhile search space image with repeating colour pattern would trigger high hit rate of false positive result. Other than that, current system does not have sophisticated cropping function. It only support square sized template. Thus, the cropping function has to be redesigned in the future to support shape searching. Future work, improvement and recommendation will be discussed in Chapter 6.

Chapter 6 Conclusion and Future work

This paper accounts the motivation of delivering an object searching application for smartphone by using computer vision technique. In order to achieve the objectives, the conventional template matching technique provided by OpenCV library was enhanced and restructured. The final deliverables of this project will benefit the users by showing the object location on the image itself.

6.1 Limitation

Along the research of this project, the final deliverable has performed up to satisfaction. However, there are few drawbacks found in the system are listed below.

- <u>Template image's colour</u>

  The selected template image will generate bad result when it colour are similar to the captured environment. Most of this scenario will generate false negative result and marked some other object as the targeted object. In addition, given the significant colour to identify the object is located at the border and less information at the center point, the returned result is likely to be a failure case as the applied mask will offset the information at the image border.

- <u>Object's lightning condition</u>

  If the targeted object brightness is greatly difference from the template image, the system will likely read the object as other colour. As a result, the searching process is failed.

6.2 Future Works

In order to make the application more practicable and usable, some functions mentioned below should be enhanced in more sophisticated ways:

Source object detection

In this project, the cropping support only square shape. Thus, the template images are always in square size and the cropped region must within the object. Extra information such as the template background will causes great impact to the searching process. If the object is correctly identified, then it can reduce the probability of false negative result.

The recommended solution is to implement cropping function that support free selection, or called as lasso. Other than that, it is recommended to find the contour of the object. The contour act as a mask to ensure only the object itself is read as ROI.

Comparing non-square shaped object

The system will perform better if the sliding window is not restricted in square dimension when searching for non-square shape object. In order to do that, the cropping function must be enhanced before the system can search for it. The detection of template image's object shape can be implemented by reading its contour and use it as the default sliding window dimensions.

Multithreading

In order to obtain the best performance for object searching, multithreading on multicore devices should be implemented to fully utilize the hardware performance. Android Render Script provides an independent platform computation engine that accelerate the application that requires extensive computational. Instead of supporting one single search space at a time, the application can be further improve by supporting multiple search space images at a time when performing search.

6.3 Potential Application

This section listed out potential applications that can be developed by using the methods proposed in this project and ultimately benefits the end users.

- Object finder apps
  - ➢ Given a warehouse with different items scatter around, the user could make use of this application to search for their targeted object easily. This is because the template image can be easily downloaded from inventory database.
- Object detection apps
  - ➢ During outdoor research activity, it is possible to snap up to hundreds of photo. It is time consuming for an individual to view through every single image to search for particular object in the image. The user could make use of this application to search for the targeted object.
- Market behaviour analysis apps
  - ➢ Given an open space environment with peoples in difference shirt colour. Marketer could use this application to analysis how many are wearing blue shirt by snapping a simple photo with their own mobile device.

Bibliography

Android, 2013. *Android - What's New*. [Online] Available at: http://www.android.com/whatsnew/ [Accessed 20 June 2013].

Gonzalez, R.C. & Woods, R.E., 2010. *Digital image processing*. 3rd ed. New Jersey: Prentice Hall.

Google, 2011. *Google Goggles*. [Online] Available at: http://www.google.com/mobile/goggles/#book [Accessed 28 November 2012].

Handa, S., 2011. *What is Google Goggles*. Lecture. Illinois: University of Illinois.

Hoffer, J., George, J. & Valacich, J., 2011. *Modern Systems Analysis and Design*. 6th ed. New Jersey: Pearson.

Ismail, N. *et al*., 2012. Analysis of Real-Time Object Detection Methods for Android Smartphone. In *3rd International Conference on Engineering and ICT*. Melaka, 2012.

Jeong, K. & Moon, H., 2011. Object Detection using FAST Corner Detector based on Smartphone Platofrms. In *First ACIS/JNU International Conference on Computers, Network, Systems, and Industrial Engineering*. Jeju Island, 2011.

JJIL, 2008. *Jon's Java Imaging Library, for mobile image processing*. [Online] Available at: https://code.google.com/p/jjil/ [Accessed 25 June 2013].

Laganière, R., 2011. *OpenCV 2 Computer Vision Application Programming Cookbook*. 1st ed. Mumbai: Packt.

OnGuardOnline, 2011. *Understanding Mobile Apps*. [Online] Available at: http://www.onguardonline.gov/articles/0018-understanding-mobile-apps [Accessed 17 June 2013].

OpenCV, 2013. *About - OpenCV*. [Online] Available at: http://opencv.org/about.html [Accessed 14 June 2013].

OpenCV, 2013. *Template Matching*. [Online] Available at: http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html [Accessed 13 June 2013].

Qian, X., Zhu, G. & Li, X., 2012. Comparison and Analysis of the Three Programming Models in Google Android. *First Asia-Pacific Programming Languages and Compilers Workshop (APPLC)*, 14 June.

Radhakrishna, A., 2008. *Salient Region Detection and Segmentation*. [Online] Available at: http://ivrgwww.epfl.ch/~achanta/SalientRegionDetection/SalientRegionDetection.html [Accessed 29 June 2013].

Raphael, J., 2009. *Google Googgles review*. [Online] Available at: http://www.pcadvisor.co.uk/reviews/software/3208325/google-goggles-review/?pn=1#ixzz2DbLweNzi [Accessed 26 November 2012].

Sachs, J., 2013. *Precise Gaussian Blur Transformation Notes*. [Online] Available at: http://www.dl-c.com/Temp/downloads/Whitepapers/Precision%20Gaussian%20Blur%20Notes.pdf [Accessed 30 June 2013].

Sams, R., 2011. *Introducing Renderscript*. [Online] Available at: http://android-developers.blogspot.com/2011/02/introducing-renderscript.html [Accessed 28 November 2012].

Schaap, J., 2011. *App Review: Google Goggles*. [Online] Available at: http://mastersofmedia.hum.uva.nl/2011/10/10/app-review-google-goggles/ [Accessed 26 November 2012].