# OBJECT SEGMENTATION BASED ON MULTI ANGLE IMAGES ON MOBILE DEVICE

BY

PONG CHANG SHENG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Perak Campus)

January 2018

UNIVERSITI TUNKU ABDUL RAHMAN

# REPORT STATUS DECLARATION FORM

**Title**: ______________________________________________

______________________________________________

______________________________________________

**Academic Session**: _____________

I ______________________________________________

**(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**:

_________________________

_________________________

_________________________

_________________________

Supervisor's name

**Date**: _____________________ **Date**: ____________________

# OBJECT SEGMENTATION BASED ON MULTI ANGLE IMAGES ON MOBILE DEVICE

BY

PONG CHANG SHENG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Perak Campus)

January 2018

# DECLARATION OF ORIGINALITY

I declare that this report entitled "OBJECT SEGMENTATION BASED ON MULTI ANGLE IMAGES ON MOBILE DEVICE" 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 express my special thanks of gratitude to my supervisors, Mr. Tou Jing Yi who has given me the golden opportunity to be involved in this project, which helped me to do a lot of research and came to know a lot about topic related to image processing.

Secondly, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

# ABSTRACT

In Computer Vision, image segmentation has long been a popular topic and serves as a basis for lots of the software and mobile application on the market. Image segmentation usually refers to the process of segmenting an object of interest apart from its background. Most of the image segmentation can produce accurate results but with the limitation of high computational complexity.

However, in the recent years, mobile devices have been a popular trend and under constant improvement. Mobile devices are now equipped with mid-end to high-end specification but doesn’t burn your pocket too much. Thus, image segmentation can now be integrate onto mobile application by reducing the computational complexity.

This project aims to produce an algorithm that is capable of automatically segments object from its background by facing the camera towards the object and input an object bounding rectangle.

A new system which integrated KCF tracking algorithm and GrabCut segmentation was proposed. The system is able to track an object in real time and then extract the object of interest from its background. The results confirms the feasibility of real-time object segmentation and tracking on a mobile device.

## TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| SfS | Shape from Silhouettes |
| GMM | Gaussian Mixture Model |
| KCF | Kernelized Correlation Filters |
| FPS | Frames per second |

# Chapter 1: Project Background

## 1.1 Background Information

From a computer perspective, an image is just a bunch of binary value or colour value stored inside a vector of pixels. Without any processing, the image is meaningless for a computer. Human has the ability to interpret meaningful information from a image automatically. However, a computer need special instruction to process the image to gain meaningful information out of it. For example, when given a picture of a man standing in a hall, human can easily know extract the object of interest which is the man. Furthermore, human can also gain information like the venue and situation based on the image. In contrast, a computer will only see the image as a bunch of colour connected together. By applying algorithm such as image segmentation or object detection, computer will be able to extract important information such as the object, foreground and background from the image.

Image segmentation is the process of segmenting an images into regions that represent something that is meaningful. Furthermore, object detection refers to the process of extracting an object from an image or a video sequence.

In computer vision, image segmentation and object detection is always a popular topic as it act a core and basis for most of the computer vision related application. Over the decades, more and more algorithm or solution with different strength has been introduced. Some of it pay more attention to computation time or resource usage while some of it might focus more on the accuracy of result and there might even be some approaches that try to balance between those two. As a result of various solution available for reference and research, there has been a constant improvement in the object segmentation solution. A traditional 2D image segmentation or object detection only accept a single image as input while the advance solution now takes in a sequence of image from different angle of an object or even a video taken at a fixed motion around an object.

## 1.2 Project Scope

The scope of the project includes:

i. Development of an application capable to track and segment an object in real time camera preview.
ii. To investigate the constraints or relationships of background/foreground between the camera preview frames.
iii. To initialise the starting point to start the segmentation.
iv. To track an object's continuously in a real time preview feed.
v. To implement image segmentation based on object tracking result.

## 1.3 Project Objectives

The objectives of the project are:

i. To segment an object from its background based on the tracking result.
ii. To track an object in real time camera preview based on a user defined object bounding box.

## 1.4 Problem Statement and Motivation

One of the famous approach in object detection is by applying background subtraction. However, this method usually requires that the background pixel values are at minimum changes which means that it is a static background while foreground pixel changes regularly (Lee *et. al*., 2007). These methods also requires prior knowledge of the background which makes the solution not so not useful in some cases where the background information is not known. Moreover, dynamic background is a long existed challenge in object detection and object segmentation applications. The background appearance undergoes various changes over time which makes the background/foreground modelling becomes complex.

Some existing approach or solution to segment object from an image required interactive user input to guide the segmentation by segmenting region of interest manually. This means that for every picture or dataset, a user will be required to manually guide the segmentation one by one. This is fine if the dataset is relatively small, however, in the industry, dataset is usually very large. A large dataset will be time consuming for a user to finish the segmentation because large dataset usually introduced a big difference between data and it is not feasible in real life. This infer that manpower will gradually become a bottleneck of the system. Furthermore, interactive input approach required the user to be equipped with basic or at least some knowledge of image segmentation and the system and this might be a problem for those who are not familiar with the system or even computer. So, an object segmentation algorithm with minimal and easy user input will be able to reduce the need of manpower and be more user friendly since it does not require much knowledge.

Furthermore, with the current popularity of mobile devices, most of the image segmentation didn’t focus on mobile aspect. The existing algorithm might be too complex to run on a mobile device. A complex algorithm might runs too slow on a mobile device due to hardware limitation.

## 1.5 Report Organization

This report includes 5 chapters in total. The first chapter is the introduction which is mainly on introducing the project background, motivation for this project and also the project scopes and objectives. The next chapter covers the review of literature and past year researches related to image segmentation.

The third chapter explains the proposed algorithm. Methods used in the algorithm is also explained in details in this chapter. Chapter 4 includes the methodology and tools used in the project. This chapter also describes the implementation issues or other experimental solution and includes the results of the proposed approach.

The last chapter concludes this project with additional information and includes some feasible further improvement

# Chapter 2: Literature Review

## 2.1 Introduction

Over the course of the years, numerous amount of methods and approaches have been developed for object segmentation and object detection. However, a challenge that always existed is to detect object from dynamic background or segmenting object where the background information is not known. For example, segment an object where single image is available per viewpoint. This problem can be relate to the classic problem in computer vision which is the Shape-from-Silhouettes (SfS). SfS is a popular problem in Computer Vision has gain a lot of attention because in many cases (e.g. texture-less objects) it is the only possible way for approximating 3D shape visually using sensors. Even in textured scenes, silhouettes are known to improve reconstruction accuracy in the cases of thin or awkward structures (Neill *et. al*., 2011).

According to Michael and Eduard (2012), 3D object reconstruction is a famous challenge in computer vision. Some of the existing SfS approaches are based on static background subtraction or manual segmentation (required interactive user input to segment the object), which neither of it is always feasible. In a large data set, system which rely on interactive user input is tedious and thus the relationship that exist in the image sequence can be exploited to produce automatic segmentation. Since the image sequence contains the same 3D object of interest, thus the object segmentation must fulfil a silhouette coherency constraints (Carlos *et. al*., 2007).

## 2.2 Existing Object Segmentation Approach

Lee, Woo and Boyer (2007) proposed a solution to identifying foreground from multiple images automatically. This method eliminates the need for interactive user input and also prior knowledge of the images' background. Probabilistic Modelling method is used in this solution to compute the dependencies between variables such as background, foreground, and other unknown variables.

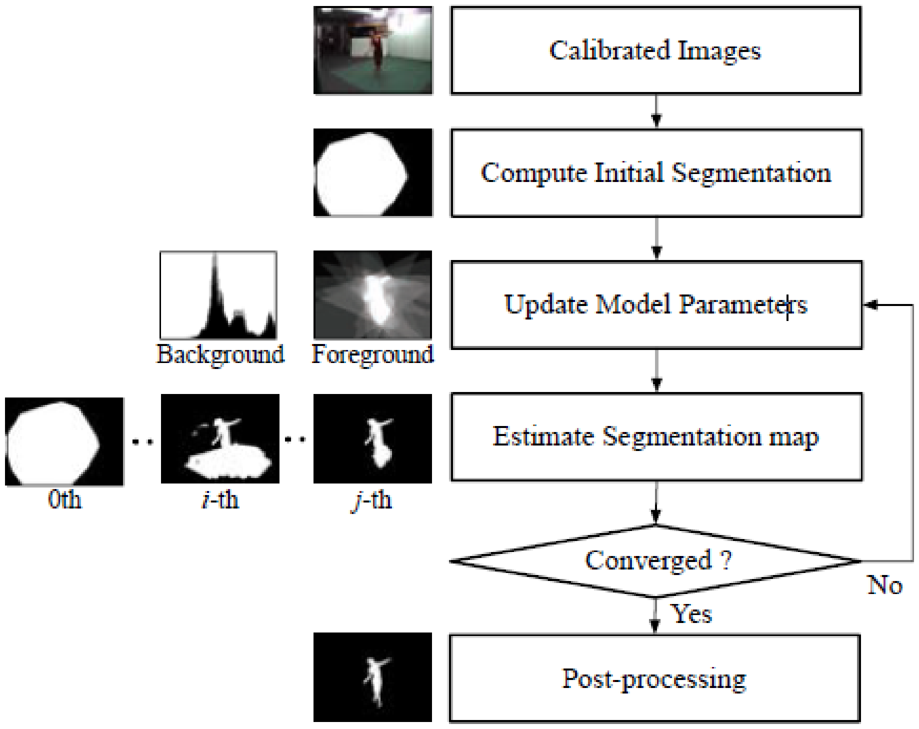


Figure 2.2.1: Overall procedure for proposed foreground extraction (Lee *et.al.*, 2007)

It is mentioned that the spatial consistency of the objects in all the images should be consistent. They used a self-modified silhouette calibration ratio with a Gaussian distribution to compute the spatial consistency:

$$R_x = e^{-(1-C_x)^2/\sigma^2}$$

An image likelihood term has been developed to help determine whether a pixel belongs to background or foreground.

$$P_r(T_x^i | B^i, S_x^i, \tau) = \begin{cases} H_B(T_x^i) & if\ S_x^i = 0 \\ P_f & if\ S_x^i = 1 \end{cases}$$

Furthermore, graph cut method is used to compute the optimal segmentation map for images of all the views. It is done iteratively until the segmentation map converges. Computing min-cut minimizes the segmentation energy defined in the equation:

$$E_{total} = \sum_{x \in I^i} \lambda_1 E_p(x) + \sum_{(x,y) \in N, S_x \neq S_y} \lambda_2 E_n(x, y)$$

Graph cut based segmentation is also used as a post-processing to remove any misclassified pixels.

Figure 2.2.2 below illustrate the result of the segmentation over iteration.

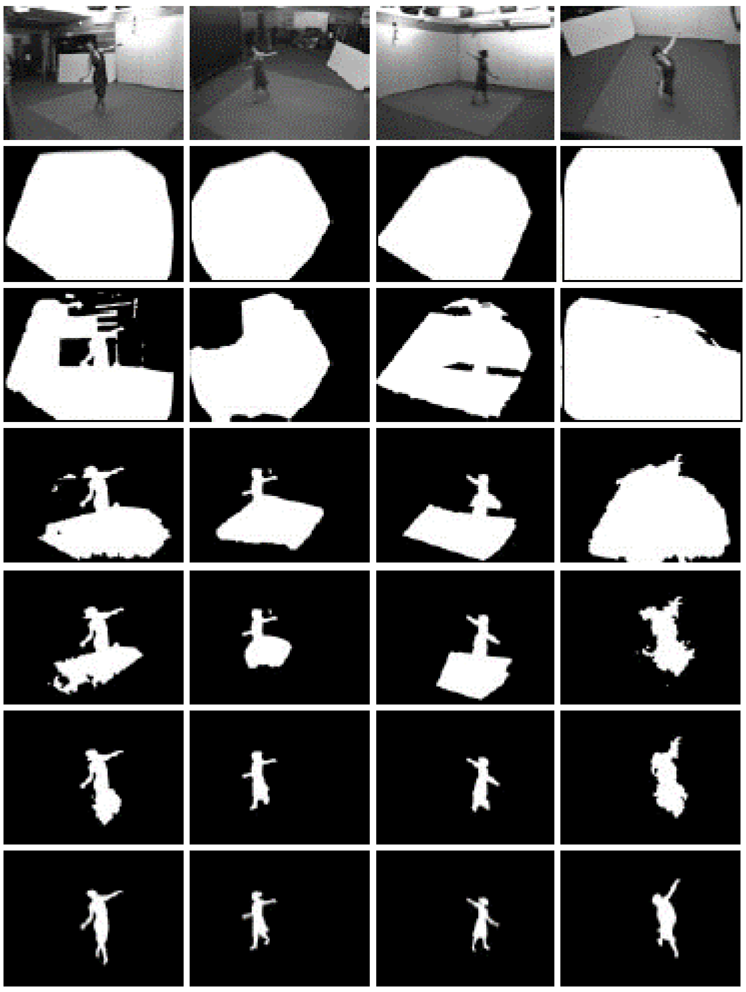
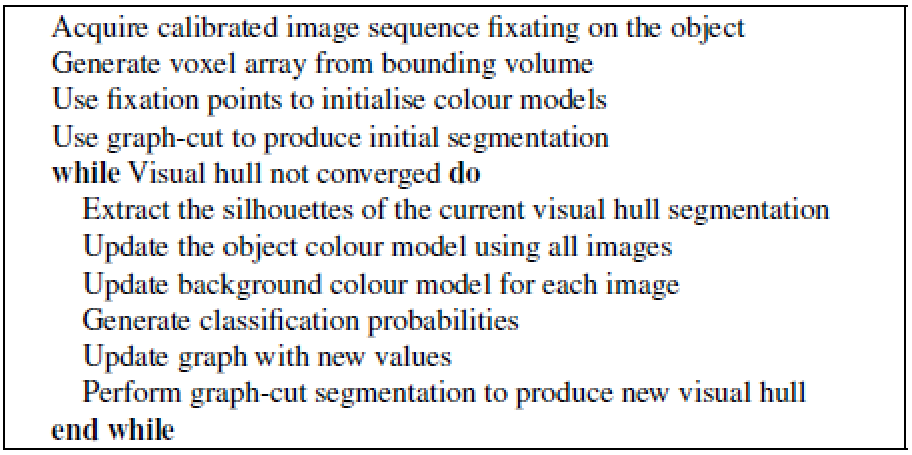


Figure 2.2.2: Foreground extraction result with 'Dancer' data set (Lee *et.al.*, 2007)

Campbell, Vogiatzis, Hernandez and Cipolla (2007) proposed a solution to segment the object of interest automatically based on the calibrated images sequence taken from certain camera views. The automation is achieved by exploiting the fixation point of the object. In this proposed solution, the object of interest is assumed to be at the position around the central of the image which gives a fixation point in order to initialize the segmenting process.



```
Acquire calibrated image sequence fixating on the object
Generate voxel array from bounding volume
Use fixation points to initialise colour models
Use graph-cut to produce initial segmentation
while Visual hull not converged do
    Extract the silhouettes of the current visual hull segmentation
    Update the object colour model using all images
    Update background colour model for each image
    Generate classification probabilities
    Update graph with new values
    Perform graph-cut segmentation to produce new visual hull
end while
```

Figure 2.2.3: The iterative segmentation algorithm (Campbell *et.al.*, 2007)

The 2 main stages in this method is developing the colour model and the volumetric graph-cut. The colour model is developed in order to provide a probabilistic likelihoods to determine whether a pixel belong to foreground or background. A K-component Gaussian Mixture Model is used to model the likelihood where K=5:

$$\rho(u|\pi_k,\mu_k,\Sigma_k) = \sum_{k=1}^{K} p(k)\, p(u|\pi_k,\mu_k,\Sigma_k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(u|\mu_k,\Sigma_k)$$

In the process of volumetric graph-cut, the energy to be minimized is given by the equation:

$$E(O,B|,\{I_m,\ominus\} = \lambda E_{vol}(O,B,\{I_m\},\ominus) + (1-\lambda)E_{surf}(O,B,\{I_m\})$$

There is two important term to be highlight in the energy minimization equation is: the Volume term, $E_{vol}$ and Boundary term $E_{surf}$.

Volume term is used to classify a voxel (pixel represented in 3d space) as outside or inside of the object boundary. This term is constructed from the colour model developed using the GMM.

$$E_{vol}(O, B, \{I_m\}, \ominus) = \sum_{v_n \in V} \begin{cases} (1 - [\mathcal{L}_o(v_n, \ominus) - \emptyset])\ v_{n\ \in O} \\ (1 + [\mathcal{L}_o(v_n, \ominus) - \emptyset])\ v_{n\ \in O} \end{cases}$$

Boundary term is used to determine the visual hull’s boundaries by using the colour discontinuities within the images.

$$E_{surface}(O, B, \{I_m\}) = \sum_{(v_i, v_j) \in \varepsilon, \substack{v_i \in O \\ v_j \in B}} \overset{max}{m}\ e^{-\beta Z_m(v_i, v_j)}$$

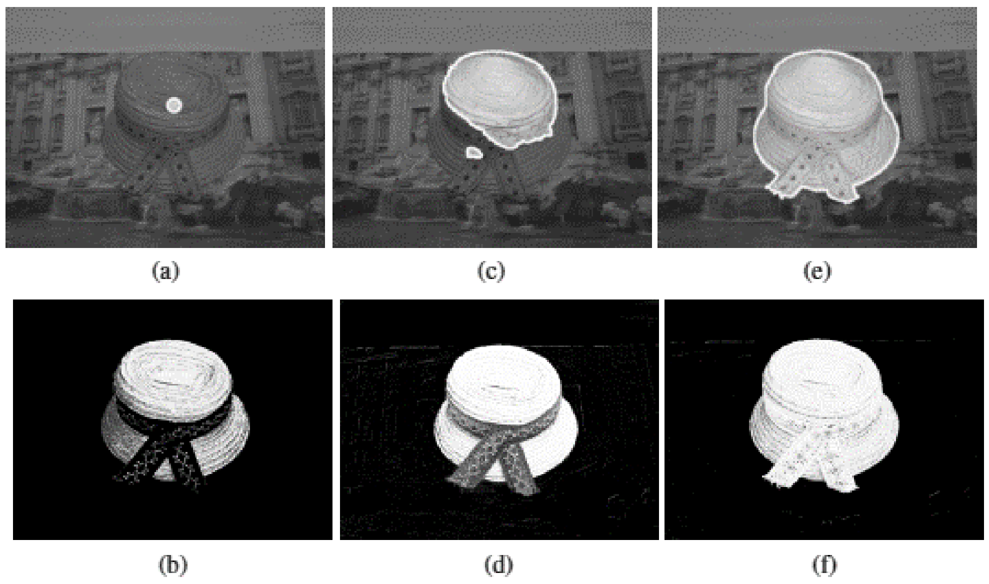


Figure 2.2.4: Colour model iteratively learning process (Campbell *et.al.*, 2007)

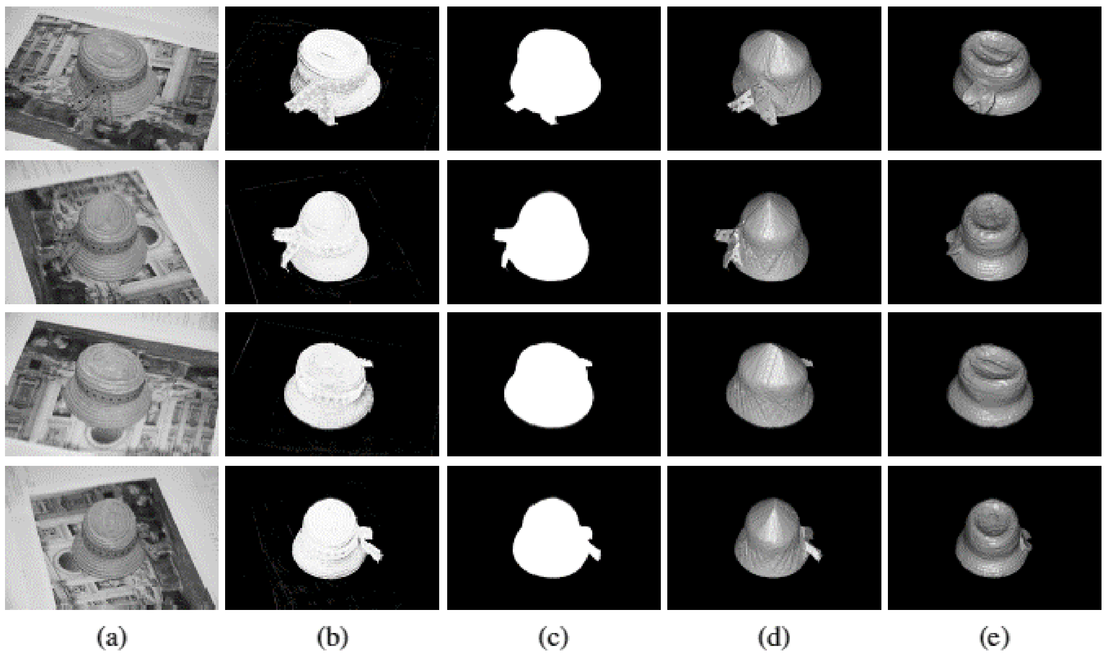


Figure 2.2.5: Hat sequence segmentation result (Campbell *et.al.*, 2007)

Kootstra, Bergstrom and Kragic (2010) proposed a solution to automatically detects and segments objects from unknown background in real-time. This solution also focuses on the computation time. The segmentation process are shown in figure 2.2.6.

I. Segment the image into super pixels.
II. Pre-segmentation of the background and foreground.
III. Update the background and foreground information.
IV. Repeat until convergence:
   a. Graph-cut segmentation.
   b. Update of background and foreground information.
V. Classify the super pixels connected to the fixated super pixels as foreground.

Figure 2.2.6: Overall object segmentation steps

Initially, multiple fixation point has been set using object detection algorithm proposed by other researcher to initialize the segmenting process. First, the image is segmented into super pixels and cluster those regions which are homogenous in colour. This can greatly reduce the computational complexity.

Secondly, those super pixels which contains the fixation point will be labelled as foreground, else labelled as the background.

Next, the foreground and background information is updated with the colour histogram and the dominant plane of image. The colour histogram of background and foreground are computed with the equation below:

$$C_F(a,b) = \sum_{s \in F} C_s(a,b)$$

$$C_B(a,b) = \sum_{s \in B} C_s(a,b)$$

After that, graph-cut segmentation is applied on the image until the foreground super pixels converges. The energy minimisation function is defined as:

$$E(l) = \sum_{s \in S} D_s(l_s) + \propto \sum_{\{s,t\} \in N} V_{s,t}(l_s, l_t) \cdot T(l_s \neq l_t)$$

where N is the set of neighbouring super pixels s and t.

During each iterative, the background and foreground information is updated after every graph-cut segmentation.

Lastly, those super pixels that is connected to the fixated super pixels are also classify as the foreground.

Figure 2.2.7 show the comparison between this proposed solution to a previous proposed solution.

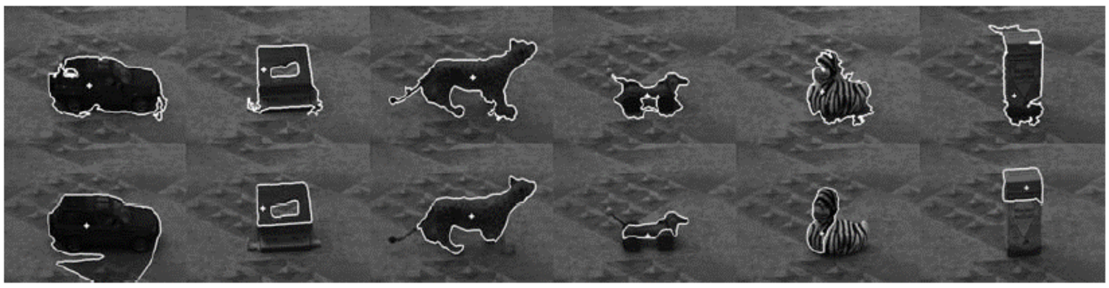


Figure 2.2.7: Result from proposed method (first row) and result of previous segmentation method (second row) (Kootstra *et.al.*, 2010)

Campbell, Vogiatzis, Hernandez and Cipolla (2011) proposed a new solution to obtain the 3D shape of the object of interest with unknown background based on image sequence of known camera calibration.

```
Input
- A calibrated set of $M$ images $I_1 .. I_M$
Initialisation
- Obtain bounding volume from visibility
foreach image $I_m$, $m = 1 .. M$ do
    - Group pixels into superpixels $\{s_i\}$
    - Extract fixation point
end
- Learn background colour model from outside bounding box
- Learn object colour model from fixation points
- Generate the edge matrix $W$
Main Loop
while visual hull not converged do
    foreach image $I_m$, $m = 1 .. M$ do
        - Evaluate object likelihood
    end
    - Perform graph-cut to label superpixels
    - Enforce silhouette consistency
    - Update object colour model from new silhouettes
end
Output
- The converged object silhouettes and visual hull
```

Figure 2.2.8: Overall algorithm of the proposed solution (Campbell *et.al.*, 2011)

Firstly, all the image sequence was over-segmented to obtain a set of super pixels. Then, each super pixels was labelled as either background or an object. Each super pixel has an associated position decided by the centre of the super pixel. The edge matrix W is generated by computing an edge adjacency matrix W where $W_{ij}$ denotes the weight of the edge between two super pixels i and j.

$$W_{i,j} = \begin{cases} p(d_n)c(s_i, s_j) & \begin{matrix} s_i \in I_m, s_j \in N(I_m) \\ d(s_i, s_j) \in [d_n] \end{matrix} \\ c(s_i, s_j) & s_i \in I_m,\ s_j \in \ I_m \end{cases}$$

Next, two colour models are maintained by using K component Gaussian Mixture Models (GMMs) mentioned before.

The main loop of the algorithm determines whether the pixel belong to foreground or background using the colour models and combined with the edge matrix W to perform graph-cut to label super pixels as foreground/background. The energy model is formulated as the equation below:

$$E(\{s_i\}) = E_d(\{s_i\}) + \varphi E_s(\{s_i, s_j\})$$

In order to ensure the silhouette consistency, the resulting silhouettes are intersected to produce a visual hull. Lastly, the object colour model is updated using the new silhouettes until convergence.

Figures 2.2.9, 2.2.10 and 2.2.11 show the result of Campbell, Vogiatzis, Hernandez and Cipolla proposed method.

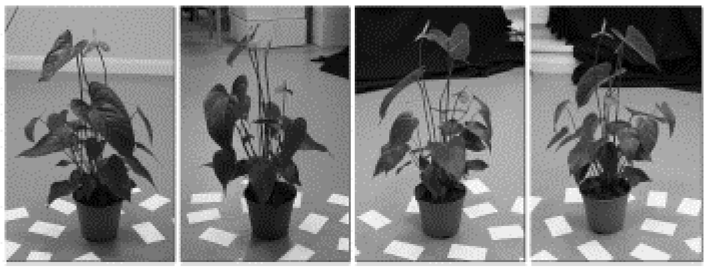


Figure 2.2.9: Image sequence of a vase from different view (Campbell *et.al.,* 2011)

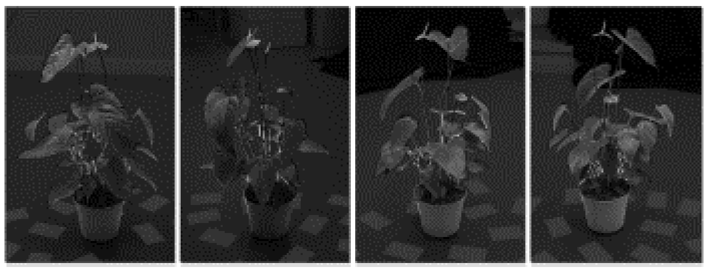


Figure 2.2.10: Result of the proposed method (Campbell *et.al.,* 2011)

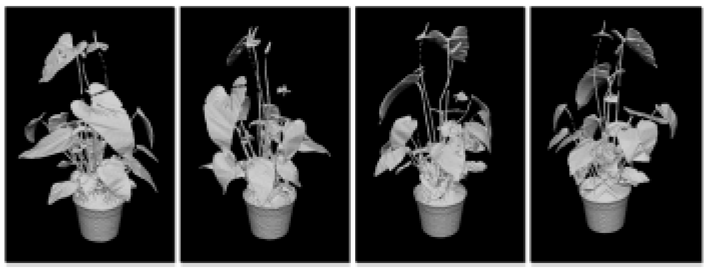


Figure 2.2.11: Result shown as a visual hull (Campbell *et.al.,* 2011)

Garcia, Kelley and Yang (2015) proposed a solution which implemented a fast and highly reliable algorithm to segment an object from an image taken using mobile devices' camera. It is an interactive application which required a certain user interaction. Due to the nature of interactive application, it needs to be fast and responsive which means it requires lower computational complexity.

1. Down-sample the input image.
2. Apply SLIC superpixel algorithm to over-segment and reduce pixels complexity.
3. Interactive section where user input is needed to mark the foreground and background.
4. Graph cut is applied on the pre-segmented image which consists of users' marking.
5. Successful foreground segmentation will be up-sample to original input image's quality.
6. Boundary is further edited by Bezier curves method.

Figure 2.2.12: Overall steps of segmentation

Firstly, the input target image will be down-sample to a lower resolution image. This action is done to reduce the number of pixels as a high resolution means higher pixels and longer computation time which might be unbearable for an interactive application. Next, SLIC superpixel algorithm is applied to further decrease the number of pixels.

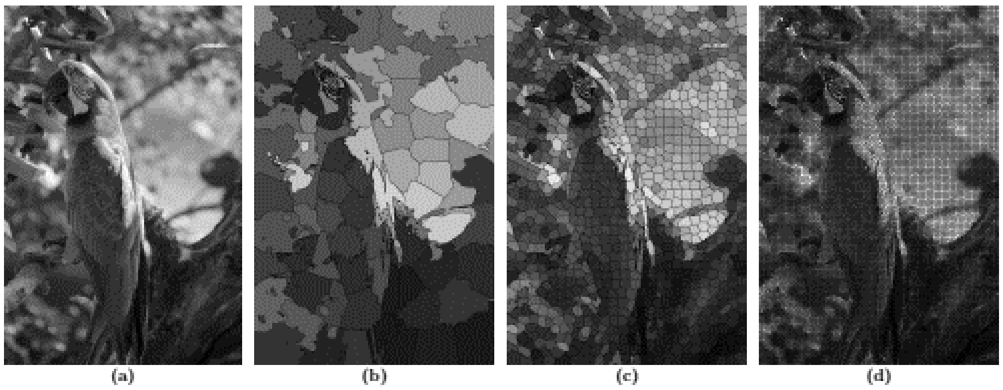


Figure 2.2.13: (a) Original image and pre-segmented image using SLIC superpixel with (b) 100, (c) 1000, and (d) 10,000 number of superpixels. (Garcia *et.al.*, 2015)

Interactive user input will be required to mark the foreground and background to be used in the graph cut segmentation process. The successful segmentation will then be up-sample back to original resolution for an accurate result. The object boundary is further edited by using Bezier curve curves.

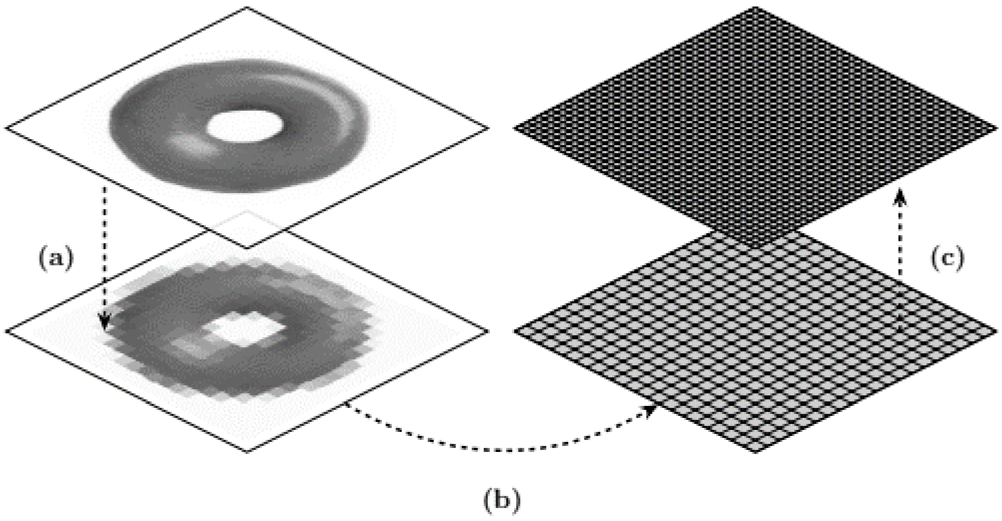


Figure 2.2.14: (a) Input image is downsample. (b) Graph cut applied on the downsampled image. (c) Segmentation is mapped onto original resolution (Garcia *et.al.,* 2015)

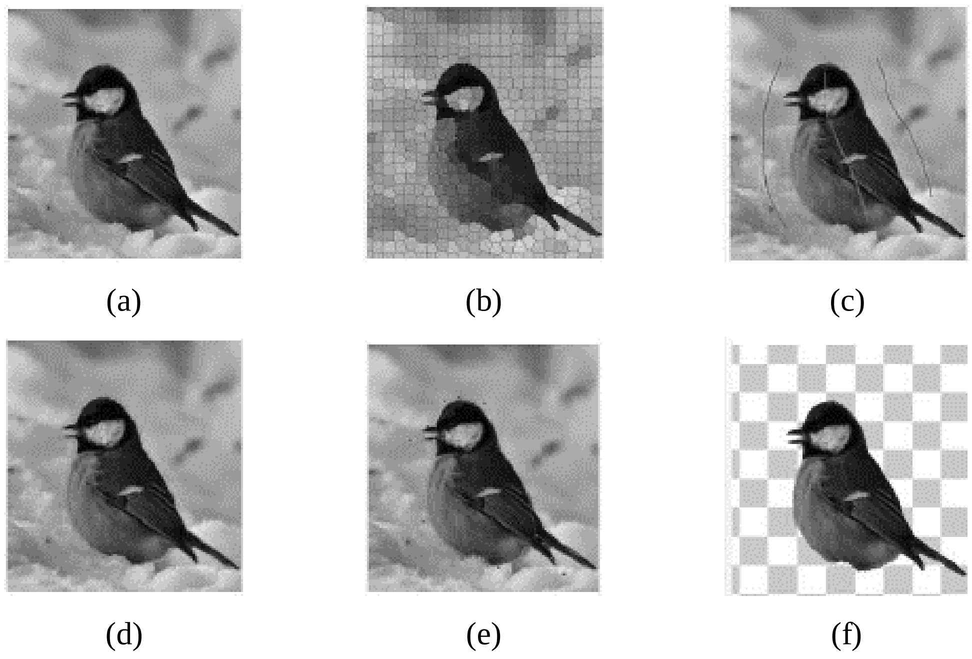


Figure 2.2.15: (a) Input image (b) Superpixel algorithm is applied (c) user mark the foreground object (d) initial graph-cut (e) segmentation is upsampled (f) final segmentation (Garcia *et.al.*, 2015)

## 2.3 Strengths of Reviewed Approaches

- Eliminates the need of interactive user input which is tedious when the data set is large.
- Exploit the spatial consistency, silhouette coherency to segment the object based on image from different views.
- High accuracy of segmentation.
- Some of the reviewed systems are capable of running in real time or on mobile devices which mean the algorithms are less computational complex and run faster.

## 2.4 Limitations of Reviewed Approaches

- In most of the reviewed system, the fixation point seems to be useful in initializing the segmentation but there exists some cases where it is insufficient to develop the colour models.
- Require images taken from different views which might not be feasible in real life application.

## 2.5 Existing Commercial Product Related to Current Work

### Sony 3D Creator

The latest Sony flagship smartphone now comes with a new application called 3D Creator. It allows the user to scan an object and construct a 3D model based on it. It works by scanning the object in a motion guided by the application then segment the object of interest and construct a 3D model of it.

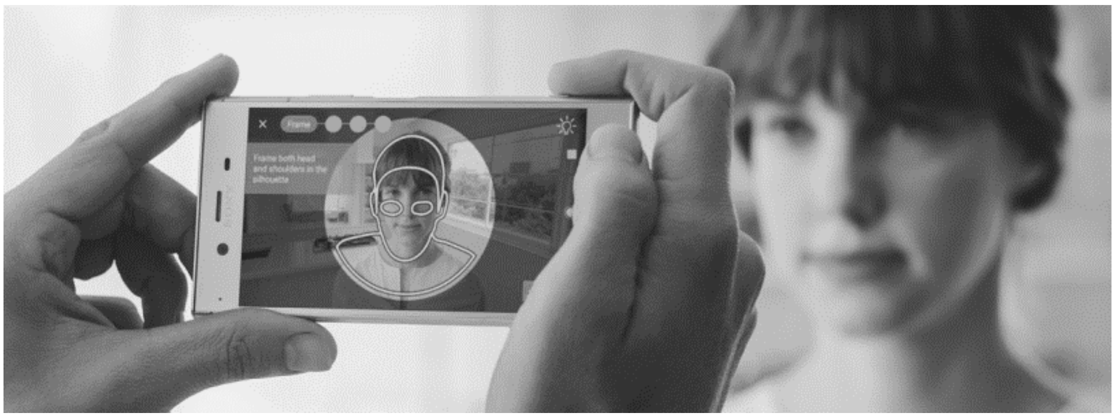


Figure 2.5.1: 3D Creator Scanning Process (n.d.)

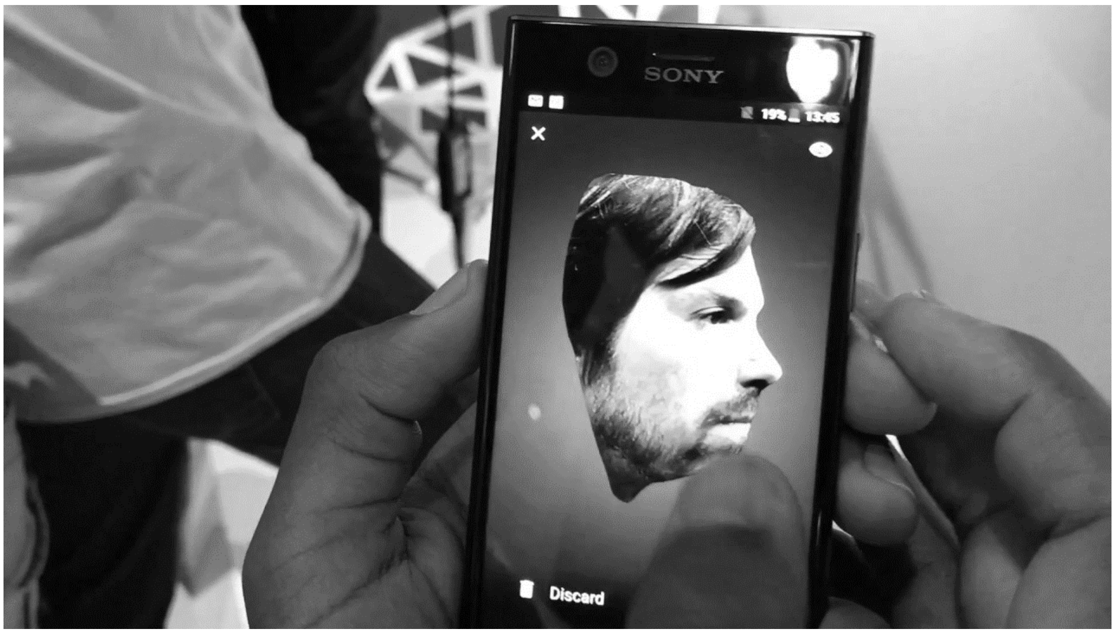


Figure 2.5.2: 3D Model Constructed through 3D Creator

## 2.6 Discussion

In the past researches regarding object segmentation, it is obvious that most of the approaches are not targeting real time tracking and segmentation probably due to the lack of processing power of mobile devices. However, with the increasing processing power in mobile device, it is capable of running more complex algorithm now.

In the reviewed existing approaches, the main technique that produces very satisfying results is Graph Cut segmentation. However, graph cut segmentation requires multiple iteration in order to produce good results. In terms of computational complexity, graph cut is a rather resource hungry algorithm. This algorithm also requires a foreground and background markers which makes it harder to produce good results. If it is an interactive system, users need to mark a foreground region explicitly.

In 2004, Carsten, Vladimir and Andrew introduced a new interactive foreground segmentation method using iterative graph cuts named GrabCut. This GrabCut algorithm outperformed the original Graph Cut segmentation in 3 aspect:

1. Requires minimal interactive user input (only require an object bounding box).
2. Allows the use of colour instead of monochrome model by using Gaussian Mixture Model (GMM).
3. Introduced an iterative algorithm that performs foreground estimation and parameter learning to construct complete foreground and background model.

# Chapter 3: System Design

## 3.1 System Design Overview

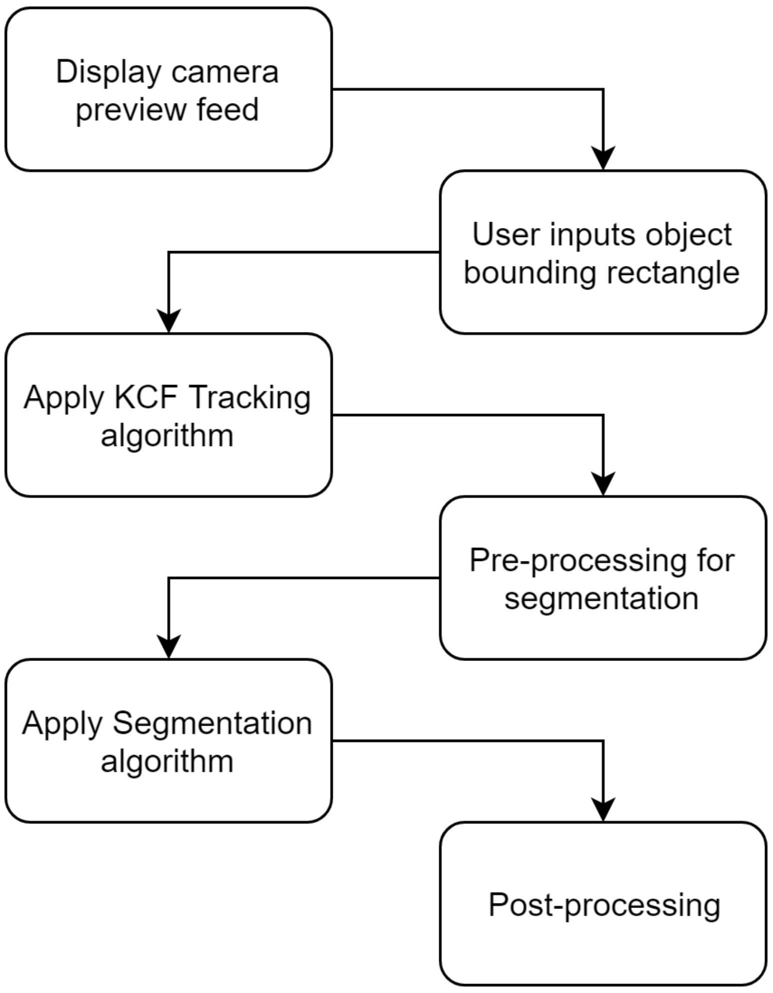

Display camera preview feed
User inputs object bounding rectangle
Apply KCF Tracking algorithm
Pre-processing for segmentation
Apply Segmentation algorithm
Post-processing

Figure 3.1.1: General overview system flowchart

**Algorithm 1 : Object Tracking and Segmentation**

```
Input: Real time camera input frames, $F_i$
Users' input object bounding rectangle, $R$
Intermediate: GrabCut segmentation mask, $M$
Output: Processed frames with object tracking rectangle and object boundary, $F_o$.
for each $F_i$ do
        Rescale the width, $w$ and height, $h$ of $F_i$
        $R_i = KCFTracking(F_i);$
        $Preprocess( F_i );$
        $M = GrabCutSegmentation(F_i, R_i );$
        $F_o = Postprocess(M);$
end for
```

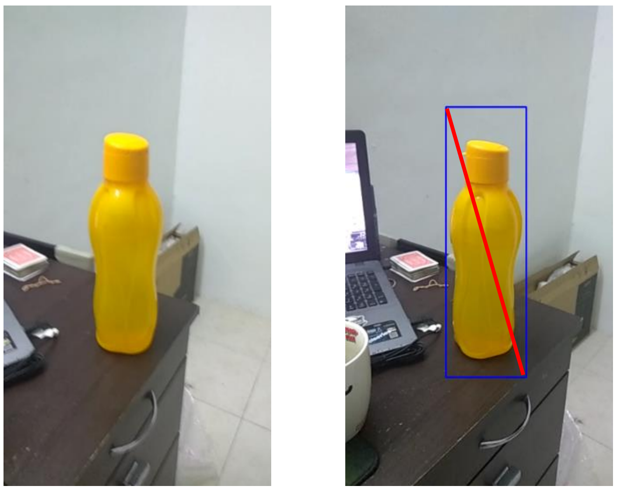
## 3.2 System Design Implementation

### 3.2.1 Display camera preview feed

At runtime, the application needs to display the camera real time preview frames and let user draw an object bounding rectangle. At UI level, camera view was implemented to display the camera frames. Since this system utilizes OpenCV library, it allows the main class to inherit `CameraBridgeViewBase.CvCameraViewListener2` interface provided by OpenCV. This interface allows the conversion of real time camera preview frames into Mat (matrix object) which can be feed directly into OpenCV related functions. By inheriting the interface, it introduced an important method `OnCameraFrame(inputFrame)` that accepts camera preview frames and returns frames to be displayed on the screen. This allows the frame processing to take place in between it. Before user input any object bounding rectangle, the `OnCameraFrame()` method will be returning original camera preview frames.

### 3.2.2 User input object bounding rectangle

When the user face the camera at the object of interest, he/she had to input a bounding rectangle by dragging from left top of the object to right bottom (or in a reverse manner) of the object (drawing a diagonal line as shown in figure 3.2.1(b)). Since the system had implement the `OnTouchListener` interface, it will listen for the touch event and trigger the method `onTouch()`. When this method is triggered, a rectangle object will be created to initialize the tracker.



(a) (b)

Figures 3.2.1: (a) Original frames (b) Processed frame after user inputted an object bounding rectangle

### 3.2.3 Apply KCF Tracking

After the user inputted a bounding rectangle and a rectangle object had been created, the KCF tracker will be initialized with it through `KCFReinitialize()` method. This method will takes in a parameter *gray* representing the grayscale frame which will be used for tracking purpose. Grayscale was chosen instead of RGB colour space because it brought significantly increase in frames-per-second (fps) without sacrificing too much accuracy. Then, for every frame, KCF tracking will be applied through `Tracking(gray,rgba)`. This will update the bounding rectangle values as long as the object is under tracking. If the object was lost tracked due to a large motion of camera, the user can usually re-track by just move the camera back to the last tracked position.

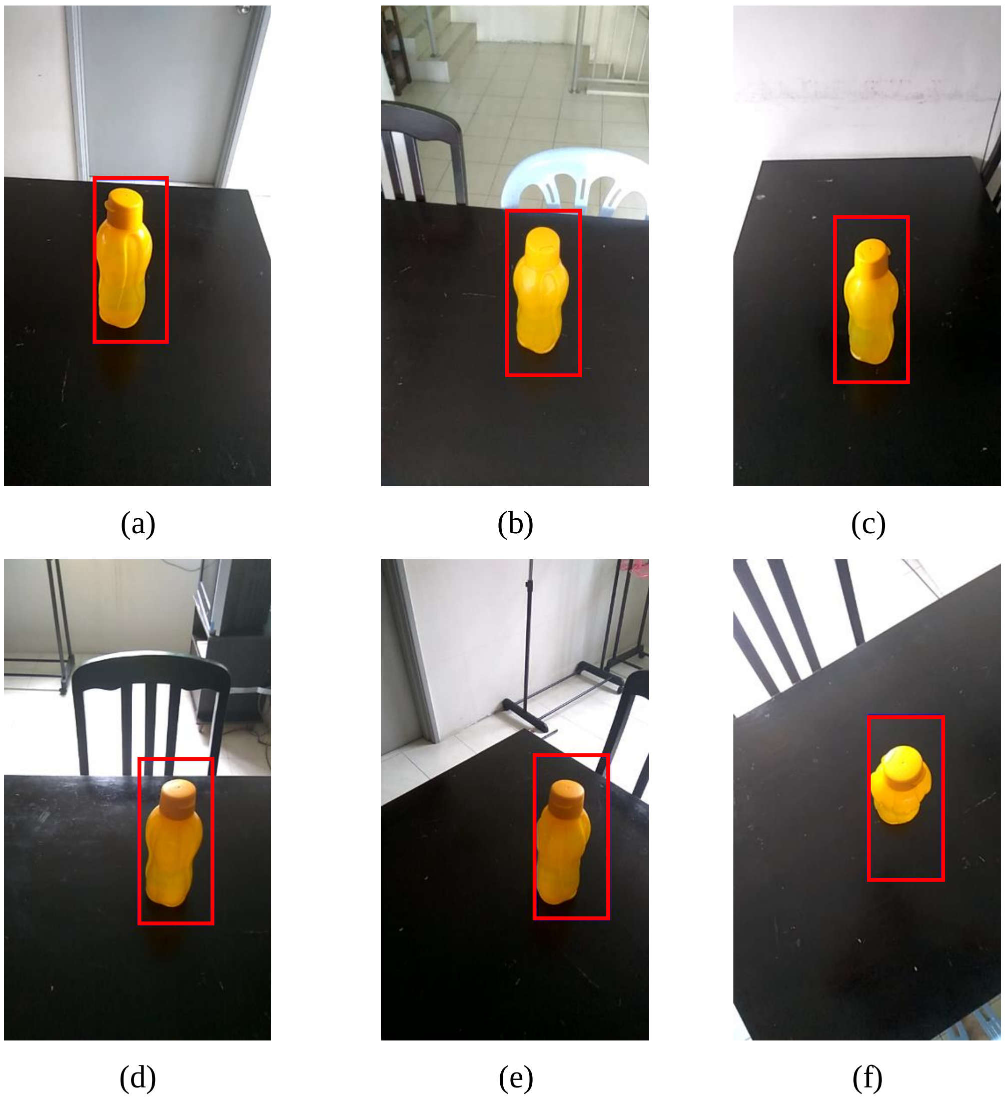


Figure 3.2.2: (a) to (f) Frames saved when moving around an object while tracking it.

Based on KCF tracking results, the system is able to keep track of the object of interest through the updated bounding rectangle. Since the object bounding is updated at every frame through KCF tracking, this eliminates the needs for extra interactive inputs.

By having this bounding rectangle, it can be safely assumed that all the pixels outside of the rectangle are sure background pixels. So, image segmentation can be done based on the object bounding rectangle to extract the object of interest from its background.

### 3.2.4 Pre-processing for segmentation

However, due to the limited processing power of a mobile device, the frame cannot be segmented directly because it will be too large and imposed too much workload for a real time system which will result in a very low fps. In order to maintain an acceptable performance, the frame need to under pre-processing before segmentation. So, this is where image pyramid comes into play. In `GrabCutSegmentation(gray)` method, the frames was downscaled (`pyrDown`) by original size/6 to reduce the number of pixels that will be processed during segmentation. This act will greatly improve the performance as the segmentation’s execution time is linear to the number of pixels being processed.

### 3.2.5 GrabCut Segmentation

After the pre-processing (down-scaling), GrabCut segmentation will take place. This is done by invoking `grabCut()`. The important parameters when invoking this method are the pre-processed frame, `downsampled`, the tracked object bounding rectangle, `rect`, and the mode of segmentation which in this case will be utilizing the rectangle, `GC_INIT_WITH_RECT`. This method will return a mask with labelled foreground and background pixels.

### 3.2.6 Post-processing

This mask is then undergo post-processing (upsampled back to original size using `pyrUp` and applied Gaussian blur to smooth out the boundary). Then, the object can be segmented by masking the segmentation mask with original frame which is shown in figure 3.2.3.

Afterwards, the result will be returned and displayed on screen. User can toggle between mask and result frame. The displayed frame can be saved by tapping the Save button which will invoke `SaveImage()` method.

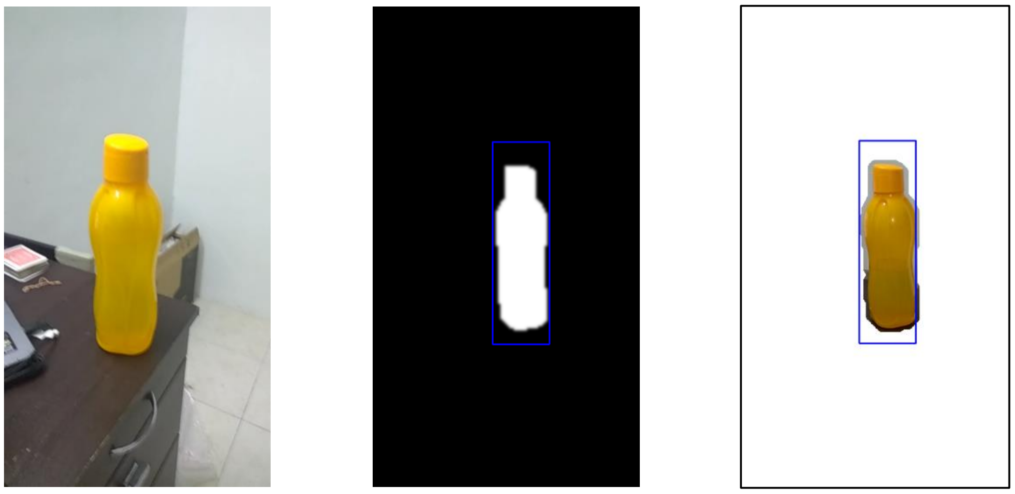

(a) (b) (c)

Figure 3.2.3: (a) Original frame (b) Mask return by grabcut segmentation (c) Result of masking (a) and (b) together to extract the object.

## 3.3 Important techniques and methods involved

### 3.3.1 KCF (Kernelized Correlation Filters) Tracking

In 2015, Henriques *et.al* introduced a KCF as a high-speed tracking algorithm. In the research, by proving that the data matrix is circulant, they managed to diagonalize it using the Discrete Fourier Transform. In this way, the storage and complexity is lessen by multiple orders of magnitude. Based on this, they managed to derive a new Kernelized Correlation Filter that outperforms state-of-the-art tracking algorithm by that time in terms of both accuracy and frames-per-second.

In this system, OpenCV variant of KCF Tracker is used. This variant of KCF is able to track on either RGB or grayscale frames. However, due to the computational power limit of mobile devices, the tracking is done based on grayscale frames to speed up the processing. Although grayscale frames have been used for tracking, it doesn't hurt the performance too much in terms of tracking precision. This will be further elaborate in the later chapter.

| **Colour space** | **Average frames per second (fps)** |
|---|---|
| RGB | 6.5 |
| Grayscale | 10.3 |

Table 3.3.1.1: Average fps achieved when using different colour space in KCF

```
Algorithm 2 : KCF Tracking
Input: Real time camera input frames, $F$
Users' input object bounding rectangle, $R$
Output: Updated rectangle, $R_i$ according to tracking result.
for each $F$ do
    if $firstTrack$ do
        Initialize KCF tracker using $R$;
    end if
    $R_i$ = KCFTracking( $F$ );
end for
```

### 3.3.2 GrabCut

In 2004, Carsten, Vladimir, and Andrew introduced GrabCut based on iterated graph cuts. This algorithm aims to overcome the limitation of graph cuts by introducing minimal user interaction.

Firstly, GrabCut requires user to input a rectangle which should include foreground region completely inside it. Pixels outside of the rectangle will be label as sure background then GMM will be used to construct the foreground and background model. Other pixels will be labelled as either possible foreground or background depending on the relationship between the pixels and the already labelled sure background or foreground pixels. Then, a graph will be generated according to the pixels distribution. The background pixels will be linked to the Sink node while the foreground pixels is linked to the Source node. Mincut algorithm will be adopted to separate the source node and sink node by using minimum cost function.

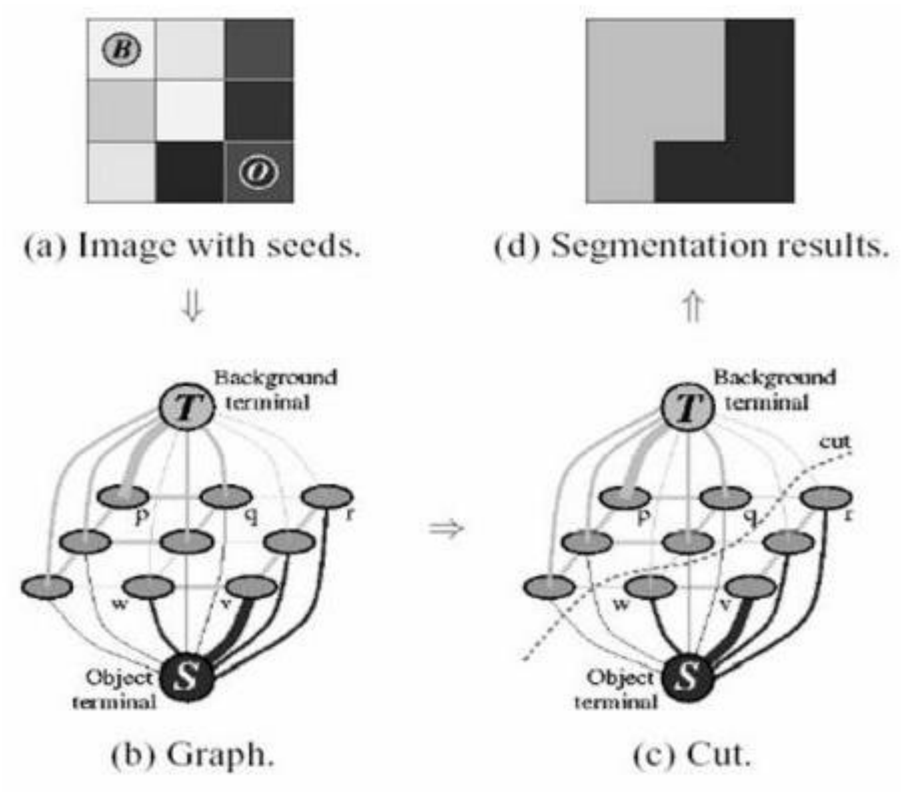


Figure 3.3.2.1: GrabCut illustration (Marsh, 2005)

### 3.3.3 Pre-processing and post-processing for segmentation

1. Image pyramid (upsample and downsample)

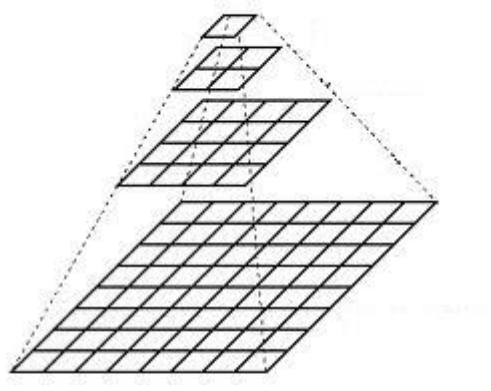


Figure 3.3.3.1: Image pyramid illustration (OpenCV)

Image pyramid was used to downsample the input frame for by the scale of 4. This had greatly improved the time needed for GrabCut segmentation.

After the segmentation, the obtained marker from GrabCut was upsampled back to original size and mask with the original frame to produce the output. Although the mask is upsampled from a much smaller size, the output is actually still quite accurate except the boundary is not perfectly segmented due to the details loss during image pyramid scaling.

2. Gaussian blur (to smooth the boundary)

As mentioned in the section before this, the boundary is not perfectly segmented due to the details loss during image pyramid scaling. The upsampled mask has a very blocky boundary and not smooth or fit to the object. Thus, Gaussian blur is applied on it to further smooth out the boundary to have a better looking output.

## 3.4 Functions and methods implementation

This system is built using Android Studio (Java) and OpenCV library.

Before implementing the system, this application requires permission for camera and internal storage. This will allows the application to access real time camera frames and write access to the storage.

```
<uses-feature android:name="android.hardware.camera" android:required="true" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

### 3.4.1 MainActivity Class

```
public class MainActivity extends FragmentActivity
        implements CameraBridgeViewBase.CvCameraViewListener2, OnTouchListener {
```

In order to utilize OpenCV function, the main class need to implements OpenCV's `CameraBridgeViewBase.CvCameraViewListener2`. This interface allows the class to override `OnCameraFrame()` method which convert the camera preview buffer into Mat object directly. Other than that, this class also implements `OnTouchListener` interface to capture touch event on the camera preview.

Class variables:

```
int         frameWidth, frameHeight, screenWidth, screenHeight;
int         frameCount = 1;
boolean     isRectCreated, isKcfInitialized, getMarker, getObject, isTracked;
Point       p1, p2;
Rect2d      trackRect;
Mat         matSave;
Button      MarkerButton, ObjectButton, SaveButton;
Tracker     tracker;
CameraBridgeViewBase mCameraview;
```

`frameWidth`, `frameHeight` are the width and height of the frames after rescaling according to phone specification while `screenWidth`, `screenHeight` are the width and height of the phone. These variables are used in the `OnTouch()` method to rescale the X and Y coordinates according to the rescaled frame's width and height.

`frameCount` records the number of frame after tracking is initialized. All the boolean variables act as toggle to trigger some event. Furthermore, *p1* and *p2* are two Point variable used to create a rectangle (`trackRect`) for tracking purpose.

In order to save a specific frame into internal storage, `matSave` will be utilized. Tracker object is created for tracking purpose. The button objects and `mCameraView` are the objects used to reference the UI layout objects.

### 3.4.2 OnCreate() Method

```
/* Initializes the preview and buttons
*  Capped the max frame size to ensure the frame size is around 896*504 (16:9) according
*  to device's specification */
@Override
protected void onCreate(@Nullable Bundle savedInstanceState)
```

During `OnCreate()`, all the UI layout objects such as *Buttons* and `CameraView` are referenced programmatically in order to utilize them. `setMaxFrameSize()` is invoked to cap the maximum frame size so that lesser computational power is needed when processing the frames.

### 3.4.3 onResume() and BaseLoaderCallBack()

```
/* To Load the OpenCV Library */
@Override
protected void onResume()
```

```
/* Load OpenCV Library through OpenCV Manager installed in phone */
private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback( AppContext: this)
```

In every android application, `onResume()` will be invoked after `onCreate()` or after `onPause()` so OpenCV library will be loaded in this method. When the application runs on real time, it will look for OpenCV library in the its application package first, if not found, it will invoked `BaseLoaderCallBack()` to look for OpenCV Manager on the phone as external library.

### 3.4.4 onCameraViewStarted()

```
/* Initialize important class variable */
@Override
public void onCameraViewStarted(int width, int height)
```

This method will be used to initialize the class variable such as `frameHeight`, `frameWidth`, and etc.

**Parameter:** `width` – The width of input camera frame.

`height` - The height of input camera frame.

### 3.4.5 onCameraFrame()

```
/* Retrieve rgb and grayscale frames
*  Initialize tracker for the first time
*  Start tracking if tracker has been initialized */
@Override
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame)
```

This method will be invoked when the frames need to be delivered for preview.

It returns the final results after segmentation to be displayed on camera view.

**Parameter:** `inputFrame` – The frame received from real time camera feed.

### 3.4.6 onTouch()

```
/* Listen for touch event to draw a bounding rectangle */
@Override
public boolean onTouch(View view, MotionEvent motionEvent)
```

This method will be invoked touch event is dispatched to the view.

**Parameter:** `view` – The View object that has touch event dispatched to it.

`motionEvent` – MotionEvent object containing information about the touch event.

### 3.4.7 SetOnClickListeners()

```
/* To set the buttons' listeners */
public void SetOnClickListeners()
```

This method will be invoked during `onCreate()` to defined the callback for each button.

### 3.4.8 KcfReinitialize()

```
/* To re-initialize the KCF tracker if new Touch event happened */
private void KcfReinitialize(Mat gray)
```

This method is invoked whenever touch event takes place and a new bounding rectangle was created. It will re-initialized the tracker by clearing the initial tracker content and create a new KCFTracker.

**Parameter:** `gray` – Grayscale input frame used to initialize the tracker.

### 3.4.9 Tracking()

```
/* Update the bounding rectangle through KCF tracking */
private Mat Tracking(Mat gray, Mat rgba)
```

This method is invoked after the tracker object is initialized. It will call `tracker.update()` to update the value of object bounding rectangle.
This method will return the processed frame obtained from `GrabCutSegmentation()`.

**Parameter:** `gray` – Grayscale input camera frame used during tracking update.
`rgba` – RGB input frame used during segmentation.

### 3.4.10 GrabCutSegmentation()

```
/* Apply GrabCut segmentation based on object tracking rectangle
*  Frame is downsampled before segmentation and upsampled afterwards */
private Mat GrabCutSegmentation(Mat rgba, Mat gray)
```

This method is responsible to pre-process the input frame, apply grabcut segmentation and apply post-processing obtain the result.
Returns either segmentation result mask or a result frame with segmented object.

**Parameter:** `rgba` – RGB input frame.

`gray` – Grayscale input frame.

### 3.4.11 SaveImage()

```
/* To save the result as image in internal storage */
public void SaveImage(Mat mat)
```

This method will be invoked when `SaveImage` button is pressed. This will saved the current on screen frame into internal storage.

**Paramter :** `mat` – The frame to be saved into internal storage.

## 3.5 Research Tools Involved

### 3.5.1 Software

- OpenCV master branch built with contrib modules (extra modules) for Android.
- Android Studio for developing the application.
- Visual Studio 2015 for testing and comparison.

### 3.5.2 Hardware

- <u>Laptop Specification</u>

| | |
|---|---|
| **Model** | Asus X550D |
| **Operating System** | Windows 10 Professional |
| **Processor** | AMD A10-5750M 2.50GHZ Quad Core |
| **Memory (RAM)** | 8GB |
| **Graphic Card** | AMD Radeon HD 8650G + HD 8670M Dual Graphic |

- 
- <u>Mobile Device</u>

| | |
|---|---|
| **Model** | Xiaomi Redmi Note 3 |
| **Operating System** | Android 7.1.2 |
| **Processor** | Snapdragon 650 |
| **Memory (RAM)** | 2GB |
| **Graphic Card** | Adreno 510 |

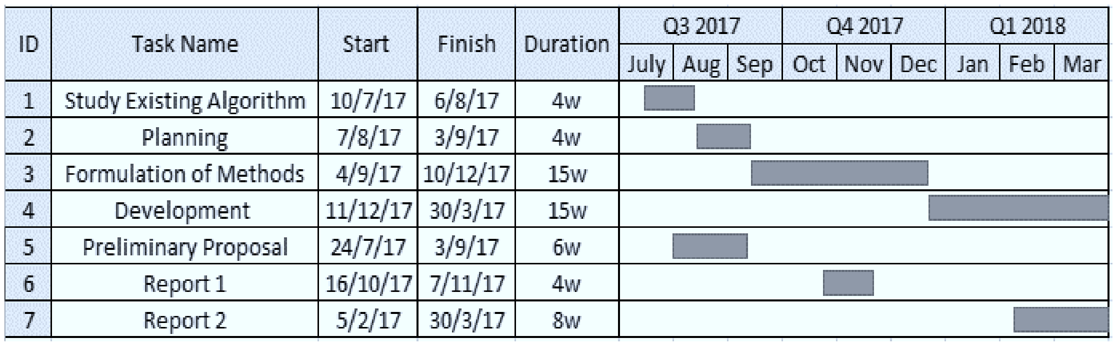
# Chapter 4: Implementation and Results

## 4.1 Implementation Issues and Challenges

During this project, several issues and difficulties are faced. First and foremost, the hardware limitation of mobile devices is the main difficulties. With the limited computational resource, the system needs to be more resource-friendly to avoid high memory consumption or extremely low frames-per-second. Other than that, some methods or libraries are not supporting Android platform yet which sometimes make the situation difficult.

## 4.2 Timeline

The project spans the duration of two and a half trimesters, which is approximately 38 weeks. Figure 3.4.1 shows the Gantt chart for the research project. There will be 3 reports submission during the project duration and a viva presentation to demonstrate the work done.



| ID | Task Name | Start | Finish | Duration |
|---|---|---|---|---|
| 1 | Study Existing Algorithm | 10/7/17 | 6/8/17 | 4w |
| 2 | Planning | 7/8/17 | 3/9/17 | 4w |
| 3 | Formulation of Methods | 4/9/17 | 10/12/17 | 15w |
| 4 | Development | 11/12/17 | 30/3/17 | 15w |
| 5 | Preliminary Proposal | 24/7/17 | 3/9/17 | 6w |
| 6 | Report 1 | 16/10/17 | 7/11/17 | 4w |
| 7 | Report 2 | 5/2/17 | 30/3/17 | 8w |

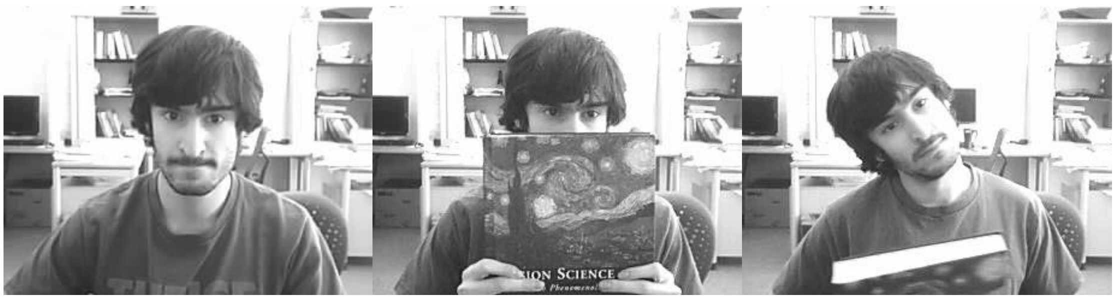
Figure 4.2.1: Gantt chart

## 4.3 Experimental Design and Comparison

### 4.3.1 Comparison between Multiple Tracking Algorithms

In this comparison and testing, a dataset “FaceOcc2” was downloaded from Visual Tracker Benchmark site, https://sites.google.com/site/trackerbenchmark/benchmarks/v10.

This dataset was chosen due to the simpler movement and partial occlusion in the image sequences. The testing was carried out using OpenCV and Visual Studio 2015, running in release mode x64. “FaceOcc2” dataset consists of 812 frames. This dataset is a 320*240 image sequences and tracked under grayscale colour space. Partial occlusion also happened in between some frames.



(a) (b) (c)

Figure 4.3.1.1: (a) frame #1 (b) frame #163 (c) frame #368 from “FaceOcc2” dataset

Along with the dataset, ground truth of the object bounding rectangle for each frame was provided. So, the precision of each tracking algorithm can be calculated by comparing the tracking rectangle’s x value, y value, width and height with the ground truth value provided by the dataset. An error margin of 20 pixels was considered during the calculation.

$$\text{Mean Precision}, P = \frac{R}{F}$$

$R =$ Number of correct rectangle (where $x, y, width, height$ values within ground truth value $\pm$ error margin)

$F =$ Number of total video frames

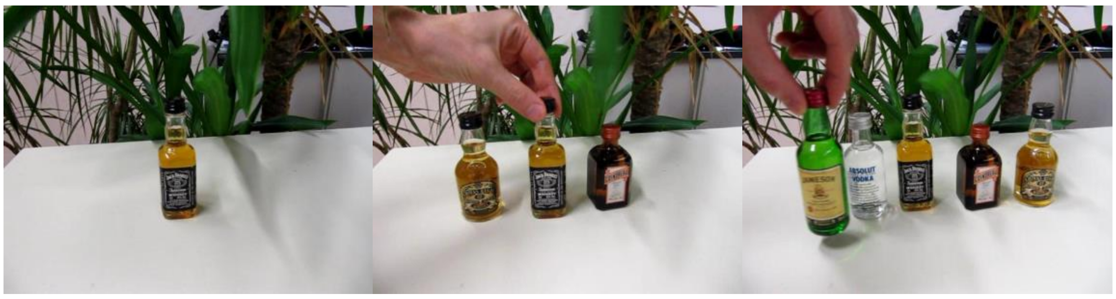
| **Tracking Algorithm** | **Execution Time (sec)** | | | | **Mean Precision (error margin 20px)** |
|---|---|---|---|---|---|
| | **1st** | **2nd** | **3rd** | **Average** | |
| KCF | 25.90 | 36.28 | 34.08 | 32.09 | 709/812 = 87.32% |
| TLD | 148.66 | 130.28 | 139.80 | 139.58 | 59/812 = 7.27% |
| Boosting | 164.38 | 172.42 | 158.31 | 165.04 | 599/812 = 73.77% |
| MIL | 362.82 | 369.89 | 354.20 | 362.30 | 593/812 = 73.03% |
| MedianFlow | 18.49 | 26.39 | 22.10 | 22.33 | 708/812 = 87.19% |

Table 4.3.1.1: Comparison between multiple tracking algorithms on dataset "FaceOcc2"

From table 4.3.1.1, it can be seen that KCF tracking algorithm has the highest tracking precision and relatively low execution time while MedianFlow gives a very similar precision to KCF and only consumes 30% less time in this testing set. The other algorithms (TLD, Boosting, and MIL) consumes too much time to process which is not feasible for a real time application. Despite the much longer execution time, these 3 algorithms still has a relatively low precision.

Looking back at KCF and MedianFlow, it might seems like these two algorithms are both competitive enough to be running on limited resources. However, MedianFlow algorithm can only works with scenario where the motion is very simple.

Another dataset was used in order to understand the capability of KCF and MedianFlow. This dataset "Liquor" is a 1741 frames, 640*480 image sequences. In this dataset, larger motion, more occlusion, and similar background will be involved.



(a) (b) (c)

Figure 4.3.1.2: (a) Frame #63 (b) Frame #308 (c) Frame #756 from "Liquor" dataset

| **Tracking Algorithm** | **Execution Time (sec)** | | | | **Mean Precision (error margin 20px)** |
|---|---|---|---|---|---|
| | **1st** | **2nd** | **3rd** | **Average** | |
| KCF | 110.22 | 87.95 | 104.04 | 100.74 | 1175/1741 = 67.49% |
| MedianFlow | 69.88 | 60.64 | 76.28 | 68.93 | 349/1741 = 20.04% |

Table 4.3.1.2: Comparison between KCF and MedianFlow tracking algorithms on dataset "Liquor"

By observing table 4.3.1.1, it is obvious that median flow is giving a very low precision which makes it not really useful. In the case of KCF, it is still showing some desired precision and the execution time taken is still acceptable compared to MedianFlow.

So, through this experiment, KCF is can be concluded as an overall winner in terms of execution time and tracking precision in different scenario and thus it is chosen as the tracking algorithm in this system.

### 4.3.2 Comparison between GrabCut and Watershed

In this section, a comparison between GrabCut and Watershed will be carried out and then explains why Grabcut has been chosen for this system. All the testing was implemented on mobile devices to have a better visualization in par with this system. Both GrabCut and Watershed was done based on KCF tracking results.

One important aspect of comparison between algorithms is the time taken to execute it. In a real-time application, an algorithm that takes longer time to execute usually results in a lower performance in terms of fps.

| Segmentation Algorithm | Average Execution Time (millisecond) |
|---|---|
| GrabCut | 130 |
| Watershed | 5 |

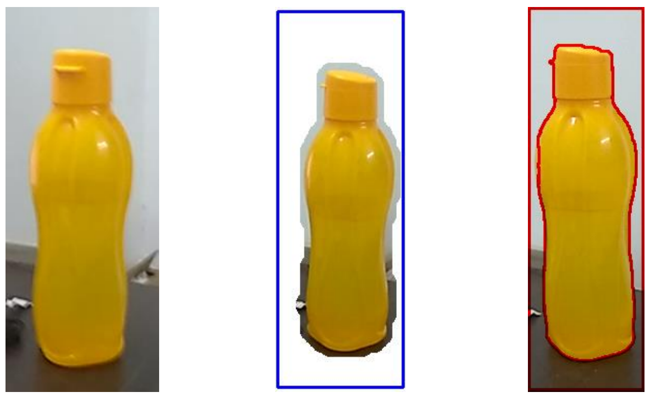
Table 4.3.2.1: Time taken to execute grabcut and watershed

From table 4.3.2.1, a dramatic difference can be seen between GrabCut and Watershed algorithm. Watershed only consumed around 5 seconds to segment an object while GrabCut is 26 times slower on average.

The reason behind this is because Watershed is a marker-based segmentation method which requires a good initial marker to guide the segmentation. With the help of markers and an object bounding rectangle, Watershed will initialize the segmentation from the labelled background and foreground markers until both of these markers meet a distinct difference which is the object boundary. In this testing, the Watershed is initialized with a cross marker in the middle of the bounding rectangle.

In the case of GrabCut, it initialize its segmentation solely based on the object bounding rectangle. All the pixels outside of the rectangle can be safely assumed as background pixels. Then, GrabCut will perform iterative foreground estimation and parameter learning to construct complete foreground and background model. It is much more complex than Watershed and requires a minimal amount of interactive input.

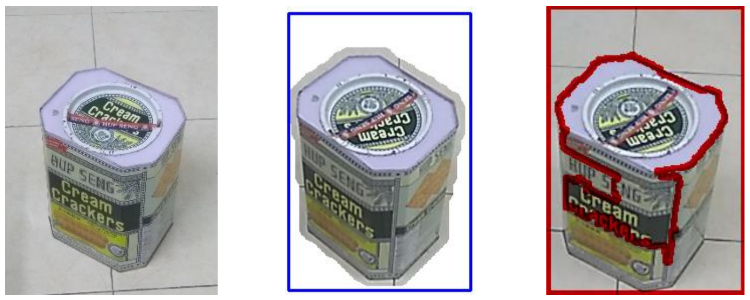
**Segmentation Result on Single Texture Object**



(a) (b) (c)

Figure 4.3.2.1: (a) Original image frame (b) Output of GrabCut segmentation (c) Output of Watershed segmentation on a bottle

**Segmentation Result on Multi Texture Object**



(a) (b) (c)

Figure 4.3.2.2: (a) Original image frame (b) Output of GrabCut segmentation (c) Output of Watershed segmentation on a cracker container

By looking at the results of both segmentation algorithm, it is obvious that when the object is a single texture (e.g. consistent colour) object, both the algorithm can performs really well. Watershed is preferable in this scenario due to its fast run time.

However, when observing the segmentation of multi-texture object, GrabCut is the clear winner. It managed to extract the object almost perfectly while Watershed failed to obtain the object boundary because the pre-defined marker is not suitable for a multi-texture scenario. Without extra interactive user input to define a new marker for each scenario, Watershed is unable to produce a satisfying result. So, GrabCut is much stable and better in terms of segmentation accuracy by sacrificing some performance. The longer execution time can be overcome by pre-processing the frame and reduce the number of pixels.

Thus, despite having a limited processing power on mobile device, GrabCut was still chosen as the segmentation technique due to its high accuracy.

## 4.4 Analysis of the complete system

### 4.4.1 Results

This section will demonstrate how this system performs under different scenarios.

**Scenario 1: Distinct background**

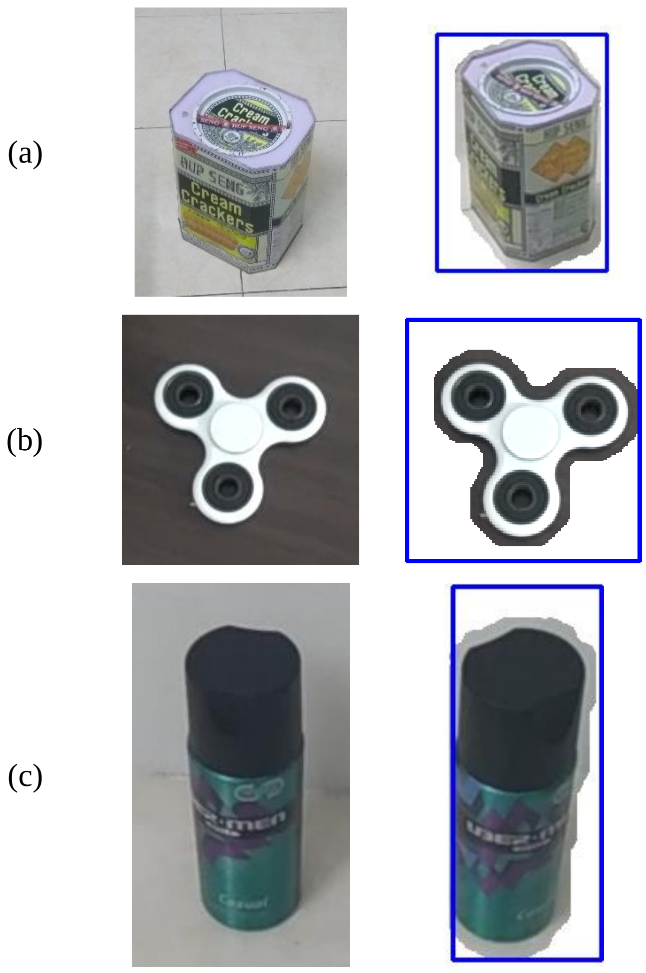


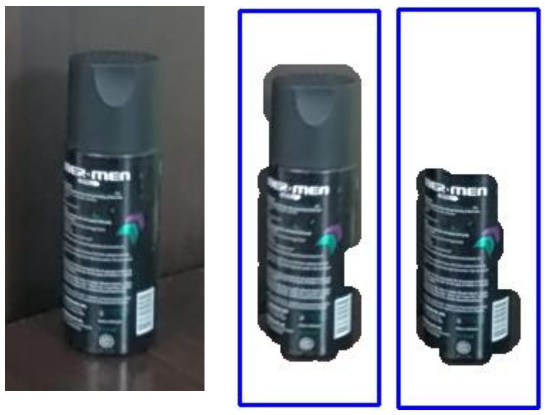
Figure 4.5.1: (a) A cracker container and output of the system

(b) A fidget spinner and output of the system

(c) A can sprayer and output of the system

**Scenario 2: Similar Colour Background**

**(a)**



**(b)**

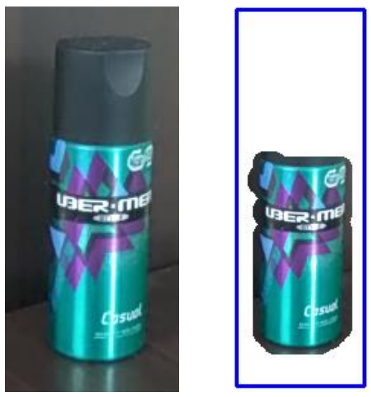


Figure 4.5.2: (a) Back side of a can sprayer and output of the system

(b) Front side of a can sprayer and output of the system

### 4.4.2 Result Analysis

In a distinct background scenario, this system is able to perform very well. In figure 4.5.1, it can be seen that the system was able to extract the object from its background successfully. Although the boundary is not perfectly segmented due to the detail loss during re-scaling of the image, the results produced were still satisfying.

However, in scenario where the background colour is similar to the object, the results might not be that convincing. In figure 4.5.2 (a), the segmentation is not stable due to the very similar background colour. The effect of similar background was more obvious in figure 4.5.2 (b). In the figure, only the part where the colour of the object differs from the background was segmented.

### 4.4.3 Strengths

1. This system was able to produce convincing and desired result in scenario where the background has a distinct colour from the object itself.
2. Minimal interactive user interaction is needed. Requires only an object bounding rectangle at the start.

### 4.4.4 Limitations

1. Performance in terms of FPS is acceptable but not very satisfying.
2. Unable to produce desired result if the background is complex or having similar texture (e.g. colour) with the object.

# Chapter 5: Conclusion

## 5.1 Project Review

Image segmentation has always been important to computer vision. In the past, lot of algorithms and applications have been developed to tackle this popular topic. However, despite having a lot of different available algorithms, most of it are actually not designed for mobile devices.

In this research study, a system which is capable of performing tracking and segmentation together in real time was proposed. This is done by analysing existing tracking and segmentation algorithm and chooses the suitable method for this system. In the end, KCF (Kernelized Correlation Filter) tracking algorithm and GrabCut was proved to be capable of producing desired result on mobile devices.

The final results produced was quite satisfying although the performance in terms of fps was considered acceptable but not perfect. This system can produce the best result in the scenario where the object of interest is a single-texture object (in terms of colour, shape or etc) and the background is simple and has a distinct colour from the object.

## 5.2 Future work

This research try to produce a real time image segmentation and tracking application. In order to keep it “real time”, the average frame per second has to be high enough so that the user won’t feel the delay caused by frame processing. Thus, further studies can be conducted to improve the execution time of the algorithms to increase the fps performance and delivers a better user experience.

Other than that, more studies can be done to improve the poor segmentation results produced under complex background.

## Bibliography

Campbell, N.D.F., Vogiatzis, G., Hernandez, C. and Cipolla, R., (2011), 'Automatic Object Segmentation from Calibrated Images', 2011 Conference for Visual Media Production, pp. 126-137.

Campbell, N.D.F., Vogiatzis, G., Hernández, C. and Cipolla, R., (2007), 'Automatic 3D object segmentation in multiple views using volumetric graph-cuts', Proceedings of the British Machine Conference, pp 58.1-58.10. BMVA Press, September 2007.

Garcia S., Kelley P.G., Yang Y., (2015), 'Fast Image Segmentation on Mobile Phone Using Multi-Level Graph Cut', GI '15 Proceedings of the 41st Graphics Interface Conference, pp. 81-88.

Hernandez, C., Schmitt, F. and Cipolla, R., (2007), 'Silhouette Coherence for Camera Calibration under Circular Motion', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.29, no. 2.

Holuša, M. and Sojka, E., (2012), 'Object Detection from Multiple Images Based on the Graph Cuts', Advances in Visual Computing. ISVC 2012. Lecture Notes in Computer Science, vol 7431, pp. 262-271.

Kootstra, G., Bergström, N. and Kragic, D., (2010) 'Fast and Automatic Detection and Segmentation of unknown objects', 2010 10th IEEE-RAS International Conference on Humanoid Robots, pp. 442-447.

Lee W., Woo W., Boyer E. (2007), 'Identifying Foreground from Multiple Images' Computer Vision – ACCV 2007. ACCV 2007. Lecture Notes in Computer Science, vol 4844, pp. 580-589.

Li, H., Meng, F. and Ngan, K.N. (2013), 'Co-Salient Object Detection From Multiple Images,' IEEE Transactions on Multimedia, vol. 15, no. 8, pp. 1896-1909.

Mallick, S, (2017), 'Object Tracking using OpenCV (C++/Python)', Available from: < https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/> [Accessed 25 February 2018]

Marsh, Matthew, (2005), Implementing the "GrabCut" Segmentation Technique as a Plugin for the GIMP. [image] Available from: http://www.cs.ru.ac.za/research/g02m1682/ [Accessed 18 March 2018]

**Bibliography**

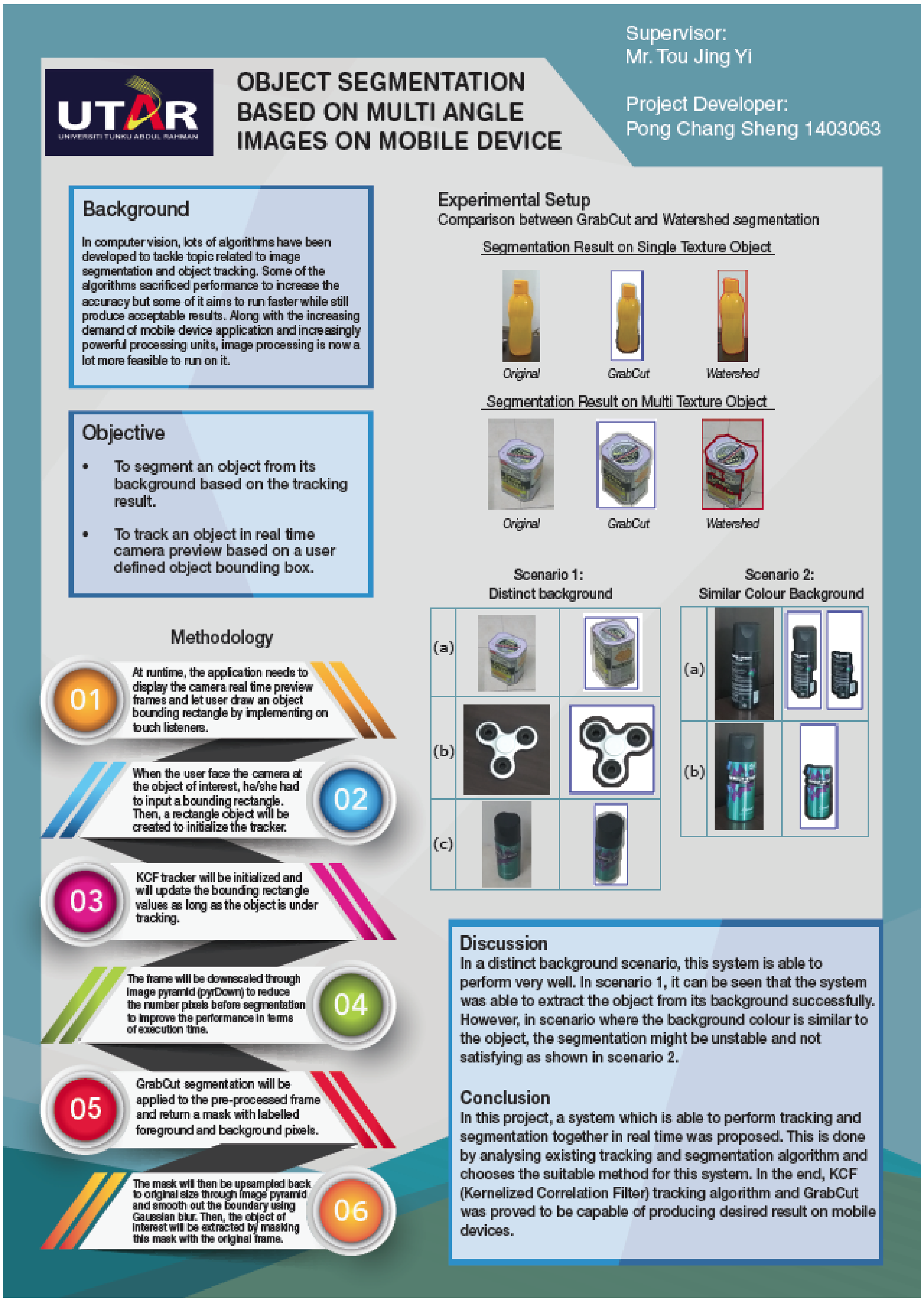
n.d., Superpixel: Empirical Studies and Applications. Available from: <http://ttic.uchicago.edu/~xren/research/superpixel/> [Accessed 17 August 2017]

OpenCV, Image Pyramids. Available from: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html> [Accessed 18 March 2018]

Visual Tracker Benchmark. Available from: < https://sites.google.com/site/trackerbenchmark/benchmarks/v10> [Accessed 18 March 2018]

Yang, Y., Zhang, Q., Wang, P., Hu, X., Wu, N., (2017), 'Moving Object Detection for Dynamic Background Based on Spatiotemporal Model', Advances in Multimedia, vol. 2017.

## <u>Poster</u>



Supervisor:
Mr. Tou Jing Yi

Project Developer:
Pong Chang Sheng 1403063

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

OBJECT SEGMENTATION BASED ON MULTI ANGLE IMAGES ON MOBILE DEVICE

**Background**

In computer vision, lots of algorithms have been developed to tackle topic related to image segmentation and object tracking. Some of the algorithms sacrificed performance to increase the accuracy but some of it aims to run faster while still produce acceptable results. Along with the increasing demand of mobile device application and increasingly powerful processing units, image processing is now a lot more feasible to run on it.

**Objective**

- To segment an object from its background based on the tracking result.
- To track an object in real time camera preview based on a user defined object bounding box.

**Methodology**

01 At runtime, the application needs to display the camera real time preview frames and let user draw an object bounding rectangle by implementing on touch listeners.

02 When the user face the camera at the object of interest, he/she had to input a bounding rectangle. Then, a rectangle object will be created to initialize the tracker.

03 KCF tracker will be initialized and will update the bounding rectangle values as long as the object is under tracking.

04 The frame will be downscaled through image pyramid (pyrDown) to reduce the number pixels before segmentation to improve the performance in terms of execution time.

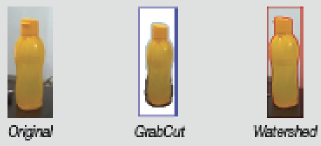
05 GrabCut segmentation will be applied to the pre-processed frame and return a mask with labelled foreground and background pixels.

06 The mask will then be upsampled back to original size through image pyramid and smooth out the boundary using Gaussian blur. Then, the object of interest will be extracted by masking this mask with the original frame.

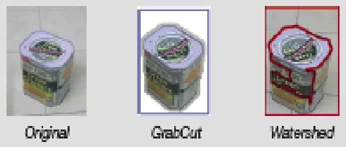
**Experimental Setup**

Comparison between GrabCut and Watershed segmentation

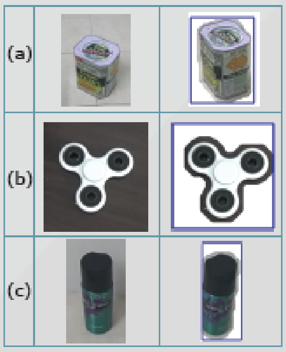
<u>Segmentation Result on Single Texture Object</u>



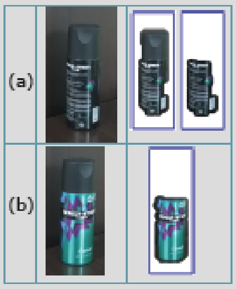
<u>Segmentation Result on Multi Texture Object</u>



Scenario 1:
Distinct background



Scenario 2:
Similar Colour Background



**Discussion**

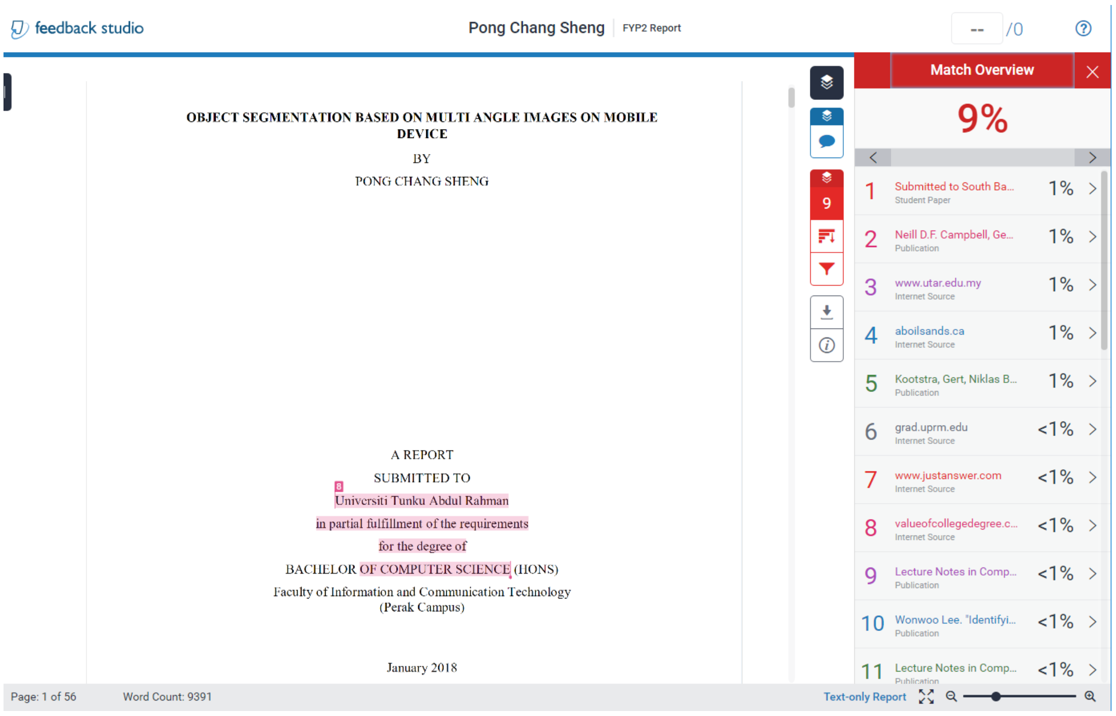
In a distinct background scenario, this system is able to perform very well. In scenario 1, it can be seen that the system was able to extract the object from its background successfully. However, in scenario where the background colour is similar to the object, the segmentation might be unstable and not satisfying as shown in scenario 2.

**Conclusion**

In this project, a system which is able to perform tracking and segmentation together in real time was proposed. This is done by analysing existing tracking and segmentation algorithm and chooses the suitable method for this system. In the end, KCF (Kernelized Correlation Filter) tracking algorithm and GrabCut was proved to be capable of producing desired result on mobile devices.

**Plagiarism Check Result**

## Plagiarism Check Result



feedback studio — Pong Chang Sheng | FYP2 Report

OBJECT SEGMENTATION BASED ON MULTI ANGLE IMAGES ON MOBILE DEVICE
BY
PONG CHANG SHENG

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONS)
Faculty of Information and Communication Technology
(Perak Campus)

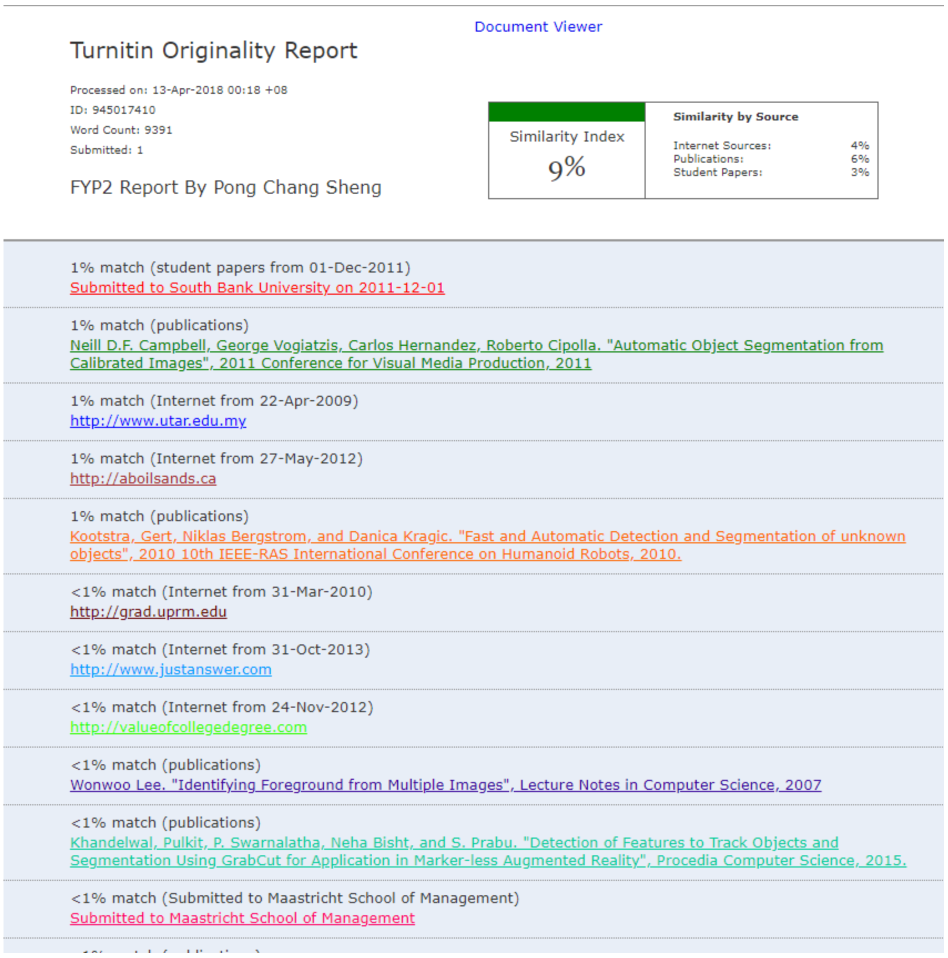
January 2018

Page: 1 of 56 Word Count: 9391

Match Overview

9%

| | | |
|---|---|---|
| 1 | Submitted to South Ba... Student Paper | 1% |
| 2 | Neill D.F. Campbell, Ge... Publication | 1% |
| 3 | www.utar.edu.my Internet Source | 1% |
| 4 | aboilsands.ca Internet Source | 1% |
| 5 | Kootstra, Gert, Niklas B... Publication | 1% |
| 6 | grad.uprm.edu Internet Source | <1% |
| 7 | www.justanswer.com Internet Source | <1% |
| 8 | valueofcollegedegree.c... Internet Source | <1% |
| 9 | Lecture Notes in Comp... Publication | <1% |
| 10 | Wonwoo Lee. "Identifyi... Publication | <1% |
| 11 | Lecture Notes in Comp... Publication | <1% |



Document Viewer

Turnitin Originality Report

Processed on: 13-Apr-2018 00:18 +08
ID: 945017410
Word Count: 9391
Submitted: 1

FYP2 Report By Pong Chang Sheng

| Similarity Index | Similarity by Source | |
|---|---|---|
| 9% | Internet Sources: | 4% |
| | Publications: | 6% |
| | Student Papers: | 3% |

1% match (student papers from 01-Dec-2011)
Submitted to South Bank University on 2011-12-01

1% match (publications)
Neill D.F. Campbell, George Vogiatzis, Carlos Hernandez, Roberto Cipolla. "Automatic Object Segmentation from Calibrated Images", 2011 Conference for Visual Media Production, 2011

1% match (Internet from 22-Apr-2009)
http://www.utar.edu.my

1% match (Internet from 27-May-2012)
http://aboilsands.ca

1% match (publications)
Kootstra, Gert, Niklas Bergstrom, and Danica Kragic. "Fast and Automatic Detection and Segmentation of unknown objects", 2010 10th IEEE-RAS International Conference on Humanoid Robots, 2010.

<1% match (Internet from 31-Mar-2010)
http://grad.uprm.edu

<1% match (Internet from 31-Oct-2013)
http://www.justanswer.com

<1% match (Internet from 24-Nov-2012)
http://valueofcollegedegree.com

<1% match (publications)
Wonwoo Lee. "Identifying Foreground from Multiple Images", Lecture Notes in Computer Science, 2007

<1% match (publications)
Khandelwal, Pulkit, P. Swarnalatha, Neha Bisht, and S. Prabu. "Detection of Features to Track Objects and Segmentation Using GrabCut for Application in Marker-less Augmented Reality", Procedia Computer Science, 2015.

<1% match (Submitted to Maastricht School of Management)
Submitted to Maastricht School of Management

**Plagiarism Check Result**

| | | | |
|---|---|---|---|
| **Universiti Tunku Abdul Rahman** | | | |
| **Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)** | | | |
| Form Number: FM-IAD-005 | Rev No.: 0 | Effective Date: 01/10/2013 | Page No.: 1of 1 |

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

| | |
|---|---|
| **Full Name(s) of Candidate(s)** | |
| **ID Number(s)** | |
| **Programme / Course** | |
| **Title of Final Year Project** | |

| **Similarity** | **Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)** |
|---|---|
| **Overall similarity index:________ %**<br><br>**Similarity by source**<br>Internet Sources: _____________%<br>Publications: _____________%<br>Student Papers: _____________% | |
| **Number of individual sources listed** of more than 3% similarity: _________ | |
| **Parameters of originality required and limits approved by UTAR are as Follows:**<br>**(i) Overall similarity index is 20% and below, and**<br>**(ii) Matching of individual sources listed must be less than 3% each, and**<br>**(iii) Matching texts in continuous block must not exceed 8 words**<br>***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.*** | |

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

***Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.***

| | |
|---|---|
| __________________________ | __________________________ |
| Signature of Supervisor | Signature of Co-Supervisor |
| Name: ____________________ | Name: ____________________ |
| Date: ____________________ | Date: ____________________ |

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (PERAK CAMPUS)

## CHECKLIST FOR FYP2 THESIS SUBMISSION

| | |
|---|---|
| Student Id | |
| Student Name | |
| Supervisor Name | |

| TICK (√) | DOCUMENT ITEMS<br>Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item. |
|---|---|
| | Front Cover |
| | Signed Report Status Declaration Form |
| | Title Page |
| | Signed form of the Declaration of Originality |
| | Acknowledgement |
| | Abstract |
| | Table of Contents |
| | List of Figures (if applicable) |
| | List of Tables (if applicable) |
| | List of Symbols (if applicable) |
| | List of Abbreviations (if applicable) |
| | Chapters / Content |
| | Bibliography (or References) |
| | All references in bibliography are cited in the thesis, especially in the chapter of literature review |
| | Appendices (if applicable) |
| | Poster |
| | Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005) |

| | |
|---|---|
| I, the author, have checked and confirmed all the items listed in the table are included in my report.<br><br>______________________<br>(Signature of Student)<br>Date: | Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.<br><br>______________________<br>(Signature of Supervisor)<br>Date: |