REAL-TIME ROBOT NAVIGATION DEVELOPMENT BASED ON GEOMETRIC PROPERTY

TEERA SIRITEERAKUL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
DOCTOR OF ENGINEERING IN ELECTRICAL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2017

KMITL-2017-EN-D-018-008

การพัฒนาระบบนำทางหุ่นยนต์แบบเวลาจริง
โดยการตรวจสอบคุณสมบัติเชิงเรขาคณิต

REAL-TIME ROBOT NAVIGATION DEVELOPMENT
BASED ON GEOMETRIC PROPERTY

ธีระ ศิริธีรากุล
TEERA SIRITEERAKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2560

KMITL-2017-EN-D-018-008

หัวข้อวิทยานิพนธ์	การพัฒนาระบบนำทางหุ่นยนต์แบบเวลาจริงโดยการตรวจสอบคุณสมบัติเชิงเรขาคณิต
นักศึกษา	นายธีระ ศิริธีรากุล
รหัสประจำตัว	57601049
ปริญญา	วิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2560
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ดร.รัชนี กุลยานนท์

บทคัดย่อ

การพัฒนาระบบนำทางให้กับหุ่นยนต์เคลื่อนที่เป็นเรื่องที่เป็นประโยชน์และมีความท้าทายเป็นอย่างยิ่ง ระบบนี้ สามารถนำไปใช้ได้ในงานที่หลากหลาย โดยอาจจะเป็นการนำไปใช้ในรถที่ขับเคลื่อนโดยอัตโนมัติ จนถึงการควบคุมอากาศยานไร้คนขับ อีกด้านหนึ่ง ความก้าวหน้าของงานวิจัยสาขาคอมพิวเตอร์วิทัศน์ ทำให้เราสามารถนำทางและควบคุมหุ่นยนต์เคลื่อนที่ได้อย่างอัตโนมัติโดยใช้ข้อมูลจากภาพ อย่างไรก็ตาม ส่วนใหญ่แล้ววิธีการเหล่านี้จะไม่สามารถนำมาใช้กับหุ่นยนต์เคลื่อนที่ได้ เนื่องจากหุ่นยนต์เคลื่อนที่ส่วนใหญ่มีความสามารถในการคำนวนต่ำ และมีพลังงานสำรองน้อยอีกด้วย ด้วยเหตุนี้ งานวิจัยชิ้นนี้นำเสนอวิธีการ รวมทั้งกรอบสัญลักษณ์สำหรับติดตามที่ถูกออกแบบมาพิเศษ เพื่อใช้ในการตรวจจับอย่างรวดเร็วและถูกต้อง โดยวิธีการที่นำเสนอจะเริ่มจากการตรวจจับองค์ประกอบในภาพที่น่าจะเป็นองค์ประกอบของกรอบสัญลักษณ์ และใช้สมบัติเชิงเรขาคณิตของกรอบสัญลักษณ์มาเพื่อใช้ตัดองค์ประกอบที่ไม่น่าจะใช่องค์ประกอบของกรอบสัญลักณ์ออกไป ซึ่งหลังจากตรวจจับสัญลักษณ์ได้แล้ว ก็จะสามารถคำนวณหาระยะหางและมุมที่จะหันไปหากรอบสัญลักษณ์นั้นได้ โดยกระบวนการนี้ได้ถูกทดลองบนเครื่องราสเบอร์รี่พายและได้แสดงให้เห็นว่า กระบวนการนี้สามารถทำงานได้รวดเร็วและถูกต้องในสภาพแวดล้อมที่มีแสงแตกต่างที่แตกต่างกันมาก และภาพที่มีฉากหลังมีความซับซ้อนสูง

Thesis	Real-time Robot Navigation Development Base on Geometric Property
Student	Mr.Teera Siriteerakul
Student ID.	57601049
Degree	Doctor of Engineering
Program	Electrical Engineering
Year	2017
Thesis Advisor	Dr.Rutchanee Gullayanon

ABSTRACT

Developing a mobile robot navigation is a useful and challenging task. The applications vary from autonomous driving to Unmanned Aerial Vehicle controlling. On the other hand, with recent advances in Computer Vision field, it is possible to develop a mobile robot which can autonomously navigate around to perform a certain task using visual information. However, most of the current visual tracking methods will not perform well under mobile robots with limited computing and battery power. Thus, this thesis provides a framework, together with a designed marker, for fast and accurate tracking. The proposed framework starts by detecting a lot of candidates and then make use of the known geometric information of the designed marker to quickly filter out any non-marker element. Then, with the detected marker, distance and turn angle between the attached camera and the marker can be calculated. This framework has been tested on a low computing unit and has demonstrated promising speed and accuracy in the scenes with various lighting and complicated background.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ดร.รัชนี กุลยานนท์ ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ รศ.ดร.สุรพันธ์ ยิ้มมั่น ประธานกรรมการ รศ.ดร.ปิติเขต สู้รักษา รศ.ดร.สุรพันธุ์ เอื้อไพบูลย์ ดร.วันวิสา ชัชวงษ์ กรรมการสอบวิทยานิพนธ์ ที่ได้กรุณาให้คำแนะนำตลอดจนข้อชี้แนะ จนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบพระคุณ ศ.ดร.วันชัย ริ้วรุจา ที่คอยให้คำแนะนำ ข้อเสนอแนะ ตลอดจนให้ความช่วยเหลือในด้านต่างๆ จนทำให้การศึกษาของข้าพเจ้าสำเร็จลงได้

ขอขอบคุณ อ.ศิริกุล ศิริธีรากุล ที่คอยช่วยเหลือ ลงแรง และให้กำลังใจ จนทำให้การศึกษาและวิทยานิพนธ์ฉบับนี้สำเร็จลงได้

สำหรับคุณงามความดีอันใด ที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดาผู้ล่วงลับ ตลอดจนครูอาจารย์ที่เคารพทุกท่าน ที่ได้ประสิทประสาทวิชาความรู้ และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

ธีระ ศิริธีรากุล

Table of Contents

Table of Contents (cont.)

List of Tables

List of Figures

List of Figures (cont.)

Chapter 1

Introduction

1.1 Problem Statements

Guiding mobile robot to navigate around using visual information is one of the attractive and challenging fields for research. With this navigation system, we can made a lot of systems possible including a system for autonomous driving vehicle [1], a system which can control Unmanned Aerial Vehicle (UAV) automatically [2], or a simple system which can carries our belonging and follow us around [3]. Although many of the system utilized various sensors to navigate around, the visual sensor (a camera) has become more popular in recent years.

Using visual information for guiding system can be challenging due to numerous factors. To start with, trying to recover the real world information from image is an ill-posed problem. To do so, we have to recover 3D information from 2D data which is similar to trying to find values of three variables using only two equation. Additionally, there are enormous noises, such as change in lighting or embedded noise from camera, and missing information, such as change in perspective or occlusion, in the way we capture visual information. Moreover, to be able to "see" the world as good as human, we have to rely on pass experience which is hard to formulate (see [4] for visual information processing example).

Before getting into detail, let us discuss overview of the mobile robot navigation system. The navigation of mobile robot can be view as art and science of how to travel through the environment [5]. There are three problems that can briefly define navigation system [6]: "Where am I?", "Where am I going", and "How do I get there?". The first question typically answered by Simultaneous localization and Mapping (SLAM) (detail in [7]) technique. However, the latter two questions are of a sufficient challenge when it comes to a dynamic scene.

The main target of this work is to develop a navigation system of a mobile robot for following a designated target in a dynamic scene. Since the main target is for a dynamic

scene, the latter two question ("Where am I going" and "How do I get there?") becomes more important. The target of our mobile robot is dynamically changing in both location can appearance. Additionally, the dynamically changing is not only apply the target, but also apply to the environment as well. The scene may have occlusion, dynamic background, and change in lighting. All of which add to the challenge in developing a navigation system.

Real-time robot navigation development based on geometric property filtering is our proposed system which is a mobile robot navigation system based on visual tracking system. In our scheme, we utilized a designated marker which is designed for easy tracking. There are two approaches explored in this work: One based on Hough Circle Transform and the other based on Connected Component Detection. In this framework, black circles which are components of our marker are detected as fast and as much as possible (high recall) and then we use known geometric properties of the marker to quickly filter out fault positive circles and keep only the correct ones at the end. Finally, with the detected marker, the real world position information, relative angle and distance, can be calculated with acceptable speed and accuracy.

1.2 Research Goal

This thesis is to develop and discuss an approach to mobile robot navigation algorithm to follow a marked object or person. The main objectives of this thesis are:

1. Develop an algorithm for to mobile robot navigation which can follow a designate object or person.
2. The algorithm developed must be robust to dynamic environment such as changing in light and clustered scene.
3. The algorithm developed must be performed in an acceptable speed and accuracy in a low-computing power setup.

1.3 Hypothesis and Contribution

We suspect that, by usage of the designed marker, we can develop an algorithm that is robust to light change and can perform in a mobile robot with low computing power in an acceptable speed and accuracy. The proposed framework is based on detecting a black and white marker. Since the color in the marker is pure black and white and we only detect its edges information, the change in global lighting should not degrade our algorithm. Although, the local lighting may still effect the detection process, it usually effects most of the algorithm as well as human perception. Thus, we consider the local lighting to be outside of our scope for now.

Furthermore, by using specially designed marker, we can make use of its chromatic and geometric properties to help with fast detection. As described, this approach only detect shape information (circles). We can relax some parameter of the process to make the detection faster with more false positive elements (circles). Then, we can filter those false positives by simply test if they follow our designed marker geometric properties. Since geometric calculation can be done fast in a computing unit, comparing to image processing task, we believe that the overall system can perform in an acceptable speed.

Another significant aspect of this research is how we incorporate both image processing based on geometric filtering and machine learning to help with marker detection. To classify circle/non-circle, we applied a framework based on Support Vector Machine and k-Nearest Neighbor. This framework improves the detection accuracy of the system without noticeable speed penalty.

The main contribution of this work is an approach that can be adopt for building a navigation system for a mobile robot to follow a designate target. This approach is robust to light change, clustered environment, and work in an acceptable speed. The navigation system developed using this approach can be employed on a robot with low computing unit (as a result, low power consumption).

1.4 Scope of The Study

1. To develop a navigation system for a mobile robot to follow a designate target.
2. To demonstrate that the system provides a sufficient accuracy.
3. To demonstrate that the system can function in a clustered environment and varying light.
4. To demonstrate that the system perform in an acceptable speed.
5. The algorithm is develop and test on selected hardware describe in the section

1.5 Outline of dissertation

This chapter describes thesis problem and rational behind it. The motivation to our approach, as well as contributions of this work are also discussed. Furthermore, the research objects and hypothesis, which describes this work are also provided in this chapter. The remaining chapters are organized as follows:

Chapter 2 describes relevant research and related theories used in this dissertation. The chapter starts by describing the robot navigation field as well as the current state of the art research in the field. The closely related researches are also discussed in the chapter. Then, the later part of this chapter provides related theories and techniques to be used in developing of our approach.

Chapter 3 presents our approach to mobile robot navigation. The chapter starts by describing the hardware setup for a mobile robot to be used in the experiments in the later chapter. Then, this chapter describes the designed marker, how to detect it, and how to use the detected information in the navigation system.

Chapter 4 describe experiments which we demonstrate our claims. The system accuracy, speed, as well as how to calibrate the necessary parameters for a particular mobile robot.

Chapter 5 summarizes our approach and review advantage and limitation of our work as demonstrated in Chapter 4. Then, the chapter goes on and discuss some suggested direction on how to make use of this finding as well as some probable ways to improve this work.

Chapter 2

Literature Reviews

In this Chapter, we will discuss the current approaches to mobile robot navigation, as well as provide some basic theories for developing our approach in the next chapter. The section will started by giving an overview of robot navigation system. Then, each theory and technique used in this work will be described in details.

2.1 Review of Robot Navigation System

With advances in computer vision, it is now plausible to develop mobile robots which can interact with environment based on visual cue. The examples of mobile robots are ranged from Mars rover Curiosity to floor cleaning robot Roomba (see Figure 2.1). While the Mars rover utilizes Navcam, a navigational camera, and other visual sensors and algorithms, the floor cleaning Roomba, on the other hand, only just added a low resolution camera for visual SLAM (see [8] for visual SLAM review) on the recently announced, top of the line model.

Figure 2.1 Mars rover Curiosity and the latest floor cleaning robot Roomba® 980

The visual information can be utilized in three main tasks of mobile robot navigation: 1) localization and mapping, 2) path planning and following, and 3) person or object tracking and following. Before computer vision, the robot localization and mapping

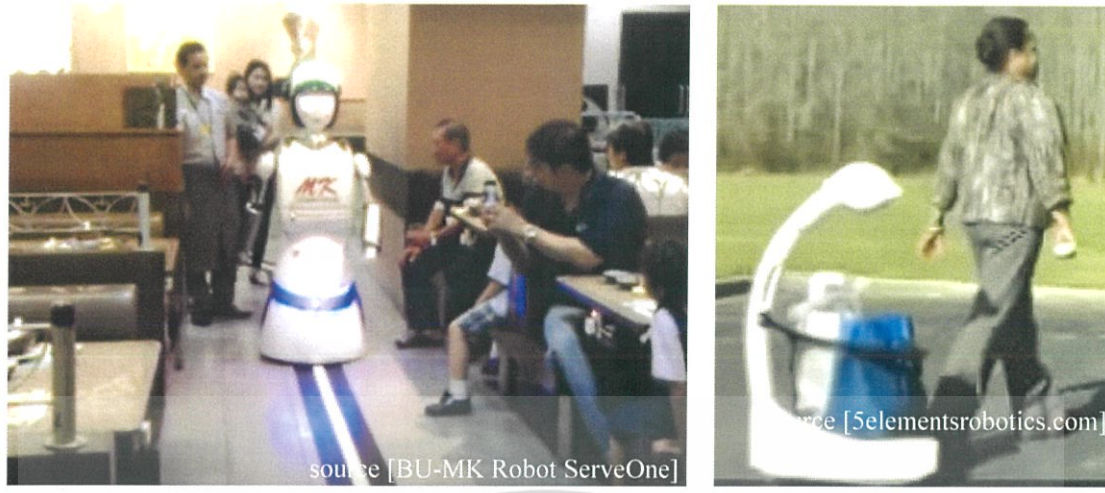


Figure 2.2 Black tape following robot BU-MK Robot (left) and signal following (produce from a device from the person right hand) Budgee™ (right)

is typically done by using laser range finders or sonar transducers which can be costly or limited accuracy. The path planning and following task can be done by pre-set programs and some markers (such as black tape on the floor as in BU-MK Robot – see Figure 2.2 left). However, the task of person or object tracking cannot be done unless there are some kinds of signal produced by the object of interest (still in used today as in a commercial robot called Budgee™ for example – see Figure 2.2 right).

In navigation, visual cue can be used for localization, building a map using SLAM technique, assisting the robot to follow a preset path, or helping to follow an object while avoiding obstacles. To find the location of itself on a map using visual cue, [9] used Abstracted Scene Representation (ASR) extracted from images in eight fixed direction around the robot while [10] used stored panoramic images around the robot as visual cue. On the other hand [11] [12] utilized direction and distance of recognized landmarks as the main information for their artificial neuron network system for localization. Another approach on localization problem based on triangulation algorithm on three detected landmarks was provided by [13]. Later, [14] developed another triangulation algorithm to better select landmarks to avoid the problem where landmarks are close together.

In simultaneous localization and mapping (SLAM) [15] mentioned in [16], work by [17] stated that the types of sensors have been a major driver of new algorithms. With the

recent advance in Computer Vision techniques as well as the 3D cameras, there has been surge in research in a visual SLAM over the past decade. Work by [18] provided a single-camera based approach using traditional SLAM while later [19] shifted to FastSLAM for more computationally efficient. Depth information from camera can also be used for SLAM to develop a dense 3D model for environment. However, for a mobile robot, the SLAM-based approaches are typically too computational expensive to utilized.

To follow a path and avoid obstacles, a mobile robot can utilize a monocular or a stereo-camera with depth information. A path for robot can be programmed, marked, or set to follow an object (or human). To follow a programmed or marked path, we can use pre-loaded map and localization techniques or simple markers or both. However, to follow an object with visual information, we must be able to detect track the object of interest in an image in real-time (approximately >10 fps).

In the task of object or person following, beside the localization and mapping, the robot must be able to detect and track the leader, handle leader occlusion, and avoid static and dynamic environment. In works by [20], [21], and [22], the robot can follow a person and avoid the obstacles when the leader always visible (no occlusion). To handle occlusion, work by [23] used Hue-Saturation-Intensity (HIS) based patch to detect and follow the leader. However, there must be no other similar image patches interfered with the tracking. Work by [24], on the other hand, used depth data and Support Vector Machine (SVM) based system to detect and verify the leader and utilized Extended Kalman Filter (EKF) (see [25] and [26] for more information on EKF) to track the leader if the occlusion occurred.

With recent development of low cost computer such as Raspberry PI (Figure 2.3 left) or Intel® Compute Stick (Figure 2.3 right) as well as the other low cost computer board, it becomes possible to develop a mobile robot with low power consumption with low cost. However, although some of the mentioned techniques may be applicable in the low computing power environment, it has not been empirical data to support that. Thus, it is beneficial to implement and test these methods, as well as improve or develop a new method to make a low-cost, low-power mobile robot available.

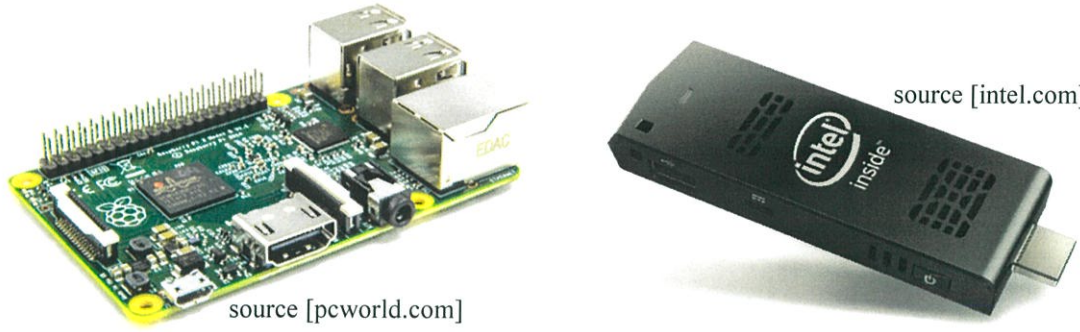


Figure 2.3 Raspberry Pi 2 Model B and Intel® Computing Stick (right)

However, this low cost computer comes with low computing power. Since most of the state of the art tracking methods require high computing power, these tracking methods would not be applicable in this low-computing-powered environment. There are very limited attempts in this area. One approach [27] makes use of chromatic information to perform fast tracking. However, the chromatic information is well-known for its unreliable dues to change in lighting. Another approach [28] [29] [30] is done by tracking object using MeanShift [31] and the improved CamShift [32]. This approach perform well in clean background which is not applicable in real world environment.

On the other hand, there are several promising attempt on low computing tracking in the field of Unmanned Aerial Vehicle (UAV). Typically, attempts in this field consist of designed markers and their marker detection algorithm. There are several tracking techniques in this field. For example, [33] used red and white landing pad (marker) for fast detection. Although the color information is sensitive to light, this approach has perform well in their controlled environment. On the other hand, approach by [34] made use of infrared-lit pad. Again, if we place the landing pad statically on the ground, it is easy to control infrared noise. However, for tracking moving objects, there can be a lot of infrared noise in the background.

There are several attempts which not make use of chromatic or infrared information, but rather use pure black and white geometric information (shape information). For example, one approach makes use of a modified version of Hough Transform [35] to detect ellipse while another approaches utilized an algorithm to detect connected components [36] [37] [38] in the scene.

2.2 Relevant Theories and Methods

Theories and methods used are for maker tracking which consists of circle detection along with circle recognizing and pruning. To detect circles, we tried Hough circle transform and later use Connected Component together with statistical filtering. Then, to classify what we detected as a circle, we developed an algorithm based on Support Vector Machine (SVM) and k-Nearest Neighbor (kNN). We will discuss all this techniques, as well as some geometric background in this chapter. Then, the next chapter will describe how we apply these technique in our algorithm development.

2.2.1 Hough Circle Transform (HTC)

Hough Circle Transform (HCT) is an adaptation of Hough Transform (HT) which can be used to find circles within an image (where HT is used to find straight lines – see [39] for a survey). Our implementation is based on an approach by [40]. To find circles within image, we create a grid of discrete range of circles and see how many foreground points, typically a result from edge detection, fall into each circle in the grid. The local maximums of voting are classified as circles. To create this grid, we start by describe a circle using parametric equation

$$x = a + Rsin(\theta) \tag{2.1}$$

$$y = b + Rsin(\theta) \tag{2.2}$$

Thus, a tribble (a, b, R) describes a circle. If we use the Hough Transform directly, we have to process all the pixel and put them in 3D grids of (a, b, R) which can be computationally expensive. The work by [40] reduces the grid dimension from 3D to 2D by consider only the center grid, grid of (a, b), first.

2.2.2 Connected Component Detection

Connected component detection (see [41] for more information) is an algorithm which is an application of graph theory. Each connected component is uniquely labeled based on a given heuristic. In this detection, the input image is viewed as a graph where each pixel is a vertex. An edge between two pixels is defined if two pixel are adjacent.

Connectivity can be 4-connected or 8-connected based on whether or not we define an edge from diagonal adjacency.

There are many approaches to the connected component detection. Two-pass version [42] is shown as in Algorithm 2.1. The general idea of this approach is to raster scan, scan from left to right from top to bottom row), and perform two tasks: 1) label each pixel with the smallest number of label available or from the other connecting pixel and 2) record the label of any labeled connecting pixel to be changed in the second pass. Then, in the second pass, we simply replace the label to be change with the smallest number in the group.

Algorithm 2.1 Two-pass connected component labeling [42]

First pass:

1. Process all pixel from left to right and top to bottom (raster scan)
2. If the pixel is in the area of interests (foreground pixel)
   2.1. Get a label from a labeled neighboring pixel
   2.2. If found one, assign it to the current pixel
   2.3. If there is no available neighboring label, create a new and unique label to assign to the current pixel.
   2.4. If there are more than one label from neighboring pixels, store them as a pair of equivalent labels.

Second pass:

1. Perform the raster scan one more time
2. If the pixel is in the area of interests
   2.1. Relabel the current pixel with the lowest equivalent label

2.2.3 Otsu Thresholding

In computer vision and image processing, the Otsu thresholding or Otsu's method [43] is a process of image thresholding which perform automatically (we do not have to manually choose the threshold value). It reduce a grayscale image to a binary image. The algorithm assume there are two classes of pixel in an image. The optimal threshold is

automatically compute by minimize their intra-class variance. An algorithm in Otsu's thresholding exhaustively search the threshold by define a weighted sum of variances of the two classes as

$$\sigma_{\omega}^2(\tau) = \omega_0(\tau)\sigma_0^2(\tau) + \omega_1(\tau)\sigma_1^2(\tau) \tag{2.3}$$

where $\omega_{0,1}$ are the weights of the probabilities of the two classes, τ is a threshold to separate the two classes, and $\sigma_{0,1}^2$ are variances of the two classes.

The class probability $\omega_{0,1}(\tau)$ can be computed from the L histogram:

$$\omega_0(\tau) = \sum_{i=0}^{\tau-1} p(i) \tag{2.4}$$

$$\omega_1(\tau) = \sum_{i=\tau}^{L-1} p(i) \tag{2.5}$$

where $p(i)$ is a probability of the pixel i.

A paper by Otsu [43] shows that minimizing the intra-class variance is equivalent to maximizing inter-class variance:

$$\sigma_b^2(\tau) = \sigma^2 - \sigma_{\omega}^2(\tau) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \tag{2.6}$$

$$= \omega_0(\tau)\omega_1(\tau)[\mu_0(\tau) - \mu_T(\tau)]^2 \tag{2.7}$$

where class probabilities ω and class means μ is expressed, while the class mean $\mu_{0,1,T}(\tau)$ is:

$$\mu_0(\tau) = \sum_{i=0}^{t-1} i\frac{p(i)}{\omega_0} \tag{2.8}$$

$$\mu_1(\tau) = \sum_{i=t}^{L-1} i\frac{p(i)}{\omega_1} \tag{2.9}$$

$$\mu_T = \sum_{i=0}^{t-1} ip(i) \tag{2.10}$$

where, from Eq. 2.8-2.10, we can verified that:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T \tag{2.11}$$

$$\omega_0 + \omega_1 = 1 \tag{2.12}$$

The class probabilities and the class means can be compute iteratively by algorithm 2.2.

Algorithm 2.2 Otsu's Thresholding

1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible threshold $\tau = 1..maximum\ intensity$
   3.1. Update ω_i and μ_i
   3.2. Compute $\sigma_b^2(\tau)$
4. The optimal threshold τ is the one that maximize $\sigma_b^2(\tau)$

The Otsu's thresholding performs well if the input image exhibits bimodal distribution and there is a sizable mean difference between the two classes. However, if the size difference between two classes is large, the Otsu's thresholding may be confused by the large variation of the larger class.

2.2.4 Morphology

Morphology is a filter taken from a Mathematical theory. Originally, it is invented to operate on binary images and later improved to handle grayscale. However, this section will be discussed operations on binary image only. The morphology is an operation which take two operands: an input image and a kernel. Both input image and the kernel are binary image. The kernel image can be of any shape where the cover area of the kernel are typically represented by pixels of value 1. Some fundamental morphology operations are:

Dilation

In image processing, dilation refers to an operation to dilate, expand, the areas of interests. Given a binary image of 0's and 1's and a kernel which is also a binary image. In the dilation process, the kernel will be centered on every pixel in the given image. Then, if the center pixel where the kernel placed is 1, all the pixels in the resulting image cover by the area of the kernel will be set to one.

After dilation, the area of interests will be larger (see Figure 2.4 for illustrated example). The expanded corners will be round. One example of dilation application is finding edges of the given image by subtract the dilation result with the original.

Erosion

Erosion is a process to make the area of interest thinner. This process is popular in the area of Optical Character Recognition (OCR) for thinning character images. Erosion process is done by centering the erosion kernel onto every pixel of the given image. A pixel at the center will be given a value of one only if every pixel from the given image in the area of interest of the kernel are all 1's.

The erosion makes the area of interest thinner as illustrated by Figure 2.4 on the right. It preserves sharp corners. However, sometimes it can disconnect the area of interest in the given image which may or may not be the desired effect.

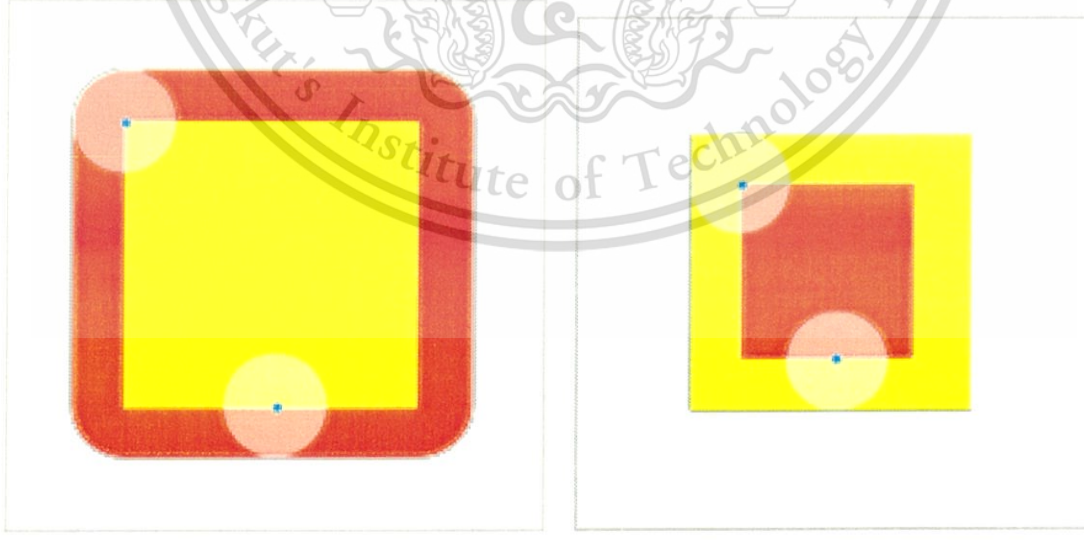


Figure 2.4 examples of dilation (left) and erosion (right)

Opening

Morphology opening is a process where we apply erosion and then dilation on the given image. By doing so, the erosion process will remove some connected lines or thin strips between large areas. Then, the dilation process will expand the large areas back to its original size but the thin-line connection will not be recovered. In our detecting algorithm, we use this process to remove a thin connection between two circles which sometimes appears from the result of motion blurring.

Closing

Morphology closing is done by applying the dilation on the given image first, then apply the erosion. By this process, the thin connection will be made visible. Some disconnected lines can also be joined together using this process.

2.2.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) [44] is a parametric classification tool widely used in computer vision and pattern recognition. Originally, SVM was developed for separating two classes by finding an optimal hyper-plane which maximizes the distance/margin between the two classes (see Figure 2.5 for an illustrate example). In a simple case where the two classes are linearly separable, the hyper-plane chosen by SVM is the one that maximizes the distance between the closest points (the support vectors) from both classes to the hyper-plane. Thus, given m training data, $\{(x_m, y_m)|m \in \{1,..,M\}, y_m \in \{-1,1\}\}$, the linear SVM classifier can be defined as:

$$f(x) = \left(\sum_i a_i y_i x_i\right) x + b \tag{2.13}$$

where $\{x_i\}$ is the set of support vector, $\omega = \sum_i a_i y_i x_i$ is the optimal solution and b is determined by solving

$$\max_{\omega} \frac{\|\omega\|^2}{2} \tag{2.14}$$

subject to $y_i(\omega x_i - b) \geq 1$.

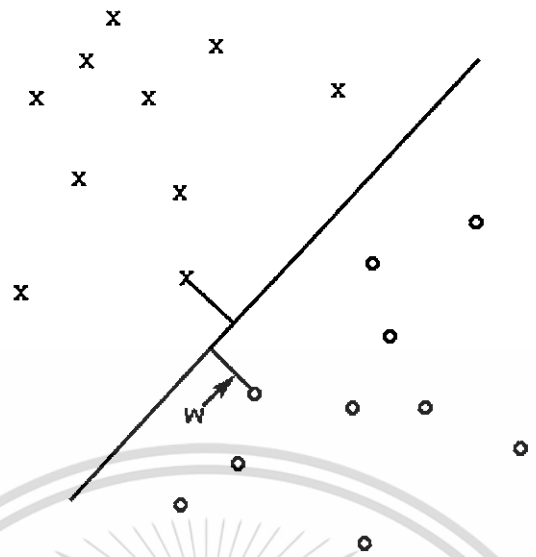


Figure 2.5 Support Vector Machine example.

However, in the general case where two classes are not linearly separable, a new constraint, $\varepsilon_j > 0$ is added. The objective function becomes:

$$\max_{\omega} \frac{\|\omega\|^2}{2} + C \sum_i \varepsilon_i \tag{2.15}$$

subject to $y_i(\omega x_i - b) \geq 1 - \varepsilon$, where C is a parameter to be set during training.

In order to use SVM for multi-class classification, we can take the "one-against-one" approach [45]. Thus, if k is the number of classes, then $k(k-1)/2$ classifiers are constructed by training with data from two classes. Then, the final prediction is the class label with the maximum number of votes.

2.2.6 k-Nearest Neighbor

The k-Nearest Neighbor (k-NN) is a non-parametric method which classifies objects by examining the closest k training samples in a feature space. Then, the class label of the input object is determined by the voting of the labels of the k nearest neighbors (see figure 6 for illustrated example). Two choices of parameters for the k-NN method are k and the distance metric which are typically determined by training samples. In a simple

case where k=1 and Euclidean distance is use, the class label of an unknown observation X is identified as that of Y which computed by,

$$Y = \min_i(\|X - Y_i\|_2) \tag{2.16}$$

where Y_i are the data from the training set and $||.||_2$ is an L2 norm or the Euclidean distance.

In the Figure 2.6, an unknown input $*$ is classified as an o by k nearest data from the training set. When $k = 1$ (inner circle), there is only one example which is o so the unknown input is classified to be o. When $k = 3$ (middle circle), there are three examples from the training set which are two o's and one x. By voting, the unknown input is classified to be o in this case. Similarly, when $k = 5$ (outer circle), there are four o's and one x which make an output o wins by voting. Note that the value of k is typically chosen to be odd so we do not have a tie in voting.

Figure 2.6 k-NN example

2.2.7 Classification Framework based on Support Vector Machine and k-Nearest Neighbor

This classier framework is based on work by [46]. In this work, an input is classified by multiple SVM to generate a vector of prediction first. Then, the vector of prediction is classified to get a predicted class using k Nearest Neighbor. Figure 2.7 illustrate motivating example of this framework. In this example, there are only two classes: A and B. However, within each class, there are three subclasses which are closer to the other class than theirs owns. If we only use one boundary to separate the class A and class B, we must use non-

linear boundary (see Figure 2.7 left for example). On the other hand, if you use three SVMs to separate each subtype of data. Let us define them as SVM_t for the triangle type, SVM_p for the pentagon type, and SVM_s for square type. We can use linear boundary for each label.

Figure 2.7 Motivating example of classification framework bases on SVM and k-NN

However, the prediction of the other subclasses can be of incorrect values as shown in Table 2.1. This table can be our lookup table (as in k-NN) for an unknown value. Each row of this table makes up a vector of prediction. Thus, if a new input produces a predicted vector as (A, B, A), from the three trained SVM's in the this example, the predicted class of this input would be B Square, which can classified to be B in our two-class system.

Table 2.1 Data prediction by three trained SVM's

Input class and subclass	Predicted class label by		
	SVM_t	SVM_s	SVM_p
A Triangle	A	B	B
B Triangle	B	A	A
A Square	B	A	B
B Square	A	B	A
A Pentagon	B	B	A
B Pentagon	A	A	B

This frameworks is demonstrated in [46] that it has benefit of speed and accuracy. The speed of prediction time compose of two steps, prediction time of SVM and k-NN. When using the k-d tree, a space-partitioning data structure for organizing points in a k-dimensional space, the prediction time of k-NN is $O(kd\,log(n))$ [47], where d is the dimension of the input, which is the number of subclasses in this case. The prediction time of SVM is $O(d)$. Thus, the combined complexity is $O(dn + dt + tn)$ where t is the number of SVM.

The accuracy of this framework has been tested comparing to another two systems. The first one is a system with one linear SVM where all subclasses are combined into one class (combined in Table 2.2). The second system is a system with voting SVM where the final prediction class is determined by voting of all SVM from each subclasses (voting in the Table 2.2). All the SVM in this comparison are linear SVM with maximum of 100 iterations with 10^{-6} tolerance error. The dataset used in this evaluation are Thai characters images of 16x16 pixel made available by NECTEC (accessible via http://www.nectec.or.th/corpus). The image are acquired from books, journals, magazines, and newspapers by scanning and cropping. There are three different subset within this dataset. First, the training set contains 146,640 images. The second is the validating set contains 117,312 images from the same sources as the training set (but not the same images). Lastly, a testing set acquired images in the similar manner (scanned and cropped) but from different sources. The framework we use in our system (SVM+k-NN in the Table 2.2) shown better accuracy when comparing to the other two. Even in the testing set of unfamiliar images. The chosen framework reported the highest accuracy.

Table 2.2 Accuracy comparison of classification frameworks

Evaluation	SVM+k-NN	combined	voting
Training set	99.78	93.58	85.89
Validating set	97.13	92.24	85.13
Testing set	89.09	87.54	78.63

2.2.7 Projective Geometry

The projective geometry is used to model a camera. This model represents a pinhole camera (Figure 2.8) where rays of light from object pass through a single pinhole to make imprints on a film which represent an image plane. In this process, we map 3D world onto 2D image plane with the following properties:

1. This is a many-to-one mapping. All points in a single line passing through the pinhole will be mapped onto one point in the image plane.
2. Each point will be mapped to a point, except the focal point – a point at the pinhole.
3. A line will be mapped to a line. In another words, three co-linear points will be mapped to three co-linear points in the image plane. Unless the co-linear points belong to a line passing through the pinhole which will be mapped to a point.
4. A plane will be mapped to a plane (or half a plane) unless it passes the focal point. In that case, a plane will be map to a line.

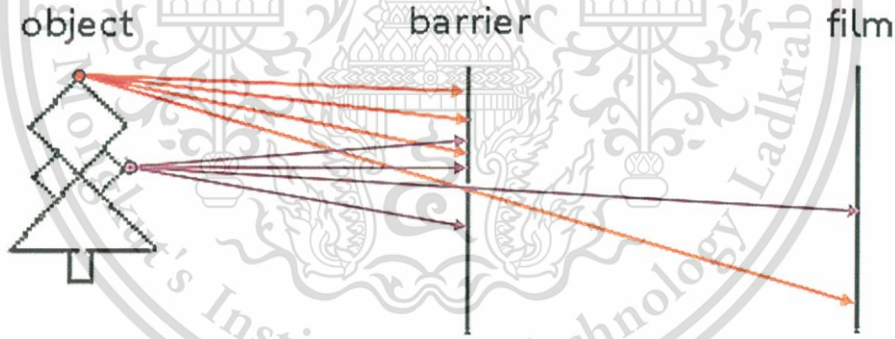


Figure 2.8 Pinhole camera

To setup a projective geometry for a camera, we move the image to the front of the focal point (Figure 2.9). In this setup, we have C as a camera focal point which will be an origin in the real-world coordinate. The point P is the origin of the image plane. The length from C to P is called a focal length which usually represent by f. A principal axis, Z-axis, originate from C and point in the direction of P. X-axis and Y-axis are horizontal

and vertical axises. Typically, we represents coordinate of a point within the image plane with small letter x and y.

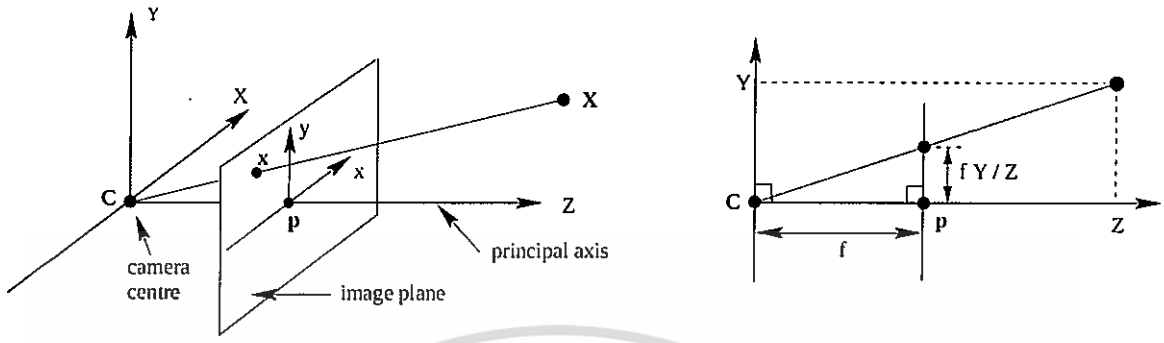


Figure 2.9 Projective geometry

Given a point in 3D-world coordinate is represented by (x, y, z) maps to a point $(\dot{x}, \dot{y})$ in the image plane. From similar triangle property, we have:

$$\frac{\dot{x}}{x} = \frac{f}{z} \tag{2.17}$$

and

$$\frac{\dot{y}}{y} = \frac{f}{z} \tag{2.18}$$

Thus, a point at (x, y, z) will be mapped to $\left(\frac{fx}{z}, \frac{fy}{z}\right)$. However, the division by z complicates the calculation. Therefore, we usually adopt Homogeneous Coordinate System to get rid of the complication.

Chapter 3

System Design and Specification

In this chapter, we will discuss a development of the proposed Visual Based Navigation System. We will start by describing our hardware setup which consist of low-computing power unit. Then the information and geometric aspect of the designed marker is provided. After that, our approach for the marker detection is given in detail. Lastly, the framework for calculate turning angle and distance from the marker to the camera attached to the mobile robot is described along with the process of how to calibrate its parameters.

3.1 Hardware and Software Setup

The hardware setup, as illustrate by Figure 3.1, consist of Raspberry PI 2 Model B+, PiCAM, two motors and a servo with their driver, a 5V power bank. Note that we do not use the servo in our navigation system since it would complicate the angle calculation. In this setup, Raspberry PI is the main computing unit. It takes a picture input from PiCAM camera. Then, the computing unit detects and computes the real-world location of the marker. Finally, the attached motor is controls by the Raspberry PI according to the computed information.

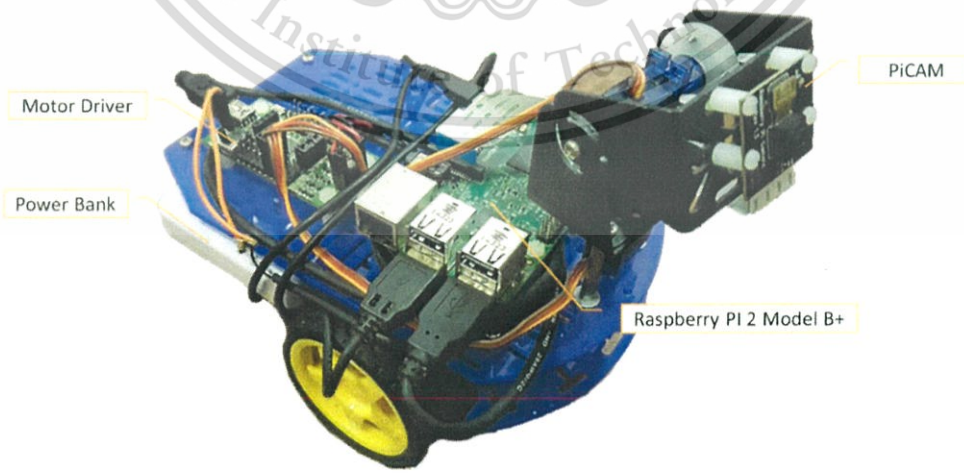


Figure 3.1 Mobile robot for testing

In our setup, the main computing unit is Raspberry PI 2 Model B+ (see Figure 3.2). The main storage used in this model is the push-push micro SD memory card. This model of Raspberry PI contains 4 USB 2.0 ports. It has 40 GPIO pins which are arranged as Figure 3.3. Although the GPIO can be used to drive our motors directly, we decided to control our motors through driver.

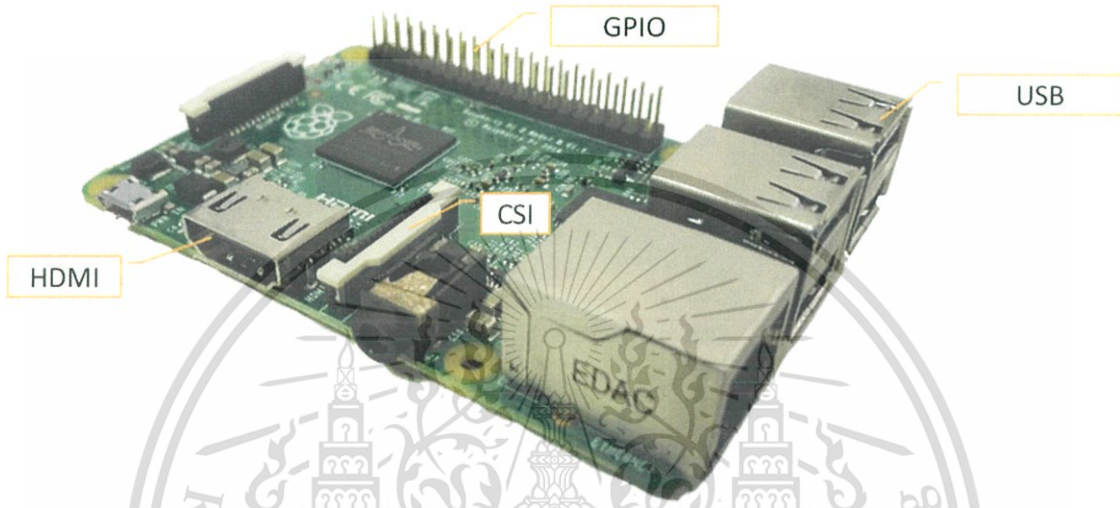
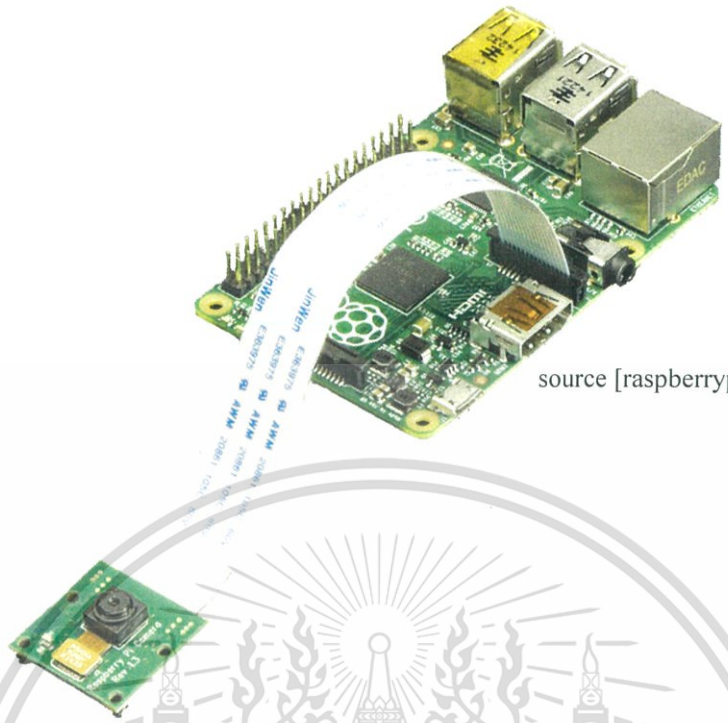


Figure 3.2 Raspberry PI 2 Model B+

Figure 3.3 GPIO pin diagram

To take image input, we make used of Camera Module of Raspberry PI which can be attached using Camera Serial Interface (CSI) as shown in Figure 3.4. This module has a small size of 25mm×20mm×9mm and weight approximately 3 grams. It has native resolution of 5 megapixel and has a fixed focus lens onboard. In our setup, we use it to capture 640x480 images sequence.



source [raspberrypi.org]

Figure 3.4 Attached Camera Module

3.2 Visual Tracking System

The flow of visual tracking system start by retrieving image from PiCAM. With the scene image, we detect candidates for circles using either Hough Circle Transform or Connected Component Detection algorithm. Then, the detected circle candidates are filtered out by series of filtering to obtain only strong candidates as describes in Section 3.2.1. After we have strong candidates, we will attempt to group them to form a four-circle of our marker as describes in 3.2.2. Then, with location information of the marker and its circle components, we can recover relative distance from the camera unit to the detected marker as described in 3.3.

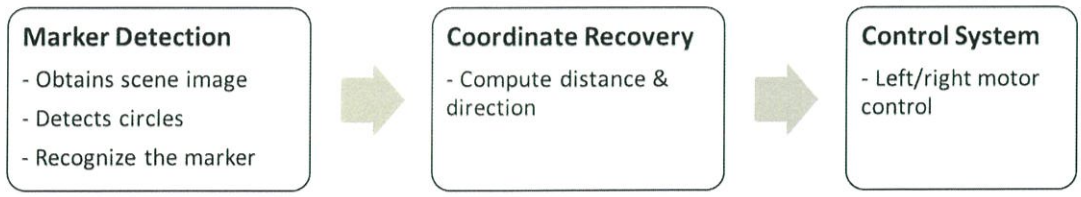


Figure 3.5 The overall process of marker detection

3.2.1 The Designed Marker

Our marker is designed for fast and easy detection in complex scene with complex lighting. It consists of four black circles on white background as illustrated in Figure 3.5. The marker can be of any size, as long as we keep the following ratio. Given that the maker is a white square of size 8x8 units, there will be four black circles within the squire. If the corner of the square is a coordinate (0,0), their centers are placed at (2,2), (2,6), (6,2), and (6,6). In another words, the four centers of these circles will form a square of 4x4 units in the middle of the marker. The radii of the circles, starting from top right and goes clock-wise, are of length 2.5, 1.5, 2, 3 units. Thus, for searching, we can look for a set of circle with ratio 3:4:5:6. Notice that ratio of any two radii are unique. Thus, if we only detected two circles, this unique ratio will allowed us to infer that which two circles are detected which in turn allowed us to predict where to find the rest of the circles.

This design allows us to easily search for circles within the marker. We can utilized Hough Circle Transform for precision. On the other hand, we can search these circles using Connected Component search together with filtering for fast detection. Our approach for detecting these circles along with how to filter out non-circle will be given later in the Section 3.2.3.

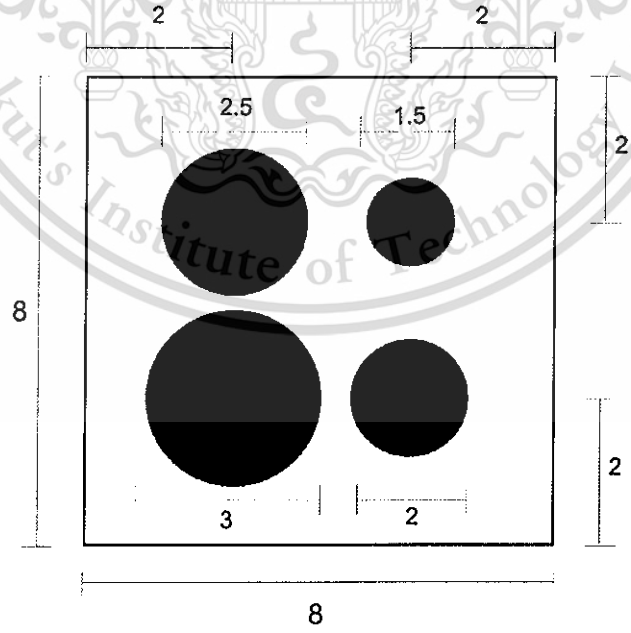


Figure 3.6 The Designed Maker

3.2.2 Circle Detection

There are two schemes that we explored for circle detection: Hough Circle Transform and Connected Component detection. The Hough Circle Transform must be performed on the grayscale version of the scene image. When using Hough Circle Transform, we relax its parameters to discover more candidate circles in attempt to maximize our change of detecting the circles in our marker. Despite the parameters relaxation, the precision of Hough Circle Transform is still higher than the Connected Component Detection approach. However, the speed of detection of Hough Circle Transform is comparatively slower. As illustrated in Figure 3.7 (left), the relaxed Hough Circle Transform has detect a lot of false positives. We have to take an extra step to filter unlikely candidate before the next process.

To apply Connected Component Detection, we first take a scene image, convert it into grayscale, and then perform Otsu's Thresholding as illustrated in Figure 3.8b. Then, before applying the Connected Component Detection, we perform morphology opening as resulted in Figure 3.8b. Finally, after the connected components has been detected, we perform **Bounding Box Ratio Filtering** by filter out the circles with their bounding box are not close to squares (hence, not likely to be enclosing circles). This can be done by filter out any bounding box with

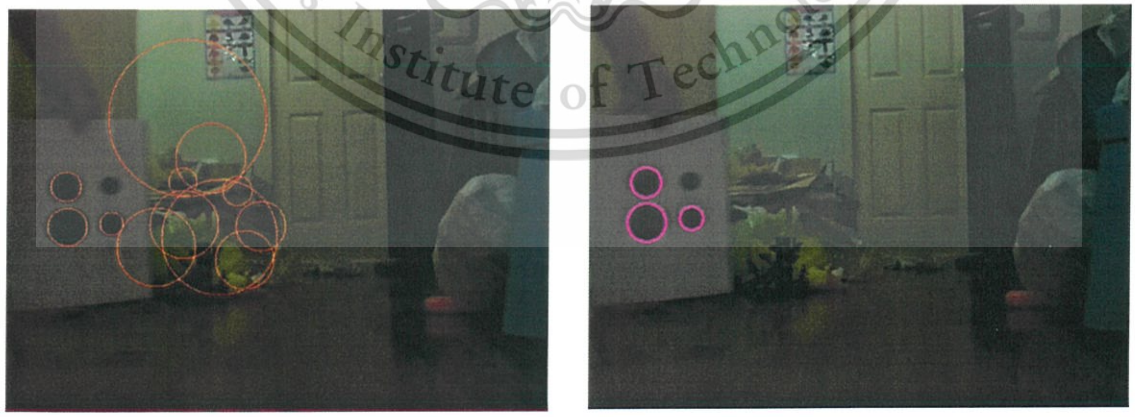


Figure 3.7 Result of Hough Circle Transform and Black Circle Filtering.

To apply Connected Component Detection, we first take a scene image, convert it into grayscale, and then perform Otsu's Thresholding as illustrated in Figure 3.8b. Then, before applying the Connected Component Detection, we perform morphology opening as resulted in Figure 3.8b. Finally, after the connected components has been detected, we perform **Bounding Box Ratio Filtering** by filter out the circles with their bounding box are not close to squares (hence, not likely to be enclosing circles). This can be done by filter out any bounding box with

$$lower\_bound < \frac{width}{height} < upper\_bound \tag{3.1}$$

where $width$ and $height$ are width and height of the bounding box, $lower\_bound$ and $upper\_bound$ are parameters to be set. In our experiment, $lower\_bound$ and $upper\_bound$ are set to be 0.9 and 1.1 respectively. Furthermore, the bounding box which are too big or too small are filtered out in this step. The result of this filtering step is illustrated in Figure 3.8d.

For Both Hough Circle Transform and Connected Component Detection approaches, we have to perform **Black Circle Filtering** by filter out non-black circles as they are unlikely to be the marker circles. One way to validate that we have a black circle, we can average all intensities of pixels within a detected circle and verify if the computed average is lower than a particular threshold. However, for speed consideration, we can randomly pick a few points within a circle to compute this average intensity. Thus, our black circle filtering can be done by

$$average = \frac{\sum_{i=1}^{n} intensity(p_i)}{n} < \tau \tag{3.2}$$

where n is a number of point picked from the circle (only 5 points are sufficient enough in our experiments), p_i are points in the circle, and τ is the intensity threshold. This approach is sufficient since there is not many pure black circles in nature (see Figure 3.8d

Figure 3.8a Complex scene with the marker

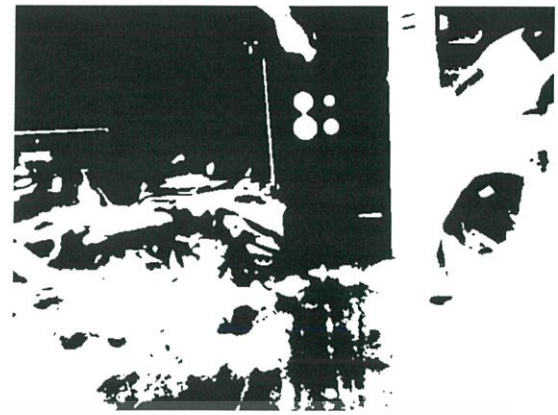


Figure 3.8b Scene after Otsu's thresholding

Figure 3.8c After morphology opening

Figure 3.8d Detected connected components

Figure 3.8e. After black circle filtering

Figure 3.8 Connected Component example

and Figure 3.8e for illustrated results). However, if some stray black circles appeared, we can filler them out in the later stage.

After the black circle filtering, we apply the prediction framework based on Support Vector Machine and k-Nearest Neighbor [46] in order to improve circle detection accuracy. In the training process, we have two classes: circle and non-circle. In the circle class, we

separate them into 4 subclasses by their size. Then, the non-circle image are randomly taken from the scene.

3.2.3 Marker Recognition

After we have likely candidate for the marker circles, we perform **Adjacency Filtering**. As in our markers, the four circles are placed close to each other. Thus, we will consider only a set of two, three, and four black circles that are not too close or too far from each other (given mathematically in Eq. 3.3). This can be done by finding a set of two, three, or four connected component in a graph where each circle is a node. To construct the graph, we connect any two nodes if it lies within the following criteria: 1) Since the closest two circles in our marker are of size 2.5 and 3 inches placed 3 inches apart. Thus, we can connect two detected circle if they are not closer than the diameter of the larger circle. 2) However, if the two circles are in our marker, it must not be too far apart. The farthest two circles, in relative to the diameter of the larger circle, is the space between a circle of size 1.5 and 2 inches with 3 inches apart. Thus, we can filter out if the two detected circles are placed apart for more than 1.5 times of the diameter of the larger circle. Thus, we can connect any two circle if

$$2 * diag_{large} - \tau_{low} \leq dist \leq 3 * dia_{large} + \tau_{high} \quad (3.3)$$

where $dist$ is the distance between the two circles, $diag_{large}$ is a diameter of the larger circle. The τ_{low} and τ_{high} are positive real number which are tolerances to relax condition to compensate for the detection error. After the graph is constructed, we search within the graph for a set of connected 2, 3, or 4 circles and consider these to be our candidate for the marker.

Next, we perform **Four Corners Filtering**. This filtering utilizes known geometric property of the marker. For all the connected circles, if they are connected in a group of three and four, we can test whether or not these groups are in our marker or not by checking if the corner(s) formed in a particular group are close to 45 or 90 degrees or not. Given three center points of any three circles in the group, c_1, c_2, and c_3, we can compute an angle at c_1 by

$$\theta = \arccos \frac{(c_2 - c_1)(c_3 - c_1)}{\|c_2 - c_1\|\|c_2 - c_1\|} \tag{3.4}$$

where θ is the angle at c_1. The computation can be repeated to compute an angle at c_2 or c_3. If there is an angle too far from right angle or half of the right angle, in the case of diagonal lines, we can filter out some of these groups of circles.

To further guarantee that the group of circles we detected is our marker, we perform **Diameter Ratio Filtering**. Since the four circles in our marker is of ratio 3:4:5:6, which is not typically occurred in nature, we can filter out a set of four circles that are not following these ratio. If only three circles are detected, the possible ratios are 3:4:5, 3:4:6, 3:4:6, and 4:5:6. On the other hand, if only two circles are detect, there are six possible ratio which are 3:4, 3:5, 3:6, 4:5, 4:6, and 5:6. Thus, we can filter out any group of circles not following these ratios. Note that since the diameters of the detected circles are subject to some error, we need to have some torrent relax the require ratio of these circles.

Lastly, we perform **The Third and The Forth Circle Verification**. If there are three circles which there centers, (c_1, c_2, c_3), from a triangle with approximately 90, 45, and 45 degrees in corners, we can compute the location of the forth circle by

$$c_4 = c_1 + (c_2 - c_1) + (c_3 - c_1) \tag{3.5}$$

given that c_1 is the corner with 90 degrees.

If only two circles are detected, we have six conditions for searching. The six conditions are judged by considering the ratio of radii of the two detected circle. If we place the smaller circle on top, the larger one on the bottom and if we found the ratio of 3:4 or 4:6 we can search to the left of both circles (see Figure 3.9). If we found the ratio of 3:5 or 5:6, we must search on the left of both circle. On the other hand, if we found the ratio of 3:6 and 4:5, we must search along the diagonal line perpendicular to the line between the two centers. Again, after we found the third and the forth location, we can

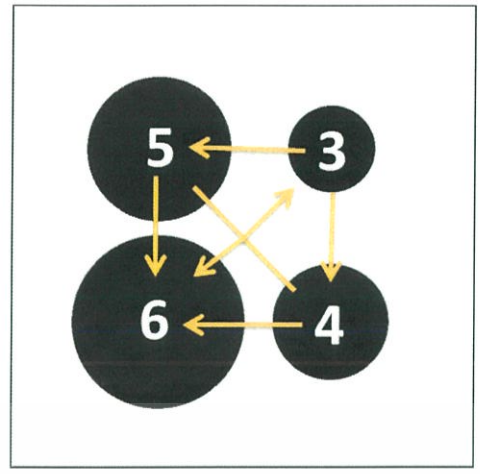


Figure 3.9 direction to find missing circles

apply **Black Circle Filtering** on the location to ensure that we have the third and the forth circle.

Although some calculation can be performed if only an isolated black circles are detected, we consider it to be too computational expensive to do so. Thus, the marker detection is considered to be a failure in this case.

3.3 Application to Real-time Navigation System

In order to control the robot according to the detected marker, we calculate real world coordinate from two pieces of information: center of the detected maker and distance between two detected circles. The reason why these information were chosen is discuss in Chapter 4. The calculation of real world distance is as follow.

3.3.1 Distance Detection

In this calculation, we have d as a distance between two detected circles, D is a distance from camera unit to the marker, f as a focal length of the camera, and s as an actual distance between two circles in the marker as illustrated in Figure 3.10. By using property of similar triangle, we can say that

$$\frac{D}{f} = \frac{s}{d} \tag{3.6}$$

which can be written as

$$D = \frac{fs}{d} \tag{3.7}$$

Since the focal length f and the distance between two circles in the marker s are fixed, we can combine them as k. The last equation can be written again as

$$D = \frac{k}{d} \tag{3.8}$$

In theory, k can be calculated from known camera parameters and known marker sizes. However, for practical purpose, we can simply estimate k by measure the actual distance from the camera to the marker, D, and the distance between two circles in the detected marker, d (in pixel unit).

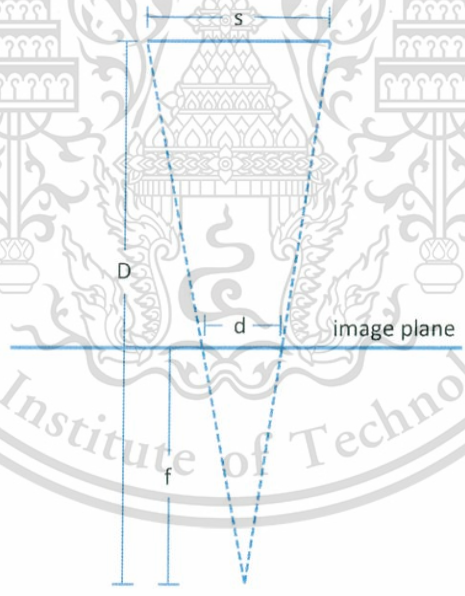


Figure 3.10 Distance calculation variables

From equation (3.8), we can estimate k by

$$k = D \times d \tag{3.9}$$

In our calibration, we vary the distance D from the camera to the marker in the step of 5 inches from 10 to 30 inches. Then, since the previous experiment demonstrate that the circle size detection is more accurate, we decided to measure d by the detected size of the largest circle. As show in Table 3.1, we estimated k to be 857.

3.3.2 Angle Detection

To calculate angle, we need the distance dx between the center of image and the center of marker in image plane as illustrated by Figure 3.11. Note that we do not need the real-world distance D from the camera to the marker and the real-world distance t from the center line of the camera to the center of the actual marker in our calculation.

Table 3.1 Calibration distance

D (inches)	d (pixels)	$D \times d$
10	85	850
15	57	855
20	43	860
25	34	850
30	29	870
	average	857

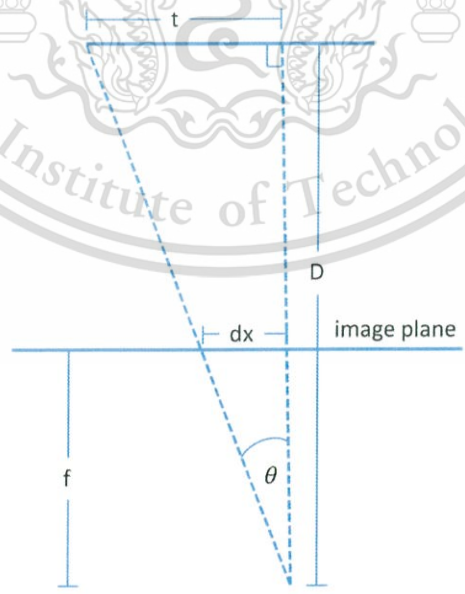


Figure 3.11 Angle calculation variables

To calculate the angle θ, we can divide dx by f to get the $arctan$ by the following calculation:

$$\theta = \arctan\frac{dx}{f} \tag{3.10}$$

Since each camera has a fix focal distance f, each angle theta would be a result from the same dx. Thus, rather than calculating the $arctan$, we can simply map values of dx to angles θ. For example, from our measurement, if the dx is measured to be 50, we can approximate the current theta to be 5 degrees.

Although we can recover f, in pixel unit, by,

$$f = \frac{dx}{\tan\theta} \tag{3.11}$$

which allow us directly calculate the θ using arctan. However, since calculating arctan can be expensive, we can alternatively measure and map a range of dx to a range of theta and use some simple interpolation to calculate the theta value in between. For example from our measurement, we approximated that $dx = 50$ pixel maps to $\theta = 5$ degree and $dx = 100$ pixel maps to $\theta = 10$ degree. If we measure dx to be 70 pixel, we can approximate θ to be 7 degree using linear interpolation.

In this calibration, we are to map the detected dx to the angle θ and interpolate the missing values without using $arctan$. However, we also calculated f using the Eq. 3.12 to verify that we have approximately the same values. From the data shown in Table 3.2, the focal length f can be estimated to be 604 pixel. If we detected the x distance between the image center to the center of the largest circle in the marker, dx, to be 50, 103, 169, and 229, (the average values) then we can estimate the theta to be 5, 10, 15, and 20 degrees respectively. Hence, if we detect the dx to be any other value between 0 and 686, we can estimate the theta using linear interpolation by

$$\theta = \theta_i + (\theta_{i+1} - \theta_i) * \frac{dx - dx_i}{dx_{i+1} - dx_i} \tag{3.12}$$

where $dx_i < dx < dx_{i+1}$ and dx_i where i=0..5 are 0, 50, 103, 169, and 229 respectively, and θ_i where i=0..5 are 0, 5, 10, 15, and 20 respectively. Note that we only need to calibrate up to 20 degrees since the angle greater than that will placed the marker out of the field of view (FOV) of the camera.

Table 3.2 Calibration angle

θ (degree)	dx (pixel)				f		
	D=10	D=15	D=20	average	D=10	D=15	D=20
5	48	51	51	50	549	583	583
10	100	103	107	103	567	584	607
15	163	167	176	169	608	623	657
20	224	236	226	686	615	648	621
							average f = 604

Chapter 4

Experimental Results and Analysis

To illustrate how well our idea performs, we conducted a series of experiments to point out several benefits and limitations to our approach. First, we will start by describing two datasets used in these experiments. Then, with the dataset available, we empirically evaluate our tracking system in two perspectives: speed and accuracy. The latter part of this chapter provides examples of how to apply our proposed method to the real-time navigation system.

4.1 Dataset

There are two datasets used in our experiments. The first dataset composes of image sequence of a complex scene as show in Figure 4.1. There are 100 images of size 640x480 pixel in the dataset. Within the scene, there are clusters of black and white along with several other color which make detection comparatively challenging for most algorithms (color tracking and MeanShift/CamShift tracking would fail in this case). There are also changes in lighting within the scenes. Also, the marker appeared in the scene can be very different in size which makes the search space larger. The complexity of the scene, the change in lighting, and the size variation slow many algorithms down, including Hough Circle Transform. Within the sequence, there are some interval where the marker moves very fast and causing motion blur. Furthermore, there are some brief interval where the marker moves out of the scene.

For the section dataset, we took 100 images of size 640x480 pixels from 5 different locations where scene complexities and lightings are noticeably different. For each location, we have 10 positive images (images with makers) and 10 negative images (images without a maker). Example images are illustrated in Figure 4.2 where top five are positive images. The positive images are taken with known distance and angle of the marker. These images will be used in Section 4.3 to evaluate the accuracy of distance and angle detection.

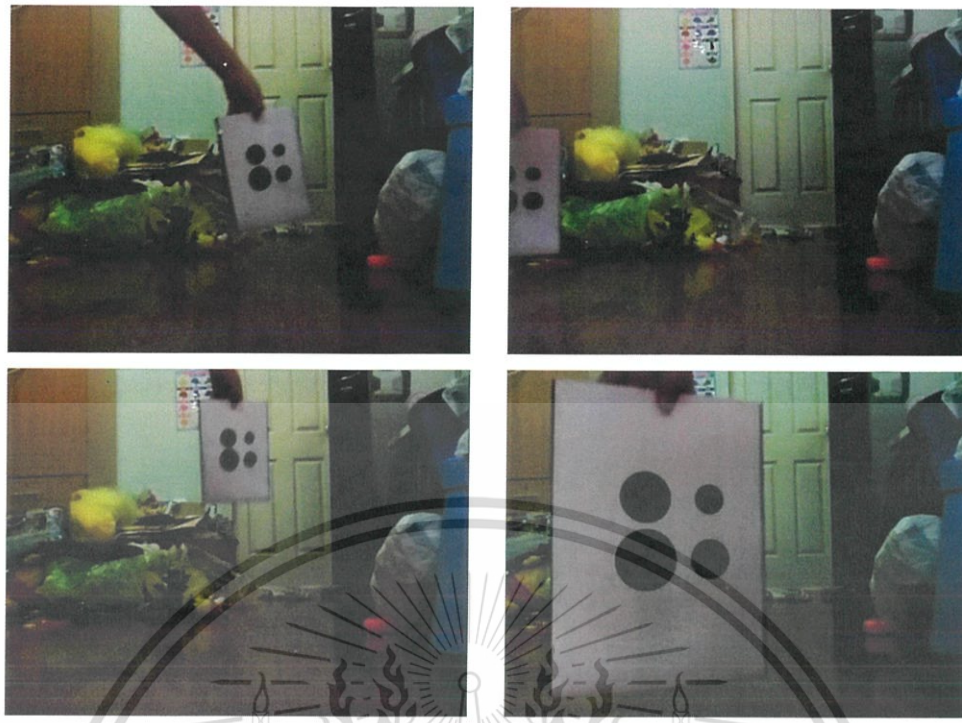


Figure 4.1 Examples of scene images from the first dataset

4.2 Visual Tracking System

The main function of our proposed system is to produce fast and accurate marker detection such that the navigation unit could recover real world location correctly and efficiently. Thus, we performed two empirical studies in this section: one for accuracy and one for speed.

4.2.1 Accuracy Performance

In this experiment, we track 100 frames of image sequence to measure reliability and accuracy of our proposed algorithm. Results of tracking using Hough Circle Transform is illustrate in Figure 4.3. Note that the six frames with missing ground-truth are the ones where our marker is not visible in the scene. Out of 100 frames, our tracking scheme with Hough Circle Transform have missed 6 frames. If we carefully examine the ground-truth in Figure 4.3, we can see that the missing detection may have caused by abrupt change in y-position of the marker. This can be caused by motion blur of the marker image (see Figure 4.4) which makes circle detection more difficult.

Figure 4.2 Examples of scene images from the second dataset

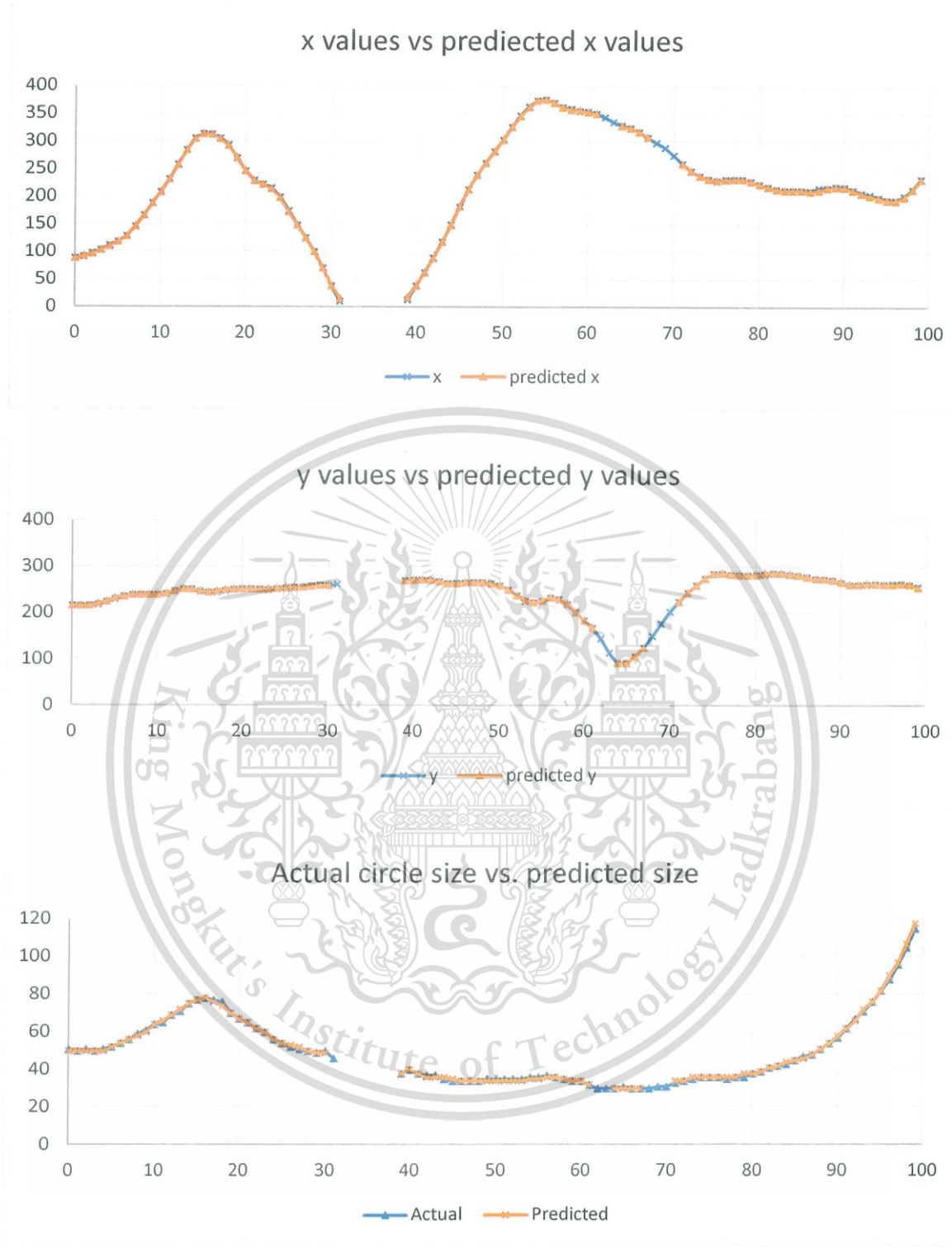


Figure 4.3 Ground truth and predicted values of center (x, y) and size of the largest circle in the marker from Hough Circle Transform approach

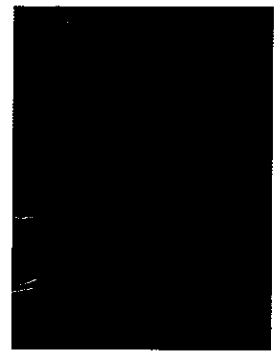


Figure 4.4 Example of motion-blurred image.

The average error of center position detected is 1.02 pixel and the size of the largest circle detected is off by approximately 0.51 pixel in average. The maximum error of size detection is approximately 5.6% of the actual size. Thus, the size information seems to be more reliable for approximate distance from the camera to the marker comparing to the marker center detection. On the other hand, the error in the marker center detection appears to be higher and the maximum error is also reported to be around 8% which is also higher than the maximum error of the size detection. Even though the error and the maximum error is higher, we suspect that the distance between detected circles can also be sufficiently used as an input for the distance approximation.

The results of our tracking method with Connected Component Detection are illustrated by Figure 4.5. Out of 100 frames, the Connected Component approach only missed 1 frame. The missing frame contains image of the marker when it is about to leave the scene causing the Otsu's thresholding to perform poorly. Apparently, this approach is more robust to speed of the moving marker. This may be the result of the morphology opening which make two circle separate. As shown in Figure 3.8b, the motion blur cause Otsu's thresholding to join the two large circles in the marker. Then, the morphology opening make them separate again as in Figure 3.8c.

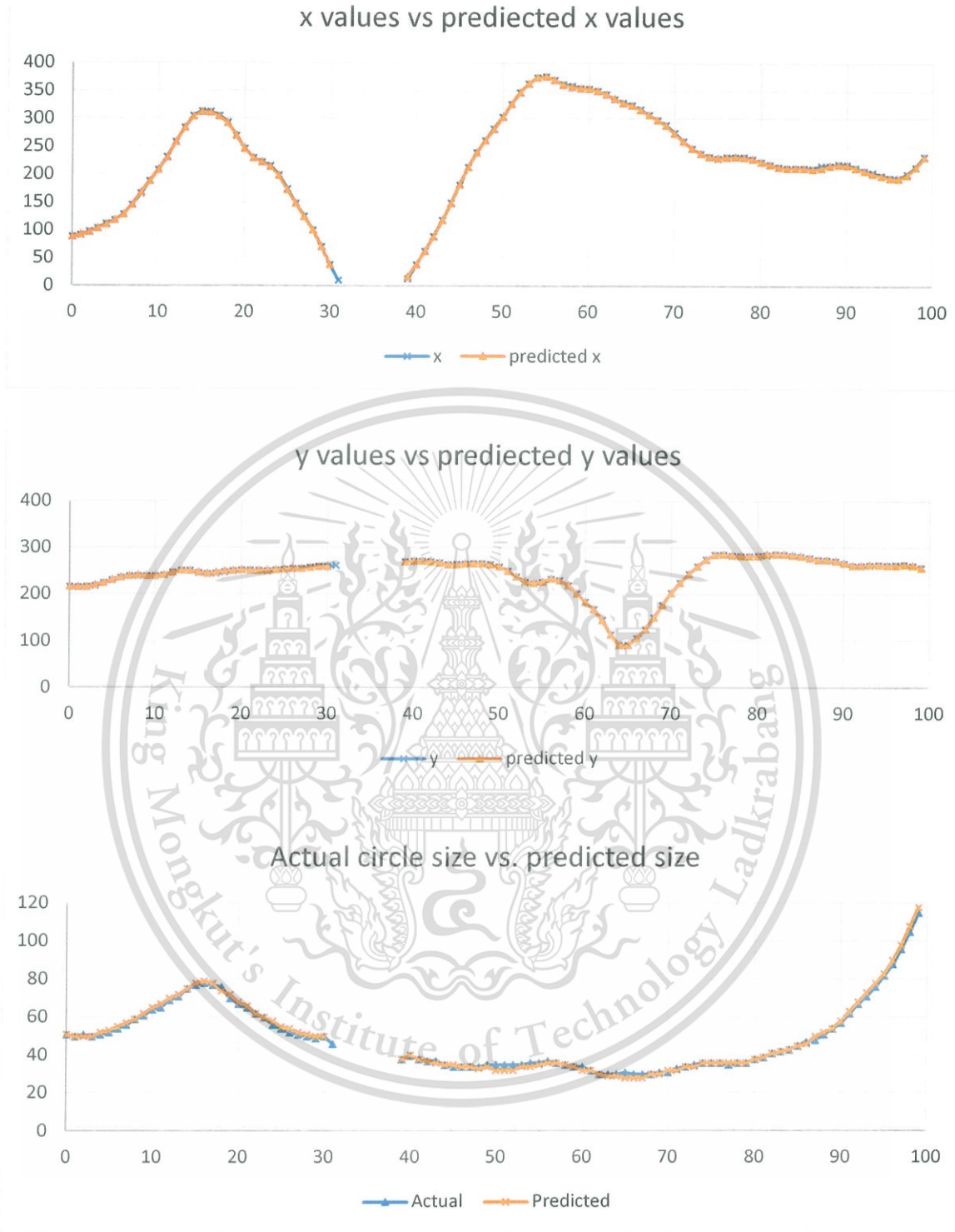


Figure 4.5 Ground truth and predicted values of center (x, y) and size of the largest circle in the marker from Connected Component Detection approach

Although the average error of largest circle size detection (Figure 4.5) is approximately 1 pixel, we still have some large error around the frame 50 and 65 where the marker moves quickly. Thus, the circle size would not be as reliable comparing to the case of Hough Circle Transform approach. On the other hand, the predicted position of the center of the largest circle also has the average error of approximately the same as in Hough Circle Transform case. Thus, we suspect that the center prediction from this approach would be sufficient enough as an input of distance approximation.

In comparison, as shown in Table 4.1, Hough Circle Transform provides higher accuracy. However, with the fact that the Hough Circle Transform missed 6 frames while the Connected Component Detection only missed 1 frame, we can conclude that Connected Component Detection approach is far more reliable for tracking.

Table 4.1 Average error and standard deviation of center and size predictions

	center-predict		size-predict		missing frames
	error (pixel)	stdev	error (pixel)	stdev	
Hough Circle Transform	1.02	0.82	0.51	0.64	6
Connected Component Detection	1.01	0.79	0.91	0.89	1

4.2.2 Speed Performance

For speed calculation, we experimented the two approaches, Hough Circle Transform and Connected Component Detection, on Raspberry PI 2 Model B+. In order to accurately measure the speed performance, we do not take images from camera but preload all images into memory first. Thus, in each loop, we only measured the time for pre-processing and actual marker detection. For both methods, we adjusted parameters in two manners. One is to maximize the accuracy and the other is to maximize the speed.

For the Hough Circle Transform approach, the highest accuracy we achieved is missing 6 frames while calculating at the speed of 6 frames per second. On the other hand, we can we can raise the speed of this approach to almost 10 frames per second with acceptable accuracy (16 frames of missing detection). On the other hand, The

Connected Component Detection achieved around 6 frames per second while missing only 1 frame. After some parameter adjusting for speed, we can achieve approximately 12 frames per second while missing only 4 frames. Note that more we cannot achieve any noticeably higher speed by from this approach by adjusting parameters. Thus, even though the Hough Circle Transform approach gives more precise location and size information, when it comes to speed performance, as well as detection accuracy, this approach is inferior to Connected Component Detection.

Table 4.2 Speed performance

	High Accuracy Setup		Low Accuracy Setup	
	FPS	Missed	FPS	Missed
Hough Circle Transform	2.54	6	9.71	16
Connected Component Detection	6.13	1	12.58	4

4.3 Result of Application to Real-time Navigation System

Lastly, we apply our proposed algorithm in the scenes captured from five different real-world locations as described in Section 4.1. We use Connected Component Detection scheme with limitation of 0.9 to 1.1 aspect ratio. We use 5 points black circle evaluation to evaluate the scenes. These scenes are complex, dynamic, and varied in lighting. There are two experiments in this section: marker detection and distance/angle calculation.

4.3.1 Rate of Detection

The first experiment was done to measure detection accuracy on 100 images where 50 of them are positive images and the other 50 are negative images.

Table 4.3 Detection rate.

	TP	TN	FP	FN
Detection rate (frames)	43	49	1	7

As shown in Table 4.3, the resulting accuracy provides 43 true positives (TP), 49 true negatives (TN), 1 false positive (FP) and 7 false negatives (FN). There are 8 frames incorrectly identify in this experiment comparing to only 1 frame in the previous experiment. This is to be expected since from the complexity and dissimilarity of the scenes.

In Figure 4.6, we show an example of one false positive and one false negative image along with their Otsu's thresholding results. The fault positive image (Figure 4.6 top) is a result of black cluster near the marker (shown in Figure 4.7 top-right). There is possible to eliminate this type of error by trying to narrow our search parameter such as aspect ratio. However, the performance from Table 4.3 should provide sufficient accuracy for the mobile navigation task.

The fault negative image (Figure 4.6 bottom) is a result of two conditions. First, the image is blurry due to focusing issue which make it hard to find the circle boundaries. Second there are light sources in the scene which make the capturing pixel appear very bright comparing to the rest of the scene. These problems can be lessen if we use better camera with faster focus or an adaptive thresholding algorithm. However, the better camera would require more power and the adaptive algorithm would slow down the speed.

Figure 4.7 provides examples of robustness of the proposed method. Our proposed method can positively detect the partially-shadowed marker (top-left), the bright maker with shaded background (top-right), the shaded marker with bright background (middle-left), the indoor maker with bright light source (middle-right), and the low-light, indoor marker with bright area (bottom). All of these scene are complicated with a lot of black clusters. Thus, the reported 92% accuracy from these scenes demonstrates the robustness to light and scene complication of our proposed method.

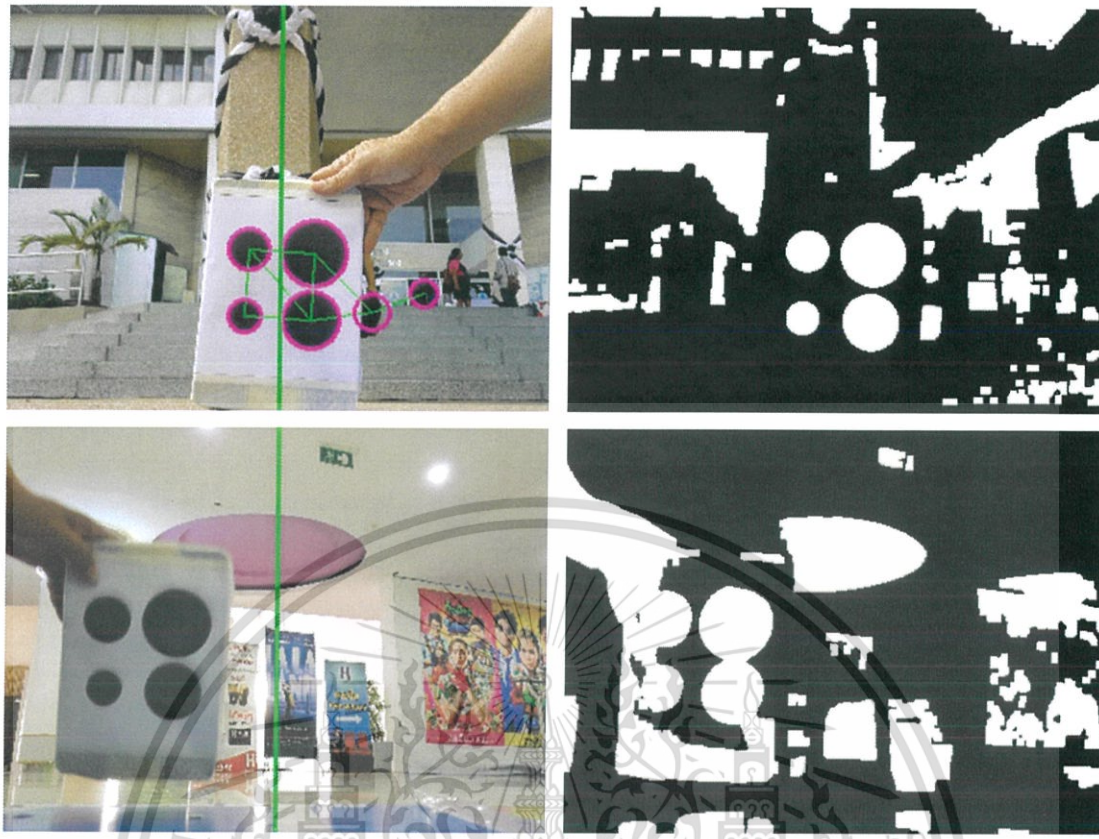


Figure 4.6 Example of a false positive and a false negative images with their corresponded pre-processing imaged.

4.3.2 Distance and Angle Computation

In the second experiment we computed distances and angles of the marker using the process described in the Section 3.3. As mentioned in the Table 4.3, we have 43 true positive images for this calculation. As a result, shown in Table 4.4, the distance detection contains an average error of 1.41 centimeters with standard deviation of 1.24. Considering the actual distance is approximately varying from 45 to 67 centimeters, the average error is computed to be at most 3% of the actual distance. For angle detection, the average error is 0.79 degrees with standard deviation of 0.58 while the actual angles are varies from 0 to 45 degrees.

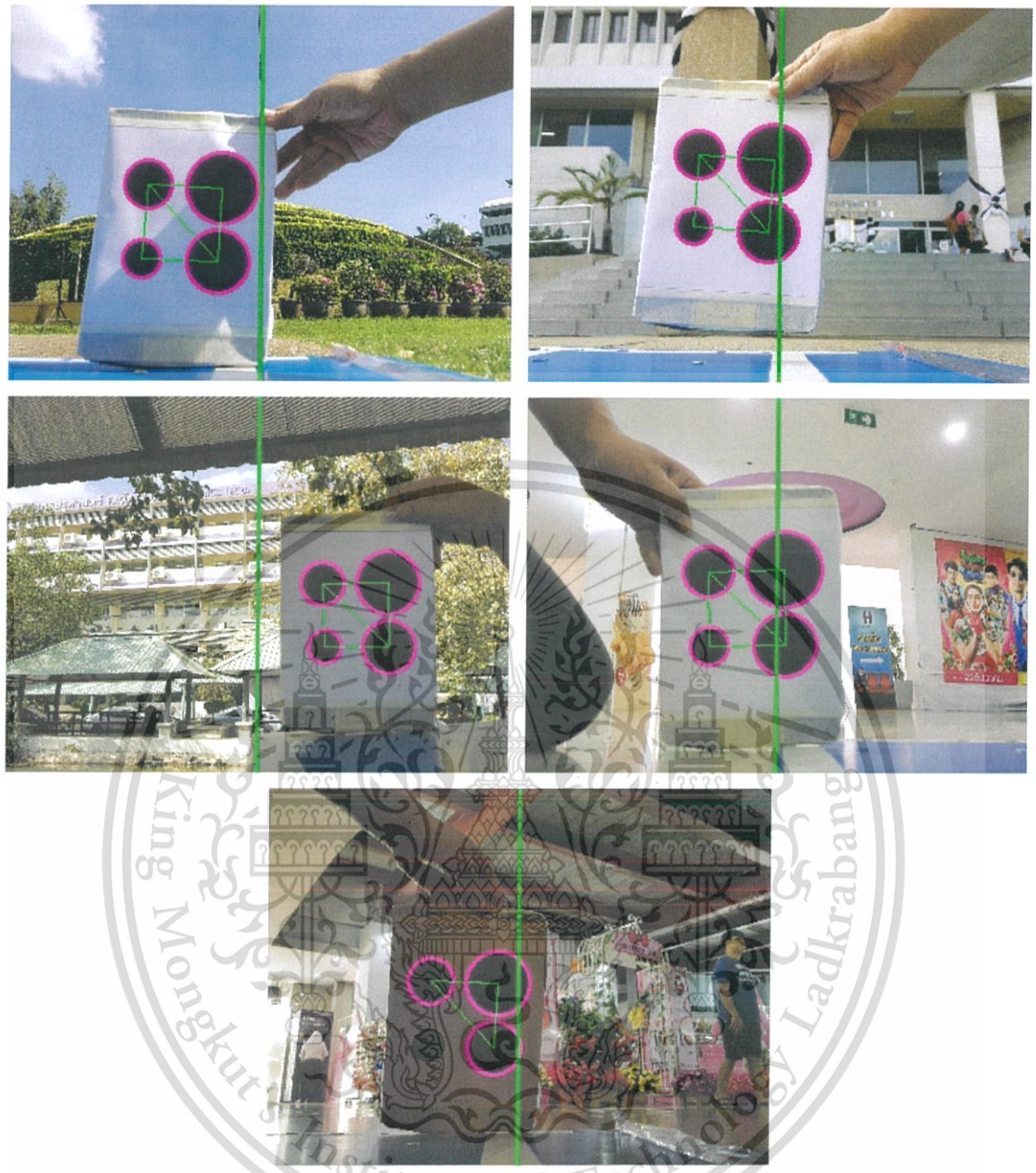


Figure 4.7 Examples of true positive in various lighting and complicated scenes

Table 4.4 Detection error and standard deviation of the error

	error	standard deviation
Distance (cm)	1.41	1.24
Angle (degree)	0.79	0.58

Chapter 5

Conclusion and Future Work

This thesis offers studied of an improved approach to an automatic navigation system based on visual information. This work utilized geometrics information which are shapes, position, and ratio of shapes, in order to allow fast detecting and filtering. The marker is carefully designed for fast and accurate detection. The general approach for detection can be describes follows. First, we utilized any fast circle detection with high recall since the system can quickly handle false positives while false negative can cause the system to miss the marker. Then, the system makes use of the known geometric properties of the marker to filter out any falsely detected circles in sufficient speed.

With the detected marker, we can calculate relative turn angle and distance between the attached camera and the marker in sufficient accuracy and speed. To give empirical evident, we perform the detection on clustered scene images with and without marker. Out of 100 frames, the best detection accuracy setup of our approach has missed only 1 frame. The best accuracy setup is based on using Connected Component Detection to find candidate for circle. The missing frame is a result of the faulty of Otsu's thresholding which happened when the marker is about to leave the scene.

For location accuracy, both Hough Circle Transform approach and Connected Component Detection approach perform relatively well. The relative error is approximately 3% of the circle size in both approach. In Hough Circle Transform approach, the detected size of the circle is determined by edge information. On the other hand, the detected circle size is based on where the Otsu's thresholding separate foreground pixels from the background ones. Although the two approaches using different methods for determine where the edge information is, they provide a very similar location accuracy.

For speed of detection, we provided empirical data by performing our proposed method on Raspberry PI 2 model B+. In this setup, Connected Component Detection approach significantly outperforms Hough Circle Transform approach. If we setup both approach for detection accuracy, Connected Component Detection approach can process

approximately 6 frames per second while the other approach only capable of processing approximately 2.5 frames per second. On the other hand, setting up the Connected Component Detection approach can process more than 12 frames per seconds while having better detection accuracy than the other approach. Having this result, it can be easily concluded that we should consider the Connected Component Detection approach for both speed and accuracy.

To calculate distance from size of the detected circle, we need to calibrate to get a parameter k which is an approximation of a multiplication result of focal length of the camera and the real-world size of the marker. This k can be calibrated by measuring multiple detected circle sizes, d, couple with the actual distance, D. To lessen the effect of noise, we can calculate k by averaging the product of each pair of D and d. This k must be calibrate for each hardware setup. In theory, we can look up both the focal length and the real-world size of the marker and use them to calculate for k. In pracetice, it is easier and provide a result with better confident just to approximate from measured data. For example, in our hardware system, the parameter k can be approximate to be 857. Therefore, if we measure the detected circle size, d, to be 35, we can approximate the distance, D, by calculating 857/35 which is just a little more than 24 inches.

Another parameter to be calibrated is the dx, a distance from the center of the detected circle to the center of the scene image. Again, this dx can be computed theorically by looking up camera parameter. Furthermore, it can also be calibrated in the way which we can solve for the focal length, f, of the camera. However, it is also easier just to measure dx from a range of θ and then use some interpolation such as linear interpolation in order to find out value of θ in between.

As illustrated in Section 4.3, our proposed scheme can be used in real-world scenarios with various lighting and complicated scenes. The reported detection accuracy rate in these scenarios is 92%. For the distance and angle calculation, the reported error is 1.41 centimeters and 0.79 degrees with the standard deviation of 1.24 and 0.58 respectively. This results suggest that our proposed scheme can be apply to the real-time navigation system where the scenes can be complicated and various in lighting. Whereas mentioned in Chapter 2, the previous fast tracking scheme would fail. For example, the

color tracking would fail in vary in lighting and MeanShift/CamShift would fail in our tested complicated scenes.

In conclusion, our proposed method is able to perform in a low computing power environment such as in Raspberry PI 2 Model B+ with sufficient speed and accuracy. This allows researchers as well as mobile robot developer to build a robot with a low battery consumption (from low computing power unit) while having capability to follow a designed target. This can open up various possibilities. For example, we can have a mobile robot following use around to help with carrying something or any other task.

One major flaw of our approach is that the marker has to appear almost parallel to the camera plane. The marker orientation can vary as much as we can still detect a circle which, from the current setup, the ratio of width and height must stay in range of 0.9 to 1.1. Hence, if we can design a marker or a 3D marker which is robust to orientation, the navigation framework base on this approach with new marker would improve significantly.

References

[1] Bengler, K., Dietmayer, K., Farber, B., Maurer, M., Stiller, C., & Winner, H. "**Three decades of driver assistance systems: Review and future perspectives.**" IEEE Intelligent Transportation Systems Magazine, vol. 6(4), 2014. pp. 6-22.

[2] Jin, S., Zhang, J., Shen, L., & Li, T. "**On-board vision autonomous landing techniques for quadrotor: A survey".** In Control Conference (CCC), 2016 35th Chinese. TCCT. 2016. pp. 10284-10289.

[3] Raja, P., & Pugazhenthi, S. "**Optimal path planning of mobile robots: A review.**" International Journal of Physical Sciences, 7(9), 2012. pp. 1314-1320.

[4] Marr, D. "**Early processing of visual information.**" Philosophical Transactions of the Royal Society of London B: Biological Sciences, 275(942), 1976. pp. 483-519.

[5] Mckerrow, P. "**Introduction to robotics.**" Addison-Wesley Longman Publishing Co., Inc. 1991.

[6] Leonard, J. J., & Durrant-Whyte, H. F. "**Mobile robot localization by tracking geometric beacons.**" IEEE Transactions on robotics and Automation, vol. 7(3), 1991. pp. 376-382.

[7] Leonard, J. J., & Durrant-Whyte, H. F. "**Simultaneous map building and localization for an autonomous mobile robot.**" In Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on. 1991. pp. 1442-1447.

[8] Fuentes-Pacheco, J., Ruiz-Ascencio, J., & Rendón-Mancha, J. M. "**Visual simultaneous localization and mapping: a survey.**" Artificial Intelligence Review, vol. 43(1), 2015. pp. 55-81.

[9] Kortenkamp, D., & Weymouth, T. "**Topological mapping for mobile robots using a combination of sonar and vision sensing.**" In Proceedings of the twelfth national conference on artificial intelligence (AAAI-94). MIT Press, 1994. pp. 979–984.

[10] Franz, M., Scholkopf, B., Georg, P., Mallot, H., & Bulthoff, H. "**Learning view graphs for robot navigation. Autonomous Robots**", vol. 5, 1998. pp. 111–125.

References (cont.)

[11] Sharp, P. E. **"Computer simulation of hippocampal place cells"**. Psychobiology, vol. 19(2), 1991. pp. 103–115.

[12] Burgess, N., Recce, M., & O'Keefe, J. **"A model of hippocampal function"**. Neural Networks, vol. 7, 1994. pp. 1065–1081.

[13] Betke, M., & Gurvits, K. **"Mobile robot localization using landmarks."** In Proceedings of the IEEE international conference on robotics and automation (ICRA-94), vol. 2. IEEE Press, 1994. pp. 135–142.

[14] Madsen, C., Andersen, C., & Rensen, J. **"A robustness analysis of triangulation-based robot self-positioning."** In Proceedings of the 5th symposium for intelligent robotics systems. 1997.

[15] Thrun, S., & Leonard, J. J. **"Simultaneous localization and mapping."** In Springer handbook of robotics, 2008. pp. 871-889.

[16] Durrant-Whyte, H., Bailey, T. **"Simultaneous localization and mapping (SLAM): Part I the essential algorithms"**, IEEE Robot. Autom. Mag. vol. 2 2006. pp. 99–110.

[17] Magnabosco, M., Breckon, T.P. **"Cross-Spectral Visual Simultaneous Localization And Mapping (SLAM) with Sensor Handover."** Robotics and Autonomous Systems vol. 63(2) 2013. pp. 195–208.

[18] Pupilli, M., & Calway, A. **"Real-Time Camera Tracking Using a Particle Filter."** In British Machine Vision Conference (BMVC). 2005.

[19] Eade, E., & Drummond, T. **"Scalable monocular SLAM."** In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (1), 2006. pp. 469-476.

[20] Yoshimi, T., Nishiyama, M., Sonoura, T., Nakamoto, H., Tokura, S., Sato, H., Ozaki, F., Matsuhira, N. & Mizogushi, H. **"Development of a Person Following Robot with Vision Based Target Detection,"** Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2006), 2006. pp. 5286-5291.

References (cont.)

[21] W. han Yun, D. Kim, and J. Lee, "Person following with obstacle avoidance based on multi-layered mean shift and force field method," in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2010. pp. 3813 –3816.

[22] Doisy, G., Jevtic, A., Lucet, E., & Edan, Y., "Adaptive person-following algorithm based on depth images and mapping, in Proceedings of the Workshop on Robot Motion Planning: Online, Reactive, and in Real-time," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012). 2012.

[23] Tarokh, M., & Merloti, P., "Vision-based robotic person following under light variations and difficult walking maneuvers," Journal of Field Robotics, (27), no. 4, 2010. pp. 387–398.

[24] Satake, J. & Miura, J. "Robust stereo-based person detection and tracking for a person following robot." in ICRA Workshop on Person Detection and Tracking. 2009.

[25] Kalman, R. E. "A new approach to linear filtering and prediction problems." Journal of basic Engineering, vol. 82(1), 1960. pp. 35-45.

[26] Ljung, L. "Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems." IEEE Transactions on Automatic Control, vol. 24(1), 1979. pp. 36-50.

[27] Rasmussen, C., Toyama, K., Hager, G. D. "Tracking objects by color alone." DCS RR-1114, Yale University. 1996.

[28] Fukunaga, K., Hostetler, L. D. "The estimation of the gradient of a density function, with applications in pattern recognition." Information Theory, IEEE Transactions on, vol. 21(1), 1975. pp. 32-40.

[29] Comaniciu, D., Meer, P. "Mean shift: A robust approach toward feature space analysis." Pattern Analysis and Machine Intelligence, vol. 24(5), 2002. pp. 603-619.

References (cont.)

[30] Allen, J. G., Xu, R. Y., Jin, J. S. **"Object tracking using CamShift algorithm and multiple quantized feature spaces."** In Proceedings of the Pan-Sydney area workshop on Visual information processing. 2004. pp. 3-7.

[31] Cheng, Yizong. **"Mean Shift, Mode Seeking, and Clustering."** IEEE Transactions on Pattern Analysis and Machine Intelligence. IEEE. vol. 17(8). 1995. pp. 790–799.

[32] Bradski, G.R., **"Real time face and object tracking as a component of a perceptual user interface,"** Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on , vol., no., 1998. pp.214,219, 19-21.

[33] Kim,J., Jung,Y., Lee,D., and Shim,D.H., **"Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera."** In Unmanned Aircraft Systems (ICUAS), 2014 International Conference on 2014. pp. 1243-1252.

[34] Wenzel, K. E., Masselli, A., and Zell, A., **"Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle."** Journal of intelligent and robotic systems, vol. 61(1-4), 2011. pp. 221-238.

[35] Ballard, D. H., **"Generalizing the Hough transform to detect arbitrary shapes."** Pattern recognition, vol. 13(2), 1981. pp. 111-122.

[36] Dotenco, S., Gallwitz, F., and Angelopoulou, E. **"Autonomous Approach and Landing for a Low-Cost Quadrotor Using Monocular Cameras."** In Computer Vision-ECCV 2014 Workshops 2014. pp. 209-222.

[37] Lange, S., Snderhauf, N., and Protzel, P., **"A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments."** In Advanced Robotics, 2009. ICAR 2009. International Conference on. 2009. pp. 1-6.

References (cont.)

[38] Merz, T., Duranti, S., and Conte, G., **"Autonomous landing of an unmanned helicopter based on vision and inertial sensing."** In Experimental Robotics IX. Springer Berlin Heidelberg. 2006. pp. 343-352

[39] Illingworth, J., & Kittler, J. **"A Survey of Efficient Hough Transform Methods."** In Alvey Vision Conference. 1987. pp. 1-8.

[40] Illingworth, J., & Kittler, J. **"The adaptive Hough transform."** IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. (5), 1987. pp. 690-698.

[41] Dillencourt, M. B., Samet, H., & Tamminen, M. **"A general approach to connected-component labeling for arbitrary image representations."** Journal of the ACM (JACM), vol. 39(2), 1992. pp. 253-280.

[42] He, L., Chao, Y., & Suzuki, K. **"A run-based two-scan labeling algorithm."** IEEE Transactions on Image Processing, vol. 17(5), 2008. pp. 749-756.

[43] Otsu, N. **"A threshold selection method from gray-level histograms."** Automatica, vol. 11(285-296), 1975. pp. 23-27.

[44] Cortes, C., & Vapnik, V. **"Support-vector networks."** Machine learning, vol. 20(3), 1995. Pp. 273-297.

[45] Knerr, S., Personnaz, L., and Dreyfus, G. **"Single-layer learning revisited: a stepwise procedure for building and training a neural network."** In J. Neurocomputing: Algorithms, Architectures and Applications, Neurocomputing, vol. 68, 1990. pp. 41–50.

[46] Siriteerakul, T., Boonjing, V., & Gullayanon, R. **"Character classification framework based on support vector machine and k-nearest neighbour schemes."** SCIENCEASIA, vol. 42(1), 2016. pp. 46-51.

[47] Arya, S., Mount, D., Netanyahu, N., Silverman, R., and Wu, A. **"An optimal algorithm for approximate nearest neighbor searching fixed dimensions."** J. ACM 45, Vol. 6, 1998. pp. 891-923.

Appendix A

Publish Articles

International Journal

Siriteerakul, T., Boonjing, V., & Gullayanon, R. **"Character classification framework based on support vector machine and k-nearest neighbour schemes."** SCIENCEASIA, 42(1), 46-51, 2016.

International Conferences

Siriteerakul, T., & Gullayanon, R. **"Robust tracking algorithm with designed marker for limited-power computer."** In 2016 8th International Conference on Knowledge and Smart Technology (KST) (pp. 245-248). IEEE, February 2016.

Siriteerakul, T., & Gullayanon, R. **"Fast Tracking Algorithm for Designed Marker."** In Recent Advances in Information and Communication Technology 2016 (pp. 293-299). Springer International Publishing, 2016.

Appendix B

Second Dataset Detection

Figure B.1 Scenes with partial shadow on the marker

Appendix B (cont.)

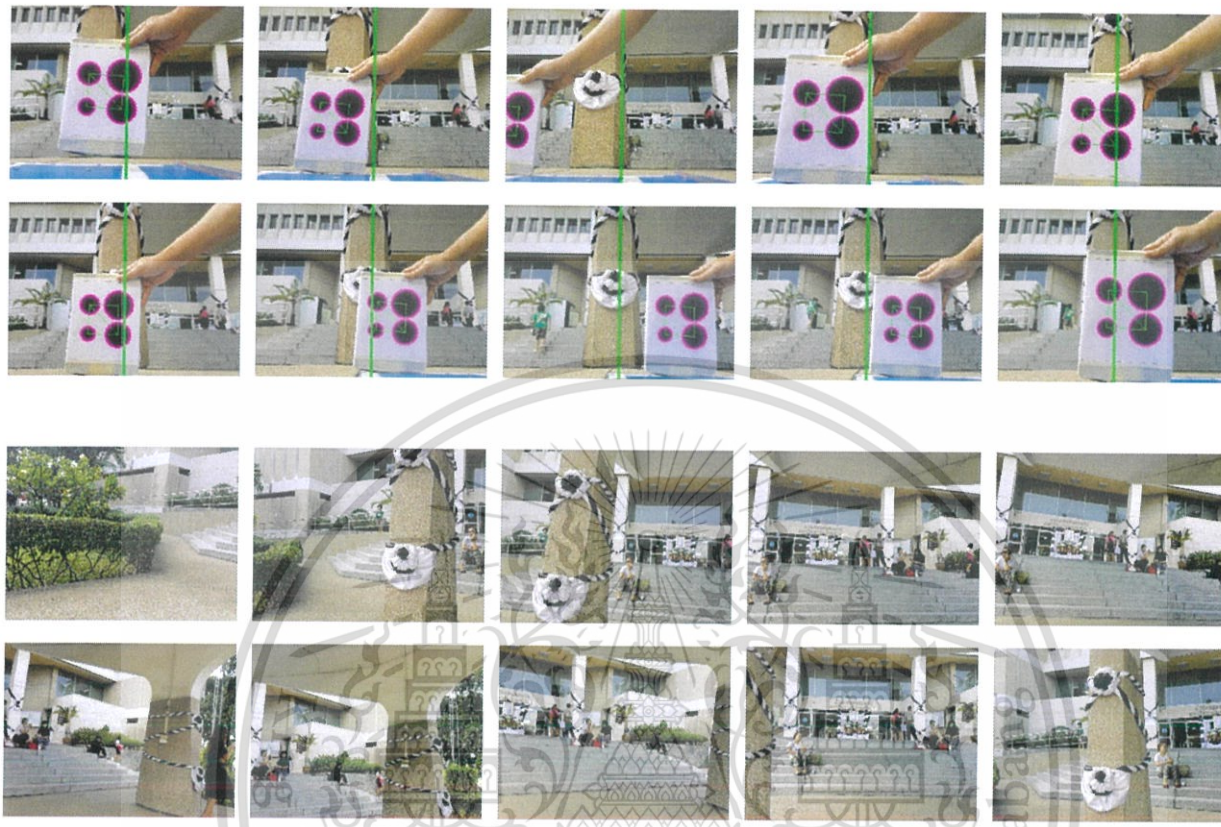


Figure B.2 Scenes with the bright marker with shadow on the background

Appendix B (cont.)

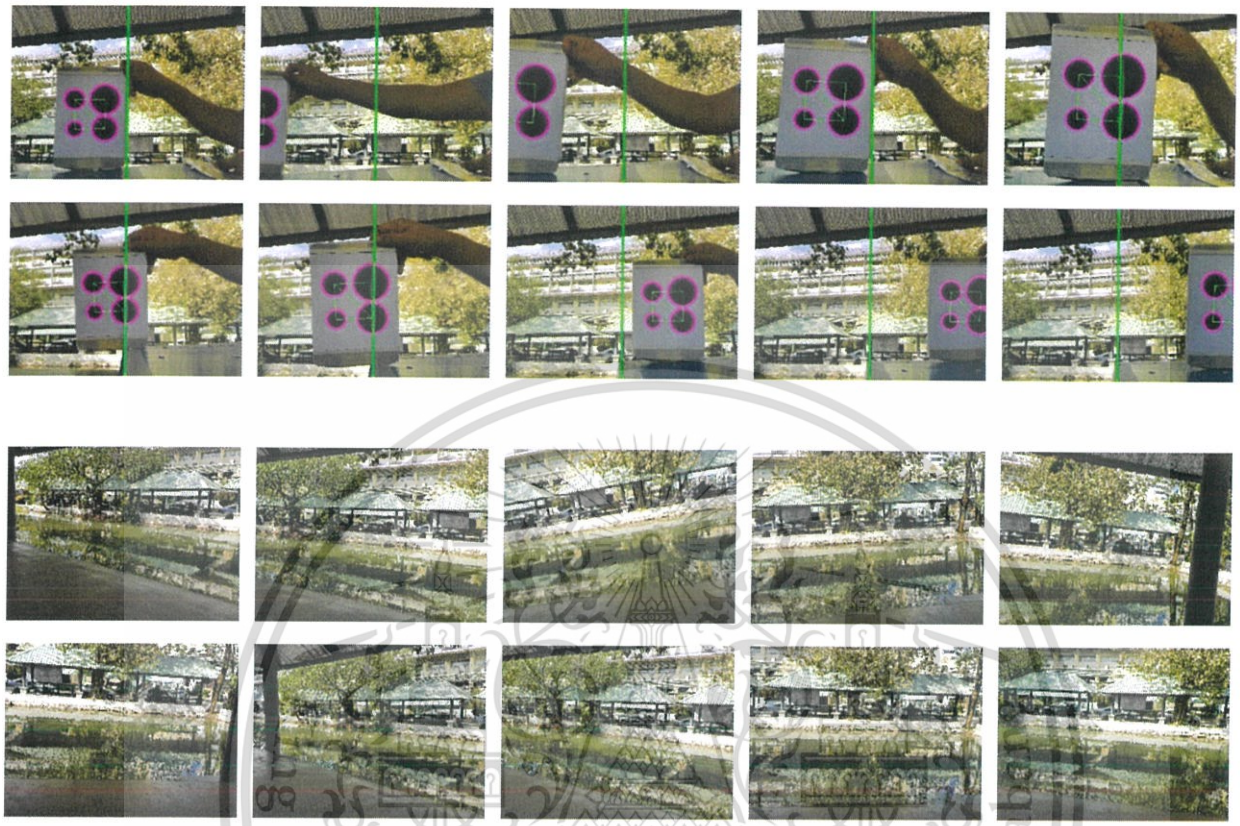


Figure B.3 Scenes with the marker in shadow with a bright background

Appendix B (cont.)

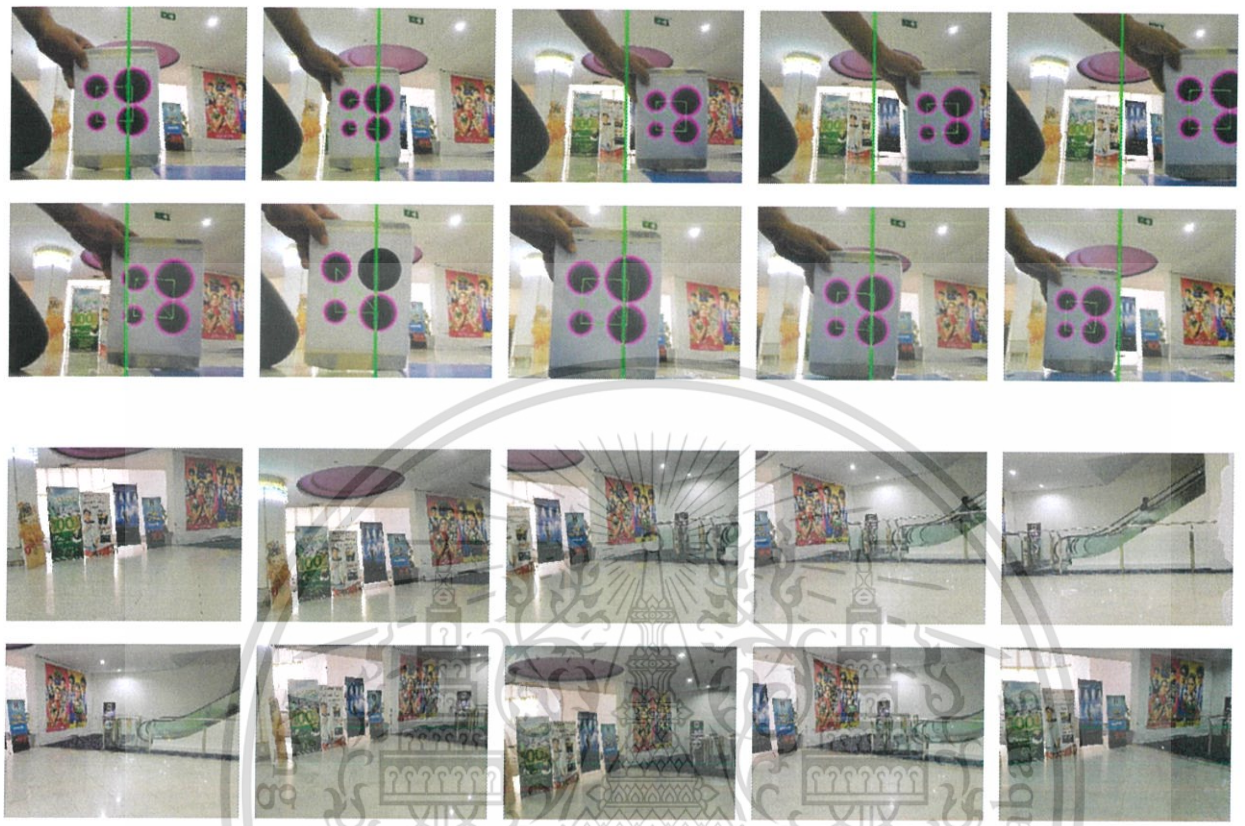


Figure B.4 Indoor scene with bright light source.

Appendix B (cont.)

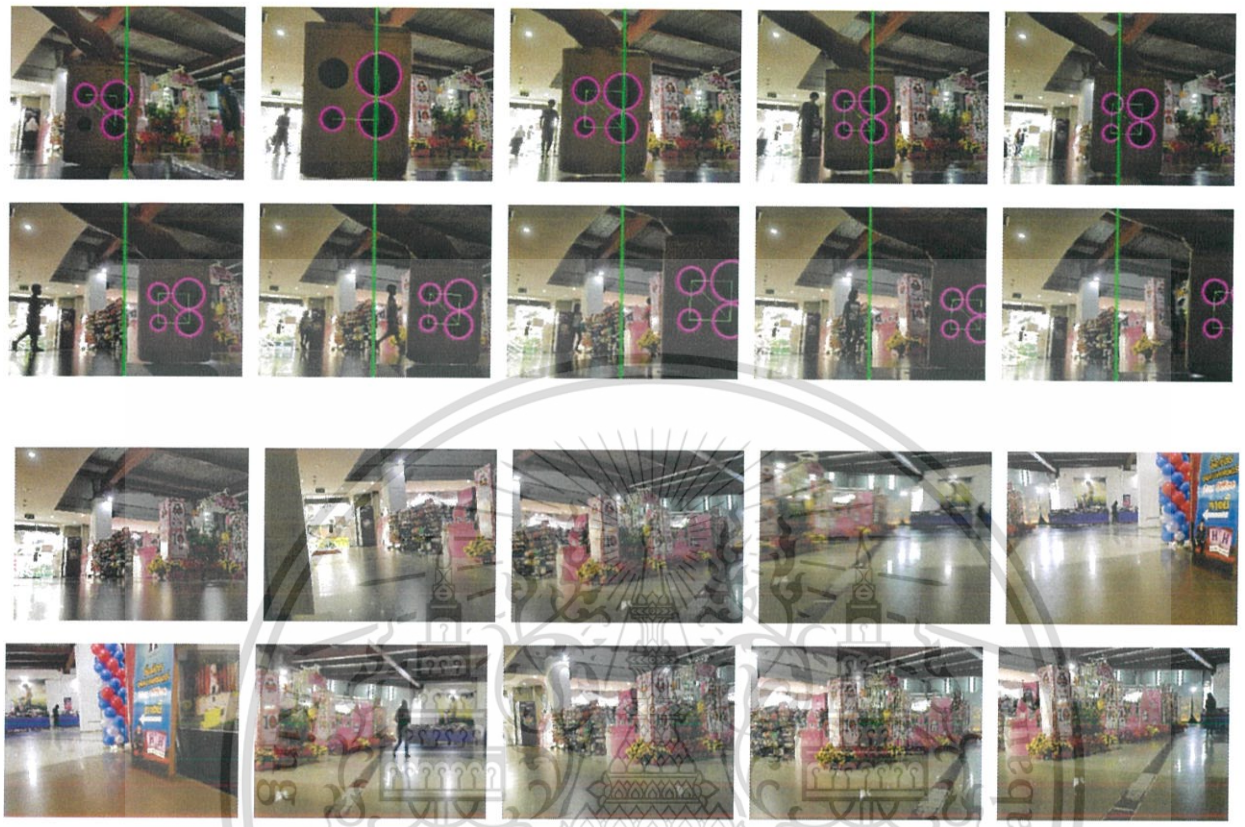


Figure B.5 Scenes with shadow and bright areas.

Biography

Name-Last name	Mr. Teera Siriteerakul
Date of Birth	25 August 1975
Address	333/93 M.11 Anaville, Chalongkrung Rd. Lumplatew
	Ladkrabang, Bangkok 10520.
Educational Background	2004 M.S. in Information Technology (Software Engineering)
	2001 M.S in Applied Mathematics
	1999 B.S. in Mathematics