TESTING THE MINIMAL BOUNDED SPACE METHOD ON VISION-BASED DRONE NAVIGATION

YAP SENG KUANG

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

TESTING THE MINIMAL BOUNDED SPACE METHOD ON VISION-BASED DRONE NAVIGATION

YAP SENG KUANG

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF COMPUTER SICENCE

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: YAP SENG KUANG

Matric No: WOA160007

Name of Degree: Master of Computer Science (Applied Computing)

Title of Dissertation: Testing the Minimal Bounded Space Method on Vision-based Drone Navigation

Field of Study: Robotics

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 20 Mar 2021

Subscribed and solemnly declared before,

Witness's Signature Date: 24 Mar 2021

Name:

Designation:

TESTING THE MINIMAL BOUNDED SPACE METHOD ON VISION-BASED DRONE NAVIGATION

ABSTRACT

The object-based approach is the most common in developing navigation strategies for robots. The object-based approach focuses on segmentation, detection, annotation, and recognition of objects or markers in the scene. For a drone, this approach is popular with the utility of sensors such as laser, vision (mono and stereo), ultrasonic, Kinect, and others. To elicit the information required, often, the object-based approach relies on these sensors as a hybrid solution. Recently, the availability of the deep learning algorithm also encourages the object-based approach for drone navigation. A critical gap in the object-based approach is that the computational resources required are massive. For a drone, especially a low-cost one, this renders the object-based approach to simulation-based only works. The other, less common navigation strategy for robots is the space-based approach. In the space-based approach, there is no object learning. Without object learning, there is no advanced processing to perform object recognition or labeling.

The space-based approach is focused on computing the openings in the surrounding space. Recent works have experimented with the space-based approach for robot navigation (Azizul, 2013; Azizul & Khanil, 2017). The space-based method used is called the Minimum Bounded Space (MBS). The name of the method is obtained from trying to bound the spatial openings immediately to the robot. In the earlier work, Azizul (2013) tested the MBS on a mobile robot equipped with a laser sensor. There is no imaging involved, but the laser sensor does record depth information. The spatial openings are derived by analyzing occlusion information from the environment, which is available from the depth information. The laser robot is shown to navigate

autonomously by moving from one spatial opening to another in an indoor environment. In the later work, Azizul & Khanil (2017) experimented with the MBS on a mobile robot equipped with a camera. Imaging is involved, but the way they processed the image is not the same as the processes involved in object-based works. Floor segmentation and analysis become the basis for finding spatial openings immediately to the robot. Eliciting the openings in the indoor environment is achieved without depth information.

The results shown from these prior works are encouraging as they do not require complex processing. Furthermore, they show how MBS is successfully implemented for real-time robot experiments. Interestingly, they show the versatility of the MBS method for autonomous robot navigation with or without depth information. However, the MBS method has not been tested on a flying robot or the outdoor environment. In this work, I show how the MBS method is implemented as the navigation strategy for a low-cost drone. The drone used in this work is called the Parrot Bebop Drone, which is equipped with a camera on board. To complete this task, I have developed a new computer vision framework to elicit openings for the MBS. The testing done shows the MBS is useful for low-cost drones flying in an indoor and outdoor environment.

Keywords: Cognitive robotics, robot navigation, spatial representation, micro aerial vehicle

PENGUJIAN METOD *MINIMAL BOUNDED SPACE* UNTUK NAVIGASI DRON VISUAL

ABSTRAK

Pendekatan berdasarkan objek adalah pendekatan paling umum dalam mengembangkan strategi navigasi robot. Pendekatan berdasarkan objek memfokuskan pada segmentasi, pengesanan, anotasi dan pengenalan atau penanda objek dari persekitaran. Pendekatan ini popular untuk dron dengan kegunaan sensor-sensor seperti berjenis laser, penglihatan (mono dan stereo), ultrasonik, Kinect dan lain-lain. Sering kali pendekatan berasaskan objek ini bergantung pada sensor-sensor tersebut sebagai suatu penyelesaian hibrid. Baru-baru ini, ketersediaan algoritma *deep learning* juga telah mendorong pendekatan berasaskan objek untuk navigasi dron. Satu jurang penting dalam pendekatan berasaskan objek adalah sumber perkomputeran yang diperlukan adalah berat. Untuk dron, terutamanya dron berkos rendah, pendekatan berasaskan objek untuk navigasi adalah cenderung ke arah penghasilan kerja-kerja bersifatkan simulasi. Strategi lain yang kurang popular untuk navigasi robot adalah pendekatan berasaskan ruang. Dalam pendekatan berasaskan ruang, tidak ada pembelajaran objek. Ini bermaksud tidak ada pemprosesan lanjutan untuk melakukan pengecaman atau pelabelan objek.

Pendekatan berasaskan ruang memfokuskan pada perkomputeran bukaan ruang di sekitar robot. Kerja-kerja terkini (Azizul, 2013; Azizul & Khanil, 2017) telah bereksperimen dengan pendekatan berasaskan ruang untuk navigasi robot. Kaedah berasaskan ruang yang digunakan dipanggil *Minimum Bounded Space* (MBS). Nama kaedah itu diperoleh daripada cubaan untuk menjangka ruang bukaan minimum bagi sesuatu robot. Azizul (2013) telah menguji kaedah MBS pada robot mudah alih yang dilengkapi dengan sensor berjenis laser. Tidak ada pengimejan yang terlibat tetapi sensor berjenis laser tersebut dapat merakam maklumat jarak. Maklumat bukaan ruang

adalah diperoleh dengan menganalisa maklumat oklusi yang diekstrak dari maklumat jarak. Robot berjenis laser tersebut berupaya melakukan navigasi secara autonomi dengan berpindah dari satu ruang terbuka ke ruang terbuka yang lain di dalam satu persekitaran dalaman. Azizul & Khanil (2017) pula telah melakukan eksperimen kaedah MBS pada robot mudah alih yang dilengkapi dengan kamera. Proses ini melibatkan pengimejan tetapi cara mereka memproses imej adalah tidak sama dengan proses yang terlibat dalam kerja-kerja yang menggunakan pendekatan berdasarkan objek. Segmentasi dan analisis lantai menjadi asas untuk membantu robot mencari bukaan ruang. Bukaan ruang di persekitaran dalaman adalah dicapai tanpa maklumat jarak.

Kerja-kerja ini juga telah menunjukkan bagaimana kaedah MBS boleh mejayakan eksperimen robot dalam masa nyata. Menariknya, mereka telah menunjukkan keupayaan serba boleh bagi kaedah MBS untuk melakukan navigasi robot secara autonomi dengan atau tanpa maklumat jarak. Walau bagaimanapun, kaedah MBS tersebut belum pernah diuji pada robot terbang mahupun di dalam persekitaran luaran. Dalam kerja ini, saya menunjukkan bagaimana kaedah MBS tersebut boleh dilaksanakan sebagai strategi navigasi untuk dron berkos rendah. Dron yang digunakan dalam kerja ini dinamakan sebagai Parrot Bebop Drone yang dilengkapi dengan kamera. Untuk menyelesaikan tugas ini, saya telah mengembangkan satu kerangka penglihatan komputer baru untuk mendapatkan bukaan ruang untuk kaedah MBS. Pengujian yang dilakukan menunjukkan bahawa kaedah MBS tersebut adalah berguna untuk navigasi dron berkos murah di dalam persekitaran dalaman atau luaran.

Kata kunci: Robotik kognitif, navigasi robot, perwakilan ruangan, kenderaan udara mikro

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep and sincere gratitude toward my supervisor turn friend, Dr. Zati Hakim Azizul Hasan, who has shown strong and continuous support throughout my study. This dissertation would have been impossible without her guidance, passion, patience, and immense knowledge. I will remember the long hours of discussions, and I do hope our friendship is long-lasting.

My appreciation towards my internal examiner and candidature panel, Prof. Dr. Loo Chu Kiong, my external examiner, Assoc. Prof. Dr. Siti Norul Huda Shekh Abdullah, and candidature panel, Dr. Muhammad Shahreeza Safiruz Kassim, for their valuable comments, which help improve my dissertation.

Finally, I would like to acknowledge with gratitude the support and love of my wife, Dr. Ang Lee Nah, to mentally and financially support the family and me for the past four years. She kept me going, and without her, this dissertation would not have been possible.

Thank you all,

March 2021,

Yap Seng Kuang

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

MAV	:	Micro Aerial Vehicles
GPS	:	Global Positioning System
VPS	:	Vision Positioning System
SLAM	:	Simultaneous Localization and Mapping
IMU	:	Inertial Measurement Unit
SURF	:	Speeded Up Robust Features
GBCM	:	Grid-Based-Color-Moments
UAVs	:	Unmanned Aerial Vehicles
UA	:	Unmanned Aircraft
UAS	:	Unmanned Aircraft Systems
RPVs	:	Remotely Piloted Vehicles
RPA	:	Remotely Piloted Aircraft
ASR	:	Absolute Space Representation
MFIS	:	Memory For Immediate Space
MBS	:	Minimal Bounded Space
DoG	:	Difference of Gaussians
SIFT	:	Scale Invariant Feature Transform
MSERs	:	Maximally Stable Extremal Regions
FAST	:	Features from Accelerated Segment Test
ArUco	:	Speeded up detection of squared fiducial markers
F-O-R	:	Frame-Of-Reference
LTI	:	Linear Translation Invariant

CHAPTER 1: INTRODUCTION

1.1 Drone navigation

Drones are not anymore limited to military applications. A large amount of small drone is not available for consumer use. Small flying drones are used for recreational, photography, video taking, delivery, construction monitoring, sensing, surveillance, and lately for extreme circuit competition. Small flying drones often include GPS units, gyroscopes, accelerometers, wireless communication capability, and a camera, making them ideal for outdoor applications. Nevertheless, indoor applications such as in-building delivery, automated inventory, and the search and rescue operations, particularly in an accident or a disaster (e.g., fire, flooding, and earthquake), are growing in interest.

Most of this small flying drone is not fitted with any global positioning system (GPS) and often suffers a frequent signal loss. Therefore, limiting small flying drone to low operation range and are usually remotely controlled. An autonomous navigation algorithm can allow a small flying drone to avoid crashing into obstacles, especially in a situation where a signal loss occurs, and the drone cannot stop instantly. An autonomous navigation algorithm can also be designed to reconnect with the remote controller when required. Secondly, if a small flying drone has to operate in an unstructured and a priori unknown environment, an extra autonomous navigation strategy layer on top of a more sophisticated layer, such as simultaneous localization and mapping (SLAM), can act as a fail-safe mechanism to help the drone to avoid obstacles.

In performing autonomous navigation, any robotics platform must first gather all its sensor data, interpret them, and make use of the interpretation for decision making. For a small flying drone, a monocular camera is one of the most common sensors. A monocular sensor is a passive sensor, which does not interfere with other sensors and provides the first-person view (FPV) sensation to the operator. From a commercial perspective, small drones are prevalent due to its low-cost. Currently, there are millions of this type of drone distributed all over the world. Therefore, developing an autonomous navigation algorithm for a small flying drone with a monocular camera becomes an active research area.

1.2 Motivation from a space-based approach

Azizul (2013) published a novel method for autonomous robot navigation called the minimal bounded space (MBS). The primary feature of this approach is to program a robotics platform, with its particular perceptual sensor, to interpret the environment by searching for spaces immediately to the robot. Once space is identified, the robot is instructed to move into space. The robot then performs exploration of the environment by moving into a series of consequent spaces identified by the MBS. Following the work, Azizul & Yeap (2015) shows how a mobile robot with a laser sensor performs autonomous navigation of an indoor environment using the MBS. The mobile robot used is a Pioneer 3DX equipped with a 180 degrees laser scanner with 360 laser beams that can achieve up to 30 meters depth accuracy.

In 2017, Azizul & Khanil extended the MBS algorithm by testing on a low-cost mobile robot with a single camera. The low-cost mobile robot is called Rovio, a (now discontinued) toy remote-controlled car manufactured by WowWee. For implementation, the Rovio has been reconfigured with Python and OpenCV to run the

MBS algorithm via WiFi connection. With the camera as the perceptual sensor, image processing is used to extract spaces from the environment. The Rovio can move into the series of spaces identified by the visual-fed MBS performing autonomous navigation in an indoor environment. The result shown is similar to the laser robot (Azizul & Yeap, 2015). However, what made the result significant is, this time, the MBS is performed when depth information is not available. Unlike a rangefinder like a laser sensor, the mono-camera is not capable of measuring distance.

While it has been encouraging that the MBS method can cross the perceptual sensing platform, i.e., performed autonomous navigation when depth is available or not, the MBS has never been tested in an outdoor environment. Prior experimental setups could not include outdoor testing due to the physical properties of the Rovio, and the 30-meter laser sensor range of the Pioneer 3DX robot fell short for outdoor testing. Inspired by this gap, the work in this dissertation focuses on testing the MBS in both indoor and outdoor environments.

In fulfilling the requirement of testing indoor and outdoor environments, careful selection of the appropriate robotics platform is required. From the physical aspect, a flying drone is attractive for the task, so long the indoor environment is spacious and has a relatively comfortable height for the ceiling. Furthermore, a flying drone is usually equipped with a mono-camera. With a mono-camera, flying the drone will require the non-depth MBS deployed in Azizul & Khanil (2017) 's Rovio.

The significant difference between this work and the work done in Azizul & Khanil (2017) is the visual feed will come from different vantage points. In Rovio, the camera is forward-facing and is about 15cm off the ground. At that height, the floor becomes significant in the Rovio visual scanning and has value in the decision making. In the visual flying drone, the camera is collecting feed at whatever height the drone is programmed to fly. The floor may or may not be significant to its visual feed, thus may or may not be influential in the decision making. The difference in the visual feed means adopting the MBS algorithm from Rovio to the flying drone cannot be straightforward. The drone is also susceptible to erratic hovering, unlike Rovio, which is stable on the ground. In terms of movement, the Rovio can drive forward, backward, and rotate but has only the planar axis to move. On the other hand, the drone can climb, hover, and descend, with horizontal and vertical axes to move. These different mechanisms of a mono-visual drone are some of the factors that will be taken into consideration in implementing the MBS algorithm.

1.3 Problem statement

The space-based approach, such as the MBS navigation method, shows promising results when tested on real-time mobile robots. Interestingly, it is capable of guiding robots for autonomous indoor navigation with or without depth information. However, the MBS algorithm has not been tested in an outdoor environment. In this work, I am interested in testing the MBS method for autonomous navigation in an indoor and outdoor environment using a mono-visual flying drone.

1.4 Research questions

The following research questions should be considered for the study:

1. How can one implement the MBS method on a mono-visual flying drone?
2. How can one segment spaces from a drone mono-visual feed?
3. How does the mono-visual drone perform using the MBS navigation method?

1.5 Objectives of the study

The following objectives are defined to answer the research questions:

1. To implement the MBS navigation method on a mono-visual drone.
2. To develop a new segmentation process for the MBS method.
3. To test the performance of the MBS navigation method on a visual drone.

1.6 Scopes of the study

The following scopes are described to accommodate the objectives of the study:

1. The drone must not utilize any depth perceiving sensor
2. Techniques such as the SLAM method are not considered to estimate the position of the drone
3. The drone height is fixed at 5 feet
4. Sufficient lighting, natural or artificial, is required during the experiment.
5. Human-made obstacles are allowed to ease experiment set up.

1.7 Significance or relevance of the study

The utility of flying drones is essential for tasks such as inspection, surveillance, and mapping. Although most of these are outdoor tasks, there is an increasing demand for indoor operation, particularly in warehouses and factories. For an indoor environment, maneuvering through narrow spaces can be challenging for a drone. Various depth sensors have been used for accurate obstacle detection and avoidance, such as LIDAR, sonar, stereo camera, and infrared. However, these rangefinders are costly in terms of load and battery capacity, thus often not found on small drones.

Small drones are usually equipped with mono-camera. Without rangefinders, small drones cannot extract depth information. Therefore, a small drone relies heavily on accurate feature extraction such as the corner, edge, SIFT, and SURF features. Interest point detection is challenging to vision processing because it requires heavy computation, which limits the utility of small drone activity in a real-time environment.

Significantly, this work contributes to the first space-based approach for a visual flying drone without depth information for an autonomous indoor and outdoor navigation. The outcome of this work is described as follows:

1. A new MBS method implementation for a mono-visual drone navigating autonomously without depth information.
2. A new segmentation process for a mono-visual drone.

1.8 Dissertation organization

This dissertation is divided into five chapters. Chapter 1 includes a background overview of the study, a problem statement, and the objectives of the work. Chapter 2 evaluates the literature relevant to this work. In particular, the review focuses on the object-based and the space-based robot navigation approaches. Chapter 3 describes the research methodology of this work. In Chapter 4, the results are described and discussed. Chapter 5 concludes the work and discuss possible future works.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

Central to this chapter is the discussion of the object-based and the space-based approaches to visual drone navigation. In object-based navigation, various vision-related techniques popular to drone navigation are discussed, such as points of interest, depth information, deep learning, and extraction of markers. The discussion on the space-based approach focuses on successful cognitive strategies implemented for robot navigation, which some are not vision-centric. For completion, a review of the available robot navigation system precedes the chapter.

2.2 The robot navigation system

Robot navigation is the ability of ground or flying robots to travel through the environment. Robot navigation is a procedure to determine the safe and suitable route between an initial and an endpoint for a traveling robot. Three classical questions can briefly define problems related to navigation: “Where am I” “Where am I going” and “How do I get there?” (Leonard & Durrant-White, 1991). The first two questions can be answered by a suitable sensory system, while the third requires a practical planning system. The navigation systems are directly related to the sensors available on the robot and environmental structure.

Robot navigation can be achieved via three systems; behavior-based system, coordinates-based system, and hybrid system. A behavior-based system needs the robot to recognize the features of an environment through its sensors. The features include the robot to recognize, for example, corridors and doors, in searching for the destinations. The coordinate-based system depends on metrical maps of an environment to produce

routes to guide the robot, while the hybrid-based system mix features of both coordinate-and behavioral-based systems.

A sub-task for the robot navigation systems include local navigation or path discovery. A local navigation task considers problems such as path following, obstacle avoidance, and the method to traverse from a source to a destination within the robot's field of vision. A sound robot navigation system allows the utility of robots for possible applications like the search and rescue missions, stunts for action movies and photography, aerial inspection of buildings and bridges, military, and construction tasks.

There are three types of navigation, which are map-based, map-building, and mapless navigations. The map-based navigation system needs a map and can be subdivided into metric map-based and topological map-based systems. A metric map-based system requires an entire map of the environment before navigation. A topological map-based system consists of nodes that mark the most distinctive places in the environment, and the nodes are linked by lines that represent the distance between two consequent nodes. The mapless navigation system tracks feature among frames or visual cues derived from optical flow and the segmentation of an image. Without a map, there is no universal representation of the environment in mapless navigation. The environment is perceived as the robot follows certain features or recognized objects.

Currently, there are two systems used for robot navigation, namely satellite-based Global Positioning Systems (GPS) and image-based Vision Positioning Systems (VPS). The GPS contains twenty-four satellites that transmit encoded radio frequency signals. A ground-based receiver can calculate its position through the travel time of three satellites' radio frequency signals, including the satellites' momentary location

information, using an advanced trilateration method. The distance between the ground receiver and the three satellites hypothetically allows for the calculation of the receiver's latitude, longitude, and altitude. The GPSs are limited by one, the GPS receiver does not work 100 percent when out-of-coverage, and two, the climatic conditions and user location can also affect the quality of geolocalization, as the signal is not easily accessible when the weather is terrible, or when the user is indoor or trapped between tall buildings.

The VPS rises in popularity because of two factors. While one is the failure of many robotics applications in a GPS-denied environment, the second is because quality vision sensors are getting cheaper. Cheap visual sensing is a game-changer, which motivates the robotics community to apply active vision processing for robot navigation. To compute the VPS, one must work through the massive data flow created by the vision sensors. Furthermore, instead of relying on traditional positioning methods such as the dead reckoning, ultrasound, and inertial estimation, the VPS makes use of optical sensors to obtain localization. At the core of the VPS system is object detection and manipulation. Section 2.3 reviews relevant literature surrounding this visual-object approach in developing navigation algorithms for autonomous robots.

2.3 Object-based approach for robot navigation

Object-based navigation refers to methods of navigations that depend on the interpretation and manipulation of objects from the environment. For a human, the object here means a material thing that can be seen and touched. For a robot, the object here means a material thing from the robot surrounding that the robot can compute through its sensors. Partial to the robot processing an object is object segmentation. Object segmentation deals with separating an object digitally from its background and

sending the raw information of that object to a series of filters for inference. According to Fuentes-Pacheco (2015), the inferencing of an object can be achieved in two phases, called the detection and description. Detection deals with the image to obtain several salient elements, while description deals with feature vectors based on visual appearance in the image. The invariance of the descriptor changes the orientation and position, which will improve image matches and the data association processes. Table 2.1 lists the earlier descriptors used as salient descriptors in the literature.

The most commonly used descriptors for object recognition is the histogram-type (SIFT descriptor), proposed by Lowe (2004), with qualities such as invariant to rotation, translation, scale, partially invariant to lighting, and viewpoint changes. The modified version of SIFT is called PCA-SIFT and is proposed by Ke & Sukthankar (2004). The modified version is as robust and distinctive as SIFT, with less component vector. These descriptors eventually paved the way for recent works in an object-based approach for visual drone navigation, particularly in the use of descriptors for obstacle avoidance.

Table 2.1: List of earlier descriptors used as salient detectors

Reference	Salient Detectors
Artieda et al., 2009	Harris corner
Rosten & Drummond, 2006	Features from Accelerated Segment Test (FAST)
Bay et al., 2006	Fast-Hessian used on SURF (Speeded up Robust Features
Lowe, 2004	The difference of Gaussians (DoG) used on SIFT (Scale Invariant Feature Transform)
Mikolajczyk & Schmid, 2002	Harris-Laplace and Hessian-Laplace points detectors, and Harris-Affine and Hessian-Affine
Matas & Chum, 2002	Maximally Stable Extremal Regions (MSERs)
Harris & Stephens, 1988	Harris corner

Table 2.2: Object-based approach for robot navigation

Reference	Aim of Work	Result	Gap
Agrawal et al., 2015	Segment obstacles using optical flow's image features, use collision cone based guidance law for obstacles avoidance	Improved performance compared to existing optical flow-based obstacle avoidance methods	Simulation in MATLAB, simple environment, use a zig-zag flight pattern and difficult to avoid frontal obstacle avoidance
Lee et al., 2011	Detect the obstacle's outline using MOPS. Calculate spatial coordinates of feature point exist in the internal outline of the obstacles through SIFT. Combine result from both MOPS and SIFT to show 3D information of the obstacles	Reconstruct 3D information of the obstacles with using MPOS and SIFT feature points	Matching the MOPS and SIFT features require two images. SIFT matching depends on the lighting. SIFT points are affected by noise from a low-resolution camera. Gray image is used
Aguilar et al., 2017	Using an onboard monocular camera to capture images and compare images obtained in real-time with a database of obstacles that must be avoided. Using SURF for fast obstacle detection.	Improved successful flights ratio compare to other algorithms	Matching is based on the vectorial distance between descriptors of each feature point in both images. Feature points are affected by lighting
Al-Kaff et al., 2017	Use feature points with expansion ratios from the convex hull from consecutive frames. Decide collision by comparing the area ratio of the obstacles and the position of the drone. Performs avoidance maneuver.	Two types of experiments were conducted - the drone moving toward the obstacles and vice versa. Accuracy of obstacle detection is 97.4%	The movement required to generate pairs of feature points may not be stable from a drone. The method is only able to detect one obstacle at a time.
Zheng et al., 2017	Use optical flow and visual odometry with ground facing monocular vision to evaluate the speed and location of the drone. Use laser for obstacle avoidance.	Kalman filter eliminates high-frequency noise from optical flow. The visual odometer produces an average error of 0.2m and a max error of 0.5m.	A predefined environment with flooring covered with a square carpet with a side length of 1m. Custom build drone with Intel NUC i5, RPLIDAR, OV2710 camera module
Park et al., 2013	Build a depth map from stereo vision—further segment obstacles from the image frame. Use the	Drone able to maneuver around a single sphere-shaped object, and multiple rectangular-	The experiment in a virtual environment where obstacles stand out from the

	collision cone method to avoid obstacles.	shaped objects	background
Ramasamy et al., 2016	Use the LIDAR system to generate a 3D representation for obstacle avoidance and a remote pilot to operate drone near the ground	The system detects and calculates distances to each of the obstacles into 3D representation. The system modifies the original trajectory to avoid a collision	Complex algorithm to construct a 3D model of the environment. High computational cost with additional payload to work with commercial drones.
Nemati et al., 2015	Simulate a drone system equipped with inertial and sonar sensors to enable autonomous flight in a GPS-denied indoor environment with obstacles.	Detect obstacle using sonar sensors and generate waypoints for navigation	Require laser and Hector SLAM, flight around 4m height, and a map is available before the flight
Tarazona et al., 2014	Control velocity through ultrasound sensors, accelerometers to detect objects, and fuzzy controllers to prevent collisions in indoor spaces.	A controller checks the velocity and angle inclination to prevent a collision. The controller is deactivated when not in use.	The alarm requires accurate depth readings from a stable/non-hovering position.
Rambabu et al., 2015	Use a Kalman filter to fuse sonar and infrared data. Cascade PID position and velocity controllers.	The drone maintains at 60cm distance from the obstacles.	The testing environment has fixed lighting conditions. Infrared data is affected by sunlight and other lighting conditions
Kouris et al., 2018	CNN trained on image collected through a drone camera and annotated with distance measured with an ultrasonic sensor	Improved decision making with two-stream regression architectures handling high ambiguity cases via richer Spatio-temporal representation	Only detects trained objects, requires high-end hardware including GPUs to train and to execute CNN
Brzozowski et al., 2016	Use the magnetic field map to localize the drone. Use permanent magnets to modify the local magnetic field that could work as beacons.	The changing magnetic field along the corridor at different heights can be detected using permanent magnets, which can create large anomalies in the previously registered magnetic field.	Electronic types of equipment can affect the magnetic field. Requires prior preparation of magnetic maps, including all its anomalies

There are generally four types of obstacle avoidance methods proposed in the object-based approach, such as the point of interest, depth information, deep learning, and markers. Table 2.2 shows works that address an object-based approach for robot navigation.

2.3.1 Point of interest

An image space contains features that refer to a pattern or unique structure found in an image such as a point or edge. They are usually found in small patches in the image space and differs distinctly from its immediate surroundings by texture, color, or intensity. These small patches are sometimes called blob, corners, or edge pixels and usually considered as the point of interest of an image. Detection of point of interest in an image space usually has the following characteristics (Lindeberg, 2015):

a) It can be described through a mathematical formula and can be searched repeatedly in the image space under varying illumination or brightness
b) It has a well-defined position in the image space, with surrounding structure supports local information contents

Local features such as corners and edges are great image features for optical flow computations because while corners provide accurate movement information to track an object in 2D space, edges provide pixel information along one dimension. The SIFT (scale-invariant feature transform) is a feature detection algorithm in computer vision to detect and describe local features in images. Lowe (2004) proposed the SIFT as an image processing technique that creates an image pyramid and extracts features points to make it unaffected by size change. It is also relatively unaffected by rotation changes because it can extract the orientation of the feature points and generate feature vectors.

Due to the challenging nature of detecting local features when face with rotation and illumination problems, the SURF (speeded up robust features) is proposed. Even though the SURF has better performance in dealing with rotation invariant, blurring, and warm transformation, the SIFT still beats the SURF when scaling is required. The SURF has higher popularity when computational speed is a factor because the SURF utilizes an integral image and box filter, making it three times faster than the SIFT.

In robot navigation, the SIFT and SURF techniques are widely used in tracking obstacles. Sometimes, researchers combine the SIFT with the multi-scale-oriented patches (MOPS) to reconstruct 3D information of the obstacles. The MOPS is a process that identifies the same points among multiple images (Brown et al., 2005). For example, Lee et al. (2011) construct 3D information of obstacles by combining approximate 3D outlines of obstacles with 3D SIFT feature points. Two consecutive images are used to elicit the result. Their steps for each image include:

- Extract the MOPS feature descriptor using the Harris corner detector. Then match the MOPS features between the first and the second images to form approximate 3D outlines,
- Grab internal outline information using SIFT, and match features between the two images, and
- Merge results from the MOPS and SIFT to show the 3D information of the obstacles.

Combining the SIFT and the MOPS methods is considered efficient to approximate the outlines of the obstacles. Further 3D information of the obstacles can be obtained when the outlines approximation is combined with the SIFT feature points. However, two images are needed for distance estimation using the SIFT and MOPS combination, considering the displacement between the two consecutive images. Furthermore, the SIFT features depend heavily on the lighting conditions of the environments and also the resolution of the camera used. The limitations of the SIFT require Lee et al. (2011) to experiment using only gray images and perform in a simulated environment.

Agrawal et al. (2015) use the optical flow of image features to segment obstacles from the image. Once the obstacles are segmented, a collision cone-based guidance law is proposed for avoidance. By combining the Harris corner features and optical flow (Sharmin & Brad, 2012), the optical flows of the corner points are considered for segmenting the image. Images are scan from left to right following a window of 5 pixels width, to calculate the average optical flow vector. For each window, an average optical flow value that is more than a predefined threshold corresponds to an obstacle.

The work by Agrawal et al. (2015) demonstrates the correlation between local interest points and the corresponding obstacles. Using optical flow to track a moving obstacle is exciting, but not useful when the tracker is the one moving, such as the case with a flying drone. Similar to Lee et al. (2011), the experiments by Agrawal et al. (2015) is also simulation-based with the environmental setup done in MATLAB. The simulation-based solution may require further extension before ready to handle feature extraction in a real-time drone experiment.

Small, low-cost drones gain popularity in recent research because of its accessibility to the public. Aguilar et al. (2017) are among the earlier work using vision from a monocular camera onboard a small, low-cost drone for obstacle avoidance. They compare the SURF features obtained in real-time from drone images with a reference database containing obstacle information. They reported the main drawback of the SURF method that it is affected by lighting, especially with the drone's low-resolution imaging, where the SURF loses information if the pixel is too dark.

Al-Kaff et al. (2017) detect the change of size among the detected feature points and combine them with the expansion ratio of the convex hull constructed around the detected feature points from consecutive frames. Obstacles that have the probability of getting closer to the drone are extracted by calculating the area ratio of the obstacle and position of the drone. If a collision is decided, the drone is required to perform an avoidance maneuver by estimating the obstacle position in the image with the tracked waypoints. AL-Kaff et al. (2017) claim that this method mimics the human behavior of detecting the collision state of an approaching obstacle. However, similar to other feature points methods, two stable consequent images are required to generate pairs of feature points for accurate inferences. Obtaining stable consequent images is not straightforward with a small, low-cost drone, suffering from hovering instability.

To avoid using consecutive images for inferences, Zheng et al. (2017) propose a method to combine monocular vision with laser radar. Monocular vision is used for speed and location estimation, while laser radar is utilized for obstacle avoidance. In doing so, Zheng et al. (2017) achieve stable navigation with autonomous obstacle avoidance at the cost of payload increment. It is motivating that they experimented in real-time, but the environment set up includes predefined square carpeted flooring at a fixed length.

Table 2.3: Drone navigation with the point of interest

Reference	Aim of Work	Result	Gap
Agrawal et al., 2015; Lee et al., 2011; Aguilar et al., 2017; Al-Kaff et al., 2017; Zheng et al., 2017;	Detect image features using SURF, SIFT, edge, 3D points cloud, and segment obstacles from the background.	The estimated distance between drone and obstacles and perform navigation by avoiding obstacles. Stable movement is required to generate pairs of feature points	1. Points are unstable, and experiments often simulate in a computer, not real-time. 2. Optical tracking between scenes requires a stable source. 3. Floating drone often gives unstable imaging.

In summary, the detection of point of interest or local features is useful for tracking obstacles when the lighting condition is stable. Besides, the technique is only successful for feature tracking when two consecutive images are in use. The technique does not consider following the local features when the tracker is in motion. Also, the technique has an accuracy issue if the pair of images used are not taken from a stable point of view. Table 2.3 shows drone works, which are based on the point of interest.

2.3.2 Depth information

Depth information describes the measurement of the distance between a range finding sensor and a solid object that intersects the sensor signal. There are various sensors to extract depth information; the most common are rangefinders like the Light Detection and Ranging (LIDAR), laser, and the sonar or ultrasonic sensor. The LIDAR and laser are accurate instruments with an unprecedented angular resolution in a wide range of incidence angles. The higher cost over laser gives LIDAR a more extended range making it an ideal solution for obstacle detection and avoidance in the outdoor environment. Although inexpensive compared to the laser and LIDAR, the sonar has

many shortcomings, such as specular reflection, foreshortening, cross-talk, and limited field of view of 30 degrees.

The sonar limitations were considered by Tarazona et al. (2014) when designing an algorithm for a drone to navigate and avoid obstacles in indoor spaces. They added an accelerometer to estimate the speed and acceleration of the drone in comparison to its surrounding. A fuzzy controller manages input from the accelerometer. These two inputs are fused using the Kalman filter for altitude, distance, and acceleration estimation. When the sonar's time-of-flight principle detects obstacles, the fuzzy controller evaluates the velocity and the angle inclination that a drone must take to reduce linear velocity and prevent a collision. However, the method neglects to consider the narrow viewing field of the sensor that may incur false alarm to the fuzzy controller due to the drone unstable hover.

While the sonar has broader and more in-depth coverage compared to the infrared, it has a lower refresh rate, which is around 13Hz to 20Hz. In comparison, another rangefinder, the infrared, has a superior refresh rate of about 100-250Hz, a higher resolution with a narrower beamwidth, but has short coverage. These observations help Rambabu et al. (2015) improvise by choosing the Kalman filter to fuse the sonar and infrared data to obtain reliable depth information for obstacle detection. The combination overcomes the sensors' disadvantages, sonar for the indoor environment, and the infrared for the outdoor environment. The sensor has to be calibrated in each environment in which it will be used, as reflections magnitude varies in a different environment. The results show that the drone can maintain the desired distance of 60cm from the obstacle. However, they reported the drone experimented in indoor spaces without sunlight, whereas the infrared is heavily affected by sunlight.

Passive sensors such as vision can also be utilized to infer depth information. Park & Kim (2013) use stereo vision to obtain depth map information to detect obstacles. A depth map is an image showing the distance of an object from a viewpoint at the pixel level. Obtaining the depth requires the camera's focal length and baseline, and the disparity of the object. The collision cone approach is adopted to avoid collision between the drone and the detected obstacle, and the result is validated in numerical simulation. The simulation result shows that the drone can maneuver around a single sphere-shaped object and multiple rectangular-shaped objects.

In summary, the depth information is considered straightforward when it comes to obstacle detection and avoidance. However, extracting depth information requires sound-based rangefinders with a broader viewing field, long-ranging signal, and fast refresh rate. When optical-based rangefinders are being considered, one must not neglect how lighting condition affects them, and that they can penetrate through transparent objects. Stereo vision can extract depth, provided a depth map can be generated in real-time. Table 2.4 shows works in drone navigation, which relies on depth perception for obstacle avoidance.

Table 2.4: Drone navigation with depth perception

Reference	Aim of Work	Result	Gap
Ramasamy et al., 2016; Nemati et al., 2015; Tarazona et al., 2014; Rambabu et al., 2015; Brzozowski et al., 2016	Using LIDAR, sonar sensors, and infrared either alone or with fusions, to detect the distance between drone and obstacles and prevent collisions	Drone maintain the desire distance from obstacles and able to maneuver over obstacles	1. Require complex algorithm to process sonar and infrared input

2.3.3 Deep learning

Machine learning's tremendous advancement has enhanced the capabilities of visual navigation, especially Deep Neural Networks, which enable the development of end-to-end learning approaches. Convolutional Neural Networks, one of the variations of Deep Neural Networks, enable feature extraction over a broad set of learnable parameters, in place of handcrafted feature selection, since handcrafted feature selection suffers from low generalization capabilities. Convolutional Neural Networks act as an enabler for visual navigation in real-world environments that inherently demonstrate significant variation in visual appearance.

Kouris (2018) introduces a self-supervised convolution neural network (CNN) based approach for indoor robot navigation. For dataset generation, three pairs of sonar and infra-red sensors are mounted on the drone pointing towards different directions within its camera's field of view, to allow automatic annotation of all data samples. The drone is drive manually in a few indoor environments for data collections.

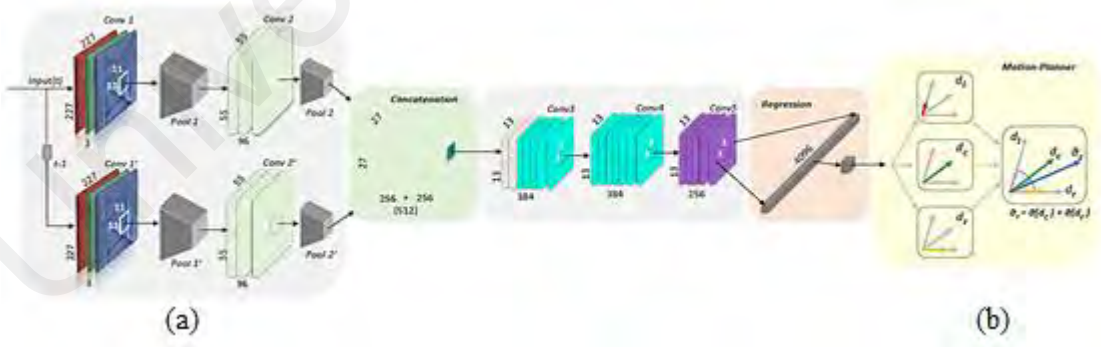


Figure 2.1: (a) Two-stream CNN architecture for Spatio-temporal feature extraction on regression tasks. (b) An instance of a local motion planner

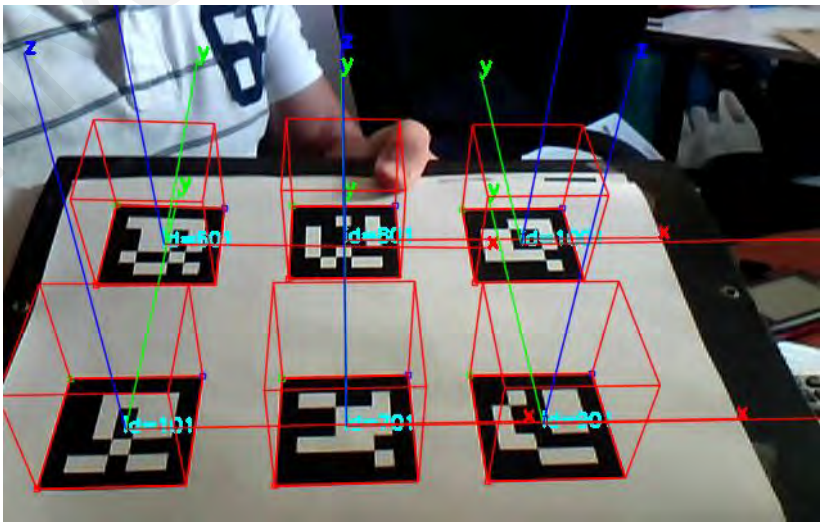
The overall CNN architecture is as shown in Figure 2.1. There are two streams to capture the temporal feature (image at time t and $t-1$). Each input image is separated into three overlapping rectangular windows, where each window has its' own distance to a possible collision, separately processed by CNN. The CONV layers of the proposed network were pre-trained on the imitation learning dataset for indoor navigation to enhance the generalization capability. The final result is to regress three distance-to-collision values. During inferences, these three distance-to-collisions are fed to a custom local motion planning algorithm, that concludes into a single control command modulating the drone's yaw and forward linear velocity. The proposed tow-stream regression architecture makes a more insightful decision in cases of high ambiguity. For example, when dealing with a reduced number of trackable features due to proximity to obstacles, by utilizing the learned, richer in information, Spatio-temporal representation of the visual input.

Briefly, some of the shortcomings of the experiment are that it is tested in environments similar to training data sets, which means that the network is not generalized for different indoor environments. Several high-end types of equipment are required for Kouris (2018) training and running the proposed network, including a desktop server equipped with an Intel Xeon E5-2630, 64GB of RAM, and a GTX1080 GPU used for training. To execute the CNN during the experiment, the hardware set up includes a laptop with minimal Intel Core i7 – 6700HQ, 16GB of RAM, and a GTX1070 GPU.

2.3.4 Markers

Markers provide visual navigation based on the detection of pre-installed markers. The advantage of using a marker is that it does not need prior feature acquisition from any objects in the environment other than the marker itself (Huang et al., 2012). Vision markers, such as the ArUco, can be quickly analyzed using computer vision (Bacik et al., 2017; Sani & Karimian, 2017). ArUco is a minimal library for augmented reality applications based exclusively on OpenCV. A virtual environment or scene is developed using black and white markers with codes that rely on calling a single function. Figure 2.2 show some marker examples with ArUco.

(a) ArUco + Ogre virtual scene



(b) Single marker detection

(c) Markerboard detection

Figure 2.2: The ArUco marker examples

Brzozowski et al. (2016) propose measuring the magnetic field as the primary source of information for indoor positioning and navigation. A magnetic field map of the specified environment should be previously determined with permanent magnets used to modify the local magnetic field that could work as a marker. The idea arises from nature, where birds can travel thousands of kilometers and find a way to their destination point using the earth's magnetic field. The advantage of this navigation is that each location has a unique signature of its magnetic flux density. The different values could be caused by natural or humanmade activity.

Nevertheless, the most important thing is that this anomaly can be detected and measured by a magnetometer. A wooden frame, as shown in Figure 2.3, is used for the experiment to mimic a drone with three magnetometers. The wooden frame dimension is established by considering a drone's span and the corridor's width where the testing is performed.

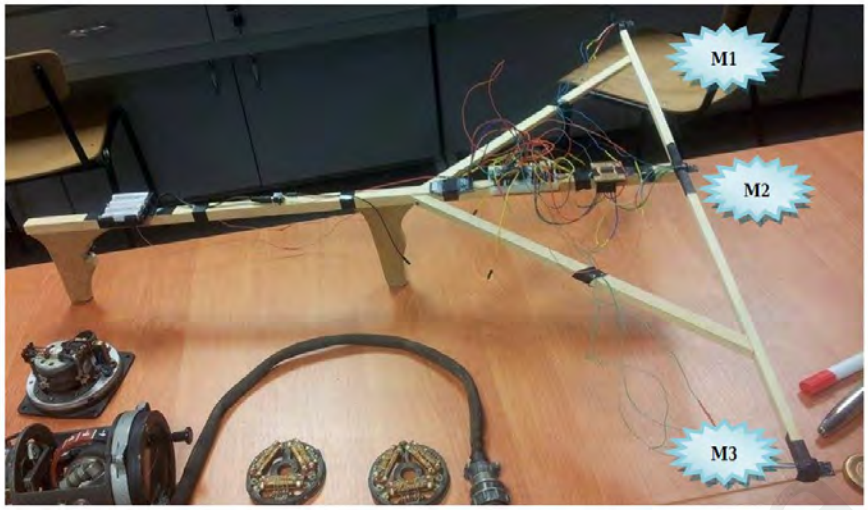


Figure 2.3: Wooden frame to mimic a drone with M1, M2, and M3 as the location of the magnetometers.

Figure 2.4: Magnetic field measuring device mounted on a rack

Figure 2.4 depicts a platform that functions to measure a magnetic field at different heights as it is possible with drone usage. Measurements were made at three different heights over the entire length of the corridor to prepare the magnetic map of the corridor. After that, permanent neodymium magnets that are made out of N42 material is placed on the floor along the corridor. The presence of the magnets is observed to correspond with large anomalies in the previously registered magnetic field.

In summary, prior preparation of the magnetic maps is required if one plans to use it for actual drone navigation. The preparation includes having two or more drones flying inside a building in a leader-follower formation. While the leader carries a range of sensors, such as the laser, sonar, and magnetometers, to build the magnetic maps, the follower can be furnished just with a magnetic field sensor. The main disadvantage of the magnetic field is that it is affected by the presence of electronic equipment, such as computers, printers, mobile phones, as well as a drone's motor. Table 2.5 summarizes Kouris & Bouganis (2018) contribution to drone navigation planning using markers.

Table 2.5: Drone navigation planning using markers

Reference	Aim of Work	Result	Gap
Kouris & Bouganis, 2018	CNN trained on image collected through a drone camera and annotated with distance measured with an ultrasonic sensor mounted on the drone	Using two-stream regression architecture generally manages to make more insightful decisions in cases of high ambiguity utilizing the learned, richer in information, Spatio-temporal representation of visual input	1. Not able to detect objects that are not used in training

2.3.5 Remarks

The object-based approach, as per name implied, detects points of interest (features) in the image and group these points together to deduce an object, consequently predicting distance from the robot. Features can be detected by using classical methods such as SIFT, SURF, edge, and 3D points cloud (Agrawal et al., 2015; Lee et al., 2011; Aguilar et al., 2017; Kaff et al., 2017; Zheng et al., 2017; Park & Kim, 2013), as well as the latest approach using a deep convolution neural network (Kouris & Bouganis, 2018). Objects can also be detected using active sensors, such as ultrasound, infrared, LIDAR, or marker (Ramasamy et al., 2016; Nemati et al., 2015; Tarazona et al., 2014; Rambabu et al., 2015). Other researcher uses a unique marker to learn features and elicit object information (Brzozowski, 2016).

2.4 Space-based approach for robot navigation

The space of navigation helps to guide us as we walk, drive, fly about in the world. Elements of the space of navigation include places, which may be rooms, buildings, parks, rivers, or oceans, and countries or planets or stars, on yet larger scales. For a room, the space of navigation is defined by the walls of the room. One can only move about within the boundedness of the room. For an outdoor environment, the space of navigation is usually not immediately comparable due to a lack of boundedness. However, outdoor spaces are interrelated with other definitions, such as paths or routes and other spatial reference frames.

For many years, psychologists and behavioral scientists put much effort into studying the behavior of humans and animals in their environment. Their findings show that insects, amphibians, reptiles, birds, and mammals, use the space of navigation differently from one another to imagine their environment. Building mental imagery of

one's environment is called the cognitive mapping process. The cognitive map in an animal's mind helps one to navigate from one place to another, find one's way home, discover shortcuts, and avoid getting lost. This difference in the cognitive mapping pattern is influenced by the fact that each species has a different type of sensors, needs for survival, and techniques to move in their environment.

2.4.1 The space of navigation

Golledge (1999, p. 6) defines wayfinding as "the process of determining and following a path or route between an origin and destination." Briefly, human wayfinding strategies can be broken down into four component problems:

a) Landmark recognition

b) Path/route decision

c) Direction decision

d) Creating an abstract layout of the environmental

Landmark recognition, path/route decision, and direction decision are activities that complete human and animal wayfinding daily. Without making a constant evaluation of the external environment, humans and animals will not navigate efficiently. When a human or animal evaluate their external environment, they will acquire much information. For example, the shape of the environment they are in, so they may consider a strategy to traverse it. A cluttered environment will require them to avoid obstacles nearby. A vast space will most likely be traversed with speed and without much maneuvering trouble.

Another example is the acquisition of information on landmarks. Landmarks are like environmental indexes that mark spaces so a person or animal using it may know their current orientation and location in the environment. For this reason, landmark identification is considered the most fundamental component in wayfinding. However, identifying or extracting distinct objects or features from the environment as landmarks is not an easy task and remains the most significant challenge. Low-level landmarks are usually well-defined features or patterns abundant and repeating in a space described by edges, segments, regions, and intersection points extracted from images (Fuentes-Pacheco et al., 2015). It is often rigid in size and shape, and well distributed in terms of position.

Low-level landmark is also the reason which contributes to the rise of an object-based approach in navigation. Figure 2.5 shows the use of junctions as low-level landmarks in Tolman's maze. At a higher level, landmarks can also be defined as a point of reference from a structural aspect. They differ from the low-level ones by their cognitive salient and prominent features (Duckham et al., 2010). These landmarks are also regularly observed and re-observed in external environments, such as a monument or a building (Basiri et al., 2014). Figure 2.6 showcases famous landmarks in several countries.

Path planning includes choosing a route to the goal. A path is not a direction but can be considered as the step in reaching the target location from a start point. A series of places, or waypoints, that will lead to the target location can be referred to as path selection. There are many models in cognitive mapping that describe paths as sequences of visual landmarks (Gupta et al., 2017; Nazareth et al., 2018). The need for a series of reference points in developing path planning puts this activity after landmark extraction in human and animal wayfinding.

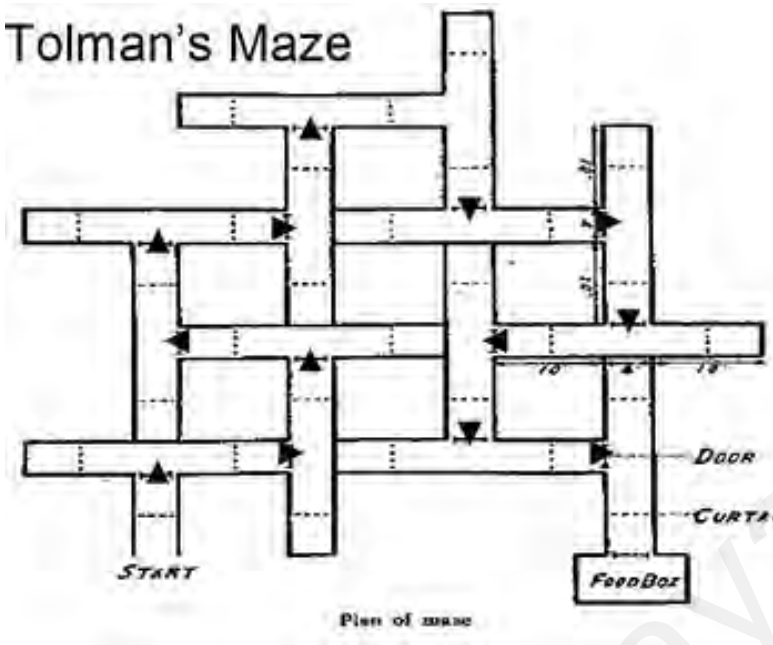


Figure 2.5: Junctions as low-level landmarks in Tolman's maze (Ciancia, 1991)

Figure 2.6: High-level landmarks examples. (a) The Kuala Lumpur Twin Tower, (b) the Statue of Liberty, and (c) the Taj Mahal

Figure 2.7 shows the comparison between landmark knowledge and path/route knowledge. Landmark knowledge depicts only the scattered location of landmarks available for space, but the route knowledge describes how one can achieve the target location from the starting point using the landmark knowledge.

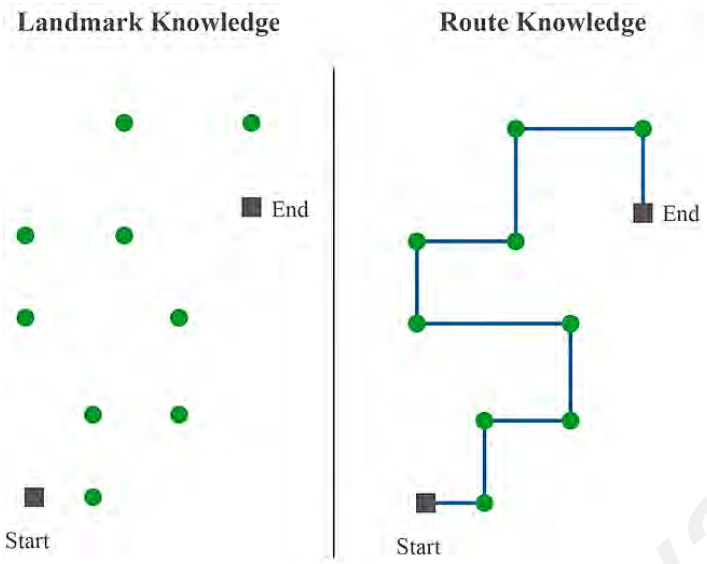


Figure 2.7: Landmark knowledge vs. route knowledge (Quesnot & Roche, 2015)

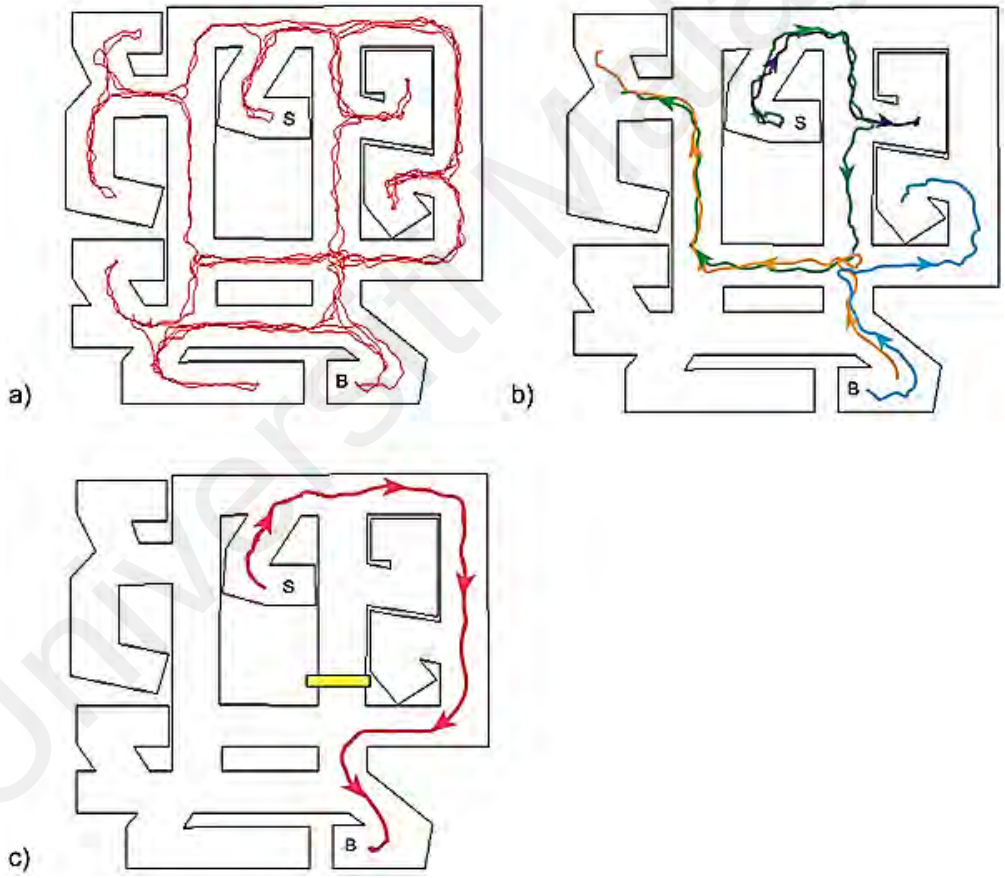


Figure 2.8: Path planning with (a) all possible paths from S to B, (b) possible shorter paths from S to B, and (c) possible short path from S to B when an obstacle is introduced (Chrastil & Warren, 2014)

The third strategy in human and animal wayfinding is deciding the direction, intending to find the shortest path or best possible path according to the navigation problem. When the target location is within a human or animal's line of sight, the reasonable direction to pick would be towards the target's orientation. For cases where the target location is not within the line of sight, the direction selection problem becomes complex to handle. A lack of information regarding space causes complexity, as one would have to guess the sequent of turns to get to the target location. This uncertainty makes direction selection at the starting point of the journey rarely sufficient in guiding the entire path planning (Kaplan & Friston, 2018). Figure 2.9 shows how foraging ants find their way home. The ants would have an unstructured outbound trajectory until the food is found, then return with a straight orientation to home.

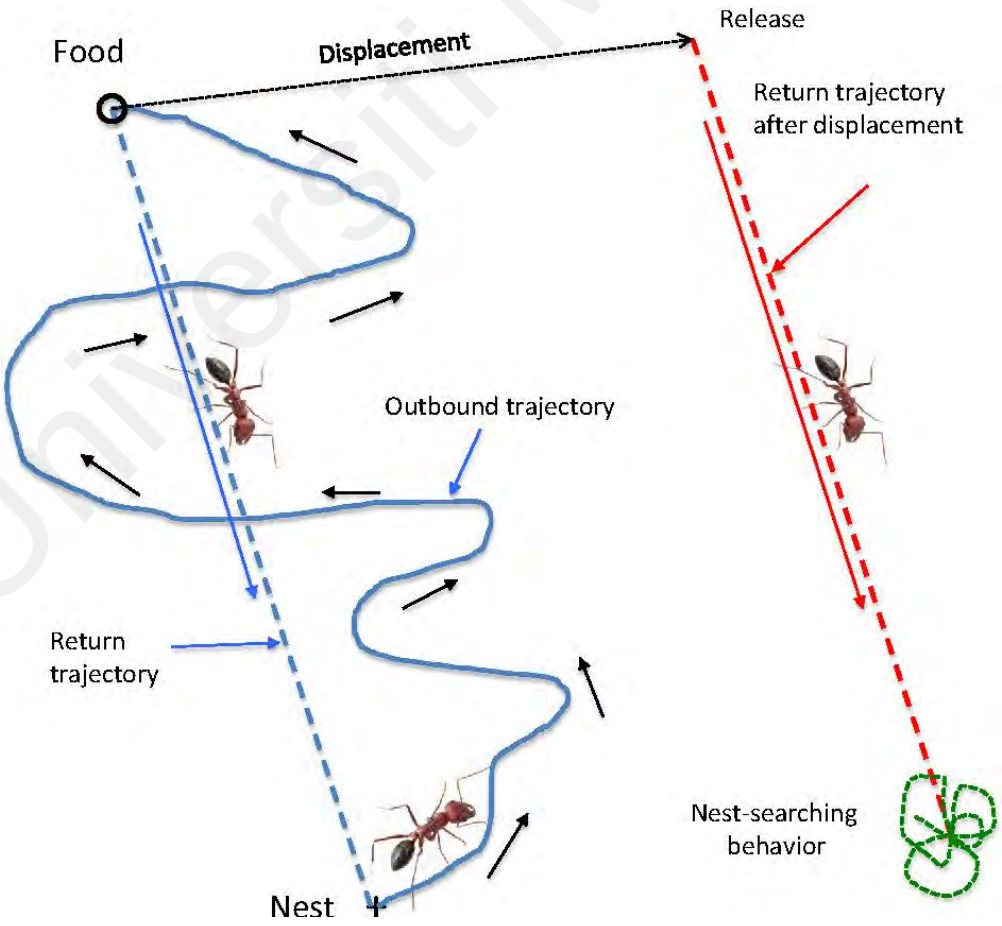


Figure 2.9: Foraging ants selecting direct orientation to home (Srinivasan, 2015)

The acquisition of this external information, landmark knowledge, route knowledge, and direction selection is crucial in updating one's mental imagery. The mental imagery, in a cognitive process, is, at most, an abstract overview representing one's external environment. The image is abstract because human and animal cognition does not store everything they perceive from the external world (Newcombe, 2018; Lohr, 2019; Irwin & Irwin, 2020). A cognitive map is then a representation that can be erroneous to describe the external surrounding. Therefore, the cognitive map requires constant updating, so it becomes logical to the user for effective wayfinding. Apart from behavioral, psychological, and cognitive scientists, robotics researchers have also begun to investigate the processes behind cognitive mapping in humans and animals. Their idea behind the work is to aim at programming robots with a cognitive map.

2.4.2 The absolute space representation

A robot essentially cannot have a cognitive map because a robot is not a cognitive agent. Nevertheless, robots do have the capacity to execute stochastics and deterministic rules, which make them a great platform to test behavioral and cognitive theories about human and animal mental maps. For example, Schmidt et al. (2006) and Wong et al. (2007) explored how mobile robots equipped with sonar sensors can have their cognitive maps. They looked at fuzzy approaches to manage noise that is generated by the sonar sensors. The sonar sensors are limited to three meters of depth perception and allowed the robot to only move in a zig-zag way in a corridor, due to the sonar's obstacle avoidance algorithm. They show how a primitive behaving mobile robot used a fuzzy cognitive map to learn about distances and directions in performing navigation.

Central to Schmidt et al. (2006) and Wong et al. (2007) works is the spatial conception called an absolute space representation (ASR). The ASR is simply a representation of each local space one visited; local space being a loose term used to describe the space one is currently in. However, in defining an ASR, Yeap & Jefferies (1999) discussed the local space as an enclosed boundary surrounding the human or animal. They also discussed the boundary having openings called *exits* so one can enter or leave a local space. To make explicit an ASR, one can derive from following the shape of the environment for the ASR boundary, often walls for an indoor environment, and use doors to note where the exits are.

For Schmidt et al. (2006) and Wong et al. (2007), their fuzzy cognitive maps are a result of their sonar robot traversing from one fuzzy ASR to another. Fuzzy because the three meters depth-sensing sonar is noisy in determining the ASR enclosed boundary. Regardless, these works mark the first computation of the ASR spatial concept on real-time robots traversing segmented circular office corridor. An exit is defined at every segment of the corridor. They show how rough distance estimation and direction selection plays a vital role in determining a robot's wayfinding. Figure 2.10 and Figure 2.11 show their sonar robot results.

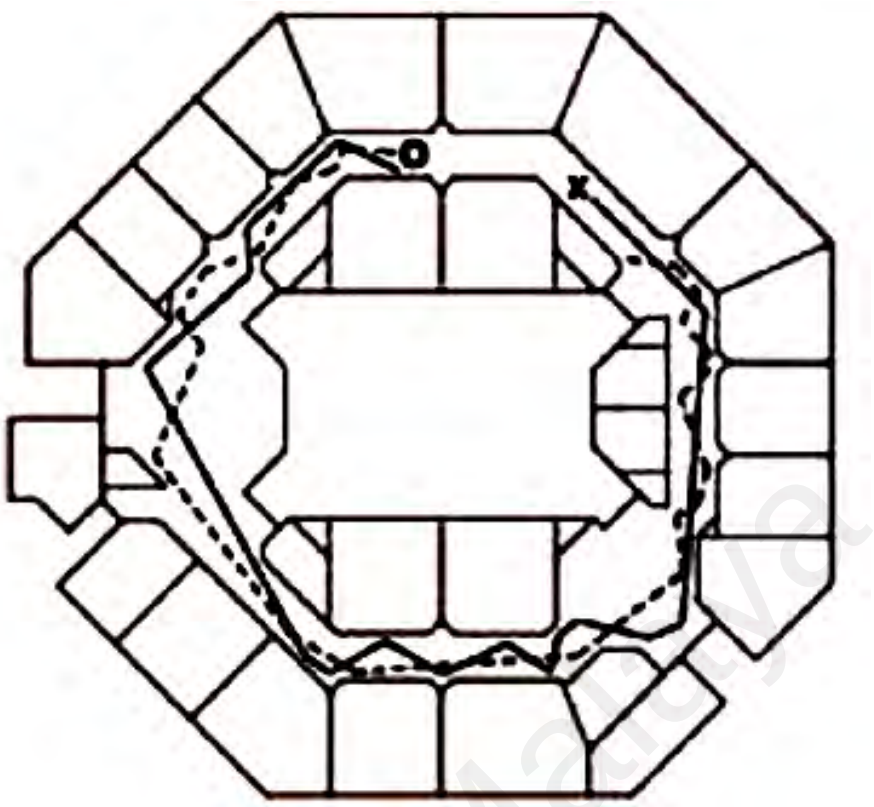


Figure 2.10: Starting location is marked X, and the target location is O. Solid lines depict traveling from X to O and dotted lines from O to X (Wong et al., 2007)

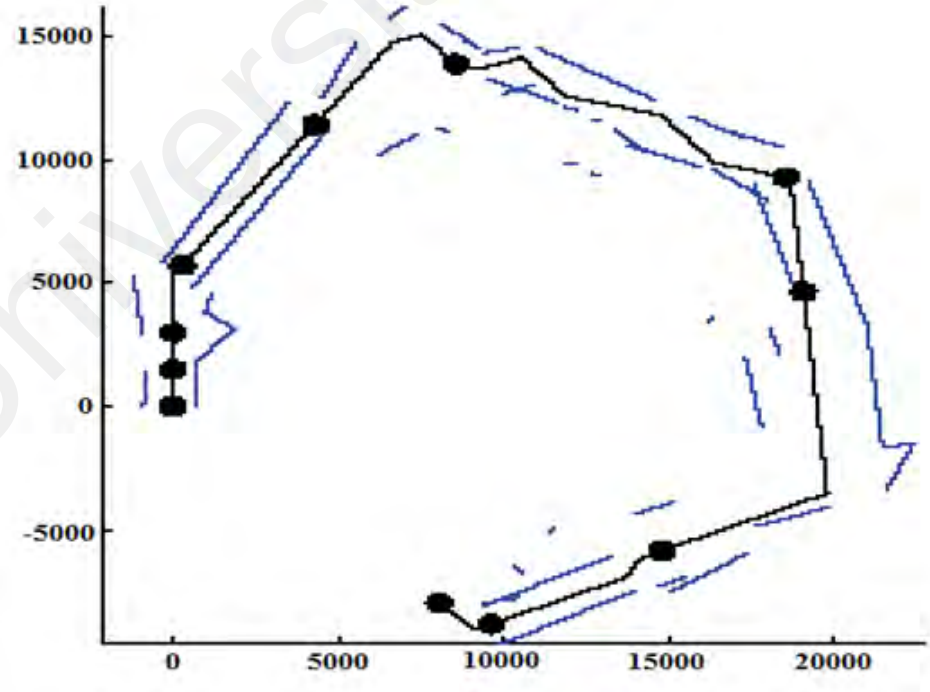


Figure 2.11: Each dots represent an ASR in the fuzzy cognitive map (Wong et al., 2007)

The success of implementing a network of ASRs is significant for a real-time mobile robot. For the first time, a robot is used to prove how humans and animals use distance and orientation information in their wayfinding. The robot calculated the distance between exits of each ASRs in the homeward journey and projected it onto the path planning created for the outward journey. The projected distance is then recalled when returning home. Merely recalling the distances between one ASR to another while backtracking home is not enough. The home finding is successful only when the robot maintains orientation to home throughout the attempt. Another significant achievement is showing how a machine uses a non-metrically precise map as a navigation reference.

Despite the success of the sonar robot's fuzzy cognitive map computation, several questions about the humans and animals' internal mapping process are unanswered. Mainly, the world is relatively stable. Indeed, with a better sensor, a robot can develop a map that sustains a rough shape of the environment. The fuzzy sonar map did not do great in this perspective, missing the overall shape of the environment. The overall fuzzy cognitive map is too vague to offer a meaningful description of the environment. Nevertheless, for humans, we can describe a place to a point where it is reflectable by another person.

It is also noteworthy that the sonar robot is limited in its movement, only moving in a zig-zag formation. Zig-zag movement is unlike humans and animals, which move through space with purpose, not leaving for luck and chance to achieve a target location. What is the algorithm for the humans and animals traversing through segments of space? How can a robot imitate it? Also, when is an opening a gap, and when is it an exit? The fuzzy cognitive maps do not offer clarity to this question

In 2015, Azizul & Yeap explored computing a network of ASRs with a more accurate depth-sensing robot, a laser type sensor. The laser can emit up to 30 meters beam at half degrees each for a 180 degrees field of view. The specification is ideal for scanning in an indoor environment. The limitation of a laser type sensor is that it is susceptible to transparent walls. Without a sonar type sensor on board to detect glasses, a slight modification to the environment is required, i.e., covering glass walls with the opaque lining. The laser type robot is fitting to compute ASRs in an indoor environment following the definition by Yeap & Jefferies (1999).

First, the depth of the scanning laser is ideal for detecting and following the boundaries of the room. With 30 meters depth range, a scan can acquire the boundary information by converting laser points into 2D lines. These lines or surfaces make up the enclosure required by an ASR. Second, the half-degree resolution of the laser type sensor describes spaces of doorways and office furniture well. Some of the spaces are large and allows the robot to pass through easily. Some of the spaces are narrow and restrictive. These openings can be evaluated and classified into gaps and exits.

Figure 2.12 shows the laser robot set up in extracting input from the environment. Given that laser beams are in the form of light and light penetrates through transparent materials, Azizul & Yeap (2015) reported that all-glass walls in the office are covered using cardboards throughout the implementation. Covering the glasses helps avoid mishap as the robot can mistaken glass wall areas as passable. They propose this is the only physical modification done to the environment. Everything else; the position of things, opening or closing of doors to the rooms, is left as it is.

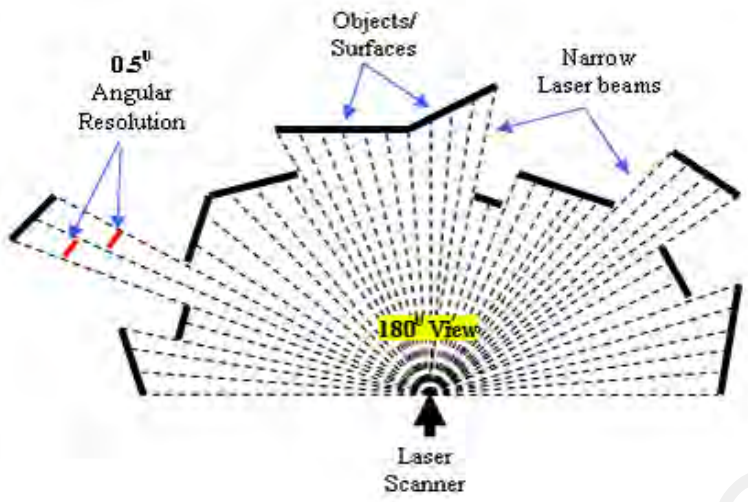


Figure 2.12: Raw laser scanning at 180^0 field of view and 0.5^0 resolution (Azizul, 2013)

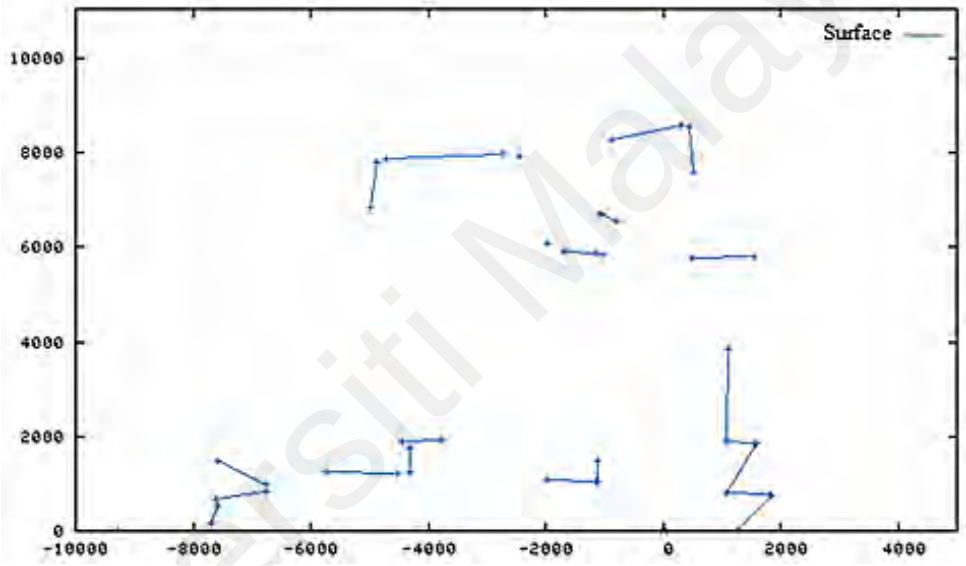


Figure 2.13: Robot at (0,0) looking out to boundaries computed of a room (Azizul, 2013)

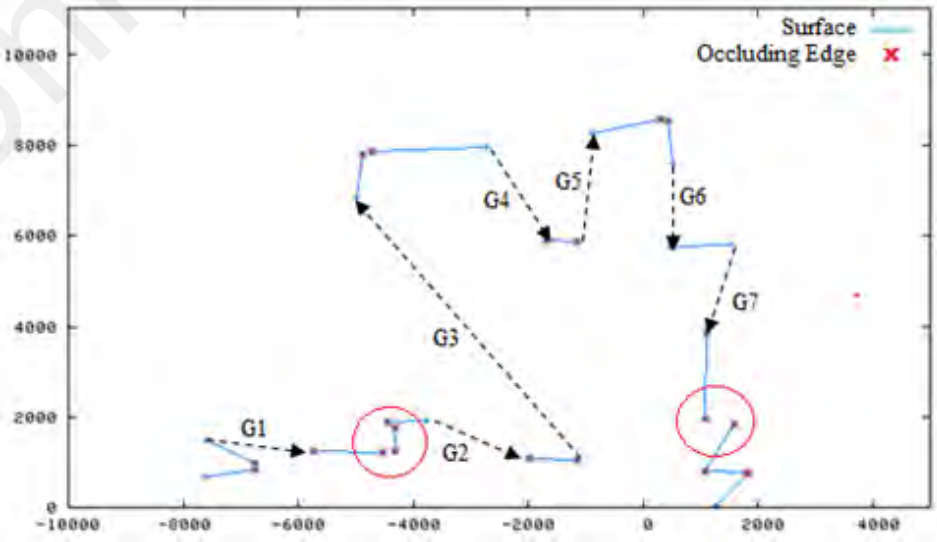


Figure 2.14: G1 to G7 indicating gaps found in between the boundaries (Azizul, 2013)

Figure 2.13 is derived after raw laser points are processed to extract line. Algorithms like the split-and-merge algorithm, the line regression algorithm, the incremental algorithm, and the Hough transform can convert points to 2D lines, providing an accurate polygonal description of the environment. Figure 2.14 denotes gaps and the possible emergence of exits. Tiny openings such as those in red circles are ignored because they are a good indicator of areas cluttered with objects. Large openings, the largest in the example is G3, indicate potential wide spaces. When the boundaries and gaps are joined together, they make up a potential ASR for the robot.

The work by Azizul & Yeap (2015) indicates that a well-defined local space can be computed using a more accurate depth sensor such as the laser. A local space from a laser robot perspective shows boundaries and gaps from the environment. However, local space is not an ASR until an exit is found and crossed by the robot. The example from Figure 2.14 does not depict the emergence of exit, which means the robot must search for it by traversing through the local space. How can the laser robot traverse a local space? How do humans and animals perform path selection? The limitation of Yeap & Jefferies (1999)'s ASR is that it does not include computing a robot's waypoints to pass-through space.

2.4.3 The minimal bounded space (MBS) method

From the waypoint strategy described by humans and animals, path planning is a crucial step when exploring the environment. However, if one has not developed a bird's eye view of space and *remembers* it, how can one perform efficient path planning? What happens during updating? What information gets an update, and what does not? Cognitive and behavioral researchers have broken down wayfinding strategies and propose numerous spatial conception and organization processes in the human mind. One that is strongly represented in revealing the processes of cognitive mapping in humans and animals is the communication between the egocentric and the allocentric representations (Ruggiero et al., 2016; Starrett & Ekstrom, 2018; Colombo et al., 2017; Wang et al., 2020).

The egocentric representation concerns the immediate space surrounding one's body, where objects from the external world are associated with the ego or the body. Figure 2.14 represents the egocentric representation of the laser robot. Features from the local space, the boundaries, and gaps, are organized as captured by the laser sensor from the robot's position. When one traverses through space, one collects a series of egocentric views from the environment. Information such as landmarks, paths, and direction, are accumulated as one move. At some point, somehow, this spatial information is chunked, linked, disjointed, neglected, and reorganized into a single, global, allocentric representation.

Cognitive and behavioral researchers believed the transition of spatial information between the egocentric to the allocentric representation is the basis in cognitive mapping updating (Meilinger & Vosgerau, 2010; McCunn & Gifford, 2018; Zhang & Mou, 2017). Collectively they further highlighted features of the cognitive map:

a) The cognitive map contains; one, a transitory egocentric system which can be disrupted with disorientation, and two, an enduring allocentric system which is less vulnerable to disorientation
b) The interchange between the egocentric and the allocentric systems mainly functions to re-establish the remembered direction of an object; either to self or to other familiar objects in the environment
c) Human is said to use the stable allocentric system to reorient for the environment's shape
d) Getting lost does not destroy the allocentric spatial representation
e) The moment a human recovers his location and orientation to the allocentric spatial representation, he regains his spatial relations with familiar objects in the surroundings, and this information is sufficient to determine where he is at in that environment

It is from these observations that Azizul (2013) proposed a minimal bounded space (MBS) method to extend Yeap & Jefferies (1999)'s ASR. ASRs on their own are explicit for robot navigation, as seen in Wong et al. (2007). A robot can explore the environment by traversing from one ASR to another through their shared exits, defining high-level path planning (see Figure 2.11). However, the algorithm for Wong's sonar robot is robust only for a corridor-like environment, where there is no requirement to search for an exit. An exit is available by default at the end of each corridor segment.

For a complex environment such as the one in Figure 2.14, the laser robot requires a conscious selection of paths in traversing local spaces. Otherwise, the laser robot may end up like the ones in Figure 2.8(a). An explicit path and direction selection strategy is required, so the laser robot always searches for exit to complete an ASR.

The goal at local spaces for an ASR is to find an exit and cross it, and the MBS aims to allow robots to perform precisely that. Central to the MBS method is evaluating occlusion information in the immediate surrounding. Each line computed by the laser robot has edges that are either an occluding point or an occluded point. An occluding point means the edge is absolute and within the robot's line of sight. An occluded point means the edge could be partially hidden. The occlusion plays a vital role in evaluating gaps in the boundary. In Figure 2.14, all the gaps G1 to G7 are drawn from one occluding edge and one occluded edge. As a result, the gaps are not perpendicular to the robot at (0,0). So even if the robot turns towards the middle point of the gaps and drives to it, the occluding edge may block the robot.

Drift is unavoidable in robot navigation. Drift causes wheels to steer out of the course and usually get worst over longer drives. Cutting robot drive shorter can reduce drift, but with gaps at the boundary, there is always distance to consider when driving to the gaps. The MBS solves this by reorganizing gap information in local spaces. The MBS redefines gaps for an ASR by drawing gap lines between two occluding edges. The algorithm to generate MBS gaps is described next (Azizul, 2013). Figure 2.15 shows the effect of the algorithm on an ASR local space.

#1 MBS gaps algorithm

Count_newGap is 0

DO

Count_m is 0

(1) FOR all G^n and G^{n+1}, go to (2)

(2) IF start_point of G^n is an occluding point, AND
end_point of G^{n+1} is an occluding point,
Increment m by 1, go to (3)
ELSE go to (11)

(3) Create new_gap_line from start_point of G^n to end_point of G^{n+1}, AND
Mark the mid_point, go to (4)

(4) Draw an imaginary_long_line (30m) from robot crossing the mid_point, go to (5)

(5) IF imaginary_long_line intersects with the virtual boundary after crossing the mid_point, go to (6)
ELSE go to (11)

(6) IF new_gap_line is between 0.6 to 1.2 meters, go to (7)
ELSE go to (8)

(7) Save new_gap_line as an exit, go to (9)

(8) Save new_gap_line as a gap, increment count_newGap by 1, go to (9)

(9) Remove G^n and G^{n+1}, go to (10)

(10) Increment n by 2, go to (12)

(11) Increment n by 1, go to (12)

(12) Repeat step, go to (1)

Rename gap from left to right order

Rename exit from left to right order

WHILE Count_m is not 0

IF count_newGap is larger than 0, remove remaining raw gaps

Rename gap from left to right order

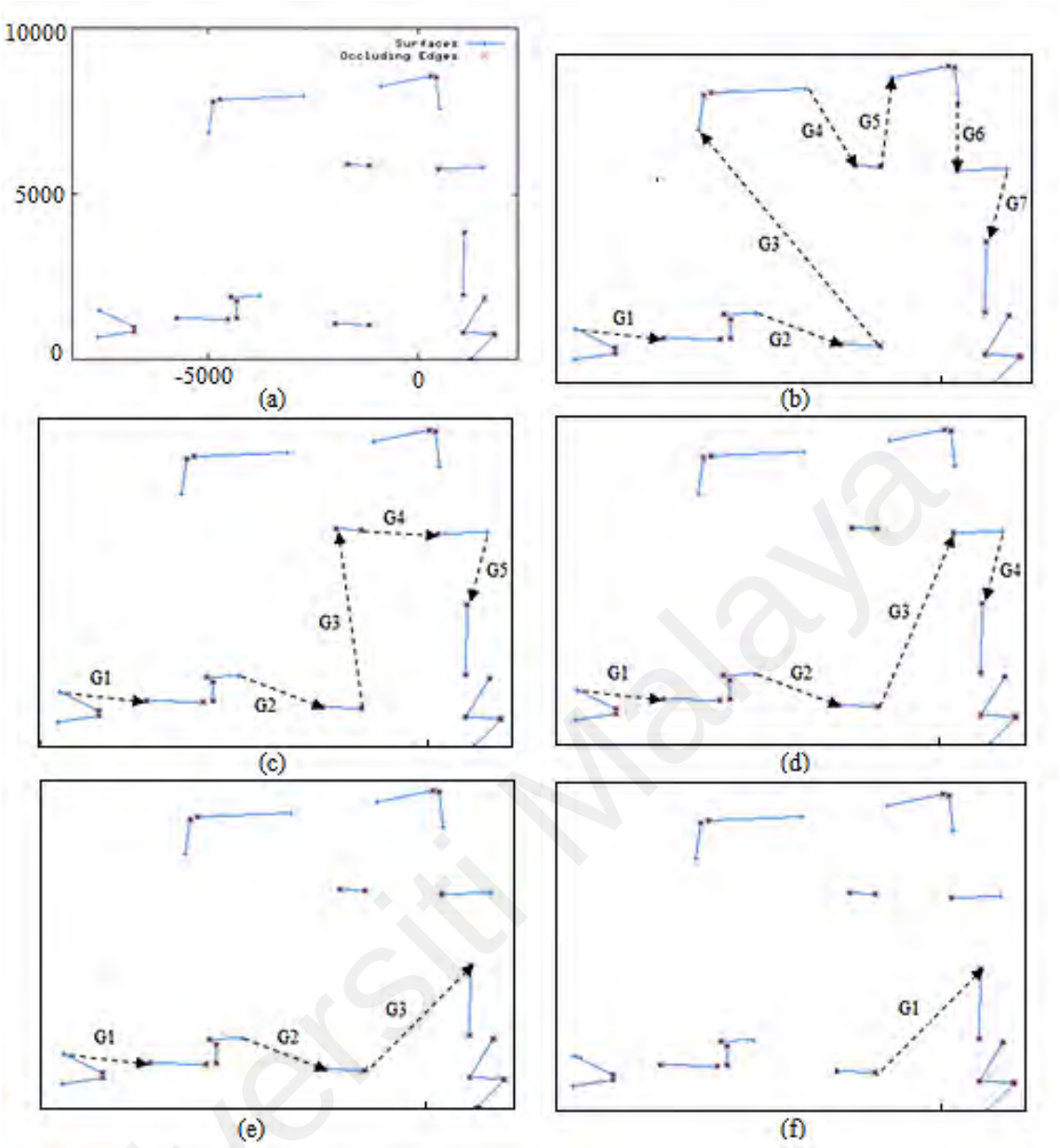


Figure 2.15: Deriving best MBS gaps for the robot to cross (Azizul, 2013)

The algorithm first checks G1 and G2 in Figure 2.15(b) but rejects a new gap line because the start point of G1 is not an occluding point. G2 and G3 fail similarly. When G3 and G4 are analyzed, they pass the requirement, and a new gap line is drawn between their occluding edges. The same goes for G5 and G6. The first iteration is exited with G7 left as it is. The second and third iterations pass only G3 and G4, respectively (see Figure 2.15(c) and Figure 2.15(d)). G1 and G2 in Figure 2.15(e) are removed, leaving one in Figure 2.15(f). Figure 2.16(a) is taken once the robot crosses G1 of Figure 2.15(f).

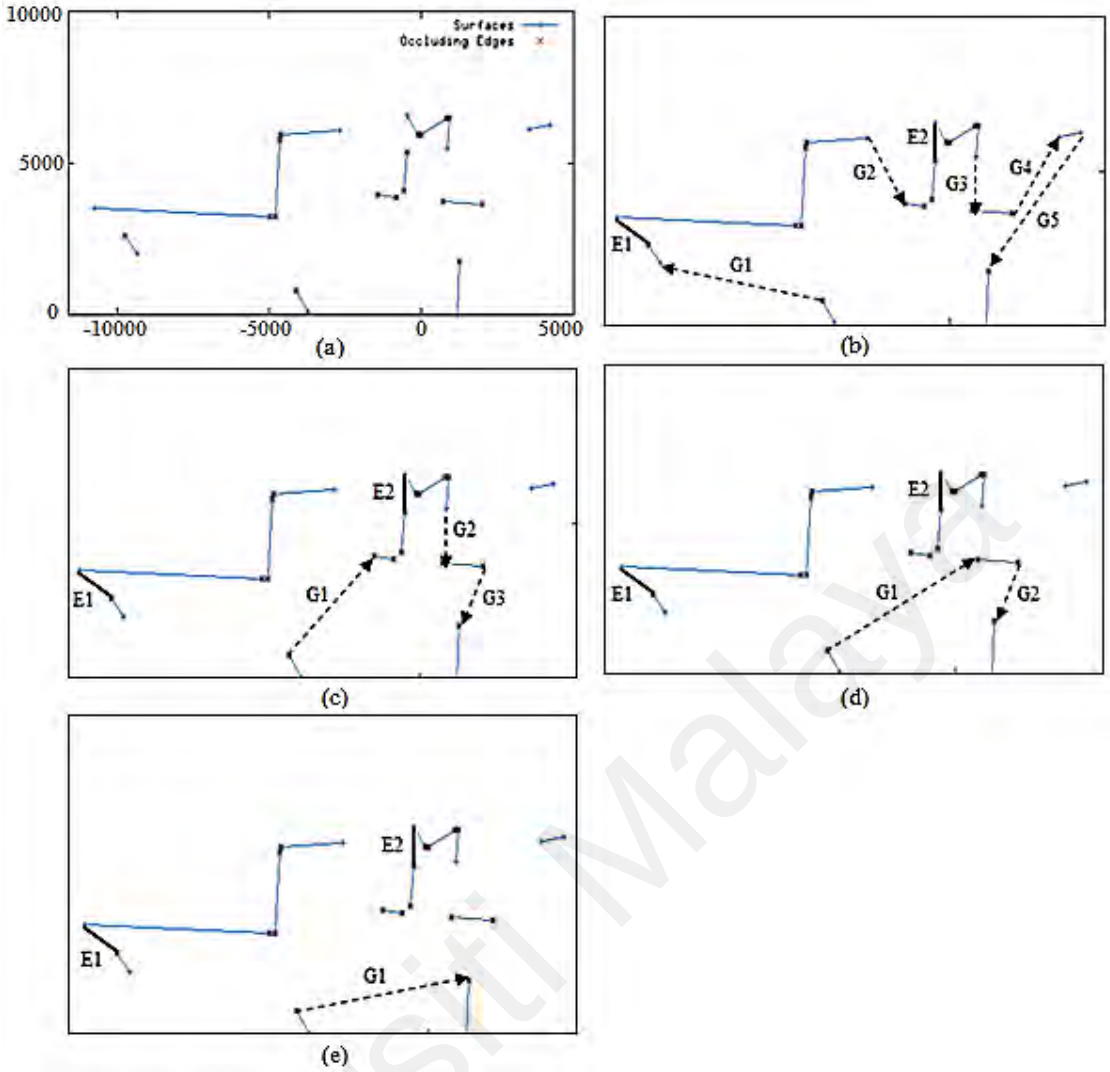


Figure 2.16: Another example of the MBS gaps (Azizul, 2013)

The next scan begins with two exits (E1 and E2) in Figure 2.16(b). The MBS gaps algorithm only processes gaps; thus, the exits are left as it is. In the first iteration, the algorithm passes G1 and G2, then G4 and G5. In the second iteration, the algorithm passes G1 and G2, and leave G3 as it is (see Figure 2.16(c)). The next iteration passes G1 and G2 (see Figure 2.16(d)). Finally, G1 and the two exits are left. The robot is instructed to turn facing G1 at the middle intersection and cross it. The robot eventually crosses E1 after several laser scans.

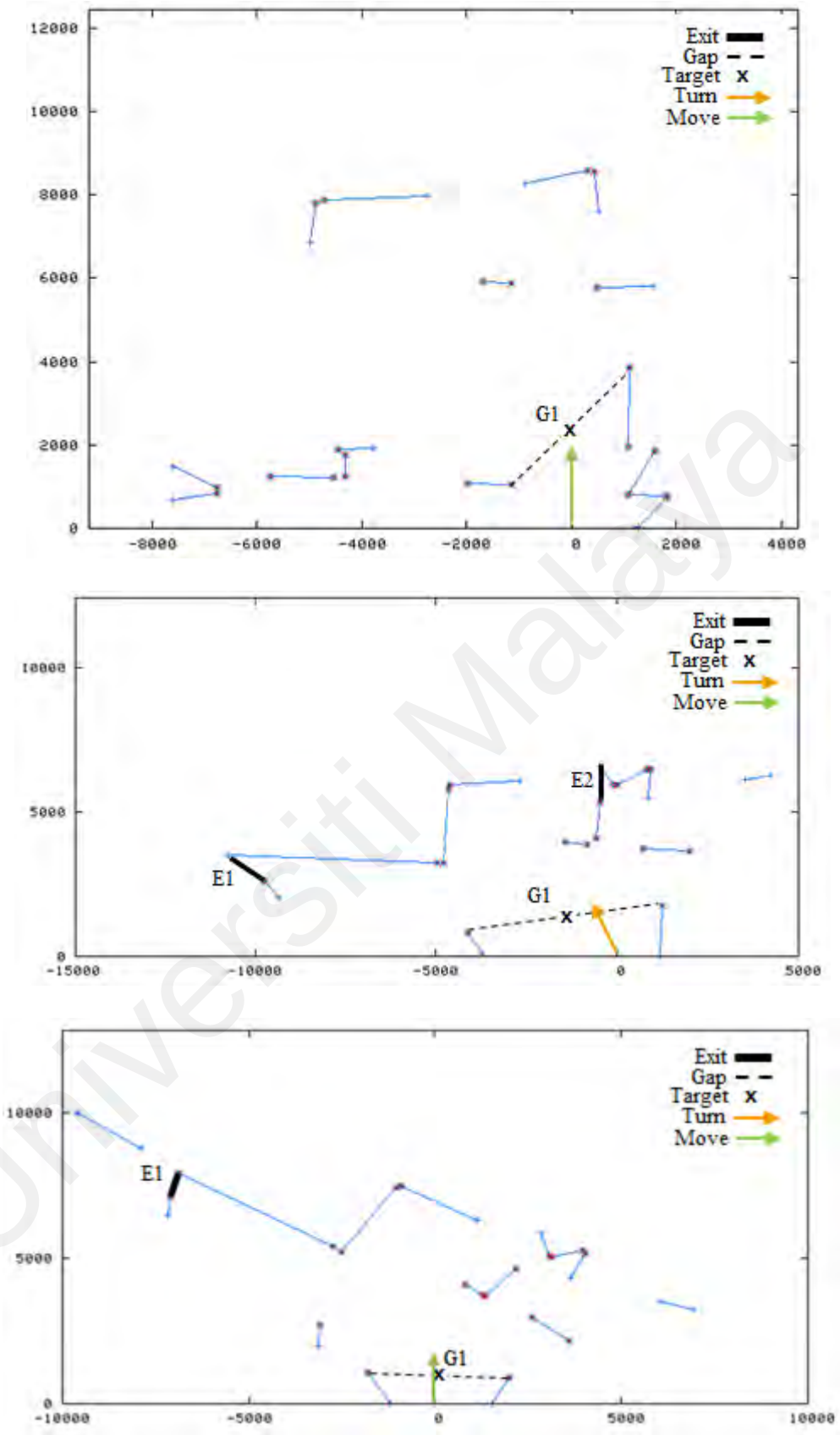


Figure 2.17: The first three MBS gaps crossed by the laser robot (Azizul, 2013)

Azizul (2013) shows how a laser robot uses the MBS to redefine gaps while traversing local spaces. The MBS strategy does not focus on obstacle avoidance; instead, it utilizes an understanding of spatial information like gaps and occlusions to plan its path. The path planning executed by the robot is also similar to humans and animals. For example, humans and animals choose to cross spaces where it is least blocked. The safest path is usually the middle of the way, with walls and furniture at the sides. Humans and animals do not wander about without purpose. They look for openings in the environment to extend their journey, and they are unlikely to follow the wall or go by random path selection. The MBS method facilitates robots to move similarly. More significantly, the MBS allows robots to be fully autonomous and support spatial frameworks like the ASR. A full report of the robot mapping process is available in Azizul & Yeap (2015).

2.4.4 Implementing the MBS on a visual robot

The practicality of the MBS method for real-time robot navigation prompted Azizul & Khanil (2017) to explore the approach on a visual robot. They found that the adoption of the MBS method from laser to vision is not straightforward. A new MBS algorithm is required, one that can redefine gaps in local spaces when depth information is not available. Without depth information, it is impossible to derive occlusion information, which has been fundamental in extracting the MBS gaps. Floor segmentation is proposed as the basis for the vision-based MBS. The robot used is called Rovio, a WIFI-enabled small robot equipped with a webcam, IR sensor, and three wheels moving forward, backward, sideways, and diagonal. The IR sensor is turned off for the MBS testing.

The Rovio can stream video, and the Python programming language and OpenCV library are useful for video processing. Generally, the MBS algorithm for the visual robot is designed to acquire a raw image from the environment, process the image, compute the MBS, and instruct the robot to cross the MBS gaps. Figure 2.18 shows the MBS method designed for Rovio navigation.

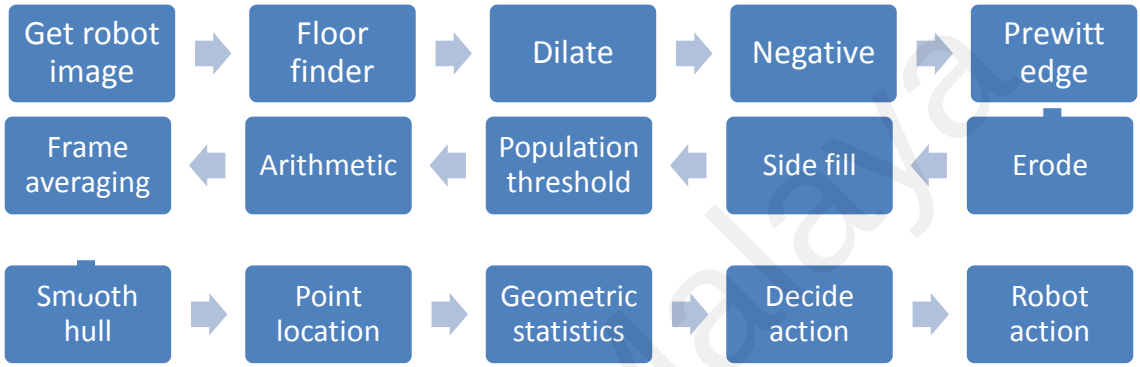


Figure 2.18: The MBS algorithm for visual robot navigation

Rovio's camera is forward-facing and positioned about 10-12cm from the ground. At that height, the viewing angle of the robot captures mainly the floor and furniture's bases and legs. The MBS bounds the immediate space in front of the robot by fixing a rectangle. When the pixel values filling up the rectangle matches the floor, the rectangle becomes the MBS for the robot. A point location on the MBS boundary may indicate the direction that is likely to take the robot to more openings. Figure 2.20 is the statistics results in Figure 2.19(13) when imposed on the original image.

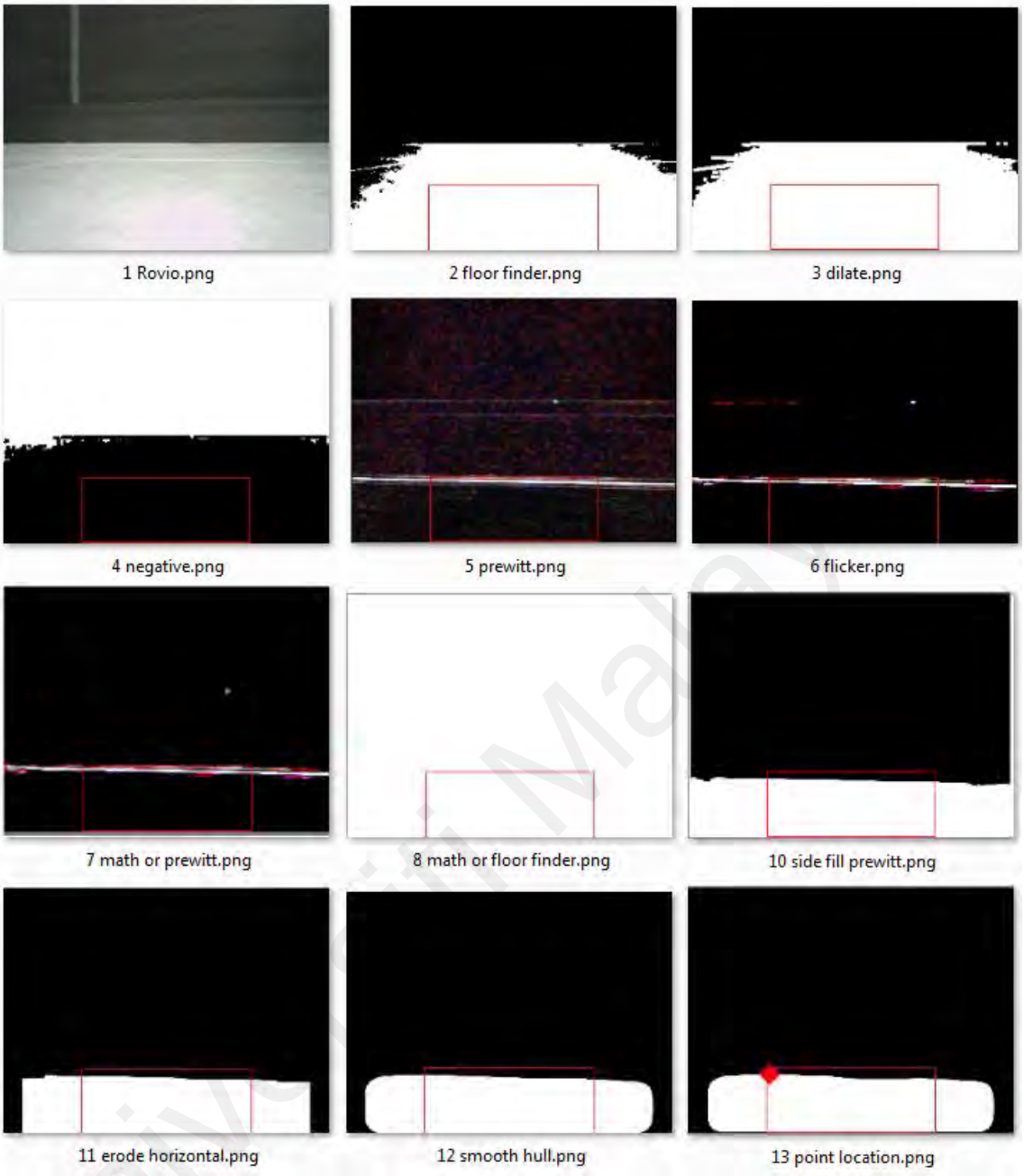


Figure 2.19: The MBS steps visualized (Khanil, 2016)

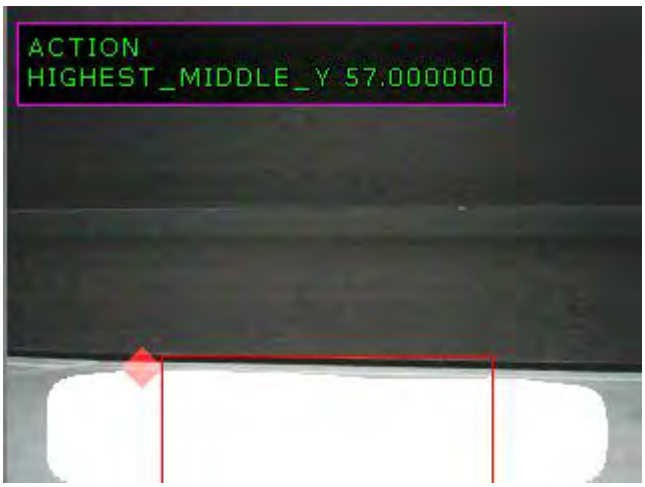


Figure 2.20: Diamond indicating the best direction selection for the MBS. Robot to turn facing diamond and cross MBS boundary (Khanil, 2016)

To test the MBS algorithm, Azizul & Khanil (2017) let their Rovio roaming in several environments; a room with flooring tiles of size 1x1 ft, a kitchen floor with flooring tiles of 2x2 ft size, living room with random tile textures, a 20x75 ft room with different colors flooring tiles, and a large hall with 1x1 ft tiles. These environments were left as it is, no changes done to the environment. The Rovio is programmed to stop if it hits an obstacle and getting stuck for more than five seconds. Figure 2.21 describes some of the robot actions.

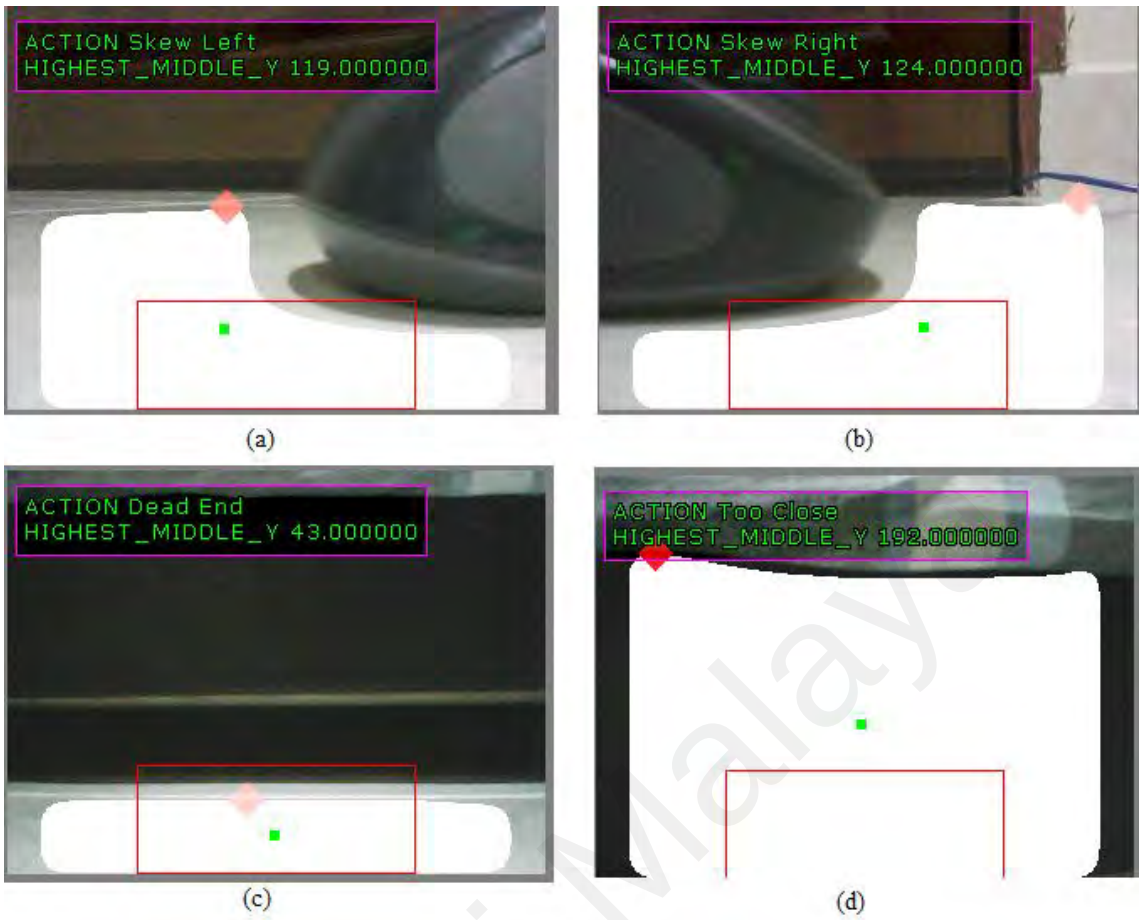


Figure 2.21: The MBS for visual robot navigation (Azizul & Khanil, 2017)

In Figure 2.21(a) and (b), the Rovio avoids obstacles by skewing left or right. The green dot projects the MBS gap, the point where the robot should stop depending on where the diamond is located. If the diamond is somewhere in the middle hemisphere of the image, then the green dot should be inside the MBS boundary. In cases where the diamond is located inside the MBS boundary, a dead-end could be found (see Figure 2.21(c)). Rovio should turn around and move away from the space. If the diamond is somewhere in the top hemisphere of the image, it means the pixel covering the entire image is similar. The pixelation is usually caused when the robot is immediately facing a wall. The action is included so Rovio can react to dynamic obstacles (see Figure 2.21 (d)).

Performance-wise, the MBS can fail if the robot encounters the following problems:

a) Viewing angle. Rovio could hit an obstacle lower than 8cm from the ground if the object missed the robot viewing angle. See Figure 2.22 and Figure 2.23.

b) Noise. Flooring texture influences the MBS algorithm. The MBS performs lesser when the flooring texture is too noisy, like the carpet in Figure

c) Lagging issue is encountered when the frame rate dropped from 30fps to 5fps

d) Communication. Poor WiFi connection can cause disruptions between the robot and the laptop performing the MBS algorithm.

The problem in (c) and (d) can be avoided if the codes are directly hard-coded onto Rovio's microcontroller. Furthermore, there are rooms for improving the vision-based MBS if the issues with image processing, such as the ones shown in Figure 2.25 and Figure 2.26 can be sorted.

Figure 2.22: Rovio viewing angle missing the chair base resulting in action go straight

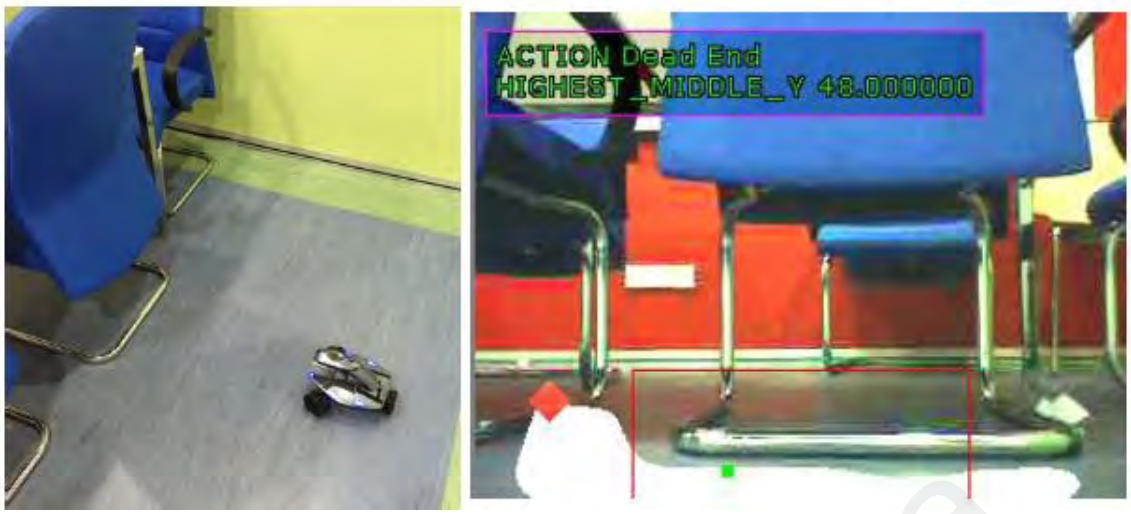


Figure 2.23: At a different scanning distance, the Rovio captures the chair base, and the MBS computes a dead-end

Figure 2.24: The MBS fails to compute when the image is too noisy

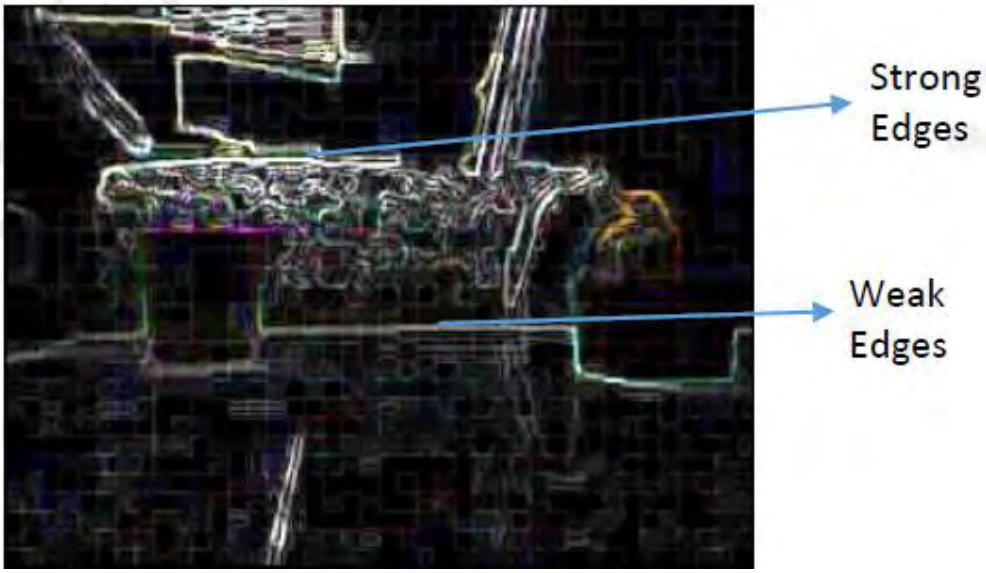


Figure 2.25: Prewitt detector missing weak edges causing the MBS to misinterpret obstacles

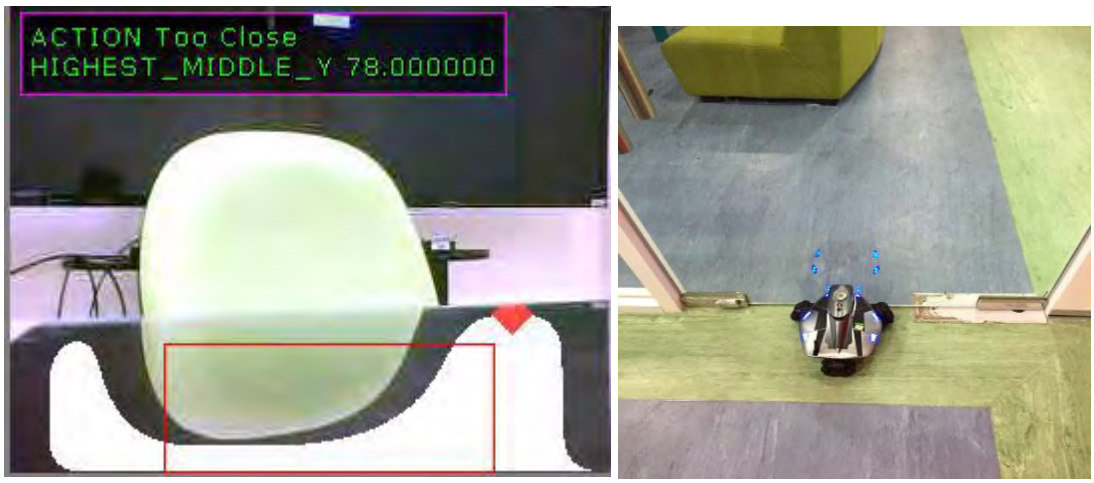


Figure 2.26: Reflection produces a false-positive image (left), and the robot sees through the glass wall (right)

2.4.5 Remarks

There are several critical problems for robot navigation; (1) finding where the robot currently is, (2) planning the path so a robot can move from one place to another, and (3) how to move the robot to follow the selected path. The probabilistic techniques such as the metrical, topological, and hybrid mapping can describe an image of the space for a robot. Path, trajectory, and motion planning can resolve path selection for a robot, while motion control is usually adopted to move a robot.

Despite the current robotics solutions, some of them are unnatural to humans' and animals' wayfinding. For example, in trajectory planning, often the precise metrical distance and direction are required for a robot to get from one point to another. Unlike humans and animals, rough distance and direction are sufficient. More importantly, humans and animals detect safe space in between obstacles and make a conscious decision to update their waypoints as they traverse. Rarely humans and animals follow walls or react to obstacles as a means to traverse the space.

Many spatial representation models describe the space of navigation. One that is implementable for robots is the ASR. The ASR describes local spaces as having a sense of boundedness, with openings to allow entries and exits. An ASR can help a robot to traverse the space by planning a path from one opening to another. However, an ASR does not define how a robot should move to fulfill such a path. To understand the space of navigation is to understand also space around the body. If an ASR is a representation of local space, then the MBS provides the method for a body to navigate through it. Two works showcase the utility of the MBS method for robot navigation. One is an implementation on a depth perceiving laser robot, and the other, a vision robot without depth information.

2.5 Sensing the world – an open challenge to robot navigation

Developing a successful robot navigation system, be it the object-based or the space-based approach, often depends on successful interpretation of the world surrounding the robot. Higher number of sensors mounted on a robot increases the chance for complete world model. Carrying an abundance of sensors is near impossible, when a robot is not able to carry load. When the load allowed is minimal, a robot designer has to choose a compact, lightweight and a non-resource hungry sensor. Often times, designers would choose a mono-camera over other sensors, for the richness of data captured.

When only mono-camera is mounted, the robot will lack attributes contributed by other types of sensors such as depth sensing and the time-of-flight wave bouncing from solid objects. As a consequent, the world model of the robot consists only data contributed by processing brightness level and cannot include range and transparent objects, without any modifications.

2.5.1 Glass detection

An active research in robot navigation includes glass detection. Glass detection is important for robot navigation because glass walls is a significant material of a building, indoor and outdoor. Several methods are proposed to overcome glass during navigation, and the repeating pattern is the presence of sensors such as the ultrasonic, depth camera, and lidar, and a technique such as the deep neural networks.

Forouher et al. (2016) combines the ultrasonic and depth camera for occupancy grid map creation. They show that the combination is able to improve the detection of transparent objects compared with using depth cameras alone. Wei et al. (2018) combines an ultrasonic and a laser scanner for glass detection. Their method utilized two maps for navigation; one laser, glass-less, SLAM map, and the other is what they called a *gmapping* map produced by the ultrasonic sensor. Wei and his fellow researchers overlay the *gmapping* map onto the laser map for a complete world model. They claim that this method increases robot navigation efficiency by 11% in glass environments compared with classical methods which use one map.

Wanga et al. (2015) perform laser beam intensity tests to detect glass in their robot's environment. They argued that the speed of light traveling through different mediums produce different intensity. They determine the intensity of light going through different glass surfaces and their results show good performance using a laser scanner, without further modification to the robot nor its algorithm.

A laser beam reflection intensity and threshold method is used in Kim et al. (2016) to determine where glass walls are by considering all candidate distances that can be measured in the direction of the glass wall. This is a novel method for improving the performance of laser range finder-based localization schemes in glass-wall environments. However, glass detection based on laser intensities measurement suffers a lot of drawbacks. For example, different types of glasses or thickness of glasses will affect the reflection intensities. When mapping is considered, the uncertain penetration lowers the probability of glass walls occupying a grid compared to opaque walls. Therefore, manual modification of the grid map is required.

Moving away from range finders, Mei et al. (2020) proposed to detect glass from a single RGB image using the neural-network methods. The proposed method successfully eliminated non-glass regions which share similar appearance with the glass regions and detected only the real glass regions. However, as with all the neural-network methods, this method fails in cases where the scene is very complex or has insufficient contexts which describe the inside and outside of the glass. Furthermore, new training data is required when moving into new environments.

2.6 Chapter Summary

Obstacle avoidance is essential for mobile robots to navigate autonomously and safely, particularly in dynamic environments. However, most robot solutions require depth information. Depth perceiving robots are either hardware expensive, the robot equipped with powerful rangefinders or stereo vision cameras, or algorithm expensive, the robot requiring complex probabilistic to elicit spatial features. When depth information becomes unavailable, a norm with low-cost and small robots, the robots are usually required to perform wall/line following or object tracking. These methods of navigation do not describe how humans and animals perform wayfinding.

The minimal bounded space (MBS) is a cognitive method inspired by how humans and animals move through spaces. The MBS marks safe spaces immediately to the robot, not by reacting to objects or obstacles, but by evaluating spatial openings in the surrounding. The MBS method has been tested on two different types of robots. While the principles are the same, the differing perceptual sensing made it difficult for a straightforward adoption between the two. Similar to cognitive species having distinct wayfinding mechanisms due to evolutions, robotics species, with their sensors and different abilities, can have different MBS algorithms. The robots tested with the MBS method performed autonomously, moving from space to space, in an indoor environment.

This chapter reviews the literature relevant to the work. The following chapter presents the methodology.

CHAPTER 3: METHODOLOGY

3.1 Overview

This chapter describes the MBS method for space computation in indoor and outdoor environments for a mono-visual drone. The chapter begins by introducing the drone and its sensor. The discussion continues with a description of a new MBS algorithm for a visual drone. Complementing the chapter is a description of a new computer vision framework to segment spaces for the MBS method.

3.2 The drone and its sensors

The drone used in this work is called the Parrot Bebop Drone 1.0 from Parrot SA. It is a light-weight drone with an average weight of 420g. The drone has been enabled to create its hotspot, which allows for a dedicated radio controller and a smart device's application to control it. The manufacturing company provides the API to send command and receive information from the drone used in this dissertation. In terms of power consumption, the drone battery lasts about 11 minutes of flights per taking off. Table 3.1 shows the specifications of the drone.

The drone is equipped with a front-facing, fish-eye lens 180^0 camera with 14 megapixels, digital video stabilization, 1920 x 1080 resolution, and 30Hz or 30 fps for onboard recording. Due to limitations in Parrot's ARDroneSDK3, the quality of the video stream is limited to 640 x 360 resolutions at 30Hz or 30 fps, with the field of view of the virtual camera is around 80 degrees on horizontal and 50 degrees on vertical. Figure 3.1 shows the general setup for the drone in this work.

Table 3.1: The Parrot Bebob 1.0 specifications

Dimensions:	330 mm x 380 mm x 36 mm
Weights:	420 g
Maximum speed:	13 m/s
Operating range:	250 m
Operating duration:	11 minutes
Processor:	P7 dual-core CPU Cortex 9 with quad-core GPU
Internal Storage:	8GB
Sensors:	Three-axis accelerometer
	Three-axis gyroscope
	Three-axis magnetometer
	Optical-flow sensor: vertical stabilization camera (data not available for use)
	Ultrasound sensor for ground altitude measurement
	Pressure sensor for ground altitude measurement
	Fish-eye lens 180° camera, 14 mega pixels, digital video stabilization, 1920 x 1080p, 30fps (not for transfer)

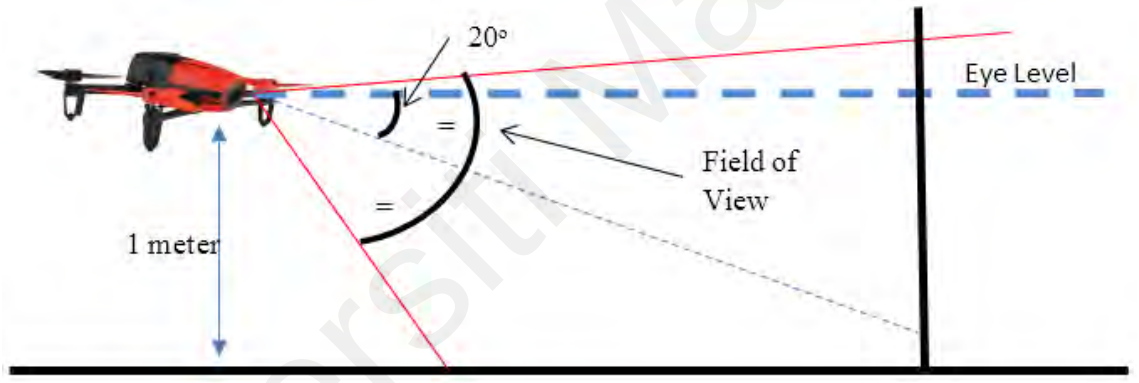


Figure 3.1: General drone set up in this work

A Robot Operating System (ROS) package called the bebop_autonomy is used to communicate with the drone via ROS. The bebop_autonomy serves as a wrapper around official Parrot's Software development kit (SDK), which is called the Parrot's ARDroneSDK3, to ease code development.

For this research, the camera is not calibrated, and it is tilted approximately 20 degrees below its eye level. Figure 3.1 shows the general set up of the drone sending a command to the camera control. The 20 degrees tilt is to maximize the view of the floor while maintaining the view at eye level. The drone's eye level in this research is defined as a

horizontal virtual line, which extends away from the drone camera's z-axis, denoted by the broken blue line in Figure 3.1. It is crucial to note that the eye-level serves as a guide for the drone operator to estimate obstacle height. The drone's eye-level is an excellent indicator to determine when the drone should fly over an obstacle or turn and escape a blocked space.

Active monitoring of the default pitch angle during flight allows for continuous tracking of the drone's aerial position to support navigation. Unlike a wheeled robot, the drone cannot rely on an odometer to measure distance traversed. Instead, the distance covered by the drone is speed-dependent, for example, reaching, on average, five to ten meters per second. The lesson from computing and crossing the MBS gaps in previous works (see Azizul & Yeap, 2015; Azizul & Khanil, 2017) include the need for short-distance intervals. An experiment is done to decide short-distance intervals that suit a drone, testing with one second, two seconds, three seconds, and four seconds of flight time. Two durations have been selected, the two and four seconds. They allow the drone to fly forward for about 1 meter and 3 meters, respectively.

The distances are an approximation and not to scale, because sometimes the drone does not stop immediately after the command is lifted. Programming a reverse pitch is proposed for a navigation system that requires a drone to stop on point. However, the reverse pitch is considered an aggressive motion and not crucial to the MBS method hence not part of the scope of this work. Finally, the drone is assuming taking off from a safe location for the MBS implementation.

3.3 Autonomous navigation for a mono-visual drone

Autonomous navigation concerns the robot's movement and decision making to maintain safe travel and escape from one space to another. The popular method is the object-based approach, where researchers perform obstacle avoidance by tracking features and markers from the environment. A less popular approach is the space-based method that focuses on finding openings in the space for path planning. One such method is called the minimal bounded space (MBS). Central to the MBS computation is space segmentation. For the laser robot of Azizul & Yeap (2015), occlusion information derived from depth perception defines their MBS gaps. For the mono-visual Rovio of Azizul & Khanil (2017), the lack of depth information led to floor segmentation, which influences their MBS gaps.

The interest of this work is to test the notion of MBS gaps on a mono-visual drone. One would suggest that the implementation should be straightforward with the success of the Rovio. While the overall pipeline should look similar, a distinguishing factor is the Rovio's floor segmentation is done with a camera about 12-15cm above the ground. The drone, on the other hand, hovers at about five feet high above the ground. Another defining factor is the Rovio, while grounded, is very stable for shot making. The drone, however, is prone to shake and can fluctuate some while hovering.

The five feet height selection has justifications. One, the height is sensible for an indoor exploration. Two, increasing the height means decreasing light variations. Low light variations can make the floor appear monotone relative to the surrounding in the video stream. Therefore, flying low can conserve the richness of information of the real environment, which is crucial for the floor segmentation to work.

3.4 Data acquisition

The first action when the drone begins to hover is to grab an image from the live stream. Figure 3.2(left) shows a sample image grabbed from a live scene. Figure 3.2(right) shows how the actual drone looks like from a human's point of view (POV). The camera angle is tilted 20 degrees below the drone's eye level to snap the image. The next time the drone grabs another image is after it has crossed an MBS gap. The drone will fly following the two seconds or four seconds forward move, before pausing to evaluate the space with the MBS method. The drone will have five seconds waiting time upon pausing to stabilize.

Figure 3.2: The drone's POV (left) and the human's POV (right)

3.5 A new segmentation process for the MBS method

The image is then passed through three segmentation filters, centered at the image center. The difference between all three is the shape of the filter box, a rectangle, a triangle, and an inverse triangle. At the base of each filter box, the width is about twice the size of the drone. The idea behind the filter boxes is derived from observing the type of spaces in front of the drone. Space can be a corridor-like opening, stable in size and shape, similar to the rectangular filter box. Space can get wider the further the environment goes; thus, the inverse triangle filter box is proposed. Also, space can get narrower with a broader opening near the drone's base, and the triangle is used to define a filter box.

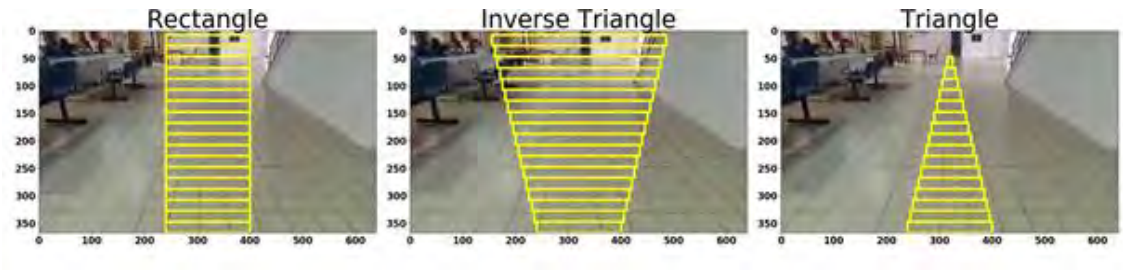


Figure 3.3: Three filter boxes, rectangle, triangle, and the inverse triangle

Each filter box is divided further into several layering zones. The height of each layer is about 50 pixels, and they imitate the depth of the physical space since the mono-vision gives none. The idea is to correlate the depth of space observed in an image to the depth space in the real world. For the drone's flying height and tilt of the camera, it is estimated the 50 pixels represent about 5 meters in the real environment. These filter boxes propose calculation of the pixel difference for each layering zones before getting the mean of the distance. The shape of the segmentation filters (bottom width x height x top width) are 160 x 360 x 160, 160 x 360 x 330 and 160 x 320 x 10 for rectangle, inverse triangle and triangle, respectively.

3.5.1 MBS boundaries for the drone

The pixel difference between the layering zones for the rectangle filter box is calculated based on the absolute error for each of the red, green, and blue (RGB) channels. For the triangle and inverse triangle filter boxes, an additional extra resize process is applied to the smaller layers to compensate for the difference in layering size. Equation (1) shows the formula to calculate the pixel difference.

Figure 3.4 shows the results of the pixel distance calculation separated by each of the RGB channels. Interestingly, the different color channels propose different curve patterns for each filter box, due to the richness of information captured by an image.

$$d^{c,i} = \sum_{x=0}^{X} \sum_{y=0}^{Y} abs\left(P_{x,y}^{i} - P_{x,y}^{i+1}\right) \quad \text{.....(1)}$$

Where d = distance, pixel
c = color channel, i.e. blue, red, green
P = pixel value at x,y with value [0,255]
i = segmentation box
$i+1$ = segmentation box above the current segmentation box
x = coordinate x of segmentation box
y = coordinate y of segmentation box

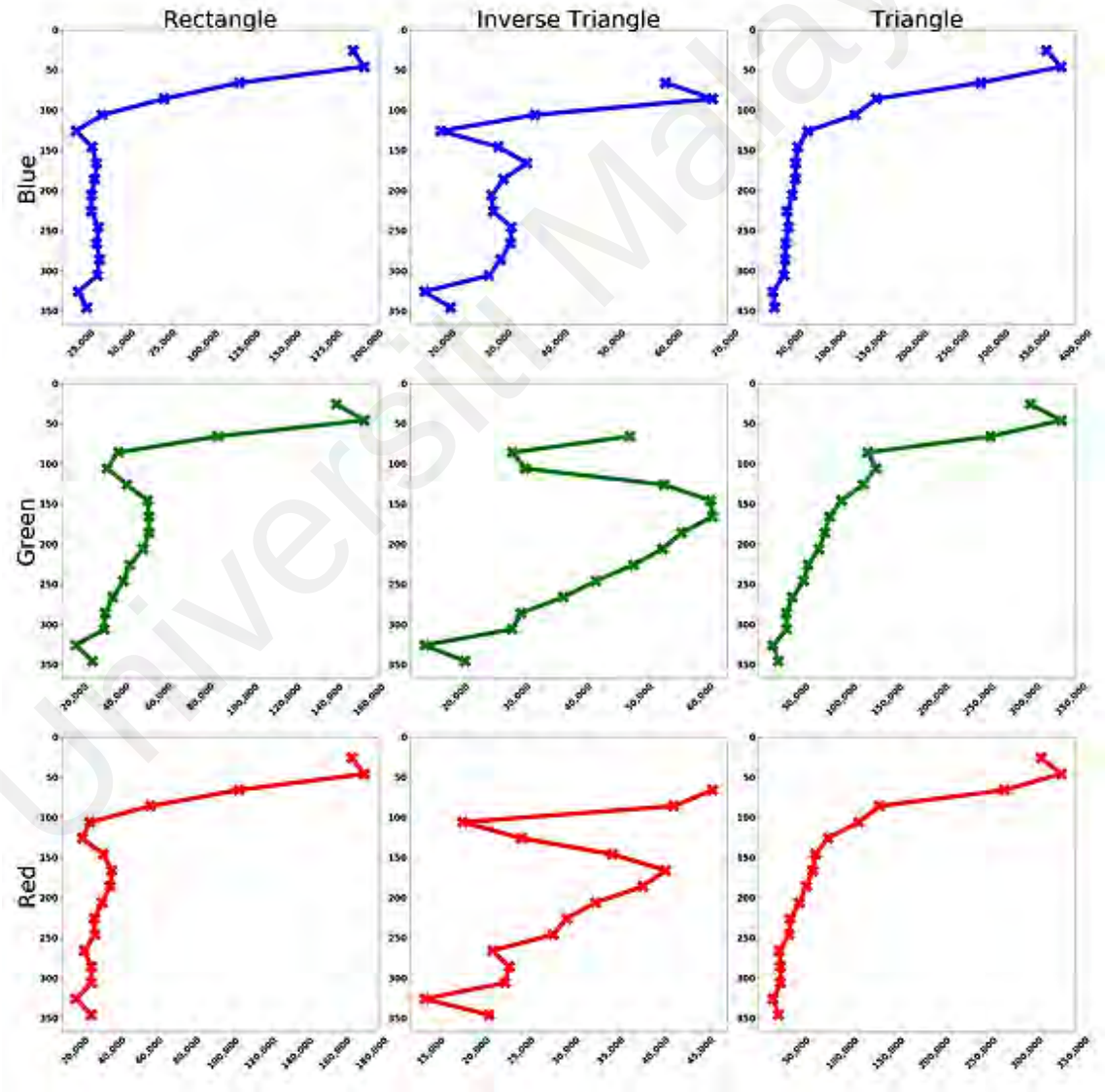


Figure 3.4: The RGB pixel distance difference for each filter box

3.5.2 Computing obstacles in the MBS boundary

Next is calculating the mean of the pixel distances for each of the filter box. The visualization of the mean for the RGB channels is shown in Figure 3.5. The horizontal axis denotes the pixel value distances, and the vertical axis denotes the layering zones separated at 50 pixels each. The furthest layering zone (see value 0 on the vertical axis) denotes the furthest space from the drone in the real world. The tracking of the RGB pixel value begins from the base (at marker 350 on the vertical axis) since the pixel value represents the floor in the real world. For the rectangle filter box, the pixel distance values on the horizontal axis range between 20,000 and 40,000 and goes up the vertical axis from 350 to marker 100. The assumption here is that, if the pixel value at marker 350 represents the floor, then tracking the value means tracking the floor. For the rectangular filter box, the pixel value of the floor significantly changes at marker 100, indicating the floor ending near marker 100 in the image and about 25 meters in the real world.

A similar trend is projected for the triangle filter box. The value at marker 350 is considerably stable until around marker 150, indicating the floor can reach about 20 meters from the drone in the real world. Figure 3.3 shows the rectangle and triangle filter boxes have only floor inside their respective segments. In contrast, the inverse triangle filter box captured not just the floor but also other objects like chairs and tables between marker 150 and marker 0 (on the vertical axis). The floor is also a bit reflective in some areas, introducing light variations to the inverse triangle filter box. As a result, the pixel value across the horizontal axis is less stable.

Figure 3.5 also includes a mean for the pixel value on the horizontal axis (see the vertical broken line in yellow). This mean line separates the floor data into two groups, the certain floor group on the left side of the mean line and the uncertain floor group on the right. The confidence is high when there is only one intersection along the mean line. The rectangle and triangle filter boxes observe the confidence floor for about 25 meters and 20 meters, respectively, while the inverse triangle group is not conclusive about the floor confidence (more than one intersection observed for the mean line). Figure 3.6 shows one intersection line for the rectangle filter box.

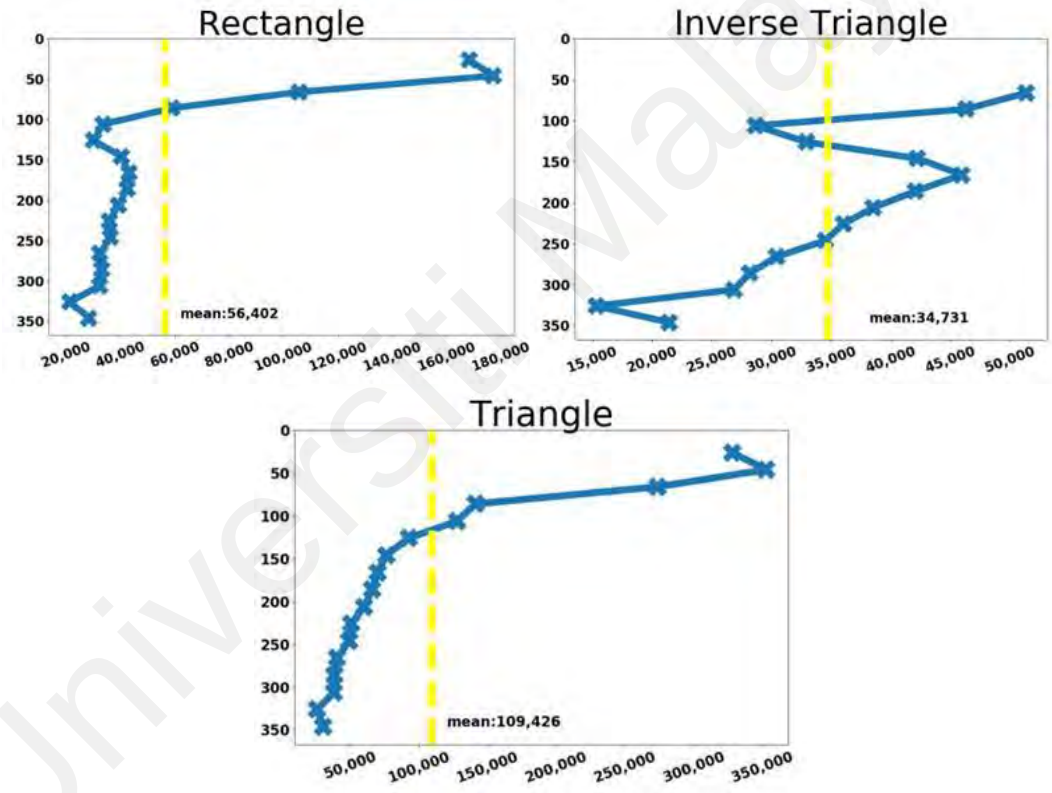


Figure 3.5: Visualizing the mean line for each filter box; the highest mean is the triangle, and the lowest mean is the inverse triangle in the example

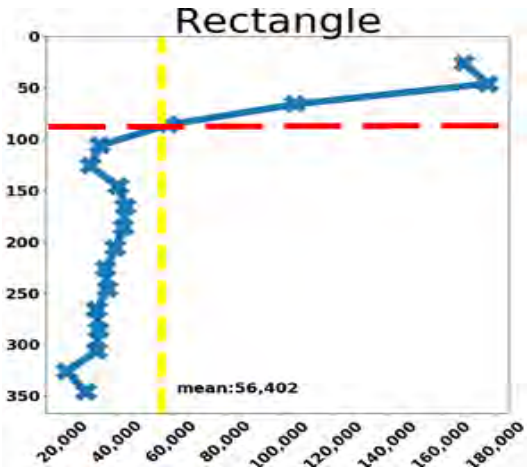


Figure 3.6: Single intersection on the mean line observed for the rectangle filter box

The MBS method suggests a local space to contain the boundary and openings of the environment. Segmenting the floor using the filter boxes allows the drone to have an impression of the space the drone is in. The next step is to decide which filter box is the better option to compute the obstacle inside the MBS boundary. The filter box with the high confidence of the floor is the rectangle and triangle. Examining the mean of the two filter boxes shows the rectangle mean is 56,402 while the triangle mean is 109,426. A lower mean indicates a lower variation (more stable) on the pixel distances on the horizontal axis. Therefore, the rectangular filter box is the better candidate over the triangle for the MBS boundary. The intersection on the mean line can specify the location of possible obstacles for the MBS boundary.

The intersecting line or obstacle line decides how much space the drone has to explore in the current MBS before a potential collision with obstacles. Figure 3.6 shows that the obstacle line of the MBS is located about 25 meters in front of the drone. However, a drone, like other navigating robots, is susceptible to drift. The drift can cause robots to steer away from the path planning, especially if the target location is far-reaching. The MBS method proposes crossing gaps before the MBS boundary to minimize drift errors while sticking to the path planning.

3.5.3 Avoiding obstacles and path planning

The real world is complex, and there are segmentation results that extract more than one intersection on the mean line, such as the inverse triangle filter box in Figure 3.5. Multiple intersection lines can mean two possibilities; one, the environment has tall obstacles, a potential for collision for the drone, or two, there are slight variations on the pixilation due to changes in brightness. When tall obstacles are observed, there should be different jumps in the pixel value. When lighting is the issue, the difference in pixel value should be small. Two noise removing algorithms is proposed to determine whether the MBS contains tall obstacles to avoid or the issue is slight illumination variations. Ideally, the intersections can be reduced to three or less for lighting issues. The first algorithm checks if the intersections can be reduced by shifting the mean line to the left.

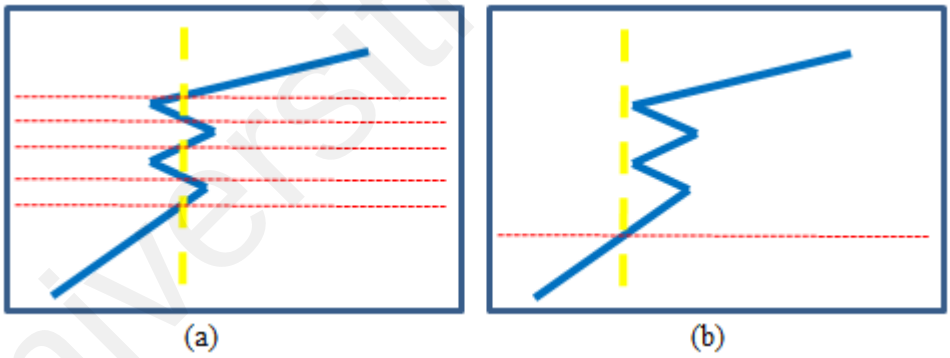


Figure 3.7: Shifting the mean line to the left to reduce the number of intersections on the mean line

Figure 3.7 is a hypothetical example of the segmentation having an illumination issue. Note how the changes in pixel value are minimal. A slight shift of the mean line to the left reduces the number of the intersection to one. Left is selected because that side of the mean line denotes the confidence floor. Note that before the line is left-shifted in Figure 3.7(a), the lowest intersection is higher than after (see Figure 3.7(b)). Algorithm 1 shows how shifting the mean line is done.

Algorithm 1: Find New Mean if Horizontal Line > 3

```
Result: New Horizontal Lines, New Mean
interval = (mean - min(axis x))/100;
counter = interval;
if number of horizontal lines > 3 then
    while number of horizontal lines > 3 do
        new horizontal lines by moving mean lines to the left by amount counter;
        counter = counter + interval;
    end
    new mean = mean - counter - interval;
end
```

Algorithm two reduces the number of intersections by combining the intersection line as one. If the height separating one intersection line and the next is lesser than 20 pixels, it means the difference in pixel value is significantly less. When a combination is required, the higher intersecting line is deleted while, the lower intersecting line is retained. Figure 3.8 describes the condition. Algorithm 2 shows how the combination is processed.

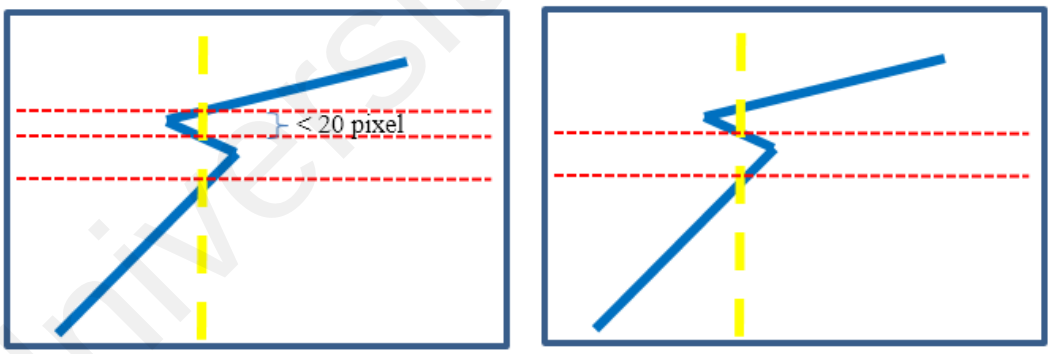


Figure 3.8: Less than 20 pixels distance separating the two top intersections in (a). They are combined in (b).

Algorithm 2: Combine Line that are Close to each other

```
Result: New Horizontal Lines
if number of horizontal lines == 3 then
    if (horizontal line 1 - horizontal line 2) < 20 then
        if (horizontal line 2 - horizontal line 3) < 20 then
            new horizontal line = []horizontal line 1]
        else
            new horizontal line = [horizontal line 1,horizontal line 3]
        end
    else
        new horizontal line = [horizontal line 1, horizontal line 2]
    end
else
    if (horizontal line 1 - horizontal line 2) < 20 then
        new horizontal line = []horizontal line 1]
    end
end
```

Algorithm 1 and 2 are two noise removing algorithms proposed to handle segmentation data. The algorithms consider shifting the mean line to the left, then check if the intersections can be combined if they are separated by less than 20 pixels. The procedures leave the segmentations with one, two, or three intersecting lines. Figure 3.9 shows three final intersecting lines for the MBS.

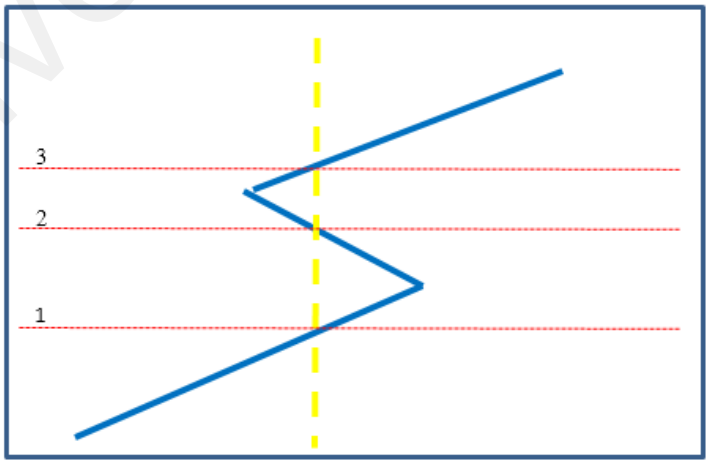


Figure 3.9: An MBS with three intersection lines

The remaining three intersecting lines are strong candidates for the location of obstacles and should be considered in obstacle avoidance. For this reason, two MBS boundaries are fixed in this work, one at marker 100 on the vertical axis, another at marker 220 on the vertical axis. The possibility to deal with tall obstacles requires consideration of whether the drone can fly over them or not. Fixing two MBS boundaries solves this problem. If the tall obstacles are dangerous to the drone, selecting the lower MBS boundary (marker 220) can get the drone fly towards the obstacles and stop before hitting them. If the height of the obstacles is manageable, the drone should fly over it, which makes the upper MBS boundary (marker 100) a better option for path planning.

Figure 3.10 shows the environment in Figure 3.2 side by side with the segmentation outcome. With one obstacle line, only one MBS boundary is required. Note the upper MBS boundary at marker 100 on the vertical axis (denoted in the display in green). The drone's eye level is fixed at marker 50 and denoted in blue in Figure 3.10. In Figure 3.11 and Figure 3.12, more than one intersecting or obstacle line computed, thus requiring both the upper and lower MBS boundaries. The intersection or obstacle line positions relative to the drone's eye level, and the MBS boundaries define if the drone should fly over an obstacle, fly forward, or turn 45 degrees counterclockwise when facing a dead end.

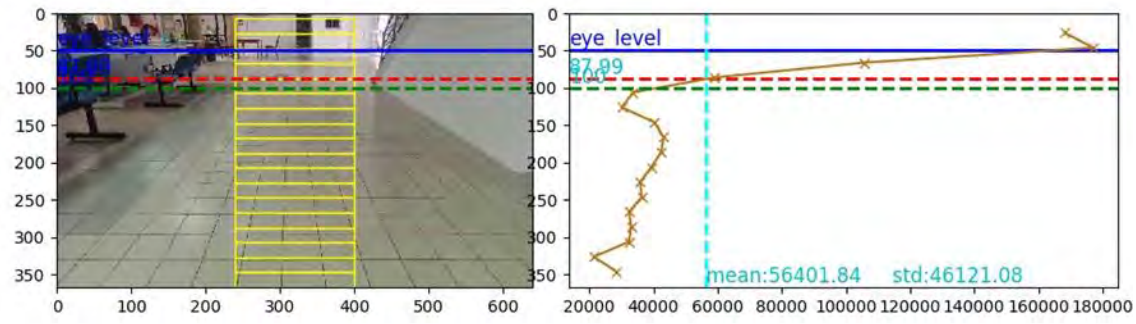


Figure 3.10: Segmentation result showing one intersecting line and one MBS boundary

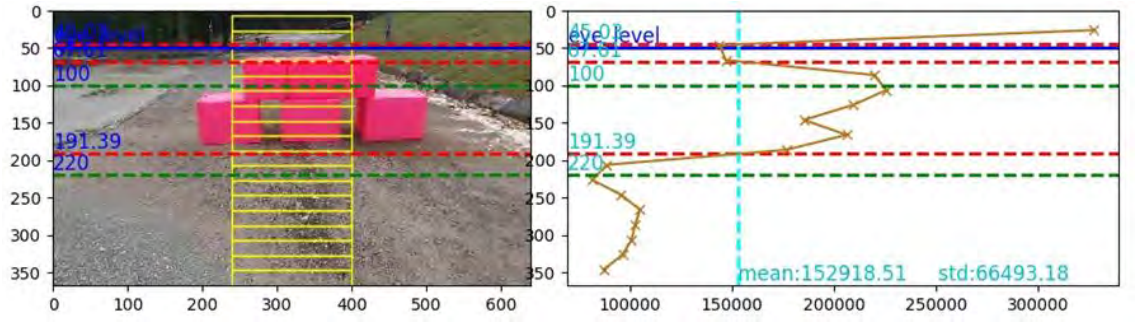


Figure 3.11: Segmentation result showing three intersecting or obstacle lines, and the upper and lower MBS boundaries

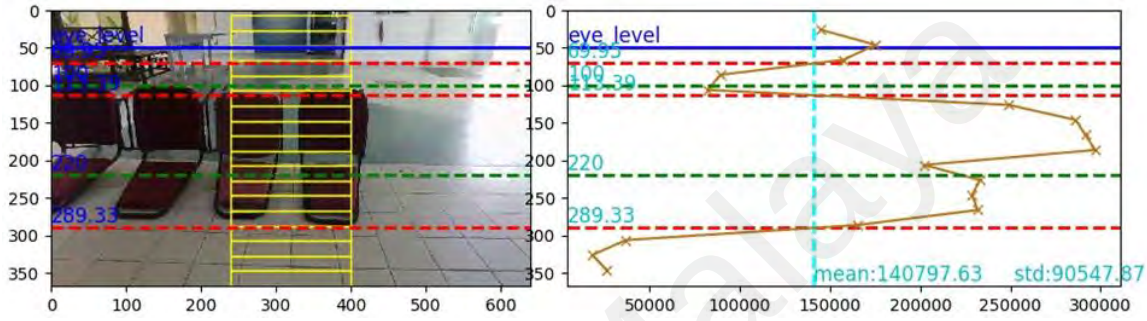


Figure 3.12: Segmentation result showing three obstacle lines, and the upper and lower MBS boundaries

3.5.4 Crossing the MBS boundary

For a single intersection, if the intersecting or obstacle line is above the MBS boundary, it means there is no obstacle within the MBS boundary. Thus, the MBS boundary becomes the drone's target location. Figure 3.10 shows the upper MBS boundary is about 25 meters away from the drone. The drone can fly in the direction of the upper MBS boundary with confidence. However, in terms of flying distance, the drone has two options, fly for two-second or four-second to minimize drift error. The four-second option is suitable since the upper MBS boundary is selected.

When there is more than one intersecting line calculated, the upper and lower MBS boundaries are considered. In Figure 3.11, the top intersecting line exceeds the drone's eye level, indicating tall boundaries dangerous to fly over. The top obstacle line is then ignored in path planning. The bottom obstacle line satisfies the requirement with the

lower MBS boundary, and the shorter distance suggests the two-second flying time is sufficient. When the top obstacle line is lower than the eye level, the upper MBS boundary is selected. The four-second flight time is selected for longer distance matching the upper MBS boundary (see Figure 3.12). When the bottom obstacle line is lower than the lower MBS boundary, that indicates that the drone is facing a dead end, and a turnaround is required to escape. Algorithm 3 shows the decision-making process.

Algorithm 3: Make Decision

```
Result: Action
if number of horizontal lines == 3 then
    if horizontal line 1 >= forward threshold(220) then
        if horizontal line 3 >= fly over threshold(100) then
            Action = "fly over"
        else
            Action = "turn ccw"
        end
    else
        Action = "forward"
    end
else if number of horizontal lines == 1 or 2 then
    if horizontal line 1 >= forward threshold(220) then
        Action = "turn ccw"
    else
        Action = "forward"
    end
else
    Action = "ERROR"
end
```

3.6 System design

For completion, the overall navigation process is summarized in Figure 3.13, while Figure 3.14 visualizes the decision-making process in a flow chart.

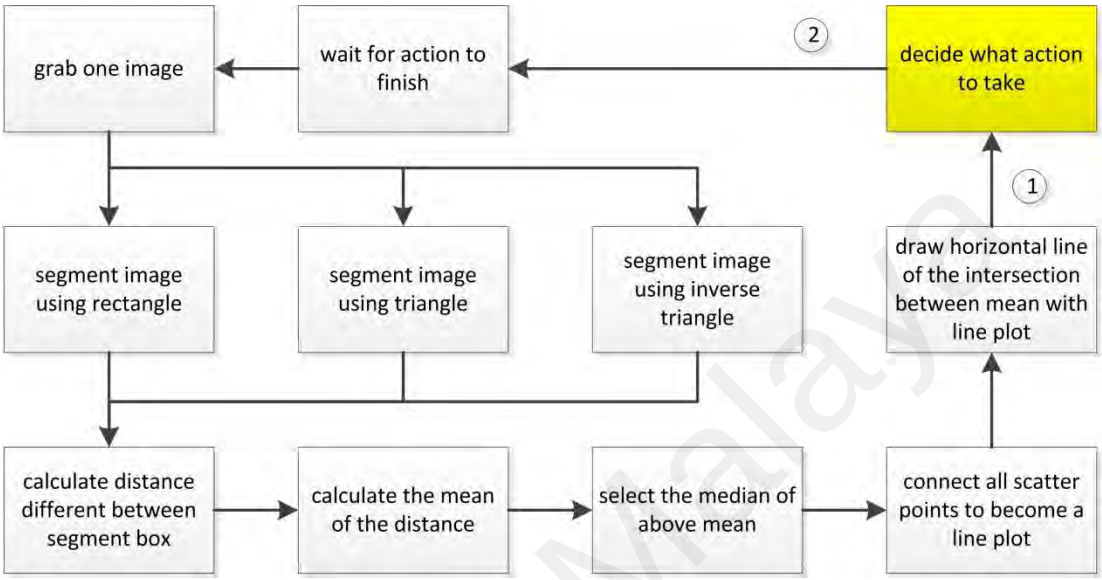


Figure 3.13: The overall navigation process. Number 1 and 2 are the input and output, respectively. The process "decide what action to take" is expanded into Figure 3.14.

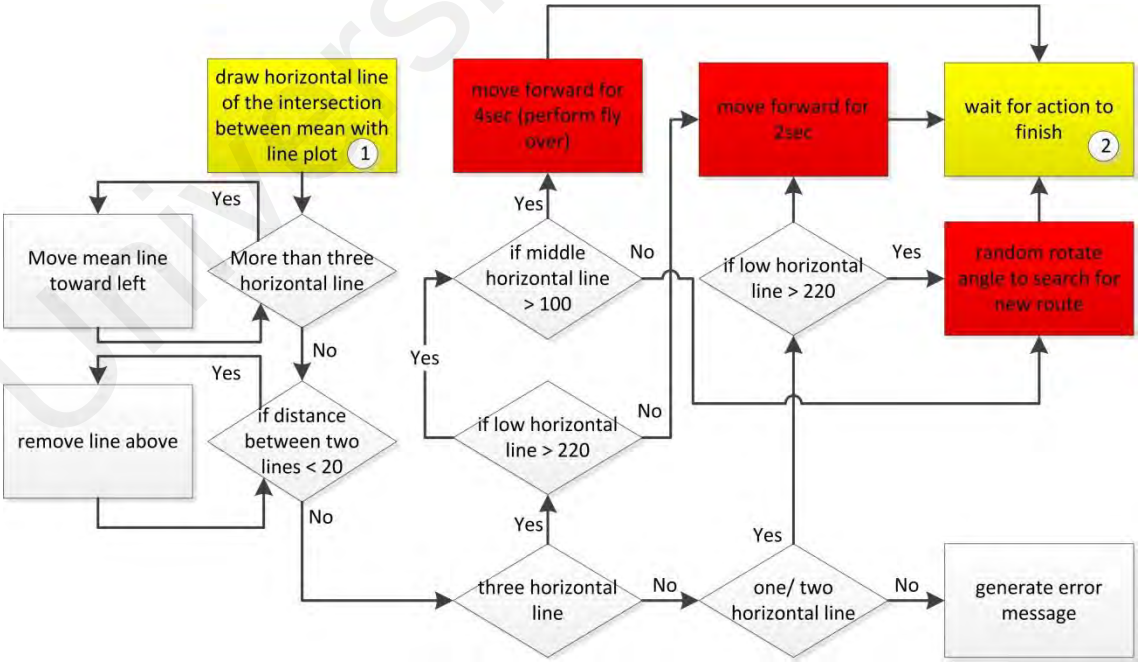


Figure 3.14: Flowchart of the decision-making process. Number 1 and 2 are the input and output, respectively, corresponding to Number 1 and 2 in Figure 3.13.

3.7 Chapter summary

In this chapter, the overall system design and research methodology for drone navigation using the MBS approach is presented. The chapter begins by introducing a new segmentation process for the MBS and methods to evaluate space. Then, the chapter continues with determining boundedness for the MBS, describing the justification for an upper and lower MBS boundary. The decision-making process completes the chapter, where path planning and gap crossing is examined. In the next chapter, the results from field testing with the drone are presented and discussed.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Overview

This chapter presents six experiments on drone navigation using the MBS method developed in Chapter 3. The goal is to demonstrate that the algorithm is robust, at least for drone navigation in an indoor, outdoor, and a semi-outdoor environment. In each experiment, the drone segments space and perform obstacle detection without relying on depth information. Other than demonstrating how the drone builds the MBS representation, the discussion also includes how the drone crosses MBS gaps and perform path planning.

For testing, middleware has been created where all commands from the central controller are redirected to the middleware. The middleware acts as a buffer, so a command can be reviewed first before executing it on the drone. Reviewing the command can avoid mishaps if the command is faulty or sending the drone to a possible crash. On top of the middleware, a fail-safe software sends a stop and land command in an emergency landing.

Throughout this chapter, a diagrammatic approach is adopted to present the results to the reader. In the diagram, the matrix imaging of computers denotes the pixel (0,0) at the top left. However, the drone's position is around (350, 350). Thus, the smaller the pixel value, the further away in-depth it gets from the robot. For example, the marker 100 on the vertical axis has more depth to the drone than marker 220.

4.2 Experiment 1 – navigating in an indoor environment

In this experiment, the drone segment spaces in an indoor foyer of a faculty. The foyer is an open space with a considerably high ceiling. There is furniture in the foyer, such as tables and chairs for students to use in between classes. The lighting condition at the foyer includes natural sunlight and artificial light. A series of 23 images show how the drone navigates autonomously by crossing 23 MBS gaps consecutively across the foyer until it detected obstacles and decide to stop. The total distance covered is about 25 to 30 meters.

Figure 4.1 shows the lower MBS boundary at 220. The upper MBS boundary is at 100, and the drone's eye level at 50. First, the lower MBS boundary is checked against the lowest obstacle line. Obstacles at 143, 149, and 148 pass the lower MBS boundary (they are not between the drone and the lower MBS boundary). Next, the upper MBS boundary is checked. There is only one obstacle line for each image, so the obstacles at 143, 149, and 148 are tested again. The obstacles fail the upper MBS boundary at 100 because they obstruct the drone's line of sight to the upper MBS boundary.

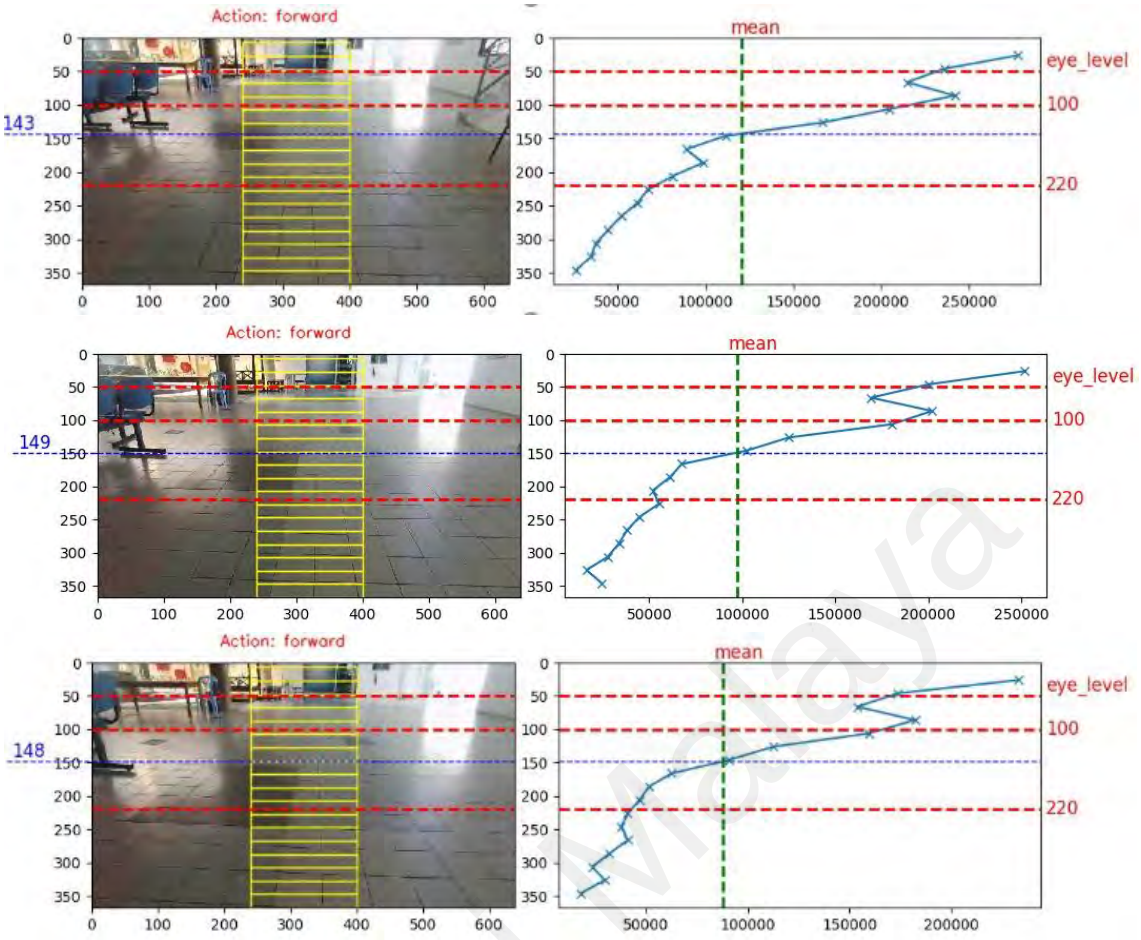


Figure 4.1: The obstacle lines at 143, 149, and 148 fails the upper MBS boundary so the move forward by crossing the lower MBS boundary (two-seconds flight time).

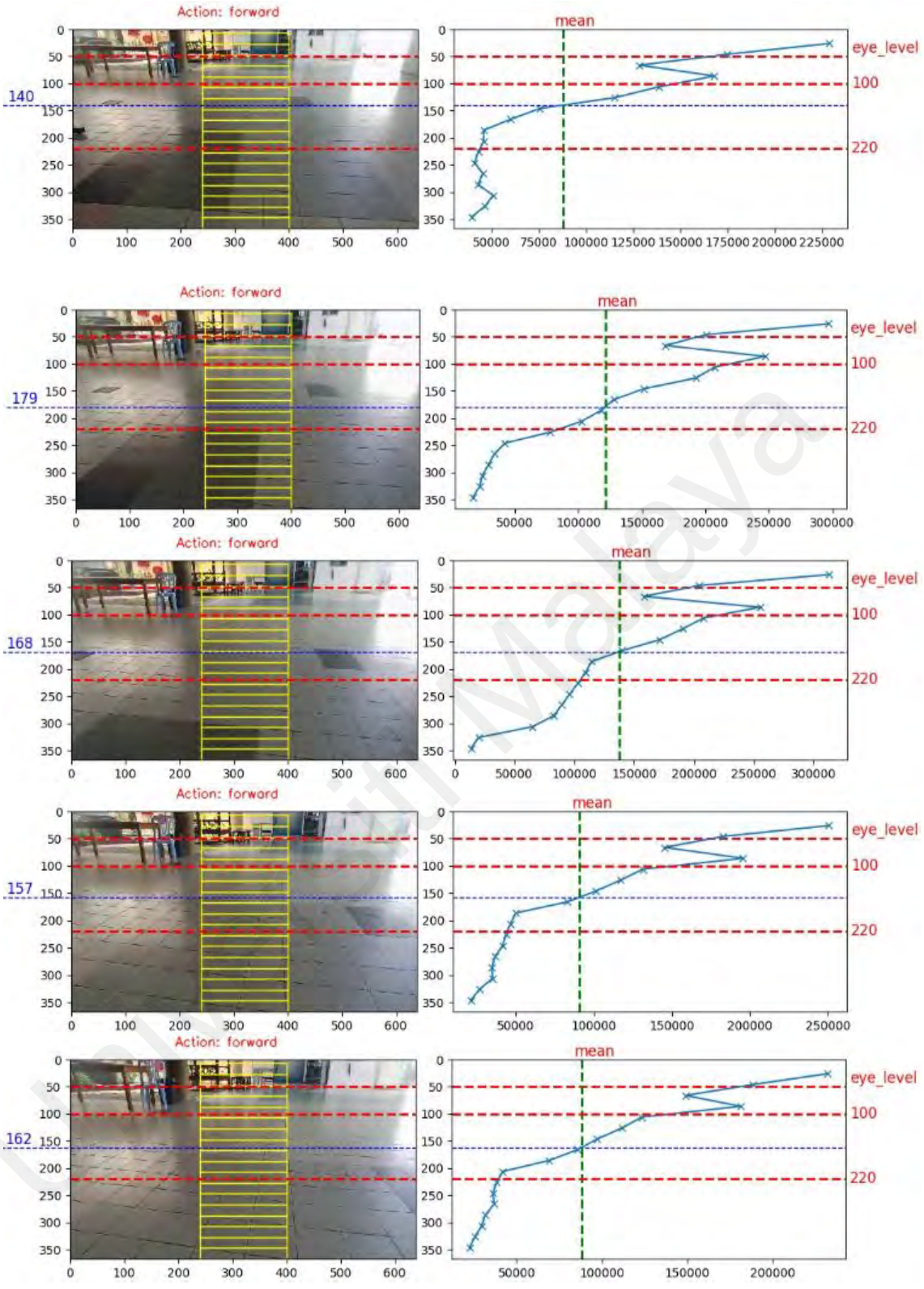


Figure 4.2: Single obstacle lines at 140, 179, 168, 157, and 162 all passes only the lower MBS boundary. The drone moves forward by crossing only the lower MBS boundary.

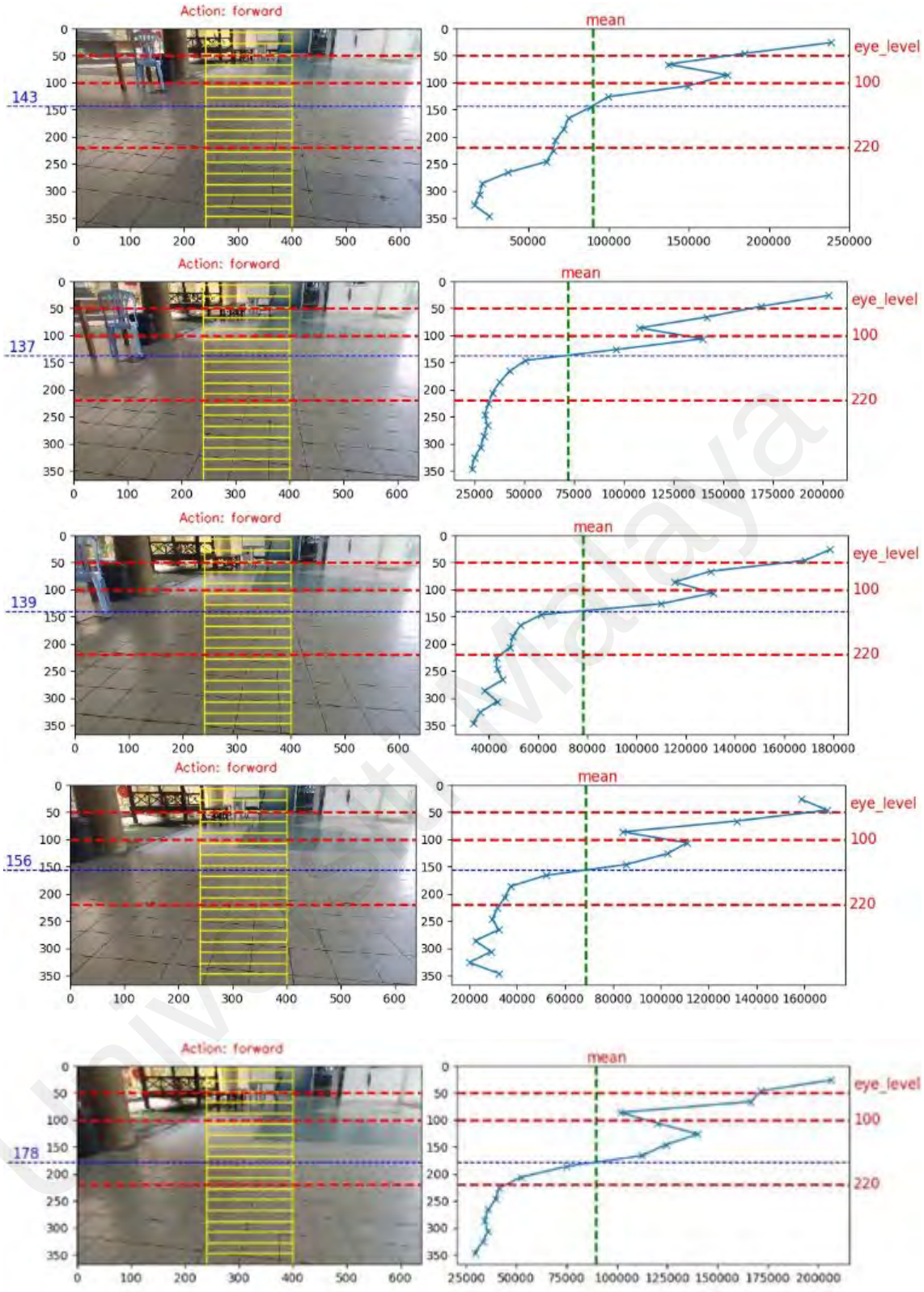


Figure 4.3: Single obstacle lines ranging from 137 to 178 passes only the lower MBS boundary suggesting forward movement to the drone.

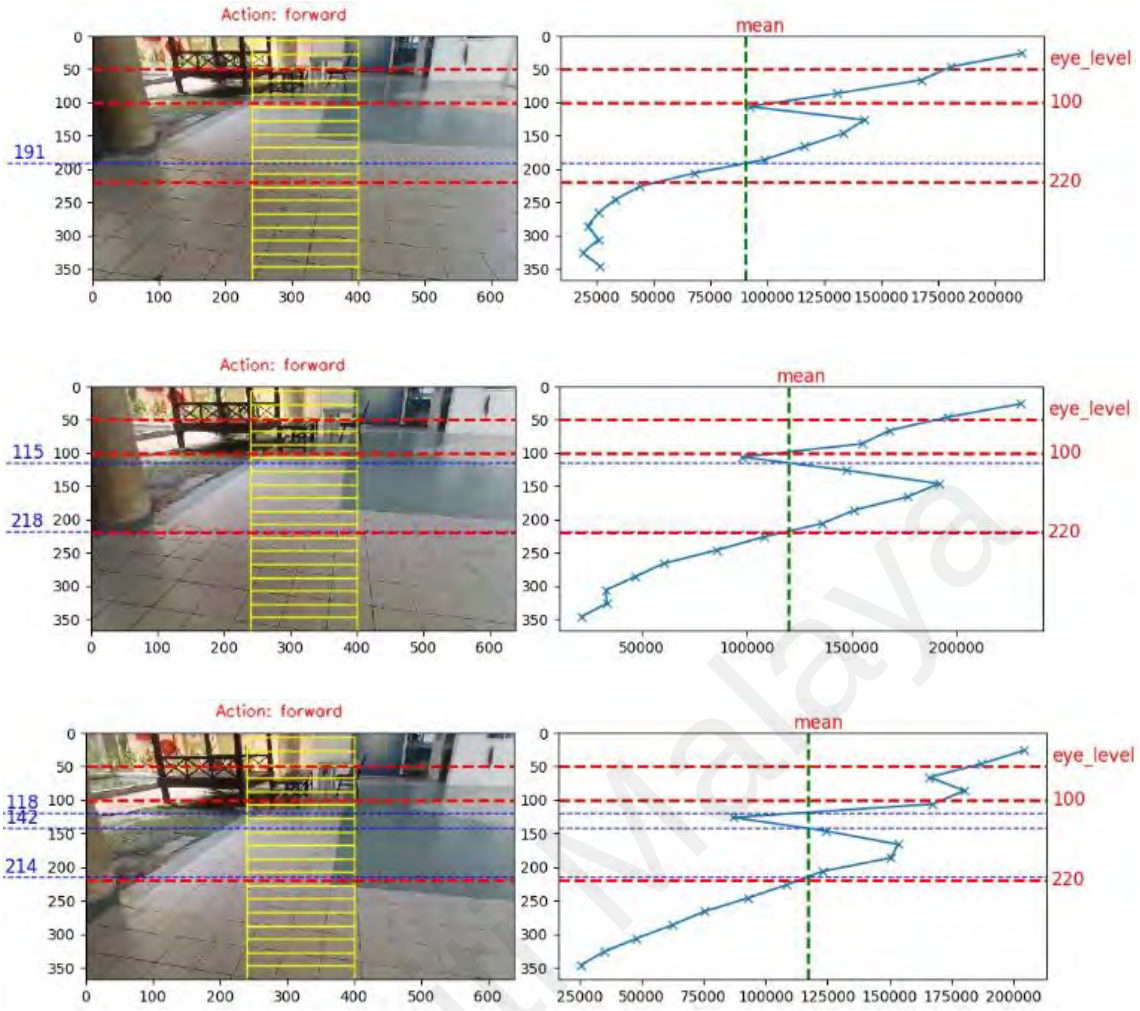


Figure 4.4: Obstacle lines are more than one in some steps. None of them passes the upper MBS boundary (value must range between 50 and 100). The drone moves forward, crossing the lower MBS boundary

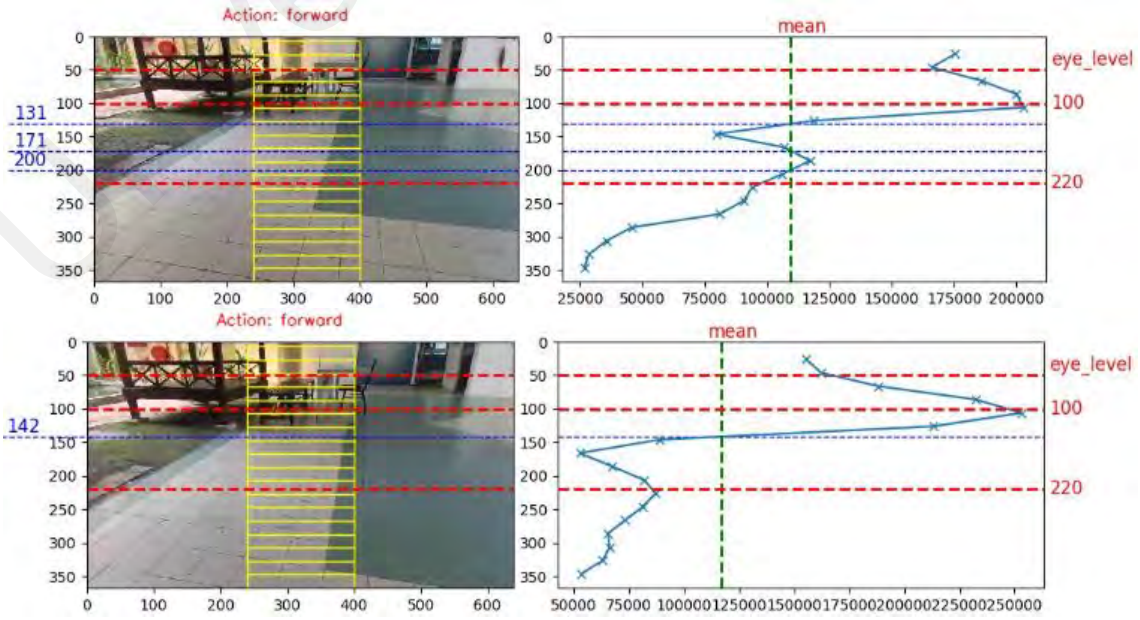


Figure 4.5: No obstacle lines have a value between 50-100, indicating the drone cannot decide to cross the upper MBS boundary for safety reasons.

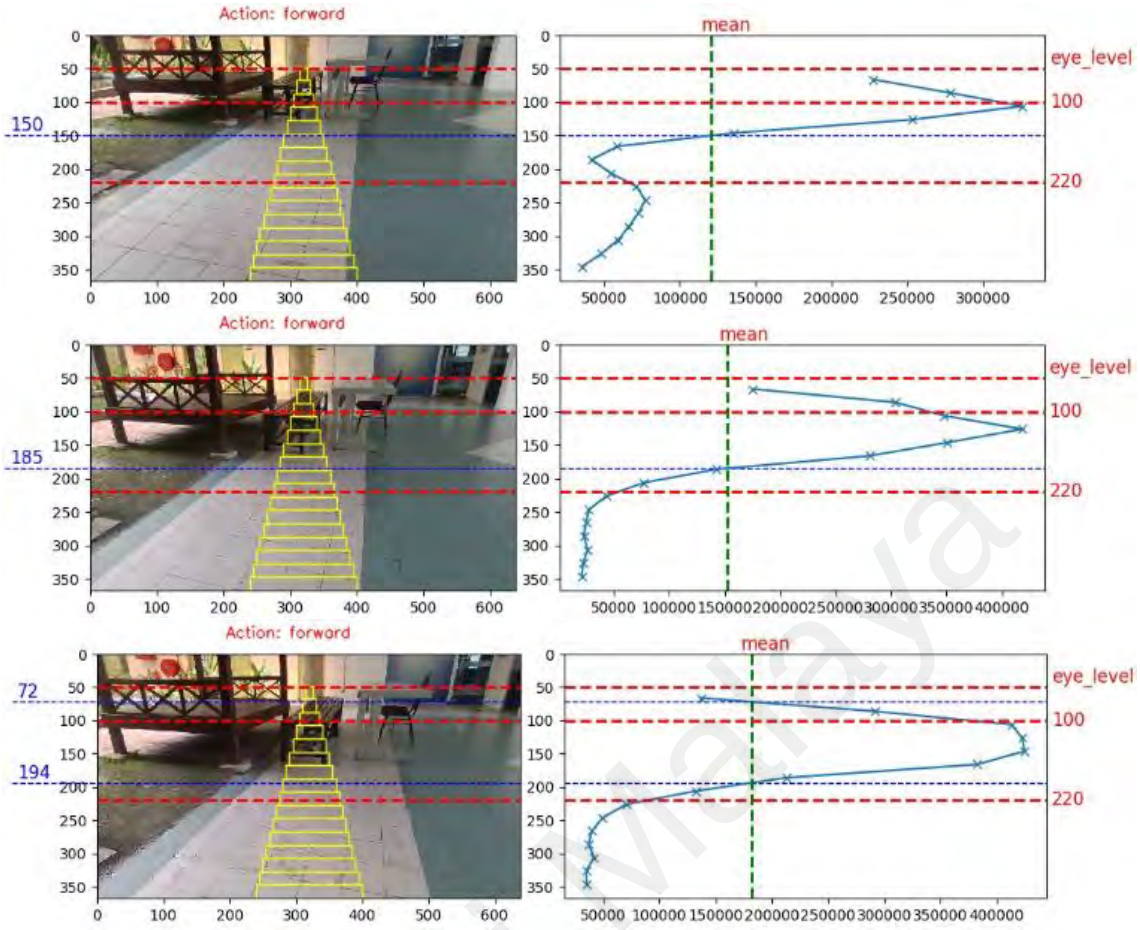


Figure 4.6: The drift has steered the drone a little as it goes forward. Only one of the obstacle line has a value between 50 and 100. However, the other obstacle line in the image has a value of 194. It would be dangerous to attempt the upper MBS boundary.

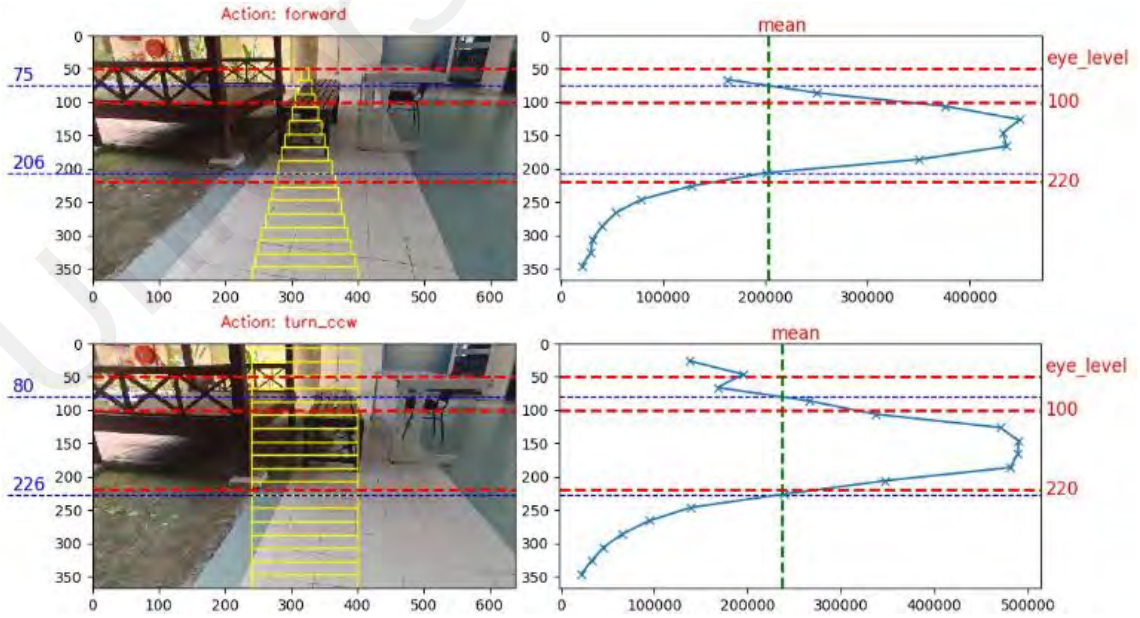


Figure 4.7: In the top image, there is an obstacle line with a value between 50 and 100. However, its pair does not pass the upper MBS boundary. The drone can only choose the lower MBS boundary to cross. In the bottom image, both obstacle lines fail even the lower MBS boundary indicating tight space for the drone. The drone plans an escape by turning.

4.3 Experiment 2 – avoiding obstacles in an outdoor environment

The goal of this experiment is to place tall obstacles in the drone's space and test if the drone can avoid them following the MBS method. The outdoor environment is an open parking place, with tar roads as the floor. The lighting condition is entirely natural sunlight, and the wind speed is minimal during testing. A series of 13 images are captured, denoting the 13 steps or MBS gaps crossed by the drone. Figure 4.6 shows the first image.

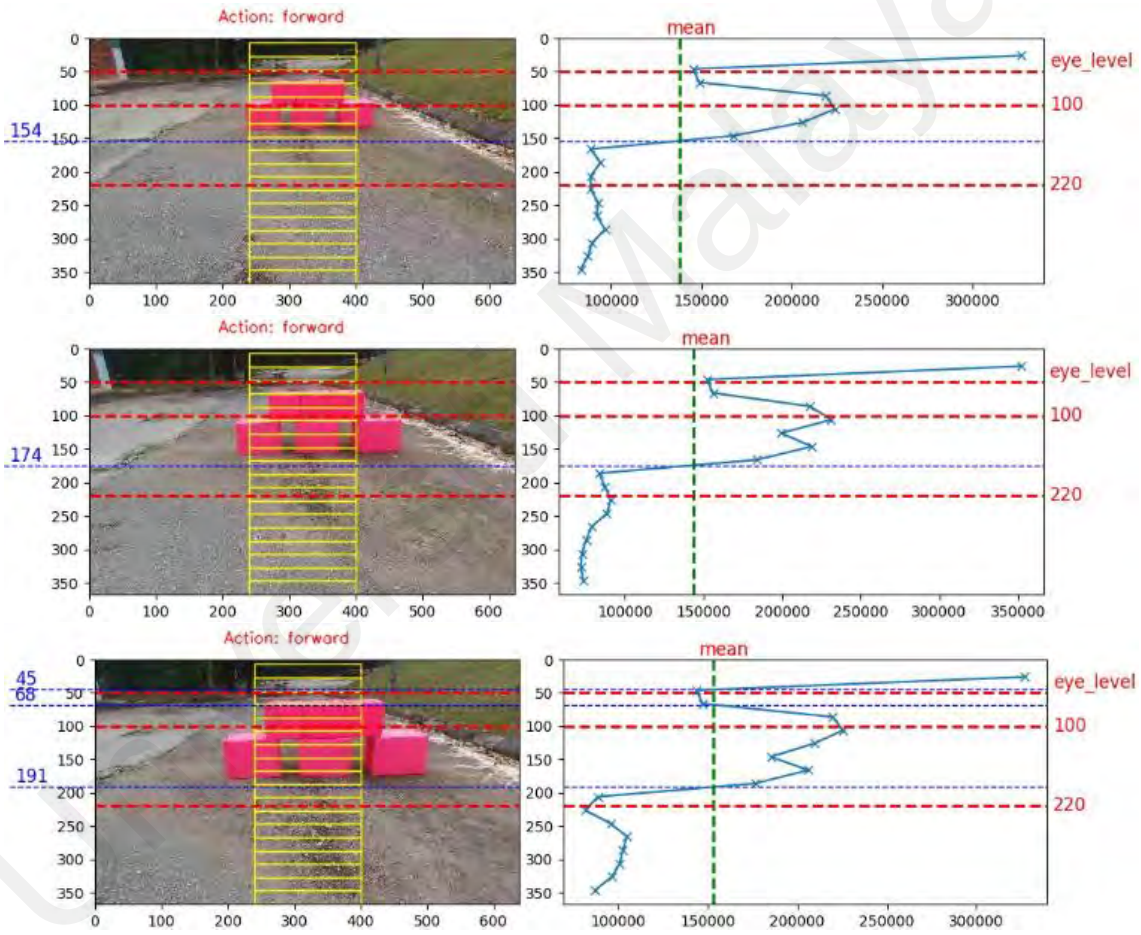


Figure 4.8: The first two images show the obstacle lines detected at 154 and 174, passing the lower MBS boundary but not the upper one. The bottom image is showing three obstacle lines at 45, 68, and 191. Obstacles 191 and 68 do not pass the upper MBS boundary, so the drone can only move forward by crossing the lower MBS boundary.

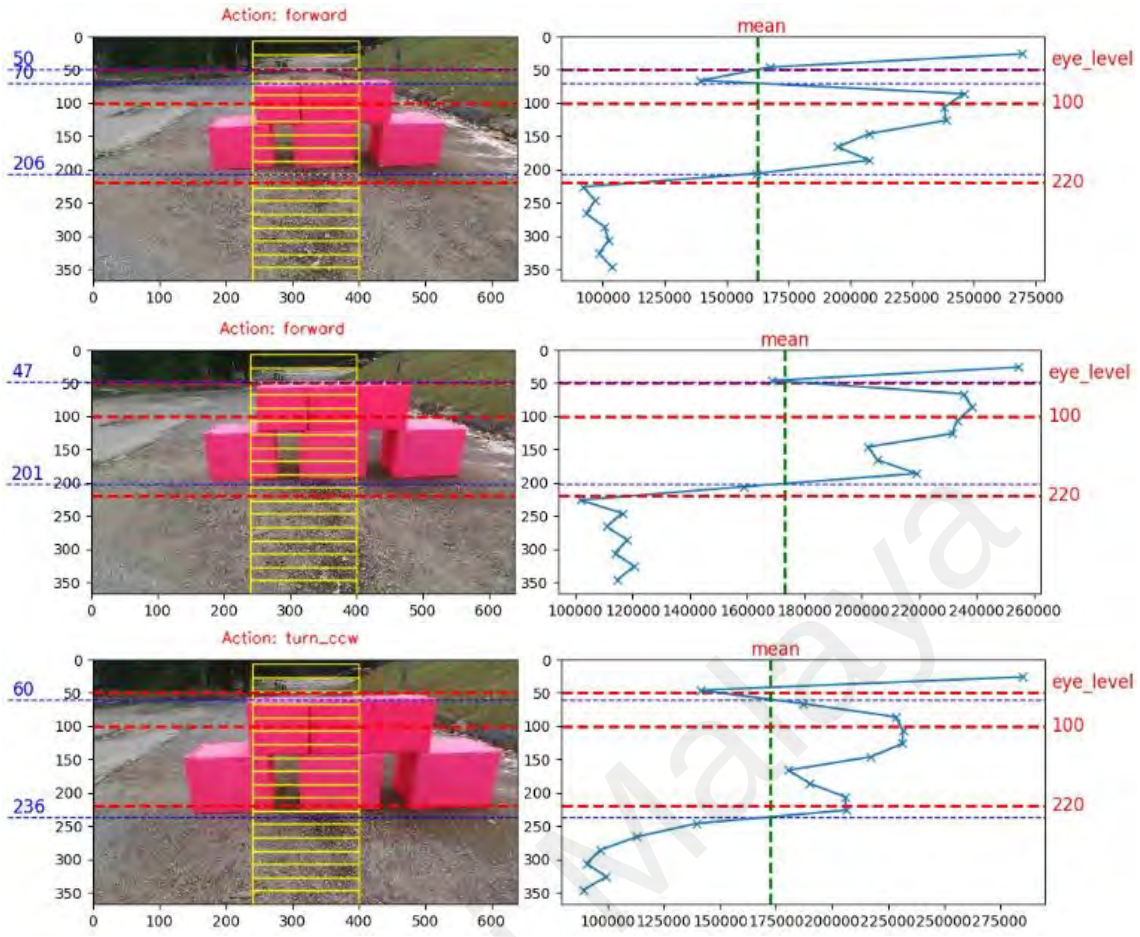


Figure 4.9: The top two images showing obstacle lines passing only the lower MBS boundary. In the bottom image, one obstacle line is obstructing the drone from getting to the lower MBS boundary indicating tight spaces. The drone escapes by turning away.

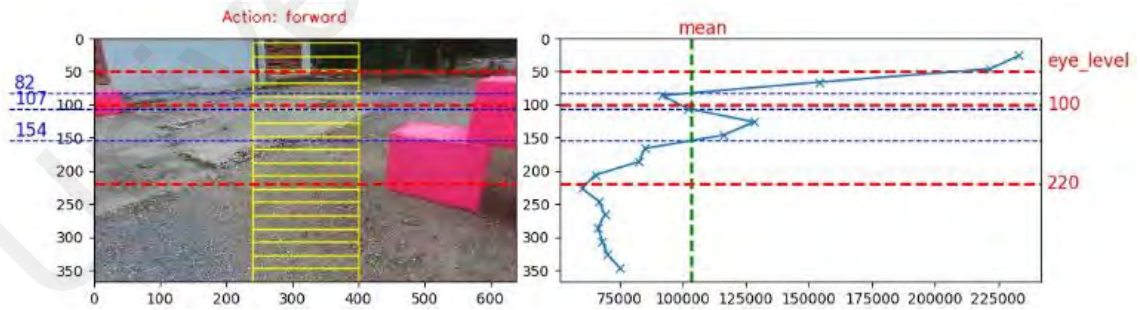


Figure 4.10: The drone successfully turns 45 degrees counter clock-wise and segments a different space. Some texture difference on the ground creates a change in the pixel values and adds another obstacle line. The three obstacle lines at 82, 107, and 154 only satisfy the lower MBS boundary.

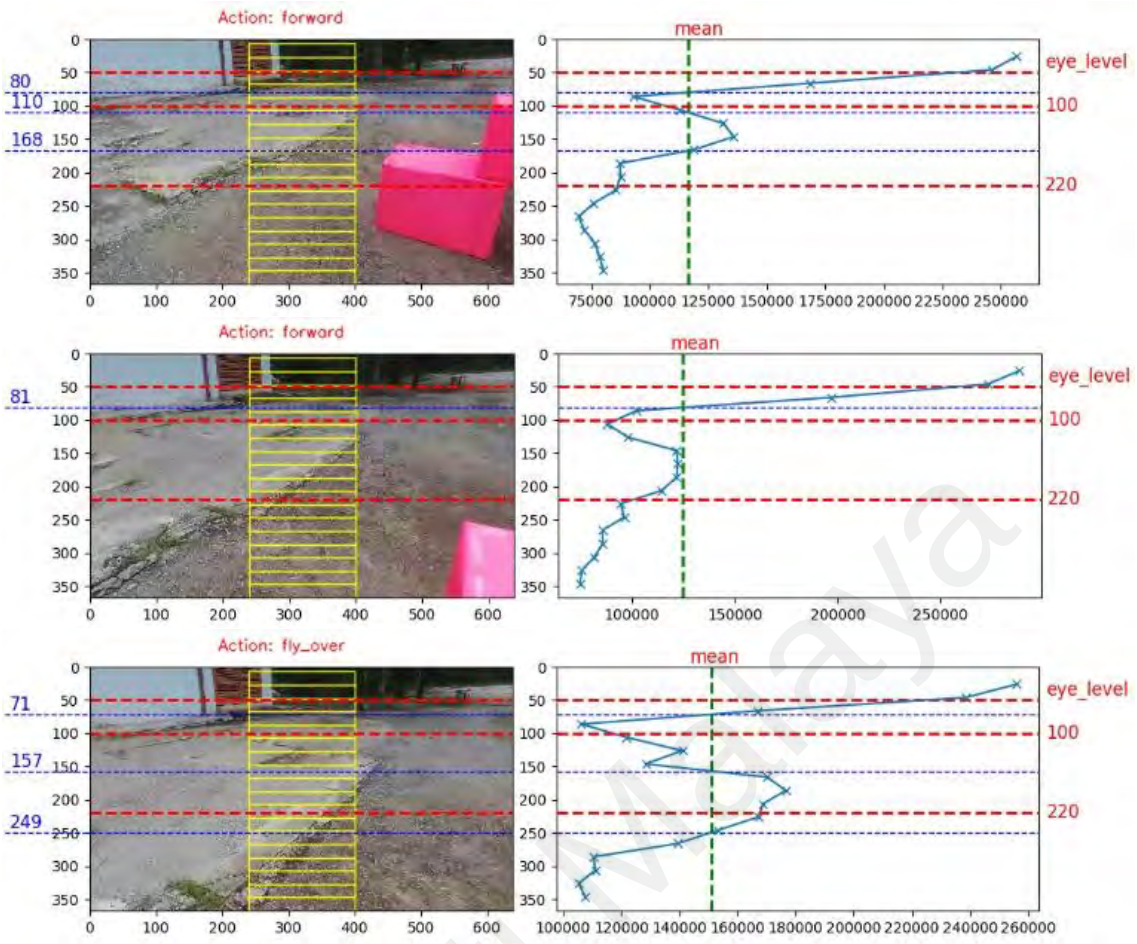


Figure 4.11: The top image has obstacle lines passing only the lower MBS boundary. The middle image has a single obstacle line at 81, passing the 50-100 value. Here, the drone can move forward by crossing the upper MBS boundary. The bottom image has a bottom obstacle line at 249, which fails the lower MBS boundary. However, this can be ignored because the top and middle obstacle lines are between the lower MBS boundary and the eye-level indicating a flyover is recommended.

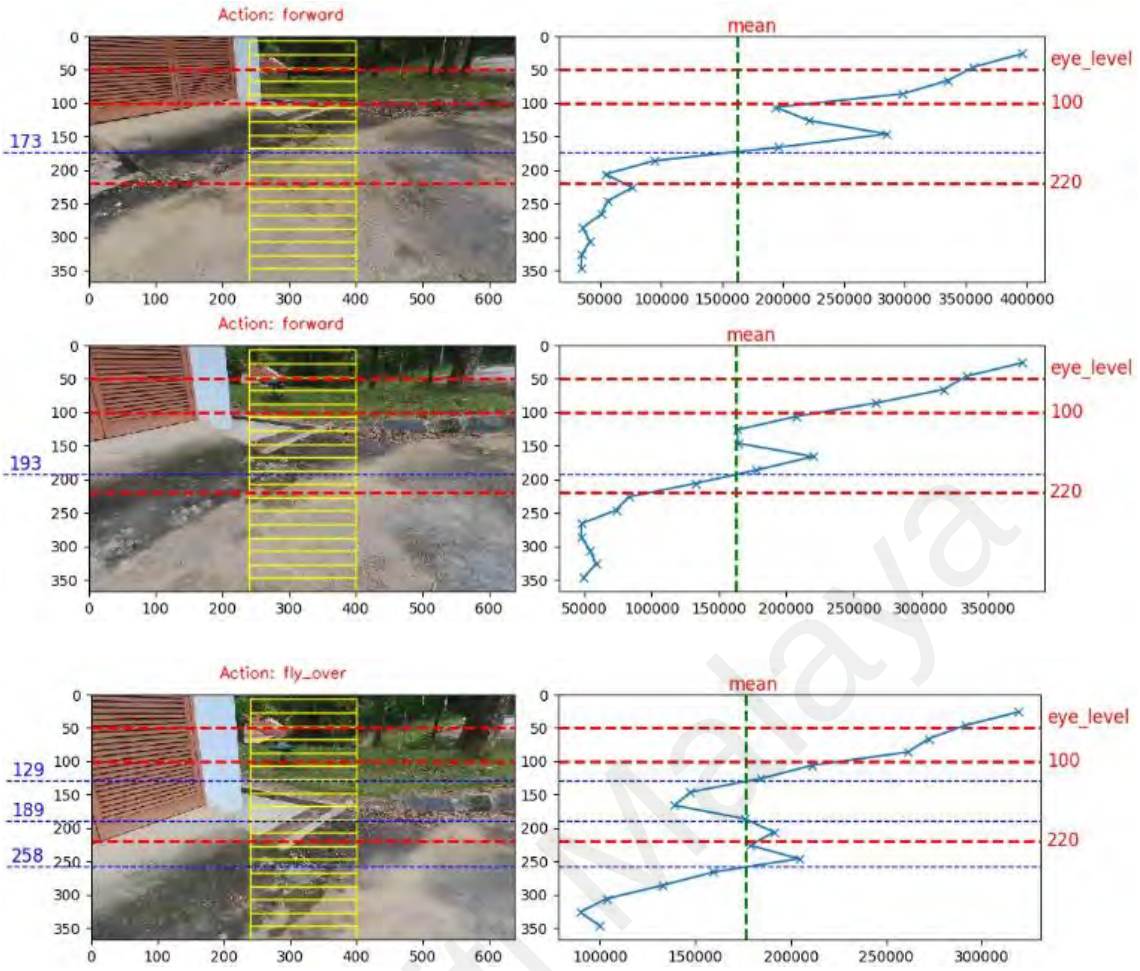


Figure 4.12: Obstacle lines at 173 and 193 pass only the lower MBS boundary. In the last image, the bottom obstacle line at 258 fails the lower MBS boundary, but that is negligible because obstacle lines 129 and 189 are between the upper and the lower MBS boundaries. The drone can fly over safely.

4.4 Experiment 3 – handling obstacle dynamics in an outdoor environment

Another outdoor experiment is conducted, and this time, two sets of obstacles are introduced. The first set is placed nearer to the drone's line of sight while the second set is placed behind the first one. There is a height difference between the two. The first set is ground obstacles that the drone should be able to fly over. The second set is high and can obstruct the drone. The test is designed to observe the robustness of the MBS method dealing with an abrupt change in obstacle information.

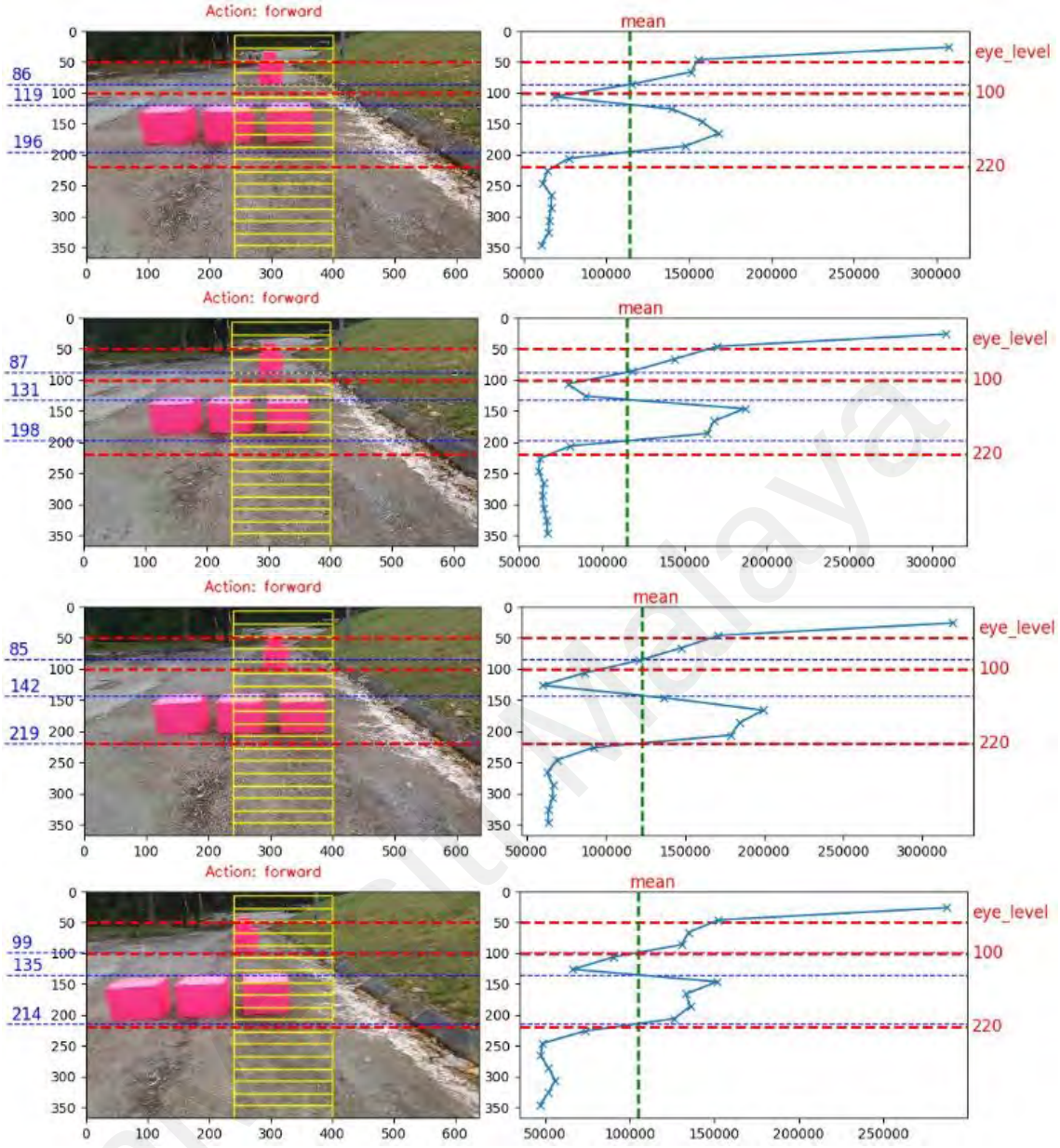


Figure 4.13: Two sets of obstacles in the drone's space. Obstacle lines ranging from 85 to 219 have no problem allowing the drone to cross the lower MBS boundary.

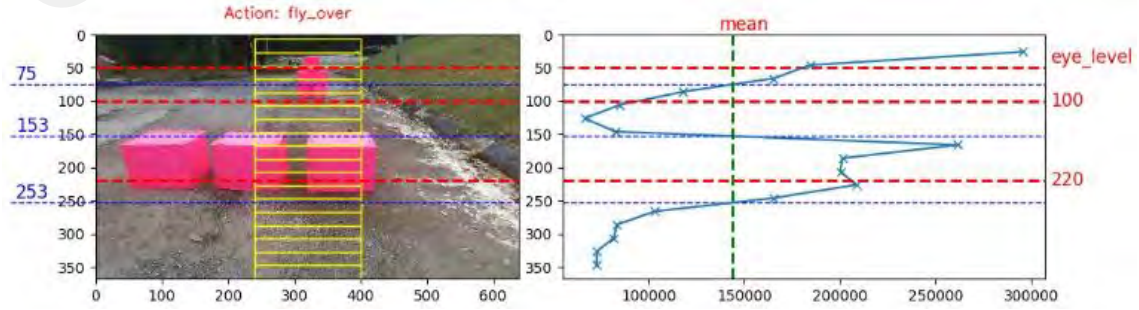


Figure 4.14: Bottom obstacle line 253, failing the lower MBS boundary. However, this can be ignored because the obstacle line 75 and 153 are in between the lower MBS boundary and drone's eye-level suggesting a fly over.

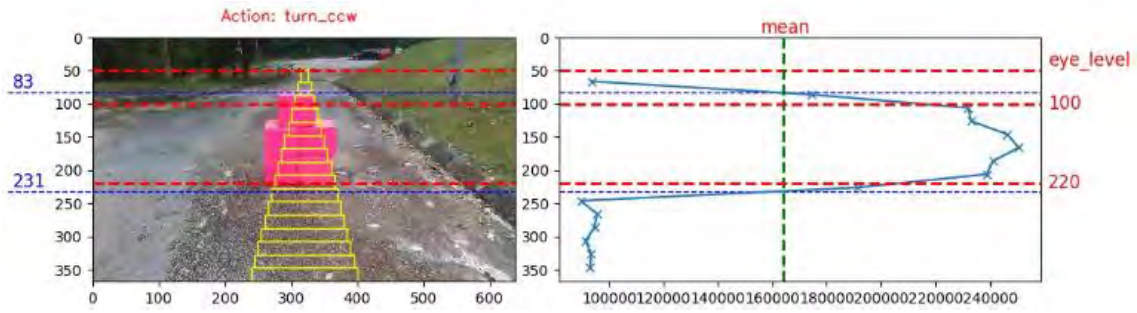


Figure 4.15: Obstacle line 83 passes the upper MBS boundary. However, the bottom obstacle, line 231, fails the lower MBS boundary. Without a middle obstacle line, it is not possible to check for a fly over. Thus, the drone has to skip the space by turning away.

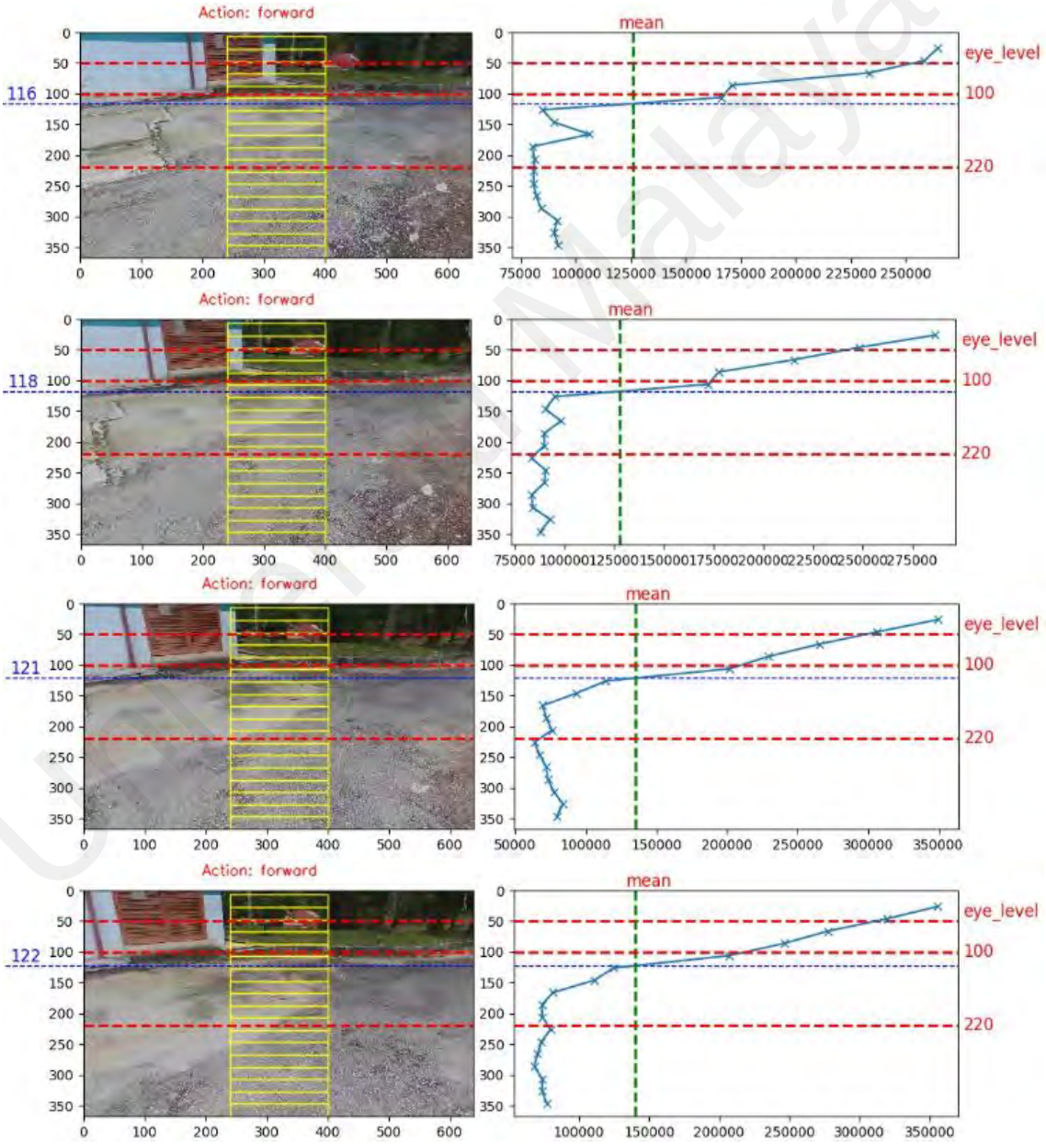


Figure 4.16: After the turn, the space segments single obstacle lines at 116, 118, 121, and 122, all passing only the lower MBS boundary.

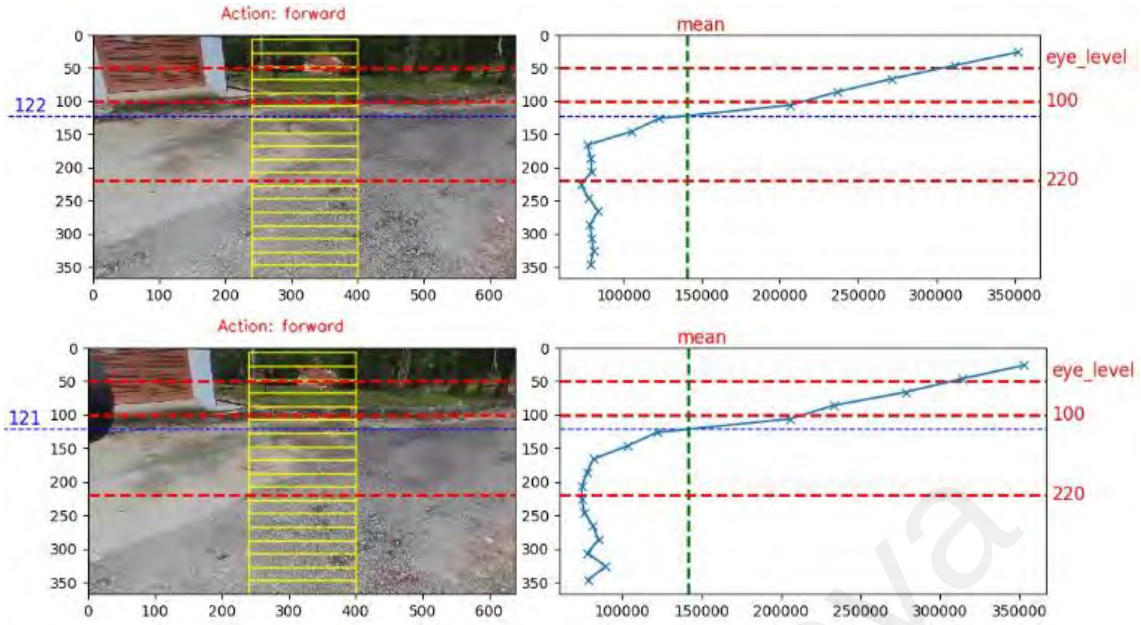


Figure 4.17: The obstacle lines passing the lower MBS boundary for the drone to move forward.

In Figure 4.13, a fault has occurred to the drone navigation system. The decision making at each step is correct, i.e., forward movement crossing the lower MBS boundary. However, the drone did not move. It hangs in the air and keeps rescanning the environment at five seconds interval. Failure to move is not detected sooner because the system design is not equipped with any localization technique. So, there is no way for the central controller to know if the drone has physically moved.

4.5 Experiment 4 – avoiding ground obstacles in the outdoor environment

The goal of this experiment is to test the effectiveness of the drone to fly over two consecutive short obstacles. The drone has to move as close as possible to the first obstacle, perform a fly over, then move as close as possible to the second obstacle before executing another fly over.

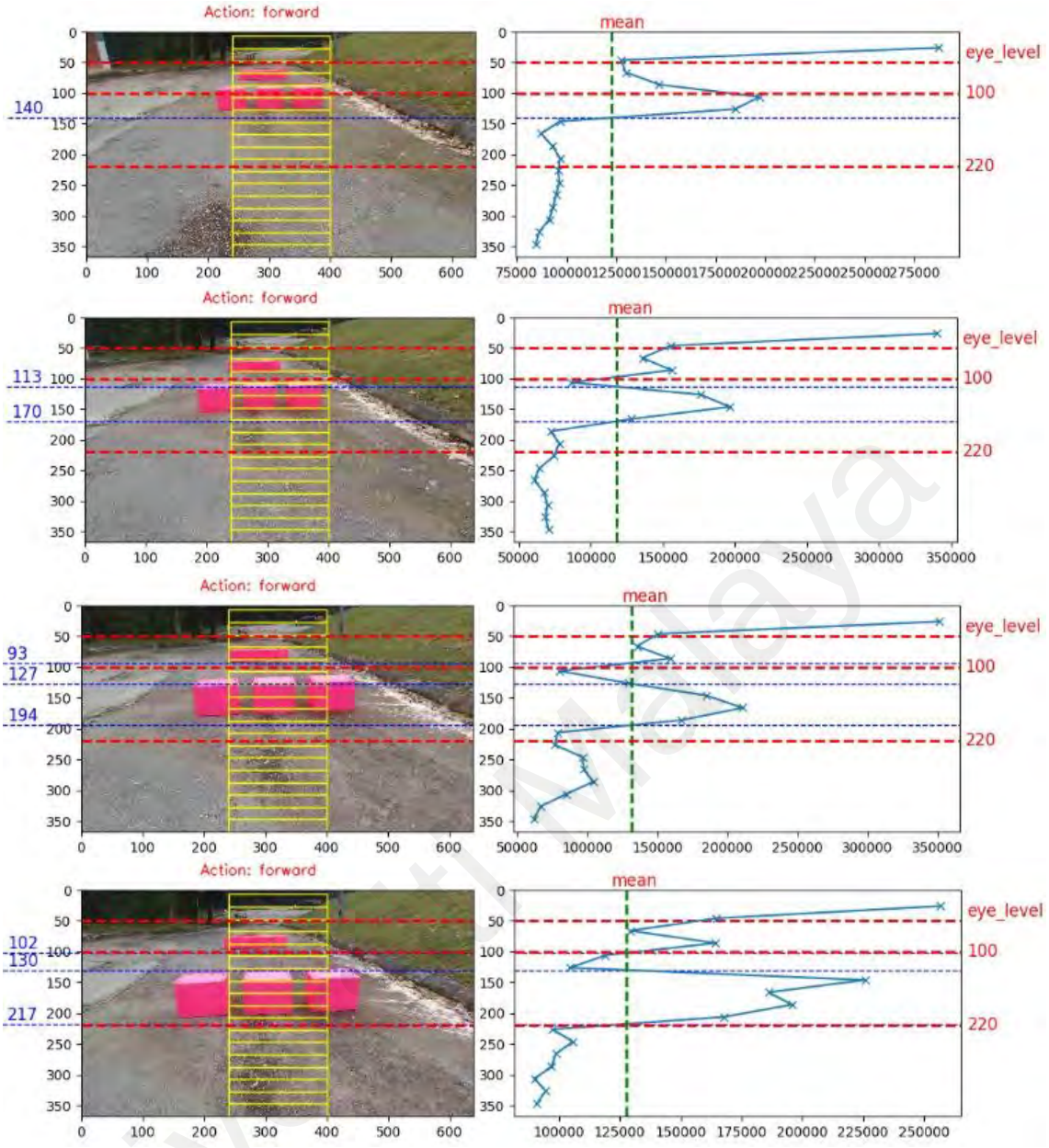


Figure 4.18: Obstacle lines for the first two images allowing the drone to cross only the lower MBS boundary. In the third image, for the first time, all three obstacle lines 93, 127, and 194 are between the lower MBS boundary and the eye-level. However, the bottom obstacle line has to fail the lower MBS boundary for fly over decision. The last image showing the obstacle lines passing the lower MBS boundary.

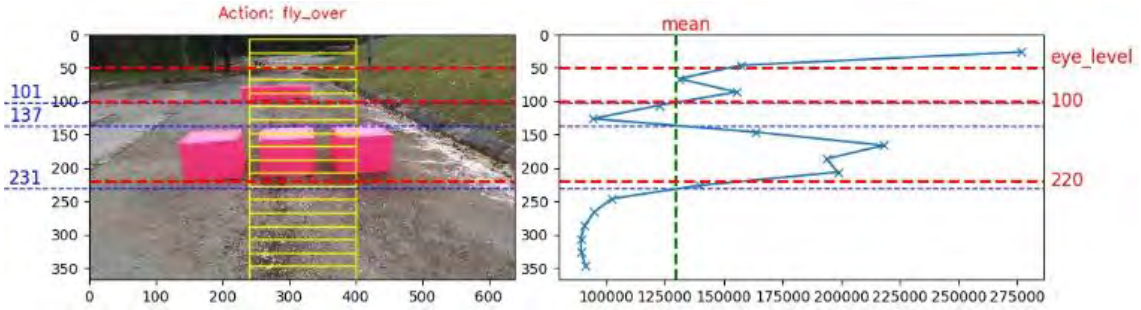


Figure 4.19: A fly over decision for the drone. The top and middle obstacle lines (101 and 137) between the eye-level and the lower MBS boundary, and, the bottom obstacle line at 231 failing the lower MBS boundary.

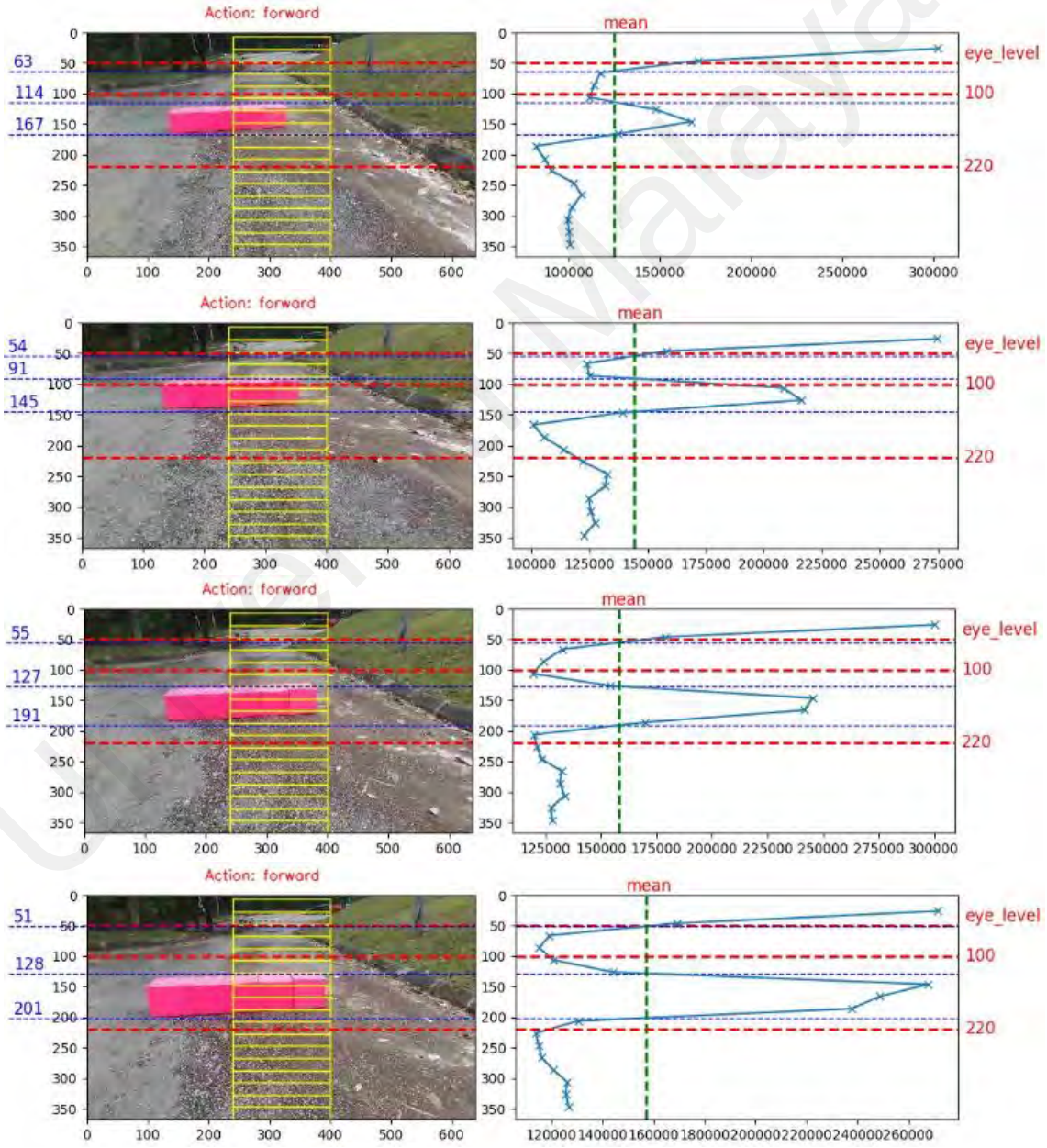


Figure 4.20: Interesting combination of obstacle lines observed. With a range from 51 to 201, the obstacle lines pass the upper MBS boundary. The drone moves with confidence crossing several upper MBS boundaries consecutively at four-seconds flight time each

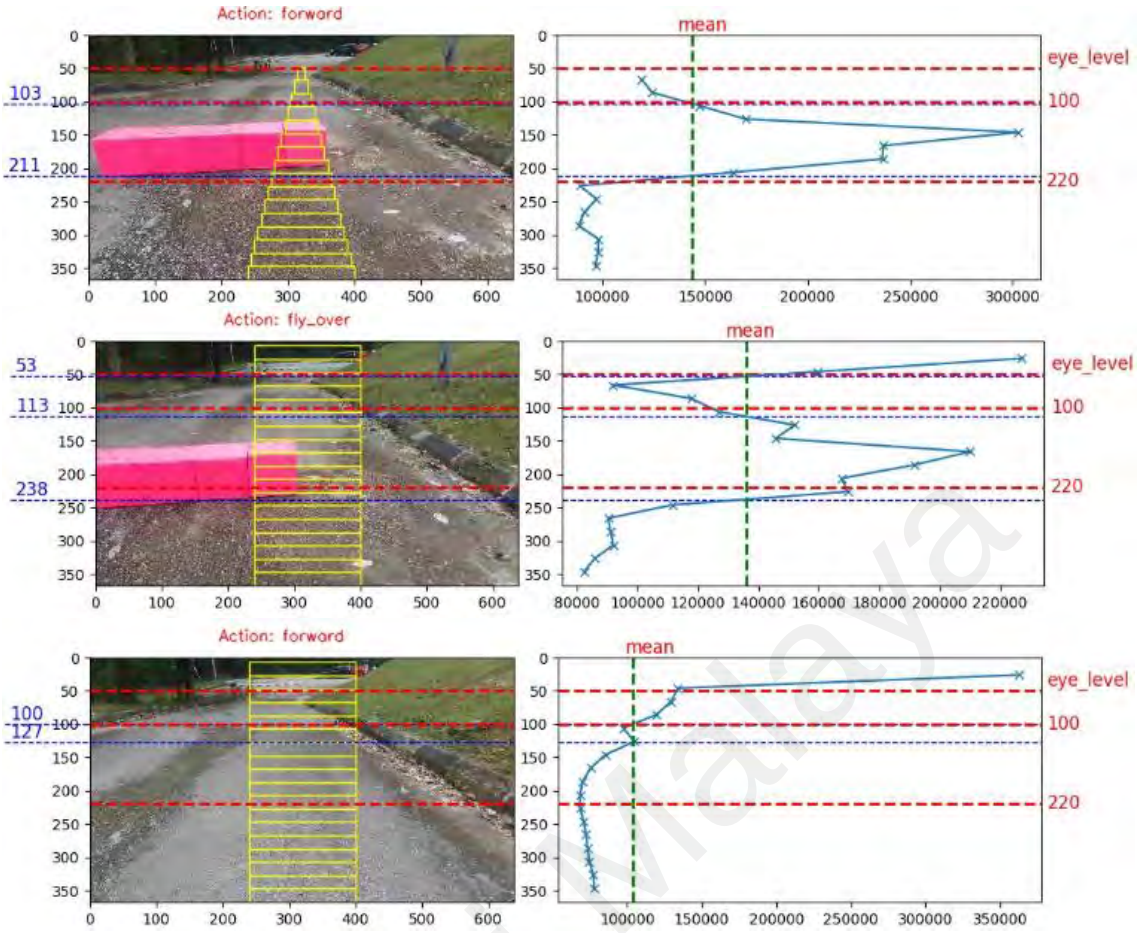


Figure 4.21: The top image sees the drone crossing the upper MBS boundary with obstacle lines 103 and 211. The middle image has an obstacle line 238 failing the lower MBS boundary but with obstacle lines 53 and 113 between the eye-level and lower MBS boundary to decide a fly over. The bottom image has an obstacle line 127, passing the lower MBS boundary.

4.6 Experiment 5 – obstacles in semi-outdoor environment

The goal of this experiment is to test autonomous mono-visual drone navigation in a semi-outdoor environment. Semi-outdoor environments post a different challenge to floor segmentation because of uneven lighting and shade. Traditional floor segmentation algorithms always fail in such an environment. The environment selected is an outdoor office pavement, an open space receiving natural sunlight, and shaded under the rooftop. The cemented flooring can be reflective when it is bright.

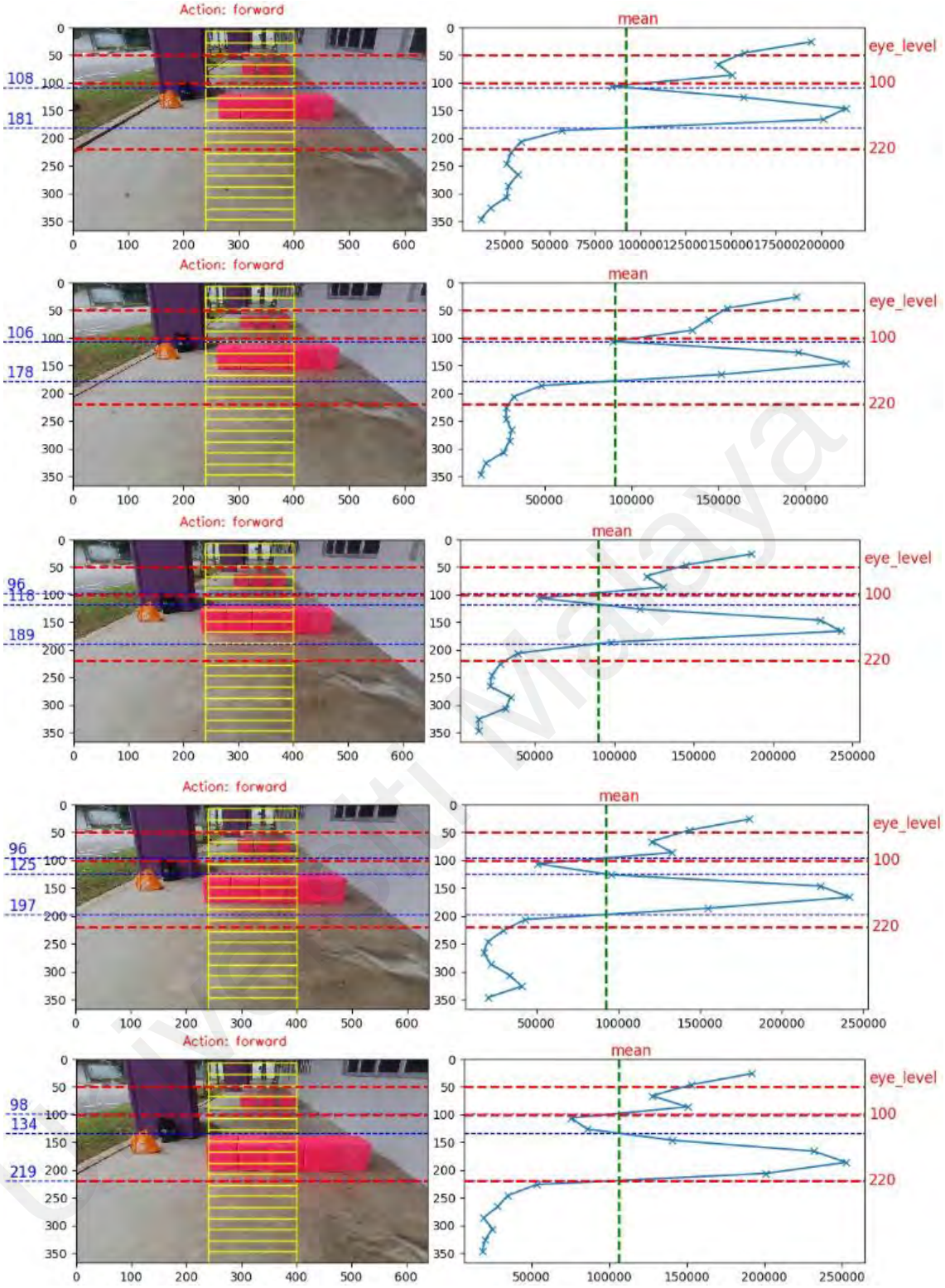


Figure 4.22: The first five steps showing obstacle lines passing only the lower MBS boundary. The drone moves forward several meters and stops before the first set of obstacles.

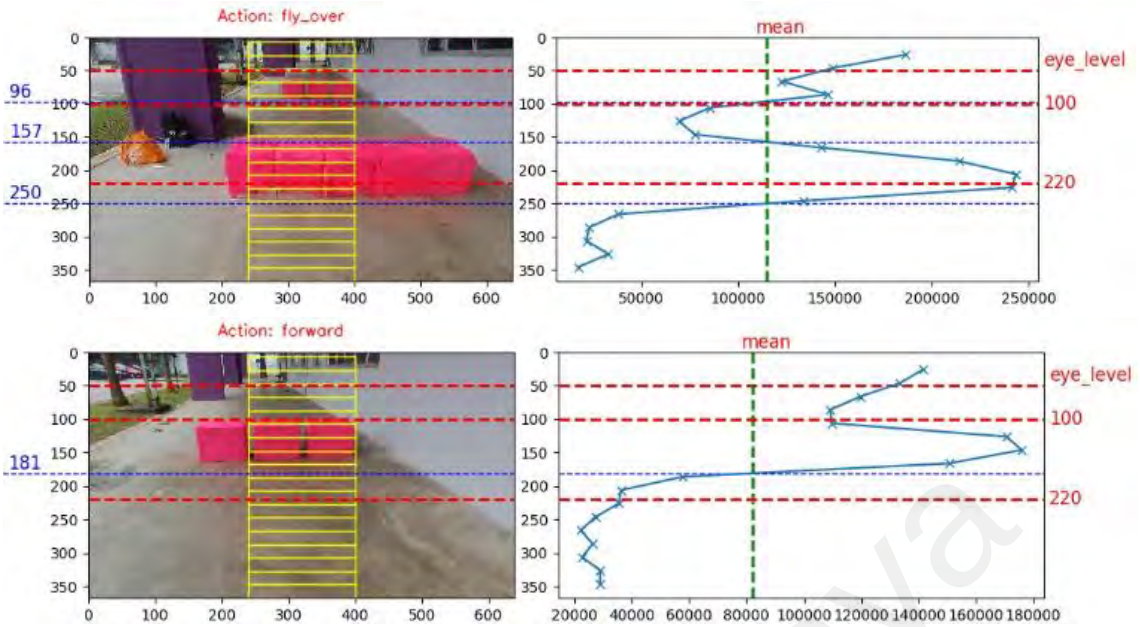


Figure 4.23: Top image has obstacle line 250 failing the lower MBS boundary, but lines 96 and 157 are between the drone's eye-level and the lower MBS boundary. The drone flies over the first set of obstacles. The bottom image is the drone observing a single obstacle line at 181, passing the lower MBS boundary.

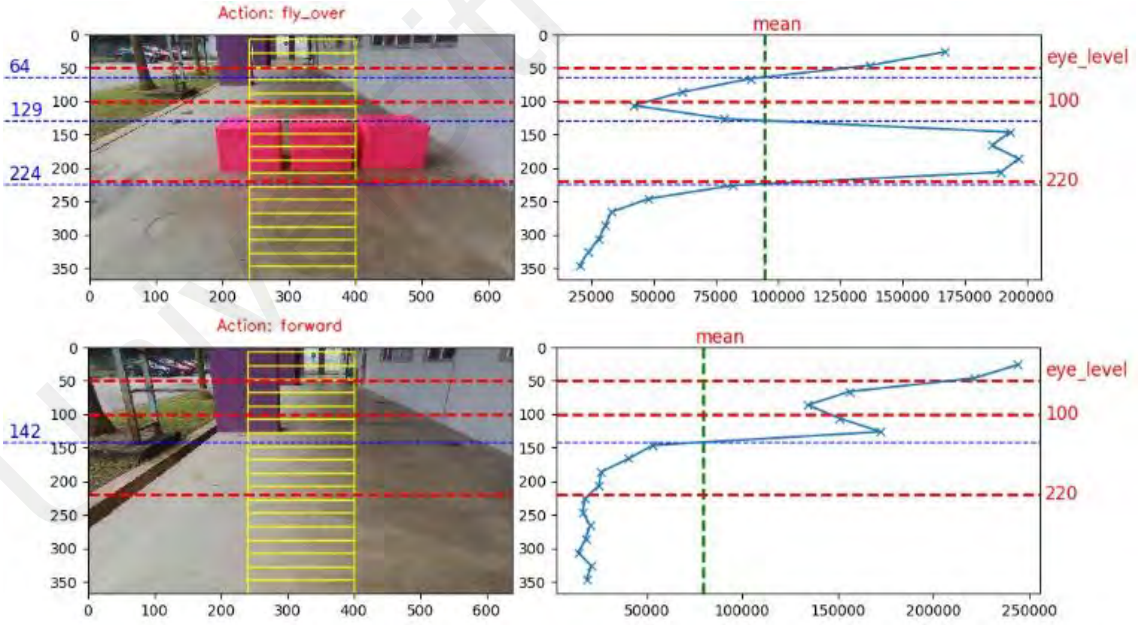


Figure 4.24: The top image has an obstacle line failing the lower MBS boundary. The drone can fly over the second set of obstacles because the other obstacles, lines 64 and 129, are between the eye-level and the lower MBS boundary. The bottom image shows the drone crossing another lower MBS boundary.

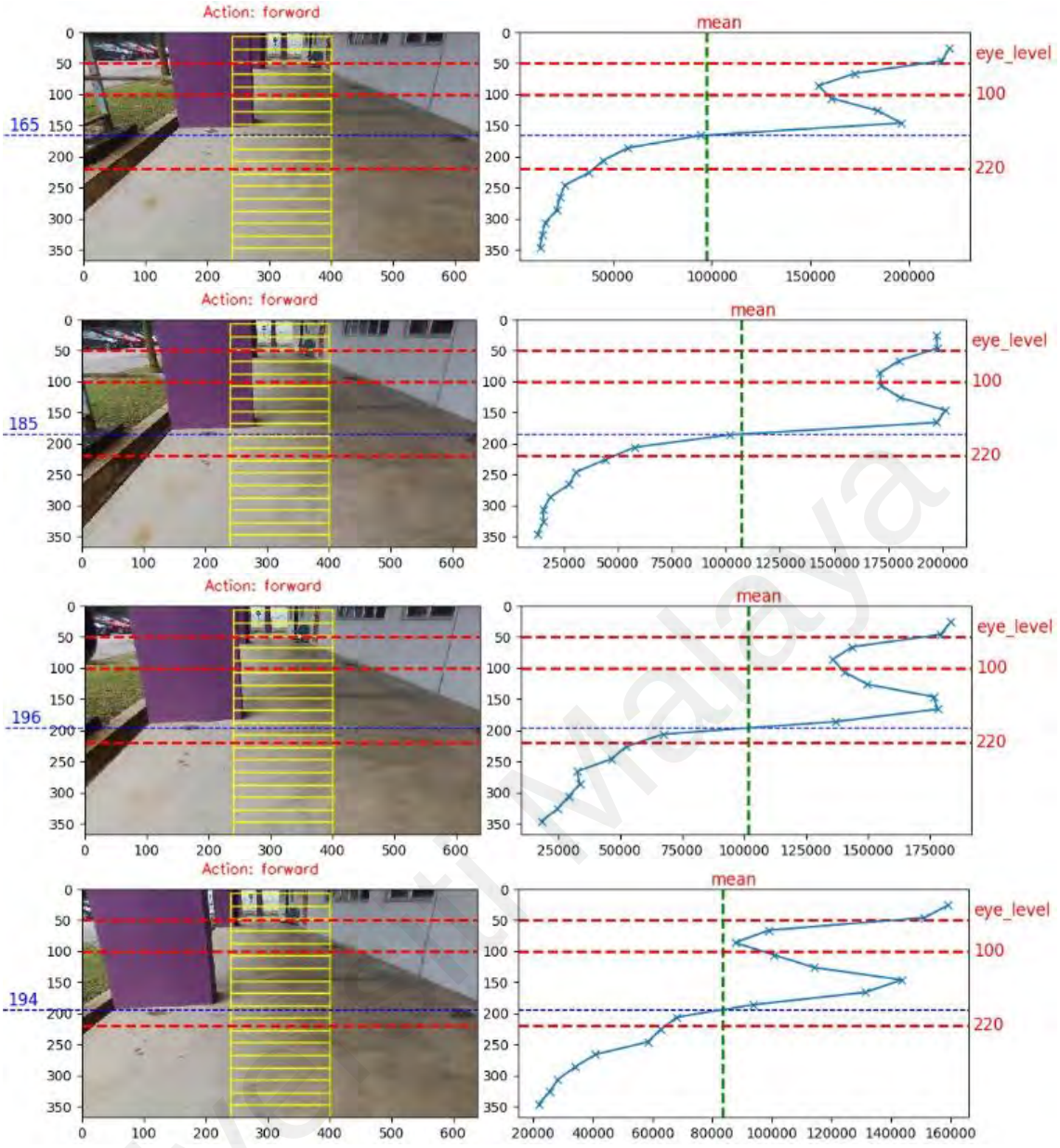


Figure 4.25: The drone moving forward several steps. Some obstacle lines are observed, and they pass the lower MBS boundary.

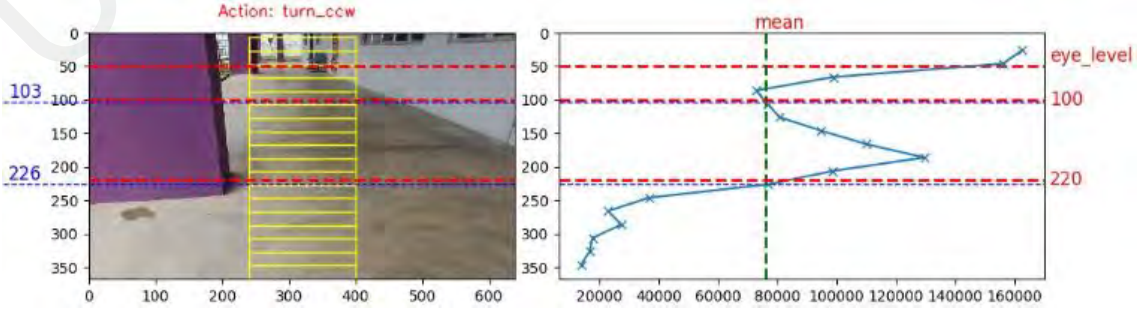


Figure 4.26: Obstacle line 103 passing the upper MBS boundary, but the obstacle line 226 failing the lower MBS boundary forcing the drone to escape the space by turning away.

4.7 Experiment 6 – testing with various types of surfaces

The new segmentation process for the MBS is tested to segment various floors and walls in an indoor environment. The aim here is to observe the performance of the segmentation method on various surfaces and conditions.

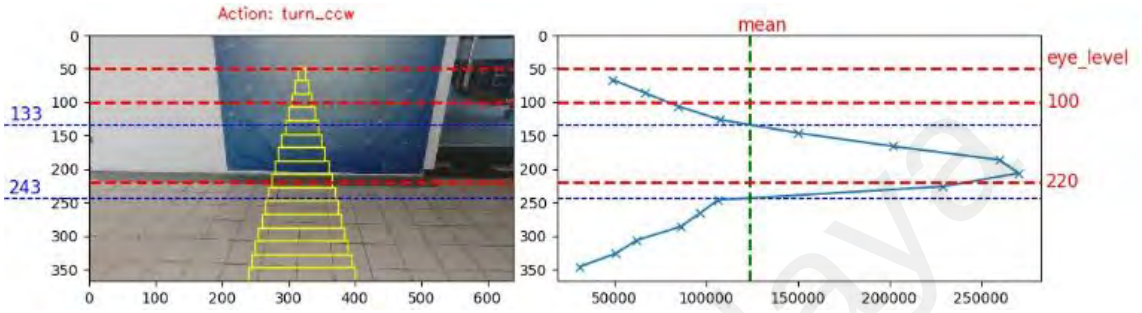


Figure 4.27: In front of a wall with a blue poster, the obstacle line 133 fails the upper MBS boundary, while 243 fails the lower MBS boundary, indicating tight spaces. The drone is recommended to escape the space by turning away.

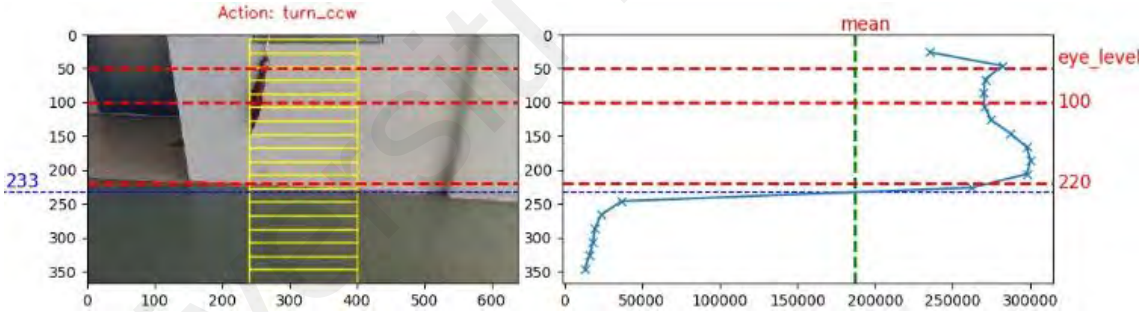


Figure 4.28: Facing a wall nearby shows an obstacle line 233, failing the lower MBS boundary and unsafe to move forward. Escaping the space is suggested.

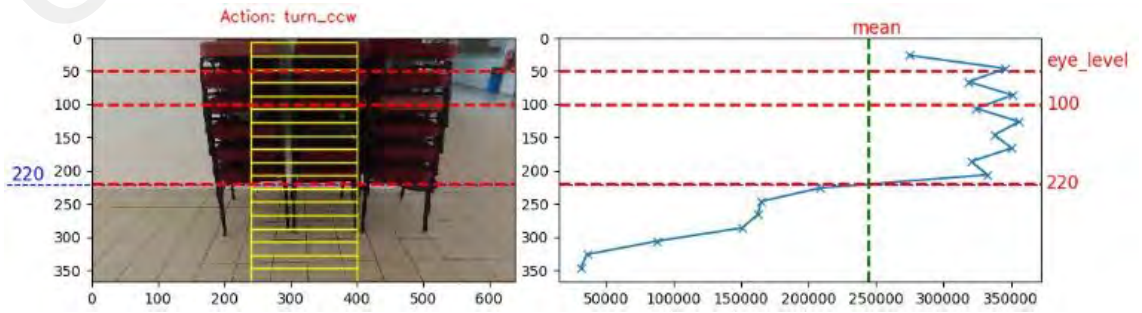


Figure 4.29: The drone in front of a stack of chairs showing an obstacle line generated at 220, which fails the lower MBS boundary. The drone has to turn away to escape the space.

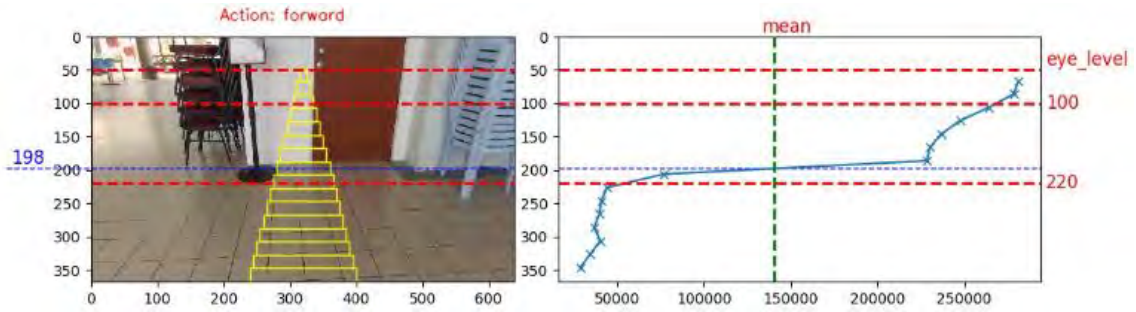


Figure 4.30: Facing a wooden door saw the drone getting one obstacle line at 198 and allowed to move forward.

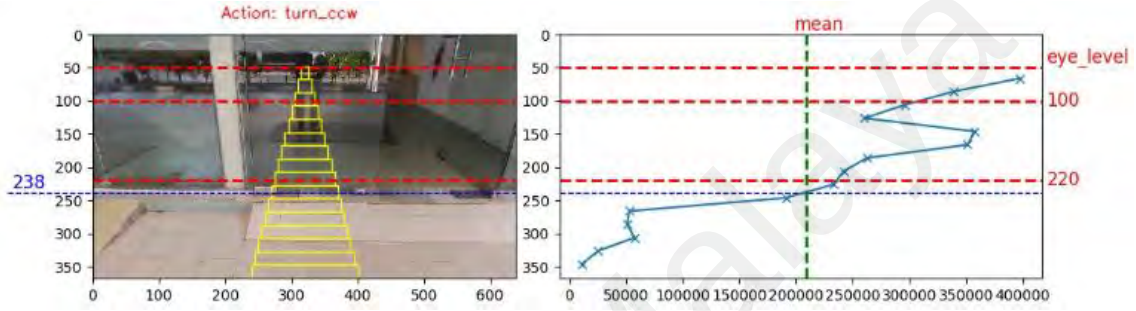


Figure 4.31: The drone is instructed to turn away when facing a glass door because the obstacle line 238 fails any MBS boundaries.

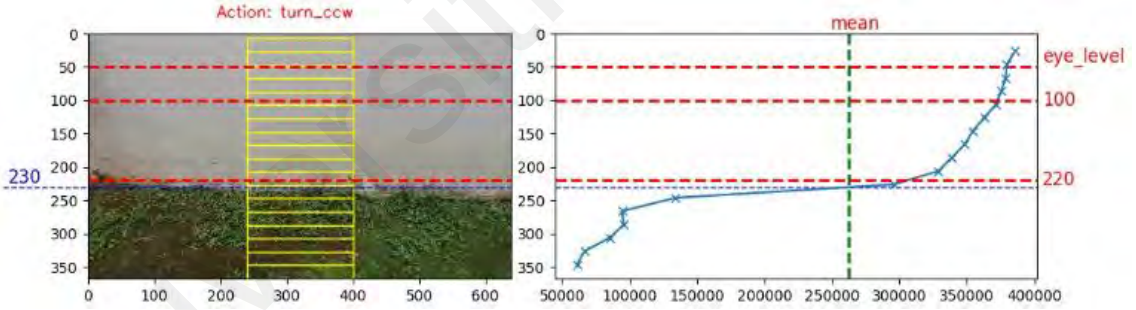


Figure 4.32: In the garden in front of the wall has the robot is instructed to move away.

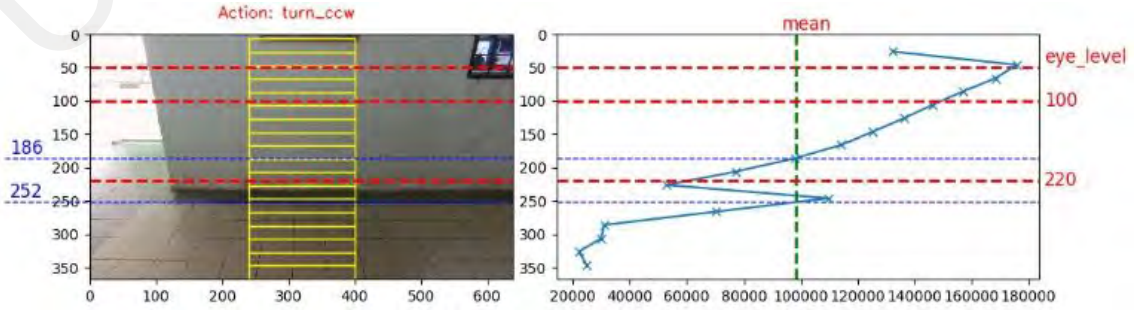


Figure 4.33: Another wall where obstacle lines 252 failing the lower MBS boundary. The drone has to escape by turning away.

4.8 Discussion

Six experiments showing the mono-visual drone performing autonomous navigation has been discussed through a series of diagrammatic representation. The experiments are:

Experiment 1 – drone navigation in an indoor environment

Experiment 2 – drone avoiding obstacles in an outdoor environment

Experiment 3 – drone handling obstacle dynamics in an outdoor environment

Experiment 4 – drone avoiding multiple ground obstacles in an outdoor environment

Experiment 5 – drone avoiding multiple ground obstacles in semi-outdoor environment

Experiment 6 – testing various types of surfaces

In the experiments, the MBS method has shown good performance in segmenting spaces, computing the MBS boundaries, computing obstacles positions, estimating opening and tightening of space, and path planning. Central to the decision making is the lower and upper MBS boundaries. Obstacle positions are evaluated based on the MBS boundaries. Some observations from the experiments include:

- Observation 1: an obstacle line cannot obstruct the drone's line of sight to any MBS boundaries; less indicates tightening of space. The drone should escape such a space by turning away.
- Observation 2: an opening of space is indicated when all obstacle lines are further than the upper MBS boundary. The drone can select the upper MBS boundary as part of path planning.
- Observation 3: light variations increases the number of intersection or obstacle lines but reduces the disparity between the pixel difference.
- Observation 4: real obstacles usually are indicated by a smaller number of the intersection but a high jump in the pixel difference.

- Observation 5: ground obstacles can be defined by nearness to the drone (bottom obstacle line failing the lower MBS boundary) and height between the lower MBS boundary and the drone's eye level. The drone can fly over the ground obstacles.
- Observation 6: tall obstacles can be defined by nearness to the drone (bottom obstacle line failing the lower MBS boundary) and height of obstacles surpassing the upper MBS boundary.
- Observation 7: the new MBS segmentation process can work on surfaces such as tiles, wooden flooring, tar roads, cemented flooring, walls, and, to some extent, glasses doors.
- Observation 8: the MBS method performs consistently in indoor, outdoor, and semi-outdoor environments.

The MBS method is meant as a cognitive approach to finding waypoints for a flying drone, without the traditional requirement to process and recognize objects in the environment. The qualitative aspect of the MBS approach is evidence when human reasoning is used as the benchmark – a human always has a sense of safeness when navigating the immediate space. A human can also project the next waypoint by sensing empty space ahead. What is that safe boundedness to a human? Why are humans so successful at navigation, and able to perform it without precise information? The MBS method answers these questions by showing how a computational model for robots can imitate the safe boundedness to a human.

4.9 Chapter Summary

This chapter presented six experiments that show the drone's performance in indoor, outdoor, and semi-outdoor environments including against obstacle dynamics such as flying-over maneuver. Additional experiments were carried out to test the MBS performance when facing various types of surfaces. The next chapter concludes the dissertation.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The dissertation asks the following research questions – how can one implement the MBS method on a mono-visual drone? How can one segment spaces from a mono-visual feed? How does a mono-visual drone perform using the MBS navigation method? In answering these research questions, this work has uncovered a new implementation of the MBS method on a mono-visual drone and contributed a new segmentation process so the MBS can handle mono-visual feed from a drone.

The MBS methods have been successful on two other robotics platforms, a depth perceiving laser robot that uses the MBS to find occlusions and openings in the space, and a mobile robot called Rovio, which showcased the first implementation of the MBS method on mono-visual feed without depth information. In this work, the MBS method is extended to the outdoor environment. Four experiments have been carried out outdoor where the drone is shown to perform obstacle avoidance, handling obstacle dynamics by avoiding a combination of ground and tall obstacles and navigating in a semi-outdoor environment. The semi-outdoor environment poses an exciting challenge. The contrasting brightness of the sunlight compared to the shades under the sunroof heavily influences the segmentation of the floor. For completion, two experiments were performed indoor – testing the drone navigation inside an open foyer and testing the segmentation on different surfaces.

Central to the development of the new segmentation process for the MBS method is the realization that for a drone, one needs to have two MBS boundaries, a lower boundary, so when the space is tight, the drone can choose to fly in small distances. Another is an upper boundary, useful when there is an opening of the space so that the drone can fly

more confidently. In the outdoor environment, often the drone path planning includes crossing the upper MBS boundary. In areas where the light condition varies greatly, the lower MBS boundary becomes handy. The lower MBS boundary keeps the drone moving and exploring the space, although in small intervals. More significantly, for the first time, obstacles are deduced not from tracking features or recognizing objects in the environment, but from analyzing pixel difference.

5.2 Future work

At the heart of this work is a combination of spatial reasoning with image processing. Learning about opening and tightening of spaces through the lens of a camera could lead to revealing new models for robot navigation. For example, the previous MBS method has already supported a space-based approach to robot navigation and mapping, such as the ASR computational theory. For the new MBS segmentation, such as the one proposed in this dissertation, it would be interesting to explore the following further:

- Exploring the filter boxes; rectangular, triangle, and the inverse triangle, with different shape, width, and height.
- Extend the segmentation area to the left and right side of the drone, and not just the front like the current work.
- Flying the drone at different heights in the outdoor environment.
- Testing the drone in a corridor-like environment to see if it can escape it
- Combine the MBS with object recognition and tracking, and create a new hybrid approach for robot navigation.

REFERENCES

Aguilar, W. G., Casaliglla, V. P., & Polit, J. L. (2017). Obstacle avoidance based-visual navigation for micro aerial vehicles. *Electronics*, *6*(1), 10.

Artieda, J., Sebastian, J. M., Campoy, P., Correa, J. F., Mondragón, I. F., Martínez, C., & Olivares, M. (2009). Visual 3-d slam from UAVs. *Journal of Intelligent and Robotic Systems*, *55*(4-5), 299.

Azizul Hasan, Z. H. (2013). *Robot mapping without a precise map* (Doctoral dissertation, Auckland University of Technology).

Azizul, Z. & Khanil, A.(2017). How Rovio navigates in its environment. In *AUN/SEED-Net Regional Conference on Computer and Information Engineering (pp. 67-70).*

Azizul, Z., & Yeap, W. (2015). Autonomous robot mapping by landmark association. In EAP Joint Conference on Cognitive Science. CEUR-WS. org.

Bacik, J., Durovsky, F., Fedor, P., & Perdukova, D. (2017). Autonomous flying with quadrocopter using fuzzy control and ArUco markers. *Intelligent Service Robotics*, *10*(3), 185-194.

Basiri, A., Amirian, P., & Winstanley, A. (2014). The use of quick response (QR) codes in landmark-based pedestrian navigation. *International Journal of Navigation and Observation*, *2014*.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.

Brown, M., Szeliski, R., & Winder, S. (2005, June). Multi-image matching using multi-scale oriented patches. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 510-517). IEEE.

Brzozowski, B., Kaźmierczak, K., Rochala, Z., Wojda, M., & Wojtowicz, K. (2016, June). A concept of UAV indoor navigation system based on magnetic field measurements. In *2016 IEEE Metrology for Aerospace (MetroAeroSpace)* (pp. 636-640). IEEE.

Chrastil, E. R., & Warren, W. H. (2014). From cognitive maps to cognitive graphs. *PloS one*, *9*(11), e112544.

Ciancia, F. (1991). Tolman and Honzik (1930) revisited or the mazes of psychology (1930-1980). *The Psychological Record*, *41*(4), 461.

Colombo, D., Serino, S., Tuena, C., Pedroli, E., Dakanalis, A., Cipresso, P., & Riva, G. (2017). Egocentric and allocentric spatial reference frames in aging: A systematic review. *Neuroscience & Biobehavioral Reviews*, *80*, 605-621.

Duckham, M., Winter, S., & Robinson, M. (2010). Including landmarks in routing instructions. *Journal of Location Based Services*, *4*(1), 28-52.

Forouher, D., Besselmann, M. G., & Maehle, E. (2016, November). Sensor fusion of depth camera and ultrasound data for obstacle detection and robot navigation. In *2016 14th international conference on control, automation, robotics and vision (ICARCV)* (pp. 1-6). IEEE.

Fuentes-Pacheco, J., Ruiz-Ascencio, J., & Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, *43*(1), 55-81.

Golledge, R. G. (1999). Human wayfinding and cognitive maps. *Wayfinding behavior: Cognitive mapping and other spatial processes*, 5-45.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., & Malik, J. (2017). Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2616-2625).

Harris, C. G., & Stephens, M. (1988, August). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, No. 50, pp. 10-5244).

Huang, W. T., Tsai, C. L., & Lin, H. Y. (2012, July). Mobile robot localization using ceiling landmarks and images captured from an RGB-D camera. In *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 855-860). IEEE.

Irwin, L. N., & Irwin, B. A. (2020). Place and environment in the ongoing evolution of cognitive neuroscience. *Journal of Cognitive Neuroscience*, *32*(10), 1837-1850.

Kaplan, R., & Friston, K. J. (2018). Planning and navigation as active inference. *Biological cybernetics*, *112*(4), 323-343.

Ke, Y., & Sukthankar, R. (2004, June). PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* (Vol. 2, pp. II-II). IEEE.

Kim, J., & Chung, W. (2016). Localization of a mobile robot using a laser range finder in a glass-walled environment. *IEEE Transactions on Industrial Electronics*, *63*(6), 3616-3627.

Kouris, A., & Bouganis, C. S. (2018, October). Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1-9). IEEE.

Lee, J. O., Lee, K. H., Park, S. H., Im, S. G., & Park, J. (2011). Obstacle avoidance for small UAVs using monocular vision. *Aircraft Engineering and Aerospace Technology*.

Lindeberg, T. (2015). Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, *52*(1), 3-36.

Löhr, G. (2019). Embodied cognition and abstract concepts: Do concept empiricists leave anything out?. *Philosophical Psychology*, *32*(2), 161-185.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

Mackintosh, N. J. (2002). Do not ask whether they have a cognitive map, but how they find their way about. *Psicológica*, *23*(1).

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, *22*(10), 761-767.

McCunn, L. J., & Gifford, R. (2018). Spatial navigation and place imageability in sense of place. *Cities*, *74*, 208-218.

Mei, H., Yang, X., Wang, Y., Liu, Y., He, S., Zhang, Q., ... & Lau, R. W. (2020). Don't hit me! glass detection in real-world scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3687-3696).

Meilinger, T., & Vosgerau, G. (2010, August). Putting egocentric and allocentric into perspective. In *International Conference on Spatial Cognition* (pp. 207-221). Springer, Berlin, Heidelberg.

Mikolajczyk, K., & Schmid, C. (2002, May). An affine invariant interest point detector. In *European conference on computer vision* (pp. 128-142). Springer, Berlin, Heidelberg.

Nazareth, A., Weisberg, S. M., Margulis, K., & Newcombe, N. S. (2018). Charting the development of cognitive mapping. *Journal of experimental child psychology*, *170*, 86-106.

Newcombe, N. S. (2018). Three kinds of spatial cognition. *Stevens' handbook of experimental psychology and cognitive neuroscience*, *3*, 1-31.

Quesnot, T., & Roche, S. (2015). Measure of landmark semantic salience through geosocial data streams. *ISPRS International Journal of Geo-Information*, *4*(1), 1-31.

Rambabu, R., Bahiki, M. R., & Azrad, S. (2015). Multi-sensor fusion-based UAV collision avoidance system. *Jurnal Teknologi*, *76*(8).

Rosten, E., & Drummond, T. (2006, May). Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430-443). Springer, Berlin, Heidelberg.

Ruggiero, G., D'Errico, O., & Iachini, T. (2016). Development of egocentric and allocentric spatial representations from childhood to elderly age. *Psychological Research*, *80*(2), 259-272.

Sani, M. F., & Karimian, G. (2017, November). Automatic navigation and landing of an indoor AR drone quadrotor using ArUco marker and inertial sensors. In *2017 International Conference on Computer and Drone Applications (IConDA)* (pp. 102-107). IEEE.

Schmidt, J., Wong, C. K., & Yeap, W. K. (2006, August). A Split & Merge Approach to Metric-Topological Map-Building. In *18th International Conference on Pattern Recognition (ICPR'06)* (Vol. 3, pp. 1069-1072). IEEE.

Sharmin, N., & Brad, R. (2012). Optimal filter estimation for Lucas-Kanade optical flow. *Sensors*, *12*(9), 12694-12709.

Srinivasan, M. V. (2015). Where paths meet and cross: navigation by path integration in the desert ant and the honeybee. *Journal of Comparative Physiology A*, *201*(6), 533-546.

Starrett, M. J., & Ekstrom, A. D. (2018). Perspective: Assessing the flexible acquisition, integration, and deployment of human spatial representations and information. *Frontiers in human neuroscience*, *12*, 281.

Tarazona, R. D. F., Lopera, F. R., & Sánchez, G. D. G. (2014, September). Anti-collision system for navigation inside an UAV using fuzzy controllers and range sensors. In *2014 XIX Symposium on Image, Signal Processing and Artificial Vision* (pp. 1-5). IEEE.

Wanga, X., Wangb, J., & Studio, S. R. (2015). Realtime glass detection with laser rangefinders for robot navigation and mapping. In *The 9th International Symposium on Mobile Mapping Technology*.

Wang, C., Chen, X., & Knierim, J. J. (2020). Egocentric and allocentric representations of space in the rodent brain. *Current Opinion in Neurobiology*, *60*, 12-20.

Wang, Z., & Yang, X. (2018, June). Moving target detection and tracking based on pyramid Lucas-Kanade optical flow. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)* (pp. 66-69). IEEE.

Wei, H., Li, X. E., Shi, Y., You, B., & Xu, Y. (2018, August). Multi-sensor fusion glass detection for robot navigation and mapping. In *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)* (pp. 184-188). IEEE.

Wong, C. K., Schmidt, J., & Yeap, W. K. (2007). Using a mobile robot for cognitive mapping. *International Joint Conferences on Artificial Intelligence.*

Yeap, W. K., & Jefferies, M. E. (1999). Computing a representation of the local environment. *Artificial Intelligence*, *107*(2), 265-301.

Zhang, L., & Mou, W. (2017). Piloting systems reset path integration systems during position estimation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *43*(3), 472.

Zheng, W., Xiao, J., & Xin, T. (2017, June). Integrated navigation system with monocular vision and LIDAR for indoor UAVs. In *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 924-929). IEEE.