

**NEAR-RANGE WATER BODY DETECTION AND OBSTACLE
DETECTION IN RAINFOREST TERRAIN/ TROPICAL TERRAIN**

By

TEOH CHEE WAY

A thesis submitted to the Department of Mechatronics and BioMedical
Engineering,
Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Engineering Science
May 2011

ABSTRACT

NEAR-RANGE WATER BODY DETECTION AND OBSTACLE DETECTION FOR AUTONOMOUS VEHICLE IN RAINFOREST/TROPICAL TERRAIN

Teoh Chee Way

The needs to use autonomous vehicle for rainforest terrain is increasing day by day. Many projects were launched with the aim of developing a fully autonomous vehicle that is able to maneuver from a location to a location without colliding with obstacle. However, the applications of these vehicles are limited due to limitation of sensors used and the knowledge on how to make use of the information provided by the sensors. Vision sensors are commonly used in vehicle guidance as they provide more information compared to other type of sensors. The interpretation of the visual information is very important, therefore it is necessary to carry out research to make use of the information for terrain classification and obstacle detection.

In this thesis, disparity and color feature from stereo camera have been studied closely to provide solution for visual guidance in rainforest terrain. New methodologies have been proposed utilizing polarizing effect on the stereo camera to solve prevalent challenges that appear in rainforest terrain. Particularly, the proposed methodologies are used to solve ground truth determination, tree trunks detection and water body detection.

The V-disparity image is used together with color information to determine the ground region that may be travelled by the autonomous vehicle. The main contribution of this ground detection module is that it is very robust to vehicle orientation and tilt. In addition, the proposed of using K-means color clustering method in ground profile determination is effective even when there is no distinct feature that may represent the ground region.

Using the proposed ground plane detection algorithm, two task-specific modules are introduced to deal with tree trunks and water body that may present in the rainforest scene. Tree trunks and water body are typical hazards that can be found in rainforest terrain. Sobel edge detection and U-disparity image is used to extract the tree trunks from the scene with the assumption that the tree trunks are vertical or near vertical. This method can detect most of the tree trunks present as tree trunks in rainforest terrain are usually vertical and tall. The proposed U-disparity scheme shows effective results for near vertical tree trunks obstacles.

The water body detection module utilized multiple features in texture, disparity and partial polarization. Stereo camera with different polarization angles are applied on each side of the stereo camera. Based on experiments conducted, it is shown that water body has significant changes in brightness and disparity when different polarization angles are applied. Based on the difference and coupled with low-texture characteristic and sky reflection detection, most of the water region can be detected using the proposed polarization technique.

AKNOWLEDGMENTS

This thesis would never have been completed if it were not for the amazing support, encouragement and kindest of so many people.

First, I would like to begin by thanking my supervisors, Assistant Professor Dr Tan Ching Seong and Assistant Professor Dr Tan Yong Chai. I sincerely appreciate them for giving me continuous guidance, constructive discussions, encouragement and endless patience. I would like to thank both of them for their useful comments and positive criticisms on the completion of this thesis.

I would like to acknowledge my fellow research mates, Chan Kim Chon and Tee Yu Hon for their help and assistance in various occasions. Special thanks to Christine Gee who have offered me a hand in many ways while I was pursuing my research.

I am grateful to UTAR for providing me facilities and financial assistance to carry out my research work. I also wish to thank workplace mates for the great environment and joyful atmosphere while carrying out this research.

Finally, I would like express my heartfelt gratitude to dear family for their continuous encouragement, understanding and love. I would like to dedicate this work to them.

APPROVAL SHEET

This thesis entitled “**NEAR-RANGE WATER BODY DETECTION AND OBSTACLE DETECTION FOR AUTONOMOUS VEHICLE IN RAINFOREST TERRAIN**” was prepared by TEOH CHEE WAY and submitted as partial fulfillment of the requirements for the degree of Master of Engineering Science at Universiti Tunku Abdul Rahman.

Approved by:

(Dr. Tan Ching Seong)
Date: 28 May 2011
Assistant Professor/Supervisor
Department of Mechatronics and BioMedical Engineering
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

(Dr. Tan Yong Chai)
Date: 28 May 2011
Assistant Professor/Co-supervisor
Department of Mechanical and Material Engineering
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

**FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 28 May 2011

PERMISSION SHEET

It is hereby certified that **TEOH CHEE WAY** (ID No: **08UEM08100**) has completed this thesis entitled “NEAR-RANGE WATER BODY DETECTION AND OBSTACLE DETECTION FOR AUTONOMOUS VEHICLE IN RAINFOREST TERRAIN” under the supervision of Dr Tan Ching Seong (Supervisor) from the Department of Mechatronics and BioMedical Engineering, Faculty of Engineering and Science, and Dr Tan Yong Chai (Co-Supervisor) from the Department of Mechanical and Material Engineering, Faculty of Engineering and Science.

I hereby give permission to my supervisors to write and prepare a manuscript of these research findings for publishing in any form, if I did not prepare it within six (6) months time from this date, provided, that my name is included as one of the authors for this article. Arrangement of names will depend on my supervisors.

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name : Teoh Chee Way

Date : 05 May 2011

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGMENTS	iv
APPROVAL SHEET	v
PERMISSION SHEET	vi
DECLARATION	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
CHAPTER 1	1
1.1 Research Motivation	1
1.2 Scope of Research	6
1.3 Thesis Outline	9
CHAPTER 2	12
2.1 Overview	12
2.2 Review of Visual Guidance for Outdoor Terrain: Structured	13
2.2.1 Methods based on NAVLAB and ALVINN	13
2.2.2 Methods based on ARGO	15
2.3 Review of Visual Guidance for Outdoor Terrain: Unstructured	16
2.3.1 Methods by DEMO III	18
2.3.2 Methods by Darpa PerceptOR	20
2.3.3 Methods by LAGR	21
2.3.4 Methods by Manduchi et al (2005)	22
2.4 Challenges in Rainforest Terrain	23
2.5 Summary	26
CHAPTER 3	30
3.1 Overview	30
3.2 Visual Guidance System	31
3.2.1 Proposed System Architecture	33
3.3 Sensor Description	36
3.3.1 Bumblebee 2 Stereo Vision System	36
3.3.2 Triclops Stereo Vision SDK (PGR Software Development Kit)	39
3.3.3 Intel OpenCV Library	40
3.3.4 Hardware Setup	40
3.4 Overview of Stereo Vision Principles	42
3.4.1 Visual Depth using Bumblebee 2 Stereo Vision System	46
3.5 Summary	50
CHAPTER 4	52
4.1 Introduction	52
4.2 Overview of Ground Plane Detection	53

4.2.1	Methods using V-Disparity by Labayrade et al (2002)	54
4.3	Feature Extraction	60
4.3.1	Analysis on Disparity Image and V-Disparity Image of Rainforest Terrain.	60
4.4	Color Feature from Stereo Imaging	62
4.4.1	Colour Distribution of Rainforest Scene	65
4.4.2	K-means Clustering	67
4.4.3	K-means Clustering: Number of Cluster Determination	69
4.5	Developed Ground Plane Detection Algorithm	71
4.6	Experimental Results and Discussions	76
4.6.1	Simple Unstructured Terrain Experiment	78
4.6.2	Moderate Terrain Experiment	81
4.6.3	Complex Terrain Experiment	85
4.7	Summary	90
CHAPTER 5		94
5.1	Introduction	94
5.2	Overview of Obstacle Detection	95
5.2.1	Method by Huertas et al (2005)	97
5.3	Scene Consideration	98
5.4	Feature Extraction	99
5.4.1	U-disparity Image	99
5.4.2	Sobel Edge Detector	102
5.5	Tree Trunk Detection Algorithm	104
5.6	Experimental Results and Discussions	106
5.7	Summary	111
CHAPTER 6		113
6.1	Introduction	113
6.2	Overview of Water Body Detection	114
6.2.1	Method by The Jet Propulsion Laboratory (JPL)	114
6.2.2	Method using Polarization-Based Camera	117
6.3	Mathematical Description for Appearance of Water Body	119
6.3.1	Water Reflectance Model	120
6.3.2	Partial Linear Polarization	125
6.4	Feature Extraction	127
6.4.1	Water Cue from Partial Polarization Feature	129
6.4.2	Water Cue from Stereo Disparity Feature	131
6.4.3	Water Cue from Texture Feature	134
6.5	Algorithm: Fusing Water Cue	135
6.5.1	Sky Reflection Detection Module	136
6.5.2	Low-Texture Detection Module	138
6.5.3	Object Reflection and Invalid Disparity Pixel Detection Module.	139
6.6	Experimental Results and Discussions	140
6.6.1	Running Water Detection	142
6.6.2	Standing Water Body Detection	146
6.6.3	Back-Lighting Error	148
6.7	Summary	150
CHAPTER 7		153

7.1	Contributions	153
7.2	Conclusion Remarks	156
7.3	Future works	157
	Author's Publication	161
	References	162
	Appendices	166
	Appendix A	167
	Appendix B	171

LIST OF TABLES

Table 2.1:	Summary of visual guidance system approaches in indoor and semi-structured outdoor terrain.	28
Table 2.2:	Summary of visual guidance system approaches in unstructured outdoor terrain.	29
Table 3.1:	Bumblebee 2 stereo camera specifications.	38
Table 3.2:	Stereo correspondence approach. (Brown, Burschka, & Hager, 2003).	44
Table 4.1:	The number of regions resulted from the number of cluster settings. The percentages of the regions are based on 30 samples.	70
Table 6.1:	Average partial polarization for different objects and terrain (average of 20 samples).	130
Table 6.2:	Average relative smoothness for 20 samples for water bodies and other objects in rainforest scene.	139

LIST OF FIGURES

Figure 2.1:	Notional Approach to Top Level Systematics of Ground Mobile Robot Systems (Eicker, 2001).	17
Figure 3.1:	General modules of visual guidance system (VGS).	32
Figure 3.2:	Architecture of the visual guidance system.	34
Figure 3.3:	Bumblebee 2 stereo vision system.	37
Figure 3.4:	Bumblebee 2 stereo vision system and polarizer assembly mounted in front.	41
Figure 3.5:	Experiment setup with the pitch angle set at -15° from the negative x-axis.	42
Figure 3.6:	Frontal parallel arrangement of two cameras. (Bradski & Kaehler, 2008).	45
Figure 3.7:	Triclops general computational stereo process (Point Grey Research Inc, 2003).	47
Figure 3.8:	Sample stereo image pair and disparity image. (a) Left image. (b) Right Image. (c) Disparity Image.	49
Figure 4.1:	Sample stereo image and its V-disparity image frame of reference (Labayrade, Aubert, & Tarel, 2002). (a) Stereo image (left). (b) Stereo image (right). (c) V-disparity image. (d) V-disparity image with ground correlation line (blue line) and obstacle profile (white line).	55
Figure 4.2:	Stereo image pair, disparity image and V-disparity image. (a) Left image. (b) Right Image. (c) Disparity Image. (d) V-disparity image.	56
Figure 4.3:	Ground correlation lines in V-disparity image. The solid slanted line corresponds to the ground correlation line	

	obtained using the static calibration data, while the dashed slanted lines are the ones expected varying the pitch value. The solid vertical line indicates the 0 disparity value, the dashed vertical line indicates the disparity value of points at infinite distance; they do not overlap due to a slight convergence of cameras optical axes (Labayrade & Aubert, 2003).	58
Figure 4.4:	Sample stereo images, disparity images and its V-disparity images frame of reference in rainforest terrain. Note: the ground plane profile and obstacle profiles are hand-labeled.	62
Figure 4.5:	2-dimensional hue-saturation histogram. The distribution of object colors (ground, tree trunk, vegetation and others) are illustrated. (a) The scene. (b) 2-dimensional histogram.	66
Figure 4.6:	Framework of the developed ground plane detection module.	72
Figure 4.7:	(a) Image of the Scene. (b) V-disparity image. (c) Candidates for ground correlation line. (d) Extracted ground correlation line.	74
Figure 4.8:	Sample result of ground plane estimation. (a) Sample scene. (b) Mapped ground pixels in red. (c) Ground correlation profile in V-disparity image.	75
Figure 4.9:	The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.	77
Figure 4.10:	Sample results of detected ground plane in rainforest terrain for simple unstructured terrain. (a)(i) and (b)(i) show the sample images and the detected ground planes are shown in (a)(ii) and (b)(ii) respectively.	79

Figure 4.11: Average ground plane error versus distance for ground plane detection in simple unstructured terrain. The number of samples used is thirty.	80
Figure 4.12: Performance matrix of the ground plane detection module in simple unstructured terrain. The number of samples used is thirty (30).	81
Figure 4.13: Sample results of detected ground plane in rainforest terrain for rainforest terrain.	82
Figure 4.14: Average ground plane error versus distance in rainforest terrain. The number of samples used is thirty (30).	83
Figure 4.15: Performance matrix of the ground plane detection module for terrain with moderate complexity. The number of samples used is thirty (30).	84
Figure 4.16: Sample results of detected ground plane in rainforest terrain for complex terrain.	86
Figure 4.17: Average ground plane error versus distance in complex terrain. The number of samples used is thirty (30).	88
Figure 4.18: Performance matrix of the ground plane detection module in complex terrain. The number of samples used is thirty (30).	89
Figure 4.19: Average ground plane detection error versus distance. The number of samples used is ninety (90).	91
Figure 4.20: Overall performance matrix of the ground plane detection module. The number of samples used is ninety (90).	92
Figure 5.1: Sample image of (a) scene with tree trunks, (b) Disparity data, (c) Corresponding U-disparity image.	100

Figure 5.2:	Disparity accumulation in U-disparity image as a cue to spot the tree trunks.	101
Figure 5.3:	A 3-by-3 kernel for a Sobel derivative; note that the anchor point is in the center of the kernel.	103
Figure 5.4:	Edge detected using Sobel Edge detector. Note that the vertical and near vertical lines are usually represent the tree trunks.	104
Figure 5.5:	Proposed tree trunk detection module using edges and U-disparity image.	105
Figure 5.6:	The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.	106
Figure 5.7:	Sample scene with tree trunks as obstacles and the detected tree trunks in magenta. Magenta and blue correspond to tree trunks and ground region respectively. Green region indicate region with no disparity information and blue color represent unclassified region which may be other types of obstacles.	108
Figure 5.8:	Performance matrix of tree trunk detection module based on samples used in this experiment.	109
Figure 5.9:	Performance matrix of tree trunk detection module based on distance from the stereo camera.	110
Figure 6.1:	Camera setup with polarizer place in front of the camera.	118
Figure 6.2:	Components of reflection and inter-reflection (Nayar, Fang, & Boulton, 1993).	121
Figure 6.3:	Theoretical fraction of incident power that is reflected from an air/pure water interface as a function of incidence angle. $R_{r, \perp}$ and $R_{r, \parallel}$ are the Fresnel reflection coefficients for	

	light polarized perpendicular to and parallel to the plane of incidence, respectively. R_r is the Fresnel reflection coefficient for unpolarized light (Rankin & Matthies, 2010).	124
Figure 6.4:	Principle of specular reflection (Xie, Xiang, Pan, & Liu, 2007).	125
Figure 6.5:	Sample image on application of polarizer to remove specular reflection. (a) Original image of a river scene. (b) Image after polarizer is applied.	130
Figure 6.6:	Sample disparity image with different polarization angle applied to the right side of the stereo camera. The gray color regions correspond to invalid pixels.	132
Figure 6.7:	Average Percentage of Invalid Pixels for Water vs Polarization Degree (20 samples).	133
Figure 6.8:	A sample image of 90o polarization degree and its corresponding line scan histogram.	134
Figure 6.9:	Framework for water bodies detection using multiple features.	136
Figure 6.10:	The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.	141
Figure 6.11:	Sample results of detected ground plane in rainforest terrain for simple unstructured terrain.	143
Figure 6.12:	Average water body detection error versus distance for running water. The number of samples used is thirty (30).	144
Figure 6.13:	Performance matrix of the ground plane detection module in running water detection. The number of samples used is thirty (30).	145

Figure 6.14: Average standing water body detection error versus distance. The number of samples used is thirty (30).	146
Figure 6.15: Performance matrix of the standing water detection module. The number of samples used is thirty (30).	147
Figure 6.16: Sample results of the detected standing water using proposed water body detection.	148
Figure 6.17: Overall performance matrix of the water detection module. The number of samples used is sixty (60).	151

CHAPTER 1

INTRODUCTION

1.1 Research Motivation

Autonomous vehicle has received wide attention and widespread relevance during the past few decades. Many projects were launched with the aim of developing a fully autonomous vehicle that is able to maneuver from a location to a location without colliding with obstacles. The potential applications of autonomous vehicle range from daily life to military. It enables wide-area environment monitoring, mining, disaster recovery, search-and-rescue activities and planetary exploration.

One of the earliest of autonomous vehicle reported is the conventional “look and move” where the autonomous vehicle look ahead and driving blindly for another short distance before taking another view (Nilsson, 1969). The autonomous robot was built to conduct experiments on the real-time control system that interact with indoor environment. The robot adopted the “plan and move” approach which it plans its task before it acts. It utilized several individual artificial programs as its abilities that are integrated as one system. There were three basic functions incorporated in this system which were problem-solving, modeling and perception. The respective functions are task analysis, learning and extracting features from the scene. Although this project

was limited to indoor task, it opened up extensive exploration of artificial intelligence and machine vision for vehicle guidance.

There are two main tasks for an autonomous vehicle to accomplish in order for successful navigation which are to avoid collisions with obstacles and to reach the destination point without any human assistance. The vehicle needs to determine the obstacles and uncertainties surrounding it by using information gathered by sensors. Thus, the autonomous navigation is very dependent on the information extracted from the sensors attached to the vehicle. There are two types of sensors that are used to gather information which are proprioceptive sensors and exteroceptive sensors (Adams, Wijesoma, & Shacklock, 2007). Proprioceptive sensors measure the internal state of the vehicle such as speed, acceleration and relative displacement by using motor encoders, IMUs, etc. The interaction between the vehicle and surrounding environments are made using exteroceptive sensors such as global positioning system (GPS), cameras, lasers, etc. Successful navigation is very dependent of the extraction of useful information from exteroceptive sensors, information of the pose and position of the vehicle and algorithms that can fuse both information from proprioceptive sensors and exteroceptive sensors (Adams, Wijesoma, & Shacklock, 2007). The challenges lie in the successful usage of the information provided by exteroceptive sensors and consequently, the autonomous vehicle research has focused on the visual sensing technology and robust sensor interpretation.

There are many types of visual sensors available that have been used in autonomous vehicle for various applications. Passive sensors such as color camera and stereo camera are general devices used to acquire data from the environment. It offers large amount of data that can be used in terrain classification and obstacle detection. In contrast, the active sensors involve illumination of the scene with radiation and the reflected radiations are collected and measured. In this project, the focus is mainly on color stereo camera as it is cheaper and more feasible to be implemented in various missions. Color stereo camera offers 2-dimensional (2D) and 3-dimensional (3D) information of the scene in non-invasive way. While active sensors are more powerful, it is not suitable for military context which active source can be easily detected by enemy. Active source also poses problems in common applications where there is more than one autonomous vehicle moving around as active source from other vehicle will be interference or noise (Broggi, Fascioli, & Bertozzi, 2000). In addition, heavy usage of active sensors could lead to unacceptable hazard and pollution (Broggi, Fascioli, & Bertozzi, 2000). Thus, the use of passive sensors is more practicable in most of the application.

Early researches and developments of autonomous vehicle were initiated by military organization (Shacklock, Xu, & Wang, 2006) as defense and surveillance are very much of their concern. Consequently, it opens up the interest application of autonomous vehicle in unstructured terrain (i.e. forest terrain, off-road terrain, etc) as the requirements for unstructured terrain navigation are very much challenging compared to structured and indoor navigation. To date, the unstructured terrain navigation is not limited to

military mission but also planetary exploration (Goldberg, Maimone, & Matthies, 2002) and search and rescue mission (Kamegawa, et al.). While natural disasters are very much a concern recently, autonomous vehicle can play major part in search and rescue mission as it can be deployed to search and provide initial help to the victims. Autonomous vehicle can be used to explore the disaster scene initially before the scene is declared safe for human admission. Similarly, the planetary exploration mission employs autonomous robot to perform surveillance and initial experiment as well. Therefore, there is a need of robust guidance system in unstructured terrain as it is the key enabling technology for unstructured terrain navigation.

In order for autonomous navigation in unstructured terrain to be feasible, there are many issues need to be resolved. In this thesis, the terrain in consideration is rainforest terrain which is very highly complicated and unstructured. The primary challenges for rainforest navigation is to extract useful features from vision system and to use the extracted information for terrain classification and obstacle detection (Manduchi, Castano, Talukder, & Matthies, 2005). Perception ability is very important for the autonomous vehicle as the vehicle cannot rely on Global Positioning System (GPS) for localization and obstacle detection. These are due to GPS signal may be covered by the tree canopies and the resolution of GPS map is too low for obstacle detection (Manduchi, Castano, Talukder, & Matthies, 2005). The inertial navigation unit (IMU) is not reliable due to uneven ground surface that may affect the measurement made by the IMU. Thus, the perception ability by vision system is essential for the task of efficient vehicle guidance.

However, vision system in rainforest terrain does have its own problems need to be resolved such as illumination problem, color constancy problem and terrain classification problem (Manduchi, Castano, Talukder, & Matthies, 2005). Contrary to structured terrain, the rainforest terrain is subject to uneven illumination and shadow effect across the scene. These will contribute to color constancy problem in the color features and further complication the segmentation process. A color of an object may appear different at different spot of the object although it belongs to the same object. Object colors may also appear differently at different time of the day and consequently complicates the classification process. The highly unstructured nature of the rainforest terrain also poses problem to the segmentation process as it is difficult to get a meaningful or salient region from the scene. For obstacle detection and terrain classification, an object that appears to be obstacle in geometric point of view may not be an obstacle (Manduchi, Castano, Talukder, & Matthies, 2005). For example, tall green grass may be perceived as obstacle but in fact, it can be driven over by the vehicle. Furthermore, there are color ambiguities in rainforest terrain where the colors of different objects appear to be very similar in color. For example, a tree bark may appear brown in color which is similar to the ground color. The problem of obstacle detection and terrain classification are widely studied, however there is no efficient and robust algorithm to solve the problems.

Other challenging situations in rainforest terrain include presence of negative obstacles (such as ditches) and water bodies (Manduchi, Castano, Talukder, & Matthies, 2005). It should be noted in rainforest, water patches are

very highly anticipated in the scene. Water body may appear in several ways in color imagery (Rankin, Matthies, & Huertas, 2004). The water may be flowing such as river, or may be standing such as water patches. However, the appearance of the water may differ depending on the illumination condition and angle. The water appears to be bright under direct sunlight and darker in intensity while under canopy. Water body may also reflect the surrounding scene or appear to be similar color with the underlying ground. To date, very few studies are conducted on water body detection and for a successful navigation in rainforest terrain, successful detection of water body is essential to determine the traversability of the water body.

1.2 Scope of Research

In this thesis, the research is focused on developing a visual guidance system for autonomous vehicle in rainforest or tropical terrain based on stereo camera. An analysis is made on the disparity data and the color co-registered with the 3D information, emphasizing on the segmentation of the region into meaningful region. Multiple cues approach is used to achieve salient image segmentation and accurate terrain classification. The research is also focused in developing obstacle detection based on stereo vision and polarizer, where common obstacles such as tree trunks, rocks and water are targeted.

The research questions addressed by this research are:

- How to extract the information and features of rainforest terrain from stereo camera? These information and features are needed to detect meaningful target to guide the autonomous vehicle to navigate tropical area.
- How to learn the tropical environment using multiple cues and to make use of information provided to segment the image into salient region and subsequently used for terrain classification?
- How to use polarizer to enhance the color features from stereo vision and to detect water hazard in rainforest terrain?

To produce a complete visual guidance system for rainforest navigation is a daunting task. There are many challenges remains to be solved and in this thesis, we are not aiming to solve all the problems but only several with upmost importance. Based on the literature review in Chapter 2, a few key challenges were identified. Table 1.2 shows the specific problems targeted by this work.

This scope of the project will be focused on, which lead to deliverables:

- Obstacle Detection and Terrain Classification using Color Information.

- To use multiple cues from color stereo camera for terrain segmentation. Features such as colors and textures are analyzed to determine its appropriateness for detecting ground region of the image. The features are then used to classify the terrain type of the scene. The focus is on segmenting the scene into meaningful region with ground region and obstacles as the main region of interest.
- The key point is to make use of disparity data in terrain segmentation where texture and color may fail to differentiate between object of similar color (e.g. tree trunk and ground).
- Ground Plane and Obstacle Detection.
 - Detecting the ground plane of the scene and obstacles in geometric point of view. The stereo information is used to determine the ground plane and to determine any obstacles protrude out from the ground.
 - The obstacles detection will be focusing on general obstacles such as near vertical tree trunk. Negative obstacles or hidden obstacles will not be considered in this work.

- Water Body Detection
 - To differentiate between mud, water patches and streams or river. It should be noted that water patches can be run over.
 - The focus of this thesis is to detect the water patches without determining the depth of the water.

1.3 Thesis Outline

In chapter 2, we present the literature review and state-of-the-art of visual guidance system for autonomous vehicle. Most of the discussions are focused on the challenges and problems present in rainforest terrain, whereby the justification to solve the problems are explained.

Chapter 3 describes the system architecture of the visual guidance system and the approach taken in solving the problems existing in rainforest terrain. The stereo camera and hardware setup are described as well.

Chapter 4 discusses on the developed ground plane detection module. The disparity and color features used in the developed module are discussed and analyzed. Then, the performance of the developed method is tested on various complexity level of rainforest terrain.

Task-Specific Processing	Descriptions
Ground Plane Estimation	Existing methods relied heavily on structured feature to extract the ground plane but they are not applicable in rainforest terrain. A method is proposed to merge existing V-disparity image with the color clustering estimate the ground plane.
Tree Hazard Detection	Tree trunks are common obstacles that appear in rainforest terrain. It is necessary to have a module specifically to detect the trees. In this thesis, the tree trunk is detected by using range data and colour information.
Water Body Detection	Water bodies may be hazardous to the autonomous vehicle in operation. We attempted to detect the presence of water body in the terrain using polarization effect.

Table 1.1: Summary of image segmentation technique and its limitation on the application in rain forest terrain.

In Chapter 5, the tree trunk detection module is described. The characteristic and appearance of tree trunks are discussed. Based on the tree trunks characteristic, the tree trunk detection module was developed and presented.

Chapter 6 discusses the water hazard detection module where the water hazard is detected based on intensity difference caused by polarization effect.

Finally, the conclusion of the thesis is presented in Chapter 7. The implications and experimental results are summarized. Finally, some suggestions for the extension of current work are presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Extensive work has been carried out on autonomous vehicle since 1990s and the works done in the area are mostly dependent on the vision systems or 3D radars to build up a map, terrain classification and obstacle detection. The progress made over the years can be grouped into a few categories depending on the functionality and terrain type of the vehicle. The general category of the autonomous vehicle is indoor navigation and outdoor navigation where the latter category can be further subdivided into structured and unstructured terrain. Since there are huge differences in the approaches for indoor and outdoor navigation, the review focuses on the outdoor terrain as the rainforest terrain in this thesis consideration falls into the unstructured terrain.

This section outlines work carried out in the area of autonomous vehicle and visual guidance, focusing on a selection of range of methods and sensors used for the vehicle perception. The contributions and limitations of each approach will be highlighted while the justification of using stereo vision in this thesis will be reasoned.

2.2 Review of Visual Guidance for Outdoor Terrain: Structured

The works on visual guidance system for structured outdoor terrain usually involve vehicle maneuvering in urban road with specific task such as obstacles detection (usually cars and pedestrian) and avoidance, landmark detection (i.e. road sign and traffic light) and road following. Works by Dickmanns (2002) and DeSouza et al (2002) provide comprehensive review of machine for road vehicle. Although the terrain in consideration of this thesis is very much different from navigation in structured road, it is important to highlight the state-of-the-art in this area as most of the functional systems in unstructured terrain are derived from structured terrain.

One of the key tasks in structured environment is road detection. The vehicle will detect the road and perform road-following. The key feature is road markings on the road and the visual guidance system will detect the lines on the road that separate the lanes and maintain the vehicle position on the road. To date, road-following is a relatively mature technology with many successful implementations such as Navlab projects, ALVINN projects and ARGO projects.

2.2.1 Methods based on NAVLAB and ALVINN

One of the earliest autonomous outdoor vehicles reported is by the Carnegie-Mellon University named Carnegie-Mellon Navigation Lab (Navlab) (Thorpe, Hebert, Kanade, & Shafer, 1988). The vehicle was a testbed which

was used to test perception modules and navigation modules. It was an onboard platform on an actual vehicle for testing in real-world environments. Over the years, it had gone through various implementations with the later implementation was called Autonomous Land Vehicle in a Neural Network (ALVINN) (Pomerleau, 1989) where the focus was on the usage of neural network to increase the robustness in road detection.

Navlab was equipped with color vision for lane tracking and scanning laser range finder for obstacle detection and avoidance. The Navlab approaches include edge detection and image color classification. It attempted to extract the edge of the road with the assumption the road is well structured and to classify the road base on the color on road and non-road area. This Navlab also attempted to fuse the texture cues with color vision features to identity road edges. The texture feature was used to aid in the road detection with the assumption that the road region will appear smoother than the non-road region. Commonly, earlier attempts tend to identify the structured features present in the road to determine the traversable region. The problem of changes in illumination was highlighted where the authors tried to solve it by pre-determining the cluster colors using separate Gaussian clusters.

The ALVINN project was a system equipped in NAVLAB as a neural-network-based navigation system (Pomerleau, 1989). There were several prominent works on ALVINN system to improve the performance of the system which can be found in Batavia et al (1996) and Jochem et al (1995). Generally, the system approach was to use artificial intelligence in autonomous

vehicle to drive automatically. The project had successfully traversed on the highway with highway speed. It utilized the neural network to learn on-line by “observing how human drive”. The digitized signal from camera is fed into neural network module and the output of the training is the steering direction which is fed to the car steering control.

Both the approaches based on the Navlab and ALVINN were only limited to structured road with many assumptions which are not applicable or suitable to rainforest terrain. The learning method used in ALVINN requires a lot of training scheme where certain scenarios can be expected and many assumptions need to be made. On the contrary to rainforest terrain, a structured road cannot be expected to be present and the assumption used on structured road cannot be applied. However, the approach had triggered the interest in using artificial intelligence in autonomous vehicle navigation.

2.2.2 Methods based on ARGO

The ARGO (Broggi, 1999) project is one of the most mature lane detection systems which aim for active safety system and automatic pilot for road vehicle. The system incorporated only passive sensors such as camera, proprioceptive sensor and commercial personal computer in its prototype. Two cameras were mounted at the top corners of the windscreen and a speedometer were used to detect the velocity of the vehicle. The images acquired from the stereo cameras and speedometer were fed to computing system located in vehicle boot. After computation and processing, the output was fed to the

actuator of the car. The computing system was able to compute lane geometry for lane following, detect common obstacle on road and detect lead vehicle. Recently, much of this area of research is concentrated on more complex road condition which takes into the consideration the traffic signs, pedestrian and etc.

Similar to Navlab and ARGO project, the systems cannot be fully applied to rainforest terrain. However, it is important to highlight that all the projects discussed previously relied heavily on vision system to detect the road and obstacles. Evidently, it can be said that the vision system is a key-enabling technology to autonomous vehicle navigation and obstacle detection.

2.3 Review of Visual Guidance for Outdoor Terrain: Unstructured

During the past decade, the research of autonomous vehicle that can operate in unstructured terrain and their related issues began to arise. Early researches and developments of autonomous vehicle were initiated by military organization (Shacklock, Xu, & Wang, 2006) as defense and surveillance were very much of their concern. There were intentions to increase the mobility of the autonomous the vehicle and also the complexity of the missions that can be completed by the autonomous vehicle of robots. Report by Eiker (2001) predicted that the functionality of the autonomous vehicle will achieve several milestones as illustrated in Figure 2.1. It is expected that the vehicle or robot can perform some combat service support, mission and terrain understanding in different type of terrains. From the figure, it can be seen the biggest challenge

in autonomous vehicle is the situational understanding or terrain understanding and the complexity is increasing when the terrain is more unstructured.

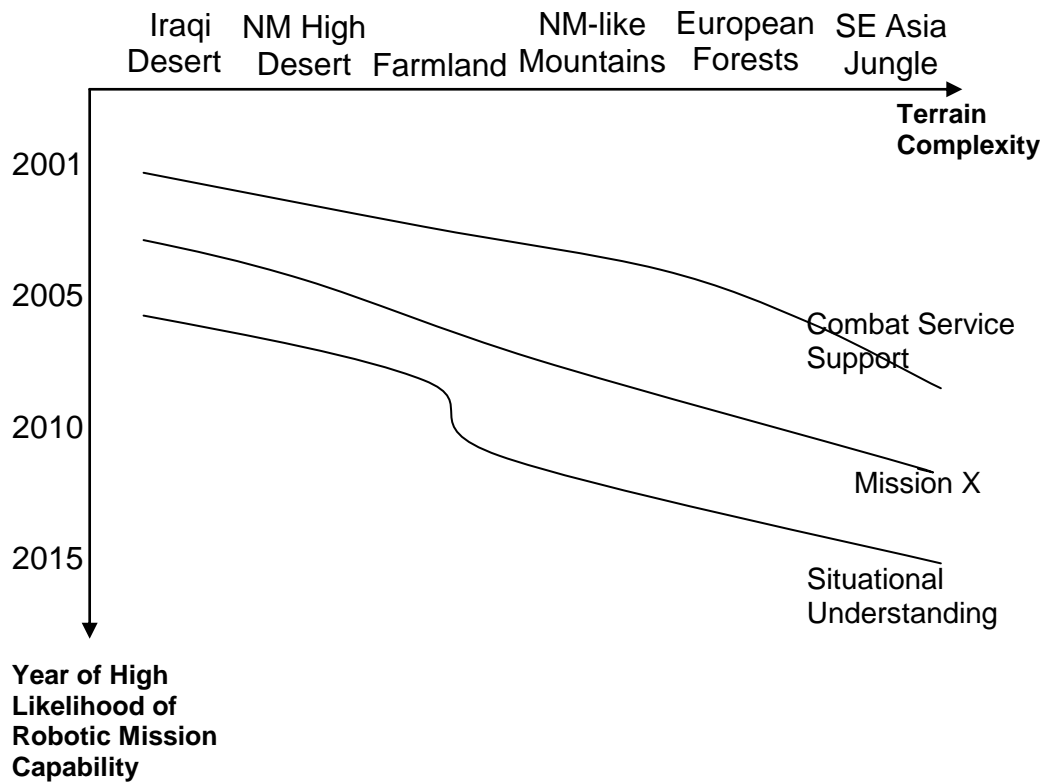


Figure 2.1: Notional Approach to Top Level Systematics of Ground Mobile Robot Systems (Eicker, 2001).

As mentioned above, the emphasis on the terrain type that can be negotiated by an autonomous vehicle is moving toward highly unstructured terrain. The shift of the focus is mainly caused by the type of the mission that needed to be completed. Moreover, recent advanced technologies promote improvement of the hardware devices and these allow faster processing which cannot be achieved previously (Broggi, Bertozzi, Fascioli, & Conte, 1999). One of the key tasks in rainforest navigation is terrain classification where the

type of terrain need to be identified before decision of which path to be taken is decided. Moreover, other key issues, such as system robustness and uncertainties present in rainforest terrain must be handled well. The following subsections will discuss on the state-of-the-art that is very much related to rainforest terrain of interest in this thesis.

2.3.1 Methods by DEMO III

One of the prominent projects is DEMO III by Jet Propulsion Laboratory (JPL) reported by Shoemaker et al (1998) and Bellutta et al (2000). The project focus was on autonomous navigation in cross-country using vision-based perception. Although the terrain type was different from the rainforest terrain, many of the challenges presented were very much relevant and the proposed approach could be used in this thesis.

DEMO III attempted to use real-time stereo systems for geometric representation and obstacle detection. While the stereo systems were able to detect positive and negative obstacles, it is not able to characterize the terrain in terms of its traversability. The main contribution of DEMO III project is that it had successfully introduced effective scene description based on geometric approach. The general problems in unstructured terrain navigation are evident as the flat-world assumption cannot be applied in this situation. In DEMO III, the ground in the terrain in consideration is relatively flat where the vehicle and sensors posed would not be affected dramatically. However, in rainforest

terrain, very bumpy road is expected and the vehicle guidance system must be able to compensate for the change of the vehicle pose from time to time.

While geometric presentation may be good to describe the terrain, the presence of compressible vegetation where the vehicle can run-over posed another challenging situation in unstructured terrain navigation. Some of the vegetation such as grass will be considered as obstacles in geometric point of view. To overcome the problems, DEMO III program performed the terrain classification based on color. The presence of compressible obstacles in DEMO III work was very much similar to the situation rainforest navigation and can be modified to be applied in rainforest navigation. However, the limitation of DEMO III is that relatively open ground is expected where most of the ground is present. In rainforest terrain, the ground terrain may be minimal and surrounded by compressible vegetation, thus in order for a successful navigation, these problems need to be solved.

Many similar projects were conducted on similar terrain as in DEMO III program, the difference of their approaches is the specific task processing for terrain classification and obstacle detection. The approaches were mainly to overcome common problems associated with color feature which are the chromatic problem and uneven illumination.

2.3.2 Methods by Darpa PerceptOR

Most of the research for autonomous vehicle is for military application and it is demonstrated by the prominent project and activities are mainly funded military agencies. The United States Defense Advanced Research Project Agency (DARPA) has been actively developing the reconnaissance capabilities of autonomous robot. For the past decade, DARPA (Krotkov, Fish, Jackel, McBride, Perschbacher, & Pippine, 2006) has engaged in Tactical Mobile Robotics program, DARPA PerceptOR and Darpa Grand Challenge program. While Tactical Mobile Robotics program focused more on urban environment, the following DARPA PerceptOR and Grand Challenge focus were off-road terrain. Darpa PerceptOr tried to sense the environment by creating 3-D model of the scene using stereo and ladar data. The idea was to create a 2-D map with obstacle identified in the map and to move towards the target area based on the map created.

The guidance system of PerceptOR was based on multiple sensors with differing field of view to detect different geometric aspects of obstacles and terrain. In addition to the sensors attached to the ground vehicle, PerceptOR was equipped with flying agent for early obstacles detection, mainly the steep negative obstacles or holes. Two SICK single axis ladars were placed in the forward direction of the vehicle with one scanning horizontally and another one scanning vertically. Two additional SICK ladar were also used for special purposes with one was mechanically rotated continuously to provide 360-degree view around the vehicle while another ladar looks rearward to provide

backup maneuvers. Digital video camera and stereo camera were used to provide colour information and alternative range data to the ladar.

In this work, the visual guidance system was divided into two sensing modes which were appearance sensing and geometric sensing. The geometric sensing module processes the range data to detect geometric hazard (i.e. extreme slope, tree, etc.) while the appearance processes passive features to classify the terrain type. Both modules were fused to calculate the traversability cost and subsequently determining non-rigidity of the terrain. This work highlighted the importance of certain features to determine the compressibility and penetrability of sensed terrain. Compressibility is defined to be the object rigidity which is based on its material composition and penetrability is defined to be the object density based on the solidity and porosity of the object. This work also identified the color to be a powerful cue to determine the compressibility and the authors used neural network to perform the classification based on colors. The penetrability of an object was determined using the density points from the scanning ladar. In addition, the authors used stereo vision with traditional vertical horopter instead of group plane horopter. It was highlighted that ground plane is not prominent in many complex scenes and difficult to isolate.

2.3.3 Methods by LAGR

There are many projects which are based on Learning Applied to Ground Robotics (LAGR) system developed at Carnegie-Mellon. It is a

platform for research which is equipped with two pairs of Bumblebee cameras and on-board GPS antenna for global localization (Happold & Ollis, 2006). Thus, projects that use this platform only deal with higher level vision processing and planning algorithm. Alberts et al (2008) and Happold et al (2006) attempted to use supervised learning in LAGR platform for unstructured outdoor terrain. It should be highlighted that both authors were using different algorithms for their LAGR systems. Alberts et al (2008) managed to introduce fully autonomous and online learning to distinguish between ground plane and other objects while Happold et al (2006) claimed that their system will be able to perform safely on forested course based on their test. While LAGR system has proven to be successful in semi-structured environment, there is very little deployment of LAGR in unstructured terrain.

2.3.4 Methods by Manduchi et al (2005)

This work (Manduchi, Castano, Talukder, & Matthies, 2005) presented sensor algorithm that are suitable for cross country algorithm. The approach used a stereo camera and single-axis lidar for obstacles detection and terrain classification. The proposed methods were based on stereo range measurement, color-based classification and lidar data to label the detected obstacles according to a set of terrain classes.

The author attempts to detect obstacles using the 3-dimensional point cloud directly. Instead of extracting the reference ground plane, the method extracted the slope and height of the visible points. Certain objects with height

above slope and height threshold will be considered as obstacles, otherwise traversable region. The impetus of this method was to minimize the effect the uneven ground that may affect the cameras tilt and roll. The color-based classification for the terrain typing concentrated mainly on the main obstacles in the terrain which are soil or rock, green (photosynthetic) vegetation and dry (non-photosynthetic) vegetation (which includes tree bark). Any terrain which was not included will be placed into a general group. Any classes with color ambiguities will be differentiated with the aid of the range data from radar.

This research is one of the pioneers of visual guidance for autonomous vehicle in complicated terrain and is very similar to the rainforest terrain of interest. The work did not attempt to solve the all navigation problems at once but to arbitrary solve important challenges one at a time. This approach worked very well as different sensors have their advantages in detecting certain obstacles over other obstacles. There are a few challenges that were highlighted by this work that require further investigation which are the varying illumination condition and color shift in atmospheric condition. There is a need for efficient and robust illumination compensation method for outdoor terrain as the illumination across the terrain is subject to shadow effect and irregular illumination. Consequently, it will affect the objects color across the scene.

2.4 Challenges in Rainforest Terrain

The preceding sections have briefly discussed the achievements and requirements of successful navigation for autonomous vehicle. However,

navigation in rain forest terrain represents a highly complex set of problems that need to be solved before it can be successfully employed. With more unknowns appear within complex environment, we anticipate more difficulties in task-specific processing in tropical terrain. Possibly the biggest challenge in rainforest terrain is to sense the environment and use the information for path planning. However, the task of sensing the environment is made difficult by very highly unstructured terrain and contains many uncertainties.

In rain forest terrain, obstacle detection based on distance and height information is not sufficient to determine whether the obstacles really pose danger to the vehicle. The following are some possible challenges that are typical in rain forest terrain:

1. Long grass or soft vegetation will be detected wrongly as obstacles in geometric point of view as the vehicle could have driven over the grass easily. In rainforest, the ground is uneven and it may contain slopes, bumps or negative obstacles.
2. Negative obstacles may pose bigger problem as the depth and size of the obstacles cannot be accurately determined until the vehicle is very near to the obstacles, depending on the sensors height and size of the obstacles. The condition can be made worse if the some of the obstacles (negative and positive) are hidden behind soft vegetation.

3. The illumination is uneven across the rain forest as most of the terrains are covered by thick forest canopy. There are direct, indirect sunlight and shadow effect which affect the colors of the object. The color distribution of an object appears to be broadened depending on the amount of light illumination. Pixels may be over-saturated if the object is heavily illuminated, and therefore affecting the color information contained in pixels. Consequently, it will affect the segmentation and feature extraction based on color features from color camera.

4. Terrain classification plays an important part in determining traversable region and non-traversable region. By classifying and determining the traversable area, the optimal path for the vehicle can be chosen. However, it is difficult to segment the scene into salient or meaningful region. Thus the focus on recent approaches are on minimizing or compensating the effect of uneven illumination.

5. Another concern in tropical terrain is color ambiguity of different objects. Some of the objects appear to be similar in color and hard to be differentiated based on color alone. For example, there are difficulties to differentiate between objects such as tree trunk, dry vegetation and soil. The color distributions of these objects are very close together and overlapping in feature-space, thus making segmentation process harder to achieve. It is difficult to form a clear cluster to represent certain object as some of the cluster center may appear to be close together and differ in size. The results of the segmentation depend on the number of clusters that

is set prior segmentation process. Different number of cluster will yield different results and in unknown environment it is difficult to assume the number of cluster beforehand.

6. The wet nature of rain forest means that the scenario may have water bodies and muddy patches around. The vehicle guidance system needs to be able to differentiate between deep water and shallow water region to avoid damage to the vehicle itself. To our knowledge, there is no work done to determine the depth of the water bodies present in rainforest terrain. In addition to that, the color of certain material changes due to wet condition and atmosphere effect. The reflection spectrum of a material may varies due to moisture content. For example, the color of soil may appear to darker in drenched condition compared to dry condition. This makes color-based classification rather difficult and it represents a major problem.
7. The uneven ground plane is one of the major problems in rainforest navigation. The uneven ground will affect the tilt and roll of the vehicle and sensors which make difficult to detect and estimate the ground plane. In addition, the ground plane may appear to be less prominent in rainforest.

2.5 Summary

Based on the literature review, we have identified several key areas that are essential for a visual guidance system for rainforest navigation. We aimed at specific problems that are generally present in the terrain of interest and may

improve existing visual guidance system functionality and reducing their limitations. The following are the challenges taken into consideration:

- Ground Plane Estimation
 - The ground plane plays an important part in navigation. The load bearing surface need to determined beforehand and act as the reference plane in the terrain and obstacle map.

- Tree Trunk Detection
 - In rainforest terrain, the tree trunk is very common in the scene. The trunk will appear in most of the scene and it is important to have task-specific processing specifically to detect this obstacle.

- Water Hazard Detection
 - There is very little work done this water body detection. It is important to detect the water bodies as it may be hazardous to the autonomous vehicle.

Title/Author	Terrain Type	Visual Sensors		Sensors					Task-Specific Processing	Contributions	Remarks/ Limitations	
		Camera	Stereo Camera	Infrared Camera	LADAR	RADAR	GPS	Others				
(Nilsson, 1969)	Indoor	x			x				x	Used line as their main feature to detect the obstacles and ground plane.	Introduces vision as perception of vehicle to detect the empty ground.	Not versatile. Visual sensors are not used to detect obstacle.
(Gennery, 1977)	Indoor		x							Correlation and matching of stereo images	Proposed the usage of stereo vision to detect ground plane and obstacles.	Priori model of the obstacles are needed
NAVLAB	(Thorpe, Hebert, Kanade, & Shafer, 1988)	Outdoor (On-road)	x			x				Road-following and obstacle detection.	Color classification and road modeling.	Failure occurs when there is drastic change of illumination and shadowing.
	(Pomerleau, 1989)	Outdoor (On-road)	x			x					Neural Network for road-following	Long algorithm development and parameter tuning (months). Require large training samples.
	(Jochem, Pomerleau, & Thorpe, 1995)	Outdoor (On-road)	x			x					Improvement from previous ALVINN projects, where road junctions' challenges were concentrated. The concept of virtual camera was introduced.	Flat world assumption and requires careful alignment and calibration.
ALVINN	(Batavia, Pomerleau, & Thope, 1996)	Outdoor (On-road)	x			x				Improved robustness and faster training time compared to previous neural network approach.	Priori knowledge on the road is required.	
	ARGO	(Broggi, Bertozzi, Fascioli, & Conte, 1999)	Outdoor (On-road)		x					Road-following and obstacle detection.	Lane and obstacle detection based on model-based approach. Full vision-based system on the road with vehicle detection and vehicle overtaking module. One of the most matured autonomous road vehicle.	Relies on the road marking on the road. Susceptible to shadowing and changing illumination
		(Broggi, 1999)	Outdoor (On-road)		x							
(Broggi, Fascioli, & Bertozzi, 2000)		Outdoor (On-road)		x								

Table 2.1: Summary of visual guidance system approaches in indoor and semi-structured outdoor terrain.

	Title/Author	Terrain Type	Visual Sensors		Sensors					Task-Specific Processing	Contributions	Remarks/ Limitations
			Camera	Stereo Camera	Infrared Camera	LADAR	RADAR	GPS	Others			
JPL DEMO III	(Shoemaker & Bornstein, 1998)	Outdoor (Off-road)		x	x	x	x		x	Geometric cover representation, terrain classification based on color, negative obstacles detection.	Successfully maneuver based on the elevation and obstacle map generated from geometric information. The terrain classification to determine the traversability of certain cover was performed based on color feature.	From the literature, the test bed seems to be tested on a relatively flat ground with certain sensors pose assumed. No work was performed on water body detection.
	(Bellutta, Manduchi, Matthies, Owens, & Rankin, 2000)	Outdoor (Off-road)		x	x	x	x		x			
JPL PerceptOR	(Stentz, et al., 2003)	Outdoor (Off-road)	x	x		x		x	x	Geometric cover representation, terrain classification based on color and range data, negative obstacles detection.	Reliable detection of non-geometric objects, but more work are needed. Successfully traverse in dessert terrain.	The use of unmanned aerial vehicle is not applicable in rainforest terrain where canopies are present as constraint.
	(Kelly, et al., 2004)	Outdoor (Off-road)	x	x	x	x		x	x		Pose-estimation based on multiple sensors available. Traversability determination based on 3D volumetric density mapping.	Level of adaptability of perception algorithms is still unsatisfied in varying terrain.
	(Huertas, Matthies, & Rankin, 2005)	Outdoor (Off-road)		x							Traversability determination based on diameter of tree.	Assume all tree barks are vertical or almost vertical.
	(Rankin, et al, 2005)	Outdoor (Off-road)	x	x	x						Performance of passive sensors in complicated terrain.	The resolution of the sensors is fairly low which result in low range accuracy.
	(Krotkov, Fish, Jackel, McBride, Perschbacher, & Pippine, 2006)	Outdoor (Off-road)	x	x	x	x	x	x	x		Evaluation experiment of various sensors in various unstructured terrain. Detailed investigation in unstructured terrain challenges.	Comprehensive tests on the unstructured terrain navigation.
JPL	(Manduchi, Castano, Talukder, & Matthies, 2005)	Outdoor (Off-road)	x	x	x	x	x	x	x	Terrain classification based on color, negative obstacles detection.	Detailed investigation in unstructured terrain challenges. Covers majority of essential items needed in unstructured terrain navigation.	Comprehensive work on unstructured terrain navigation. Highlighted the importance of water body detection and illumination compensation.

Table 2.2: Summary of visual guidance system approaches in unstructured outdoor terrain.

CHAPTER 3

SYSTEM ARCHITECTURE

3.1 Overview

Based on the previous literature review, a task-specific visual guidance system is developed in this thesis for autonomous vehicle navigation. The proposed ground plane detection and water body detection solve some of the critical requirements for autonomous vehicle navigation in rainforest terrain. The problems are solved independently as task-specific module. By examining the reports and publications, it is clear that the key enabling for this autonomous vehicle is the visual guidance system (VGS) of vehicle. In order for this autonomous vehicle to be feasible, the visual and perception information of view must be fully utilized for scene understanding and control. Essentially, the role of visual guidance system is to capture the raw sensory data and convert the data into meaningful information for model representations of the terrain (Shacklock, Xu, & Wang, 2006).

In this work, a visual guidance system has been developed based on stereo camera to achieve our target missions. There are many visual features available from the image pair, such as color, texture, motion, disparity and so on. It is necessary to extract the suitable and useful features from the obtained image pairs. Color features will be used together with the disparities features since it is co-registered with the stereo images provided with the stereo camera.

First compared with other visual guidance systems, this system is designed with the following three solutions for navigation in rainforest terrain:

1. Robust and simple ground plane detection for rough ground terrain where the load-bearing surface is uneven.
2. Obstacle detection in particular vertical tree trunks using disparity data.
3. Water patches detection using two polarizers with different polarization angle on stereo camera.

In other words, this thesis intends to solve the problems mentioned specifically above. The missions are achieved by modifying and improving existing algorithm and by using additional hardware setup.

In the next section, the proposed visual guidance system and methodology are discussed. The approach to solve the challenges in rainforest terrain by problem-based model is explained. Next, the features from stereo image pair and polarization effect are described. This includes the analysis of the features in rainforest terrain.

3.2 Visual Guidance System

Figure 3.1 shows the general framework for a visual guidance system where the modules are sensors, modeling and calibration, task-specific

processing, sensor fusion and world mapping. The sensors are grouped together under one module. The main function of this module is to perceive the scene in front of the vehicle and its features is further processed for terrain classification, localization, mapping and path planning. This module may consist of multiple sensors which include color camera, stereo camera, laser range finder and other vision sensors. The modeling and calibration modules are used to model the sensors used and to calibrate the sensors accordingly if they are going to be used as measured devices. The raw data acquired will be converted into features and modeled into projective representation of the scene.

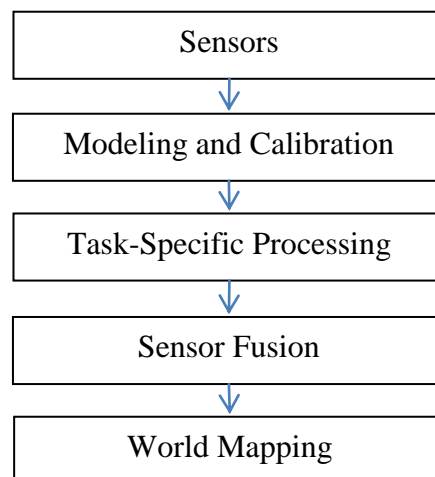


Figure 3.1: General modules of visual guidance system (VGS).

The task-specific processing module includes the function to process the information acquired from the sensors. Each data stream from different type of sensors may has different type of features that may be used to perform different functions. Essentially, its role is to extract raw sensor data and extract useful features that may be used for specific function. For example, a camera

has texture features that are useful for terrain classification while stereo camera contains additional depth information that is useful for obstacle detection. While having more features may increase the likelihood of correct classification and detection, it comes with higher cost and higher processing load.

The sensor fusion module function is to merge all the processed features to increase the chances of correct classification and detection. As mentioned earlier, each feature from different type of sensors may detect different obstacles more effectively with certain confidence level. Associating those features together may increase the confidence level and improve the probability of correct detection. The output of feature fusions will be fed to world mapping module which further map the classification and obstacle detection result into real world.

3.2.1 Proposed System Architecture

The proposed visual guidance system has the ability to determine the ground plane of the scene and detecting obstacles and water hazard. The approach taken in this thesis is module-based system where each module developed is aimed to perform specific task. It is task-oriented module and the modules are formulated based on the challenges discussed in section 2.4. A general visual guidance system is developed with three main modules which perform the task specific processing of interest in this thesis. The earlier

overview of this chapter specified the three challenges that we intend to solve in this work.

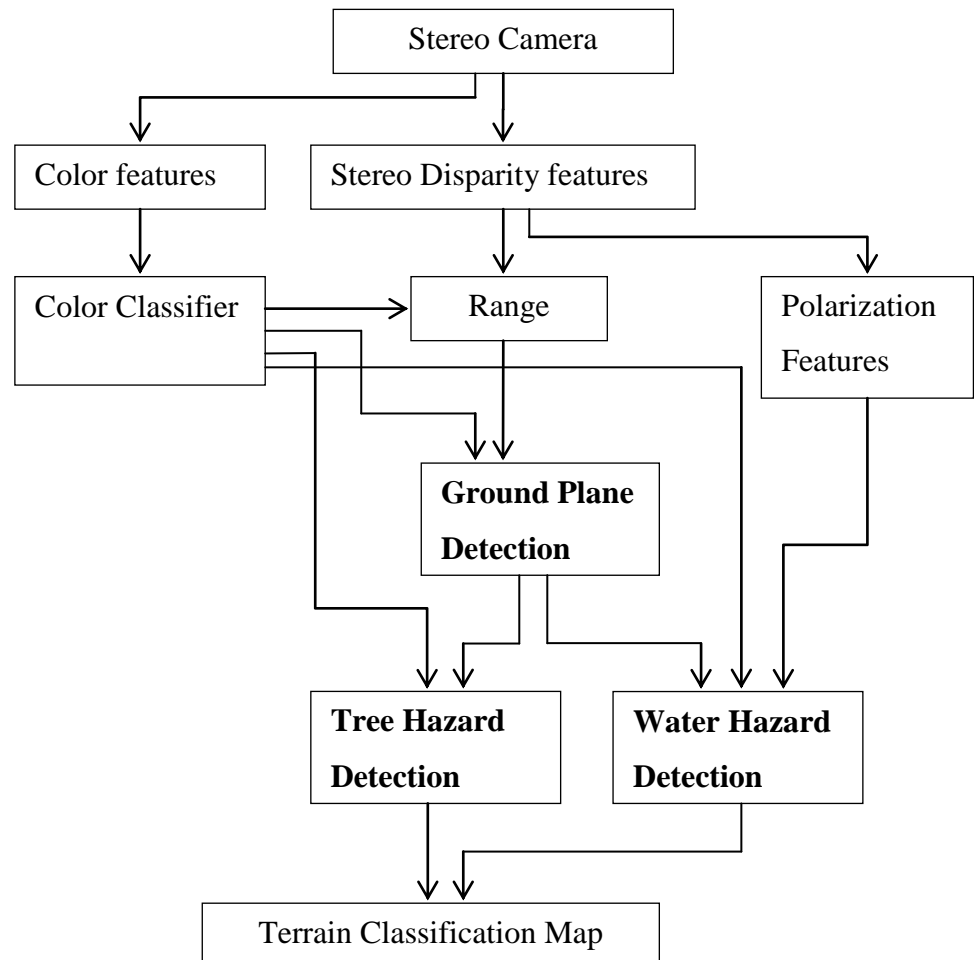


Figure 3.2: Architecture of the visual guidance system.

Figure 3.2 shows the architecture of the visual guidance system developed in this thesis. The features are obtained from polarized stereo camera which are colors, disparity, and polarization feature. The color classifier is used to process the features available before being used to perform certain detection tasks. The three task-specific processing modules are ground

plane detection, tree hazard detection and water hazard detection. Each module will perform the specific task and the outputs will be fused to generate a terrain classification and obstacle map.

In short, this thesis proposes a feature-based visual guidance system with task specific modules to solve the problems highlighted mentioned in mentioned earlier.

Initially, stereo image pairs are grabbed from the calibrated stereo camera. Both side of the lens are polarized with different polarization angle where the difference of the polarization angle is 90° .

Then, the color information and disparity feature are extracted from the image pair. Several pre-processing procedures are done on the features before each cue are fed to task-specific processing.

- Next, the color feature, polarized-color feature and range feature are used for task-specific module. Color segmentation and ground plane estimation are performed earlier as the information is needed by other modules.
- The color classifier module is used mainly for terrain segmentation and classification. The primary function of this module is to segment the scene into few regions that may pose as obstacles without determining the compressibility. The tree and water hazard

detection modules are used to detect the tree barks and water patches present in the scene.

- Finally, the outputs from each module from each module are fused to produce the terrain classification map that can be used for navigation.

3.3 Sensor Description

This section describes the stereo camera used in this thesis to gather sample test images and to perform the experiment. This includes the computer vision library used to grab images from the camera and perform general image processing. Generally, it consists of a stereo camera, central processing unit (laptop), two circular polarizers and the relevant mounting accessories. The stereo image pair of the scene is captured using stereo camera with the polarizer in front of the lenses and passed on to laptop to process the frame.

3.3.1 Bumblebee 2 Stereo Vision System

The stereo camera (Bumblebee 2) is from Point Grey Research, Inc (Point Grey Research Inc, 2003). As shown in Figure 3.3, the stereo camera is designed to incorporate two Sony 1/3" progressive scan CCD. The stereo camera is pre-calibrated for lens distortion and camera misalignments where

the left and right images are aligned to within 0.05 pixel RMS error. The stereo camera specifications are listed in Table 3.1.



Figure 3.3: Bumblebee 2 stereo vision system.

For this project, the low-resolution mode is used where the resolution is set to 640×480 with the frame rates of $15fps$. Most of the researches used lower resolution image in real-time application. However for this thesis, all the initial tests and analysis are based on higher resolution before it is scaled down for real-time application. The frame transmission to the processing unit is done via standard 6-pin IEEE-1394 connector and the synchronization of the frames is incorporated internally.

Specification	Low-Res(640x480)	High-Res(1024x768)
Imaging Sensor	Two Sony 1/3" progressive scan CCD	
	ICX424(648x488 max pixels)	ICX204(1024x768 max pixels)
	7.4 μm square pixels	4.65 μm square pixels
Baseline	12cm	
Lens Focal Length	3.8mm with 70° HFOV or 6mm with 50° HFOV or	
A/D Converter	Analog Devices 12.bit analog-to-digital converter	
Video Data Output	8, 16 and 24-bits digital data	
Frame Rates	48, 30, 15, 7.5, 3.75, 1.875 FPS	18, 15, 7.5, 3.75, 1.875 FPS
Interfaces	6-pin IEEE-1394 for camera control and video data transmission, 4 general-purpose digital input/output(GPIO) pins	
Voltage Requirements	8-32V	
Power Consumption	Less than 3W	
Gain	Automatic/Manual/One-push Gain modes(0dB to 24dB)	
Shutter	Automatic/ Manual/ One-Push Shutter modes	
	0.01ms to 66.63ms @ 15FPS	
	Extended Shutter modes	
	0.01ms to 7900ms @ 15FPS	0.01ms to 5200ms @ 15FPS
Gamma	0.50 to 4.00	
Trigger Modes	DCAM v1.31 Trigger Modes 0, 1, 3, and 4	
Signal To Noise Ratio	Greater than 60dB at 0dB gain	
Dimensions	157mm x 36mm x 47.4mm	
Mass	343grams	
Camera Specification	IIDC 1394-based Digital Camera Specification v1.31	
Emissions Compliance	Complies with CE rules and Part 15 Class A of FCC Rules	
Operation Temperature	Commercial grade electronics rated from 0° to 45°	
Storage Temperature	-30° to 60°	

Table 3.1: Bumblebee 2 stereo camera specifications.

The Bumblebee 2 stereo camera comes with automatic gain control where it control the image brightness correspond to lighting condition. It automatically increases the gain of the camera in the event that the scene is low in illumination. This feature is heavily used in the project as the polarizers affect the light going into the CCD sensors.

3.3.2 Triclops Stereo Vision SDK (PGR Software Development Kit)

The Triclops Stereo Vision Software Development Kit (Triclops Library) is provided by Point Grey Research (PGR) Incorporation to configure and grab stereo images from Bumblebee 2 Stereo Vision camera. The goal of the Triclops Library is to provide the Bumblebee 2 Stereo Vision with accurate and fast depth map generation.

The Triclops library system allows user to specify and process all characteristics of stereo image such as image resolution, disparity range, etc. The library is developed to produce efficient stereo processing and depth image generation and the software system allows the specification of multiple stereo processing that may occur on a single set of images. Thus, the stereo image pair sequence in this project are grabbed from the camera using this library and subsequently, the disparity image is produced before being passed processed by Intel OpenCV library function.

3.3.3 Intel OpenCV Library

OpenCV is an open source (see <http://opensource.org>) computer vision library available from <http://SourceForge.net/projects/opencvlibrary>. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. It was designed for computational efficiency and with strong focus on real-time applications. OpenCV is written in optimized C and can take advantage of multicore processors. One of OpenCV's goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly (Bradski & Kaehler, Learning OpenCV, 2008). The OpenCV library contains over 500 functions that span many areas in vision and machine learning.

In this project, the images grabbed by Triclops Library will be processed using Intel OpenCV Library to fully utilize Intel microprocessor architecture, thus speeding up the runtime of basic image processing. The application of OpenCV in this project includes the pre-processing of color images and machine learning function for terrain classification. Although the machine learning function provided is relatively general and basic, it is sufficient to perform the needed functionality.

3.3.4 Hardware Setup

From the Figure 3.4, the experimental setup is hooked on camera tripod which enables the yaw, pitch and roll angle to be adjusted. The Bumblebee 2

stereo camera is enclosed in the metal casing with two circular polarizers mounted in front of the stereo camera. The basic setup for the actual visual guidance system is very similar to this setup except it is not mounted on an autonomous vehicle.

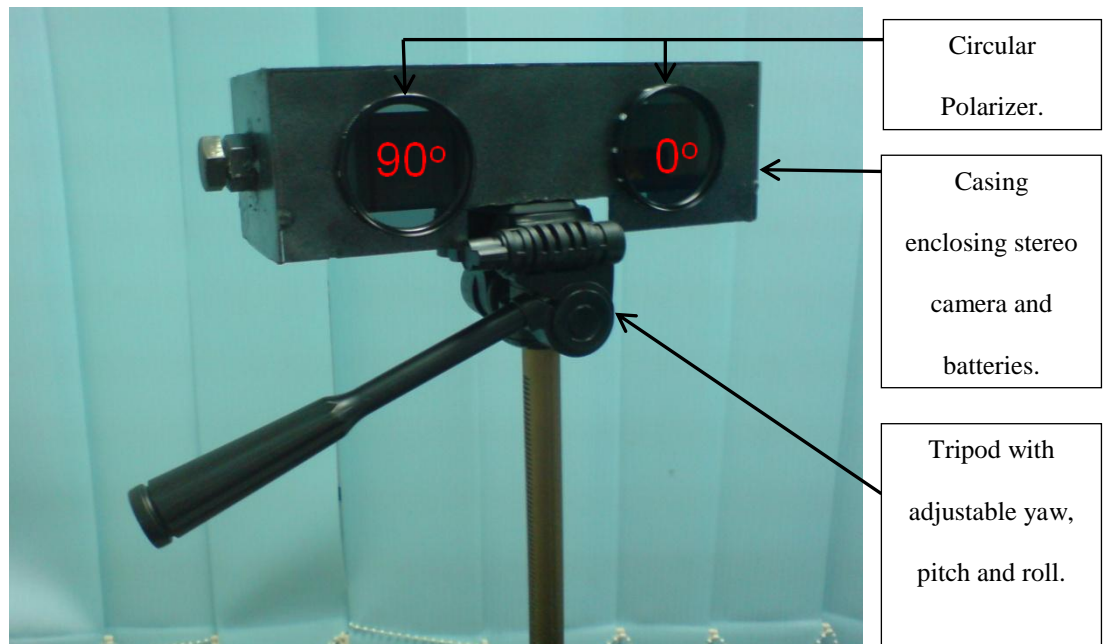


Figure 3.4: Bumblebee 2 stereo vision system and polarizer assembly mounted in front.

The stereo camera is mounted 1.3 meter from ground level. At this height, it is sufficient for the camera to have higher range of view. The camera pitch is set to be -15° from the negative x-axis. From autonomous vehicle operation point of view, the pitch angle of the camera will oscillate as it is susceptible to vibration due to the uneven ground level. It is not feasible to assume the pitch angle of the camera is the same during the operation time. Thus, there is a need to have a robust system that is independent of pitch angle

or able to detect the pitch angle real-time. In this thesis, we tried to solve the problem by using V-disparity image with additional color clustering process.

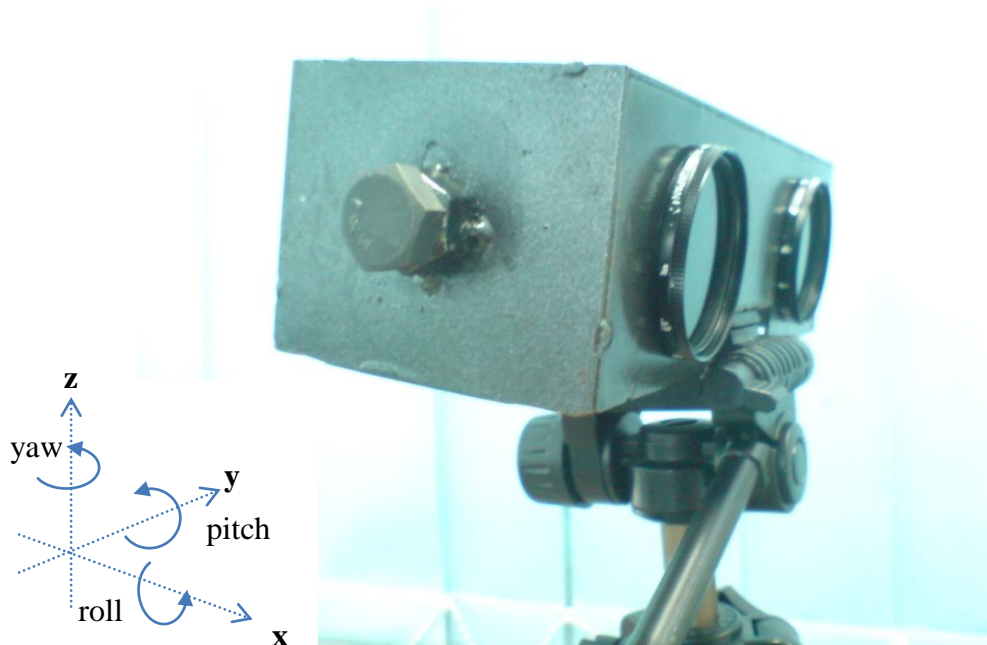


Figure 3.5: Experiment setup with the pitch angle set at -15° from the negative x-axis.

3.4 Overview of Stereo Vision Principles

Stereo vision is an approach to perform range measurement from two images taken from two different viewpoints (Hartley & Zisserman, 2003). The aim is to extract three-dimensional (3D) information from its image pair where the disparities map or depth map are generated. In the image pair, if the image points correspond to the same physical point in real-world-space can be determined, the 3-dimensional location of the physical point can be

determined. The primary problems to be solved to obtain three-dimensional information are calibration, correspondence, and reconstruction (Brown, Burschka, & Hager, 2003). Calibration is the process of relating camera system internal geometry (focal length, optical centers and lens distortion) and external geometry (the relative positions and orientations of each camera). However, the problem of calibrating the camera is well-understood and there are fairly many methods available to solve this problem. The correspondence problem is the process of determining the locations in each image in the image pair that are the projection of the same physical point in real-world-space. No general solution to the correspondence problem exists, due to ambiguous matches (e.g., due to occlusion, specularities, or lack of texture). To make the correspondence between two images possible, there are variety of assumptions and constraints made, depending on the problem of interest. Table 3.2 shows the available methods to solving stereo correspondence problems. All these methods attempted to match pixels in an image with the corresponding pixels in its pair with different constraint such as epipolar geometry and assumptions.

Approach	Brief Description
Local Methods	
Block Matching	Search for maximum match score or minimum error over small region, typically using variant of cross-correlation or robust rank metrics.
Gradient-Based Optimization	Minimize a functional typically the sum of squared differences, over a small region.
Feature Matching	Match dependable features rather than intensities themselves.
Global Methods	
Dynamic Programming	Determine the disparity surface for a scanline as the best path between two sequences of ordered primitives. Typically, order is defines by the epipolar ordering constraint.
Intrinsic Curves	Map epipolar scanlines to intrinsic curve space to convert the search problem to a nearest-neighbors lookup problem. Ambiguities are resolved using dynamic programming.
Graph Cuts	Determine the disparity surface as the minimum cut of the maximum flow in a graph.
Nonlinear Diffusion	Aggregate support by applying a local diffusion process.
Belief Propagation	Solve for disparities via message passing in a belief network.
Correspondenceless Methods	Deform a model of the scene based on an objective function.

Table 3.2: Stereo correspondence approach. (Brown, Burschka, & Hager, 2003).

The reconstruction problem is the process of determining three-dimensional structure from a disparity map, based on known camera geometry (Brown, Burschka, & Hager, 2003). The depth of a point in real-world-space

can be determined by performing triangulation on the ray and camera geometry. Figure 3.6 shows two camera arrangement with the image planes are coplanar to each other, the optical axes are exactly parallel with a distance T apart, and with equal focal lengths $f_l = f_r$. The principle points (c_x^l and c_x^r) which are the intersections of the principal ray projected from the center of projections (O_l and O_r) are assumed to be at the center of the image plane.

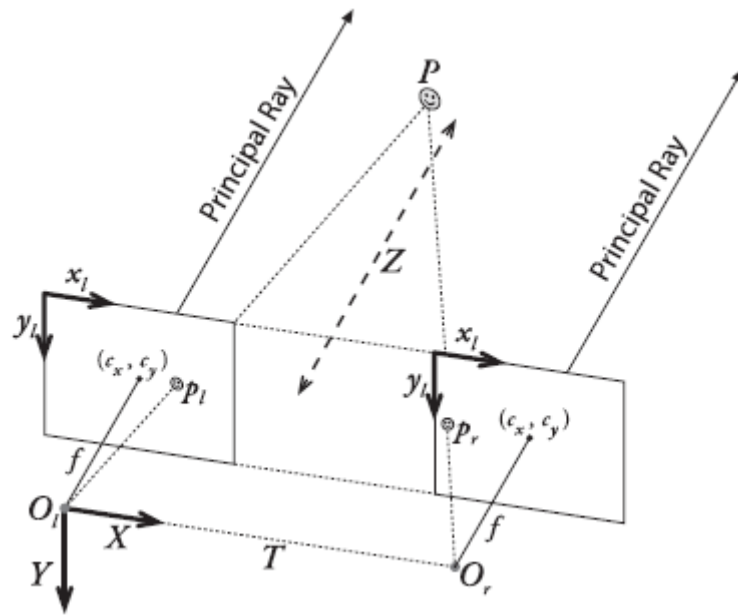


Figure 3.6: Frontal parallel arrangement of two cameras. (Bradski & Kaehler, 2008).

The image planes are assumed to be row aligned and that every pixels row of one camera aligns exactly with the corresponding row in the other camera, thus making the search of the correspondence to be just horizontal. A point P in the real world can be viewed in both image planes as p_l and p_r where the respective coordinates are (x_l, y_l) and (x_r, y_r) .

The disparity is defined as

$$d = x_l - x_r \quad 3.1$$

The depth Z is the distance of the object from the camera and can be determined using triangulation. Referring to Figure 3.6, we have

$$\frac{T - (x_l - x_r)}{Z - f} = \frac{T}{Z}$$

$$Z = \frac{fT}{x_l - x_r} = \frac{fT}{d}$$

3.2

It is assumed that the stereo image provided in Figure 3.6 is perfectly undistorted, aligned and measured. The assumptions are made to simplify the description of stereo vision operation. However in practice, it is difficult to build systems with a frontal parallel arrangement of two cameras. Instead, the images from two cameras need to be mathematically rectified into a frontal parallel arrangement.

3.4.1 Visual Depth using Bumblebee 2 Stereo Vision System

In this thesis, the three-dimensional depth information is obtained from the disparity image. The stereo camera (Point Grey Research (PGR)

Bumblebee 2 Stereo Vision System) used is factory pre-calibrated and stereo computation are performed using Triclops library (PGR Software development kit). The process of computational stereo performed is divided into two blocks where the first processing block is the image pre-processing block that applies a low-pass filter, rectifies the images and performs edge detection. The second processing block does stereo matching, validation of results and subpixel interpolation.

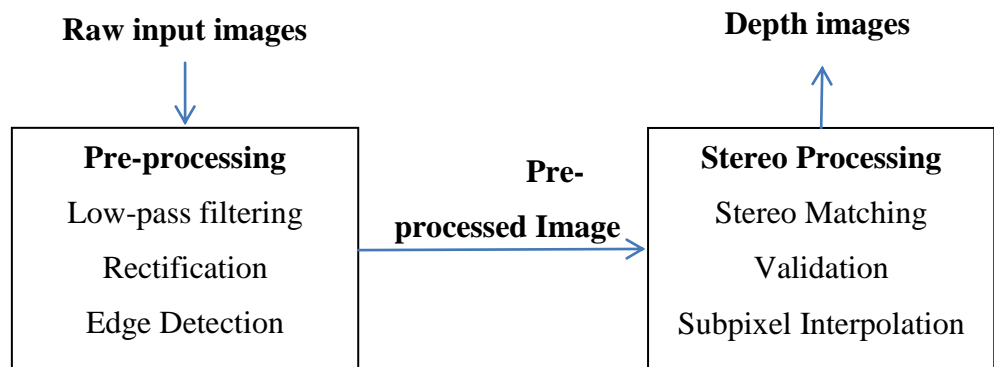


Figure 3.7: Triclops general computational stereo process (Point Grey Research Inc, 2003).

The pre-processing block of the Triclops library prepares the raw image pair from the stereo camera for stereo processing. The low-pass filtering step is done to reduce the aliasing effects. The rectification step is performed to remove the lens distortion and to make the image pair be row aligned. The edge detection allows matching on the changes in the brightness rather than the absolute values of the pixels in the images. This feature is useful because the cameras in the Triclops camera module have auto gain control. If the auto gains

in the cameras do not change identically, the absolute brightness between images may not be the same. While absolute brightness is not the same, the change in the intensity stays constant. Therefore, using edge detection will help in environments where the lighting conditions change significantly (Point Grey Research Inc, 2003).

The stereo processing block performs the computational stereo process where the correspondence of image pair is determined. Block matching is used to find the correspondence between the image pair. This method perform matching by searching for matching pixels in its corresponding image pair over a small number of pixels surrounding the pixels of interest. An image in the image pair is selected as a reference image and a neighborhood of a given block in other image is selected. The block act as a correlation window and its function is to search for the possible matching pixel. The matching pixel is chosen from the correlation window based on correlation measure. For discussion of block matching methods see work by Brown et al (2003) as it is beyond the scope of this thesis to discuss it.

In this thesis, the Sum of Absolute Difference (SAD) is employed as a measure to correlate the pixels between the image pair. This method is popular for its computational efficiency as it only matches the image pair within a region of interest instead of whole image and the computational time is reduced (Brown, Burschka, & Hager, 2003). The SAD is performed using the following equation:

$$\min_{d_{min}}^{d_{max}} \sum_{i=m/2}^{m/2} \sum_{j=m/2}^{m/2} (I_{right}[x+i][y+j] - I_{left}[x+i+d][y+j]) \quad 3.3$$

where d_{min} , d_{max} , m , I_{right} and I_{left} are the minimum disparities, maximum disparities, mask size, intensity value of right image and intensity value of left image respectively.



Figure 3.8: Sample stereo image pair and disparity image. (a) Left image. (b) Right Image. (c) Disparity Image.

The validation process is to ensure the stereo matching is correct. In some cases where occlusion occurs or the scene is lack of texture, it is not possible to perform stereo matching and the correspondence cannot be found. In order to avoid incorrect measurement, the Triclops library introduced two validation method using texture and uniqueness. Texture validation determines whether disparity values are valid based on levels of texture in the correlation mask (Point Grey Research Inc, 2003). Uniqueness validation determines whether the best match for a particular pixel is significantly better than other matches within the correlation mask. Even if the correlation mask has enough

texture, the correct match may not exist due to an occlusion. If the correlation result is not strong enough, the pixel will be declared invalid.

The Triclops library allows matching between images to subpixel accuracy. The library takes advantage of the matching results of the neighboring pixels of the resulting disparity to determine an approximation that is within a fraction of a pixel. Accurate calibration between cameras allows an accuracy of 0.2 of a pixel. However, this function marginally increases the computation time.

3.5 Summary

This chapter describes the stereo camera-based visual guidance system setup in this project. The framework of the whole system is designed based on the problems formulated from investigation on literature review and rainforest scene.

The stereo camera with circular polarizers is mounted in front collect all sensor data on the scene and the depth image is produced using the Triclops library. This library is used as it is specifically designed to grab images from stereo camera. Further basic pre-processing is done using Intel OpenCV. Then, each of the images will be passed to respective task-specific modules.

It is clear that each module will have its own task-specific mission and the module will solve the problem assigned to each module. The reason of

adopting this module-based architecture is because it will provide room for functionality expansion in future. In this thesis, each module represents a problem of interest in the rainforest terrain.

CHAPTER 4

GROUND PLANE DETECTION

4.1 Introduction

As mentioned in Chapter 3, this thesis intends to solve several problems that are critical for visual guidance for autonomous vehicle in rainforest terrain. Visual guidance system is essential to detect the vehicle and classify the according to its classes. The obstacles detected and classified terrain will be mapped into a map for path planning to guide the vehicle to its destination. The approach taken is module-based where each module will be performing the task to solve the problem of interest.

In this thesis, ground plane detection is presented as first step for the visual guidance system. There are several methodology approaches to detect the obstacles. An approach is to detect the obstacles directly from the features such as images, 3D-point cloud etc. Another approach is to use a reference plane and maps the obstacles according to the reference plane. The developed method is based on V-disparity image and is modified for guidance in visual guidance system. There are two main features used in this method, mainly V-disparity and color feature. The V-disparity image is obtained from the disparity feature. The method by Labayrade et al (2002) is improved by using additional color feature to improve the robustness of the developed algorithm.

Section 4.2 reviews the state-of-the-art of the ground plane detection used in autonomous navigation. Section 4.3 presents the characteristic of the ground plane in rainforest scene where necessary feature can be utilized to detect the ground plane. Section 4.5 discusses the ground plane detection based on modified V-disparity image. Section 4.6 presents the results of the developed method and finally the summary is presented in Section 4.7.

4.2 Overview of Ground Plane Detection

In this section, the state-of-the-art for estimation of the ground plane using stereo vision is presented. Ground plane estimation plays an important part in stereo vision-based visual guidance. In order to detect the obstacle, the ground plane need to be extracted and base on the ground plane as foundation, possible obstacles can be detected.

Murray et al (2000) approach is to calibrate camera rig where the position of the camera to the ground is normally fixed. With the setup arrange, the disparity of the ground can be determined during calibration stage. However, this approach is not suitable for outdoor or complicated terrain as the ground plane is uneven and the position of the camera relative to the ground is always changing. Nedevschi et al (2004) tried to obtain the camera position by using four sensors mounted on the chassis and wheels where pitch angle is acquired real-time. However, the method added complication in calibration and is not cost effective. Yu et al (2005) used RANSAC Plane Fitting method to find the disparity of the ground pixel. Similarly, Li et al (2004) developed a

plane fitting algorithm utilizing road features to detect road. These methods are not robust in outdoor terrain and only suitable when the ground is visibly present in large part of the image.

Recently, V-disparity image introduced by Labayrade et al (2003) has become popular for ground plane estimation. This method allows the camera pitch angle to be obtained at the time of image acquisition. By acquiring the camera pitch angle real-time, the ground plane can be estimated robustly and it is suitable to be applied to uneven ground terrain. It utilized the disparity data of stereo camera mapped into vertical image and the resulting ground plane can be estimated by extracting the ground correlation line.

4.2.1 Methods using V-Disparity by Labayrade et al (2002)

After the extraction of disparity image from the stereo image pair, the primary task is to extract features available from the disparity image. One of the famous features is the V-disparity concept introduced by Labayrade et al (2002). Usually, it is used to determine the ground truth of a scene and also to isolate obstacles based on the disparity data.

V-disparity image is based on disparity map($I_{V\Delta}$) obtained from the stereo camera. The V-disparity image is a three-dimensional image which contains the abscissa-axis, ordinates-axis and intensity-axis. The abscissa-axis (d) plots the disparities which the correlation has been computed. The ordinates-axis (V) plots the image row number and the intensity-axis set to be

proportional to measured correlation. Figure 4.1 shows a stereo image and its corresponding V-disparity image where the abscissa-axis (d) is built by accumulation the pixels of the same disparity from disparity map ($I_{V\Delta}$) along the ordinates-axis (V). In other words, V-disparity image is the disparity-based histogram where the accumulation of disparity of the same value projected onto ordinates-axis (V).

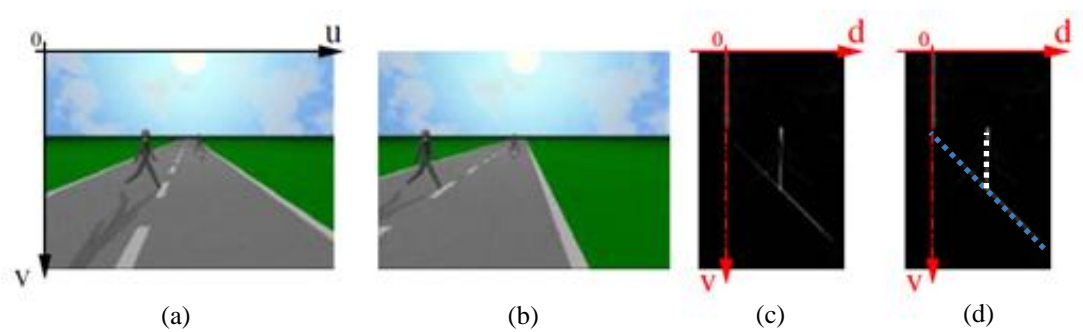


Figure 4.1: Sample stereo image and its V-disparity image frame of reference (Labayrade, Aubert, & Tarel, 2002). (a) Stereo image (left). (b) Stereo image (right). (c) V-disparity image. (d) V-disparity image with ground correlation line (blue line) and obstacle profile (white line).

Detecting objects and obstacles is a question of extraction of features corresponds to the objects in the scene in V-disparity. Usually, the objects and obstacles in the scene are represented by vertical straight lines in the V-disparity image. Figure 4.1(d) shows the V-disparity image for sample image pair. The blue line corresponds to decreasing disparity where it represents the ground. Assuming that the ground plane is flat, the disparity will be lesser when it is further away from the stereo camera. The vertical white line

represents the human (obstacle) as the disparity pixels for the human are of the same value over the v-plane. Next sub-chapter will discuss on how the ground plane profile and obstacles profiles appear in rainforest scene.

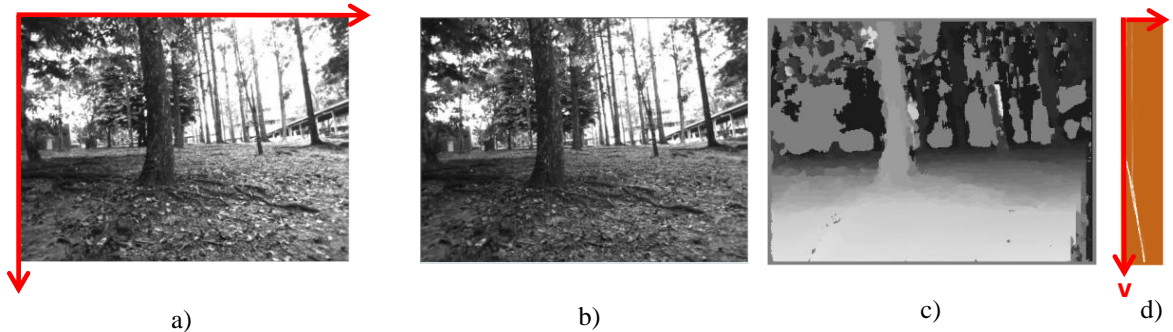


Figure 4.2: Stereo image pair, disparity image and V-disparity image. (a) Left image. (b) Right Image. (c) Disparity Image. (d) V-disparity image.

Figure 4.2 shows an example of image pair, disparity image and its corresponding V-disparity. It can be seen that the trees in the images are represented by vertical segment in the center of the V-disparity image. The tree surface has almost similar distance from the camera, thus the tree has constant disparity. The ground in the image pair is represented by the slanted segment in the V-disparity image. This is due to the linear change of disparity along the ground plane. The further the ground plane, the lesser the disparity.

The ground correlation line contains information of the ground plane and also the stereo camera pitch angle at the time of acquisition. The ground plane can be determined from the V-disparity image if the ground correlation line can be extracted. Various methods were used to extract the correlation with the methods differing depending on the features available.

4.2.1.1 Property of V-disparity Image

The extracted ground correlation profile has the same slope regardless of the pitch variation. The ground correlation line will only shift parallel to its own line. Broggi et al (2005) had experimentally proved that the behavior of the ground correlation line during a pitch variation is to oscillate, parallel to itself like in Figure 4.3. In work by Zhao et al (2007), the characteristic was mathematically proven to be valid. With this characteristic, the ground plane correlation line can be extracted as the camera pitch variation negligible effect.

In addition, the ground correlation profile will have the lower slope compared to obstacle profile. This is due to the linearly changing of disparity of the ground plane. Most of the obstacles will have similar value of disparity with respect to the distance from the camera. This characteristic can be used to determine the ground correlation line when there are multiple profiles found in the V-disparity image.

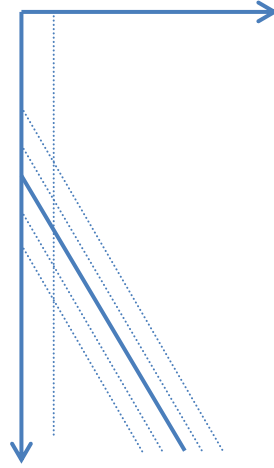


Figure 4.3: Ground correlation lines in V-disparity image. The solid slanted line corresponds to the ground correlation line obtained using the static calibration data, while the dashed slanted lines are the ones expected varying the pitch value. The solid vertical line indicates the 0 disparity value, the dashed vertical line indicates the disparity value of points at infinite distance; they do not overlap due to a slight convergence of cameras optical axes (Labayrade & Aubert, 2003).

4.2.1.2 Ground Profile Extraction from V-Disparity Image

Over the years, there are plenty approaches using V-disparity image (Broggi et al., 2005; Toulminet et al., 2006; Zhao et al., 2007) but with different method of extracting the ground lines. Broggi et al (2005) applied V-disparity image in their work with the intention of extracting road out of the stereo image. Their work is limited to conventional road as they assume the road to be straight line. Zhao et al (2007) applied global correlation method and

Labayrade (2002, 2003) uses Hough transform to extract the ground correlation line. Most of the work are concentrated on-road or semi-structured terrain and in this thesis, the V-disparity image to estimate the ground plane with different kind of constrained. The following sub-section will discusses on the challenges in detection of ground plane in rainforest terrain.

4.2.1.3 Challenges of Ground Plane Estimation in Rainforest Terrain

The application of V-disparity image found in literature is rather limited to on-road or off-road with clear path. The approaches in Section 4.2.1.2 rely heavily on distinct road feature such as lane marking and clear road path. The assumption made on the road profile in V-disparity image is that a straight line correlation can be found. Zhao et al (2007) method of using global correlation method can achieved better result with the assumption that the road near the obstacle is clear of obstacles. In rainforest terrain, V-disparity method holds a great advantage in such a way that the camera position can ground plane can be estimated during the time the image is acquired. However, the assumptions made in previous approaches in estimating the ground correlation line are not suitable to estimate the plane in rainforest terrain.

Previous approaches (as described in Section 4.2.1.2) assume that the ground plane occupies large portion of the image. This scenario is not applicable to rainforest terrain as the route for the vehicle to traverse may be a small area. Thus, in the case of where the ground plane occupies just a small part or multiple parts of the image, there is no sufficient ground pixels to

extract the ground correlation line. Zhao et al (2007) method used higher correlation window cannot be applied in tropical terrain as the assumption made is that the ground plane near the vehicle is clear of obstacles. The assumption made is not applicable to rainforest terrain as obstacles may be present near the vehicle. By using higher correlation window, it also assumed that depth is equal for all pixels in the correlation window which is not the case in our terrain consideration as the depth may contain discontinuities. The discontinuities may be a major hazard as it can be a negative obstacle that the stereo camera cannot detect. Based on the problem statements, it is necessary to devise one more features that can be used to determine the ground plane in tropical terrain.

4.3 Feature Extraction

In this section, the V-disparity image and color feature of the rainforest terrain are examined. The characteristics and cues representing the ground plane and obstacles in the V-disparity image and color feature are described.

4.3.1 Analysis on Disparity Image and V-Disparity Image of Rainforest Terrain.

The V-disparity image is obtained from disparity image produced by the image pair. Since our proposed method will utilize V-disparity image to detect the ground plane, it is important to highlight the difficulties in using V-disparity as a feature. In this section, the behaviors of ground plane and

obstacles profiles in V-disparity image are presented. The motivation is to study the characteristic of the features in order to determine suitable feature to be used in task specific processing.

Figure 4.4 shows a few sample images of outdoor scene with different complexity where the complexity level is easier in the first image pair and increases in the subsequent sample image pair. The considered scenario in this figures include the obstacles and the appearance of the ground plane. Note that the ground and obstacle profile of this sample image is hand-labeled, and the question posed is how to extract this line profile automatically. However, this demonstrates that V-disparity image can be used to extract the ground plane and obstacles.

Figure 4.4 (a) shows an outdoor image where the ground plane dominates the whole image. The correspondence V-disparity image illustrates strong profile of the ground plane and clear obstacles profile.

Figure 4.4 (b) shows more complicated scene where the ground plane appears in large portion of the image but with closer obstacles to the camera. It can be observed in the V-disparity image that the ground profile is fairly apparent. Typical scene in rainforest terrain will appear similar to the sample provided. The obstacles are mainly tree trunk and vertical in nature. Thus, the obstacles can be easily isolated when the ground profile is successfully extracted.



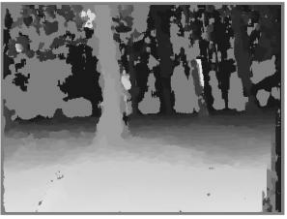




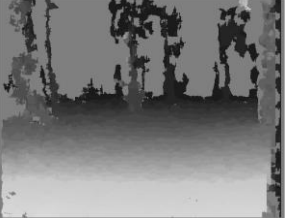




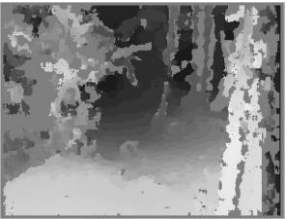


<i>Stereo Image Pair</i>			Disparity Image	V-Disparity	
<i>Left Image</i>	<i>Right Image</i>	Original		Ground Profile	
a					
b					
c					

Figure 4.4: Sample stereo images, disparity images and its V-disparity images frame of reference in rainforest terrain. Note: the ground plane profile and obstacle profiles are hand-labeled.

Figure 4.4 (c) shows a highly complex scene where the ground plane only appears in small part of the image and the scene is populated by obstacles. The V-disparity image does not exhibit ground profile due to the low number of ground surface in the scene.

4.4 Color Feature from Stereo Imaging

In this section, the color feature of a rainforest scene is presented. Color feature is a very fundamental feature available to machine vision and it is

usually considered at the same time with three-dimensional information. This feature can be used to perform color segmentation or clustering and subsequently, terrain classification. To date, the vast information contained in two-dimensional color images is yet to be exploited (Shacklock, Xu, & Wang, 2006). The major problem is to find alternative way to extract the knowledge from the color images and use the information within the algorithm.

In RGB color space, the color appears in primary spectral components of red, green and blue (Gonzalez & Woods, 2010). RGB color space is based on the human eye which consist millions of cones that responsible for color vision. The cones are very sensitive to these three primary colors where approximately 65% of all cones are sensitive to red, 33% are sensitive to green and only 2% are sensitive to blue. Due to these absorption characteristic of the human eye, colors are seen to be variable combinations of the primary colors of red, green and blue. By using this three primary color, most of the colors in the color spectrum can be regenerated.

Although the RGB color space matches with the fact human eyes are strongly sensitive to red, green and blue, it is not practical for human interpretation (Gonzalez & Woods, 2010). For example, one does not refer to the color of an object by referring to the percentage of each primary color representing its color. In term of image processing, the colors will be represented in image planes. Thus modifying one of the color planes will alter the color information in the pixels. In color image processing, one does not wish to alter the color information as it is an important feature to represent an

object. Thus, it can be said RGB color space is only suitable to store and regenerate image but the usage as color description is rather limited (Gonzalez & Woods, 2010).

A human perception distinguishes colors from others based on hue, saturation and brightness information (Gonzalez & Woods, 2010). Hue is an attribute associate with the dominant wavelength in the mixture of light wave where it contains pure color information. Saturation gives the purity of color or the measure of pure color diluted with white light. The combination of both hue and saturation will give the chromaticity. Brightness describes the achromatic attributes which is the intensity. Thus, a color may be characterized by its chromaticity and brightness.

A HSI (hue, saturation, intensity) color model is a model that represents colors in a natural and more intuitive way to human. This model is also commonly known as HSV (hue, saturation, value) color model. As described above, each hue, saturation and intensity (or value) component in the color model represents the hue, saturation and brightness information respectively. This color space decouples the intensity component from the color components in a color image (Gonzalez & Woods, 2010) and the advantage of this separation is that it enables image processing based on color information alone.

4.4.1 Colour Distribution of Rainforest Scene

An initial analysis on the image from camera reveals that that it is difficult to differentiate different classes of obstacles and terrain using color feature alone. The similarity of color made it difficult to differentiate between two different clusters that are closed together. Figure 4.5(a) and Figure 4.5 (b) shows a test image of a rain forest terrain in consideration and its 2D-histogram in hue and saturation. Few fragments of tree trunks, green leaves and ground were extracted and their distributions are viewed in hue-saturation feature space.

The inset image in Figure 4.5 (b) shows the histogram for the whole image. The color distribution for each region is overlaid on top of the overall histogram. It can be observed that the color distribution of tree trunk and ground are close and overlapping whereas the green vegetation can be easily separated. The appearance of an object is dependent on various factors such as illuminant, the reflectance of the object, illumination geometry and also the sensor parameters (Kamegawa, et al.). Figure 4.5 (b) shows the variation of saturation of objects where the saturation are distributed even it belongs to the same object. The variation may be greater when the conditions of the scene change.

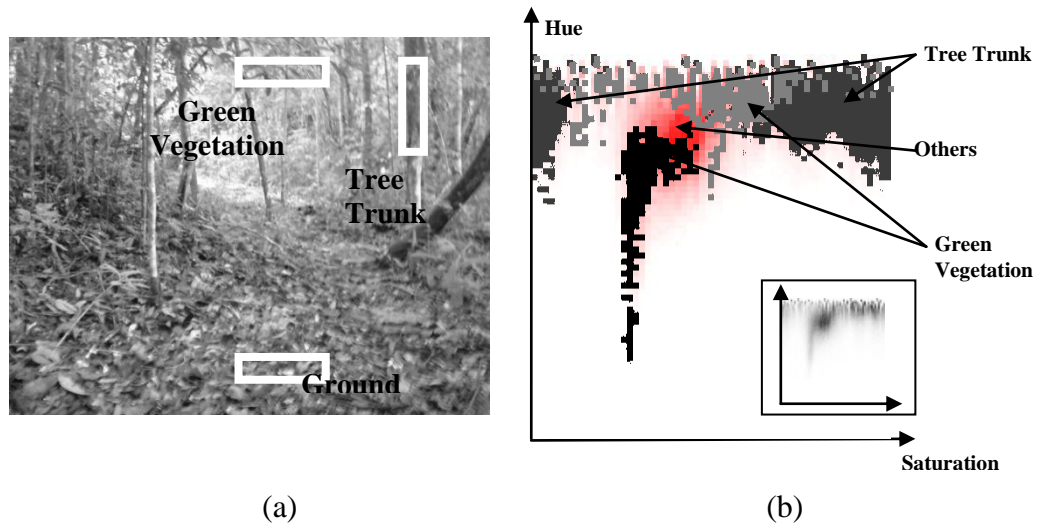


Figure 4.5: 2-dimensional hue-saturation histogram. The distribution of object colors (ground, tree trunk, vegetation and others) are illustrated. (a) The scene. (b) 2-dimensional histogram.

Different type of grounds and trees will have different appearances depending on the illumination, types and texture of soil and tree barks. Generally, both ground and tree barks appear to be similar in color. Thus, there is a need to use multiple cues to differentiate between these two classes. In this project, the disparity feature is used to detect ground plane and any obstacles above the ground plane. It is anticipated that any brown-in-color object or terrain appearing on the ground correlation profile of V-disparity is ground plane. Thus, the ground can be segregated from the tree trunk.

4.4.2 K-means Clustering

Image segmentation is the process of dividing an image of a scene into constituent part and extracting these parts of interest. As our approach in ground plane detection employs segmentation technique as supplement to the V-disparity feature, it is necessary to describe the functionality of the segmentation method used.

In this thesis, K-means clustering is used to cluster the image according to the colors. While there are many techniques that are available, K-means clustering is employed because of its simplicity. The only parameter that needs to be pre-determined is the number of cluster. Due to uncertainty that present in rainforest terrain, it is necessary to have low pre-determined parameter to avoid over-fitting or bias in the clustering process.

To describe the algorithm flow of K-means clustering, let \vec{x} denote the pixel of an image

$$\vec{x} = [x, y, R, G, B] \quad 4.1$$

where x, y are the pixel location and R, G, B are the intensities of the color components at that point.

The K-means clustering method aims to classify the terrain according to the performance index

$$J = \sum_{l=1}^K \sum_{c_l^{(i)}}^n \left\| \vec{x} - u_l^{(i+1)} \right\|^2 \quad 4.2$$

where $c_l^{(i)}$ denotes the set of samples assigned to the cluster l after the i th iteration, and μ_l denotes the mean of the l th cluster. This index measures the sum of the distances of each sample from their respective cluster means.

The algorithm runs as follow:

1. Randomly assign the cluster center location.

$$\mu_1^{(1)}, \mu_2^{(1)}, \dots, \mu_K^{(1)} \quad 4.3$$

2. At the i th iteration assign each pixel, \vec{x} to one of the K clusters according to relation

$$\vec{x} \in c_j^{(i)} \quad \text{if } \|\vec{x} - u_j^i\| < \|\vec{x} - u_l^i\| \quad 4.4$$

For all $l = 1, 2, \dots, K, l \neq j$, where $c_j^{(i)}$ denotes the set of samples whose cluster center is u_j^i .

3. The cluster center mean, u_j^{i+1} is updated where l is the cluster number.
4. If $u_j^{i+1} = u_j^i$ for all l , the algorithm has converged and the algorithm is terminated. Else, go to step 2.

The K-means clustering algorithm requires the number of clusters to be known beforehand. The choice of the number of clusters for this thesis is determined based on a few assumptions and validated with experimental data. Next in this chapter, the developed ground plane detection ground plane detection algorithm is described.

4.4.3 K-means Clustering: Number of Cluster Determination

The major setting in this method is to determine the number of cluster required in order for the proposed method to be working well in rainforest terrain. The motivation of using K-means clustering method is to segment the lower half of the image to possible different regions. However, we do not anticipate over segment or under segment the images.

Ideally, there should be only two classes which are obstacles and non-obstacles. However, due to color ambiguity of different object, the regions in the image cannot be group into just two groups. Thus, it is necessary to determine a universal parameter (number of cluster) that can work in majority of the conditions in rainforest terrain.

The following are the assumption that we employed in determining the number of regions:

1. The image contains the scene which can be divided into two major classes, traversable ground region and obstacle region. Thus the minimum region expected in the image is two.

2. The major colors in rainforest terrain are green, brown and black. Each class may be any of the colors. Assuming that every object color is constant (no color constancy problem), the ideal number of region is six (two major classes multiply by three major color appearance).

However, there may be multiple objects of an object class appearing in the scene; therefore there may be multiple region of an object class. Thus, it is expected the suitable number of region lay around six regions.

No. of region			
No. of Clusters	<5	5-8	>8
2	1.00	0	0
3	1.00	0	0
4	0.30	0.60	0.10
5	0	0.64	0.36
6	0	0.44	0.56
7	0	0.16	0.84
8	0	0	1.00

Table 4.1: The number of regions resulted from the number of cluster settings.

The percentages of the regions are based on 30 samples.

Table 4.1 shows the number of regions in the image using different number of clusters. The percentage shown is based on 30 sample images. Regions with less than 100 pixels are omitted. The number of region anticipated is approximately six, thus it can be seen that the highest percentage of six regions generated when number of clusters is set to five.

4.5 Developed Ground Plane Detection Algorithm

In this thesis, the approach taken to find suitable ground feature is to segment the image into a few salient regions before the V-disparity image is considered. Since the disparity data are associated with color information, the color feature can be used to divide the stereo image into a few region of interest. Each of the regions will generate different V-disparity image and different correlation line. The correlation lines with high gradient will be discarded while the correlation line with lowest gradient will be taken into consideration as ground consideration line. The assumption made is that the ground plane is usually associates with the correlation line with lowest gradient.

Figure 4.6 shows the framework of developed ground plane detection algorithm. Color image pair was obtained from stereo camera. The image pair was then rectified before disparity data was obtained. The colour image was converted to HSI color space where the hue and saturation channel are used together with disparity data for clustering. The value channel is not considered to minimize the effect of uneven brightness across the scene.

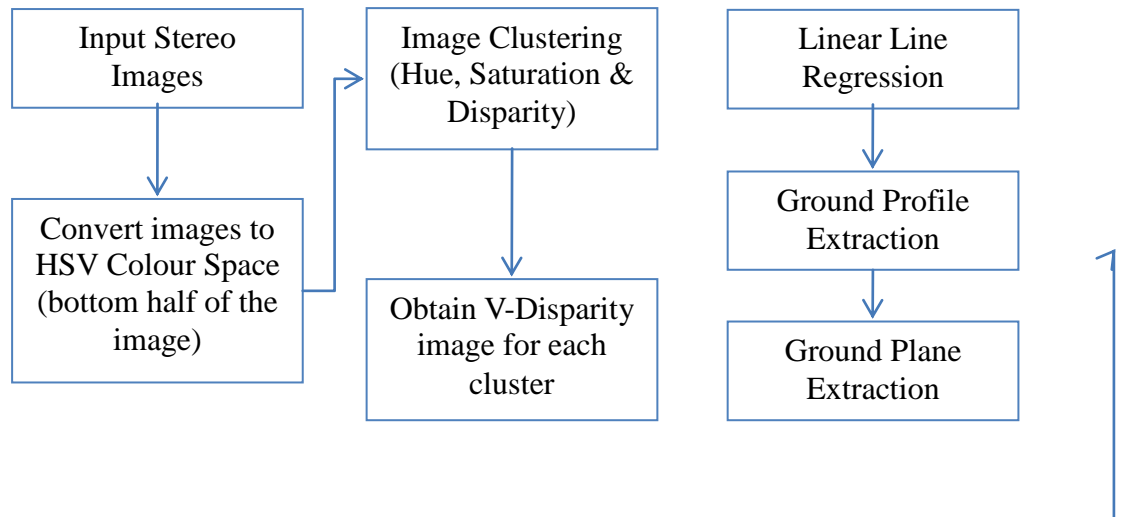


Figure 4.6: Framework of the developed ground plane detection module.

Then, three-dimensional feature consist of hue, saturation and disparity was used to divide the scene into a few regions. Only the bottom half of the image was considered as it is assumed the ground plane will always appear at the distance nearer to the camera. However, once the ground correlation profile is estimated, it will be applied to the whole image to extract the ground plane.

In this work, K-means clustering was used to cluster the bottom half of the image into five (5) clusters. The number of clusters was chosen based on extensive testing as describe in Section 4.4.3 and it was found that five (5) clusters were sufficient to generate suitable candidate lines.

Each of the clusters point was mapped to the V-disparity image and each cluster will generate a candidate ground correlation line. Five color clusters will generate five candidate ground correlation lines. To generate lines corresponding to each of the cluster, a simple linear line regression was used.

The best fit line associate with the n points $(v_1, d_1), (v_2, d_2) \dots (v_n, d_n)$ has the form of

$$v = md + c \quad 4.5$$

where v and d correspond to row of the V-disparity image and disparity accumulation. The slope, m and v-intercept, c are given as

$$m = \frac{n \sum vd - (\sum v)(\sum d)}{n \sum (d^2) - (\sum d)^2} \quad 4.6$$

and

$$c = \frac{\sum v - m \sum d}{n} \quad 4.7$$

Each of the regions will generate different V-disparity image and different correlation line. The correlation lines with high gradient will be discarded while the correlation line with lowest gradient will be taken into consideration as ground consideration line. The choice is made based on the V-disparity property discussed in Section 4.2.1.1, where ground plane usually associates with the correlation line with lowest gradient.

The selected ground correlation profile was then applied to the V-disparity image of the whole image to extract the ground pixels. Figure 4.7 (c) shows the selected ground correlation line mapped onto the V-disparity image.

The disparity that appears above line and 10 pixels below the ground correlation will be considered as ground pixel.

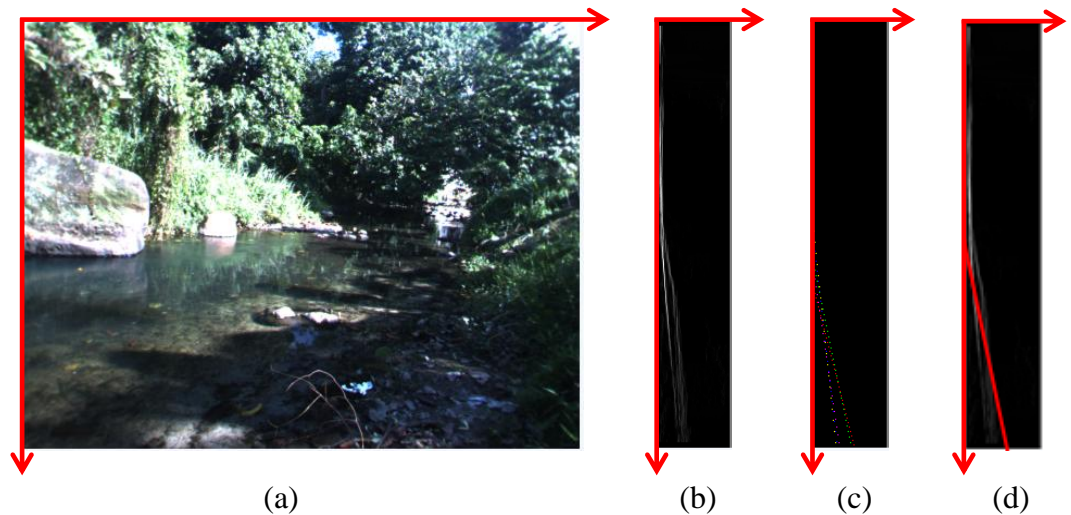


Figure 4.7: (a) Image of the Scene. (b) V-disparity image. (c) Candidates for ground correlation line. (d) Extracted ground correlation line.

Figure 4.7(a) shows an example of scene in the consideration. The V-disparity image is illustrated in Figure 4.7(b). The K-Mean clustering algorithm is implemented to segment into several regions. The feature used in the clustering is hue, saturation and disparity. The K-means clustering algorithm is very sensate to brightness or the image, thus the brightness feature is not used and replaced with disparity feature. The motivation of using disparity as a feature in K-means clustering is to differentiate between two different objects with similar color. This clustering step is applied only on the lower part of the image. After the color clustering, the candidate lines were generated as illustrated in Figure 4.7(c).

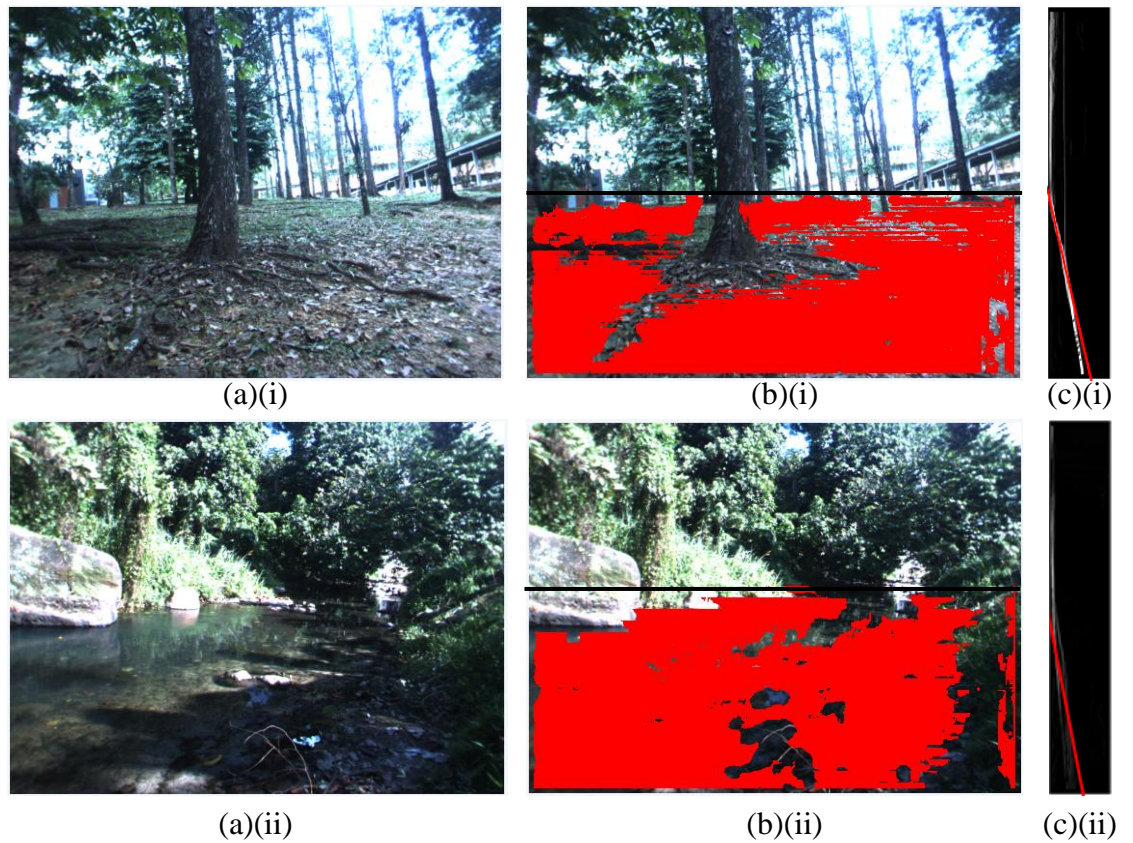


Figure 4.8: Sample result of ground plane estimation. (a) Sample scene. (b) Mapped ground pixels in red. (c) Ground correlation profile in V-disparity image.

The correlation lines with high gradient will be discarded while the correlation line with lowest gradient is chosen to be ground correlation line (Figure 4.7(d)). The ground correlation profile is visualized in red. From the ground correlation profile, the disparity points correspond to the ground plane can be mapped.

4.6 Experimental Results and Discussions

Now after the V-disparity image, K-means Clustering and proposed ground detection algorithm were described above; the experimental results of the developed method are done in this section.

The algorithm is tested with 90 samples of rainforest terrain. The test images cover usual simple unstructured scene, rainforest scene with moderate complexity and complex scene. Thirty (30) samples are used for each type of scene. The image size is 512x384.

The performance of our ground plane detection algorithm is quantified by discrete classifier model. In this thesis, there are true class and hypothesized classes which represent the actual situation of the scene and predicted situation by developed algorithm respectively.

Figure 4.9 illustrates the performance classifier used. The true classes of interest in this developed module is ground and not ground. Given that there are two true classes, there are four possible outcomes. If the object is ground and its predicted class is YES, it is classified as true positive (TP). If the predicted class is NO, it is classified as false negative (FN). Similarly, if the object is not ground and its predicted class is NO, it is classified as true negative (TN). False positive (FP) occurs when the true class is not ground and the predicted class is ground (YES).

		Ground	Not Ground
yes	TP	FP	
no	FN	TN	

Figure 4.9: The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.

Based on Figure 4.9, it can be seen that the performance of developed module is good when the true positive and true negative result is high as they represent the correct results. Both false negative and false positive are not desirable as it represent wrong classification of the scene.

However, in this module, the false positive result holds higher significance as it represents wrong classification that will harm the autonomous vehicle. If the true class of the object is not ground, it may be an obstacle that can lead to collision. Thus, it is necessary for the false positive classification to be kept as low as possible.

The following subsections discuss the result of the developed ground detection module applied in unstructured terrain.

4.6.1 Simple Unstructured Terrain Experiment

First, the ground detection algorithm is tested on the simple unstructured terrain image sequence. We define simple unstructured terrain as a relatively flat ground and little obstacles around the obstacles.

As discussed in Section 4.3.1, for the scene where ground plane occupy majority of the image, the ground correlation line in V-disparity image is relatively clear. The ground correlation line is fairly straight and it is easier to extract as the disparity data in V-disparity image mostly represent the ground plane. Low presence of obstacles disparity data in the V-disparity image space directly contributed to lower noise, thus making the ground correlation profile extraction less affected by noise.

Figure 4.10(a)(i) and Figure 4.10(b)(i) depict two sample images of simple unstructured terrain. It can be seen in the images that there is no drastic change of ground slope and the obstacles are relatively scarce. There are a lot of areas that could be traversed by the autonomous vehicle. Figure 4.10(a)(ii) and Figure 4.10 (b)(ii) show the ground plane detected using developed module. It can be observed that developed module managed to identify the ground region in the scene.

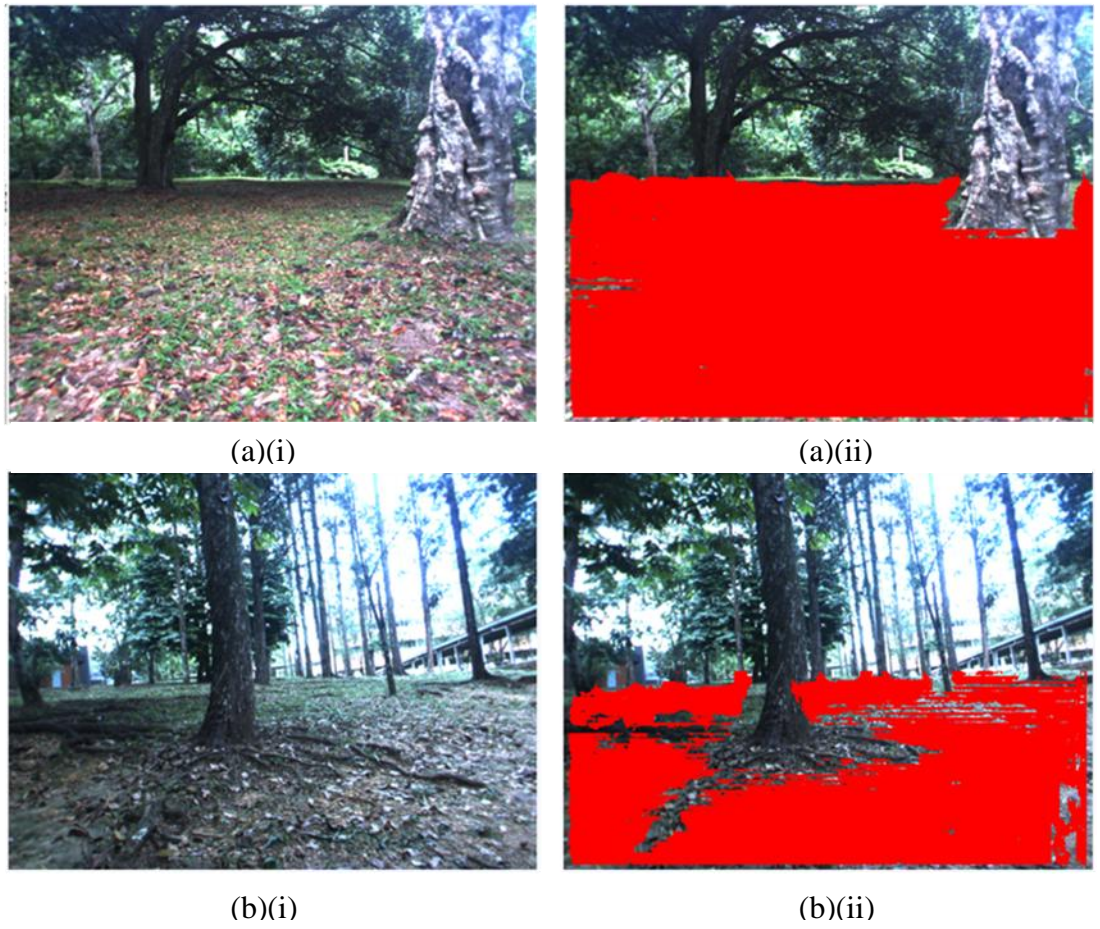


Figure 4.10: Sample results of detected ground plane in rainforest terrain for simple unstructured terrain. (a)(i) and (b)(i) show the sample images and the detected ground planes are shown in (a)(ii) and (b)(ii) respectively.

Figure 4.11 shows the average error of detected ground plane versus the distance from the camera. The error is calculated based on comparison between detected ground pixel and hand labeled ground pixel. It can be seen that the errors are below 7 percent for the first eight meters. The errors start to increase after 8 meters. This is due to the stereo camera minimum disparity setting which limit the distance of object detectable by the stereo camera. At further distance, disparity data for objects obtained from the stereo camera is low in

resolution, thus making it difficult to distinguish between objects. There errors occurred at the boundary of objects.

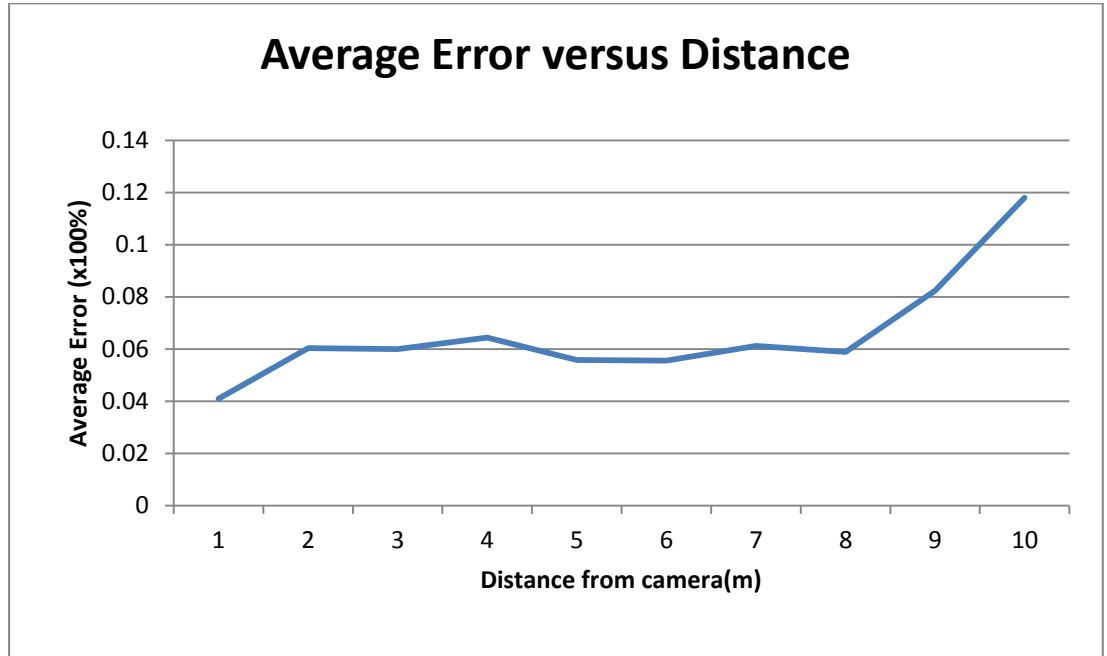


Figure 4.11: Average ground plane error versus distance for ground plane detection in simple unstructured terrain. The number of samples used is thirty.

Figure 4.12 shows the performance of developed module in details. The true positive and true negative outcomes are maintained above 90 percent within the first 10 meters. The false positive and false negative and false negative outcomes are below 10 percent.

The false negative occurs when our developed algorithm wrongly classify a region as non-traversable area when the region is a traversable region. This type of error does not bring hazard to the vehicle as the error will not lead the vehicle to the obstacles.

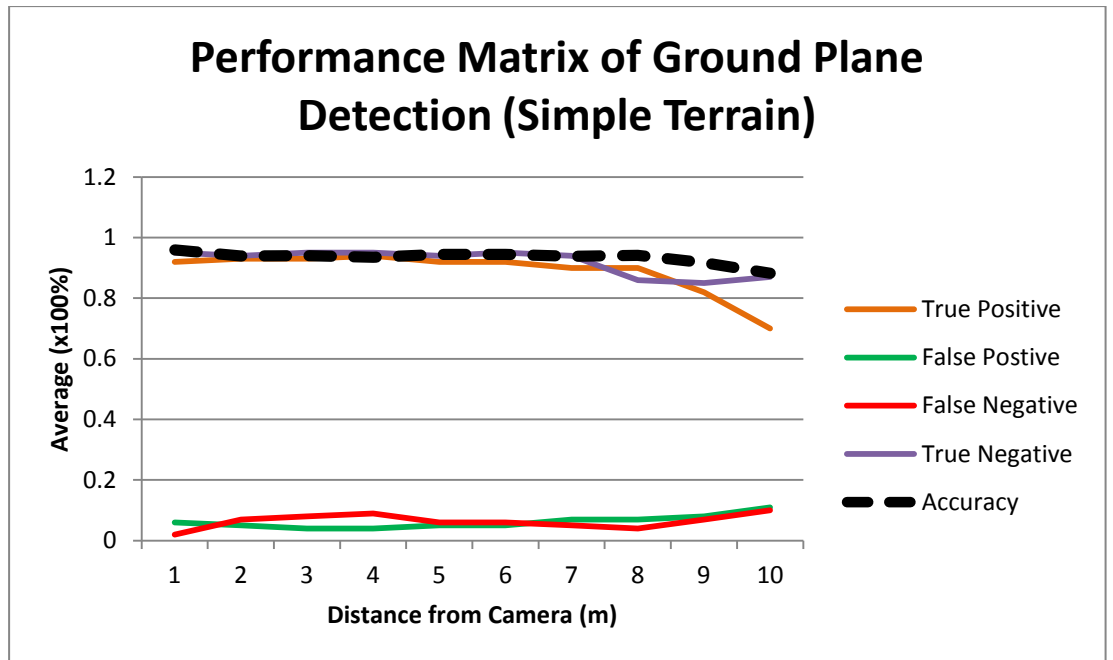


Figure 4.12: Performance matrix of the ground plane detection module in simple unstructured terrain. The number of samples used is thirty (30).

The false positive outcome is not desirable as it directly affect the safety of the vehicle itself. This wrong classification will bring hazard to the vehicle as it will mislead the vehicle. In our developed module, the false positive outcome is kept low throughout the 10 meters range from the camera. The error usually occurs at the edge of detected ground region which is near the boundary of ground plane and obstacles.

4.6.2 Moderate Terrain Experiment

Next, the ground plane detection algorithm is tested on typical rainforest scene with moderate complexity. As discussed in Section 4.3.1, the

ground correlation profile for rainforest scene is fairly apparent in the V-disparity image. The obstacles are mainly tree trunk and vertical in nature. Thus, the obstacles can be easily isolated when the ground profile is successfully extracted.

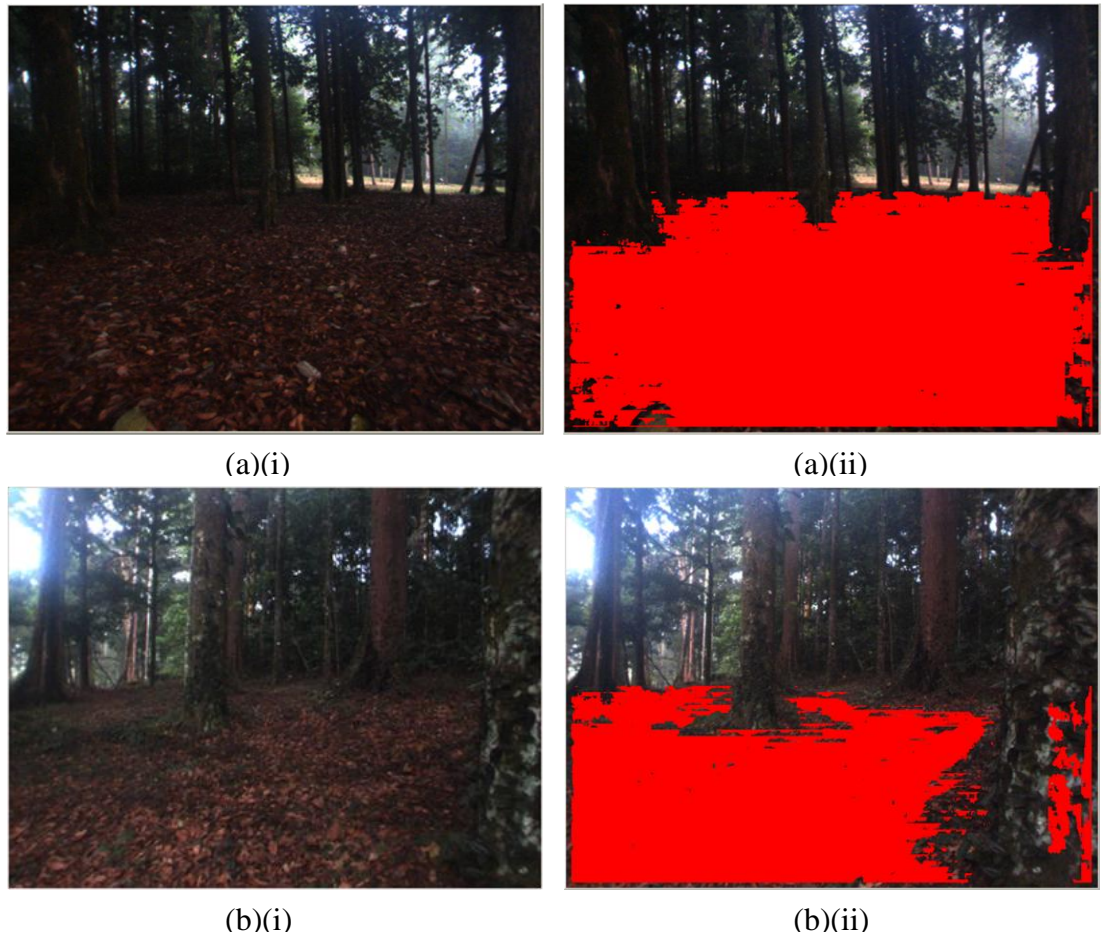


Figure 4.13: Sample results of detected ground plane in rainforest terrain for rainforest terrain.

Figure 4.13(a)(i) and Figure 4.13(b)(i) depict two sample images of typical rainforest scene. The ground plane is relatively flat throughout of the scene. There is no drastic change to the ground slope. In rainforest terrain,

there are few green vegetation and compressible vegetation. The obstacles are mainly tree trunks which are tall, straight and vertical. The lack of green vegetation on the ground is due to the canopies which prevent sun light from reaching the ground. There are a lot of areas that could be traversed by the autonomous vehicle. Figure 4.13(a)(ii) and Figure 4.13(b)(ii) show the ground plane detected using developed module. It can be observed that developed module managed to identify the ground region in the scene.

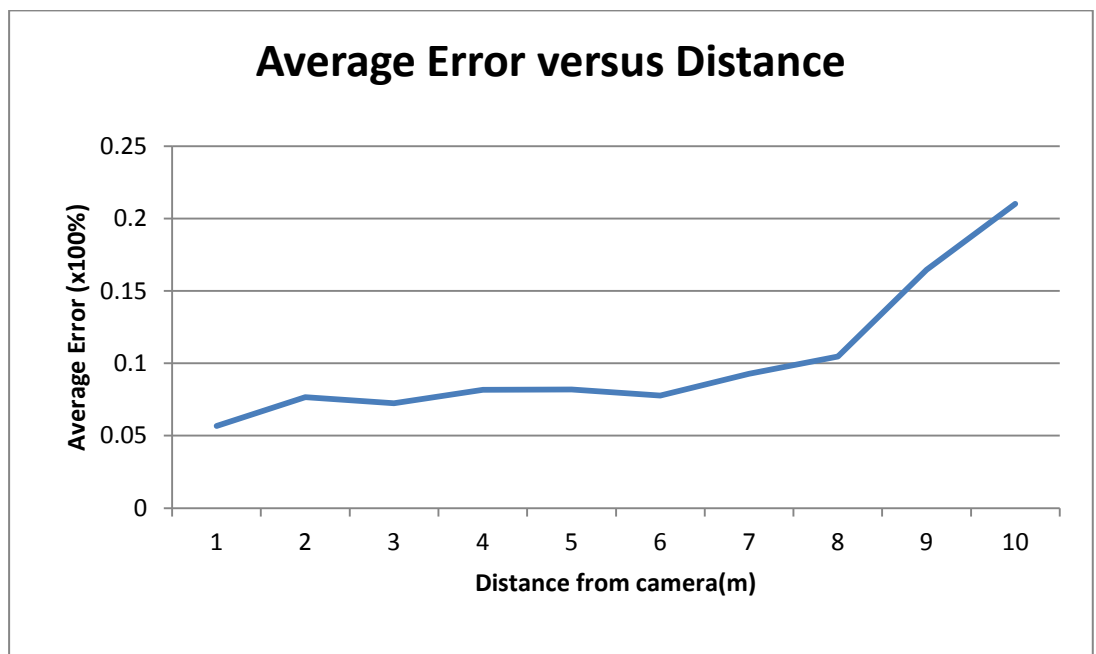


Figure 4.14: Average ground plane error versus distance in rainforest terrain.

The number of samples used is thirty (30).

Figure 4.14 shows the average error of detected ground plane versus the distance from the camera. It can be seen that the errors are below 10 percent for the first eight meters. The errors start to increase after 8 meters. The cause of error is similar to application in simple unstructured terrain where it is due to

the stereo camera minimum disparity setting. The stereo camera minimum disparity setting limits the distance of object detectable by the stereo camera. These errors occurred at the boundary of objects.

The errors also caused by the environment itself where in rainforest terrain, the environment is usually low in brightness. This is because the scene is covered by the canopies which prevent the sunlight from illuminating the scene. Under this circumstance, image with low contrast will be obtained. This will contribute to difficulties to differentiate between two regions. Figure 4.13(a)(i) depicts this scenario where the darker regions does not show clear boundaries between the ground plane and the tree trunks.

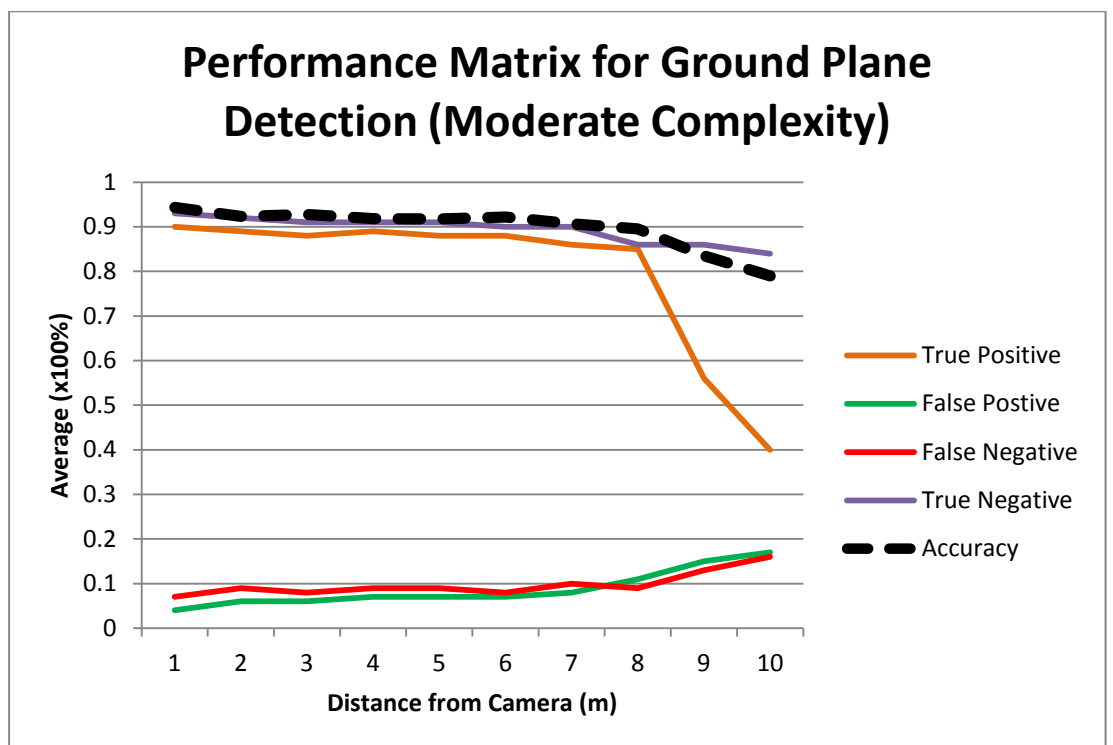


Figure 4.15: Performance matrix of the ground plane detection module for terrain with moderate complexity. The number of samples used is thirty (30).

Figure 4.15 shows the performance of developed module in rainforest terrain in details. The true positive and true negative outcomes are maintained above 90 percent within the first 10 meters. The false positive and false negative and false negative outcomes are below 10 percent. The performance of the developed module in rainforest terrain is similar to the performance in simple unstructured terrain where the false negative and false positive results are kept low at acceptable level.

4.6.3 Complex Terrain Experiment

After the developed algorithm was tested in rainforest terrain, it is tested in more complicated terrain. Although this terrain is not common in rainforest terrain, it is interesting to experiment the algorithm in complex terrain.

In this thesis, we define complex terrain as terrain with minimal ground region, presence of compressible vegetation and large number of obstacles in the scene. Note that in our experiment, it is not in the consideration to classify whether a possible obstacles to be compressible or non-compressible. It is our interest to detect possible traversable region by the vehicle.

In Section 4.3.1, it was mentioned that the ground correlation profile for complex terrain is not prominent in the V-disparity image. This is due to minimal appearance of the ground plane in the image. Thus, the disparity data representing the ground plane in the V-disparity image is minimal compared to

other object. The low count in the V-disparity image makes it hard to extract accurate ground correlation profile.

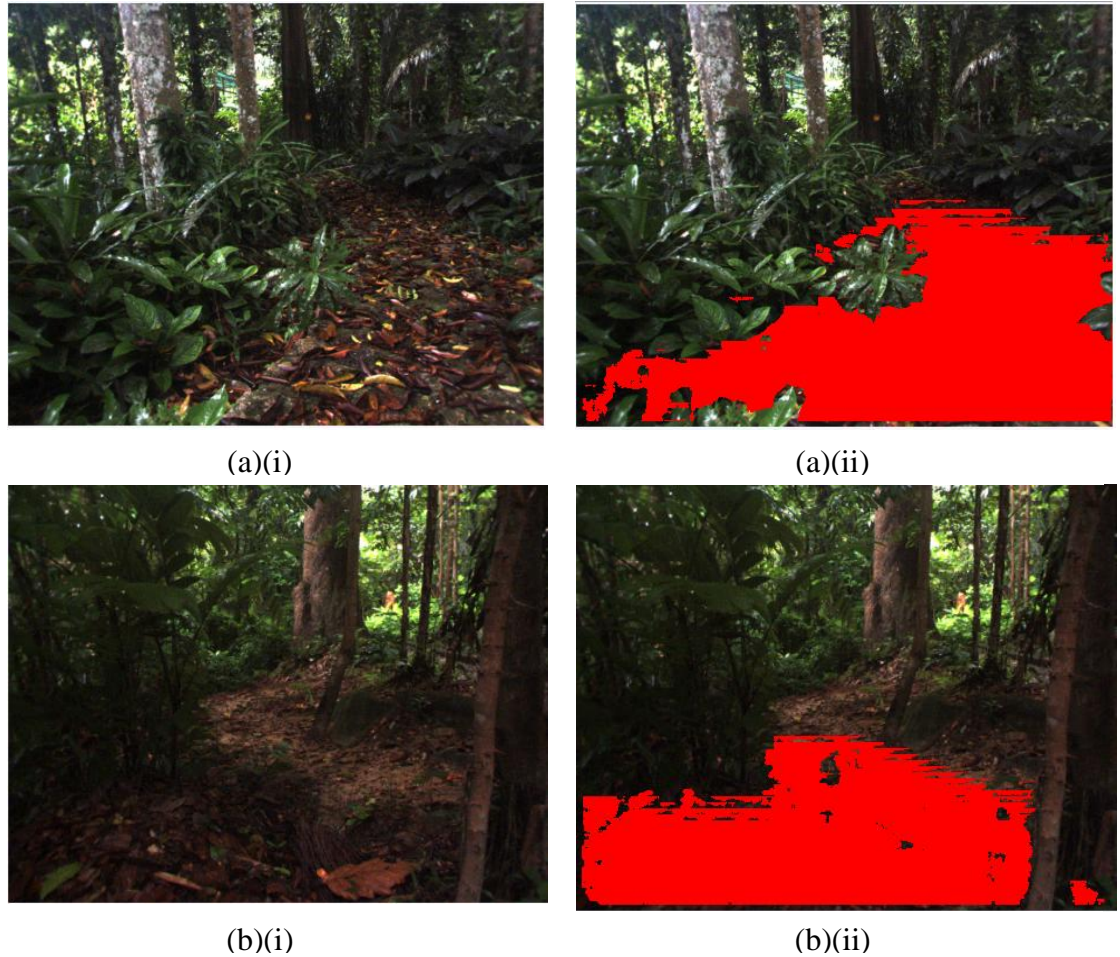


Figure 4.16: Sample results of detected ground plane in rainforest terrain for complex terrain.

Figure 4.16(a)(i) and Figure 4.16 (b)(i) depict two sample images of complex terrain used in the experiment. The ground plane appearance is minor while the obstacles are very close to the camera. The ground slope is not constant throughout the scene. In Figure 4.16 (a)(i), it can be seen that the

ground plane is relatively flat in the scene. However, the ground plane is not the major object in the image and there are obstacles close to the camera. The detected ground plane in Figure 4.16 (a)(ii) shows some error in lower left of the image where some of the green vegetation are classified as ground plane. This is due to the green vegetation height is below the threshold set by developed algorithm. However, the region may be traversed as it is low in height.

In Figure 4.16 (b)(i), the ground plane is uneven throughout the scene, where the ground plane is going downhill. The developed algorithm fails to recognize the ground plane (at the center of the image) as the ground profile cannot be represented in a straight ground profile. When the ground is going downhill, the ground region profile deviates from the detected ground profile. In this thesis, the region will not be considered as ground plane and it is classified as undefined region. However, the algorithm managed to detect the region as the camera moved closer.

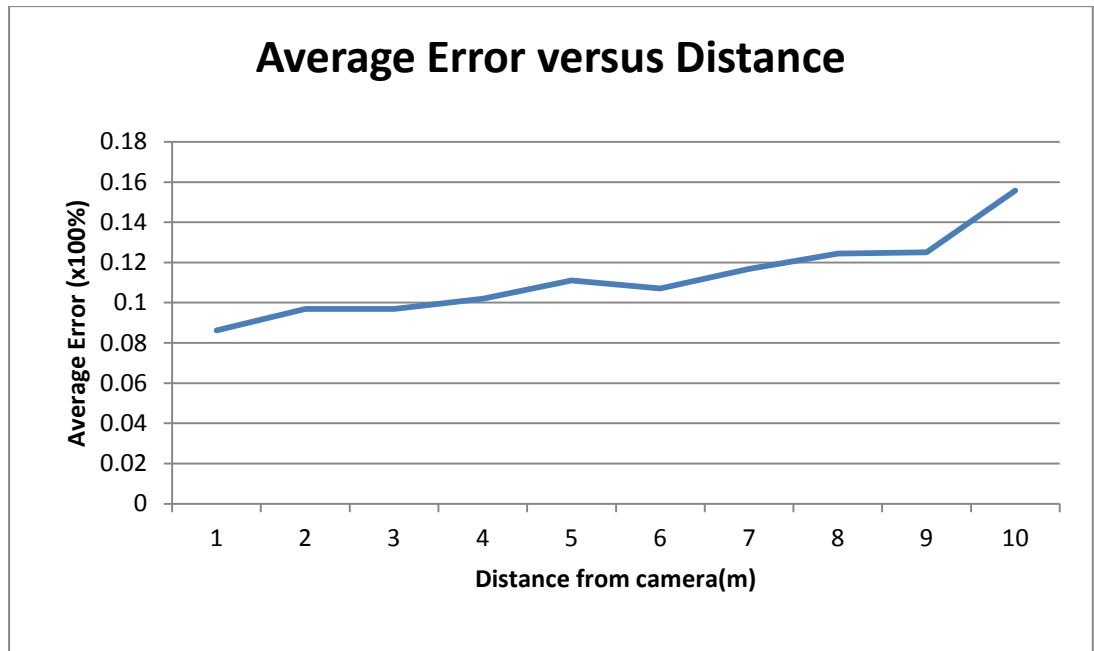


Figure 4.17: Average ground plane error versus distance in complex terrain.

The number of samples used is thirty (30).

Figure 4.17 shows the average error of detected ground plane versus the distance from the camera. It can be seen the error occurred in complex terrain is higher compared to simple unstructured terrain and rainforest terrain. The error is due to presence of various obstacles which are low in height such as compressible green vegetation. Since the developed algorithm does not use machine learning to classify the type of obstacles, any object appear between the ground plane profile tolerance will be considered as ground plane. Thus, the average error for the application in complex terrain is higher. However, we manage to maintain the error to be below 15 percent for near range ground plane detection. Similarly to previous experiment, the errors occurred at the boundary of detected ground plane.

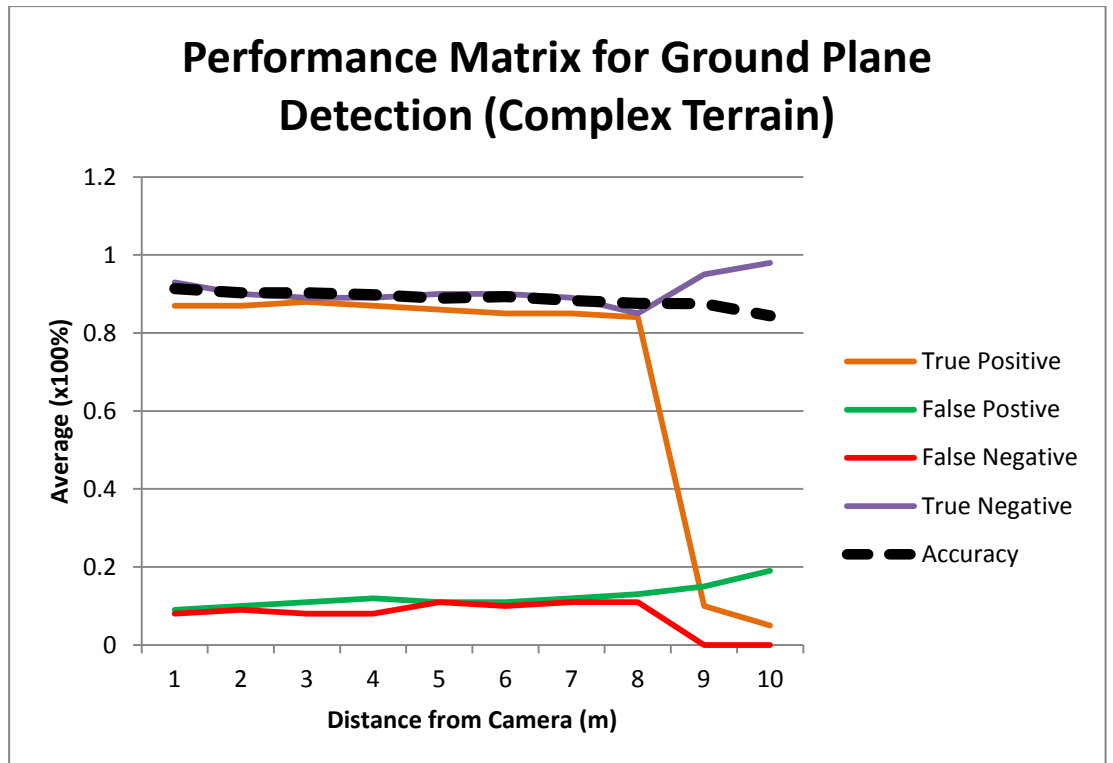


Figure 4.18: Performance matrix of the ground plane detection module in complex terrain. The number of samples used is thirty (30).

Figure 4.18 shows the performance of developed module applied in complex terrain in details. The true positive and true negative outcomes are maintained above 85 percent within the first 10 meters. The true positive outcome drops sharply due to the samples used in this experiment are mainly sloppy terrain. The developed algorithm is not able to detect ground plane with non-straight line ground profile. However, this error can be eliminated once the camera is moved closer to the region.

The false positive and false negative outcomes are below 20 percent. The false positive outcomes are mainly caused by tree trunk roots and

compressible vegetation. In rainforest terrain, the tree root may appear on top of the ground and may be considered obstacles if the height of the root is above threshold. However, in the hand-labeled ground plane, all the tree roots are considered as obstacles which include the roots that can be run over. This contributes to the high false positive outcome in this experiment.

4.7 Summary

The algorithm produced good results in the condition in most of the rainforest terrain condition. It was able to determine the ground plane of the scene and mapped the corresponding ground pixels to the image.

Based on results of three experiments performed in simple unstructured terrain, rainforest terrain and complex terrain, the average result is presented in Figure 4.19 and Figure 4.20. Based on the results, the average error over 90 samples is less than 10 percent within 8 meter from the camera.

We managed to maintain the average accuracy of 90 percent over the three tests. The false positive outcome is very critical as this condition is hazardous to the vehicle. In this thesis, the developed algorithm managed to maintain the false positive alarm below 10 percent for most of the conditions.

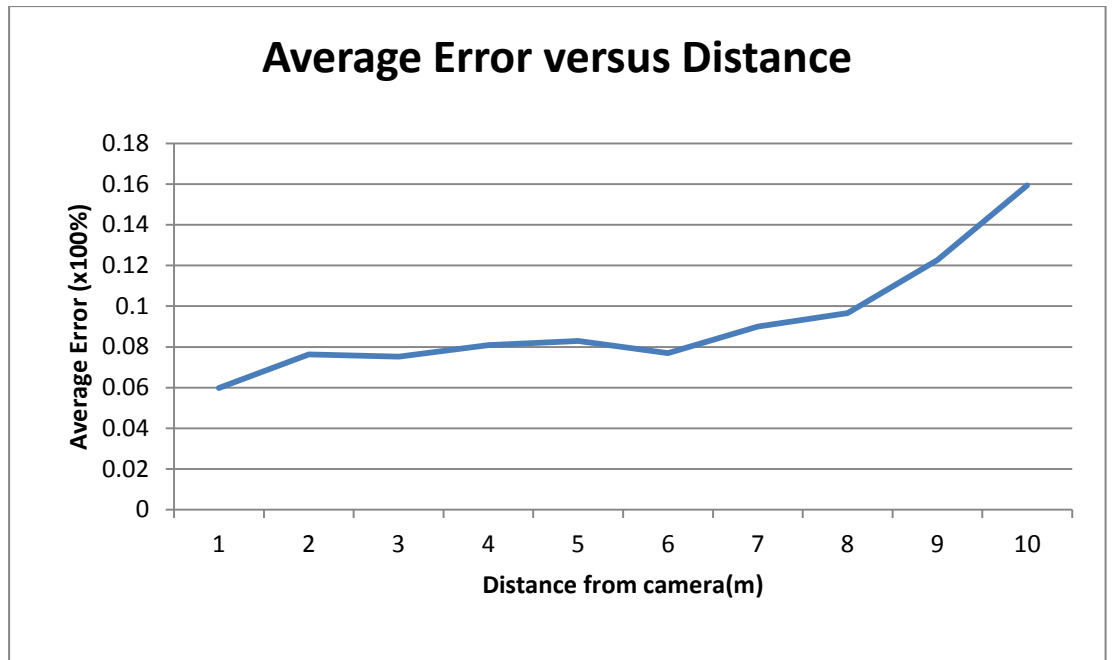


Figure 4.19: Average ground plane detection error versus distance. The number of samples used is ninety (90).

Based on observation on the detected ground plane, the false positive alarm usually occurs at the edge of the ground plane. The boundary of the ground region and obstacles are not clear due to similarities of color between some of the obstacles and ground plane. In rainforest terrain, the ground region may be similar to the color of the tree trunks. The color similarities made the ground plane to be inseparable using color cue. However, the ground region and obstacles can be distinguished using disparity data. The error occurs at the boundary as there is continuity of disparity data of ground plane into the disparity data at edge of the obstacles before the disparity data of the obstacles become constant.

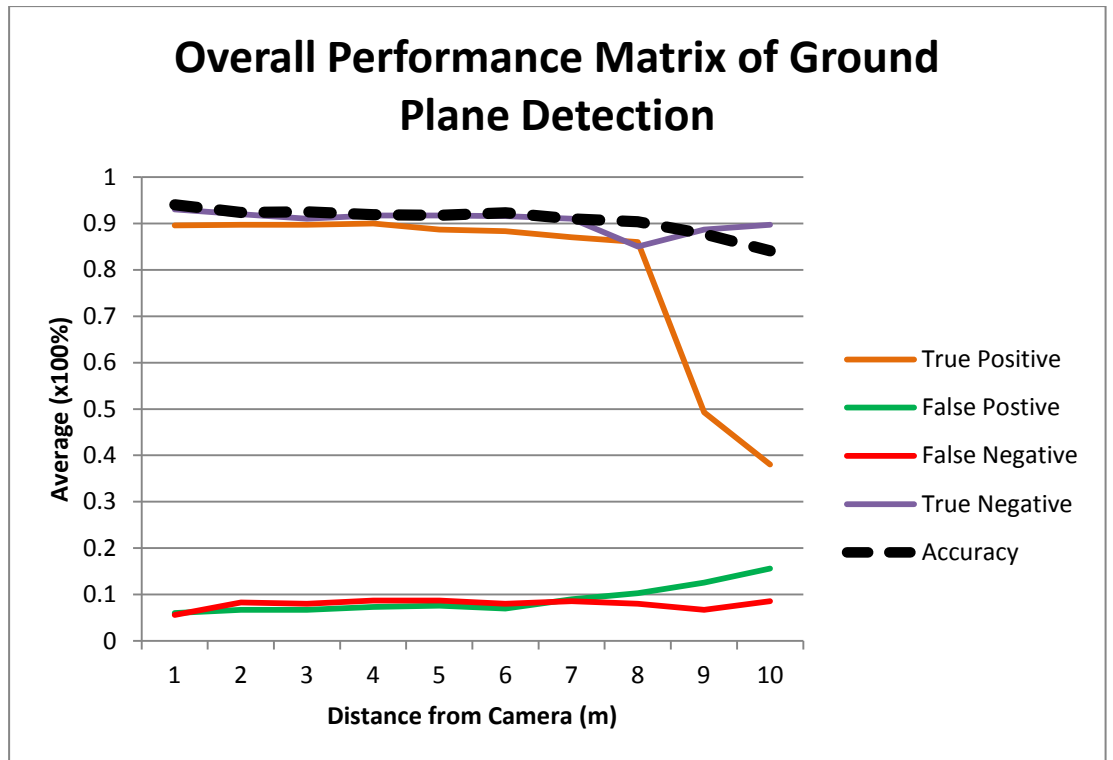


Figure 4.20: Overall performance matrix of the ground plane detection module. The number of samples used is ninety (90).

The unstructured natures of the terrain make the classification of the terrain harder. It is difficult to segment the region with clear boundary separation. However, our developed module can be used together with vehicle path planner with the confidence level at the edge detected ground plane set to be lower compared to the middle of the detected region. With the confidence level, the decision can be made to traverse through the ground region with higher confidence level. Consequently, the risk of false positive outcome can be minimized.

In the case of ground region with downhill profile, the developed algorithm fail to detect the region until the camera is closer to the region. This is due to the fact in complicated terrain; there are multiple ground profiles to describe the ground region. However, this error can be corrected once the stereo camera is closer to the region where the pose of the camera is aligned to the ground plane.

This algorithm can be used in real time application as it is not computationally heavy. On a processing system (Intel Core 2 Duo CPU 2.2 Ghz processor and 2 Gb of RAM), a total computational time of ~60ms was obtained on 512x384 pixels images. This permits the algorithm to be devised and used together with other obstacle detection algorithm.

CHAPTER 5

TREE TRUNK DETECTION

5.1 Introduction

This section discusses about tree trunk detection module based on disparity feature, edge feature and tree verticality cue. The detection of tree trunk in the rainforest scene is an important task due to high occurrence of these obstacles and it may causes damage to the autonomous vehicle. The task is accomplished by utilizing multiple features from the stereo camera and tree characteristic cues of the tree trunk in the rainforest terrain.

To begin, several characteristics of tree trunk that can be used to isolate the tree trunk from the rainforest scene are listed. The following are the typical appearance of tree trunks in rainforest terrain:

1. Verticality – Tree trunks are usually tall and vertical covered by canopy at the top. Therefore, the edges of the tree trunks are usually vertical or near vertical. The disparity information of the tree trunks will have large continuities across the rows of the image.
2. Tree bark geometric appearance – Typically, the background of tree trunks in rainforest scene is rather bare due to the presence of the canopy limiting

the growth of smaller vegetation. Thus, the change of disparity information at the edge of the tree trunks and background is rather abrupt.

Using these characteristics, the tree trunk detection module is formulated to detect the vertical and near vertical tree trunks. The state-of-the-art of obstacle detection is discussed before going into details of tree trunk detection. Then, detailed descriptions of the proposed method and results are presented.

Section 5.2 discusses on the current approach of the obstacle detection in general. Most of the works done are mainly on wide-ranging obstacles which are present in rainforest terrain. However, it is necessary to discuss the work done as tree trunks is a category of obstacles. Section 5.3 describes the situation under our consideration. Section 5.4 describes the operators or methods used in the developed algorithm. Section 5.5 present the algorithm and steps to detect the tree trunks based on stereo disparity and edges. Section 5.6 presents the results of the developed method and finally the summary is presented in Section 5.7.

5.2 Overview of Obstacle Detection

One of the earliest obstacle detection algorithms was done by Talukder (2003). The authors used 3-D obstacle detection algorithm to locate and segment obstacles in the scene for autonomous terrain vehicle navigation. The scene is segmented into clusters, where each cluster identifies an isolated

obstacle in 3D space. Then, rule-based classification using 3D geometrical measures derived for each segmented obstacle is then used to reject obstacles that are small in volume or lower than threshold height. The work used solely geometrical data to detect obstacles that is higher than certain slope and height threshold. While the developed method managed to detect most of the possible obstacle in place, it does not provide any information regarding the traversability.

Approach by Hebert (2003) tried to segment the scene using the point clouds acquired from LADAR. The work focused primarily on segmentation of point clouds into classes of points corresponding to surfaces and group similar points into consensus regions corresponding to large pieces of surfaces. The terrain types include vegetation, rocks, wires, and small-diameter objects. The challenges associated with this approach are the computational cost and the resolution of the sensor. The computational cost is relatively higher compared to other vision techniques. The resolution of the sensor of the LADAR sensor is not sufficient to capture the terrain and obstacle shape in a single run.

Manduchi et al (2005) approach to detect obstacles is based on multiple sensors. The elevation map is generated directly from stereo camera and LADAR sensor information. Analysis of one-dimensional profiles corresponding to range values is analyzed. These measurements sample the trace left by the visible surface on the slicing planes defined by each pixel column and the focal point of the camera in each pixel column in the image

plane. The slope of the one-dimensional range profile with respect to the horizontal plane can be used to detect obstacles.

5.2.1 Method by Huertas et al (2005)

This paper describes a stereo-based tree traversability algorithm implemented and tested on a robotic vehicle under the DARPA PerceptOR program. Edge detection is applied to the left view of the stereo pair to extract long and vertical edge contours. A search step matches anti-parallel line pairs that correspond to the boundaries of individual trees. Stereo ranging is performed and the range data within trunk fragments are averaged. The diameters of each tree is then estimated, based on the average range to the tree, the focal length of the camera, and the distance in pixels between matched contour lines.

The ability to detect and estimate the diameters of trees depends on the ability of the edge detector to resolve the tree boundaries, and the ability of the stereo algorithm to produce range information. Under some illumination conditions, part of the same tree trunk appears bright and part appears dark. This affects the performance of the edge detector where some of the edges in the bright regions may be missed out.

The approach is based on the special characteristics of the tree trunks given that the tree trunks are expected to be vertical or near vertical in off-road terrain. While the approach is only constrained to a class of obstacle, it is of

particular important as tree trunk is a class of obstacles that is one of the most hazardous to the autonomous vehicle.

5.3 Scene Consideration

In this section, we describe the appearance of the tree trunks in the environment. The tree trunks can appear in various forms and it is not feasible to have a module that can fit to the entire situation. We will consider the common appearance of tree trunks with certain exceptions and assumptions.

Generally, the rainforest terrain is under the tree canopies and the illumination condition is normally low in intensity. The scene is under the shadow of the tree canopies regularly with some occurrence of direct sunlight. In order for the passive stereo camera to be functional, it is expected that the scene will be sufficiently bright to be functional. Without considerable brightness, the stereo camera will not be able to capture good quality images. The illumination source must not be in front of the camera to avoid the pixels of the images to be over-saturated. When the pixels are over-saturated, the color information of the images will be lost.

There are various appearances of tree trunks in the rainforest scene. Majority of the tree trunks are vertical or near vertical. This module is intended to detect the common presence of the tree trunks which is the straight near vertical tree trunks. Tree trunks that are collapsed horizontally on the ground will be considerate part of the ground plane. If the collapsed tree trunk is lower

than certain threshold and slope, it will be treated as part of ground plane and classified as traversable. Otherwise, it will be placed as non-ground area before further classification.

5.4 Feature Extraction

In this section, the U-disparity image and Sobel edge detector are described. Both V-disparity image and Sobel edge detector are used to extract meaningful cues that can be used to detect tree trunks. The characteristics and cues representing the tree trunks are presented.

5.4.1 U-disparity Image

Similar to V-disparity image, U-disparity image is based on disparity map ($I_{V\Delta}$) obtained from the stereo camera. The difference between U-disparity image and V-disparity image is that the accumulation of the same value is projected onto abscissa-axis instead of ordinates-axis.

The U-disparity image contains three-axis which are the abscissa-axis, ordinates-axis and intensity-axis. The ordinates-axis (d) plots the disparities which the correlation has been computed. The abscissa-axis (U) plots the image column number and the intensity-axis set to be proportional to measured correlation. Figure 5.1 shows a stereo image and its corresponding U-disparity image where the abscissa-axis (U) is built by accumulation the pixels of the

same disparity from disparity map($I_{U\Delta}$) along the ordinates-axis (d). Figure 5.1(c) shows U-disparity image is the disparity-based histogram where the accumulation of disparity of the same value projected onto ordinates-axis (d). To detect obstacles in the scene, the features that may represent the obstacles must be extracted.

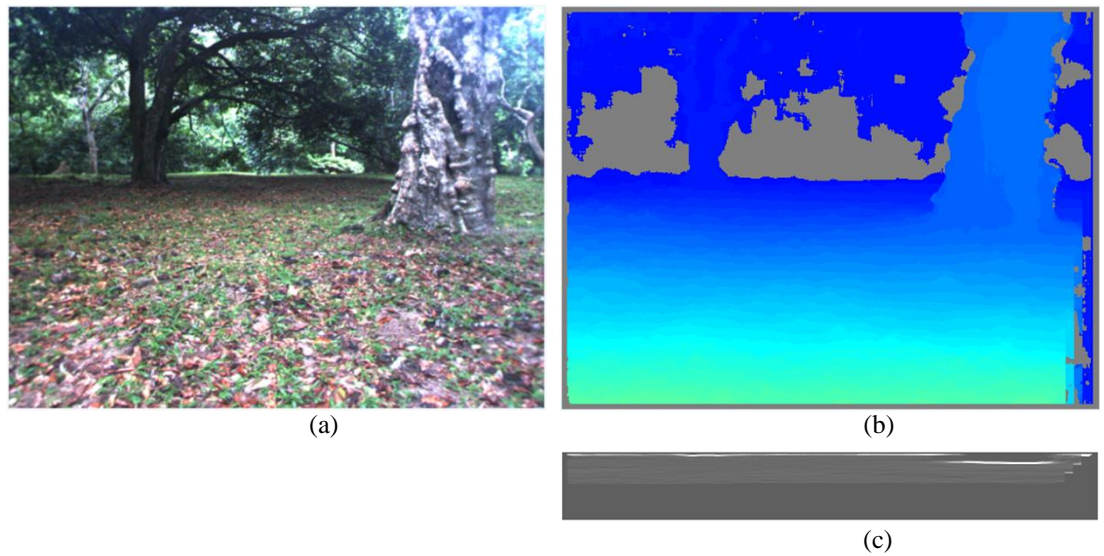


Figure 5.1: Sample image of (a) scene with tree trunks, (b) Disparity data, (c) Corresponding U-disparity image.

In the developed algorithm, we utilized the disparity accumulation (the intensity of the U-disparity image) to detect the presence of obstacles. The corresponding U-disparity is shown below the disparity image. The ground plane disparity is changing across the image, thus intensity representing the disparity accumulation is low. However, for an obstacle with certain height, there will be similar disparity information across the surface of the obstacles.

Thus, high intensity in U-disparity image corresponds to region that does not belong to the ground plane.

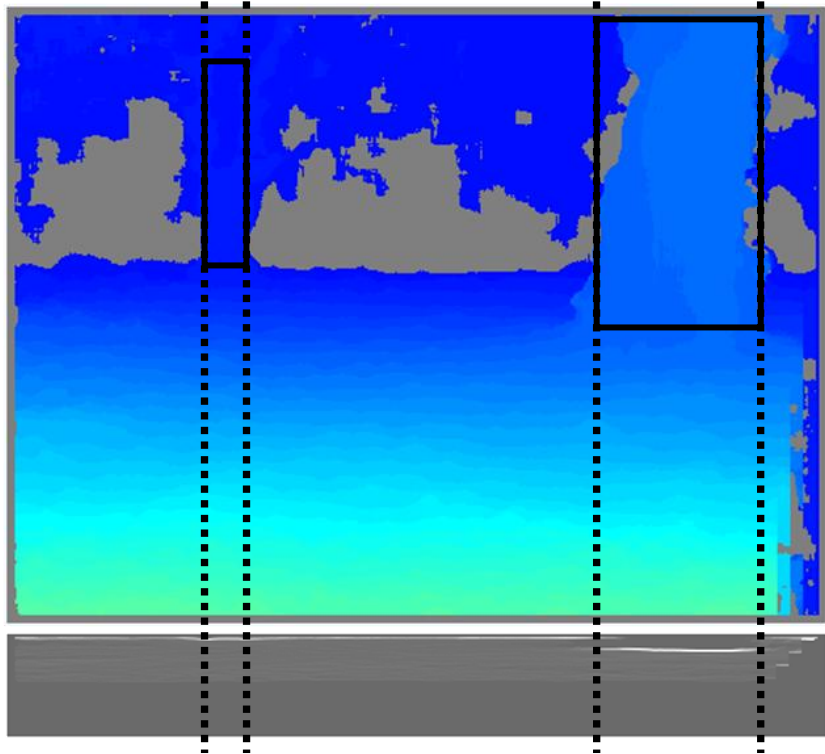


Figure 5.2: Disparity accumulation in U-disparity image as a cue to spot the tree trunks.

Figure 5.2 shows enlarge disparity data and U-disparity image. In the image, the tree trunks are identified and hand labeled (in black box). In rainforest terrain, the tree trunks are typically tall and near vertical. As mentioned before, the surrounding background in the rainforest scene is considerably spacious due to the presence of the canopy limiting the growth of smaller vegetation. Consequently, the high intensity in U-disparity image

usually represents tree trunks. This feature in the U-disparity image is used as one of the cue to detect the tree trunks.

5.4.2 Sobel Edge Detector

Edge detection is one of the most commonly used operations in image analysis. Edges are boundaries between objects and background, and it may represent the outline for objects. Edge based method are based on the assumption that the pixel values change rapidly at the edge between two regions. The edge detection is based on the change of gray level at the boundaries of two regions. In this thesis, we used gradient method namely Sobel edge operator to detect the edges.

Sobel edge detector detects the edges by searching for maximum and minimum and minimum of the first derivative of the image. The edge detection is done by applying Sobel convolution kernel across the image. If we define the image to be $I(x, y)$, the kernel to be $G(i, j)$ (where $0 < i < M_i - 1$ and $0 < j < M_j - 1$), and the center point(anchor point) to be located at (a_i, a_j) in the coordinates of the kernel, then the convolution $H(x, y)$ is defined by the following expression:

$$H(x, y) = \sum_{i=0}^{M_i-1} \sum_{j=0}^{M_j-1} I(x + i - a_i, y + j - a_j)G(i, j)$$

The value of convolution of a particular pixel is computed by convoluting with a convolution kernel. Figure 5.3 depicts a 3-by-3 convolution kernel with the anchor located at the center of the array. For each kernel point, the value for the kernel at that point and a value for the image at the corresponding image point are multiplied together and summed. The result is then placed in the resulting image at the location corresponding to the location of the anchor in the input image. This process is repeated for every point in the image by scanning the kernel over the entire image.


1	-2	1
2	 -4	2
1	-2	1

Figure 5.3: A 3-by-3 kernel for a Sobel derivative; note that the anchor point is in the center of the kernel.

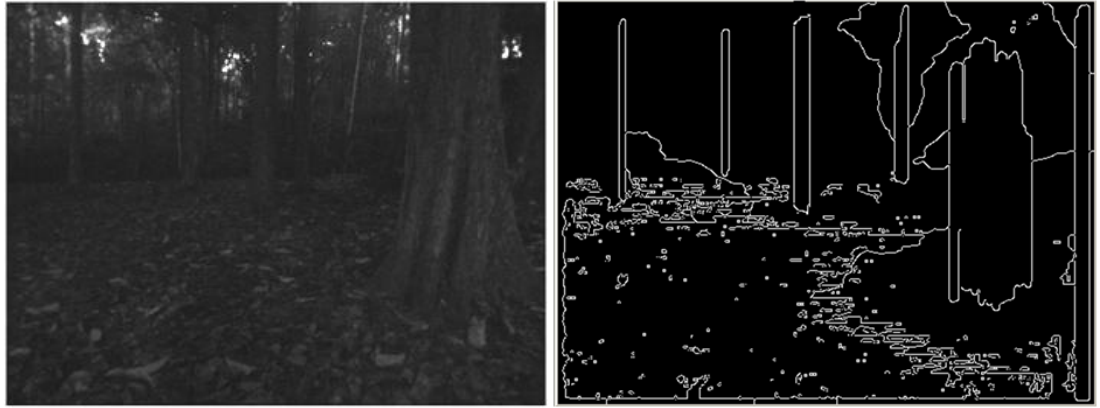


Figure 5.4: Edge detected using Sobel Edge detector. Note that the vertical and near vertical lines are usually represent the tree trunks.

5.5 Tree Trunk Detection Algorithm

The developed algorithm has two main parts which are edge detector and disparity threshold. Both parts are specifically used to detect the tree trunks characteristics. The work flow of the proposed module is shown in Figure 5.5.

Once the scene frames are captured. The ground plane detection algorithm describe in Chapter 4 are employed to detect the ground plane of the scene. Based on the detected ground plane, we anticipate that any region that is not classified as ground region will be potential obstacle region. The search for tree trunk will be done in non-ground region.

The first part of tree trunk detection module detects the tree trunk based on the edge of the tree trunk. It is expected that the tree trunk is vertical or near

vertical. The edge contours are extracted using Sobel edge detector as described in Section 5.4.2. The Sobel edge detector produces contours that represent portions of tree trunks. Based on the Sobel edge detection cue alone, there are various contours produced by other objects as well. The contours that represent the tree trunks are extracted if they match the vertical profile produced by U-disparity cue.

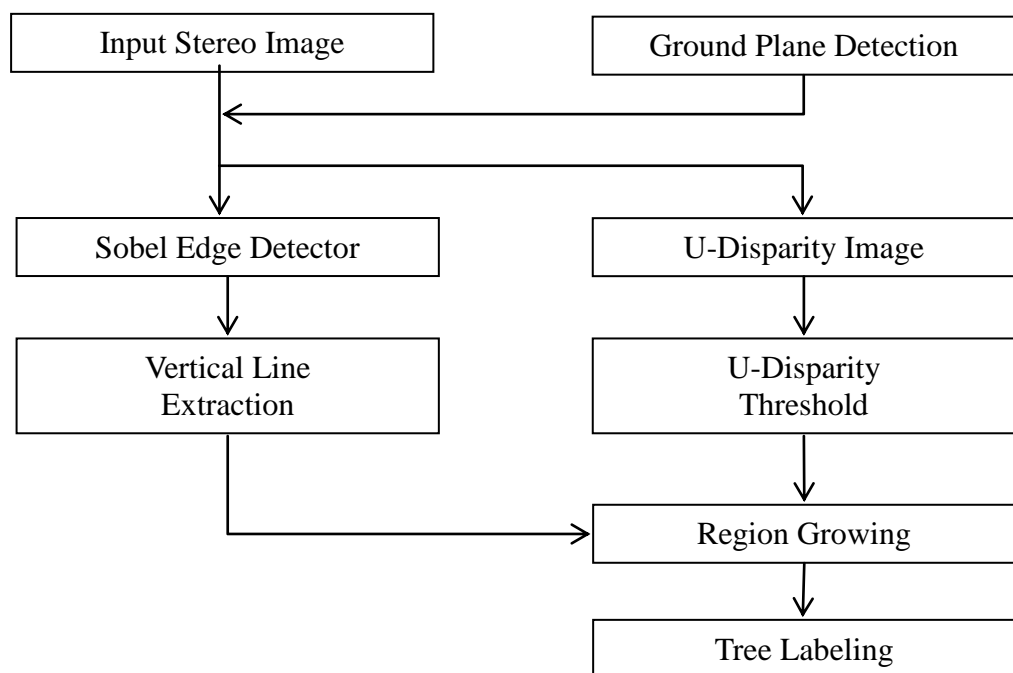


Figure 5.5: Proposed tree trunk detection module using edges and U-disparity image.

The U-disparity image is employed to group object with similar disparity value along the image column (x-axis) as described in Section 5.4.1. Since it is expected that the tree trunks are tall and straight, the accumulation of disparity will be higher compared to other object.

Once the parts of the trunk are detected, region growing is applied to detect the whole portion of the tree trunks. Combining features from detected edges and U-disparity enable effective detection of tree trunks. The following section depicts the experimental results of the tree detection module.

5.6 Experimental Results and Discussions

The experimental results of the tree detection algorithm are discussed in this section. The results are evaluated based on the error of samples used and distances from the camera. The algorithm is tested with 15 different samples of typical rainforest terrain. The image size is 512x384.

		Trunk	Not Trunk
yes	TP	FP	
no	FN	TN	

Figure 5.6: The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.

Similar to ground plane detection, the performance of our ground plane detection algorithm is quantified by discrete classifier model. Figure 5.6

illustrates the performance classifier used. The true class of interest in this developed module is tree trunk and not tree trunk. Given that there are two true classes, there are four possible outcomes. If the object is tree trunk and its predicted class is YES, it is classified as true positive (TP), else if the predicted class is NO, it is classified as false negative (FN). Similarly, if the object is not tree trunks and its predicted class is NO, it is classified as true negative (TN). False positive (FP) occurs when the true class is not tree trunk and the predicted class is tree trunk (YES).

Based on Figure 5.6, it can be seen that both false negative and false positive are not desirable as it represent wrong detection. However, in this module, the false negative result holds higher significance as it represents missing tree trunk detection that will harm the autonomous vehicle. If there is tree trunk and there is no detection, it may lead to damage to them autonomous vehicle. Thus it is essential to keep the false negative classification.

Figure 5.7 shows sample images of forest scene with tree trunks and the classified region by developed algorithm. In the first image, the scene has very little tree trunks and the algorithm managed to detect tree trunks. Note that the algorithm does not detect the horizontal tree branches. In the second image, there are many tree trunks in the image. The tree detection algorithm performs well in detecting the tree trunks that are near to the camera. However, there are misses of tree trunks that are very far away from the camera. The third sample image shows the detection in complicated terrain. It can be seen that the algorithm managed to detect the tree trunk even it is partially hidden behind

other obstacles. The green vegetation in front of the tree trunks are labeled as unclassified since it is not in the interest of this thesis.

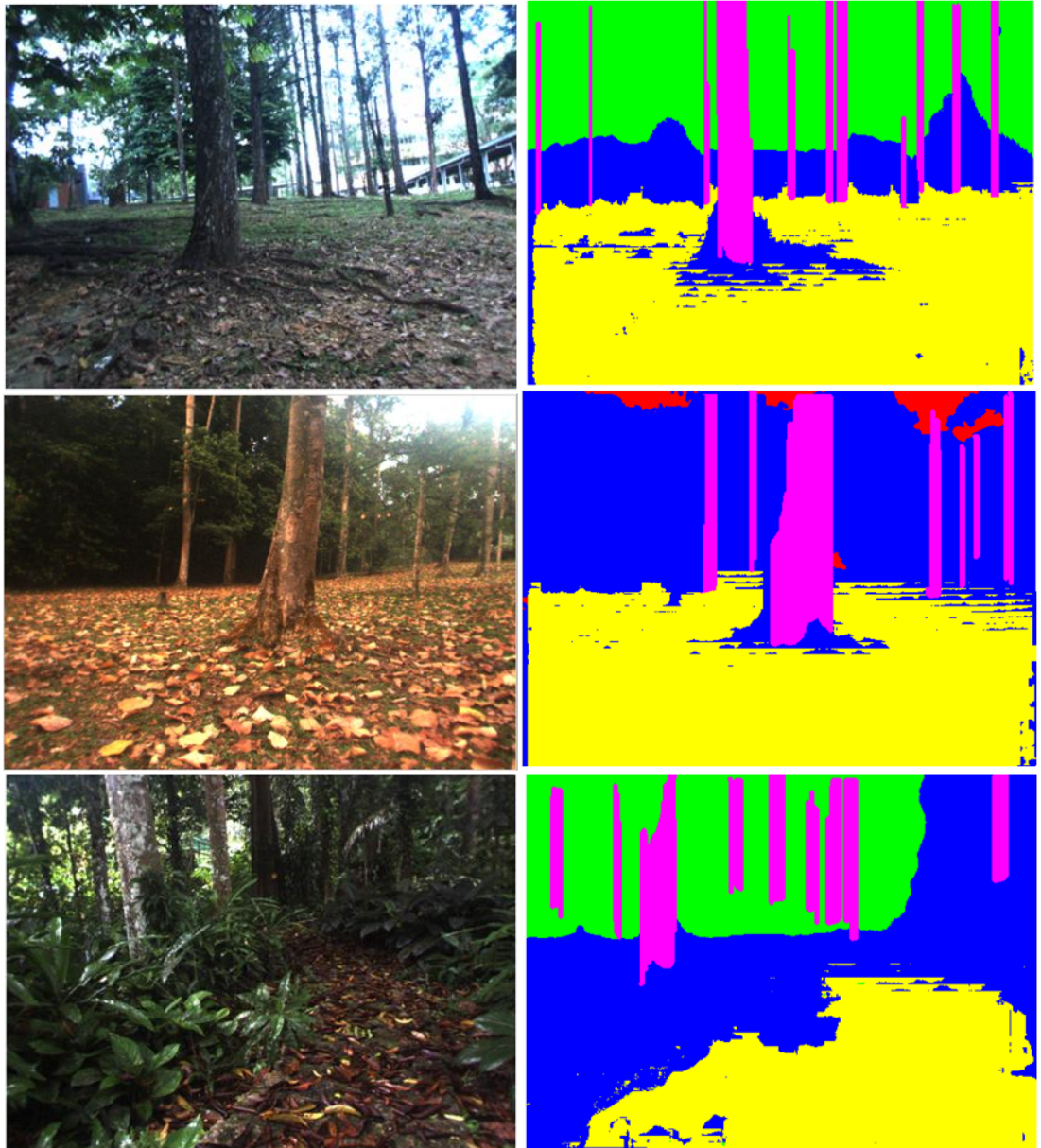


Figure 5.7: Sample scene with tree trunks as obstacles and the detected tree trunks in magenta. Magenta and blue correspond to tree trunks and ground region respectively. Green region indicate region with no disparity information

and blue color represent unclassified region which may be other types of obstacles.

When an object is very far away from the stereo camera, the disparity contrast between regions is very low. Consequently it is harder to distinguish between two different regions when the contrast is low. The tree detection module fails to identify the tree trunks when the tree trunks are very far away. However, as the stereo camera approaching the tree trunks, the correct detections are achieved.

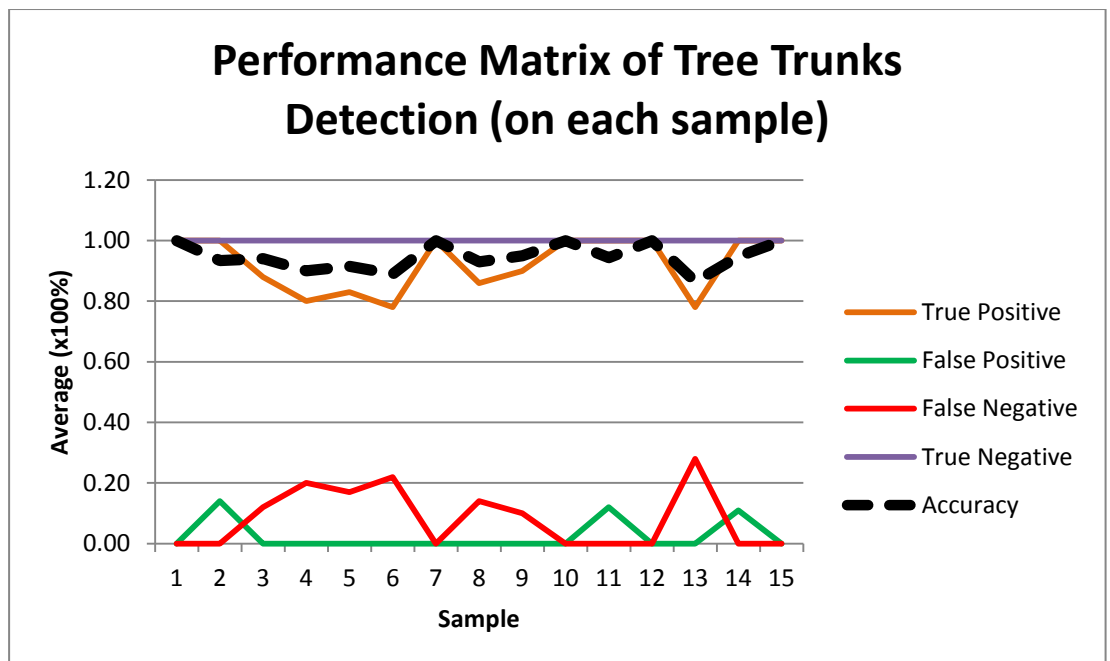


Figure 5.8: Performance matrix of tree trunk detection module based on samples used in this experiment.

Figure 5.8 shows the performance of the tree trunk detection based on 15 samples of different scene in rainforest scene. The accuracy of the algorithm

for each case is around 90% while maintaining low false negative in most of the cases. The false positive is usually caused by other obstacles which is very tall in height. While the false positive is undesirable, it usually represents obstacles that cannot be run over due to the object height which includes compressible vegetation. The true positive rate is maintained at above 80%. This true positive rate is directly affected by the false negative result as the false negative increases, the true positive rate decreases.

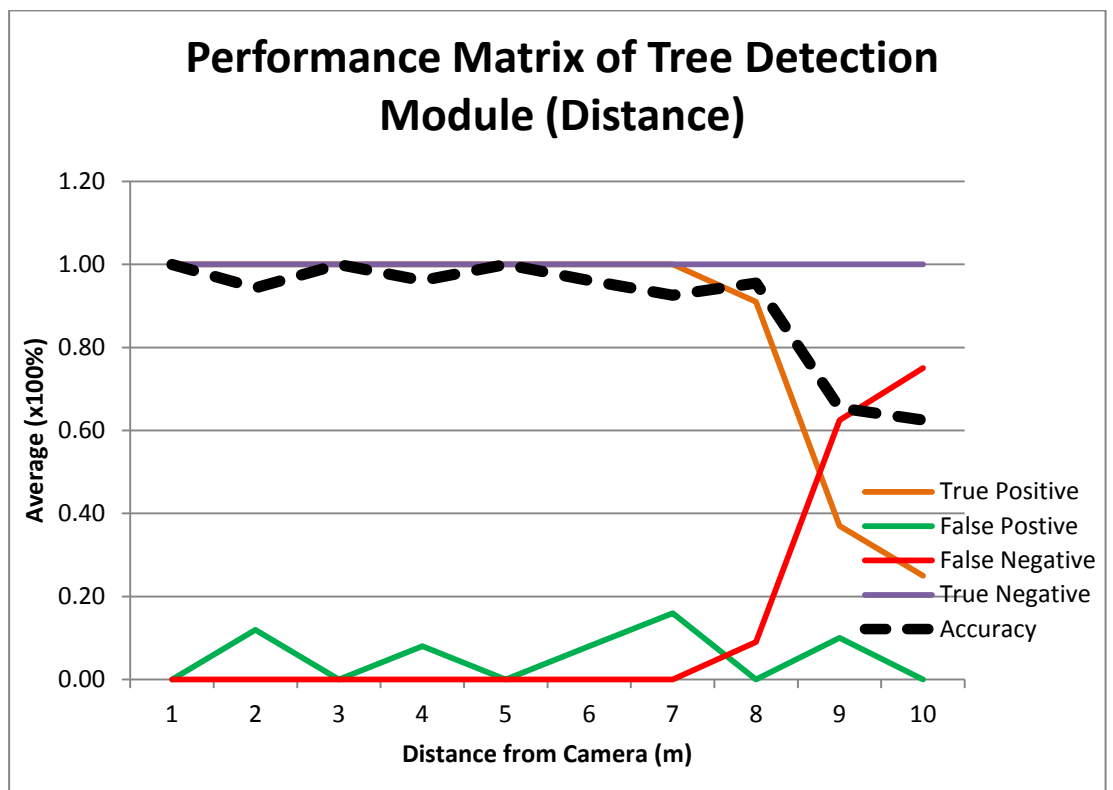


Figure 5.9: Performance matrix of tree trunk detection module based on distance from the stereo camera.

Figure 5.9 shows the performance of the detected module based on the distance from the stereo camera. The tree detection module managed to

maintain good true positive and false negative rate at the distance less than 7 meter from the stereo camera. The developed algorithm fails to register the disparity data due to the low contrast of disparity data at far distance. Objects at far distance will have smaller disparity values and the disparity variances between two pixels are low. In addition, tree trunks at far distance appear to be smaller, thus disparity count in the U-disparity image will fall below the threshold set to detect the tree trunks. The false negative which is hazardous to the vehicle happened at the distance which is relatively far from the vehicle and tree detection module will be able to detect the tree trunks when it is nearer to the tree trunks. The false positive curve in this graph does not represent any significant finding as it can happened anywhere when there are obstacles which are relatively tall.

5.7 Summary

The algorithm produced good results in the condition in most of the rainforest terrain condition. It was able to determine the tree trunks of the scene and mapped the corresponding tree trunks pixels to the image.

We managed to maintain the average accuracy of 90 percent over the experiments. The false negative outcome is very critical as this condition is hazardous to the vehicle. In this thesis, the developed algorithm managed to maintain the false negative alarm below 10 percent for most of the conditions.

There are conditions that proposed algorithm may not perform effectively. First, in darker scene where the image contrast is low, the tree will appear to be part of the background scene. The stereo correspondence cannot be found and it may fail to detect the tree trunks at a distance. However, the proposed will be able to detect the tree trunks when it is closer to the tree trunks.

Another condition which it may fail is when the scene is very complicated where there are other obstacles which are of medium height. Since the proposed algorithm is based on height and verticality, it may produce high false positive alarm under this condition.

At this stage, the proposed tree trunk detection algorithm can only be used in rainforest terrain. In secondary forest where compressible vegetation is dominant, the performance of tree trunks detection is very limited. Nevertheless, it is still possible to improve the performance of this tree trunks detection module by reducing the false positive alarm through addition module to detect other types of obstacles. This will involve another work scope and will be considered for future research.

CHAPTER 6

WATER BODY DETECTION

6.1 Introduction

Water body detection such as water patches and river are one of the most challenging obstacles commonly found in rainforest terrain. Traversing through a water body may damage the autonomous vehicle electronic component. Also, the autonomous vehicle may be trapped in the water patches if the water level is too deep. Thus there is a need to detect the water body for the vehicle control to steer away from the hazard. However, detecting water in rainforest terrain is complicated due to different appearances of the water body. The water body is different from common obstacles as it does not have geometric appearance that can be tracked. In addition, the color of the water body may be of the environment reflection, sky reflection, background color and etc.

Due to the different appearances of the water body, it is not possible to use a single feature to detect all the appearances of the water body. There are several attempts to use color, texture, disparity feature and polarization camera to detect the water body found in the literature. Most of the proposed methods are able to extract the water body from the scene in unstructured outdoor terrain. In this thesis, we extended the water body detection to rainforest terrain utilizing the polarization filter on the stereo camera.

A detailed description of the state-of-the-art along complication in the water body detection is presented in next section. Then, the basic principles of our method are explained. Finally, the results of our proposed method are presented.

6.2 Overview of Water Body Detection

A review on the work done revealed that only a limited number of works done on the detection of water body detection. Manduchi et al (2005) in their works that water body detection remains a problem to be solved. Rankin et al (2010) provides a comprehensive review on water body detection system and evaluation of the performance of water detection algorithm. As mentioned previously, there are several works done utilizing multiple features to detect water body. In this section, we will highlight the state-of-the-art of the water body detection

6.2.1 Method by The Jet Propulsion Laboratory (JPL)

Under the Robotics Collaborative Technology Alliances (RCTA), there are several researchers developed methods for water detection using vision technology. The Jet Propulsion Laboratory (JPL) participated in this program, with the focus on analyzing the features for water that can be exploited from a color stereo camera mounted in front of the autonomous vehicle.

Rankin (2004) in the early of program developed a rule-based algorithm to combine water cues from color, texture and stereo camera information. Each of the detectors is designed to target specific water appearances. The rules for fusing the water cues are designed to maximized water body detection while minimizing false detection. This work reveals the appearance of sky in the scene and also reflection of sky from the water body can be segregated using the color cue. Reflection of sky and actual sky cover are clustered in high brightness and low saturation region. The approach is to detect the presence of sky on top of the image. Then, the lower part of the image is searched if the sky is present. However, this color-based method is susceptible to false detection where the intensity image is saturated.

Texture cue is used to detect region that is low in texture. Variance filter is scanned throughout the saturation channel and green channel of the image to detect low texture region. The water cue from texture is susceptible to false detections on dirt roads having low texture, in the sky, in vegetation, and where the image is overexposed. The initial work also shows that stereo ranging outputs a range image that can be used to detect reflections. Reflections of surrounding extend from the trailing edge of a water body and can span a portion or all of the water body, depending upon the reflected objects' height and distance from the water. The range to a reflection roughly matches the range to the reflected object. However, the reflection plots lower than the true ground elevation. In addition, zero disparity pixels can also provide evidence of a reflection. Zero disparity occurs when the stereo correlator matches the same column in rectified left and right images. When

zero disparity pixels occur in the lower half of the disparity image, it is likely caused by reflections of ground cover that is far away. Thus, zero disparity pixels can be a reflection-based water cue.

Rankin et al (2006) further improved their work by using ground detection algorithm to estimate the elevation of detected water bodies and locate them within instantaneous and world terrain maps. Temporal filtering in a world map suppresses false detections and relocates detected water as water body elevation estimates improve. The water detection regions from a single cue are connected prior to multi-cue fusion. This modification to the algorithm has improved the detection of water bodies that are narrow in image space. In addition, a ground detector is designed to detect the ground plane to improve the positive detection of the water body. The elevation of a water body is estimated by averaging the elevation of the detected ground surface around the perimeter of the fused water detection region. This work managed to detect small puddles starting at the distance 7 meters from camera and large water body from the range 13 meters.

Ranking et al (2010) explored the possibility of developing a water detector based on the observation that the color of a water body tends to gradually change from the leading edge to the trailing edge, when other naturally occurring terrain types typically do not. Moving from higher to lower incidence angles, water body saturation and brightness move in opposite directions, with saturation increasing at a much higher rate than brightness decreases. This work utilized frame sequences of the scene to detect the water

body as the slope magnitude tends to be higher for water than other naturally occurring terrain, such as soil and vegetation

The works by JPL provide a detailed characteristic of water body and cues that can be utilized to detect different appearances of the water body. The works outlined that water body can be detected by using brightness cue, low-texture characteristic and disparity unusual characteristics.

6.2.2 Method using Polarization-Based Camera

In this section, we present a brief overview of a method to remove the highlights caused by specular reflection using polarization filter (polarizer). Over the years, there are various approaches to use polarization methods along with machine vision. Polarization methods were first introduced by Koshikawa (1979) for shape interpretation and recognition of glossy objects. Linear polarization was also used for highlight removal and material classification (Wolff & Boult, 1991). More recently, Nayar examined the use of color and polarization to remove the specularities component from reflected light (Nayar, Fang, & Boult, 1993). Most of the research done was concentrated on the detection of the polarization angle of the light. The details of polarization theory are beyond this thesis and a comprehensive can be found in work by Wolff (1997).

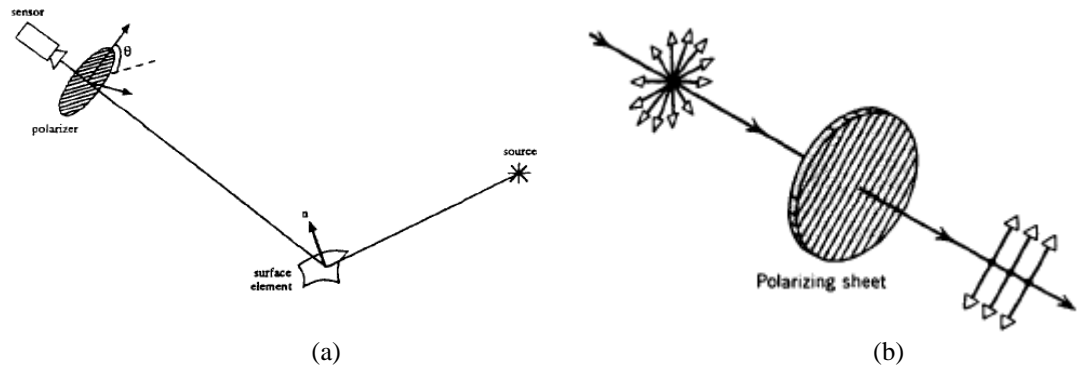


Figure 6.1: Camera setup with polarizer placed in front of the camera.

All of the above approaches focused solely on detecting the specularities and the polarization state of the image. Since our application intends to detect the highlights and the specular reflection as a feature for image segmentation, there is no need to determine the polarization state of the scene. Our approach follows closely the approaches found in literature except for the polarization state determination. Figure 6.1 illustrates the general approach to grab an image using a colour camera. A polarizer is placed in front of the sensor while the scene is illuminated by a source. The scene of interest in this thesis is rainforest scene and the illumination source will be the sunlight and also specular light reflection from surrounding.

Partially linearly polarized light from surrounding that projected onto the transmission axis of the polarizer will become linearly polarized according to the transmission axis. The magnitude of the transmitted radiance through the polarizer is the component of the polarization along the transmission axis.

Light component that is unpolarized will be attenuated (usually half) regardless of its orientation.

Xie et al (2007) attempted to detect the water body in outdoor scene by using polarization filter with three different polarization degree. A single camera with a mechanically rotating linear polarizing filter placed in front of the camera lens. Three polarization images with the polarizing filter setting to 0, 45, 90 degrees, respectively. It is based on the physical principle that the light reflected from water surface is partial linearly polarized and the polarization phases of them are more similar than those from the scenes around. Water hazards can be detected by comparison of polarization degree and similarity of the polarization phases. This method worked well to detect water region that reflect vegetation and sky. The advantage of this method is there is no need for the vehicle to stop to acquire multiple frames of different polarization angle of a scene. However, missing detections occur where ripple when water exists.

6.3 Mathematical Description for Appearance of Water Body

This section describes the characteristic of the water in mathematical form. From the model and conducted experiments, the features that may represent the water are identified.

6.3.1 Water Reflectance Model

In this thesis, we have been dealing with color image which is the result of sensing of light reflected from scene using camera. Any interaction of light with matter whose optical properties are asymmetrical along directions transverse to the propagation vector provides a means of polarizing light (Pedrotti, Pedrotti, & Pedrotti, 2007). All natural occurring light outdoors and underwater, scattered and reflected, as well as light in most indoor environment is partially linear polarized (Waterman, 1981). This is due to the reflected from surfaces may undergo diffusion or specular reflection or both (Nayar, Fang, & Boulton, 1993). The diffusion occurs from light rays penetrating the surface, undergoing multiple reflection and refractions, and reemerging on the surface. This diffusion component will be distributed in a wide range of directions around the surface normal. The specular reflection occurs on the surface of reflection such that light incident on the surface will be reflected at the angle of incident.

Figure 6.2 describes the reflection and inter-reflection mechanism that happens in a scene. Two points (A and B) in the scene are in consideration. Reflection from point A contains diffuse and specular component where the diffuse component arises from scattering of light rays that enter the surface and undergo multiple reflection and refractions. The specular component is a surface phenomenon and resulted from single reflection of light incident ray. Assuming that the surface is rough, its specular component will appear to be

spread in specular direction. The width of the distribution will depend on the roughness of the surface (Nayar, Fang, & Boult, 1993).

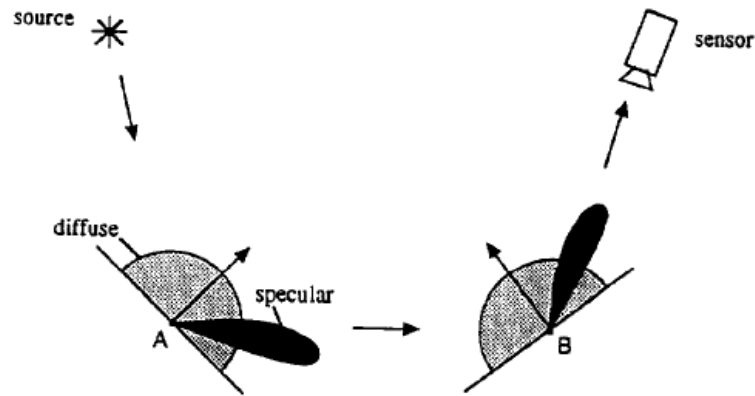


Figure 6.2: Components of reflection and inter-reflection (Nayar, Fang, & Boult, 1993).

The above illustration considers only the light from the light source. However, in a scene, the surface does not only receive light from the light source but also from other reflected light of other scene points. Thus the resulting reflected light can be combination of four possible inter-reflection components: diffuse-diffuse, specular-diffuse, diffuse-specular and specular-specular (Nayar, Fang, & Boult, 1993). When specular reflection is present, even for marginally rough surfaces, the concentration of energy reflected will cause strong highlights in the brightness of the scene. In most of the cases, the color information in the pixel will be lost due to over-saturation.

Rankin et al (2010) developed a partial model of the reflection coefficients from their experimental data. The total reflection coefficient R_{total} from a water body to a camera is the sum of the reflection coefficients for energy reflected from water body is given as

$$R_{total} = R_r + R_p + R_s + R_b \quad 6.1$$

where R_r is the energy reflected off the water surface, R_p is the energy scattered by water molecules, R_s is energy reflected or scattered by materials suspended in the water and R_b is energy reflected off the bottom of the water to the camera.

The fraction of the incident power that is reflected from an air/water interface is given by Fresnel equations for light polarized perpendicular to, $R_{r,\perp}$ and parallel to, $R_{r,\parallel}$ the plane of incidence,

$$R_{r,\perp}(\theta) = \left[\frac{n_1 \cos \theta - n_2 \sqrt{1 - \frac{n_1}{n_2} (\sin \theta)^2}}{n_1 \cos \theta + n_2 \sqrt{1 - \frac{n_1}{n_2} (\sin \theta)^2}} \right]^2 \quad 6.2$$

$$R_{r,\parallel}(\theta) = \left[\frac{n_1 \sqrt{1 - \frac{n_1}{n_2} (\sin \theta)^2 - n_2 \cos \theta}}{n_1 \sqrt{1 - \frac{n_1}{n_2} (\sin \theta)^2 + n_2 \cos \theta}} \right]^2 \quad 6.3$$

where n_1 is the refractive index of air, n_2 is the refractive index of water, and θ is the angle of incidence. The refractive index of air and pure water is 1.03 and 1.33, respectively. The most significant factors that can affect the refractive index of water are the wavelength of the light entering it and its salinity. However, these factors only alter the refractive index of water by as much as 1%.

Figure 6.3 shows the experimental result by Rankin et al (2010) where the energy reflected off the surface of water bodies increases with increasing incidence angle. Unpolarized incident light would be present on overcast days where the fraction of the incident power that is reflected from an air-water interface is the average of the polarized reflection coefficients,

$$R_r(\theta) = \frac{R_{r,\perp}(\theta) + R_{r,\parallel}(\theta)}{2} \quad 6.4$$

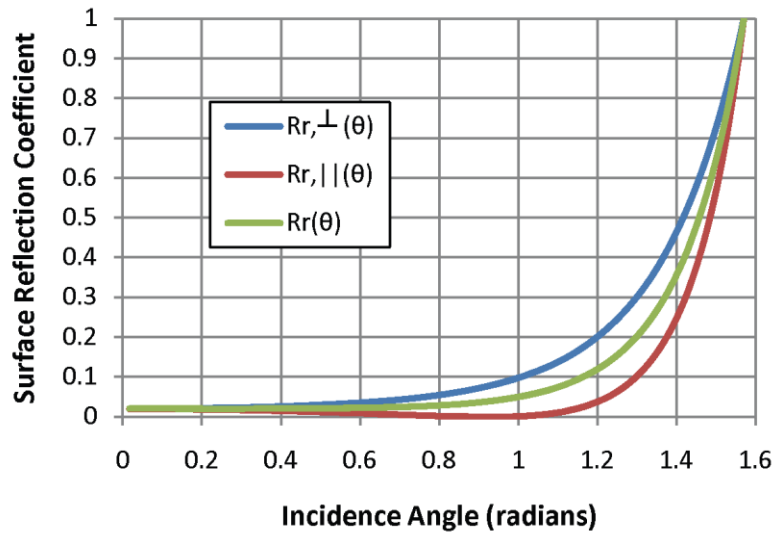


Figure 6.3: Theoretical fraction of incident power that is reflected from an air/pure water interface as a function of incidence angle. $R_{r,\perp}$ and $R_{r,\parallel}$ are the Fresnel reflection coefficients for light polarized perpendicular to and parallel to the plane of incidence, respectively. R_r is the Fresnel reflection coefficient for unpolarized light (Rankin & Matthies, 2010).

The intensity values of water pixels are related to the total reflection coefficient by

$$I = LR_{total} \quad 6.5$$

where L is an illumination factor. Substituting R_{total} into the equation yield

$$I = L[R_r(\theta) + R_p(\theta) + R_s(\theta) + R_b(\theta)] \quad 6.6$$

6.3.2 Partial Linear Polarization

Light is fundamentally a transverse electromagnetic wave possessing a state of polarization characterizing the vibrational orientations of the electric field, E . After reflection from flat surfaces of certain materials at oblique angle of incidence ψ the E field is partially polarized (see Figure 6.4).

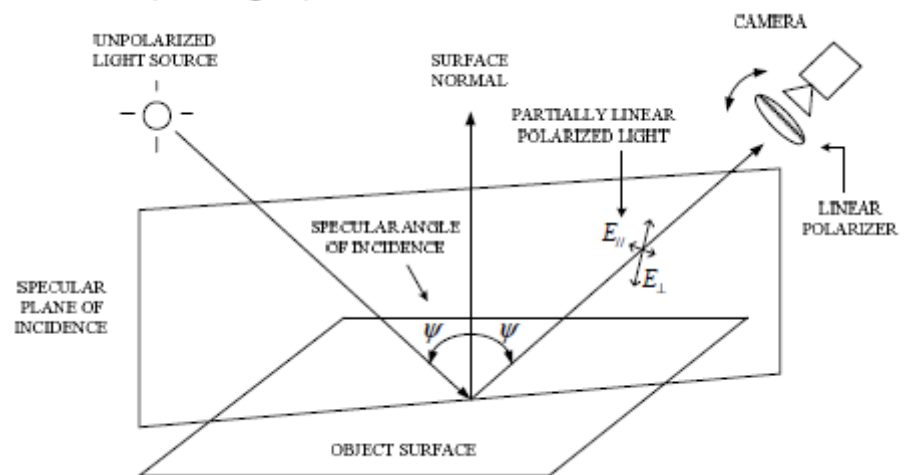


Figure 6.4: Principle of specular reflection (Xie, Xiang, Pan, & Liu, 2007).

Partial linear polarization can be described by three polarization parameters of interest which are the light intensity, degree of polarization and the angle of polarization. The partial linear polarization can be measured at a pixel level by the transmitted radiance through a polarization filter. The radiance varies sinusoidal with filter orientation. Wolff (1997) presented that a sinusoid can be uniquely characterized using three points, thus three transmitted radiance measurements can be taken between 0° and 180° to determine partial linear polarization. In his work, measurements were taken at

0°, 45° and 90°, and the intensity for respective angle is represented as I_0 , I_{45} and I_{90} . The parameters for partial linear polarization is given as

$$\text{phase, } \theta = \frac{1}{2} \tan^{-1} \left(\frac{I_0 + I_{90} + 2I_{45}}{I_{90} + I_0} \right) \quad 6.7$$

$$\text{if } (I_{90} < I_0) [\text{if } (I_{45} < I_0) \theta = \theta + 90 \text{ else } \theta = \theta - 90]$$

$$\text{intensity, } I = I_0 + I_{90} \quad 6.8$$

$$\text{partial polarization, } P = \frac{I_{90} - I_0}{(I_{90} + I_0) \cos 2\theta} \quad 6.9$$

The application presented in this thesis is dealing with partially linearly polarized reflected light, thus we do not require as comprehensive a description. In particular, we do not require the analysis of the phase of the mutually orthogonal reflected polarization components. From Equation 6.9, it can be observed that the optimum polarization angle difference between two polarizers is 90°. The ratio of the difference and sum between the two images taken with 90° is the best measure to represent the polarization information in the present case of a stereo vision set up. Thus, the measure of the proportion of how much initially unpolarized light becomes partially polarized is given by Equation 6.10.

$$\text{partial polarization} = \frac{I_{90} - I_0}{I_{90} + I_0} \quad 6.10$$

The parameter I_{90} and I_0 correspond to maximum and minimum transmitted intensity through the polarizer. The partial polarization measure varies from 0 to 1 which represents the proportion of the magnitude of the reflected light that is polarized. At partial polarization = 0, reflected light is unpolarized whereas for partial polarization = 1, reflected light is completely polarized.

6.4 Feature Extraction

The overview of work done on water detection and mathematical description of water body were briefed in previous section. In this thesis, different polarization angles effect on the stereo images are to be experimented in this section.

In our approach, two polarizers are mounted on stereo camera. One side of the experiment setup is set as reference point at 0° . The other polarizer is a variable with reference of the first polarizer. In previous section, it is shown computationally that the optimum difference between two polarizers is 90° . However, it is necessary to investigate whether the value set is suitable to detect the water body of our interest. Figure 3.4 shows the experimental setup where one side of the polarizer is set at 0° while the other polarizer angle is

varied for the investigation. In this work, the angle in assessment is 0° , 30° , 45° , and 90° .

As our approach involves two images extracted from the stereo camera, the brightness of the image of each image will be affected by the camera auto gain control (AGC). When the polarization angle is varied, the light reflected from the scene into the camera will be affected. The camera auto gain control will increase and decrease the gain accordingly, thus there will be difference in image brightness of the image pair. Subsequently, the partial polarization parameter will be affected as well. Equation 6.10 for the partial parameter is altered and is given by

$$\text{partial polarization} = \frac{I_{\max} - GI_{\min}}{I_{\max} + GI_{\min}} \quad 6.11$$

where G is the auto gain control parameter of the camera.

Light passing through the polarizers will be polarized parallel to the orientation of the polarizer. Since both polarizers used differ in the angle of the polarization, the image pair produced by the stereo vision will differ in term of brightness and color information. The difference produced multiple features that can be used as water cue to detect the water body. The following subsection illustrates the water cue that can be obtained from the variation of polarizer's angle.

6.4.1 Water Cue from Partial Polarization Feature

Several observations can be made on the image pair of the stereo camera once there is polarization angle difference between each side of the stereo camera. The obvious difference is the intensity of the each image and the difference of intensity is further amplified by the auto gain control of the camera. This is due to specular light is minimized when the polarization is applied. The image will appear darker and the stereo camera will increase the gain to produce brighter image. The image pair brightness will be increased with the same gain, producing an image with higher brightness compared to another image. Figure 6.5 shows a sample image pair of rainforest scene to illustrate the outcome of polarization filter applied fitted in front of the camera. We describe the changes in the brightness of the image pair using partial polarization formula in Equation 6.11. The reference image set at polarization angle 0° exhibit high brightness and is described as I_{\max} . Another image with different polarization angle is fixed as I_{\min} . Comparing the image pair, we examine the changes occurred in common objects in rainforest terrain.



(a)



(b)

Figure 6.5: Sample image on application of polarizer to remove specular reflection. (a) Original image of a river scene. (b) Image after polarizer is applied.

Material/ Object	Magnitude of Partial Polarization		
	30°	45°	90°
Tree Bark	0.01	0.01	0.03
Green Vegetation	0.10	0.10	0.11
Ground/Sand	0.12	0.12	0.15
Water Body	0.30	0.30	0.33

Table 6.1: Average partial polarization for different objects and terrain (average of 20 samples).

Table 6.1 shows the average results of partial polarization of 20 samples in rainforest terrain. Different polarization angles are examined to verify the suitability and effectiveness of the polarization angle setup. The water body exhibit partial polarization changes compared to other objects. This is due to the fact that water body reflects specular light.

Example in Figure 6.5 shows that the brightness of the two images differs especially at the water region. This is due to the specular reflection on the river is filtered by the polarizer. Other objects in the scene do not exhibit many changes in term of image brightness and this in turn provides a feature to detect part of the water body.

6.4.2 Water Cue from Stereo Disparity Feature

There are a few features in stereo disparity data that can provide water body cue. As the water body may reflect the object or surrounding in the scene, there will be some deviations of stereo disparity data. Normally, the stereo disparity data is decreasing with object distance increasing from the camera. However, the water reflection may contain disparity that deviates such as zero disparity pixels, dramatic change of disparity pixels and invalid pixels.

Zero disparity pixels provide evidence of a reflection. When zero disparity pixels occur in the lower half of the disparity image, it is likely caused by reflections of objects that are far away (Rankin & Matthies, 2006). Zero disparity pixels imply that the stereo co-relator matches the same column in rectified left and right images. These zero disparity pixels usually occur when the object is very far away from the camera. By comparing each of the zero disparity pixels to the ground plane disparity pixels, it can be determined whether the zero disparity pixels belong to object that is far away or reflection from object that is far away. Usually, zero disparity pixels caused by reflection from object far away are not consistent with the ground plane profile and can be isolated.

Another appearance of the stereo disparity data is the dramatic change of disparity pixels. This feature of stereo disparity is very similar to zero disparity pixels except it is caused by reflection of objects very close to the water body. Instead of causing zero disparity pixels, there are range data

reflected in the water body. There is sudden change of range profile where the disparity data deviates away from the range profile.

Invalid stereo disparity occurs when the correlation cannot be found in rectified left and right images. By using two polarizers with different polarization angle, it is expected that the two image produced by the stereo camera will differ in term of color and brightness. Section 6.4.1 demonstrates that the differences between the image pair are greater on the water body. Thus, it is predictable that majority of the water body pixels will be invalid.

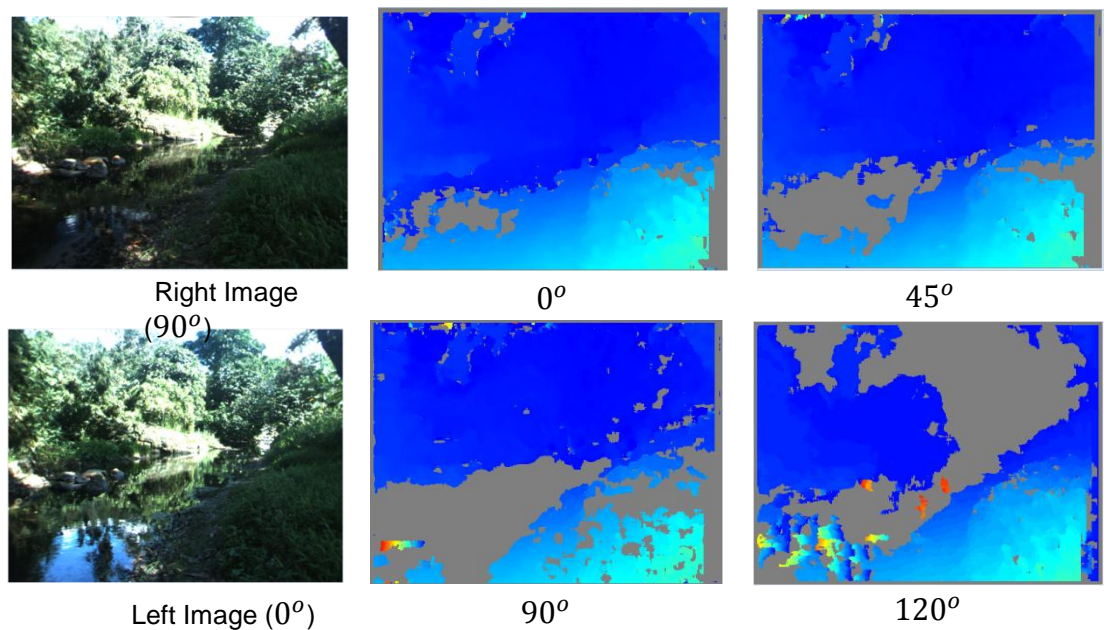


Figure 6.6: Sample disparity image with different polarization angle applied to the right side of the stereo camera. The gray color regions correspond to invalid pixels.

Figure 6.6 shows the disparity data correspond to different polarization angle. It can be observed that the invalid pixels for water body differs when the polarization angle degree changes. The invalid pixels region is increasing with the polarization angle degree until 90° . Note that there are some invalid pixels occur at the green vegetation in the scene. This may cause false detection if only a single feature is used to detect the water body. However, this false detection can be filtered out by comparing the position of the pixels to the ground plane. The false detection usually fall on higher elevation area and can be removed.

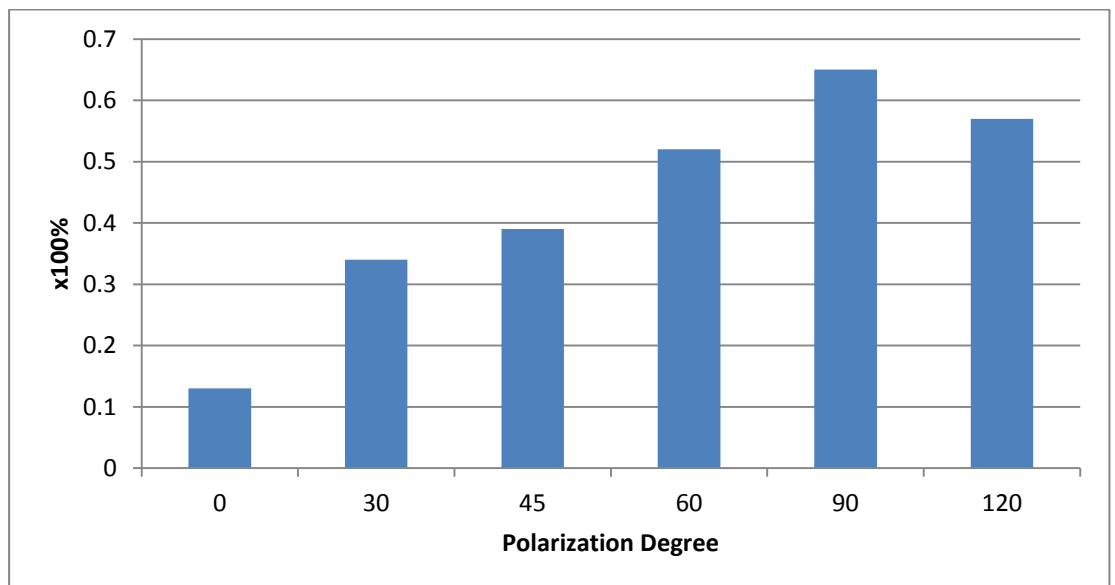


Figure 6.7: Average Percentage of Invalid Pixels for Water vs Polarization Degree (20 samples).

Figure 6.7 shows the percentage of invalid pixels region in water body over 20 samples of different type of scene. The highest average is approximately 65% at the polarization angle of 90° . Thus, it is justifiable to fix set the polarization angle to 90° for the water body detection.

6.4.3 Water Cue from Texture Feature

In images, texture quantifies grayscale intensity differences (contrast), a defined area over which differences occur, and directionality, or lack of it. Water body may appear in many types of appearances. One of the considerations is water body under the influence of the shadow of surrounding. From observation, water bodies usually appear low in texture, especially the region where the specular light is not dominant. When the texture is low, there is low variation of brightness across the region.

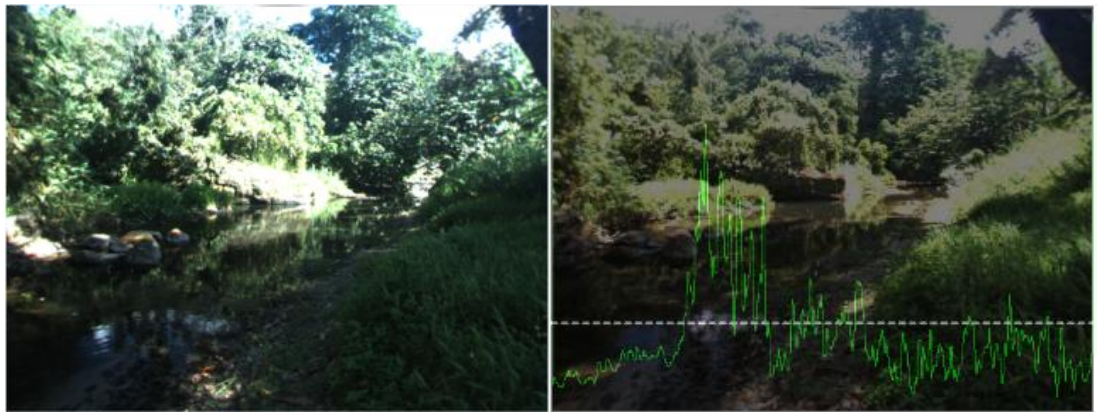


Figure 6.8: A sample image of 90° polarization degree and its corresponding line scan histogram.

Figure 6.8 shows a sample image of a scene where the water body is under the influence of canopy. The line-scan histogram across the image row highlighted by dotted line is shown in green. It can be observed that under the influence of the shadow, the texture of the water body is low as there is lesser

variation of brightness. Another region with sky reflection possesses high variation due to specular reflection. The ground and grass region can be easily differentiated from water body as they possess high variance in the image brightness. Note that the sample image used in Figure 6.8 is 90° polarized and majority of the specular reflection is removed. This indirectly aids to increase the region detectable by low-texture characteristics.

6.5 Algorithm: Fusing Water Cue

Water bodies may appear in different kind of appearances thus is not expected that no single features can be used to detect the entire water body. In this thesis, multiple-feature approach is taken to enhance the detection of water bodies with multiple cues. Previous sections discusses features that can be used to target specific water attribute in the while minimizing the false detection. Figure 6.9 shows the framework of water detection module in this thesis.

The input images are polarized stereo camera image with 90° difference polarization angle between the image pair. Stereo processing is performed on the image pair to produce disparity data. Note the detected ground plane is also fed into this module to reduce the false detection of the water bodies. Any detection of water region above the ground plane will be filtered as water bodies presence will only occurs on the ground plane.

There are four sub-modules which target different type of water attributes. The sub-modules are sky-reflection detection based on color and

brightness information, low-texture region detection, invalid disparity detection and object reflection detection. Once the sub-modules detect the water body region of interest, the region is refined by post-processing. The following sections discuss in details the operation of each sub-module.

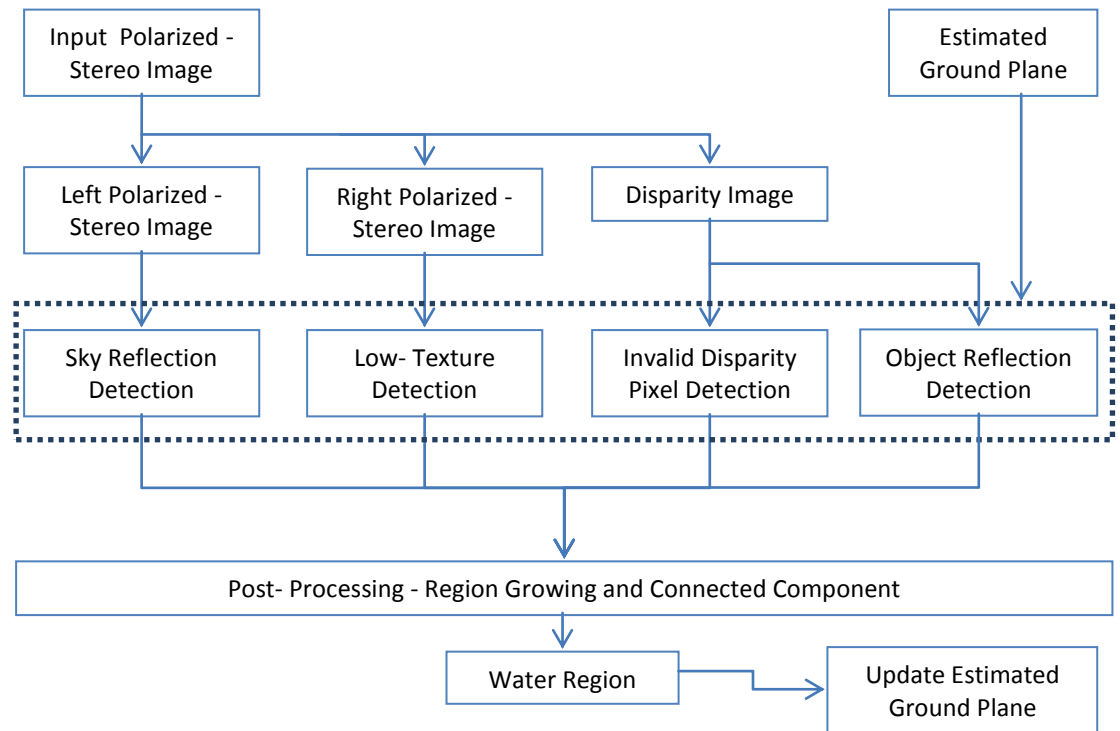


Figure 6.9: Framework for water bodies detection using multiple features.

6.5.1 Sky Reflection Detection Module

The RGB images selected from our archive for processing were converted to hue, saturation, and value (HSV) color space. There are several factors that contribute to the surface color of water bodies. Among them include the depth of the water, the amount and type of sediment in the water,

the color of the sky reflecting on the water, the color of background material casting a shadow on the water, and whether or not the water is moving. As these factors have great variation, it is difficult to predict the hue of water (Rankin, Matthies, & Huertas, 2004). Note that the reflection of the sky in water has low saturation values and high brightness values.

In this sub-module, we follow closely the work by Rankin et al (2004). The following are the rules imposed in this work:

$$S = 0 \tag{6.12}$$

$$S \leq 0.27 \text{ and } B \geq 0.73$$

$$S \leq 0.1 \text{ and } B > B_{min}(S)$$

$$S \leq 0.3 \text{ and } B > B_{min}(S) \text{ and } 240 < H < 285$$

where, S is saturation, B is brightness and H is hue. Rule 1 and rule 2 targets the sky reflection characteristics where it has low saturation and high brightness value. Rule 3 is lower brightness thresholds are applied only if the sky is detected in the imagery. Rule 4 is the only one that uses hue. It targets deep bodies of water, which tend to have a blue hue.

Note that in his approach, the sky detection is done the top ten rows of the image. Rule 3 and rule 4 are only activated if the sky is detected. However in this thesis, the sky detection is not performed as the search for water bodies in the scene is constrained to the region below the horizon. This is due to the

nature of water body to be on the ground plane rather than areas of higher elevation. Thus, false detection such as sky as water bodies can be eliminated.

6.5.2 Low-Texture Detection Module

Texture is an important approach for region description. Although no formal definition of texture exists, intuitively this descriptor provides measure of properties such as smoothness, coarseness and regularity. We use one of the simplest approaches to describe texture which is variance. Variance can be used in texture description as it is a measure of intensity contrast and is give as

$$\sigma^2 = \left(\frac{1}{N} \sum_{i=0}^{N-1} x_i^2 \right) - \left(\frac{1}{N} \sum_{i=0}^{N-1} x_i \right)^2 \quad 6.13$$

where N is the number of samples and x is the intensity of the pixel. It can be used to establish descriptors of relative smoothness using the following

$$R(z) = 1 - \frac{1}{1 + \sigma^2(z)} \quad 6.14$$

A 3x3 intensity variance filter is passed over grayscale image. At each pixel, the window variance and relative smoothness is calculated. Then the water body region with relative smoothness less that threshold set will be selected. Table 6.2 shows the values of the relative smoothness of water bodies and other objects. Water bodies appear to be smoother compared to other

objects where the relative smoothness is rather coarser. Based on the average values, we formulate the threshold that will be used to detect water bodies cue and differentiate it from other objects.

Objects	Average R(normalized)	Threshold Set for R
Water Bodies	~0.001	$R < 0.001$
Other Objects	~0.079	$R > 0.01$

Table 6.2: Average relative smoothness for 20 samples for water bodies and other objects in rainforest scene.

6.5.3 Object Reflection and Invalid Disparity Pixel Detection Module.

In this module, the partial polarization feature is used to remove the specular reflection in one of the stereo image. In Section 6.4.1, we have shown that water bodies will reflect more light compared to other objects. Using these characteristics, we will be able to distinguish water body from other regions.

In this thesis, partial polarization of 90° is used since the magnitude of partial polarization for water is highest at this angle. When different polarization angle is applied to the stereo camera, the image pair appears to be different. Therefore, the stereo correspondence cannot be found. Consequently, there will be no disparity data for the water region.

6.6 Experimental Results and Discussions

The experiments were done in various scenes in rainforest terrain. In the rainforest, it is anticipated that water body present in two types of category, mainly standing water and running water. Standing water represents stationary water body such as water patches while running water represent moving water such as stream or river. Various water body cues were used to identify different appearance of the water. In this section, we present the experiment results using developed algorithm in the scenes describe above. The error caused by back-lighting are also discussed.

The algorithm is tested with 70 samples of water body in rainforest terrain. The test images cover both standing water and running water scene. Thirty (30) samples are used for each type of scene while ten (10) samples are used for error explanation. The image size is 512x384. The image size is sufficient to detect wider Field of View (FOV) in our applications.

The performance of our water body detection algorithm is quantified by discrete classifier model. Figure 6.10 illustrates the performance classifier used. The true classes of interest in this developed module are water and not water. Given that there are two true classes, there are four possible outcomes. If the object is water and its predicted class is YES, it is classified as true positive (TP). If the predicted class is NO, it is classified as false negative (FN). Similarly, if the object is not water and its predicted class is NO, it is classified

as true negative (TN). False positive (FP) occurs when the true class is not water and the predicted class is water (YES).

		Water	Not Water
yes		TP	FP
no		FN	TN

Figure 6.10: The performance classifier of developed algorithm. The true classes of the objects are indicated in the column while the predicted classes are indicated in the row.

Based on Figure 6.10, it can be seen that the performance of developed module is good when the true positive and true negative result is high as they represent the correct results. Both false negative and false positive are not desirable as it represent wrong classification of the scene.

However, in this module, the false negative result holds higher significance as it represents wrong classification that will harm the autonomous vehicle. If the true class of the object is water and it is wrongly classified as not water, it may lead to damage to the autonomous vehicle. Thus, it is necessary for the false negative classification to be kept as low as possible.

The following subsections discuss the result of the developed water body detection module applied in unstructured terrain.

6.6.1 Running Water Detection

The developed water detection algorithm is tested on running water image sequence. We define running water as water stream such as river. Such water body appearance usually has water ripple associate with it. Figure 6.11 show a few sample images of the running water body used in the experiment.

The detection of water body is an application extension based on ground plane detection discussed in Chapter 4. Once the ground plane is detected, the plane is searched for presence of water body. The detection is done on ground plane only as it is based on the assumption water body will only appear on the ground plane.

Figure 6.11 (a)(i) and Figure 4.10(b)(i) depict two sample images of a stream. It can be observed that running water exhibit specular reflection in majority of the water body region. This is due to the reflection caused by the ripple while the water is flowing. Reflection from the surrounding is very minimal in running water.

Figure 6.11 (a)(ii) and Figure 6.11 (b)(ii) show the detected water region using developed module. It can be observed that developed module managed to identify the water region in the scene effectively.

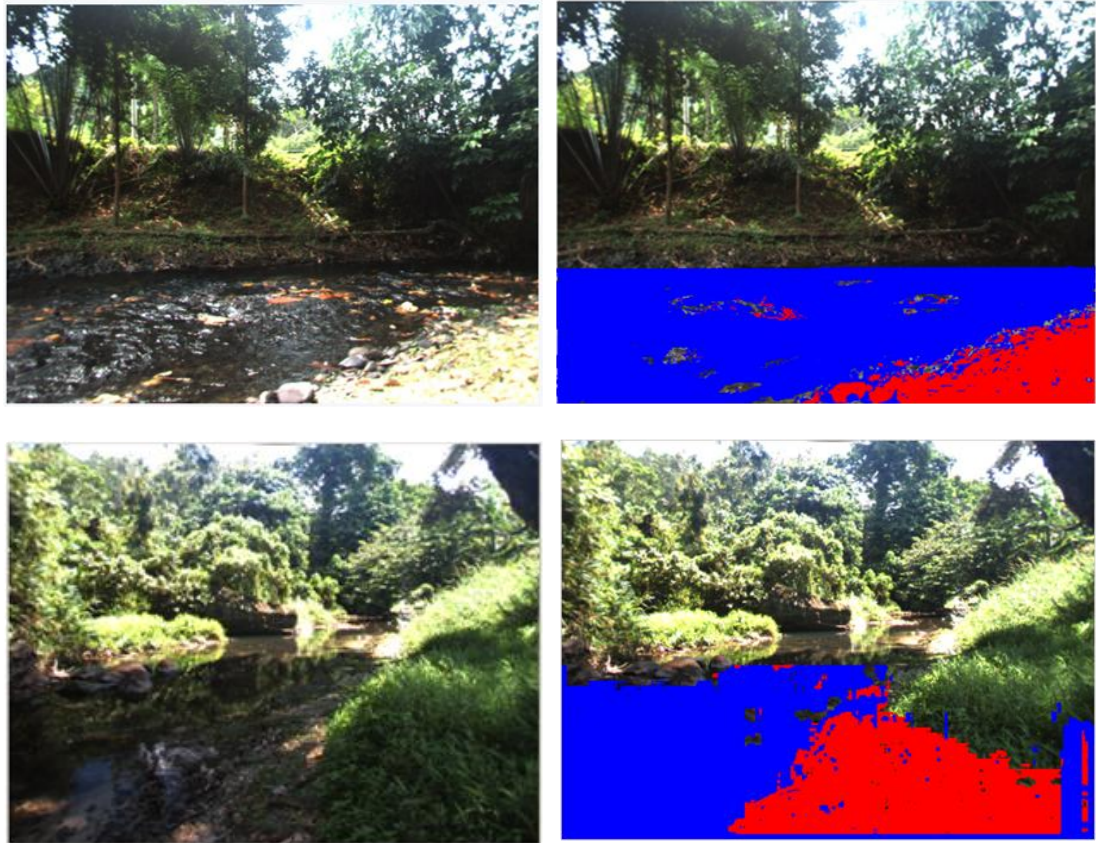


Figure 6.11: Sample results of detected ground plane in rainforest terrain for simple unstructured terrain.

Figure 6.12 shows the average error of detected water region versus the distance from the camera. Similarly to the ground plane detection, the error is calculated based on comparison between detected water region and hand labeled water pixel. It can be seen that the errors are below 10 percent for the first eight meters. The errors start to increase after 7 meters. This is due to the ground plane detection limit (discussed in Section 4.6.1).

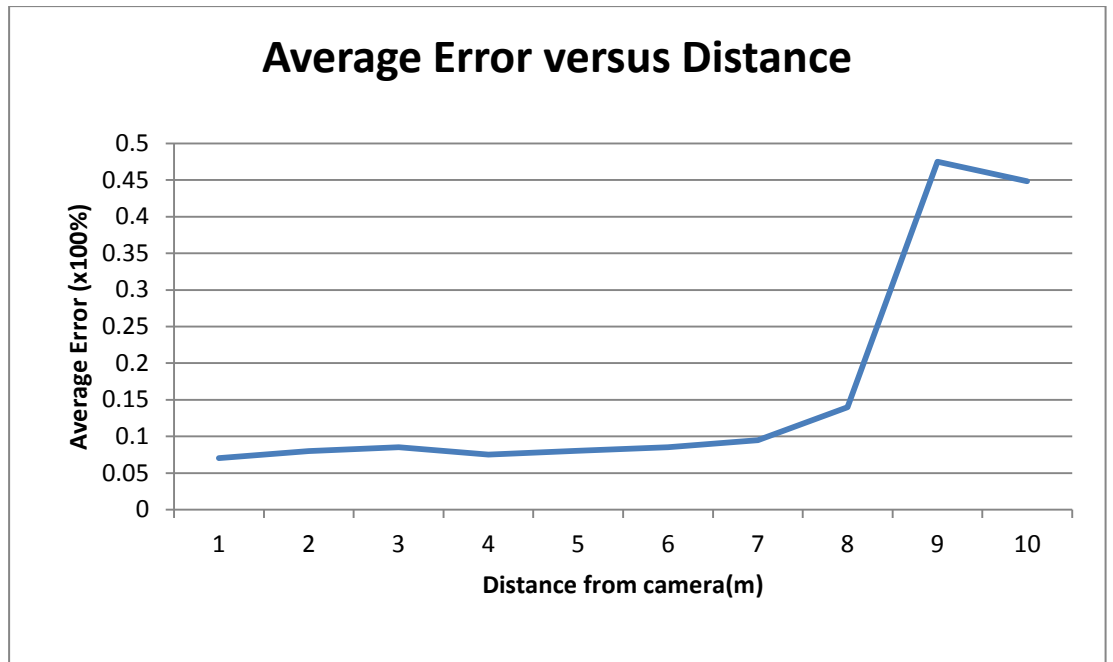


Figure 6.12: Average water body detection error versus distance for running water. The number of samples used is thirty (30).

Figure 6.13 shows the performance of the developed module in detect the running water detection. Similarly to tree trunk detection, the false negative which indicates no water region while there is water region is unwanted result. The developed module is able to effectively detect the water regions within 7 meters from the stereo camera. The accuracy within 7 meters is more than 90 percent while the false negative is kept below 5 percent.

The false negative detection is increasing after 7 meters due to the restriction by our developed ground plane detection. The detection of water region is based on the detected ground plane where this water detection module will only search for water region on the detected ground plane. It is based on the assumption that water will only present on ground plane.

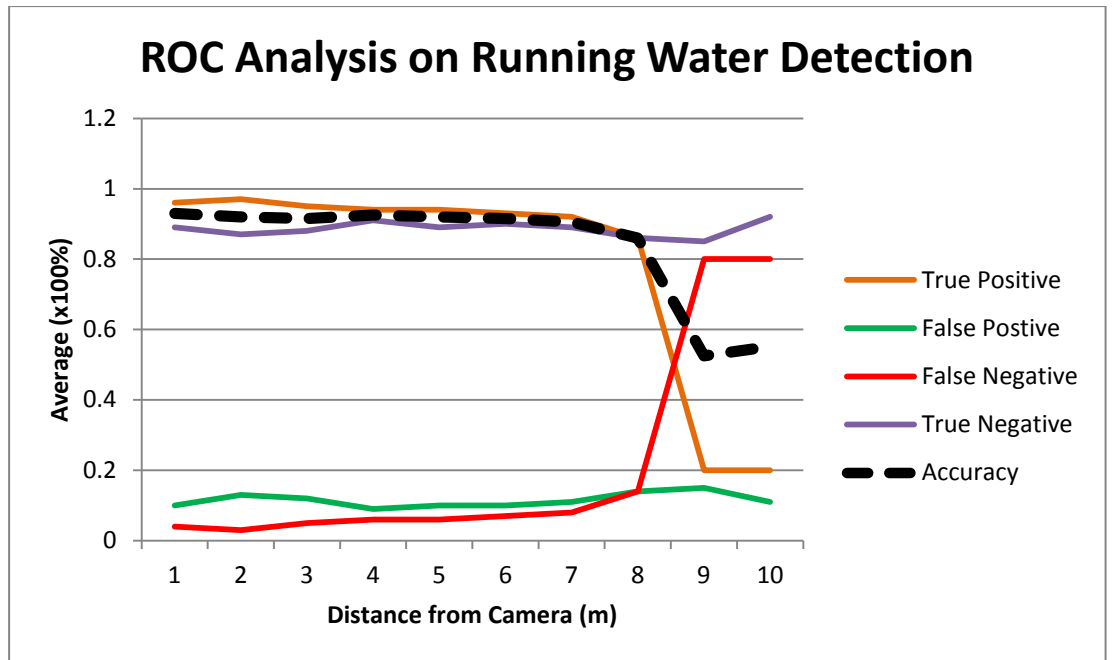


Figure 6.13: Performance matrix of the ground plane detection module in running water detection. The number of samples used is thirty (30).

The basis of searching the water region on the detected ground plane is due to the fact that one of the feature used in this module are the high brightness caused by sky reflection. As mentioned, one of the appearances of water body is sky reflection where higher brightness is expected. If the search for water region is done in other region other than ground region, it may contribute to false positive as the sky may be present in the scene. Thus, in order to reduce false positive detection, the search for water region is done only in detected ground region.

6.6.2 Standing Water Body Detection

Next, the developed water body detected is tested to detect standing water. Figure 6.14 shows the average error of detected water region versus the distance from the camera. Similar to running water detection, the average errors are below 10 percent for the first 7 meters.

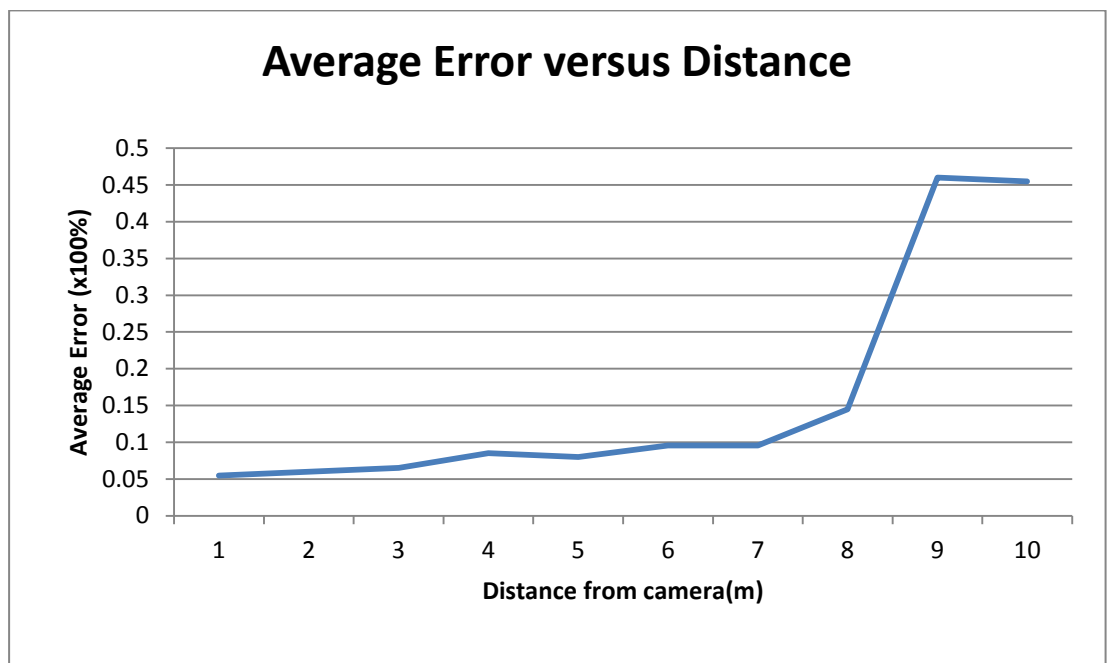


Figure 6.14: Average standing water body detection error versus distance. The number of samples used is thirty (30).

Similar to running water scene, developed algorithm managed to keep false detection to be lower than 10%. In standing water detection, the scene is more susceptible to reflection of the environment surrounding the water body. The low-texture cue plays important part in detecting standing water as the

reflection usually are lower in texture. Figure 6.15 shows the overall performance of the proposed algorithm in detecting standing water region.

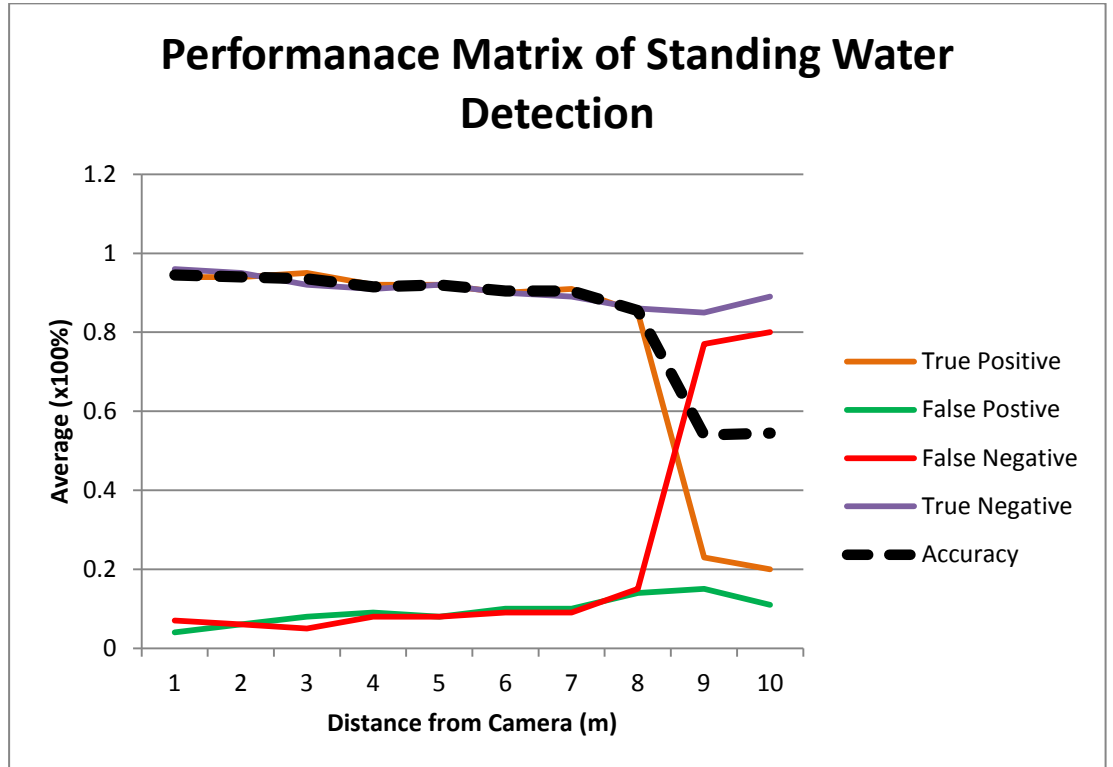


Figure 6.15: Performance matrix of the standing water detection module. The number of samples used is thirty (30).

Figure 6.17 shows samples results of the water body detection algorithm. The blue region corresponds to the water region while the red region corresponds to ground region. Note that the search of water region is considered only in detected ground region as it is assumed that water region only present on the ground. By considering only the ground region, the algorithm will not include the sky region in the result as sky region usually appear similar to sky reflection on water region.

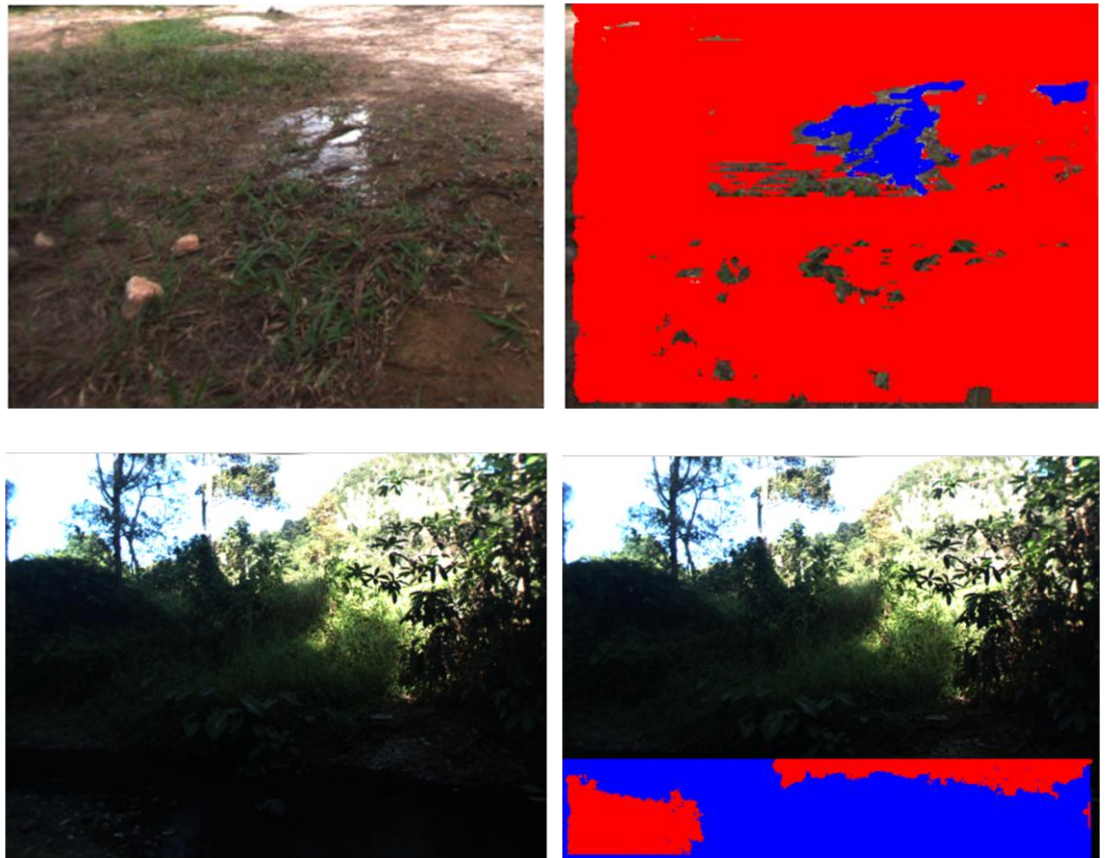


Figure 6.16: Sample results of the detected standing water using proposed water body detection.

6.6.3 Back-Lighting Error

This section address the problem of pixel over-saturated for the developed ground plane detection and water body detection algorithms. Pixel saturation is the event where incident light at a pixel causes the color channels of the camera sensor to respond at its maximum value. The frames pixels lose color information of the scene when the scene is illuminated with high intensity of light. While it is not in the scope of this thesis to solve this problem, it is

important to highlight this error to see how it affects the results of our developed algorithm.

This problem is likely to occur when the illuminant has a strong intensity and it is a problem associated with digital imaging. In outdoor terrain, this problem can occur in circumstances where the illuminant is in front of the camera. Saturation of pixels is particularly noticeable as the pixels appear to be achromatic or very bright (close to white) under direct illumination. The auto gain control (AGC) of the stereo camera will automatically adjust the gain of the sensors, resulting in darkening of regions not directly under the illuminant. The regions under canopy or shadow will appear to be very dim (close to black), consequently affecting the color information as well.

When saturation happens, it affects the clustering component in our developed algorithm. The color information is affected since the image frames are captured in RGB color space. The brighter region will appear to be very bright and the darker region will appear to be very dark as the regions suffer from narrow brightness ranges at low and high range. Consequently, the color distribution of the image will be very close and hard to be segregated during clustering step.

6.7 Summary

The algorithm produced good results in the condition in most of the rainforest terrain condition. It was able to detect the water region in the scene and mapped the corresponding ground pixels to the image.

Based on results of three experiments performed in simple unstructured terrain, rainforest terrain and complex terrain, the average result is presented in Figure 6.17. Based on the results, the average error over 90 samples is less than 10 percent within 8 meters from the camera.

We managed to maintain the average accuracy of 90 percent over the standing water and running water tests. The false positive outcome is very critical as this condition is hazardous to the vehicle. In this thesis, the developed algorithm managed to maintain the false positive alarm below 10 percent for most of the conditions. While the water body detection module can detect the presence of the water body, it cannot detect the depth of the water body. Until this point, all detected water regions are deemed to be non-traversable.

The proposed algorithm can detect most of the water appearances provided the condition is suitable for the stereo camera to be operational. To be effective, the scene needs to be sufficiently illuminated. The scene must not be neither too dark nor heavily illuminated with the source (sun in this case) not directly shining towards the camera. The color information will be lost when

the scene is dark, resulting in poor contrast and subsequently affecting the stereo correspondence. If the source of illumination is too bright and in front of the stereo camera, the image pixels will be oversaturated. Thus it can be said the performance of the proposed algorithm is limited by the sensor of the stereo camera.

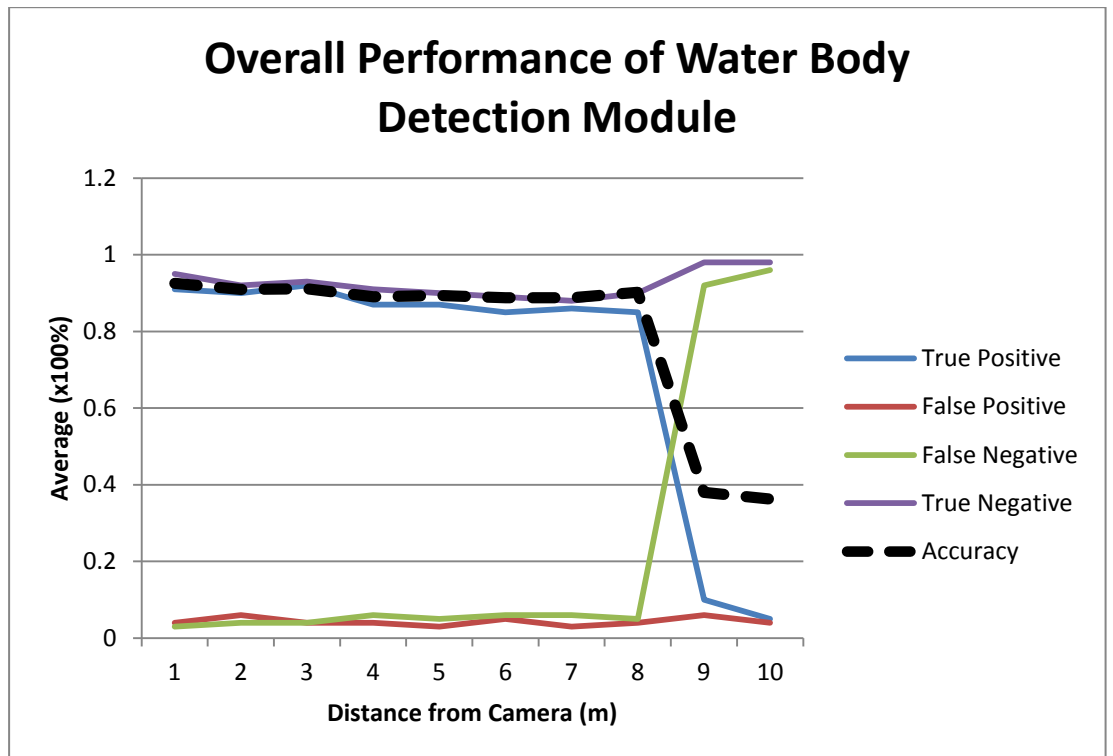


Figure 6.17: Overall performance matrix of the water detection module. The number of samples used is sixty (60).

The utilization of polarizer with stereo camera shows promising results that yet to be fully utilized. In this thesis, two different polarization angles are used to detect the water body. The degrees (0° and 90°) are chosen based on the maximum disparity changes when the polarizer is applied. However,

different polarization angles may be used to represent different reflection on the water body. Therefore, more usage of polarization angles can be considered in future research scopes.

CHAPTER 7

CONCLUSIONS

The work described in this thesis proposes near-range visual guidance method for rainforest terrain. The entire scheme is designed to detect ground detection, tree trunks detection and water body detection based on stereo camera.

In this chapter, the contributions of this thesis are summarized, followed by conclusions and lastly future works to be done.

7.1 Contributions

In general, the objectives of this scope are to solve the problem pertaining in visual guidance system in rainforest terrain. In particular, we identified that the essential components are the ground plane detection, tree trunks detection and water body detection. In order to successfully identify ground plane and water body detection, two methods based on stereo image processing are developed in this thesis.

- In Chapter 4, color feature and stereo disparity data are used together for the ground plane extraction from the image with rainforest scene. Using the K-means clustering, the color feature is used to segregate the image into few possible regions. The K-means

clustering method is used as it is easy to implement and fast in computational time which are a requirement in real-time implementation. Although K-means clustering is less accurate, it is sufficient for the developed module as we need only the initial estimate of the segmentation. The clusters produced from the K-means clustering method are used to generate multiple V-disparity images. Then, multiple correlation profiles are generated. The profile with the lowest gradient is chosen to be ground correlation profile. The ground correlation profile is used to determine the ground region in the frame. The ease of determining the ground correlation profile without any distinct man-made feature made the developed method suitable for rainforest guidance. This method overcomes the limitation of having to detect the man-made feature in V-disparity method.

- In Chapter 5, the near vertical tree detection module follows closely method by Huetas et al (2005) with additional U-disparity data to detect the tree trunks. The assumptions made are the tree trunks are tall and vertical or near vertical. The complex terrain in this project cannot be analyzed in a single stage of stereo vision research works. It is suggested to be scope down this works focus on the near vertical tree trunks obstacles. Based on the assumption, the edges of the tree trunks are extracted using Sobel edge detector. The U-disparity is used to accumulate the disparity with similar value as each region in a scene usually appears to be similar in disparity

value if it belongs to the same object. The detected edges are combined with the disparity information from U-disparity to extract the tree trunk fragment. This method is suitable to extract the tree trunks as in rainforest terrain; the tree trunks are tall, resulting in high accumulation of disparity. Consequently, the tree trunks can be extracted by using high disparity threshold.

- After the ground plane is successfully detected, Chapter 6 presents the water body detection algorithm. The developed method utilized multiple features in texture, disparity and partial polarization. Stereo camera with different polarization angles are applied on each side of the stereo camera. The intention of using polarizer is to remove specular reflection from the water body on one side of the camera and create two dissimilar appearance of the water body. Based on experiment conducted, it is shown that water body has big changes in brightness when different polarization angles are applied. Based on the difference, we can determine the water region due to failure to correlate between the two images. Coupled with low-texture characteristic and sky reflection detection, most of the water region can be detected. Through experimental results, the algorithm is demonstrated to be able to detect standing and running water in most condition except while the illumination source is in front of the camera.

7.2 Conclusion Remarks

In conclusion, it can be stated that the proposed short range visual guidance scheme is suitable to determine the ground plane, detect the tree trunk and spot the water body.

In this thesis, the proposed methods show satisfactory offline results through experimental testing under different situations. However, there are also limitations encountered in these methods:

- Due to hardware settings issue, the experiment is limited to single frame testing of the scene. There are no optimization to reduce the computational load due to frames processing (each frames are processed separately and treat independently).
- The algorithms do not solve the illumination problem which may affect the camera performance. The algorithm may fail mainly due to the image condition caused by excessive illumination in front of the camera.
- The ground plane detection module does not take into account negative obstacles. It may fail to detect the negative obstacle if the negative obstacles are not large enough or when the camera pitch angle low (with reference from negative-x axis).

- The compressible vegetation is not considered in this thesis. The compressible vegetation is deemed non-traversable if the height is above certain threshold.
- The water body detection module focuses in detecting the water region without considering the depth of the water body. Water body with very shallow depth is treated the same with deep water body.

7.3 Future works

As for future work, each part of the module of this visual guidance system should be improved to be able to work as complete visual guidance system as this thesis focuses on solving specific challenges in rainforest terrain.

- High level of artificial intelligence scheme can be developed to supply to make this work more accurate and robust. A system with prior knowledge of the scene will help the scene to classify the terrain better. The artificial intelligent scheme will be able to act as additional cue to increase the confidence level of the results. In addition, the artificial intelligent scheme can be used as a classifier to classify whether the terrain is traversable or not.
- In order to achieve more accurate result and robust, the illumination compensation has to be improved. Until this point, this thesis does not include any compensation algorithm except using HSI color

space to minimize the uneven illumination. With the illumination compensation, the color constancy problem can be minimized, enabling more accurate region boundaries of the terrain.

- The ground plane profile detected in this thesis assumes that the ground profile is linear (or single profile). Thus the ground correlation profiles in this thesis only true for near-range. It is not able to track the ground plane at far range if the slope of the ground plane is going downhill (relatively high down-slope) until it is near to the slope. In addition, a ground region with higher slope may be missed due to different slope profile. Thus, there is need to detect non-linear ground profile.
- The tree trunk detection module is able to detect vertical and near vertical tree trunks in the rainforest terrain. The assumption is that all tree trunks are not traversable regardless of the size or diameter of the tree trunks (the logical deduction is we usually do not run-over very tall object even it is compressible). However, the diameter estimation can be addition to determine the traversability. In addition, the algorithm needs to be improved to be able to detect tree trunks that are slanted and not near vertical.
- Generally, there are occlusion problems caused by partial or complete overlap of multiple objects in the scene. In tree trunk detection, tree trunks fragment with are assumed to be of the same

tree trunks if it belong to the same column of the image and has similar disparity value. However, this assumption cannot be used at the condition where there other fragment is complete hidden (particularly hazardous if it is at the base of the tree). We isolate this problem to certain extends by labeling this region as not-classified. However, there is a need to solve the occlusion problems for an effective visual guidance system.

- The current water detection algorithm can only detect the water region. In fact, whether a water body is hazardous to the autonomous vehicle or not depend on the size of the water body and also the depth of the water body. Hence, it is essential to further investigate the polarization method to see whether it will be able to determine the depth of the water body.
- The range of the visual guidance system should be extended in order for vehicle path planning to be feasible. In this thesis, stereo camera with fixed baseline is used, limiting the detection range. We propose the use of multiple stereo cameras with different baseline or stereo camera with real-time adjustable base line to detect different range of the terrain.
- The near-range visual guidance system is developed for the autonomous vehicle navigation in rainforest terrain. The developed algorithm is tested only on image frames from rainforest terrain.

Thus, the system needs to be implemented and tested in real-time autonomous vehicle.

AUTHOR'S PUBLICATION

1. CheeWay Teoh, ChingSeong Tan and Yong Chai Tan. (2010), "Preliminary Study on Visual Guidance System for Autonomous Vehicle in Rainforest Terrain", IEEE Conference on Robotics Automation and Mechatronics (RAM), 28-30 June 2010, page(s): 403-408 .
2. CheeWay Teoh, ChingSeong Tan and Yong Chai Tan. (2010), "Ground plane detection for autonomous vehicle in rainforest terrain", 2010 IEEE Conference Sustainable Utilization and Development in Engineering and Technology (STUDENT 2010), 20-21 Nov. 2010, page(s): 7 - 12.

REFERENCES

- Adams, M., Wijesoma, S. W., & Shacklock, A. (2007). Autonomous Navigation: Achievements in Complex Environment. *IEEE Instrumentation & Measurement Magazine*, (pp. 15-22).
- Alberts, J., Edwards, D., Soule, T., Anderson, M., & O'Rourke, M. (2008). Autonomous Navigation of an Unmanned Ground Vehicle in Unstructured Forest Terrain. *ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, (pp. 103-109).
- Batavia, P. H., Pomerleau, D. A., & Thope, C. E. (1996). *Applying Advanced Learning Algorithms to ALVINN*. Carnegie Mellon University: Technical Report CMU-RI-TR-96-31, Robotics Institute.
- Bellutta, P., Manduchi, R., Matthies, L., Owens, K., & Rankin, A. (2000). Terrain Perception for DEMO III. *Proceedings of the 2000 Intelligent Vehicles Conference*, (pp. 326-331).
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV*. USA: O'Reilly Media.
- Broggi, A. (1999). The ARGO Autonomous Vehicle's Vision and Control Systems. *International Journal of Intelligent Control Systems*, 409-441.
- Broggi, A., Bertozzi, M., Fascioli, A., & Conte, G. (1999). *Automatic Vehicle Guidance: The Experience of the ARGO Autonomous Vehicle*. London: World Scientific.
- Broggi, A., Fascioli, A., & Bertozzi, M. (2000). Architectural Issues on Vision-Based Automatic Vehicle Guidance: The Experience of the ARGO Project. *Real-Time Imaging*, 313-324.
- Brown, M. Z., Burschka, D., & Hager, G. D. (2003). Advances in Computational Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 993-1008.

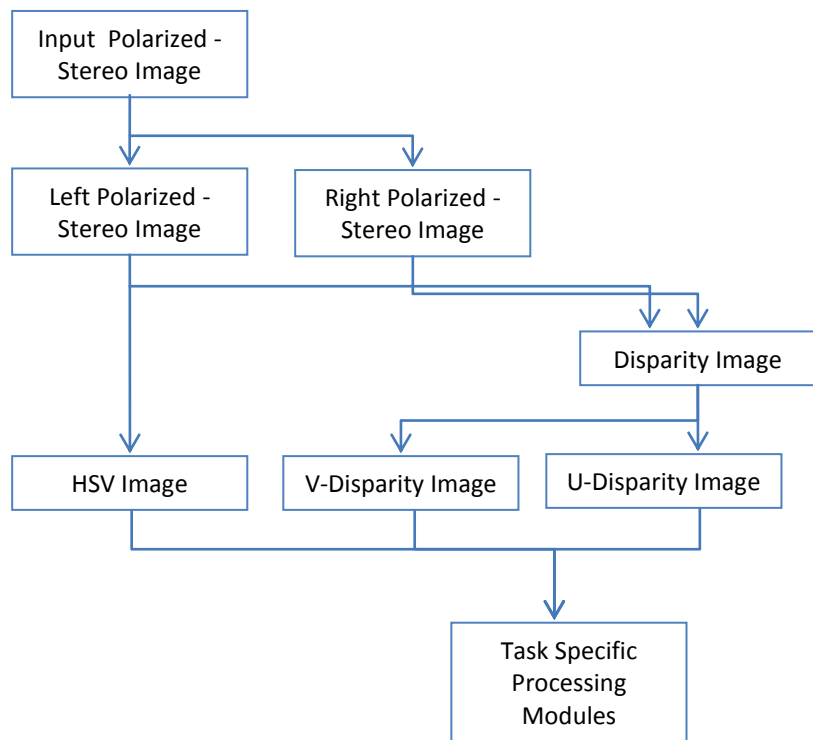
- DeSouza, G. N., & Kak, A. C. (2002). Vision for Mobile Robot Navigation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 237-267.
- Dickmanns, E. D. (2002). The Development of Machine Vision for Road Vehicles in the Last Decade. *Proceedings of IEEE Intelligent Vehicle Symposium*, (pp. 268-281). Versailles, France.
- Eicker, P. J. (2001). *The Embudito Mission: A Case Study of the Systematics of Autonomous Ground Mobile Robots*. United States of America: Sandia National Laboratories.
- Gennery, D. B. (1977). A Stereo Vision System for an Autonomous Vehicle. *International Joint Conference On Artificial Intelligence*.
- Goldberg, S. B., Maimone, M. W., & Matthies, L. (2002). Stereo Vision and Rover Navigation Software for Planetary Exploration. *IEEE Aerospace Conference Proceedings*.
- Goldberg, S. B., Maimone, M. W., & Matthies, L. (2002). Stereo Vision and Rover Navigation Software for Planetary Exploration. *IEEE Aerospace Conference Proceedings*.
- Gonzalez, R. C., & Woods, R. E. (2010). *Digital Image Processing, 3rd Edition*. New Jersey: Pearson Education Inc.
- Happold, M., & Ollis, M. (2006). Autonomous Learning of Terrain Classification within Imagery for Robot Navigation. *2006 IEEE International Conference on Systems, Man, and Cybernetics*, (pp. 260-266). Taipei.
- Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. United Kingdom: Cambridge University Press.
- Huertas, A., Matthies, L., & Rankin, A. (2005). Stereo-Based Tree Traversability Analysis for Autonomous Off-Road Navigation. *Proc. IEEE Work-shop on Applications of Computer Vision*. Breckenridge.

- Jochem, T. M. (1994). *Using Visual Active Vision Tools to Improve Autonomous Driving Task*. Carnegie Mellon University: Technical Report CMU-RI-TR-94-39.
- Jochem, T. M., Pomerleau, D. A., & Thorpe, C. E. (1995). Vision-Based Neural Network Road and Intersection Detection and Traversal. *International Conference on Intelligent Robots and Systems-Volume 3*, (pp. 344-349).
- Kelly, A., Amidi, O., Happold, M., Herman, H., Pilarski, T., Rander, P., et al. (2004). Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments. *International Symposium on Experimental Robotics*. Singapore.
- Koshikawa, K. (1979). A polametric approach to shape understanding. *Proceeding IJCAI*, (pp. 493-495).
- Krotkov, E., Fish, S., Jackel, L., McBride, B., Perschbacher, M., & Pippine, J. (2006). The DARPA PerceptOR Evaluation Experiments. *Auton Robot*, 19-35.
- Labayrade, R., Aubert, D., & Tarel, J. P. (2002). Real Time Obstacle Detection in Stereo Vision on Non Flat Road Geometry Through "V-disparity". *IEEE Intelligent Vehicle Symposium*.
- Manduchi, R., Castano, A., Talukder, A., & Matthies, L. (2005). Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. *Autonomous Robots 18*, Autonomous Robots 18.
- Nayar, S. K., Fang, X.-S., & Boulton, T. (1993). Removal of Specularities Using Color and Polarization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (pp. 583-590).
- Nilsson, J. N. (1969). A Mobius Automaton: An Application of Artificial Intelligence. *International Joint Conference on Artificial Intelligence*. Washington.
- Pedrotti, F. L., Pedrotti, L. M., & Pedrotti, L. S. (2007). *Introduction to Optics, 3rd Edition*. New Jersey: Pearson Education Inc.

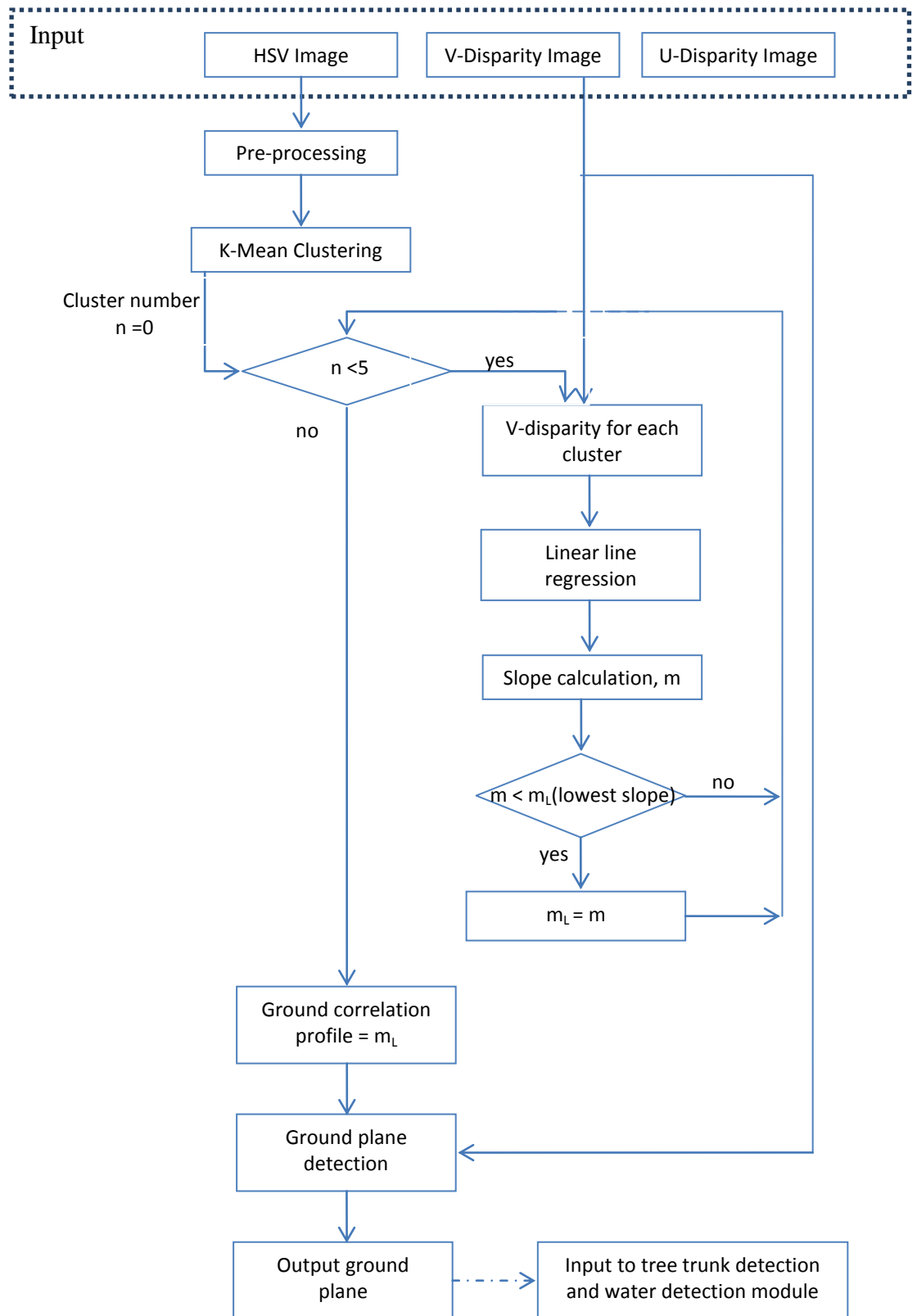
- Point Grey Research Inc. (2003). Triclops StereoVision System Manual Version 3.1.
- Pomerleau, D. A. (1989). *ALVINN: An Autonomous Land Vehicle In a Neural Network*. Carnegie Mellon University.
- Rankin, A. L., Bergh, C. F., Goldberg, S. B., Bellutta, P., Huertas, A., & Matthies, L. H. (2005). Passive Perception System for Day/Night Autonomous Off-Road Navigation. *SPIE Defense and Security Symposium: Unmanned Ground Vehicle Technology VI Conference*, (pp. 343-353). Orlando.
- Shacklock, A., Xu, J., & Wang, H. (2006). Visual Guidance for Autonomous Vehicles: Capability and Challenges. In S. Sam Ge, & F. L. Lewis, *Autonomous Mobile Robot: Sensing, Control, Decision Making and Applications* (pp. 5-40). USA: CRC Press.
- Shoemaker, C. M., & Bornstein, J. A. (1998). The Demo III UGV Program: A Testbed for Autonomous Navigation Research. *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, (pp. 644-651). Gaithersburg.
- Stentz, A., Kelly, A., Rander, P., Herman, H., Amidi, O., Mandelbaum, R., et al. (2003). Real-Time, Multi-Perspective Perception for Unmanned Ground Vehicles. *Proceeding Association for Unmanned Vehicle Systems International*. Baltimore.
- Thorpe, C., Hebert, M. H., Kanade, T., & Shafer, S. A. (1988). Vision and Navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, 362-373.
- Waterman, T. H. (1981). Polarization Sensitivity. In H. J. Altrum, *Handbook of Sensory Physiology*, Vol. 7. Berlin: Springer Verlag.
- Wolff, L. B. (1997). Polarization Vision: a new sensory approach to image understanding. *Image and Vision Computing* Vol. 15, (pp. 81-93).
- Wolff, L. B., & Boulton, T. (1991). Constraining Object Features using Polarization Reflectance Model. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, Vol 13, No. 7, (pp. 635-657).

APPENDICES

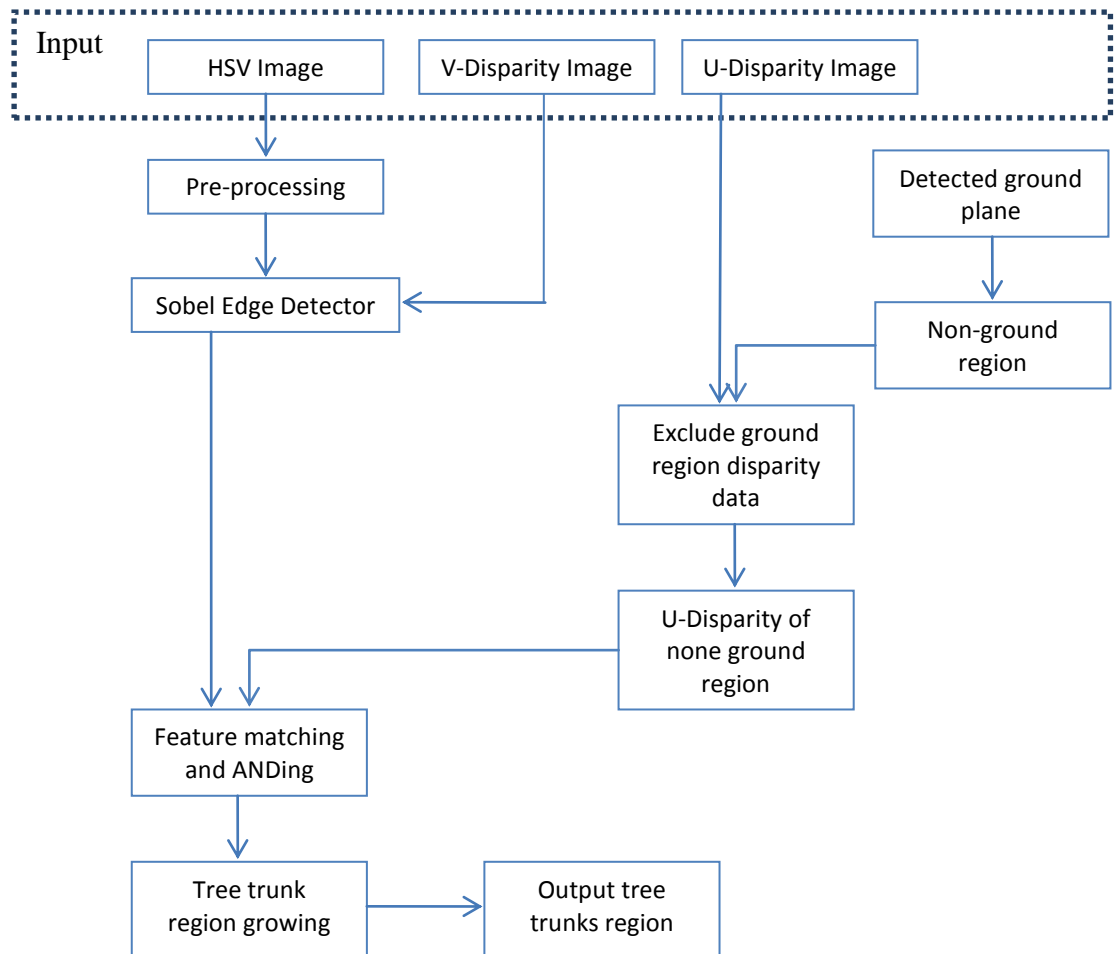
APPENDIX A



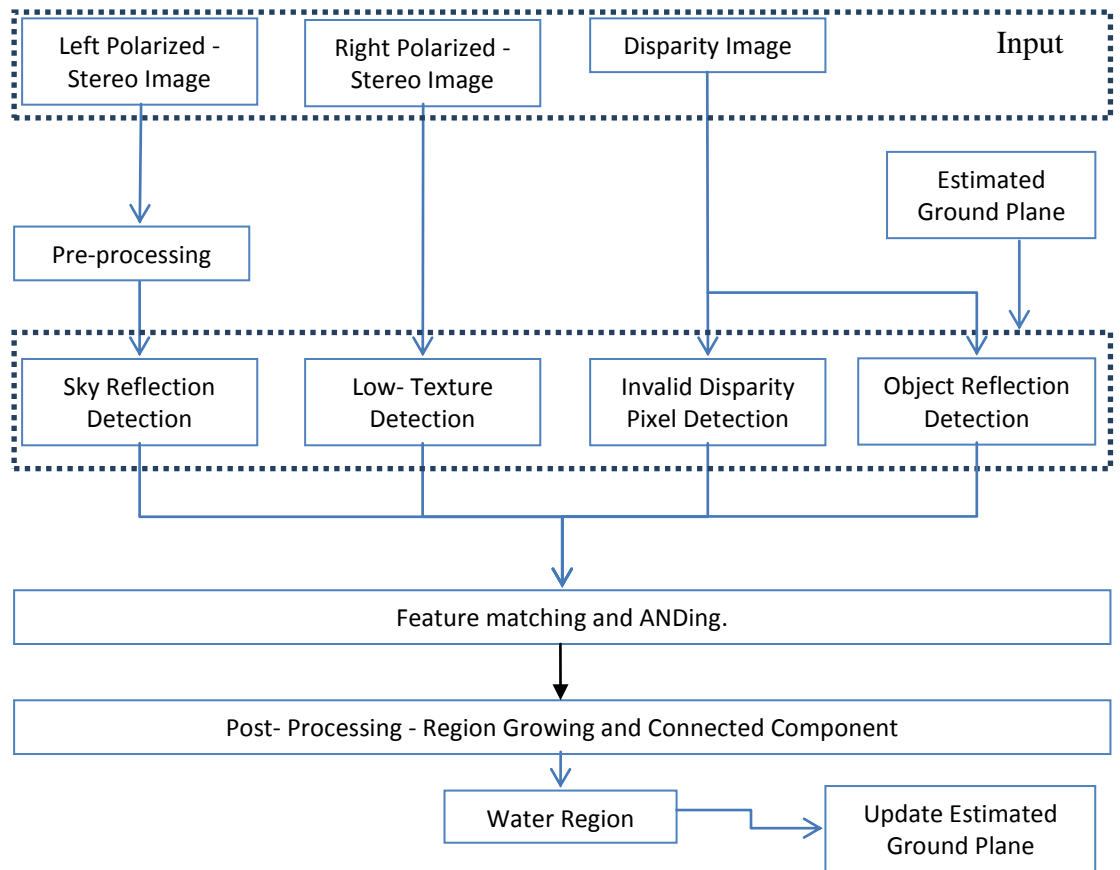
Appendix A.1: Proposed system architecture image acquisition and feature extraction.



Appendix A.2: Proposed system architecture for ground plane detection.



Appendix A.3: Proposed system architecture for tree trunk detection.



Appendix A.4: Proposed system architecture for water body detection.

APPENDIX B

```
#include "StdAfx.h"
#include "classBumbleContext.h"

#pragma region Constructor and Destructor
classBumbleContext::classBumbleContext()
{ // constructor
}
classBumbleContext::~classBumbleContext(void)
{ // destructor
}
#pragma endregion

#pragma region Load Calibration File
void classBumbleContext::load_calibrationFile(char* fileName)
{
    triclopsGetDefaultContextFromFile( &triclopsContext,
fileName);
    //return triclopsContext;
}

void classBumbleContext::load_calibrationFile_default()
{
    triclopsGetDefaultContextFromFile( &triclopsContext,
"input.cal" );
    //return triclopsContext;
}
#pragma endregion
#pragma region GRAB PPM
void classBumbleContext::grab_ppmImage(char* fileName)
{
    ppmReadToTriclopsInput( fileName, &m_tiStereo );
    pgmReadToTriclopsInput( fileName, &m_tiStereoRGB );

    // -----
    // Generationg RAW Stereo, Left & Right OpenCV Image ----
    // -----

    IplImage* img_ipl_RGBU =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows),8,4);
    img_ipl_RGBx = cvCreateImage(cvSize(m_tiStereo.ncols,
m_tiStereo.nrows),8,3);

    IplImage* img_ipl_red =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows),8,1);
    IplImage* img_ipl_green =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows),8,1);
    IplImage* img_ipl_blue =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows),8,1);
    IplImage* img_ipl_test =
cvCreateImage(cvSize(m_tiStereo.ncols,
m_tiStereo.nrows),8,1);/////

    img_ipl_RGBU->imageData =
(char*)m_tiStereo.u.rgb32BitPacked.data;
```

```

        cvSplit(img_ipl_RGBU, img_ipl_blue, img_ipl_green,
img_ipl_red, img_ipl_test);////////
        cvMerge(img_ipl_blue, img_ipl_green, img_ipl_red, NULL,
img_ipl_RGBx);

        m_tiStereo.ncols = m_tiStereo.ncols/2;
        m_tiStereo.timeStamp.sec = 0;
        m_tiStereo.timeStamp.u_sec = 0;

        // -----
-----
        // Build triclops input: m_tiRawColorImages[4]
        img_ipl_raw_colour_left =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,3);
        img_ipl_raw_colour_right =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,3);

        for( int y=0; y<(img_ipl_RGBx->height); y++ )
        { //To split RAW into LEFT and RIGHT RAW
            uchar* ptr_oriRight = (uchar*) (img_ipl_RGBx-
>imageData + y * img_ipl_RGBx->widthStep );
            uchar* ptr_oriLeft = (uchar*) (img_ipl_RGBx-
>imageData + img_ipl_RGBx->widthStep/2 + y * img_ipl_RGBx-
>widthStep );

            uchar* ptrRight = (uchar*)
(img_ipl_raw_colour_right->imageData + y* img_ipl_RGBx-
>widthStep/2);
            uchar* ptrLeft = (uchar*) (img_ipl_raw_colour_left-
>imageData + y* img_ipl_RGBx->widthStep/2);

            for( int x=0; x<(img_ipl_RGBx->width); x++ )
            {
                ptrLeft[3*x]      = ptr_oriLeft[3*x];
                ptrLeft[3*x+1]    = ptr_oriLeft[3*x+1];
                ptrLeft[3*x+2]    = ptr_oriLeft[3*x+2];

                ptrRight[3*x]     = ptr_oriRight[3*x];
                ptrRight[3*x+1]   = ptr_oriRight[3*x+1];
                ptrRight[3*x+2]   = ptr_oriRight[3*x+2];
            }
        }

        // -----
-----
        // Build triclops input: m_tiRawColorImages[4]

        //[0] = right
        m_tiRawColorImages[0].ncols = m_tiStereo.ncols; // 1024
        m_tiRawColorImages[0].nrows = m_tiStereo.nrows; // 768
        m_tiRawColorImages[0].rowinc = m_tiStereo.rowinc/2; //
4096
        m_tiRawColorImages[0].inputType =
TriInp_RGB_32BIT_PACKED;

        //[2] = left
        m_tiRawColorImages[2].ncols = m_tiStereo.ncols; // 2048
        m_tiRawColorImages[2].nrows = m_tiStereo.nrows; // 768
        m_tiRawColorImages[2].rowinc = m_tiStereo.rowinc/2; //
4096

```

```

        m_tiRawColorImages[2].inputType =
TriInp_RGB_32BIT_PACKED;

        m_tiRawColorImages[0].u.rgb32BitPacked.data = new
unsigned
char[m_tiRawColorImages[0].ncols*m_tiRawColorImages[0].nrows
*4];
        m_tiRawColorImages[2].u.rgb32BitPacked.data = new
unsigned
char[m_tiRawColorImages[2].ncols*m_tiRawColorImages[2].nrows
*4];

        uchar* ptrTri_right =
(uchar*)m_tiRawColorImages[0].u.rgb32BitPacked.data;
        uchar* ptrTri_left =
(uchar*)m_tiRawColorImages[2].u.rgb32BitPacked.data;

        for( int y=0; y<(img_ipl_RGBx->height); y++ )
        {
                uchar* ptr_right = (uchar*)
(img_ipl_raw_colour_right->imageData + y *
img_ipl_raw_colour_right->widthStep );
                uchar* ptr_left = (uchar*)
(img_ipl_raw_colour_left->imageData + y *
img_ipl_raw_colour_left->widthStep );

                uchar* ptrTriclopsInput_right = (uchar*)
(ptrTri_right + y*m_tiRawColorImages[0].rowinc );
                uchar* ptrTriclopsInput_left = (uchar*)
(ptrTri_left + y*m_tiRawColorImages[2].rowinc );

                for( int x=0; x<(img_ipl_RGBx->width/2); x++ )
                {
                        ptrTriclopsInput_right[4*x] = ptr_right[3*x];
                        ptrTriclopsInput_right[4*x+1] =
ptr_right[3*x+1];
                        ptrTriclopsInput_right[4*x+2] =
ptr_right[3*x+2];

                        ptrTriclopsInput_left[4*x] = ptr_left[3*x];
                        ptrTriclopsInput_left[4*x+1] =
ptr_left[3*x+1];
                        ptrTriclopsInput_left[4*x+2] =
ptr_left[3*x+2];
                }
        }

}
#pragma endregion
#pragma region GRAB JPEG
void classBumbleContext::grab_jpeg(char* fileName)
{
        // -----
        ----- // Generationg RAW Stereo, Left & Right OpenCV Image -----
        -----

        img_ipl_RGBx = cvLoadImage(fileName);

        IplImage* img_ipl_RGBU =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx-
>height), 8, 4);

```

```

        IplImage* img_ipl_red =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx-
>height),8,1);
        IplImage* img_ipl_green =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx-
>height),8,1);
        IplImage* img_ipl_blue =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx-
>height),8,1);

        cvSplit(img_ipl_RGBx, img_ipl_blue, img_ipl_green,
img_ipl_red, NULL);
        cvMerge(img_ipl_blue, img_ipl_green, img_ipl_red, NULL,
img_ipl_RGBU);

        -----
        //-----
        // Allocate memory for m_tiStereo
ppmReadToTriclopsInput( "dummyImage.ppm", &m_tiStereo );
pgmReadToTriclopsInput( "dummyImage.ppm",
&m_tiStereoRGB);

        m_tiStereo.u.rgb32BitPacked.data = img_ipl_RGBU-
>imageData;
        m_tiStereo.u.rgb.red = img_ipl_RGBU->imageData;
        m_tiStereo.u.rgb.green = img_ipl_RGBU->imageData;
        m_tiStereo.u.rgb.blue = img_ipl_RGBU->imageData;
        //-----
        -----

        m_tiStereo.ncols = img_ipl_RGBU->width/2;
        m_tiStereo.nrows = img_ipl_RGBU->height;
        //m_tiStereo.rowinc = img_ipl_RGBU->widthStep/4;
        m_tiStereo.timeStamp.sec = 0;
        m_tiStereo.timeStamp.u_sec = 0;
        //m_tiStereo.inputType = TriInp_RGB;

        -----
        ///// -----
        -----
        ///// Build triclops input: m_tiRawColorImages[4]
img_ipl_raw_colour_left =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,3);
img_ipl_raw_colour_right =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,3);

        IplImage* img_ipl_raw_colour_left_m_tiStereoRGB =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,1);
        IplImage*img_ipl_raw_colour_right_m_tiStereoRGB =
cvCreateImage(cvSize(m_tiStereo.ncols, m_tiStereo.nrows ),8,1);

        for( int y=0; y<(img_ipl_RGBx->height); y++ )
        { //To split RAW into LEFT and RIGHT RAW
            uchar* ptr_oriRight = (uchar*) (img_ipl_RGBx-
>imageData + y * img_ipl_RGBx->widthStep );
            uchar* ptr_oriLeft = (uchar*) (img_ipl_RGBx-
>imageData + img_ipl_RGBx->widthStep/2 + y * img_ipl_RGBx-
>widthStep );

```

```

        uchar* ptrRight = (uchar*)
(img_ipl_raw_colour_right->imageData + y* img_ipl_RGBx-
>widthStep/2);
        uchar* ptrLeft = (uchar*) (img_ipl_raw_colour_left-
>imageData + y* img_ipl_RGBx->widthStep/2);

        uchar* ptrRight_RGB = (uchar*)
(img_ipl_raw_colour_right_m_tiStereoRGB->imageData + y*
img_ipl_RGBx->widthStep/2);
        uchar* ptrLeft_RGB = (uchar*)
(img_ipl_raw_colour_left_m_tiStereoRGB->imageData + y*
img_ipl_RGBx->widthStep/2);

        for( int x=0; x<(img_ipl_RGBx->width); x++ )
        {
            ptrLeft[3*x]      = ptr_oriLeft[3*x];
            ptrLeft[3*x+1]    = ptr_oriLeft[3*x+1];
            ptrLeft[3*x+2]    = ptr_oriLeft[3*x+2];

            ptrRight[3*x]     = ptr_oriRight[3*x];
            ptrRight[3*x+1]   = ptr_oriRight[3*x+1];
            ptrRight[3*x+2]   = ptr_oriRight[3*x+2];

            ptrLeft_RGB[1*x]   = ptr_oriLeft[3*x];
            ptrRight_RGB[x];//  = ptr_oriRight[3*x];
        }
    }

// -----
-----
// Build triclops input: m_tiRawColorImages[4]
//[0] = right
m_tiRawColorImages[0].ncols = m_tiStereo.ncols; // 1024
m_tiRawColorImages[0].nrows = m_tiStereo.nrows; // 768
m_tiRawColorImages[0].rowinc = m_tiStereo.rowinc/2; //
4096
m_tiRawColorImages[0].inputType =
TriInp_RGB_32BIT_PACKED;
//[1] = center
//m_tiRawColorImages[1].ncols = imageCols; // 2048
//m_tiRawColorImages[1].nrows = imageRows; // 768
//m_tiRawColorImages[1].rowinc = imageRowInc*2; // 8192
//m_tiRawColorImages[1].inputType =
TriInp_RGB_32BIT_PACKED;
//[2] = left
m_tiRawColorImages[2].ncols = m_tiStereo.ncols; // 2048
m_tiRawColorImages[2].nrows = m_tiStereo.nrows; // 768
m_tiRawColorImages[2].rowinc = m_tiStereo.rowinc/2; //
4096
m_tiRawColorImages[2].inputType =
TriInp_RGB_32BIT_PACKED;
//[3] = both
//m_tiRawColorImages[3].ncols = imageCols*2; // 2048
//m_tiRawColorImages[3].nrows = imageRows; // 768
//m_tiRawColorImages[3].rowinc = imageRowInc*4; // 8192
//m_tiRawColorImages[3].inputType =
TriInp_RGB_32BIT_PACKED;

m_tiRawColorImages[0].u.rgb32BitPacked.data = new
unsigned

```



```

char[m_tiRawColorImages[0].ncols*m_tiRawColorImages[0].nrows
*4];
    m_tiRawColorImages[2].u.rgb32BitPacked.data = new
unsigned
char[m_tiRawColorImages[2].ncols*m_tiRawColorImages[2].nrows
*4];

    uchar* ptrTri_right =
(uchar*)m_tiRawColorImages[0].u.rgb32BitPacked.data;
    uchar* ptrTri_left =
(uchar*)m_tiRawColorImages[2].u.rgb32BitPacked.data;

    for( int y=0; y<(img_ipl_RGBx->height); y++ )
    {
        uchar* ptr_right = (uchar*)
(img_ipl_raw_colour_right->imageData + y *
img_ipl_raw_colour_right->widthStep );
        uchar* ptr_left = (uchar*)
(img_ipl_raw_colour_left->imageData + y *
img_ipl_raw_colour_left->widthStep );

        uchar* ptrTriclopsInput_right = (uchar*)
(ptrTri_right + y*m_tiRawColorImages[0].rowinc );
        uchar* ptrTriclopsInput_left = (uchar*)
(ptrTri_left + y*m_tiRawColorImages[2].rowinc );

        for( int x=0; x<(img_ipl_RGBx->width/2); x++ )
        {
            ptrTriclopsInput_right[4*x] = ptr_right[3*x];
            ptrTriclopsInput_right[4*x+1] =
ptr_right[3*x+1];
            ptrTriclopsInput_right[4*x+2] =
ptr_right[3*x+2];

            ptrTriclopsInput_left[4*x] = ptr_left[3*x];
            ptrTriclopsInput_left[4*x+1] =
ptr_left[3*x+1];
            ptrTriclopsInput_left[4*x+2] =
ptr_left[3*x+2];
        }
        m_tiStereoRGB.u.rgb.blue =
img_ipl_raw_colour_right_m_tiStereoRGB->imageData ;
        m_tiStereoRGB.u.rgb.green =
img_ipl_raw_colour_left_m_tiStereoRGB->imageData;
        m_tiStereoRGB.u.rgb.red =
img_ipl_raw_colour_right_m_tiStereoRGB->imageData;
    }
#pragma endregion
#pragma region AVI Video
void classBumbleContext::load_avi(char* fileName)
{
    //cvNamedWindow( "Example2", CV_WINDOW_AUTOSIZE );
    CvCapture* capture = cvCreateFileCapture( fileName );
    IplImage* frame;
    while(1)
    {
        frame = cvQueryFrame( capture );
        if( !frame ) break;
        grab_frame(frame);
    }
}

```

```

}
#pragma endregion

#pragma region GRAB FRAME
void classBumbleContext::grab_frame(IplImage* frame)
{
    // -----
    // Generationg RAW Stereo, Left & Right OpenCV Image ----
    // -----
    img_ipl_RGBx = frame;
    IplImage* img_ipl_RGBU =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx->
height),8,4);

    IplImage* img_ipl_red =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx->
height),8,1);
    IplImage* img_ipl_green =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx->
height),8,1);
    IplImage* img_ipl_blue =
cvCreateImage(cvSize(img_ipl_RGBx->width, img_ipl_RGBx->
height),8,1);

    cvSplit(img_ipl_RGBx, img_ipl_blue, img_ipl_green,
img_ipl_red, NULL);
    cvMerge(img_ipl_blue, img_ipl_green, img_ipl_red, NULL,
img_ipl_RGBU);

    //-----
    // Allocate memory for m_tiStereo
    m_tiStereo.u.rgb32BitPacked.data = new unsigned
char

```

```

        uchar* ptr_oriLeft = (uchar*) (img_ipl_RGBx-
>imageData + img_ipl_RGBx->widthStep/2 + y * img_ipl_RGBx-
>widthStep );

        uchar* ptrRight = (uchar*)
(img_ipl_raw_colour_right->imageData + y* img_ipl_RGBx-
>widthStep/2);
        uchar* ptrLeft = (uchar*) (img_ipl_raw_colour_left-
>imageData + y* img_ipl_RGBx->widthStep/2);

        for( int x=0; x<(img_ipl_RGBx->width); x++ )
        {
            ptrLeft[3*x]      = ptr_oriLeft[3*x];
            ptrLeft[3*x+1]    = ptr_oriLeft[3*x+1];
            ptrLeft[3*x+2]    = ptr_oriLeft[3*x+2];

            ptrRight[3*x]     = ptr_oriRight[3*x];
            ptrRight[3*x+1]   = ptr_oriRight[3*x+1];
            ptrRight[3*x+2]   = ptr_oriRight[3*x+2];
        }
    }

    cvNamedWindow("Rectify GRAY: RGBx", CV_WINDOW_AUTOSIZE );
    cvShowImage( "Rectify GRAY: RGBx",
img_ipl_raw_colour_left );
    cvNamedWindow("Rectify GRAY: RGBU", CV_WINDOW_AUTOSIZE );
    cvShowImage( "Rectify GRAY: RGBU",
img_ipl_raw_colour_right );

    // -----
    // Build triclops input: m_tiRawColorImages[4]
    // [0] = right
    m_tiRawColorImages[0].ncols = m_tiStereo.ncols; // 1024
    m_tiRawColorImages[0].nrows = m_tiStereo.nrows; // 768
    m_tiRawColorImages[0].rowinc = m_tiStereo.rowinc/2; //
4096
    m_tiRawColorImages[0].inputType =
TriInp_RGB_32BIT_PACKED;
    // [1] = center
    // m_tiRawColorImages[1].ncols = imageCols; // 2048
    // m_tiRawColorImages[1].nrows = imageRows; // 768
    // m_tiRawColorImages[1].rowinc = imageRowInc*2; // 8192
    // m_tiRawColorImages[1].inputType =
TriInp_RGB_32BIT_PACKED;
    // [2] = left
    m_tiRawColorImages[2].ncols = m_tiStereo.ncols; // 2048
    m_tiRawColorImages[2].nrows = m_tiStereo.nrows; // 768
    m_tiRawColorImages[2].rowinc = m_tiStereo.rowinc/2; //
4096
    m_tiRawColorImages[2].inputType =
TriInp_RGB_32BIT_PACKED;
    // [3] = both
    // m_tiRawColorImages[3].ncols = imageCols*2; // 2048
    // m_tiRawColorImages[3].nrows = imageRows; // 768
    // m_tiRawColorImages[3].rowinc = imageRowInc*4; // 8192
    // m_tiRawColorImages[3].inputType =
TriInp_RGB_32BIT_PACKED;

    m_tiRawColorImages[0].u.rgb32BitPacked.data = new
unsigned

```

```

char[m_tiRawColorImages[0].ncols*m_tiRawColorImages[0].nrows
*4];
    m_tiRawColorImages[2].u.rgb32BitPacked.data = new
unsigned
char[m_tiRawColorImages[2].ncols*m_tiRawColorImages[2].nrows
*4];

    uchar* ptrTri_right =
(uchar*)m_tiRawColorImages[0].u.rgb32BitPacked.data;
    uchar* ptrTri_left =
(uchar*)m_tiRawColorImages[2].u.rgb32BitPacked.data;

    for( int y=0; y<(img_ipl_RGBx->height); y++ )
    {
        uchar* ptr_right = (uchar*)
(img_ipl_raw_colour_right->imageData + y *
img_ipl_raw_colour_right->widthStep );
        uchar* ptr_left = (uchar*)
(img_ipl_raw_colour_left->imageData + y *
img_ipl_raw_colour_left->widthStep );

        uchar* ptrTriclopsInput_right = (uchar*)
(ptrTri_right + y*m_tiRawColorImages[0].rowinc );
        uchar* ptrTriclopsInput_left = (uchar*)
(ptrTri_left + y*m_tiRawColorImages[2].rowinc );

        for( int x=0; x<(img_ipl_RGBx->width/2); x++ )
        {
            ptrTriclopsInput_right[4*x] = ptr_right[3*x];
            ptrTriclopsInput_right[4*x+1] =
ptr_right[3*x+1];
            ptrTriclopsInput_right[4*x+2] =
ptr_right[3*x+2];

            ptrTriclopsInput_left[4*x] = ptr_left[3*x];
            ptrTriclopsInput_left[4*x+1] =
ptr_left[3*x+1];
            ptrTriclopsInput_left[4*x+2] =
ptr_left[3*x+2];
        }
    }
}
#pragma endregion // Grabbing stereo image and frames
#pragma region GRAB RECTIFIED, DISPARITY, EDGE
void classBumbleContext::grab_grayscale(int
image_type_triclops, int image_triclops_leftRight, int
image_resolution_width, int image_resolution_height)
{
    // image_type_triclops; // 0 = Disparity, 1 = RAW, 2 =
Rectified, 3 = Edge
    triclops_preprocess_3D(image_resolution_width,
image_resolution_height);
    if (image_type_triclops == 0)
    {
        // Grab disparity Image
        triclopsGetImage( triclopsContext,
TriImg_DISPARITY, TriCam_REFERENCE, &img_triclops_disparity );
        IplImage* testing =
convert_Triclops2Ipl_1D(img_triclops_disparity);
        cvNamedWindow("Rectify GRAY: LEFT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "Rectify GRAY: LEFT", testing);
    }
}

```

```

else if (image_type_triclops == 1)
{
    // Grab RAW Image
    if (image_triclops_leftRight == 3)
    {
        // left
        triclopsGetImage( triclopsContext,
TriImg_RAW, TriCam_LEFT, &img_triclops_gray_RAW );
        IplImage* testing =
convert_Triclops2Ipl_1D(img_triclops_gray_RAW);
        cvNamedWindow("RAW: LEFT", CV_WINDOW_AUTOSIZE
);
        cvShowImage( "RAW: LEFT", testing);
    }
    else if (image_triclops_leftRight == 1)
    {
        // right
        triclopsGetImage( triclopsContext,
TriImg_RAW, TriCam_RIGHT, &img_triclops_gray_RAW );
        IplImage* testing =
convert_Triclops2Ipl_1D(img_triclops_gray_RAW);
        cvNamedWindow("RAW: RIGHT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "RAW: RIGHT", testing);
    }
}
else if (image_type_triclops == 3)
{
    // Grab EDGE Image

    if (image_triclops_leftRight == 3)
    {
        // left
        triclopsGetImage( triclopsContext,
TriImg_EDGE, TriCam_LEFT, &img_triclops_edge_left );
        IplImage* testing =
convert_Triclops2Ipl_1D(img_triclops_edge_left);
        cvNamedWindow("EDGE: LEFT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "EDGE: LEFT", testing);
    }
    else if (image_triclops_leftRight == 1)
    {
        // right
        triclopsGetImage( triclopsContext,
TriImg_EDGE, TriCam_RIGHT, &img_triclops_edge_right );
        IplImage* testing2 =
convert_Triclops2Ipl_1D(img_triclops_edge_right);
        cvNamedWindow("EDGE: RIGHT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "EDGE: RIGHT", testing2);
    }
}
}

void classBumbleContext::grab_color(int image_type_triclops,
int image_triclops_leftRight, int image_resolution_width, int
image_resolution_height)
{
    triclops_preprocess_3D(image_resolution_width,
image_resolution_height);
    if (image_type_triclops == 4)
    {
        if (image_triclops_leftRight == 3)
        {

```

```

        triclopsRectifyColorImage(triclopsContext,
TriCam_LEFT, &m_tiRawColorImages[2],
&img_triclops_rectified_colour_left);
        IplImage* img_ipl=
cvCreateImage(cvSize(img_triclops_rectified_colour_left.ncols,
img_triclops_rectified_colour_left.nrows),8,3);
        IplImage* img_ipl_red=
cvCreateImage(cvSize(img_triclops_rectified_colour_left.ncols,
img_triclops_rectified_colour_left.nrows),8,1);
        IplImage* img_ipl_green=
cvCreateImage(cvSize(img_triclops_rectified_colour_left.ncols,
img_triclops_rectified_colour_left.nrows),8,1);
        IplImage* img_ipl_blue=
cvCreateImage(cvSize(img_triclops_rectified_colour_left.ncols,
img_triclops_rectified_colour_left.nrows),8,1);
        img_ipl_rectified_colour_left =
cvCreateImage(cvSize(img_triclops_rectified_colour_left.ncols,
img_triclops_rectified_colour_left.nrows),8,3);

        unsigned char* ptr_triclops_red = (unsigned
char*) img_triclops_rectified_colour_left.red;
        unsigned char* ptr_triclops_green = (unsigned
char*) img_triclops_rectified_colour_left.green;
        unsigned char* ptr_triclops_blue = (unsigned
char*) img_triclops_rectified_colour_left.blue;

        img_ipl_red->imageData =
(char*)ptr_triclops_red;
        img_ipl_green->imageData =
(char*)ptr_triclops_green;
        img_ipl_blue->imageData =
(char*)ptr_triclops_blue;
        cvMerge(img_ipl_blue, img_ipl_green,
img_ipl_red, NULL, img_ipl_rectified_colour_left);

        cvNamedWindow("Rectify Color: LEFT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "Rectify Color: LEFT",
img_ipl_rectified_colour_left );
    }
    else if (image_triclops_leftRight == 1)
    {
        triclopsRectifyColorImage(triclopsContext,
TriCam_RIGHT, &m_tiRawColorImages[0],
&img_triclops_rectified_colour_right);
        IplImage* img_ipl=
cvCreateImage(cvSize(img_triclops_rectified_colour_right.ncols,
img_triclops_rectified_colour_right.nrows),8,3);
        IplImage* img_ipl_red=
cvCreateImage(cvSize(img_triclops_rectified_colour_right.ncols,
img_triclops_rectified_colour_right.nrows),8,1);
        IplImage* img_ipl_green=
cvCreateImage(cvSize(img_triclops_rectified_colour_right.ncols,
img_triclops_rectified_colour_right.nrows),8,1);
        IplImage* img_ipl_blue=
cvCreateImage(cvSize(img_triclops_rectified_colour_right.ncols,
img_triclops_rectified_colour_right.nrows),8,1);
        img_ipl_rectified_colour_right =
cvCreateImage(cvSize(img_triclops_rectified_colour_right.ncols,
img_triclops_rectified_colour_right.nrows),8,3);

```

```

        unsigned char* ptr_triclops_red = (unsigned
char*) img_triclops_rectified_colour_right.red;
        unsigned char* ptr_triclops_green = (unsigned
char*) img_triclops_rectified_colour_right.green;
        unsigned char* ptr_triclops_blue = (unsigned
char*) img_triclops_rectified_colour_right.blue;

        img_ipl_red->imageData =
(char*)ptr_triclops_red;
        img_ipl_green->imageData =
(char*)ptr_triclops_green;
        img_ipl_blue->imageData =
(char*)ptr_triclops_blue;
        cvMerge(img_ipl_blue, img_ipl_green,
img_ipl_red, NULL, img_ipl_rectified_colour_right);

        cvNamedWindow("Rectify Color: RIGHT",
CV_WINDOW_AUTOSIZE );
        cvShowImage( "Rectify Color: RIGHT",
img_ipl_rectified_colour_right );
    }
}

#pragma endregion

#pragma region DEPTH
void classBumbleContext::grab_disparity_map(int
nStereo_disparity_min, int nStereo_disparity_max, int
nStereo_maskSize, int bGenerateLUTbefore)
{
    triclopsSetDisparityMappingOn( triclopsContext, false );
    triclopsSetDisparity(triclopsContext,
nStereo_disparity_min, nStereo_disparity_max);
    triclopsSetStereoMask(triclopsContext, nStereo_maskSize);
    triclopsSetEdgeCorrelation(triclopsContext, 1);
    triclopsSetEdgeMask(triclopsContext, 11);

    triclopsSetSurfaceValidation( triclopsContext, 1 );
    triclopsSetSurfaceValidationSize( triclopsContext, 200 );
    triclopsSetSurfaceValidationDifference( triclopsContext,
1.0 );

    triclopsSetTextureValidation( triclopsContext, 1 );
    triclopsSetTextureValidationThreshold( triclopsContext,
0.2 );

    triclopsSetUniquenessValidation( triclopsContext, 0 );

    triclopsPreprocess( triclopsContext, &m_tiStereoRGB );
    triclopsRectify(triclopsContext, &m_tiStereoRGB );
    triclopsStereo( triclopsContext );

    triclopsGetImage( triclopsContext, TriImg_DISPARIETY,
TriCam_REFERENCE, &img_triclops_disparity);

    // GENERATING LUT -----
-----
    int nMinDisparity; // value in context
    int nMaxDisparity; // value in context

```

```

        int nDisparityOffset; // value in context
        triclopsGetDisparity( triclopsContext, &nMinDisparity,
&nMaxDisparity );
        triclopsGetDisparityOffset( triclopsContext,
&nDisparityOffset );

        if(bGenerateLUTbefore == 0 || nMinDisparity-
nDisparityOffset != nStereo_disparity_min ||
nStereo_disparity_max-nDisparityOffset !=
nStereo_disparity_max)
        {
            bGenerateLUTbefore = 1;
            nStereo_disparity_min = nMinDisparity-
nDisparityOffset; // yesterday stop here
            nStereo_disparity_max = nMaxDisparity-
nDisparityOffset;
            generate_LUT( nStereo_disparity_min,
nStereo_disparity_max);
        }
        //-----
        -----

        unsigned char* pRow = img_triclops_disparity.data;
        int iPixelInc= img_triclops_disparity.rowinc;
        img_ipl_disparity_colourReMap =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
img_triclops_disparity.nrows),8,3);

        img_ipl_disparity=
cvCreateImage(cvSize(img_triclops_disparity.ncols,
img_triclops_disparity.nrows),8,1); //disparity_gray

        for ( int y = 0; y < img_triclops_disparity.nrows; y++ )
        {
            uchar* ptr_ipl = (uchar*)
(img_ipl_disparity_colourReMap->imageData + y *
img_ipl_disparity_colourReMap->widthStep );
            uchar* ptr_ipl_disparity = (uchar*)
(img_ipl_disparity->imageData + y * img_ipl_disparity-
>widthStep ); //disparity_gray

            for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
            {
                if (pRow[x] > 239)
                { // invalid pixel
                    int index = pRow[x];
                    ptr_ipl[ 3*x ] =
ucInvalidDisparityMapLUT[index][0]; // red
                    ptr_ipl[ 3*x+1] =
ucInvalidDisparityMapLUT[index][1]; // green
                    ptr_ipl[ 3*x+2] =
ucInvalidDisparityMapLUT[index][2]; // blue
                    ptr_ipl_disparity[x] = index;
                } //disparity_gray
                else
                { // valid pixel
                    int index = pRow[x] << (8 -
DISPARITY_LUT_SHIFT_BITS);

```



```

        ptr_ipl[3*x] =
ucValidDisparityMapLUT[index][0]; // red
        ptr_ipl[3*x+1] =
ucValidDisparityMapLUT[index][1]; // green
        ptr_ipl[3*x+2] =
ucValidDisparityMapLUT[index][2]; // blue
        ptr_ipl_disparity[x] = index;
//disparity_gray
    }
    }
    pRow += iPixelInc;
}
cvNamedWindow("DISPARITY: COLOUR", CV_WINDOW_AUTOSIZE );
cvShowImage( "DISPARITY: COLOUR",
img_ipl_disparity_colourReMap);
}

//GENERATE LUT
//-----
--
// inline function
inline void disparityToTemperature(TriclopsContext, // may
want to use this in the future...
    unsigned int uiDisp16Bit,
    double dMinDisp,
    double dMaxDisp,
    double *pdRed,
    double *pdGreen,
    double *pdBlue )
{
    // note: context not used now but maybe later
    double dDisp = (double) uiDisp16Bit/256.0;

    if ( dDisp < dMinDisp )
    {
        dDisp = dMinDisp;
    }
    if ( dDisp > dMaxDisp )
    {
        dDisp = dMaxDisp;
    }

    double dRange = dMaxDisp - dMinDisp;

    // This code is a little more complicated than it should be,
    but it
    // does allow one to easily tweak the sizes of the different
    color
    // zones.
    // you can change the width of the red zone, blue zone and
    green zone
    // by tweaking the dRed, dGreen and dBlue values
    double dRed = 1;
    double dGreen = 1;
    double dBlue = 1;
    double dTotal = dRed + dGreen + dBlue;
    double dNormDisp = (dDisp-dMinDisp)/dRange;

```

```

double dBGThresh = dBlue/dTotal;
double dGRThresh = (dBlue+dGreen)/dTotal;

if ( dNormDisp < dBGThresh )
{
    double dInBand = dNormDisp/dBGThresh;
    *pdRed = 0;
    *pdGreen = 255*dInBand;
    *pdBlue = 255;
}
else if ( dNormDisp < dGRThresh )
{
    double dInBand = (dNormDisp-dBGThresh)/(dGRThresh-
dBGThresh);
    *pdRed = 255*dInBand;
    *pdGreen = 255;
    *pdBlue = 255*(1-dInBand);
}
else
{
    double dInBand = (dNormDisp-dGRThresh)/(1-dGRThresh);
    *pdRed = 255;
    *pdGreen = 255*(1-dInBand);
    *pdBlue = 0;
}

return;
}
void classBumbleContext::generate_LUT(int
nStereo_disparity_min, int nStereo_disparity_max)
{
    // get the invalid pixel mappings
    unsigned char ucInvalidTexture;
    unsigned char ucInvalidUniqueness;
    unsigned char ucInvalidSurface;
    unsigned char ucInvalidBackForth;
    unsigned char ucInvalidSubpixel;

    triclopsGetTextureValidationMapping(    triclopsContext ,
&ucInvalidTexture );
    triclopsGetUniquenessValidationMapping( triclopsContext ,
&ucInvalidUniqueness );
    triclopsGetSurfaceValidationMapping(    triclopsContext ,
&ucInvalidSurface );
    triclopsGetSubpixelValidationMapping(   triclopsContext ,
&ucInvalidSubpixel );
    triclopsGetBackForthValidationMapping(  triclopsContext ,
&ucInvalidBackForth );

    -----
    // WITHOUT SUBPIXEL INTERPOLATION
    for( int i = 0; i < DISPARITY_VALID_LUT_ENTRIES; i++ )
    { // generate VALID LUT
        unsigned int uiDisp16Bit = i <<
DISPARITY_LUT_SHIFT_BITS;
        if( uiDisp16Bit <= 0xFF00 )
        {
            double dRed, dGreen, dBlue;
            disparityToTemperature( triclopsContext,
                uiDisp16Bit,

```

```

nStereo_disparity_min,                (double)
nStereo_disparity_max,                (double)
                                        &dRed,
                                        &dGreen,
                                        &dBlue );
char) dRed;                            ucValidDisparityMapLUT[i][2] = (unsigned
char) dGreen;                          ucValidDisparityMapLUT[i][1] = (unsigned
char) dBlue;                          ucValidDisparityMapLUT[i][0] = (unsigned
                                        }
                                        }
    for (int i = 0; i < DISPARITY_INVALID_LUT_ENTRIES; i++)
    { // generate INVALID LUT // make them all grey
        ucInvalidDisparityMapLUT[i][0] = 127;
        ucInvalidDisparityMapLUT[i][1] = 127;
        ucInvalidDisparityMapLUT[i][2] = 127;
    }
}
#pragma endregion // generate disparity and LUT

void classBumbleContext::generate_VDisparity(int
nStereo_disparity_max)
{
    int xn=0;
    int yn=0;
    int xn2 = 0;
    long long int sum_xy = 0;
    long long int sum_x = 0;
    long long int sum_y = 0;
    long long int sum_x2 = 0;
    long long int sum2_x = 0;
    long long int n = 0;
    long long int slope_m = 0;
    long long int intercept_c = 0;
    int index_max;

    img_ipl_vdisparity =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,1);

    IplImage* img_ipl_vdisparity_min_y =
cvCreateImage(cvSize(nStereo_disparity_max, 1),8,1);
    IplImage* img_ipl_vdisparity_min =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,1);
    cvZero(img_ipl_vdisparity_min_y);
    cvZero(img_ipl_vdisparity_min);

    cvZero(img_ipl_vdisparity);
    index_max = 0;

    unsigned char* pRow = img_triclops_disparity.data;
    int iPixelInc= img_triclops_disparity.rowinc;

    for ( int y = 0; y < img_triclops_disparity.nrows; y++ )
    {

```

```

        uchar* ptr_ipl_vdisparity = (uchar*)
(img_ipl_vdisparity->imageData + y * img_ipl_vdisparity->
widthStep );
        uchar* ptr_ipl_vdisparity_min_y = (uchar*)
(img_ipl_vdisparity_min_y->imageData);

        for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
        {
            if (pRow[x] < 239)
            {
                int index = pRow[x] << (8 -
DISPARITY_LUT_SHIFT_BITS);
                // -----
                // calculating v-disparity
                ptr_ipl_vdisparity[pRow[x]] =
ptr_ipl_vdisparity[pRow[x]]+1; //test
                if(ptr_ipl_vdisparity[pRow[x]] >
index_max)
                {
                    index_max =
ptr_ipl_vdisparity[pRow[x]];
                }
                if(y >=
ptr_ipl_vdisparity_min_y[pRow[x]])
                {
                    ptr_ipl_vdisparity_min_y[pRow[x]]
= y;
                }
                // preparation to calculating ground
                plane -
                n = n+1;
                sum_xy = sum_xy + (y*pRow[x]);
                sum_x = sum_x + pRow[x];
                sum_y = sum_y + y;
                xn2 = pRow[x]*pRow[x];
                sum_x2 = sum_x2 + xn2;
                //-----
            }
        }
        pRow += iPixelInc;

        cvNamedWindow("min", CV_WINDOW_AUTOSIZE );
        cvShowImage( "min", img_ipl_vdisparity_min_y);

        //-----
        // estimating ground plane and horizon
        slope_m = ((n*sum_xy)-(sum_x*sum_y))/((n*sum_x2)-
sum_x*sum_x); // 8.94x13
        intercept_c = (sum_y-(slope_m*sum_x))/n;

        for ( int y = 0; y < img_ipl_vdisparity->height; y++ )
        {
            uchar* ptr_ipl_vdisparity = (uchar*)
(img_ipl_vdisparity->imageData + y * img_ipl_vdisparity->
widthStep); //test

```

```

        uchar* ptr_ipl_vdisparity_min_y = (uchar*)
(img_ipl_vdisparity_min_y->imageData);
        uchar* ptr_ipl_vdisparity_min = (uchar*)
(img_ipl_vdisparity_min->imageData + y *
img_ipl_vdisparity_min->widthStep); //test

        for ( int x = 0; x < img_ipl_vdisparity-
>width; x++ )
        {
            ptr_ipl_vdisparity[x] =
ptr_ipl_vdisparity[x]*255/index_max;

            if (ptr_ipl_vdisparity_min_y[x] == y)
            {
                ptr_ipl_vdisparity_min[x] = 255;
            }
        }
    }
    cvNamedWindow("V-Disparity", CV_WINDOW_AUTOSIZE );
    cvShowImage( "V-Disparity", img_ipl_vdisparity);

    cvNamedWindow("V-Disparity - min", CV_WINDOW_AUTOSIZE );
    cvShowImage( "V-Disparity - min",
img_ipl_vdisparity_min);
}

void classBumbleContext::ground_detection(int
nStereo_disparity_max)
{

    #pragma region U-Disparity
    IplImage* img_ipl_udisparity =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
nStereo_disparity_max), 8,1);
    cvZero(img_ipl_udisparity);

    IplImage* img_ipl_udisparity_thresh =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
nStereo_disparity_max), 8,1);
    cvZero(img_ipl_udisparity_thresh);

    IplImage* img_ipl_disparity_thresh =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
img_triclops_disparity.nrows), 8,1);
    cvCopy(img_ipl_disparity, img_ipl_disparity_thresh);

    for (int y=0; y<img_triclops_disparity.nrows; y++ )
    { // Generate U-Disparity
        unsigned char* pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;
        for(int x=0; x<img_triclops_disparity.ncols; x++)
        {
            unsigned char* ptr_ipl_udisparity =
(uchar*)(img_ipl_udisparity->imageData +
pRow[x]*img_ipl_udisparity->widthStep);
            if (pRow[x] < 239)
            {

```

```

        ptr_ipl_udisparity[x] =
ptr_ipl_udisparity[x]+1;
        }
    }

    cvNamedWindow("U-Disparity", CV_WINDOW_AUTOSIZE );
    cvShowImage( "U-Disparity", img_ipl_udisparity);
    #pragma endregion // U-Disparity

#pragma region world xyz

    IplImage* img_ipl_depth =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
img_triclops_disparity.nrows),8,3);

    float u, v, z;

    int disparity;
    float u_max = 0;
    float v_max = 0;
    float z_max = 0;

    float u_min = 0;
    float v_min = 0;
    float z_min = 0;

    float centerCols;
    float centerRows;
    float focalLength;
    float baseline;

    triclopsGetImageCenter(triclopsContext, &centerRows,
&centerCols);
    triclopsGetFocalLength(triclopsContext, &focalLength);
    triclopsGetBaseline( triclopsContext, &baseline);

    float distance;
    float real_u;
    float real_v;

    int depth_x, depth_y , depth_z;
    for ( int y = 0; y < img_triclops_disparity.nrows; y++ )
    {
        // to get the maximum distance of sensed
        uchar* ptr_ipl_depth = (uchar*) (img_ipl_depth-
>imageData + y * img_ipl_depth->widthStep );
        unsigned char* pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;

        for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
        {
            if (pRow[x] < 239)
            {
                disparity = pRow[x];
                triclopsRCD8ToXYZ( triclopsContext, x,
y, disparity, &u, &v, &z );

                if ((u)>u_max)
                {
                    u_max = u;

```

```

    }
    if ((v)>v_max)
    {
        v_max = v;
    }
    if (z>z_max)
    {
        z_max = z;
    }

    if ((u)<u_min)
    {
        u_min = u;
    }
    if ((v)<v_min)
    {
        v_min = v;
    }
    if (z<z_min)
    {
        z_min = z;
    }
}
}
}
#pragma endregion

#pragma region Image preparation
img_ipl_HS_disparity_half =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)/2),8,3);
for(int y=0; y<img_ipl_HS_disparity_half->height; y++)
{
    // splitting the HSD image into half / taking just
the lower half into consideration
    uchar* ptr_1 = (uchar*) (img_ipl_HS_disparity_half-
>imageData + y * img_ipl_HS_disparity_half->widthStep );
    uchar* ptr_2 = (uchar*) (img_ipl_HSV_left-
>imageData + (y+192)* img_ipl_HSV_left->widthStep);
    uchar* ptr_3 = (uchar*) (img_ipl_disparity_thresh-
>imageData + (y+191)* img_ipl_disparity_thresh->widthStep);
    for( int x=0; x<img_ipl_HS_disparity_half->width;
x++)
    {

        ptr_1[3*x] = ptr_2[3*x];
        ptr_1[3*x+1] = ptr_2[3*x+1];
        ptr_1[3*x+2] = ptr_3[x];
//ptr_2[3*x+2]; //<-----dont take disparity
into consideration
    }
}

// -----
// removing POTENTIAL obstacle area from HS-Disparity
Image
// taking into consideration just the POTENTIAL ground
plane
for (int y=0; y< img_ipl_udisparity->height; y++ )

```

```

        { // setting threshold to isolate the obstacle from
ground
        uchar* ptr1 = (uchar*) (img_ipl_udisparity-
>imageData + y*img_ipl_udisparity->widthStep);
        for(int x=0; x< img_ipl_udisparity->width; x++)
        {
            if (ptr1[x] > 20)
            { // if there is 20 pixels with the same
value in the column, then it is more likely to be obstacles
                int disparity_value = y; // in u-
disparity image, the row/height/y is the disparity value
                for (int h=0; h<
img_ipl_HS_disparity_half->height; h++ )
                { // browsing through the column of
the disparity image
                    uchar* ptr3 = (uchar*)
(img_ipl_HS_disparity_half->imageData +
h*img_ipl_HS_disparity_half->widthStep);
                    unsigned char* pRow =
img_triclops_disparity.data +
(h+191)*img_triclops_disparity.rowinc;
                    if (pRow[x] == disparity_value &&
pRow[x] < 239 )
                    { // REMOVING THE POTENTIAL
OBSTACLE AREA
                        ptr3[3*x] = 0;
                        ptr3[3*x+1] = 0;
                        ptr3[3*x+2] = 0;
                    }
                }
            }
        }
    }
}
// -----
-----
//Split Image
IplImage* plane1 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
IplImage* plane2 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
IplImage* plane3 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
cvSplit(img_ipl_HS_disparity_half, plane1, plane2,
plane3, NULL);
#pragma endregion //preparing HS_disparity image

// INITIALIZATION FOR K-MEAN
int numOfCluster= 5; // Number of Cluster
int iVector=0; // counter for each input plane / size of
EACH input plane into kmean

#pragma region kmean: colour assignment
// colour assignment to each colour
int colour_Tab[5][3];
colour_Tab[0][0]=255;
colour_Tab[0][1]=0;
colour_Tab[0][2]=0;

colour_Tab[1][0]=0;
colour_Tab[1][1]=255;
colour_Tab[1][2]=0;

```



```

colour_Tab[2][0]=0;
colour_Tab[2][1]=0;
colour_Tab[2][2]=255;

colour_Tab[3][0]=255;
colour_Tab[3][1]=0;
colour_Tab[3][2]=255;

colour_Tab[4][0]=0;
colour_Tab[4][1]=255;
colour_Tab[4][2]=255;
# pragma endregion // colour for each cluster

#pragma region kmean: preparation of input plane vector
// MEMORY ALLOCATION
// piCluster = output cluster ID
int* piCluster=(int*)malloc((img_ipl_HS_disparity_half-
>width)*(img_ipl_HS_disparity_half->height)*sizeof(int));
// planeVector = array plane vector for input
CvVect32f
*planeVector=(CvVect32f*)malloc((img_ipl_HS_disparity_half-
>width)*(img_ipl_HS_disparity_half->height)*sizeof(CvVect32f));

// converting IplImage into array vector
unsigned char pixelPointer[3]; // array pointer
for(int i=0; i<plane1->height; i++)
{
    for( int x=0; x<plane1->width; x++)
    {
        pixelPointer[0] = *(plane1-
>imageData+i*plane1->width +x);
        pixelPointer[1] = *(plane2-
>imageData+i*plane2->width +x);
        pixelPointer[2] = *(plane3-
>imageData+i*plane3->width +x); //

        planeVector[iVector]=(CvVect32f)malloc(3*sizeof(unsigned
char)); // allocation memory for each plane vector
        planeVector[iVector][0]=(unsigned
char)pixelPointer[0];
        planeVector[iVector][1]=(unsigned
char)pixelPointer[1];
        planeVector[iVector][2]=(unsigned
char)pixelPointer[2];
        iVector++;
    }
}
#pragma endregion // preparation of input plane vector
for kmean

#pragma region k-mean clustering
// cvKMeans(no of cluster, input vector, size of input
vector, no of vectors, clustering criteria, output cluster ID )
cvKMeans (numOfCluster,
planeVector, (img_ipl_HS_disparity_half-
>width)*(img_ipl_HS_disparity_half->height),
3, cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER,
10, 1.0 ), piCluster);
#pragma endregion //K-mean Function

```

```

IplImage* img_ipl_vdisparity_3c =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,3);

IplImage* img_ipl_dummy =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,1);
cvZero(img_ipl_dummy);
cvMerge(img_ipl_dummy, img_ipl_vdisparity,img_ipl_dummy,
NULL, img_ipl_vdisparity_3c);

IplImage* img_ipl_vdisparity_3c_iter2 =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,3);
cvCopy( img_ipl_vdisparity_3c,
img_ipl_vdisparity_3c_iter2);

#pragma region calculating v-disparity and displaying
clusters
// Memory allocation for output cluster
IplImage *img_ipl_outputClusterImage =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)),8,3);

IplImage *clusterImage[5];
for (int i=0; i<5; i++)
{
clusterImage[i] =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)),8,3);
}

// variables for curve fitting
int index[5] = {0};
int index_max[5] = {0}; //test
int xn[5] = {0};
int yn[5] = {0};
int xn2[5] = {0};
long long int sum_xy[5] = {0};
long long int sum_x[5] = {0};
long long int sum_y[5] = {0};
long long int sum_x2[5] = {0};
long long int sum2_x[5] = {0};
long long int n[5] = {0};
long long int slope_m[5] = {0};
long long int intercept_c[5] = {0};

int clusterNumber;
iVector = 0;
int disparity_count_max = 0;
IplImage* img_ipl_vdisparity_histogram =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,1); // test
cvZero(img_ipl_vdisparity_histogram);

for (int y=192; y< img_triclops_disparity.nrows; y++ )
{
uchar* pRow = (uchar*) (img_triclops_disparity.data
+ y * img_triclops_disparity.rowinc);

```

```

        //uchar* ptr_VDhist = (uchar*)
(img_ipl_vdisparity_histogram->imageData + y *
img_ipl_vdisparity_histogram->widthStep );
        for(int x=0; x<img_triclops_disparity.ncols; x++)
        {
            clusterNumber = piCluster[iVector];
            if (pRow[x] < 239)
            {
                // collecting data for curve fitting
formula
                n[clusterNumber] = n[clusterNumber]+1;
                sum_xy[clusterNumber] =
sum_xy[clusterNumber] + (y*pRow[x]);
                sum_x[clusterNumber] =
sum_x[clusterNumber] + pRow[x];
                sum_y[clusterNumber] =
sum_y[clusterNumber] + y;
                xn2[clusterNumber] = pRow[x]*pRow[x];
                sum_x2[clusterNumber] =
sum_x2[clusterNumber] + xn2[clusterNumber];
            }
            iVector++;
        }
    }
    //-----
    // curve fitting
    slope_m_best = 0;
    intercept_c_best = 0;

    for(int i=0; i<numOfCluster; i++)
    {
        slope_m[i] = ((n[i]*sum_xy[i])-
(sum_x[i]*sum_y[i]))/((n[i]*sum_x2[i])-sum_x[i]*sum_x[i]));
        intercept_c[i] = (sum_y[i]-
(slope_m[i]*sum_x[i]))/n[i];

        if(slope_m[i]>slope_m_best)
        {
            // choosing the best v-disparity line
            slope_m_best = slope_m[i];
            intercept_c_best = intercept_c[i];
            horizon_y = intercept_c_best;
        }
    }
    //-----

    IplImage* img_ipl_best_line =
cvCreateImage(cvSize(img_ipl_vdisparity->width,
img_ipl_vdisparity->height),8,3); // test
    cvZero(img_ipl_best_line);
    for ( int y = 0; y < img_ipl_vdisparity->height; y++ )
    {
        uchar* ptr_ipl_best_line = (uchar*)
(img_ipl_best_line->imageData + y * img_ipl_best_line-
>widthStep);
        uchar* ptr_ipl_best_line_3c = (uchar*)
(img_ipl_vdisparity_3c->imageData + y * img_ipl_vdisparity_3c-
>widthStep);
    }

```

```

        for ( int x = 0; x < img_ipl_vdisparity-
>width; x++ )
        {
            for(int i=0; i<numOfCluster; i++)
            {
                if(y-
(slope_m[i]*x)==intercept_c[i])
                {
                    ptr_ipl_best_line[3*x] =
                    ptr_ipl_best_line[3*x+1] =
                    ptr_ipl_best_line[3*x+2] =

                    ptr_ipl_best_line_3c[3*x]
                    ptr_ipl_best_line_3c[3*x+1]
                    ptr_ipl_best_line_3c[3*x+2]

                    = 0;
                    = 0;
                    = 255;

                }
            }
            if(y-
(slope_m_best*x)==intercept_c_best)
            {
                ptr_ipl_best_line_3c[3*x] = 255;
                ptr_ipl_best_line_3c[3*x+1] =
                ptr_ipl_best_line_3c[3*x+2] =

                255;
                255;

            }
        }

        img_ipl_groundPlane =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
        cvCopy(img_ipl_rectified_colour_right,
img_ipl_groundPlane);

        IplImage *img_ipl_markers =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
        cvZero(img_ipl_markers);

        -----
        // Preparing POTENTIAL OBSTACLE AREAS for Obstacle
Detection
        IplImage* img_ipl_obstacles_initial =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
        cvCopy(img_ipl_rectified_colour_right,
img_ipl_obstacles_initial);

        for(int y=0; y<img_ipl_obstacles_initial->height; y++)
        {
            //HS_DISPARIITY FULL

```

```

        uchar* ptr_1 = (uchar*) (img_ipl_obstacles_initial-
>imageData + y * img_ipl_obstacles_initial->widthStep );
        uchar* ptr_2 = (uchar*) (img_ipl_HSV_right-
>imageData + y* img_ipl_HSV_right->widthStep);
        uchar* ptr_3 = (uchar*) (img_ipl_disparity-
>imageData + y* img_ipl_disparity->widthStep);
        for( int x=0; x<img_ipl_rectified_colour_right-
>width; x++)
        {
                ptr_1[3*x] = ptr_2[3*x];
                ptr_1[3*x+1] = ptr_2[3*x+1];
                //ptr_1[3*x+2] = 0; //ptr_3[x];
                ptr_1[3*x+1] = ptr_3[x];
        }
}

IplImage* img_ipl_water_testing =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
img_ipl_markers_brightness_littleChanges =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
img_ipl_markers_brightness_largeChanges =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
cvZero(img_ipl_markers_brightness_littleChanges);
cvCopy(img_ipl_rectified_colour_right,
img_ipl_water_testing);

// Duplicating original image for markers
IplImage* img_ipl_markers_clone =
cvCloneImage(img_ipl_rectified_colour_right );

//output Image
IplImage* img_ipl_watershedOutput =
cvCloneImage(img_ipl_rectified_colour_right );
//-----
-----
for (int y=0; y< img_triclops_disparity.nrows; y++ )
{ // marking the ground plane
        uchar* pRow = (uchar*) (img_triclops_disparity.data
+ y * img_triclops_disparity.rowinc);
        uchar* ptr = (uchar*) (img_ipl_groundPlane-
>imageData + y * img_ipl_groundPlane->widthStep );
        uchar* ptr_markers = (uchar*) (img_ipl_markers-
>imageData + y * img_ipl_markers->widthStep );
        uchar* ptr_obstacles_initial =
(uchar*) (img_ipl_obstacles_initial->imageData + y *
img_ipl_obstacles_initial->widthStep);

        uchar* ptr_HSV_left = (uchar*) (img_ipl_HSV_left-
>imageData + y * img_ipl_HSV_left->widthStep);
        uchar* ptr_HSV_right = (uchar*) (img_ipl_HSV_right-
>imageData + y * img_ipl_HSV_right->widthStep);

        uchar* ptr_testing =
(uchar*) (img_ipl_water_testing->imageData + y *
img_ipl_water_testing->widthStep);
        uchar* ptr_changes =
(uchar*) (img_ipl_markers_brightness_littleChanges->imageData +
y * img_ipl_markers_brightness_littleChanges->widthStep);

```

```

        uchar* ptr_large_changes =
        (uchar*)(img_ipl_markers_brightness_largeChanges->imageData + y
        * img_ipl_markers_brightness_largeChanges->widthStep);

        for(int x=0; x<img_triclops_disparity.ncols; x++)
        {
            if (pRow[x] < 239 && y
            >=(slope_m_best*pRow[x] + intercept_c_best - 10)) //-10
            {
                ptr[3*x] = 0;
                ptr[3*x+1] = 0;
                ptr[3*x+2] = 255; //255

                ptr_markers[x] = 255;

                int ref = x + pRow[x];
                int diff_v = (ptr_HSV_right[3*x+2] -
                ptr_HSV_left[3*ref+2]);

                if ( diff_v > -20 )
                {
                    ptr_testing[3*ref] = 255;
                    ptr_testing[3*ref+1] = 255;
                    ptr_changes[x] = 255;
                }
                if ( diff_v < -20 )
                {
                    ptr_large_changes[x] = 255;
                }
            }
        }

        cvNamedWindow("Ground Plane", CV_WINDOW_AUTOSIZE );
        cvShowImage( "Ground Plane", img_ipl_groundPlane);

        cvNamedWindow("testing", CV_WINDOW_AUTOSIZE );
        cvShowImage( "testing",
        img_ipl_markers_brightness_littleChanges );

        cvNamedWindow("V-disparity: ground profile",
        CV_WINDOW_AUTOSIZE );
        cvShowImage( "V-disparity: ground profile",
        img_ipl_vdisparity_3c);

        IplImage* img_ipl_rangeLimit =
        cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
        img_ipl_rectified_colour_right->height),8,3);
        cvCopy(img_ipl_rectified_colour_right,
        img_ipl_rangeLimit);

        for (int y=0; y< img_ipl_udisparity->height; y++ )
        { // removing area that has little disparity
        information
            uchar* ptr1 = (uchar*) (img_ipl_udisparity->
            imageData + y*img_ipl_udisparity->widthStep);
            uchar* ptr_rangeLimit = (uchar*)
            (img_ipl_rangeLimit->imageData + y*img_ipl_rangeLimit->
            widthStep);

```

```

        for(int x=0; x< img_ipl_udisparity->width; x++)
        {
            if (ptr1[x] > 1 && ptr1[x] < 239) // >100
original
                { // if there is 20 pixels with the same
value in the column, then it is more likely to be obstacles
                    int disparity_value = y; // in u-
disparity image, the row/height/y is the disparity value
                    for (int h=0; h<
img_ipl_obstacles_initial->height; h++ )
                        { // browsing through the column of
the disparity image
                            uchar* ptr_markers = (uchar*)
(img_ipl_markers->imageData + h * img_ipl_markers->widthStep );
                            uchar* ptr2 = (uchar*)
(img_ipl_obstacles_initial->imageData +
h*img_ipl_obstacles_initial->widthStep);
                            unsigned char* pRow =
img_triclops_disparity.data + h*img_triclops_disparity.rowinc;

                            disparity = pRow[x];
                            triclopsRCD8ToXYZ(
triclopsContext, x, y, disparity, &u, &v, &z );

                            if(z>5)
                            {
                                ptr_rangeLimit[3*x] = 0;
                                ptr_rangeLimit[3*x+1] = 0;
                                ptr_rangeLimit[3*x+2] = 0;

                            }
                            if (pRow[x] == disparity_value &&
pRow[x] < 239 )
original
                                { // marking the potential
obstacle

                                    ptr2[3*x] = (z/z_max)*255;
                                    ptr2[3*x+1] =
(z/z_max)*255;
                                    ptr2[3*x+2] =
(z/z_max)*255; // making it invalid pixels

                                    ptr_markers[x] =
(z/z_max)*255;

                                }
                            }
                        }
                    }
                }
            }

        cvMorphologyEx(img_ipl_markers, img_ipl_markers, 0, 0,
CV_MOP_OPEN, 1 );
        cvMorphologyEx(img_ipl_markers, img_ipl_markers, 0, 0,
CV_MOP_CLOSE, 1 );

        cvNamedWindow("Markers", CV_WINDOW_AUTOSIZE );
        cvShowImage( "Markers", img_ipl_markers);

        cvNamedWindow("Range Limit", CV_WINDOW_AUTOSIZE );
        cvShowImage( "Range Limit", img_ipl_rangeLimit);

#pragma endregion

```

```

}

void classBumbleContext::water_detection(int
nStereo_disparity_max)
{
    IplImage* img_ipl_water =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
    cvZero(img_ipl_water);

    #pragma region Polarization Cue
    // Image to mark the water area
    IplImage* img_ipl_water_polarization =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
    IplImage* img_ipl_markers_polarization =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
    cvCopy(img_ipl_rectified_colour_right,
img_ipl_water_polarization);

    for ( int y = horizon_y; y <
img_triclops_disparity.nrows; y++ )
    { // marking the water region based on POLARIZATION
        unsigned char* pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;
        uchar* ptr_ipl_water = (uchar*)
(img_ipl_water_polarization->imageData + y *
img_ipl_water_polarization->widthStep );
        uchar* ptr_ipl_marker = (uchar*)
(img_ipl_markers_polarization->imageData + y *
img_ipl_markers_polarization->widthStep );
        uchar* ptr_ipl_marker2 = (uchar*) (img_ipl_water-
>imageData + y * img_ipl_water->widthStep );

        for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
        {
            if (pRow[x] > 239 )
            {
                ptr_ipl_water[3*x] = 255;
                ptr_ipl_water[3*x+1] = 0;
                ptr_ipl_water[3*x+2] = 0;
                ptr_ipl_marker[x] = 255;

                ptr_ipl_marker2[x] = 255;

            }
        }
    }

    #pragma endregion

    #pragma region Texture Cue
    IplImage *img_ipl_grayscale_right =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);

```



```

        cvCvtColor(img_ipl_rectified_colour_right,
img_ipl_grayscale_right, CV_BGR2GRAY);
        cvSplit(img_ipl_rectified_colour_right,
img_ipl_grayscale_right, NULL, NULL, NULL);

        IplImage *img_ipl_low_texture =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
        cvZero(img_ipl_low_texture);

        float variance_sq_x1;
        float variance_mean_sq_x;
        float variance_mean;
        float variance_sq_mean;
        float variance;
        float variance_normalized;
        float relative_smoothness;

        float element_1_1;
        float element_1_2;
        float element_1_3;
        float element_2_1;
        float element_2_2;
        float element_2_3;
        float element_3_1;
        float element_3_2;
        float element_3_3;

        CvScalar mean;
        CvScalar std_dev;

        double element;

        for (int y=1; y<(img_ipl_rectified_colour_right->height -
1); y++ )
        {
            // Detecting the low texture region in WHOLE image
            unsigned char* ptr_gray_top =
(uchar*)(img_ipl_grayscale_right->imageData + (y-
1)*img_ipl_grayscale_right->widthStep);
            unsigned char* ptr_gray_mid =
(uchar*)(img_ipl_grayscale_right->imageData +
y*img_ipl_grayscale_right->widthStep);
            unsigned char* ptr_gray_bottom =
(uchar*)(img_ipl_grayscale_right->imageData +
(y+1)*img_ipl_grayscale_right->widthStep);
            unsigned char* ptr_low_text =
(uchar*)(img_ipl_low_texture->imageData +
y*img_ipl_low_texture->widthStep);

            for(int x=1; x<(img_ipl_rectified_colour_right-
>width - 1); x++)
            {
                element_1_1 = ptr_gray_top[x-1];
                element_1_2 = ptr_gray_top[x];
                element_1_3 = ptr_gray_top[x+1];
                element_2_1 = ptr_gray_mid[x-1];
                element_2_2 = ptr_gray_mid[x];
                element_2_3 = ptr_gray_mid[x+1];
                element_3_1 = ptr_gray_bottom[x-1];
                element_3_2 = ptr_gray_bottom[x];
                element_3_3 = ptr_gray_bottom[x+1];
            }
        }

```

```

        variance_sq_x1 = (element_1_1*element_1_1) +
(element_1_2*element_1_2) + (element_1_3*element_1_3) +

(element_2_1*element_2_1) + (element_2_2*element_2_2) +
(element_2_3*element_2_3) +

(element_3_1*element_3_1) + (element_3_2*element_3_2) +
(element_3_3*element_3_3);

        variance_mean_sq_x = variance_sq_x1/9;

        variance_mean = ((element_1_1) +
(element_1_2) + (element_1_3) +
                                (element_2_1) +
(element_2_2) + (element_2_3) +
                                (element_3_1) +
(element_3_2) + (element_3_3))/9;

        variance_sq_mean =
variance_mean*variance_mean;

        variance = (variance_mean_sq_x -
variance_sq_mean);
        variance_normalized = variance/(255*255);

        relative_smoothness = 1 -
(1/(1+variance_normalized));

        if (relative_smoothness < 0.01)
        {
            ptr_low_text[x] = 255;
        }
    }
}

IplImage *img_ipl_texture =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);

for ( int y = horizon_y; y <
img_triclops_disparity.nrows; y++ )
    { // Marking the water region detected by texture and
brightness cue
        uchar* ptr_ipl_water = (uchar*)
(img_ipl_water_polarization ->imageData + y *
img_ipl_water_polarization ->widthStep );

        uchar* ptr_ipl_texture = (uchar*)
(img_ipl_low_texture->imageData + y * img_ipl_low_texture->
widthStep );

        uchar* ptr_ipl_texture1 = (uchar*)
(img_ipl_texture->imageData + y * img_ipl_texture->widthStep );
        uchar* ptr_ipl_ground = (uchar*)
(img_ipl_markers_brightness_littleChanges->imageData + y *
img_ipl_markers_brightness_littleChanges->widthStep );
        uchar* ptr_ipl_largeChanges= (uchar*)
(img_ipl_markers_brightness_largeChanges->imageData + y *
img_ipl_markers_brightness_largeChanges->widthStep );
    }
}

```

```

        uchar* ptr_ipl_marker2 = (uchar*) (img_ipl_water-
>imageData + y * img_ipl_water->widthStep );

        unsigned char* pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;

        for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
        {
            if (pRow[x] < 239 && y
>=(slope_m_best*pRow[x] + intercept_c_best - 10)) //-10
            {
                if (ptr_ipl_texture[x] == 255 &&
ptr_ipl_ground[x] == 0)
                {
                    ptr_ipl_texture1[x] = 255;
                    ptr_ipl_water[3*x+1] = 255;
                    ptr_ipl_marker2[x] = 255;
                }
                if(ptr_ipl_largeChanges[x] == 255)
                {
                    ptr_ipl_water[3*x+2] = 255;
                    ptr_ipl_marker2[x] = 255;
                }
            }
        }
    }

    IplImage* img_ipl_water2 =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
    cvZero(img_ipl_water2);

    cvMorphologyEx(img_ipl_water, img_ipl_water, 0, 0,
CV_MOP_OPEN, 1 );
    cvMorphologyEx(img_ipl_water, img_ipl_water, 0, 0,
CV_MOP_CLOSE, 1 );

    unsigned char* pRow;
    unsigned char* pRow_p;
    uchar* ptr_ipl_water;
    uchar* ptr_ipl_water_p;
    uchar* ptr_ipl_water_n;

    uchar* ptr_ipl_water2;

    int bStartInflection = 0;
    int bEndInflection = 0;

    int startInflection_y;
    int endInflection_y;

    for ( int x = 0; x < img_triclops_disparity.ncols; x++ )
    {
        for ( int y = img_triclops_disparity.nrows - 10; y
> horizon_y; y-- )

```

```

        {
            ptr_ipl_water = (uchar*) (img_ipl_water -
>imageData + y * img_ipl_water ->widthStep );
            ptr_ipl_water_p = (uchar*) (img_ipl_water -
>imageData + (y+1)* img_ipl_water ->widthStep );
            ptr_ipl_water_n = (uchar*) (img_ipl_water -
>imageData + (y-1)* img_ipl_water ->widthStep );

            ptr_ipl_water2 = (uchar*) (img_ipl_water2 -
>imageData + y * img_ipl_water2 ->widthStep );

            pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;
            pRow_p = img_triclops_disparity.data +
(y+1)*img_triclops_disparity.rowinc;

            if(ptr_ipl_water[x] == 0 &&
ptr_ipl_water_p[x] == 255)
                { // then start of inflection point
                    bStartInflection = 1;
                    startInflection_y = y;
                }
            if(ptr_ipl_water[x] == 0 &&
ptr_ipl_water_n[x] == 255)
                { // end of inflection point
                    bEndInflection = 1;
                    endInflection_y = y;
                }

            if(bStartInflection == 1 && bEndInflection ==
1)
                { // water region
                    for(int h = endInflection_y; h <=
startInflection_y; h++ )
                        {
                            uchar* ptr_r = (uchar*)
(img_ipl_water->imageData + h * img_ipl_water->widthStep );
                            ptr_r[x] = 255;

                        }
                    bStartInflection = 0; // found a
region, restart
                    bEndInflection = 0;
                }
        }
        bStartInflection = 0; // column end, restart
        bEndInflection = 0; // column end, restart

    }

    for ( int y = horizon_y; y <
img_triclops_disparity.nrows; y++ )
        { // Marking the water region detected by texture and
brightness cue
            uchar* ptr_ipl_water = (uchar*)
(img_ipl_groundPlane ->imageData + y * img_ipl_groundPlane -
>widthStep );

```

```

        uchar* ptr_ipl_mark = (uchar*) ( img_ipl_water-
>imageData + y * img_ipl_water ->widthStep );

        for ( int x = 0; x < img_triclops_disparity.ncols;
x++ )
        {
            if (ptr_ipl_mark[x] == 255)
            {
                ptr_ipl_water[3*x] = 255;
                ptr_ipl_water[3*x+1] = 0;
                ptr_ipl_water[3*x+2] = 0;
            }
        }
    }

    cvNamedWindow("Water", CV_WINDOW_AUTOSIZE );
    cvShowImage( "Water", img_ipl_water_polarization);

    cvNamedWindow("Water: marker2", CV_WINDOW_AUTOSIZE );
    cvShowImage( "Water: marker2", img_ipl_groundPlane);

    #pragma endregion
}

```

```

void classBumbleContext::convert_RGB2HSV(IplImage*
img_ipl_rectified_colour_left, IplImage*
img_ipl_rectified_colour_right)
{
    img_ipl_HSV_left =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
img_ipl_rectified_colour_left->height),8,3);
    img_ipl_HSV_right =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);

    cvCvtColor(img_ipl_rectified_colour_left,
img_ipl_HSV_left, CV_BGR2HSV);
    cvCvtColor(img_ipl_rectified_colour_right,
img_ipl_HSV_right, CV_BGR2HSV);
}
void classBumbleContext::triclops_preprocess_3D(int
image_resolution_width, int image_resolution_height)
{
    triclopsSetResolution( triclopsContext,
image_resolution_height, image_resolution_width );

    triclopsPreprocess( triclopsContext, &m_tiStereoRGB );
    triclopsRectify(triclopsContext, &m_tiStereoRGB );
    triclopsStereo( triclopsContext );
}
IplImage*
classBumbleContext::convert_Triclops2Ipl_1D(TriclopsImage
img_convertTriclops)
{

```

```

        IplImage* img_ipl =
cvCreateImage(cvSize(img_convertTriclops.ncols,
img_convertTriclops.nrows),8,1);

        for( int y=0; y<(img_convertTriclops.nrows); y++ )
        {
            uchar* ptr_ipl = (uchar*) (img_ipl->imageData + y *
img_ipl->widthStep );

            uchar* ptr_triclops = (uchar*)
(img_convertTriclops.data + y*img_convertTriclops.rowinc );

            for( int x=0; x<(img_convertTriclops.ncols); x++ )
            {
                ptr_ipl[x] = ptr_triclops[x];
            }
        }

        return img_ipl;
    }

IplImage*
classBumbleContext::convert_Triclops2Ipl_3D(TriclopsColorImage
img_triclops)
{
    IplImage* img_ipl=
cvCreateImage(cvSize(img_triclops.ncols,
img_triclops.nrows),8,3);
    IplImage* img_ipl_red=
cvCreateImage(cvSize(img_triclops.ncols,
img_triclops.nrows),8,1);
    IplImage* img_ipl_green=
cvCreateImage(cvSize(img_triclops.ncols,
img_triclops.nrows),8,1);
    IplImage* img_ipl_blue=
cvCreateImage(cvSize(img_triclops.ncols,
img_triclops.nrows),8,1);

    unsigned char* ptr_triclops_red = (unsigned char*)
img_triclops.red;
    unsigned char* ptr_triclops_green = (unsigned char*)
img_triclops.green;
    unsigned char* ptr_triclops_blue = (unsigned char*)
img_triclops.blue;

    img_ipl_red->imageData = (char*)ptr_triclops_red;
    img_ipl_green->imageData = (char*)ptr_triclops_green;
    img_ipl_blue->imageData = (char*)ptr_triclops_blue;

    cvMerge(img_ipl_blue, img_ipl_green, img_ipl_red, NULL,
img_ipl);

    return img_ipl;
}

```

```

void classBumbleContext::ground_testing(int
nStereo_disparity_max)
{
    #pragma region U-Disparity
    IplImage* img_ipl_udisparity =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
nStereo_disparity_max),8,1);
    cvZero(img_ipl_udisparity);

    IplImage* img_ipl_udisparity_thresh =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
nStereo_disparity_max),8,1);
    cvZero(img_ipl_udisparity_thresh);

    IplImage* img_ipl_disparity_thresh =
cvCreateImage(cvSize(img_triclops_disparity.ncols,
img_triclops_disparity.nrows),8,1);
    cvCopy(img_ipl_disparity, img_ipl_disparity_thresh);

    for (int y=0; y<img_triclops_disparity.nrows; y++ )
    { // Generate U-Disparity
        unsigned char* pRow = img_triclops_disparity.data +
y*img_triclops_disparity.rowinc;
        for(int x=0; x<img_triclops_disparity.ncols; x++)
        {
            unsigned char* ptr_ipl_udisparity =
(uchar*)(img_ipl_udisparity->imageData +
pRow[x]*img_ipl_udisparity->widthStep);
            if (pRow[x] < 239)
            {
                ptr_ipl_udisparity[x] =
ptr_ipl_udisparity[x]+1;
            }
        }
    }

    cvNamedWindow("U-Disparity", CV_WINDOW_AUTOSIZE );
    cvShowImage( "U-Disparity", img_ipl_udisparity);
    #pragma endregion // U-Disparity

    #pragma region Image preparation
    img_ipl_HS_disparity_half =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)/2),8,3);
    for(int y=0; y<img_ipl_HS_disparity_half->height; y++)
    { // splitting the HSD image into half / taking just
the lower half into consideration
        uchar* ptr_1 = (uchar*) (img_ipl_HS_disparity_half->
imageData + y * img_ipl_HS_disparity_half->widthStep );
        uchar* ptr_2 = (uchar*) (img_ipl_HSV_left->
imageData + (y+192)* img_ipl_HSV_left->widthStep);
        uchar* ptr_3 = (uchar*) (img_ipl_disparity_thresh->
imageData + (y+191)* img_ipl_disparity_thresh->widthStep);
        for( int x=0; x<img_ipl_HS_disparity_half->width;
x++)
        {
            ptr_1[3*x] = ptr_2[3*x];
            ptr_1[3*x+1] = ptr_2[3*x+1];
        }
    }
}

```

```

ptr_1[3*x+2] = 0;//ptr_3[x];
//ptr_2[3*x+2]; //<-----dont take disparity
into consideration
    }
}

// -----
// removing POTENTIAL obstacle area from HS-Disparity
Image
// taking into consideration just the POTENTIAL ground
plane
for (int y=0; y< img_ipl_udisparity->height; y++ )
{    // setting threshold to isolate the obstacle from
ground
    uchar* ptr1 = (uchar*) (img_ipl_udisparity-
>imageData + y*img_ipl_udisparity->widthStep);
    for(int x=0; x< img_ipl_udisparity->width; x++)
    {
        if (ptr1[x] > 20)
        {    // if there is 20 pixels with the same
value in the column, then it is more likely to be obstacles
            int disparity_value = y; // in u-
disparity image, the row/height/y is the disparity value
            for (int h=0; h<
img_ipl_HS_disparity_half->height; h++ )
            {    // browsing through the column of
the disparity image
                uchar* ptr3 = (uchar*)
(img_ipl_HS_disparity_half->imageData +
h*img_ipl_HS_disparity_half->widthStep);
                unsigned char* pRow =
img_triclops_disparity.data +
(h+191)*img_triclops_disparity.rowinc;
                if (pRow[x] == disparity_value &&
pRow[x] < 239 )
                {    // REMOVING THE POTENTIAL
OBSTACLE AREA
                    ptr3[3*x] = 0;
                    ptr3[3*x+1] = 0;
                    ptr3[3*x+2] = 0;
                }
            }
        }
    }
}
// -----
//Split Image
IplImage* plane1 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
IplImage* plane2 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
IplImage* plane3 =
cvCreateImage(cvGetSize(img_ipl_HS_disparity_half),8,1);
cvSplit(img_ipl_HS_disparity_half, plane1, plane2,
plane3, NULL);
#pragma endregion //preparing HS_disparity image

// INITIALIZATION FOR K-MEAN
int numOfCluster= 5; // Number of Cluster

```



```

    int iVector=0; // counter for each input plane / size of
    EACH input plane into kmean

    #pragma region kmean: colour assignment
    // colour assignment to each colour
    int colour_Tab[5][3];
    colour_Tab[0][0]=255;
    colour_Tab[0][1]=0;
    colour_Tab[0][2]=0;

    colour_Tab[1][0]=0;
    colour_Tab[1][1]=255;
    colour_Tab[1][2]=0;

    colour_Tab[2][0]=0;
    colour_Tab[2][1]=0;
    colour_Tab[2][2]=255;

    colour_Tab[3][0]=255;
    colour_Tab[3][1]=0;
    colour_Tab[3][2]=255;

    colour_Tab[4][0]=0;
    colour_Tab[4][1]=255;
    colour_Tab[4][2]=255;
    # pragma endregion // colour for each cluster

    #pragma region kmean: preparation of input plane vector
    // MEMORY ALLOCATION
    // piCluster = output cluster ID
    int* piCluster=(int*)malloc((img_ipl_HS_disparity_half-
    >width)*(img_ipl_HS_disparity_half->height)*sizeof(int));
    // planeVector = array plane vector for input
    CvVect32f
    *planeVector=(CvVect32f*)malloc((img_ipl_HS_disparity_half-
    >width)*(img_ipl_HS_disparity_half->height)*sizeof(CvVect32f));

    // converting IplImage into array vector
    unsigned char pixelPointer[3]; // array pointer
    for(int i=0; i<plane1->height; i++)
    {
        for( int x=0; x<plane1->width; x++)
        {
            pixelPointer[0] = *(plane1-
            >imageData+i*plane1->width +x);
            pixelPointer[1] = *(plane2-
            >imageData+i*plane2->width +x);
            pixelPointer[2] = *(plane3-
            >imageData+i*plane3->width +x); //

            planeVector[iVector]=(CvVect32f)malloc(3*sizeof(unsigned
            char)); // allocation memory for each plane vector
            planeVector[iVector][0]=(unsigned
            char)pixelPointer[0];
            planeVector[iVector][1]=(unsigned
            char)pixelPointer[1];
            planeVector[iVector][2]=(unsigned
            char)pixelPointer[2];
            iVector++;
        }
    }

```

```

    }
    # pragma endregion // preparation of input plane vector
for kmean

    #pragma region k-mean clustering
    // cvKMeans(no of cluster, input vector, size of input
vector, no of vectors, clustering criteria, output cluster ID )
    cvKMeans (numOfCluster,
planeVector, (img_ipl_HS_disparity_half-
>width)*(img_ipl_HS_disparity_half->height),
3, cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER,
10, 1.0 ), piCluster);
    #pragma endregion //K-mean Function

    IplImage* img_ipl_vdisparity_3c =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows), 8, 3);
    IplImage* img_ipl_vdisparity_3c2 =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows), 8, 3);

    IplImage* img_ipl_dummy =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows), 8, 1);
    cvZero(img_ipl_dummy);
    cvMerge(img_ipl_dummy, img_ipl_vdisparity, img_ipl_dummy,
NULL, img_ipl_vdisparity_3c);
    cvMerge(img_ipl_dummy, img_ipl_vdisparity, img_ipl_dummy,
NULL, img_ipl_vdisparity_3c2);

    #pragma region calculating v-disparity and displaying
clusters
    // Memory allocation for output cluster
    IplImage *img_ipl_outputClusterImage =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)), 8, 3);

    IplImage *clusterImage[5];
    for (int i=0; i<5; i++)
    {
        clusterImage[i] =
cvCreateImage(cvSize(img_ipl_rectified_colour_left->width,
(img_ipl_rectified_colour_left->height)), 8, 3);
    }

    // variables for curve fitting
int index[5] = {0};
int index_max[5] = {0}; //test
int xn[5] = {0};
int yn[5] = {0};
int xn2[5] = {0};
long long int sum_xy[5] = {0};
long long int sum_x[5] = {0};
long long int sum_y[5] = {0};
long long int sum_x2[5] = {0};
long long int sum2_x[5] = {0};
long long int n[5] = {0};
long long int slope_m[5] = {0};
long long int intercept_c[5] = {0};

int clusterNumber;

```

```

        iVector = 0;
        int disparity_count_max = 0;
        IplImage* img_ipl_vdisparity_histogram =
cvCreateImage(cvSize(nStereo_disparity_max,
img_triclops_disparity.nrows),8,1); // test
        cvZero(img_ipl_vdisparity_histogram);

        for (int y=192; y< img_triclops_disparity.nrows; y++ )
        {
            // collecting data for curve fitting
formula
            n[clusterNumber] = n[clusterNumber]+1;
            sum_xy[clusterNumber] =
sum_xy[clusterNumber] + (y*pRow[x]);
            sum_x[clusterNumber] =
sum_x[clusterNumber] + pRow[x];
            sum_y[clusterNumber] =
sum_y[clusterNumber] + y;
            xn2[clusterNumber] = pRow[x]*pRow[x];
            sum_x2[clusterNumber] =
sum_x2[clusterNumber] + xn2[clusterNumber];
        }
        iVector++;
    }
}
//-----
// curve fitting
int slope_m_best = 0;
int intercept_c_best = 0;
for(int i=0; i<numOfCluster; i++)
{
    slope_m[i] = ((n[i]*sum_xy[i]) -
(sum_x[i]*sum_y[i]))/((n[i]*sum_x2[i]) - sum_x[i]*sum_x[i]);
    intercept_c[i] = (sum_y[i] -
(slope_m[i]*sum_x[i]))/n[i];

    if(slope_m[i]>slope_m_best)
    { // choosing the best v-disparity line
        slope_m_best = slope_m[i];
        intercept_c_best = intercept_c[i];
    }
}
//-----

IplImage* img_ipl_best_line =
cvCreateImage(cvSize(img_ipl_vdisparity->width,
img_ipl_vdisparity->height),8,3); // test
        cvZero(img_ipl_best_line);
        for ( int y = 0; y < img_ipl_vdisparity->height; y++ )
        {
            //uchar* ptr_VDhist = (uchar*)
(img_ipl_vdisparity_histogram->imageData + y *
img_ipl_vdisparity_histogram->widthStep ); //test
            uchar* ptr_ipl_best_line = (uchar*)
(img_ipl_best_line->imageData + y * img_ipl_best_line-
>widthStep);
            uchar* ptr_ipl_best_line_3c = (uchar*)
(img_ipl_vdisparity_3c->imageData + y * img_ipl_vdisparity_3c-
>widthStep);

```

```

        for ( int x = 0; x < img_ipl_vdisparity-
>width; x++ )
        {
            for(int i=0; i<numOfCluster; i++)
            {
                //ptr_VDhist[x] =
ptr_VDhist[x]*255/disparity_count_max;
                if(y-
(slope_m[i]*x)==intercept_c[i])
                {
                    ptr_ipl_best_line[3*x] =
colour_Tab[i][0];
                    ptr_ipl_best_line[3*x+1] =
colour_Tab[i][1];
                    ptr_ipl_best_line[3*x+2] =
colour_Tab[i][2];
                }
            }
            if(y-
(slope_m_best*x)==intercept_c_best)
            {
                ptr_ipl_best_line_3c[3*x] = 255;
                ptr_ipl_best_line_3c[3*x+1] =
255;
                ptr_ipl_best_line_3c[3*x+2] =
255;
            }
        }
    }

    IplImage *img_ipl_groundPlane =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
    cvCopy(img_ipl_rectified_colour_right,
img_ipl_groundPlane);

    IplImage *img_ipl_markers =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);
    cvZero(img_ipl_markers);

    //-----
    // Preparing POTENTIAL OBSTACLE AREAS for Obstacle
Detection
    IplImage* img_ipl_obstacles_initial =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
    cvCopy(img_ipl_rectified_colour_right,
img_ipl_obstacles_initial);

    for(int y=0; y<img_ipl_obstacles_initial->height; y++)
    {
        //HS_DISPARITY FULL
        uchar* ptr_1 = (uchar*) (img_ipl_obstacles_initial-
>imageData + y * img_ipl_obstacles_initial->widthStep );
        uchar* ptr_2 = (uchar*) (img_ipl_HSV_left-
>imageData + y* img_ipl_HSV_left->widthStep);

```

```

        uchar* ptr_3 = (uchar*) (img_ipl_disparity-
>imageData + y* img_ipl_disparity->widthStep);
        for( int x=0; x<img_ipl_rectified_colour_right-
>width; x++)
        {
                ptr_1[3*x] = ptr_2[3*x];
                ptr_1[3*x+1] = ptr_2[3*x+1];
                ptr_1[3*x+2] = 0; //ptr_3[x];

                //ptr_1[3*x+2] = 0;
        }
}

cvPyrMeanShiftFiltering( img_ipl_obstacles_initial,
img_ipl_obstacles_initial, 20, 40, 2);

cvNamedWindow("testing", CV_WINDOW_AUTOSIZE );
cvShowImage( "testing", img_ipl_obstacles_initial);
//-----
-----
for (int y=0; y< img_triclops_disparity.nrows; y++ )
{ // marking the ground plane
        uchar* pRow = (uchar*) (img_triclops_disparity.data
+ y * img_triclops_disparity.rowinc);
        uchar* ptr = (uchar*) (img_ipl_groundPlane-
>imageData + y * img_ipl_groundPlane->widthStep );
        uchar* ptr_markers = (uchar*) (img_ipl_markers-
>imageData + y * img_ipl_markers->widthStep );
        uchar* ptr_obstacles_initial =
(uchar*) (img_ipl_obstacles_initial->imageData + y *
img_ipl_obstacles_initial->widthStep);
        for(int x=0; x<img_triclops_disparity.ncols; x++)
        {
                if (pRow[x] < 239 && y>=(slope_m_best*pRow[x]
+ intercept_c_best-10))
                {
                        ptr[3*x] = 0;
                        ptr[3*x+1] = 0;
                        ptr[3*x+2] = 255; //255

                        ptr_markers[x] = 255;

                        //ptr_obstacles_initial[3*x] = 0;
                        //ptr_obstacles_initial[3*x+1] = 0;
                        //ptr_obstacles_initial[3*x+2] = 255;
// >239 =invalid pixels
                }
        }
}
cvNamedWindow("Ground Plane", CV_WINDOW_AUTOSIZE );
cvShowImage( "Ground Plane", img_ipl_groundPlane);

cvMorphologyEx(img_ipl_markers, img_ipl_markers, 0, 0,
CV_MOP_OPEN, 1 );
cvMorphologyEx(img_ipl_markers, img_ipl_markers, 0, 0,
CV_MOP_CLOSE, 1 );

cvNamedWindow("V-disparity: ground profile",
CV_WINDOW_AUTOSIZE );

```

```

        cvShowImage( "V-disparity: ground profile",
img_ipl_vdisparity_3c);

        for (int y=0; y< img_ipl_udisparity->height; y++ )
        { // removing area that has little disparity
information
            uchar* ptr1 = (uchar*) (img_ipl_udisparity-
>imageData + y*img_ipl_udisparity->widthStep);
            for(int x=0; x< img_ipl_udisparity->width; x++)
            {
                if (ptr1[x] > 100 && ptr1[x] < 239)
                { // if there is 20 pixels with the same
value in the column, then it is more likely to be obstacles
                    int disparity_value = y; // in u-
disparity image, the row/height/y is the disparity value
                    for (int h=0; h<
img_ipl_obstacles_initial->height; h++ )
                    { // browsing through the column of
the disparity image
                        uchar* ptr_markers = (uchar*)
(img_ipl_markers->imageData + h * img_ipl_markers->widthStep );
                        uchar* ptr2 = (uchar*)
(img_ipl_obstacles_initial->imageData +
h*img_ipl_obstacles_initial->widthStep);
                        unsigned char* pRow =
img_triclops_disparity.data + h*img_triclops_disparity.rowinc;
                        if (pRow[x] == disparity_value &&
pRow[x] < 239 )
                        { // marking the potential
obstacle
                            ptr2[3*x] = 100;
                            ptr2[3*x+1] = 100;
                            ptr2[3*x+2] = 100; //
making it invalid pixels

                                ptr_markers[x] = 100;
                            }
                        }
                    }
                }
            }

//Split Image
IplImage* planeH =
cvCreateImage(cvGetSize(img_ipl_obstacles_initial),8,1);
IplImage* planeS =
cvCreateImage(cvGetSize(img_ipl_obstacles_initial),8,1);
IplImage* planeD =
cvCreateImage(cvGetSize(img_ipl_obstacles_initial),8,1);
cvSplit(img_ipl_obstacles_initial, planeH, planeS,
planeD, NULL);

// INITIALIZATION FOR K-MEAN
numOfCluster= 3; // Number of Cluster
iVector = 0; // counter for each input plane / size of
EACH input plane into kmean

#pragma region kmean: preparation of input plane vector
// MEMORY ALLOCATION
// piCluster = output cluster ID

```

```

        int* piCluster2=(int*)malloc((img_ipl_obstacles_initial-
>width)*(img_ipl_obstacles_initial->height)*sizeof(int));
        // planeVector = array plane vector for input
        CvVect32f
        *planeVector2=(CvVect32f*)malloc((img_ipl_obstacles_initial-
>width)*(img_ipl_obstacles_initial->height)*sizeof(CvVect32f));

        // converting IplImage into array vector
        unsigned char pixelPointer2[3]; // array pointer
        for(int i=0; i<planeH->height; i++)
        {
            for( int x=0; x<planeH->width; x++)
            {
                pixelPointer2[0] = *(planeH-
>imageData+i*planeH->width +x);
                pixelPointer2[1] = *(planeS-
>imageData+i*planeS->width +x);
                pixelPointer2[2] = *(planeD-
>imageData+i*planeD->width +x); //

                planeVector2[iVector]=(CvVect32f)malloc(3*sizeof(unsigned
char)); // allocation memory for each plane vector
                planeVector2[iVector][0]=(unsigned
char)pixelPointer2[0];
                planeVector2[iVector][1]=(unsigned
char)pixelPointer2[1];
                planeVector2[iVector][2]=(unsigned
char)pixelPointer2[2];
                iVector++;
            }
        }
        # pragma endregion // preparation of input plane vector
        for kmean

        // cvKMeans(no of cluster, input vector, size of input
vector, no of vectors, clustering criteria, output cluster ID )
        cvKMeans (numOfCluster,
planeVector2,(img_ipl_obstacles_initial-
>width)*(img_ipl_obstacles_initial->height),
3,cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER,
10, 1.0 ), piCluster2);

        iVector = 0;
        IplImage* img_ipl_outputClusterImage2 =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,3);
        for (int y=0; y< img_ipl_obstacles_initial->height; y++ )
        {
            uchar* ptr = (uchar*) (img_ipl_outputClusterImage2-
>imageData + y * img_ipl_outputClusterImage2->widthStep );
            for(int x=0; x<img_triclops_disparity.ncols; x++)
            {
                clusterNumber = piCluster2[iVector];
                {
                    // Output Cluster Image
                    ptr[3*x] =
colour_Tab[clusterNumber][0];
                    ptr[3*x+1] =
colour_Tab[clusterNumber][1];

```

```

        ptr[3*x+2] =
colour_Tab[clusterNumber][2];
    }
    iVector++;
}

for (int y=0; y< img_triclops_disparity.nrows; y++ )
{
    // marking the ground plane
    uchar* pRow = (uchar*) (img_triclops_disparity.data
+ y * img_triclops_disparity.rowinc);
    uchar* ptr_outputClusterImage2 =
(uchar*) (img_ipl_outputClusterImage2->imageData + y *
img_ipl_outputClusterImage2->widthStep);
    for(int x=0; x<img_triclops_disparity.ncols; x++)
    {
        if (pRow[x] < 239 && y>=(slope_m_best*pRow[x]
+ intercept_c_best-10))
        {
            ptr_outputClusterImage2[3*x] = 0;
            ptr_outputClusterImage2[3*x+1] = 255;
            ptr_outputClusterImage2[3*x+2] = 255;
// >239 =invalid pixels
        }
    }

    for (int y=0; y< img_ipl_udisparity->height; y++ )
    {
        // removing area that has little disparity
information
        uchar* ptr1 = (uchar*) (img_ipl_udisparity->
imageData + y*img_ipl_udisparity->widthStep);

        CvPoint pt1;
        CvPoint pt2;
        int bFirstTime;

        for(int x=0; x< img_ipl_udisparity->width; x++)
        {
            if (ptr1[x] > 100 && ptr1[x] < 239)
            {
                // if there is 20 pixels with the same
value in the column, then it is more likely to be obstacles
                int disparity_value = y; // in u-
disparity image, the row/height/y is the disparity value

                bFirstTime = 0;

                for (int h=0; h<
img_ipl_obstacles_initial->height; h++ )
                {
                    // browsing through the column of
the disparity image
                    uchar* ptr_outputClusterImage2 =
(uchar*) (img_ipl_outputClusterImage2->imageData + h *
img_ipl_outputClusterImage2->widthStep);
                    unsigned char* pRow =
img_triclops_disparity.data + h*img_triclops_disparity.rowinc;
                    if (pRow[x] == disparity_value &&
pRow[x] < 239 )
                    {
                        // marking the potential
obstacle

```



```

ptr_outputClusterImage2[3*x]    = 255;

ptr_outputClusterImage2[3*x+1] = 0;

ptr_outputClusterImage2[3*x+2] = 255; // >239 =invalid
pixels

                                pt1.x = x;
                                pt2.x = x;
                                if (bFirstTime == 0)
                                {
                                    pt1.y = h;
                                    bFirstTime = 1;
                                }
                                else
                                {
                                    pt2.y = h;
                                }
                                }
                                }
                                cvLine( img_ipl_outputClusterImage2,
ptr1, pt2, CV_RGB(255,0,255), 3, 8, 0 );
                                }
                                }

cvNamedWindow("final", CV_WINDOW_AUTOSIZE );
cvShowImage( "final", img_ipl_outputClusterImage2);

IplImage *img_ipl_outputGray =
cvCreateImage(cvSize(img_ipl_rectified_colour_right->width,
img_ipl_rectified_colour_right->height),8,1);

cvCvtColor( img_ipl_outputClusterImage2,
img_ipl_outputGray, CV_BGR2GRAY);
cvCanny( img_ipl_outputGray, img_ipl_markers, 50, 150,
3);

cvNamedWindow("line", CV_WINDOW_AUTOSIZE );
cvShowImage( "line", img_ipl_markers);

#pragma endregion
}

```