



AMERICAN UNIVERSITY OF BEIRUT

Towards Fully Self-Supervised Free Space  
Estimation For Unmanned Ground Vehicles

by  
Ali Harakeh

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Engineering  
to the Department of Mechanical Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
April 2016

# AMERICAN UNIVERSITY OF BEIRUT

## Towards Fully Self-Supervised Free Space Estimation For Unmanned Ground Vehicles

by  
Ali Harakeh

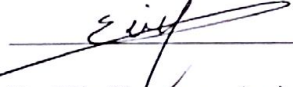
Approved by:



Dr. Daniel Asmar, Associate Professor

Advisor

Mechanical Engineering



Dr. Elie Shammas, Assistant Professor

Member of Committee

Mechanical Engineering



Dr. Imad Elhaji, Associate Professor

Member of Committee

Electrical and Computer Engineering

Date of thesis defense: April 8, 2016

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: \_\_\_\_\_  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

# Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Prof. Daniel Asmar for the continuous support of my Masters study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Elie Shamma, and Prof. Imad Elhajj for their encouragement, insightful comments, and hard questions.

I thank my fellow labmates at AUB's Vision and Robotics Lab: Salah Bazzi, Bilal Hammoud, Mohammad Alsalman, and Abdelrahman Elmakdah for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last two years.

Last but not the least, I would like to thank my family: my mother Iman, and two sisters, Miray and Dana for fully supporting me in any path I pursue throughout my life.

I dedicate this thesis to the memory of my father, Samir Harakeh. May I always make you proud.

Finally, I thank Jessica and Nini for keeping me sane. I love you both.

# An Abstract of the Thesis of

Ali Harakeh for Master of Engineering  
Major: Mechanical Engineering

Title: Towards Fully Self-Supervised Free Space Estimation For Unmanned Ground Vehicles

With the era of autonomous robots about us, the problem of scene understanding has become particularly important. The transformation of robots from human supervised systems to fully autonomous agents requires these robots to have a reliable understanding of their environment. In its most basic form, this understanding reduces to delineating occupied space from free space. Such reliable detection of free space is essential for a system to safely navigate its environment and therefore is a major interest for the robotics research community.

This thesis provides a system that automatically learns, classifies and maps free space in an environment using a stereo sensor. This is done through employing self-supervision by designing an algorithm that extracts training data automatically and reliably from free space in stereo data. The proposed algorithm is shown to be superior to other algorithms in literature by benchmarking on three stereo datasets. The results of free space classification and mapping are presented and analyzed to further validate the proposed system.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature Review</b>	<b>3</b>
<b>3 Background</b>	<b>5</b>
3.1 Stereo Sensor Model . . . . .	5
3.2 The V-Disparity Image . . . . .	7
3.3 The Projection of The Ground Class Onto The V-Disparity Image . . . . .	8
<b>4 Self Supervised Free Space Estimation For UGVs</b>	<b>10</b>
4.1 V-Disparity Image Filtering . . . . .	10
4.2 A Stochastic Model For Pixel Occupancy Probability . . . . .	13
4.2.1 Modeling The Uncertainty In The Stereo Sensor: . . . . .	13
4.2.2 Defining The Vector Space $v_s$ over $\mathbb{R}$ : . . . . .	13
4.3 Online Learning of The Occupancy Probability Density Function Via Bayesian Linear Regression . . . . .	14
4.3.1 Learning The Predictive Distribution . . . . .	14
4.3.2 Bayesian Linear Regression . . . . .	15
4.3.3 Learning The Precision Parameters . . . . .	16
4.3.4 Using The Learned PDF For Training Pixels Extraction . . . . .	18
4.4 The Second Stage Classifiers . . . . .	21
4.4.1 The Positive Naive Bayes Classifier . . . . .	21
4.4.2 The $v$ -Support Vector Classifier . . . . .	22
4.5 Feature Space Selection and The Mapping Algorithm . . . . .	23
4.5.1 Free Space Mapping . . . . .	23
4.5.2 The Selected Feature Space . . . . .	24

<b>5 Experiments And Results</b>	<b>26</b>
5.1 Hardware . . . . .	26
5.2 Datasets . . . . .	26
5.3 Assessing The Quality Of Extracted Training Pixels . . . . .	27
5.4 Comparison With Other Training Data Extraction Algorithms In Literature . . . . .	34
5.5 Free Space Classification and Mapping Results . . . . .	40
<b>6 Conclusion and Future Work</b>	<b>46</b>
<b>Bibliography</b>	<b>47</b>



# List of Figures

3.1	The stereo sensor coordinate system. . . . .	6
3.2	Transformation from $(u, d, v)$ to $(v, d, s)$ [1] . . . . .	7
3.3	Examples of a disparity image and its corresponding v-disparity image. . . . .	9
4.1	Flowchart of the proposed system. . . . .	10
4.2	V-disparity image filtering. . . . .	12
4.3	Bayesian linear regression results. . . . .	19
4.4	The effect of varying the confidence interval on the amount and quality of training pixels extracted in the image. . . . .	20
5.1	Deployed sensors and robot. The Zed Stereo Camera is rigidly mounted on top of the Clearpath Husky UGV. . . . .	27
5.2	Outlier fraction. . . . .	30
5.3	Easy dataset histogram comparison. . . . .	31
5.4	Medium dataset histogram comparison. . . . .	32
5.5	Difficult dataset histogram comparison. . . . .	33
5.6	Plots of recall values of the v-SVC and computational time (in seconds) of the three training data extraction methods per frame of the datasets. . . . .	38
5.7	Examples of the results of three training data extraction methods. . . . .	39
5.8	Comparison of second stage classifiers. . . . .	42
5.9	PNB results. . . . .	43
5.10	v-SVC results. . . . .	44
5.11	Mapping results. . . . .	45

# List of Tables

4.1	Comparison of feature computation time. . . . .	25
5.1	A comparison of the estimated feature histogram with ground truth . .	29
5.2	Comparison with other training data extraction algorithms. . . . .	35

# Chapter 1

## Introduction

With the era of autonomous robots about us, the problem of scene understanding has become particularly important. The transformation of robots from human supervised systems to fully autonomous agents requires these robots to have a reliable understanding of their environment. In its most basic form, this understanding reduces to delineating occupied space from free space. Such reliable detection of free space is essential for a system to safely navigate its environment and therefore is a major interest for the robotics research community.

Fully autonomous free space estimation is considered one of the holy grails of robotics research. Despite an abundance of algorithms tackling this problem, it is still considered an unsolved problem mainly due to the particularity of proposed algorithms to certain environments. This phenomenon is much more prominent in environments where the properties of free space vary, whether the variation was spatial or temporal. Algorithms tailored specifically to an environment are expected to perform rather poorly when the properties of the environment change.

This led to a great emphasis on machine learning when trying to tackle scene understanding tasks such as free space estimation. The main issue with learning based algorithms is that they usually require a training phase, where training data describing free space is used as an input to model its properties. The extraction and classification of training data is usually performed through direct human supervision, which becomes impractical and time consuming as the range of properties to be learned becomes larger. Furthermore, the resulting system cannot extend classification beyond environments it learned before restricting its autonomy.

Recent free space estimation approaches tackle this problem through self-supervision, the process in which one classifier directly supervises input to a second classifier. The first classifier can usually reliably label a small portion of the environment as free space, which is then provided as input to the second classifier that extends the labeling over the whole environment. It is within this framework that this thesis find its call, mainly using self-supervision to design a fully autonomous self-supervised free space estimation algorithm.

The main contributions of this thesis are as follows:

- A filtering algorithm that increases the reliability of estimating the ground correlation line in the v-disparity space.
- A mathematical proof that measuring the uncertainty in disparity measured by a stereo sensor is sufficient to measure the uncertainty in 3D-coordinates of a point in space.
- A method to model this uncertainty for points that belong to the ground plane through Bayesian linear regression.
- A novel training data extraction method that uses the modeled uncertainty to extract training pixels from the ground class in a stereo image and that is fast, accurate, and can be applied in structured and unstructured environments.
- Using the proposed algorithm to supervise two classifiers and to perform free space mapping.
- Validated the method on outdoor datasets.

The remainder of this thesis is structured as follows: Chapter 2 provides a review of recent trends in self-supervised free space estimation. Chapter 3 provides an explanation of the stereo sensor model, the v-disparity image construction, and the projection of the ground class onto the v-disparity space. Chapter 4 describes the different components of the proposed system including the filtering algorithm and its results, the necessary equations to model the uncertainty in disparity for pixels belonging to the ground plane, how to use this uncertainty to reliably select training pixels belonging to free space, a description of the two supervised classifiers used, feature space selection, and the free space mapping algorithm. Chapter 5 presents the experimental setup and results and describes the datasets acquired and then provides an analysis of how well the feature distributions of the ground class are represented by the extracted training data. Finally, Chapter 6 concludes the thesis and provides the direction for future research.

# Chapter 2

## Literature Review

This chapter reviews recent work on free space estimation with emphasis on self-supervised learning approaches.

A variety of sensors and sensor combinations have previously been employed for free space estimation. Sugar et al.[2] employed a 3-D LIDAR to find the occupancy probability of the environment through a semi-supervised learning approach. The robot is driven by a human operator through a safe trajectory where it collects the remission and spatial features of free space, which are used as training data for a one-class classifier. Dahlkamp et al.[3] used a 2-D LIDAR to extract training data belonging to free space using the Probabilistic Terrain Analysis (PTA) algorithm proposed in [4]. The training data is then projected to a monocular camera and used to build a color based classifier. The PTA algorithm requires unknown parameters to be learned offline using human supervision. These two systems are suitable when the properties of the robot's operating environment resemble these of the training environment. The system presented in this paper differs from both methods in that it is totally independent of any human supervision and it does not have free parameters that need to be trained prior to deployment in a given environment.

Radars have also been successfully employed for self-supervision in free space estimation. Milella et al.[5] used the echo in a radar image to identify ground patches and then projected these patches to a monocular camera coordinate frame in order to train a visual classifier. The classification was done through Mahalanobis distance thresholding. The optimal threshold is determined by constructing ROC curves on a training dataset. In their work, the radar produces training patches at a specified distance of 11.4 meters in front of the robot. Unfortunately, in some scenarios distance patches might not possess the same features as closer ones, thereby causing the latter to be classified as obstacles. The system presented in this paper gets around this problem by extracting training patches from all over the field of view of the camera.

Stereo cameras are also used for self-supervised free space estimation and provide a dense 3-D representation of the scene with additional color information. Milella et al.[6] utilizes a stereo camera to extract geometric features that are used to classify voxels in a 3-D point cloud belonging to free space through the same classifier used

in [5]. Reina et al.[7] also used a stereo sensor to classify free space via a mixture of Gaussians model with automatic estimation of the number of components. The main weakness in these two systems is that in order to create the ground model, both systems need to be initialized in an area free of obstacles. The requirement for initialization is problematic when the systems fails and the human operator cannot intervene to reinitialize them. My proposed system does not need any special initialization and in fact can be launched inside a heavily cluttered scene.

Vernaza et al.[8] used a stereo sensor in a Markov Random Field framework to classify pixels in the image belonging to the ground plane. The largest planar region is assumed to be the ground plane, and pixels belonging to it are taken as ground pixels. Hadsell et al.[9] uses the hough transform up to three times to fit planes to stereo point clouds, while bounding the maximum slope a UGV can drive on, to remove points belonging to the ground plane. Moghadam et al.[10] uses Ransac plane fitting to determine points belonging to the largest plane in 3-D stereo generated point clouds, which is assumed to be the ground plane. These points are then used to supervise a supervised classifier to classify far away pixels in stereo images. These training data extraction method fails in scenarios where the ground plane is not the largest plane in the image. The novel training data extraction algorithm presented in this thesis utilizes the properties of the projection of the ground on the v-disparity image, and is able to extract training pixels even if the ground is not planar.

Kim et al.[11] uses the assumption that free space in stereo data should have a low derivative in stereo disparity space to provide training labels for a supervised classifier that describes class data in a codebook method. The requirement of a threshold to determine the definition of low derivative makes the training data extraction algorithm unreliable. The training training data extraction algorithm presented in this thesis overcomes this problem by only requiring a single free parameter that determine the reliability vs number of training points extracted.

Proprioceptive sensors such as vibration sensors have also been used to provide labels for free space classification tasks [12], [13]. These sensors require the robot to have previously driven over a patch to determine if it belongs to the ground class. Furthermore, the vibration based classifiers usually requires manual tuning [13]. The proposed algorithm in this thesis requires minimal human supervision and does not require manual tuning.

It is noticed that the main weakness in the state of the art is in the supervision methods, i.e. the training data extraction methods. The weakness of these methods is that they impose strong assumptions on the geometry of the ground plane.

# Chapter 3

## Background

This chapter aims to provide an explanation of the stereo sensor model, the v-disparity image construction, and the projection of the ground class onto the v-disparity space.

### 3.1 Stereo Sensor Model

A stereo setup (Fig.3.1) consists of two cameras that are assumed identical with equal calibration matrices; after rectification the image planes of both cameras become coplanar with their centers of projections separated by a baseline  $b$ .

The coordinate frames  $C_r(X_r, Y_r, Z_r)$  and  $C_l(X_l, Y_l, Z_l)$  are pinned to the optical center of the right and left cameras respectively, while the world coordinate frame  $W(X_w, Y_w, Z_w)$  of the stereo rig is located at the midpoint between the two cameras at a distance  $\frac{b}{2}$  from both  $C_l$  and  $C_r$ . The focal length is designated as  $f$  and the optical centers of the images as  $u_0$  and  $v_0$ . Another assumption in my proposed system is that the stereo rig can only rotate in the pitch angle,  $\theta$ , and as such no angular rotation occurs in the roll and yaw directions. For each camera, projective geometry [14] results in the following relation between a 3D point and its projection:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = Q \times \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix}, \quad (3.1)$$

where  $X_p, Y_p, Z_p$  are the coordinates of  $P$  with respect to the world coordinate frame, and  $Q$  is known as the projection matrix and is given by:

$$Q = \begin{bmatrix} f & u_0 \sin \theta & u_0 \cos \theta & \pm \frac{b}{2} \\ 0 & f \cos \theta + v_0 \sin \theta & -f \sin \theta + v_0 \cos \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \end{bmatrix} \quad (3.2)$$

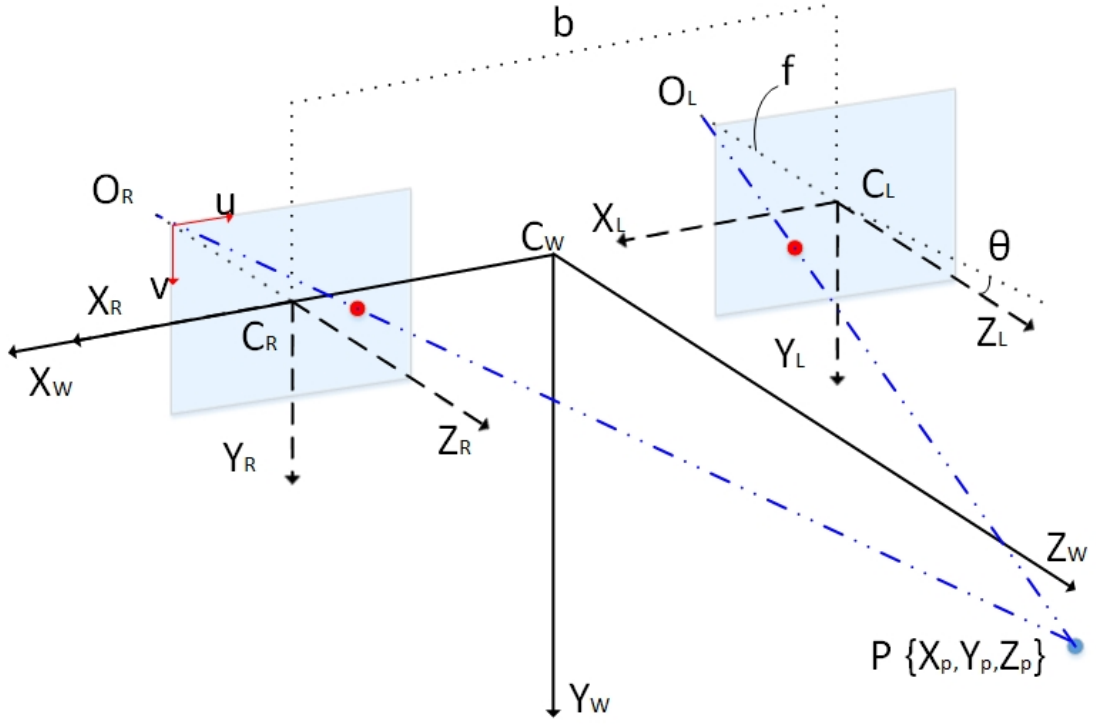


Figure 3.1: The stereo sensor coordinate system.

Expanding (3.1) the image coordinates of a point  $P$  are:

$$u_{l,r} = u_0 + f \frac{X_p \pm \frac{b}{2}}{Y_p \sin \theta + Z_p \cos \theta} \quad (3.3)$$

$$v = v_0 + f \frac{Y_p \cos \theta - Z_p \sin \theta}{Y_p \sin \theta + Z_p \cos \theta} \quad (3.4)$$

The epipolar constraint reduces the search for corresponding pixels in the two images to a 1-D search problem along the epipolar line. The disparity  $d$  of the matched feature is thus calculated as:

$$d = f \frac{b}{Y_p \sin \theta + Z_p \cos \theta} \quad (3.5)$$

Following the method presented in [15]  $X_p$ ,  $Y_p$ ,  $Z_p$  can be derived from  $u_l$ ,  $v$  and  $d$  as follows to get:

$$X_p = b \frac{d - 2u + 2u_0}{2d} \quad (3.6)$$

$$Y_p = b \frac{(v - v_0) \cos \theta + f \sin \theta}{d} \quad (3.7)$$

$$Z_p = b \frac{f \cos \theta + (v_0 - v) \sin \theta}{d} \quad (3.8)$$



## 3.2 The V-Disparity Image

The v-disparity algorithm was first proposed by Labayrade et al.[16] for road plane estimation in urban scenes. It transforms a disparity image to a v-disparity image by forming a 256-bin histogram of disparity values for each row of the disparity image and concatenating these histograms in the same order as the rows they were generated from. Fig.3.3 provides an example of a v-disparity image with its corresponding disparity image and a pseudocode describing the construction procedure is provided in algorithm 1.

---

### Algorithm 1: v-Disparity Image Construction

---

**Input:**  $m \times n$  Disparity Image,  $D$   
**Output:**  $m \times 256$  v-Disparity Image,  $VD$

- 1 Initialize  $VD$  as an empty array ;
- 2 **begin**
- 3     **for** every row  $i$  in  $D$  **do**
- 4         Compute the 256 bin histogram  $h$  of disparity values in row  $i$  ;
- 5         Insert  $h$  at the bottom of  $VD$  ;
- 6     **end**
- 7 **end**

---

Fig.3.2 provides a visual representation of the transformation in the algorithm. For further reference on the construction of the v-disparity image, refer to [16].

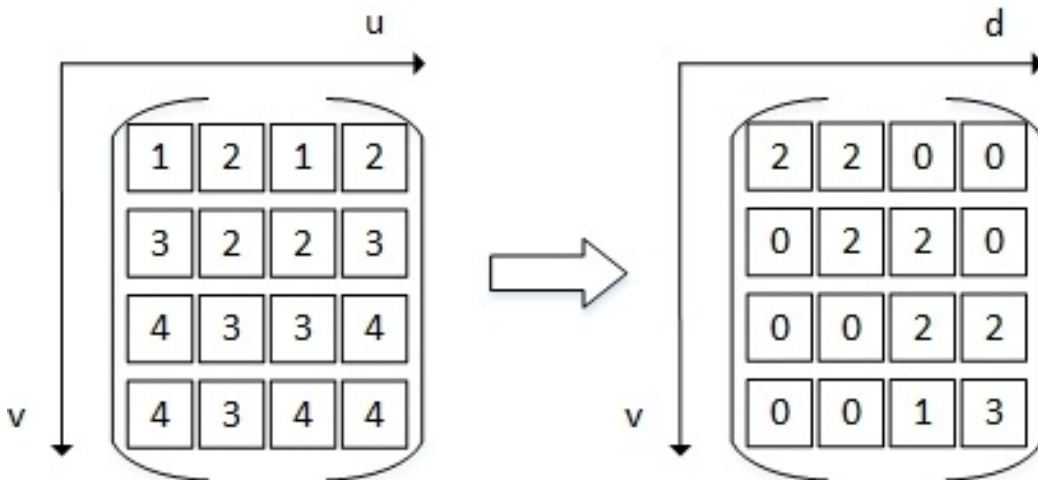


Figure 3.2: Transformation from  $(u, d, v)$  to  $(v, d, s)$  [1]

### 3.3 The Projection of The Ground Class Onto The V-Disparity Image

For unmanned ground vehicles, the ground class in the world coordinate system is made up of either horizontal planes, oblique planes, or a mixture of both. The equations of such planes in the world coordinate system are:

$$Y_w = h \quad (3.9)$$

for the horizontal plane and

$$Z_w = \psi Y_w + \kappa \quad (3.10)$$

for the oblique plane, where  $h$  is the height of the stereo sensor with respect to the plane and  $\psi$  and  $\kappa$  are constants. By combining the above equations with (3.3), (3.4), and (3.5), the equations of the projection of the horizontal and oblique planes in the disparity space can be written as:

$$\frac{h}{b}d = f \sin \theta + (v - v_0) \cos \theta \quad (3.11)$$

for the horizontal planes and

$$\frac{\kappa}{b}d = f(\cos \theta + \psi \sin \theta) + (v_0 - v)(\sin \theta + \psi \cos \theta). \quad (3.12)$$

for the oblique planes. The resulting equations suggest that for pixels belonging to horizontal and oblique planes, the pixel's disparity  $d$  is a linear function of its row coordinate  $v$ . As such, the projection of the ground class onto the v-disparity space produces a mixture of slanted lines with a slope proportional to the pitch angle  $\theta$  [16]. As a final note, vertical planes are also projected onto the v-disparity space, but as vertical lines. For a more detailed analysis of the projection of planes onto the v-disparity space, I refer the reader to the work by Hu and Uchimura [15].

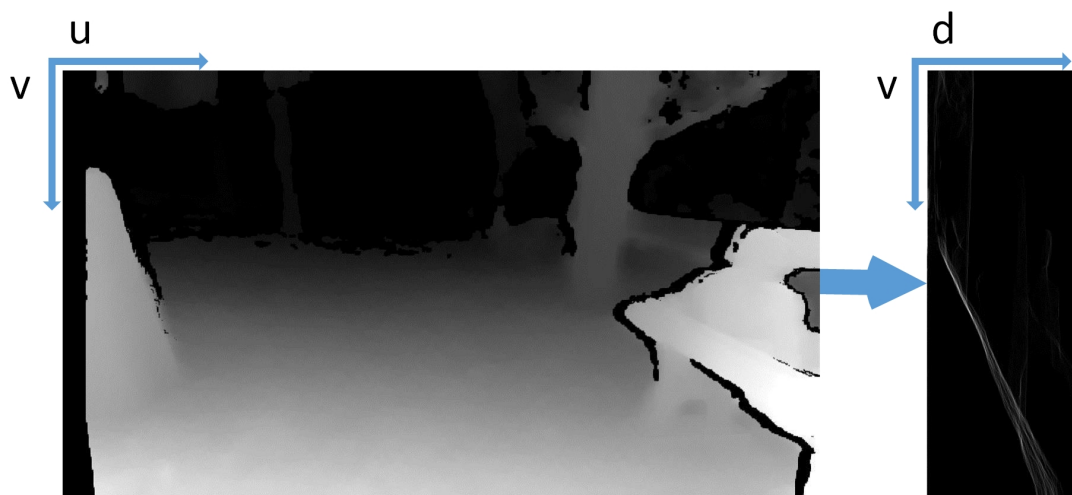


Figure 3.3: Left: disparity image. The bright color signifies larger disparity and hence smaller depth. Right:  $v$ -disparity image. The projection of the ground class in the scene is a slanted line, which is visible in the  $v$ -disparity image.

# Chapter 4

## Self Supervised Free Space Estimation For UGVs

A flowchart of our proposed system is shown in Fig.4.1, where the training data extraction algorithm is shown in the top row. This chapter explains in details the theoretical aspects of each block of the proposed system.

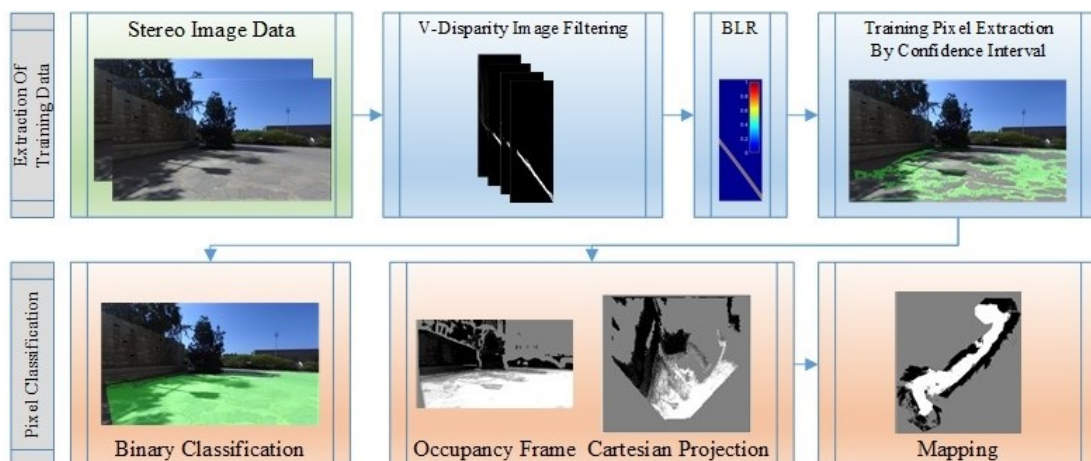


Figure 4.1: A flowchart that shows the different components of the system. In this specific example, the results of the Positive Naive Bayes Classifier are shown.

### 4.1 V-Disparity Image Filtering

The raw v-disparity image contains the projection of both the ground class and the obstacle class, the latter appearing as vertical lines. Direct detection of the ground correlation line <sup>1</sup> in the raw v-disparity image is unreliable, especially in cluttered

<sup>1</sup>The slanted line projection of the ground class is termed the ground correlation line [16]

and unstructured scenes [17],[1],[18]. This section provides a solution to this problem through a modified version of the v-disparity filtering algorithm provided by the authors in [17].

The proposed filtering algorithm takes as an input the v-disparity image and provides as an output a binary filtered version that only contains pixels belonging to the ground correlation line. The first step of the filtering algorithm is the removal of vertical line segments from the v-disparity image. This is done through Sobel horizontal edge detection. Vertical lines are greatly attenuated in the resulting image, reducing to residuals with a small area. A majority black morphological operation is then applied to eliminate noise induced edge blobs, and to fill holes found within the ground correlation line. The final filtering stage filters the remaining blobs by area, such that is set blobs with an area smaller than a certain threshold are removed, cleaning any residuals that remain after previous filtering stages. To avoid the limitation of changing the threshold to accommodate different v-disparity images, the value of the threshold is set to be the total area of all blobs divided by a random integer sampled from a uniform distribution defined between 5 and 25. The area filtering is then continuously iterated while sampling a new random integer value at each iteration until the correlation coefficient of the  $v,d$  coordinate pairs remaining in the resulting image is greater than or equal to 0.98. This convergence criterion forces the remaining  $v,d$  coordinate pairs in the image to have a pseudo-linear relation while at the same time allowing some deformation in the ground correlation line to accommodate multi-planar ground class geometry. The three stages of the filtering algorithm are presented in Fig.4.2.

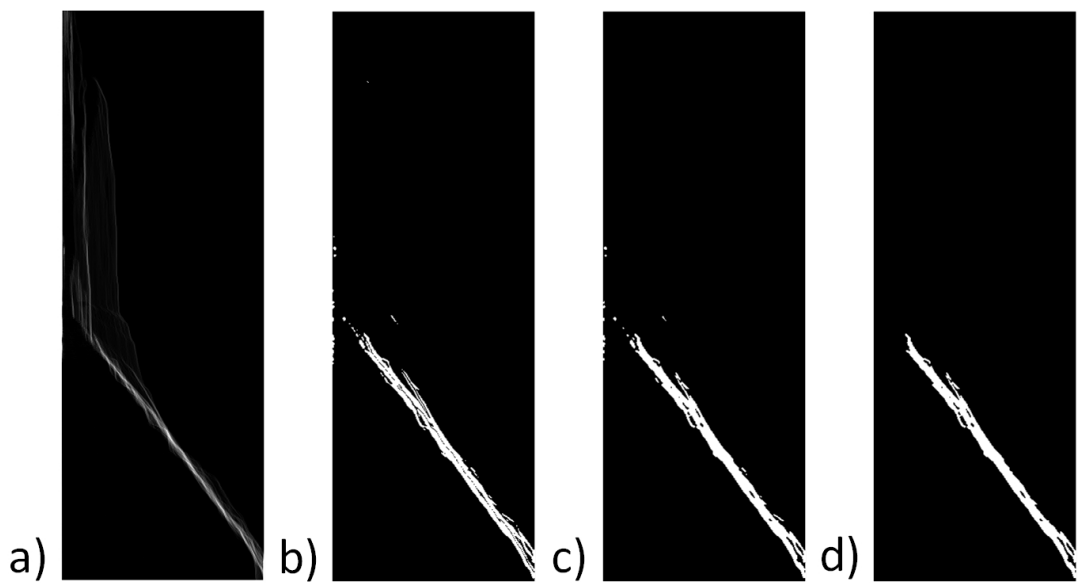


Figure 4.2: a) Original v-disparity image with visible vertical lines representing obstacles. b) Application of Sobel horizontal edge detection. c) Application of a majority black morphological operation. d) Application of the randomized binary area opening morphological operation. The final filtered version of the v-disparity image after the randomized binary area opening procedure.

## 4.2 A Stochastic Model For Pixel Occupancy Probability

This section provides a proof that for a subset of pixels laying on the ground plane, estimating the probability of their disparity value to belong to the ground correlation line is sufficient to describe their occupancy probability.

### 4.2.1 Modeling The Uncertainty In The Stereo Sensor:

The random vector  $(u, v, d)$  is denoted as the measurement vector  $\mathbf{Z}$  and  $(X_p, Y_p, Z_p)$  as the state vector  $\mathbf{X}$ .  $\mathbf{Z}$  belongs to the vector space  $\mathfrak{v}$  defined over  $\mathbb{R}^3$ . From (3.3), (3.4), and (3.5) there exists a transformation  $J$  that projects the world coordinates of the scene to the image coordinate frame according to:

$$\mathbf{Z} = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \text{round}(J \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} + \varepsilon) \quad (4.1)$$

The measurement vector is assumed to be a noisy projection of the state vector  $\mathbf{X}$ , corrupted by Gaussian noise. Such a noise model is due to errors in the imaging process of each individual camera, matching errors in the disparity computation algorithm, and truncation errors due to rounding. The noise is modeled by adding the Gaussian random vector  $\varepsilon$  in (4.1). Accordingly, the vector  $\mathbf{Z}$  is a random vector with mean  $\mu_z = [u, v, d]$  and covariance  $\Sigma_z$  expressed as:

$$\Sigma_Z = \begin{bmatrix} \sigma_u^2 & \lambda_{u,v} & \lambda_{u,d} \\ \lambda_{v,u} & \sigma_v^2 & \lambda_{v,d} \\ \lambda_{d,u} & \lambda_{d,v} & \sigma_d^2 \end{bmatrix}$$

where  $\lambda_{x_i, x_j}$  is the cross-covariance of  $x_i$  and  $x_j$ .

### 4.2.2 Defining The Vector Space $\mathfrak{v}_s$ over $\mathbb{R}$ :

At first, vector  $\mathbf{Z}$  is mapped from the disparity space to the  $v$ -disparity space. This mapping preserves the value of  $v$  and  $d$  and as such is linear for both variables, which implies that the Gaussian uncertainty in both variables is preserved. Furthermore, the frequency  $s$  is a sum of logical outcomes and as such is deterministic and is not relevant when applying uncertainty analysis. This transformation redefines a new vector space over  $\mathbb{R}^2$  and as such, the measurement vector becomes a function of  $v$  and  $d$  with a covariance matrix:

$$\Sigma_Z = \begin{bmatrix} \sigma_v^2 & \lambda_{v,d} \\ \lambda_{d,v} & \sigma_d^2 \end{bmatrix} \quad (4.2)$$

It was noted in Section 3.3 that the ground class is projected as a slanted lines in the  $v$ -disparity image. Rearranging (3.12) and combining constant terms into a single variable would result in this line having the form:

$$d = w_0 + w_1 v \quad (4.3)$$

Due to this linear relation, the covariance matrix of the measurement vector  $\mathbf{Z}$  in the vicinity of this line changes from positive definite to positive semi-definite. This implies that as long as this relation holds, only one of the variables is sufficient to describe the state. To recover the positive definite property of the covariance matrix, a new vector space  $\mathbf{v}_s$  is now defined over  $\mathbb{R}$  where the measurement vector reduces to a single variable  $d$  with a covariance  $\sigma_d^2$ . The new measurement variable  $Z$  can be written as:

$$d = \mathcal{N}(\mu_d, \sigma_d^2) \quad (4.4)$$

The above simplification implies that given the linear relation in (4.3) there exists a subset of pixels, specifically pixels belonging to the ground plane in  $\mathbb{R}^3$  where estimating the occupancy probability of  $d$  is sufficient to estimate the occupancy probability of the pixel with measurement vector  $\mathbf{X}$ . It should be noted that for a UGV, the occupancy probability of a pixel is equivalent to the probability of its measurement vector  $\mathbf{X}$  to belong to the ground plane, and hence to the probability of its disparity value  $d$  to belong to the ground correlation line.

### 4.3 Online Learning of The Occupancy Probability Density Function Via Bayesian Linear Regression

To learn the distribution of the Gaussian random variable  $d$  in 4.4, the  $(v, d)$  pairs that remain in the image after filtering are used as training data input to Bayesian linear regression, with  $v$  as the input variable and  $d$  as the target variable. Although the predictive distribution  $P(d/v)$  learned through Bayesian linear regression is usually used to predict new values of  $d$ , it can be used to compute the probability of a measured disparity value from the disparity image to belong to the ground correlation line.

#### 4.3.1 Learning The Predictive Distribution

The non-planar nature of the ground plane in off-road scenarios leads to a distorted ground line projection in the  $v$ -disparity image that might not be straight. To accommodate this case, the disparity  $d$  is modeled as a second degree polynomial function of  $v$  which has the form:

$$d = w_0 + w_1 v + w_2 v^2 + \delta = \mathbf{w}^T \phi^*(v) + \delta, \quad (4.5)$$



where  $\phi^*(v)$  are the set of second degree polynomial basis function,  $[\phi_0(v) \phi_1(v) \phi_2(v)] = [v^0 v^1 v^2]$  and  $\mathbf{w}$  the parameter vector  $\mathbf{w} = [w_0 w_1 w_2]$ .  $\delta$  is a zero mean Gaussian random variable with precision  $\beta$ . The conditional distribution of  $d$  takes the following form:

$$P(d|v, \mathbf{w}, \beta) = \mathcal{N}(d; \mathbf{w}^T \phi^*(v), \beta^{-1}). \quad (4.6)$$

The vectors  $\mathbf{v} = [v_1 \dots v_N]$  and  $\mathbf{d} = [d_1 \dots d_N]$  are now defined as the training data pairs, where  $v_n, d_n$  are coordinate pairs extracted from the filtered v-disparity image. Target training variables  $[d_1 \dots d_N]$  are assumed to be IID variables drawn from the conditional distribution in (4.6) and as such, their likelihood function has the expression :

$$P(\mathbf{d}|\mathbf{v}, \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(d_n; \mathbf{w}^T \phi^*(v_n), \beta^{-1}). \quad (4.7)$$

### 4.3.2 Bayesian Linear Regression

At first, it should be noted that throughout this section, the variables  $v$  and  $\mathbf{v}$  will be added to the conditional variables through the independence assumption. To begin with the Bayesian treatment of linear regression, a prior distribution is defined over the model parameter vector  $\mathbf{w}$  as:

$$P(\mathbf{w}|\alpha) = P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta) = \mathcal{N}(0, \alpha^{-1}I), \quad (4.8)$$

For simplicity, the prior is considered to be zero mean and isotropic Gaussian with a single precision parameter  $\alpha$ . This assumption reduces the number of unknown parameters in the prior to only  $\alpha$  and results in a Gaussian posterior distribution when multiplied with the likelihood function in (4.7). Having set the prior, the posterior distribution of the parameter vector  $\mathbf{w}$  given the training data can be written using Bayes rule as:

$$P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \Gamma P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta), \quad (4.9)$$

where  $\Gamma$  is a normalization coefficient and  $P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w})$  is the likelihood function in (4.7). The posterior distribution is computed by completing the squares in the exponential and then making use of the standard form of the normalization coefficient of the Gaussian, and has the form:

$$P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \mathcal{N}(\mathbf{w}; \mu_w, \Sigma_w), \quad (4.10)$$

where  $\mu_w$  is the mean:

$$\mu_w = \beta \Sigma_w \Phi^T \mathbf{d}, \quad (4.11)$$

and  $\Sigma_w$  is the  $3 \times 3$  covariance matrix:

$$\Sigma_w^{-1} = \alpha I + \beta \Phi^T \Phi. \quad (4.12)$$

Here,  $I$  is a  $3 \times 3$  identity matrix and  $\Phi$  is the design matrix, written in terms of the input vector  $\mathbf{v}$  as:

$$\Phi = \begin{bmatrix} 1 & v_1 & v_1^2 \\ \cdot & & \\ \cdot & & \\ 1 & v_n & v_n^2 \end{bmatrix} \quad (4.13)$$

The predictive distribution is expanded according to the theorem of total probability as:

$$P(d|\mathbf{v}, \mathbf{d}, \alpha, \beta) = \int_{\mathbf{w}} P(d|\mathbf{v}, \mathbf{d}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|\mathbf{v}, \mathbf{d}, \alpha, \beta) d\mathbf{w}. \quad (4.14)$$

It is noted that the predictive distribution is the result of a convolution of two Gaussian distributions in (4.6) and (4.10). Accordingly, the predictive distribution has the following form:

$$P(d|\mathbf{v}, \mathbf{d}, \alpha, \beta) = \mathcal{N}(d; \mu_w^T \phi^*(v), \Sigma_p), \quad (4.15)$$

where the variance  $\Sigma_p$  can be written as:

$$\Sigma_p = \frac{1}{\beta} + \phi^*(v)^T \Sigma_w \phi^*(v). \quad (4.16)$$

Although the unknown parameter  $\mathbf{w}$  has been marginalized, the previous equations require precise knowledge of the precision parameters  $\alpha$  and  $\beta$ , which might not be available a priori.

### 4.3.3 Learning The Precision Parameters

In a fully Bayesian treatment, the predictive distribution would be expanded using the theorem of total probability over all three unknown parameters  $\alpha$ ,  $\beta$ , and  $\mathbf{w}$ . This expansion would have the form:

$$P(d|\mathbf{v}, \mathbf{d}) = \int_{\alpha} \int_{\beta} \int_{\mathbf{w}} P(d|\mathbf{v}, \mathbf{d}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|\mathbf{v}, \mathbf{d}, \alpha, \beta) P(\alpha, \beta|\mathbf{v}, \mathbf{d}) d\mathbf{w} d\beta d\alpha,$$

which has no closed form solution due to the lack of knowledge of the conditional joint PDF  $P(\alpha, \beta|\mathbf{v}, \mathbf{d})$ . An approximation of the fully Bayesian treatment of this hierarchical model is computed by setting the hyperparameters at the highest level of the hierarchy ( $\alpha$  and  $\beta$ ) to their most likely values instead of integrating them out [19].

We start by assuming that the conditional joint pdf is sharply peaked around the values of the true hyper-parameters  $\hat{\alpha}$  and  $\hat{\beta}$ . The predictive distribution in this case can be estimated as:

$$P(d|v, \mathbf{v}, \mathbf{d}) \simeq P(d|v, \mathbf{v}, \mathbf{d}, \hat{\alpha}, \hat{\beta}) = \int_{\mathbf{w}} P(d|v, \mathbf{v}, \mathbf{d}, \hat{\alpha}, \hat{\beta}, \mathbf{w}) P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \hat{\alpha}, \hat{\beta}) d\mathbf{w}.$$

To estimate the two hyperparameters, the conditional joint pdf is expanded using Bayes theorem as:

$$P(\alpha, \beta|v, \mathbf{v}, \mathbf{d}) \propto P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) P(\alpha, \beta|v, \mathbf{v}). \quad (4.17)$$

Due to the lack of knowledge of the hyperparameters  $\alpha$  and  $\beta$  their joint prior  $P(\alpha, \beta|v, \mathbf{v})$  is assumed to be uniform and thus is relatively flat. Because of the previous assumption, maximizing the conditional joint pdf  $P(\alpha, \beta|v, \mathbf{v}, \mathbf{d})$  is equivalent to maximizing  $P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta)$  and as such, the true hyper parameters can be estimated as:

$$\begin{aligned} \hat{\alpha} &= \underset{\alpha}{\operatorname{argmax}} P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta), \\ \hat{\beta} &= \underset{\beta}{\operatorname{argmax}} P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta), \end{aligned} \quad (4.18)$$

The estimates of the hyperparameters require the computation of the likelihood function  $P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta)$ , which has the form:

$$P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) = \int_{\mathbf{w}} P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta) d\mathbf{w}, \quad (4.19)$$

Working out the convolution, the evidence function  $P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta)$  has the form:

$$P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) = \left( \frac{\beta}{2\pi} \right)^{\frac{N}{2}} (\alpha) |\Sigma_w^{-1}|^{-\frac{1}{2}} \exp \left[ \frac{-\beta}{2} \|\mathbf{d} - \Phi \mu_w\|^2 + \frac{\alpha}{2} \mu_w \mu_w^T \right].$$

Maximizing the evidence function is the same as maximizing its natural logarithm and as such, the hyper-parameters can be computed by setting the partial derivative of the logarithm of the evidence function with respect to the respective hyper-parameter to zero. The natural logarithm of the evidence function can be written as:

$$\ln P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) = \ln \alpha + \frac{N}{2} \ln \beta - \frac{\ln |\Sigma_w^{-1}|}{2} - \frac{N}{2} \ln(2\pi) - \frac{\beta}{2} \|\mathbf{d} - \Phi \mu_w\|^2 - \frac{\alpha}{2} \mu_w \mu_w^T.$$

The derivative equation with respect to  $\alpha$  is:

$$\frac{\partial \ln P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta)}{\partial \alpha} = \frac{1}{\alpha} - \frac{1}{2} \left[ \mu_w \mu_w^T + \frac{\partial \ln |\Sigma_w^{-1}|}{\partial \alpha} \right]. \quad (4.20)$$

The determinant of the matrix  $\Sigma_w^{-1}$  can be rewritten in terms of the eigenvalues of the matrix  $\beta\Phi^T\Phi$  as:

$$|\Sigma_w^{-1}| = \prod_i (\lambda_i + \alpha).$$

Computing the partial derivative the following equation is obtained:

$$\frac{\partial \ln |\Sigma_w^{-1}|}{\partial \alpha} = \sum_i \frac{1}{\lambda_i + \alpha}. \quad (4.21)$$

Setting the partial derivative in (4.20) to zero, the hyper-parameter  $\alpha$  will have the form:

$$\alpha = \frac{1}{\mu_w \mu_w^T} \sum_i \frac{\lambda_i}{\lambda_i + \alpha}. \quad (4.22)$$

Similar analysis is done with respect to the hyper-parameter  $\beta$  to obtain:

$$\frac{1}{\beta} = \frac{1}{N - \sum_i \frac{\lambda_i}{\lambda_i + \alpha}} \sum_{n=1}^N [d_n - \mu_w^T \phi^*(v_n)]^2. \quad (4.23)$$

It is noted that both solutions are implicit solutions of the parameters themselves. To solve for the hyper-parameters, an initial value must be chosen to calculate  $\mu_w$  and the sum  $\sum_i \frac{\lambda_i}{\lambda_i + \alpha}$  and then compute  $\alpha$  and  $\beta$  using (4.22) and (4.23) until convergence, which is determined when the difference between the old and new value of the hyper-parameters is less than a specified tolerance. The tolerance is set to a very low value of  $10^{-10}$  for both hyper parameters. Furthermore, the initial value of the hyperparameter does not affect the convergence of the algorithm.

#### 4.3.4 Using The Learned PDF For Training Pixels Extraction

After learning the hyperparameters  $\hat{\alpha}$  and  $\hat{\beta}$  from (4.22) and (4.23) respectively,  $\Sigma_p$  can be computed from (4.16) resulting in a tractable form of the predictive distribution written as:

$$P(d|v, \mathbf{v}, \mathbf{d}, \hat{\alpha}, \hat{\beta}) = \mathcal{N}(d; \mu_w^T \phi^*(v), \Sigma_p). \quad (4.24)$$

A visual representation of the predictive distribution can be seen in Fig.4.3. The predictive distribution (4.24) is used to extract training pixels by labeling pixels with a disparity value  $d$  belonging to a certain confidence interval as training pixels. Fig.4.4 shows the effect of varying the confidence interval on the quality and quantity of training pixels. The first row shows training pixels extracted at 90% confidence interval. The confidence interval is decreased by 20% for each lower row up to the last row, which shows training pixels extracted at 10% confidence interval. It can be noted that as the width of the confidence interval decreases, the number of correctly labeled training pixels (Green) decreases while the number of outliers (Red) increases. It can be seen that as the confidence interval decrease, less but more precise training pixels are obtained.

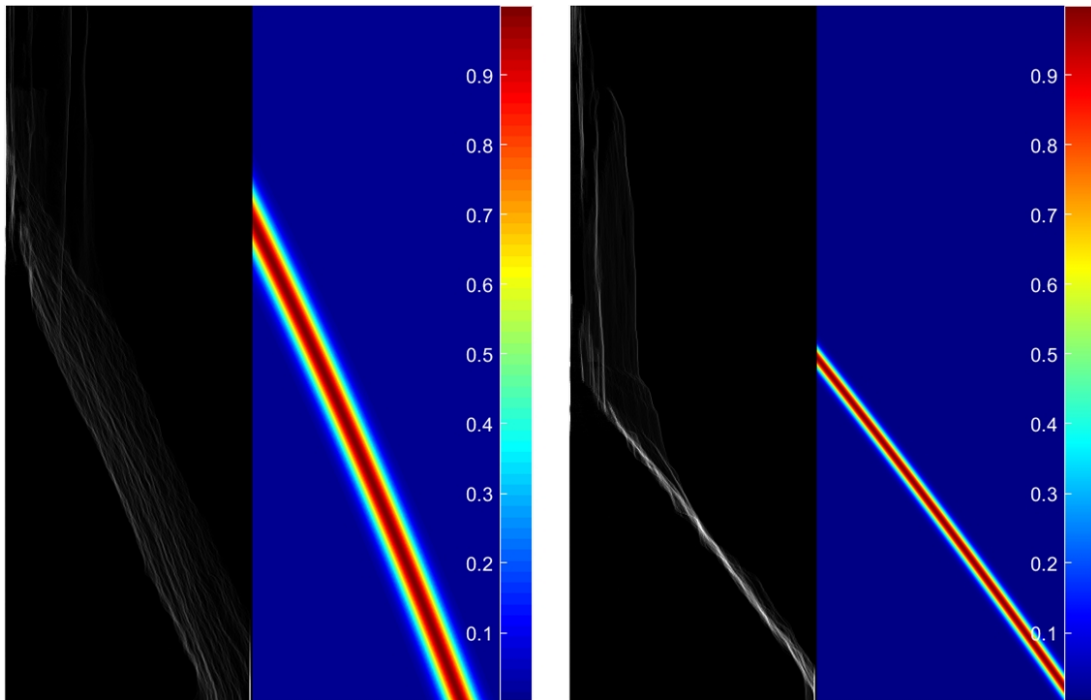


Figure 4.3: The original v-disparity image (black and white) and the resulting probability density function in equation (4.24) estimated from Bayesian linear regression (colored). The estimated PDF can be seen to closely resemble the ground correlation line, with an additional probabilistic interpretation. The color of each pixel determines the probability of a point in the v-disparity image to belong to the ground class.

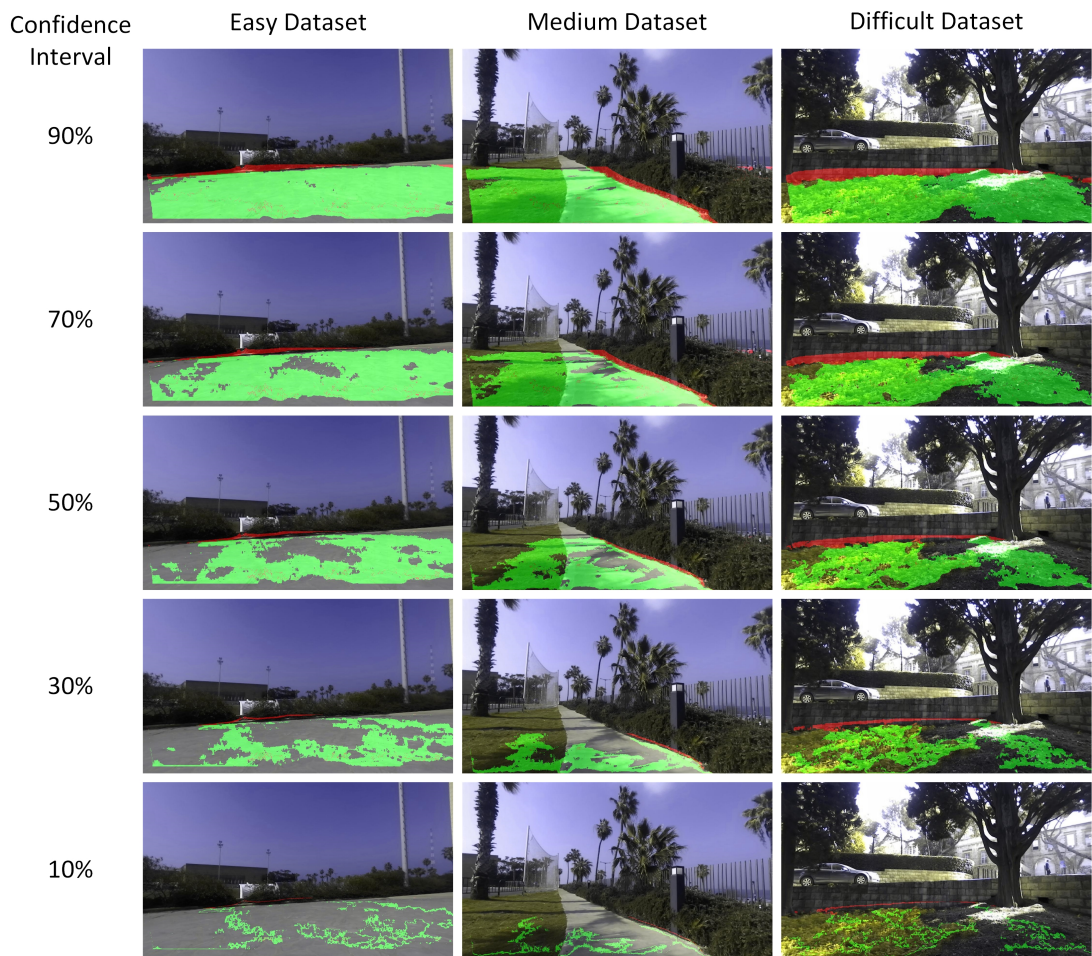


Figure 4.4: The effect of varying the confidence interval on the amount and quality of training pixels extracted in the image.

## 4.4 The Second Stage Classifiers

The training data extraction algorithm proposed earlier provides incomplete pixel labels that contain no negative samples. The algorithm separates  $N$  image pixels into two subsets:  $TP$  denoting pixels with a label  $l_{free} = 1$ , and  $UP$  denoting unlabeled pixels with  $l = \emptyset$ . Furthermore, each pixel in both subsets is assigned an  $M$  dimensional feature vector  $\vec{f} = (f_1 \dots f_m)$ . The second stage classification task is thus defined as providing a label  $l \in \{free, occupied\} = \{1, 0\}$  for every pixel in  $UP$ . This section provides two classifiers that can be used to perform this task.

### 4.4.1 The Positive Naive Bayes Classifier

The Positive Naive Bayes (PNB) classifier as defined by Denis et al.[20] estimates the probability of a pixel to belong to a class by counting the frequencies of its observed features. It then provides unlabeled pixels a label  $l$  according to:

$$l = \underset{l \in \{0,1\}}{\operatorname{argmax}} P(l|f_1, f_2, \dots, f_m) \quad (4.25)$$

$$= \underset{l \in \{0,1\}}{\operatorname{argmax}} \eta P(l) P(f_1, f_2, \dots, f_m|l), \quad (4.26)$$

where  $\eta$  is a normalization coefficient, and where (4.26) was derived from (4.25) through Bayes rule. Assuming conditional independence of each component of the feature vector, (4.26) reduces to:

$$l = \underset{l \in \{0,1\}}{\operatorname{argmax}} \eta P(l) \prod_{i=1}^M P(f_i/l). \quad (4.27)$$

For now, it is assumed that each component of the feature vector lies in a strictly positive discrete feature space such that  $f_i \in [0, 2, \dots, K] \forall i \in [1, M]$ , creating a vocabulary  $V$  of discrete features. Furthermore, the features are assumed to be multinomially distributed given the class label such that  $P(f_i/l) \sim \text{multinomial}$ . The functions  $C(f_i, S)$  and  $C(S)$  are counting functions that return the number of occurrences of feature  $f_i$  in the set  $S$  and the number of elements of set  $S$  respectively. Mathematically, these functions are defined as:

$$C(f_i, S) = \sum_{j=1}^N \mathbb{1}\{f_i = f_j \wedge l = l_S\} \quad (4.28)$$

$$C(S) = \sum_{j=1}^N \mathbb{1}\{l = l_S\}, \quad (4.29)$$

where  $l_S$  is the label associated with a set  $S$  and  $\mathbb{1}$  is the indicator function. The PNB classifier estimates the positive class conditional probability for each component of the

feature vector as:

$$P(f_i|l=1) = \frac{\zeta_p + C(f_i, TP)}{\zeta_p \text{Card}(V) + C(TP)}, \quad (4.30)$$

where  $\text{Card}(V)$  is the cardinality of the vocabulary  $V$ , and  $\zeta_p$  is a smoothing parameter.

Estimating the negative class conditional probability is a non-trivial problem due to the absence of negative labeled training data. The derivation is formulated using the law of total probability is used to write the  $P(f_i)$  as:

$$P(f_i) = P(f_i|l=0)P(l=0) + P(f_i|l=1)P(l=1), \quad (4.31)$$

The negative probability is then written as:

$$P(f_i|l=0) = \frac{P(f_i) - P(f_i|l=1)P(l=1)}{P(l=0)}. \quad (4.32)$$

Furthermore,  $P(f_i)$  is estimated from the unlabeled data using the counting functions defined above as:

$$P(f_i) = \frac{C(f_i, UD)}{C(UD)}, \quad (4.33)$$

Finally, the negative class conditional probability estimate can be written as:

$$P(f_i/l=0) = \frac{1 + \max(0, C(f_i, UD) - P(l=1)P(f_i/l=1)C(UD))}{\text{Card}(V) + (1 - P(l=1))C(UD)}, \quad (4.34)$$

where the  $\max$  function was used to insure a non-negative probability [21], and where Laplace smoothing was applied. It has to be noted that the estimation of the prior probability  $P(l=1)$  directly from the available data is not possible, and should be provided as an input. Finally, equations (4.30) and (4.34) are substituted in (4.27), and labels can be generated for each pixel in the scene.

#### 4.4.2 The $\nu$ -Support Vector Classifier

The  $\nu$ -SVC was proposed by Scholkopf et al.[22] in 1999 and became a popular kernel based learning algorithm for one class classification problems. The  $\nu$ -SVC learns a hyperplane in a higher dimensional feature space, such that the set  $TP$  is separated from the origin with maximum margin. The hyperplane is found by solving the following quadratic program:

$$\begin{aligned} \min_{\alpha} & \frac{1}{2} \sum_{jk} \alpha_j \alpha_k k(\bar{f}_j, \bar{f}_k) \text{ subject to :} \\ 0 & \leq \alpha_j \leq \frac{1}{C(TP)}, \\ \sum_j & \alpha_j = \nu, \end{aligned} \quad (4.35)$$



where  $\bar{f}_1 \dots \bar{f}_{C(TP)}$  are feature vectors of pixels belonging to the set  $TP$ , and  $k(\bar{f}_j, \bar{f}_k)$  is some kernel function that maps the data to a higher dimensional feature space. The parameter  $\nu \in [0, 1]$  is related to the number of pixels to be considered as support vectors. After constructing the hyperplane, pixels in  $UD$  are given a label  $l$  such that:

$$l = \max(0, \text{sgn}(\sum_j \alpha_j k(\bar{f}_j, \bar{f}) - \rho)), \quad (4.36)$$

$$\rho = \sum_k \alpha_k k(\bar{f}_j, \bar{f}_k), \text{ for any } 0 \leq \alpha_k \leq \frac{1}{C(TP)}. \quad (4.37)$$

Unlike the PNB classifier the feature space need not be discrete. The kernel used in this implementation is the radial basis function (RBF) kernel with equation:

$$k(\bar{f}_j, \bar{f}_k) = \exp\left(\frac{-(\bar{f}_j - \bar{f}_k)^2}{2s^2}\right), \quad (4.38)$$

where  $s$  is the scale parameter. I refer the reader to the work of Scholkopf et al.[22] for a detailed analysis of the effect of the  $\nu$  and  $s$  on the classification results of the  $\nu$ -SVC.

## 4.5 Feature Space Selection and The Mapping Algorithm

The aim of this chapter is to present the feature space selection and the free space mapping algorithm.

### 4.5.1 Free Space Mapping

To be able to use the output of the proposed classifiers for occupancy grid mapping, alterations must be done to transform the binary output to usable probabilities. The world occupancy grid is initialized as a  $1000 \times 1000$  cell grid with each cell representing a  $10 \times 10$  cm world patch.

Initially, the occupancy probability of each cell is set to 0.5. The occupancy probability is then updated as the unmaned ground vehicle explores the environment using the following equation:

$$O_t = O_{t-1} + O_{sensor}, \quad (4.39)$$

where  $O_{t-1}$  represents the previous occupancy probability value in the cell, and  $O_{sensor}$  represents the current probability update. Obtaining  $O_{sensor}$  is specific to each classifier. For the PNB classifier,  $O_{sensor}$  is defined as:

$$O_{sensor} = \log(\eta P(l = 1) \prod_{i=1}^M P(f_i/l = 1)) - \log(\eta P(l = 0) \prod_{i=1}^M P(f_i/l = 0)). \quad (4.40)$$

The value computed in 4.40 is positive only if the first term i.e., the posterior probability of a cell to belong to the ground class is the larger term. Therefore, the occupancy probability for each cell within the field of view of the UGV will change at each frame in proportion to the difference between the two class posterior probabilities.

Modifying the result of the v-SVC is a bit trickier. At first, the raw score inside the  $sgn$  function in 4.37 is transformed via the logistic function to transform score values to the interval  $[0, 1]$ . The transformed value is then doubled, and 1 is subtracted from it to obtain the new occupancy probability as:

$$O_{sensor} = \frac{2}{1 + \exp(-\sum_j \alpha_j k(\bar{f}_j, \bar{f}) + \rho)} - 1. \quad (4.41)$$

The logic behind the above equation is that the logistic function maps any positive value to a value greater than 0.5 and any negative value to a value less than 0.5, and thus by doubling the transformed value and subtracting 1,  $O_{sensor}$  values between  $[-0.5, +0.5]$  are obtained.

## 4.5.2 The Selected Feature Space

The selection of discriminant features to be used by the second stage classifiers is essential for good classification results. However, to maintain real time performance, the computational requirement of extracting features should also be taken into consideration. In these notes a subset of appearance and geometric based features are chosen to provide a decent compromise between discriminating power and computational time.

The system proposed is collecting training data per frame and in *real time* and thus the features need not be temporally invariant. For this reason, the computational efficiency of features is used as the selection criterion. Table 4.1 provides the computational time required to extract different features used in literature. In the performed experiments, the three geometric features in Table 4.1 were found to be highly discriminant and thus were all chosen to belong to the feature space. From image data, the mean of the 3 dimensional RGB color space was chosen due to its low computational time requirement. The final 6 dimensional feature vector thus comprises of the mean height, height variance, maximum absolute difference in height, and the mean of R, G, and B channels.

The positive naive Bayes classifier requires the feature space to be discrete. Color features chosen above are already discrete, taking values between 0 and 255. On the other hand, the chosen geometric features take continuous values and as such require discretization before being learned by the PNB classifier. The discretization is performed according to the following equation:

$$f_{new} = 255 \times \frac{f_{old} - \min(f_{old})}{\max(f_{old}) - \min(f_{old})}, \quad (4.42)$$

producing feature values between 0 and 255 for all components of the feature vector.

Table 4.1: Computation time required to extract different features from  $720 \times 1280$  stereo images, pixel wise and block wise. It should be noted that histogram-based features are too computationally expensive to be defined pixel wise, and thus are defined only per  $5 \times 32$  pixel blocks. Furthermore, raw measurement data such as RGB color data and height data require no processing time.

Color Features			
Feature	Dimension	per Frame Pixel Wise	per Frame Block Wise
RGB[11],[12],[3],[8]	3	NA	69.2 ms
HSV[11],[23]	3	21.9 ms	77.7 ms
rg Chromaticity[5]	2	22.6 ms	74.2 ms
Lab[10]	3	331.4 ms	384.7 ms
Texture Features			
Feature	Dimension	per Pixel	per Block
RGB Histograms[24]	15	NA	2058 ms
HS Histograms[11]	13	NA	1338 ms
Gabor Magnitude[10]	4	681 ms	690 ms
LBP histograms[25]	59	NA	3545.2 ms
Geometric Features			
Feature	Dimension	per Pixel	per Block
Mean Height [2],[6],[12]	1	NA	22.8 ms
Height Variance [12],[26],[6],[13]	1	49 ms	50.7 ms
maximum absolute difference in height[12],[13],[4],[26],[2]	1	123.1 ms	122.5 ms

# Chapter 5

## Experiments And Results

This chapter presents the experimental setup used for data acquisition and describes the datasets acquired. It then provides an analysis of how well the feature distributions of the ground class are represented by the extracted training data. Finally, the classification and environment mapping results of the proposed second stage classifiers using my proposed algorithm for training data extraction are presented and analyzed.

### 5.1 Hardware

As shown in Fig.5.1, Stereo Lab's Zed Camera [27] is used to acquire  $720 \times 1280$  RGB images as well as 3D point clouds at 10 frames per second. The camera is mounted on the clearpath Husky UGV that inputs odometry and IMU information to an extended Kalman filter, which outputs the pose relative to the world coordinate frame. Training data extraction, feature vector extraction, free space classification, and occupancy grid mapping were implemented using Matlab and ran on an Intel Core<sup>®</sup> i7<sup>™</sup> processor at 3 GHz with 32 GB of RAM.

### 5.2 Datasets

To be able to perform the necessary experiments, three datasets were created with terrains ranging from planar to non-planar ground. Each frame in the dataset is comprised of a stereo pair of  $720 \times 1280$  colored images, their corresponding disparity image, and pixel  $X, Y,$  and  $Z$  coordinate with respect to the camera's coordinate frame. Furthermore, the frame contains odometry information representing the frame's position and orientation with respect to a world coordinate frame. The algorithm provided by the camera's SDK was used to generate the disparity image and the point cloud coordinates. The three datasets include:

**Difficult dataset:** 88 images taken with the a stereo camera mounted on a UGV driven on a highly non-planar off-road terrain.



Figure 5.1: Deployed sensors and robot. The Zed Stereo Camera is rigidly mounted on top of the Clearpath Husky UGV.

**Medium dataset:** 120 images taken with the a stereo camera mounted on UGV driven on a slightly non-planar park-like terrain.

**Easy dataset:** 145 images taken with the a stereo camera mounted on a UGV driven on a highly planar man-made terrain .

It has to be noted that pixels lacking geometric features due to rectification, occlusion, or being located beyond the stereo camera’s maximum range are not considered in this evaluation. Finally, ground truth is generated manually for every frame of the three datasets.

### 5.3 Assessing The Quality Of Extracted Training Pixels

Although I established that the proposed algorithm can indeed provide training pixels, the quality of the training pixels extracted remains an unanswered question. To begin with the analysis, goodness criteria needs to be specified. The extracted training data should contain a minimal number of outliers. Furthermore, it should be a representative sample of the class from which it has been extracted. It has to be noted that I set the confidence interval to be 30% for all the tests performed.

I begin with an analysis of the outlier fraction, that is the fraction of extracted pixels that are labeled as training pixels from the ground class, but do not actually belong to the ground class (Fig.5.2, Top). The bottom part of Fig.5.2 provides a plot of the

fraction of outliers in the extracted training data per frame of the three datasets. It can be seen that the training data extraction algorithm becomes more prone to erroneously label data as the environment becomes harsher. On the other hand, the mean outlier fraction produced by the algorithm is 0.0268, 0.0452, 0.1098 for the easy, medium and difficult datasets respectively, which is tolerable and can be handled through second stage classification. It has to be noted that the fraction of outliers can go as high as 0.2879 for the worst case frame in the difficult dataset.

I noticed that the outlier fraction is highly correlated with the quality of the disparity images, and tend to increase as the quality of the disparity images deteriorates. This is a natural outcome of the dependence of the proposed algorithm on the disparity space for training data extraction and I anticipate the proposed algorithm to produce a very low outlier fraction as disparity generating algorithms become better.

To assess how well the extracted training data resembles the true class distribution, a comparison of the sufficient statistics of the true distribution and the estimated distribution of the 6 features in the selected feature space is performed. Although an invalid assumption, and for the sake of comparison, the features are assumed to be normally distributed and their mean and standard deviation are compared. Table 5.1 present a comparison between the sufficient statistics of the true and estimated distributions from a randomly selected frame from each dataset. It can be seen that the estimated distributions' mean and variance closely resembles that of the true distribution with minimal error. This implies that the proposed algorithm accurately models the true feature distribution for the ground class.

To further validate the goodness of the estimation, the true distribution is visualized by plotting the normalized histogram of the all the pixels that belong to ground class, found from the ground truth. The estimated distributions' plots is superposed over that of true distributions' plots in Fig.5.3, Fig.5.4, Fig.5.5 for a random frame from each of the three datasets. In all three cases, the estimated probability distribution closely resembles the true distribution.

The reported results validate the claim that the proposed algorithm chooses a representative sample of the ground class as training data with a minimal number of outliers. The remainder of this section is focused on reporting the results of using the extracted training data to supervise classifiers for free space classification tasks and for environment mapping.

Table 5.1: A comparison of the true ground class distribution's vs the estimate ground class distribution sufficient statistics for a random frame from each of the three datasets. It can be seen that the estimated distribution

Easy Dataset						
Statistic	Red	Blue	Green	Mean Height	Height Variance	Height Difference
True Mean	151.1	151.4	149.7	24.7	5.1	12.5
Estimated Mean	153.7	153.8	152.3	24.7	5.2	11.8
True Variance	323	323	291	5	115	1867
Estimated Variance	290	311	330	6	113	1804
Medium Dataset						
Statistic	Red	Blue	Green	Mean Height	Height Variance	Height Difference
True Mean	112.9	130.3	100.9	15.4	2.2	13.2
Estimated Mean	114.6	132.2	102.5	15.2	1.9	12
True Variance	682	431	662	4	25	1842
Estimated Variance	680	431	639	6	23	1376
Difficult Dataset						
Statistic	Red	Blue	Green	Mean Height	Height Variance	Height Difference
True Mean	71.5	90	69.7	5.6	1.6	14.4
Estimated Mean	66.3	78.4	64	5.	1.8	14.9
True Variance	844	1531	805	2	12	1616
Estimated Variance	718	981	637	2	13	1714

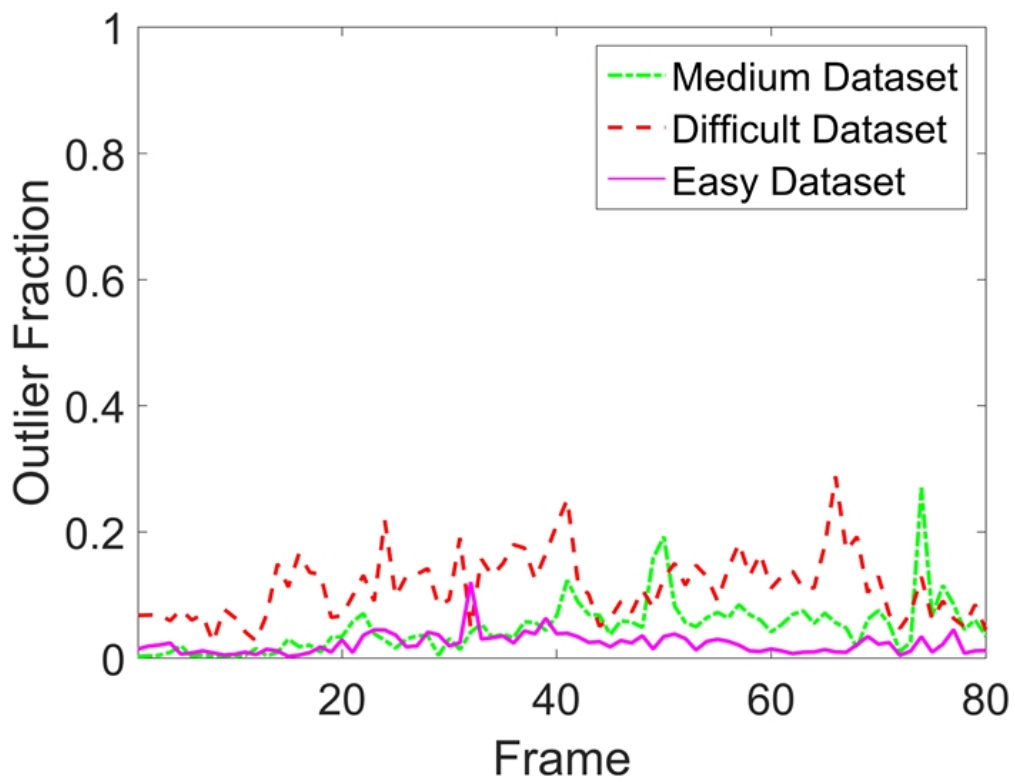
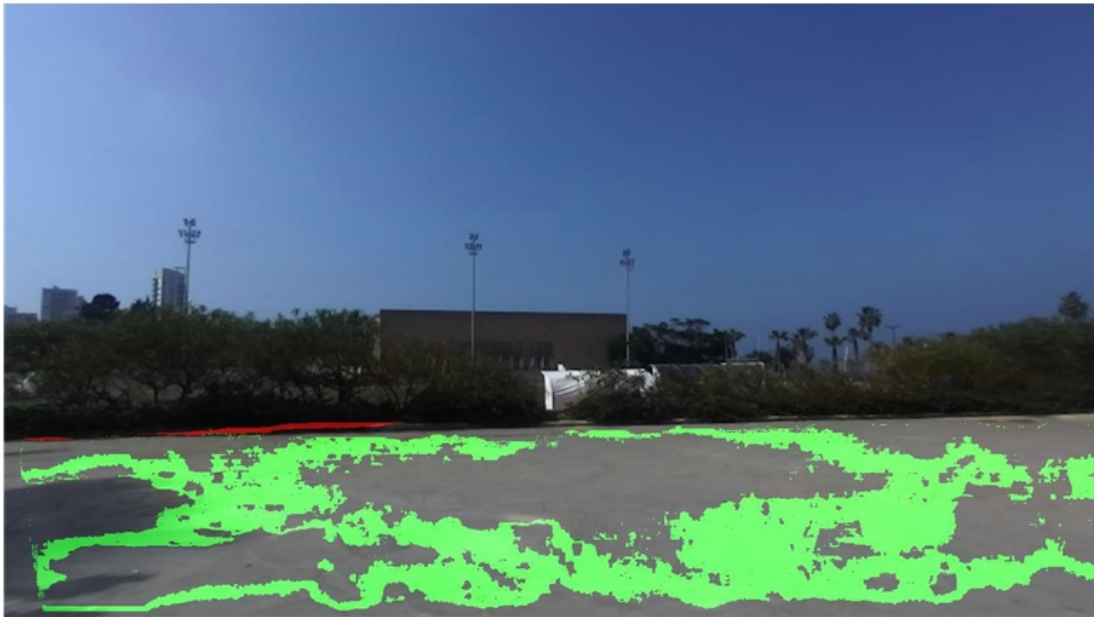


Figure 5.2: Top: An example of the presence of outliers in the extracted training pixels. Correctly labeled training pixels are shown in green, where as outliers are shown in red. Bottom: The fraction outliers computed by running the proposed algorithm over the three datasets.



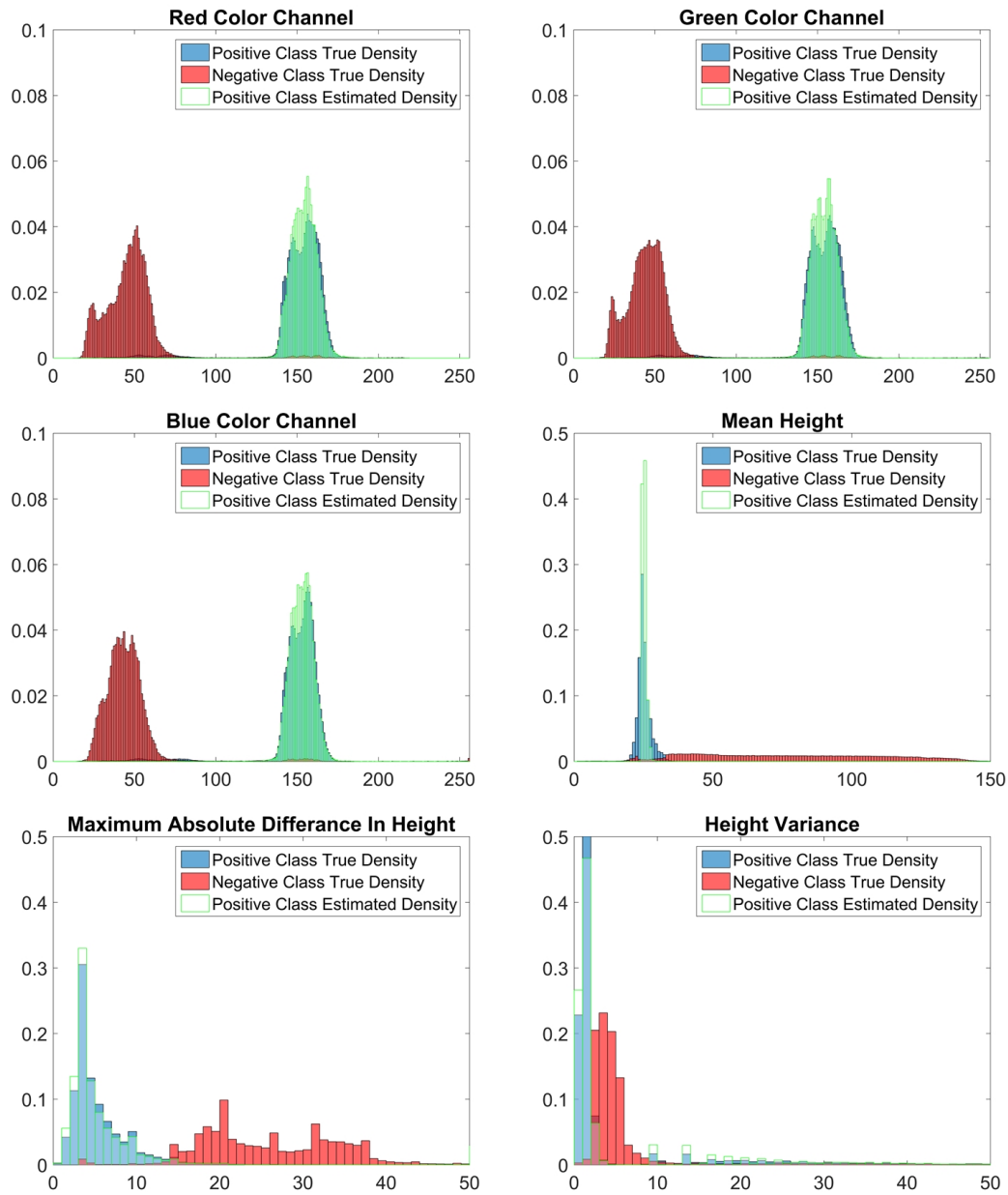


Figure 5.3: The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the easy dataset. The estimated probability distribution for the positive class (green) is found using training pixels extracted via my algorithm. The features are all linearly separable, and unimodal thus approximated well by a normal distribution.

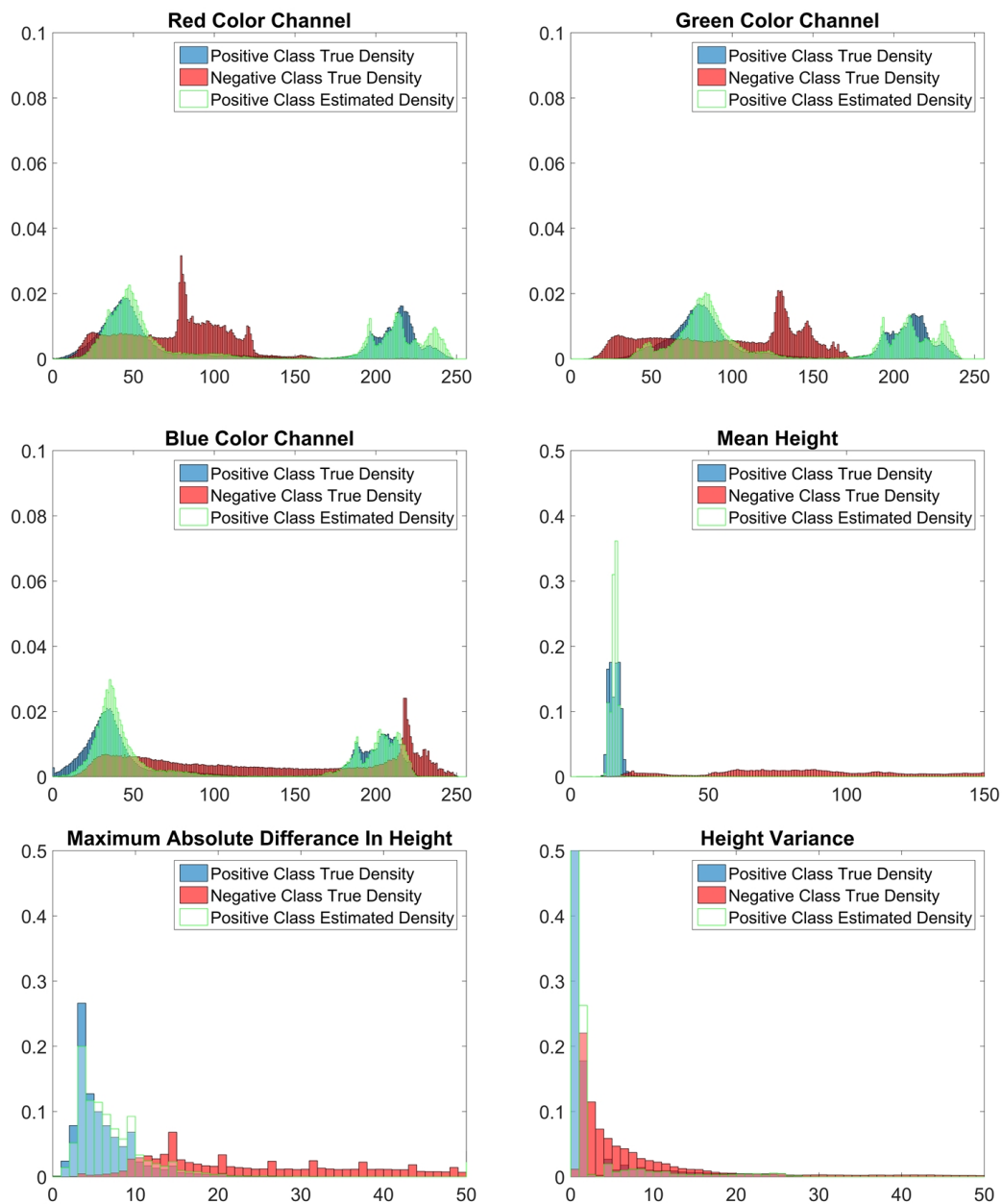


Figure 5.4: The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the medium dataset. The estimated probability distribution for the positive class (green) is found using training pixels extracted via my algorithm. The color features' distributions are seen to exhibit two modes due to the fact that the ground class is characterized by two different colors in this dataset. Furthermore, the geometric features are seen to still be linearly separable and unimodal.

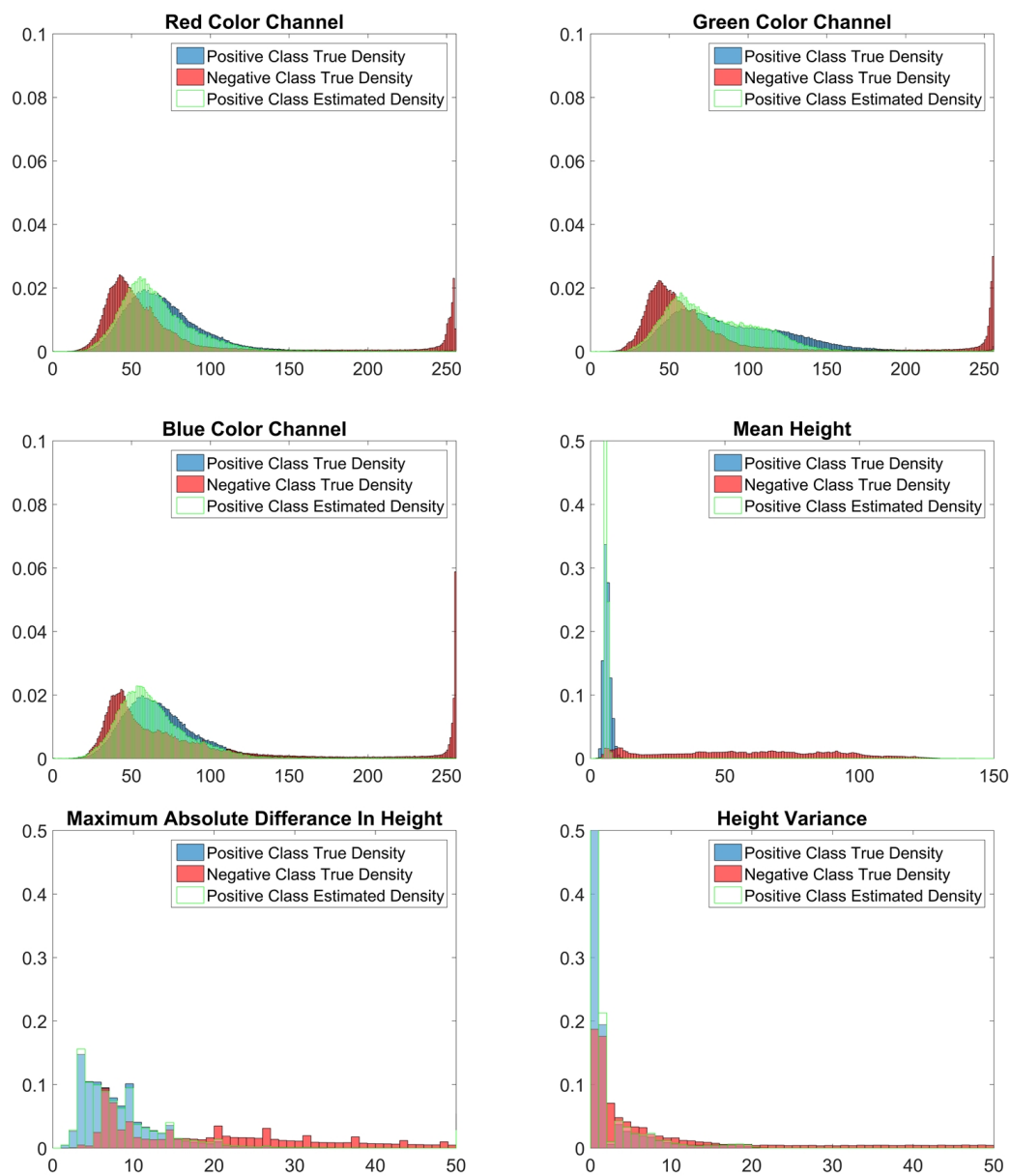


Figure 5.5: The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the difficult dataset. The estimated probability for the positive class (green) is found using training pixels extracted via my algorithm. The color features' positive and negative class distributions seem to be superposed and highly inseparable. Furthermore, positive and negative class distributions of the height variance and the maximum absolute difference in height are also highly inseparable.

## 5.4 Comparison With Other Training Data Extraction Algorithms In Literature

This section presents an analysis of the goodness of training data extracted by the proposed algorithm by comparing it against training data extracted via other techniques in the literature. This is done by inputting each set of data to the  $\nu$ -SVC and comparing the three corresponding output pixel labels.

Two training data extraction algorithms are used for the sake of comparison with the proposed algorithm. The two algorithms are:

**Bootstrapping:** Bootstrapping was used in [6] and relies on the assumption that the properties of the ground plane will not change much as the UGV moves through the environment. This algorithm is implemented by manually providing the robot with positive labels in the first frame, which it then uses as training data for classification in the second frame. Positively labeled data in the second frame are used as training data in the third and so on.

**Plane Fitting:** this algorithm was used in [8] and [11] and relies on plane fitting in the stereo generated point cloud to determine patches belonging to the ground plane. For maximum robustness towards outliers, M-estimator SAmple Consensus (MSAC) algorithm is used for plane fitting. The expected normal vector of the ground plane is required to be provided as an input, and inlier points determined by the algorithm are used as training data input to the  $\nu$ -SVC algorithm.

A standard off-the-shelf implementation of the  $\nu$ -SVC is used and thus only the selection of free parameters in this implementation will be discussed. The three free parameters in the  $\nu$ -SVC algorithm are  $\nu$ , the kernel scale, and the outlier fraction.  $\nu$  is a parameter that lies between 0 and 1 and controls the fraction of training data to become support vectors. In this implementation, it is set to 1 which results in using all training data as support vectors. The outlier fraction, which determines the percentage of training data to belong to the negative class is set to be 5%. This combination of  $\nu$  and outlier fraction allows the  $\nu$ -SVC classifier to be robust to only small amounts of wrong labels in the training data. A large amount of wrongly labeled training data will change the shape of the decision boundary, emphasizing the effect of training data extraction algorithms on the quality of the final pixel classification and allowing an objective comparison between training extraction algorithms. As part of the algorithm, the scale of the Gaussian kernel is selected automatically, using a heuristic procedure based on training data subsampling. Finally, due to the difference in their scale, features are standardized by subtracting their mean value and dividing by their standard deviation.

All experiments were done with the feature vector and  $\nu$ -SVC parameters held constant across all three datasets. Furthermore, the free parameters of the three training data extraction methods are also fixed over all trails. The labels obtained from the  $\nu$ -SVC using the three algorithms are compared to ground truth labels to compute three performance criteria, which are the recall, precision, and specificity.

Table 5.2: Evaluation of the  $\nu$ -SVC using the three training data extraction algorithms over the three datasets. The evaluation is based on the average recall, precision, specificity, and computation time ( of the training data extraction algorithm, in seconds) over all the frames of each dataset. As the terrain becomes harsher, my algorithm proves to produce better results.

$\nu$ -SVC using My Training Extraction Algorithm				
Dataset	Recall	Precision	Specificity	Time
Easy	0.8671	0.9604	0.9853	0.0547
Medium	0.8514	0.9326	0.9781	0.0592
Difficult	0.8147	0.9855	0.9725	0.0598
$\nu$ -SVC using plane fitting				
Dataset	Recall	Precision	Specificity	Time
Easy	0.9646	0.9340	0.9731	0.1681
Medium	0.9422	0.8931	0.9592	0.4833
Difficult	0.5784	0.9960	0.9957	1.1138
$\nu$ -SVC using Bootstrapping				
Dataset	Recall	Precision	Specificity	Time
Easy	0.0326	0.9787	0.9995	NA
Medium	0.0869	0.9853	0.9963	NA
Difficult	0.2838	0.9959	0.9992	NA

Recall describes the fraction of ground patches retrieved by the classifier, while precision describes fraction of the retrieved patches that are correct. Specificity on the other hand, describes the fraction of correctly identified negative instances, which in this case are the obstacles. The proposed algorithm’s aim is two-fold, first to maximize all three performance criteria of the  $\nu$ -SVC classifier and second, to keep its performance relatively the same over all the three types of terrain. Table 5.2 summarizes the mean recall, precision, and specificity of the  $\nu$ -SVC classifier using training data from the three algorithms over all the frames of each datasets.

The  $\nu$ -SVC classifier using Bootstrapping performed the worst of all three having a mean recall of 0.134 over the three datasets and is found to be unusable for reliable free space estimation. The low recall is attributed to the deterioration of the classification as the camera moves away from its initial position due to the change in the properties of the ground. This phenomenon can be clearly seen in Fig.5.6-left where the recall is plotted as function of frames. Better relative performance of the  $\nu$ -SVC using Bootstrapping on the Difficult dataset is mainly due to the constant color properties of the ground in this dataset. The results of the  $\nu$ -SVC using Bootstrapping are shown in the fifth column of Fig.5.7. At the early frames of operation (third and fourth rows), it provides good results, while at later frames (first, second and fifth rows), the quality of classification greatly deteriorates. One advantage of bootstrapping is its low computation time due to the low requirements for training data extraction.

The  $v$ -SVC utilizing plane fitting reaches 0.9646 and 0.9422 recall on the easy and medium dataset respectively. Compared to the  $v$ -SVC using my algorithm, which has a recall of 0.8671 and 0.8114 on the same datasets, the  $v$ -SVC utilizing plane fitting seems to perform better. I attribute the better performance to the much larger amounts of training data provided by plane fitting in cases of planar ground. However, the increase in recall comes at the expense of a decrease in precision and specificity. On the two datasets, the  $v$ -SVC using my algorithm achieves a precision of 0.9604 and 0.9326 respectively vs 0.9340 and 0.8931 for the  $v$ -SVC using plane fitting. The specificity of the  $v$ -SVC using my algorithm was also better, achieving 0.9853 and 0.9781 on the two datasets vs a specificity of 0.9731 and 0.9592 for the  $v$ -SVC using plane fitting. On the Difficult dataset, a deterioration in the quality of classification of the  $v$ -SVC using plane fitting was observed. In highly non planar environments, plane fitting only provides training data from the largest locally planar patch with a normal vector closest to that provided as input for the algorithm (Fig.5.7 third column, first row). This leads to a reduction in recall to a value of 0.5784. As the recall decreases, the precision increases to 0.9959 and the specificity to 0.9992. On the other hand, the  $v$ -SVC using my algorithm is able to provide a recall value of 0.8147, providing an increase of 0.2368 over the recall of the  $v$ -SVC using plane fitting. This high recall is accompanied with high values of precision and specificity, 0.9855 and 0.9725 respectively. This shows that my algorithm is able to provide reliable training data on highly non planar terrain.

Another important criterion to consider is the computation time of each training extraction algorithm. The proposed algorithm includes  $v$ -disparity image generation, filtering and Bayesian linear regression and was implemented in Matlab, as were the other two data extraction algorithms. All the algorithms ran on the same Laptop. The fifth column of Table 5.2 shows that as the nature of the scene becomes more non-planar, the computation time of plane fitting increases. Furthermore, Fig.5.6-right shows that the variance of the computation time between frames is very large for plane fitting, which is mainly due to the dependence of its computation time on the density of the point cloud. The proposed algorithm shows a more consistent computation time whether across datasets (Table 5.2, fifth column) or across frames (Fig.5.6).

The intuition behind the improved performance provided by the  $v$ -SVC using my proposed algorithm for training data extraction is that in non-planar environments, the ground plane is actually made up of many small oblique and horizontal planes, which are all projected to slanted lines in the  $v$ -disparity image. Using the  $v$ -disparity filtering algorithm to extract these lines is conceptually equivalent to fitting planes to the whole scene in one shot. This allows the user to extract training data over the whole scene even in highly non-planar scenarios (Fig.5.7-first column, first row) and results in the computational time of my algorithm to remain approximately the same whether the terrain is planar or non-planar. Finally, selecting training data by using the confidence interval allows picking only high confidence pixels for training, increasing the final classification's precision and specificity. Such examples can be seen in the final row of Fig.5.7, where the training data provided by my algorithm results in better classifi-

cation results. Plane fitting can be seen to provide a large amount of wrongly labeled training pixels resulting in a deterioration in the quality of the final classification.

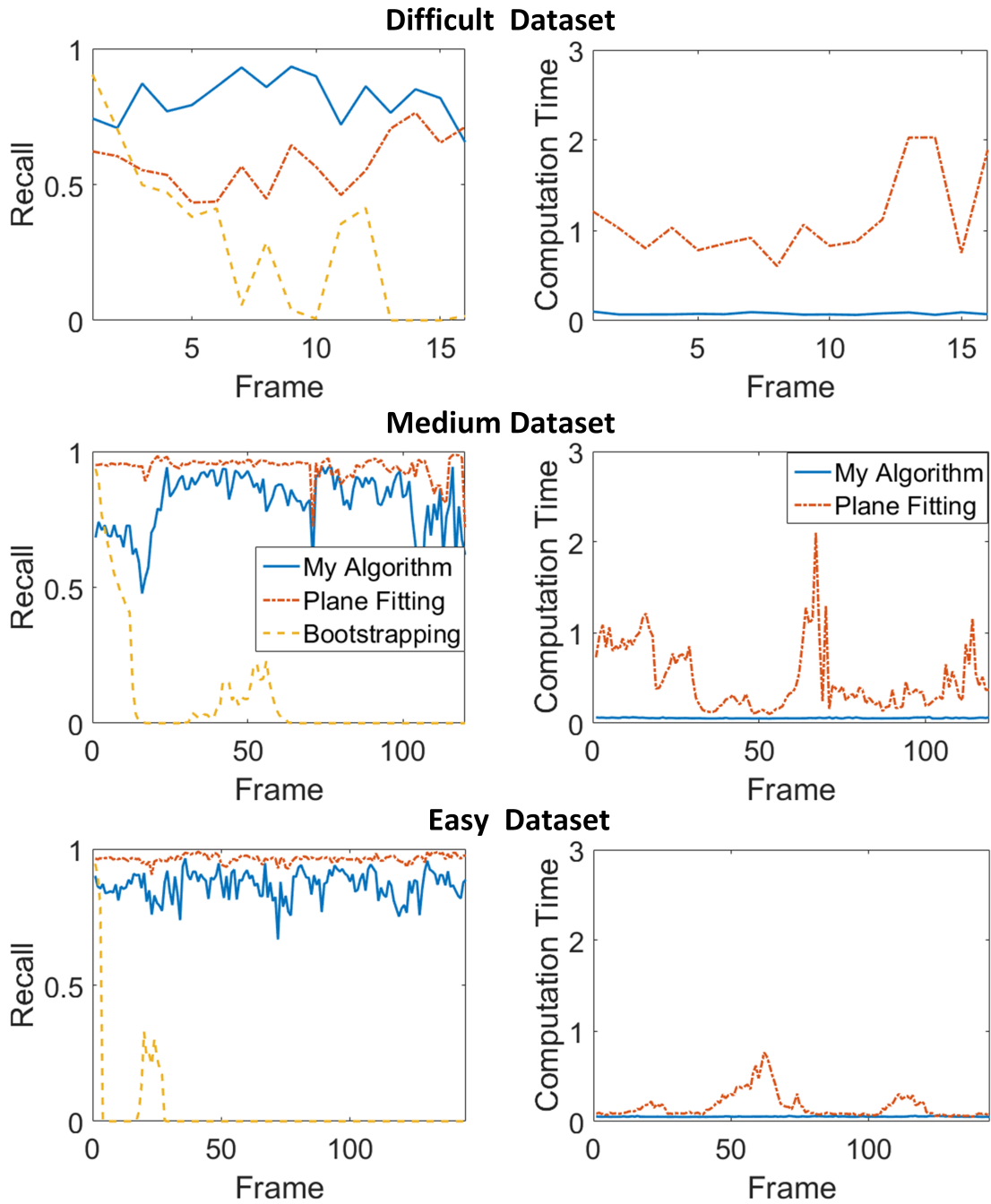


Figure 5.6: Plots of recall values of the  $\nu$ -SVC and computational time (in seconds) of the three training data extraction methods per frame of the datasets.



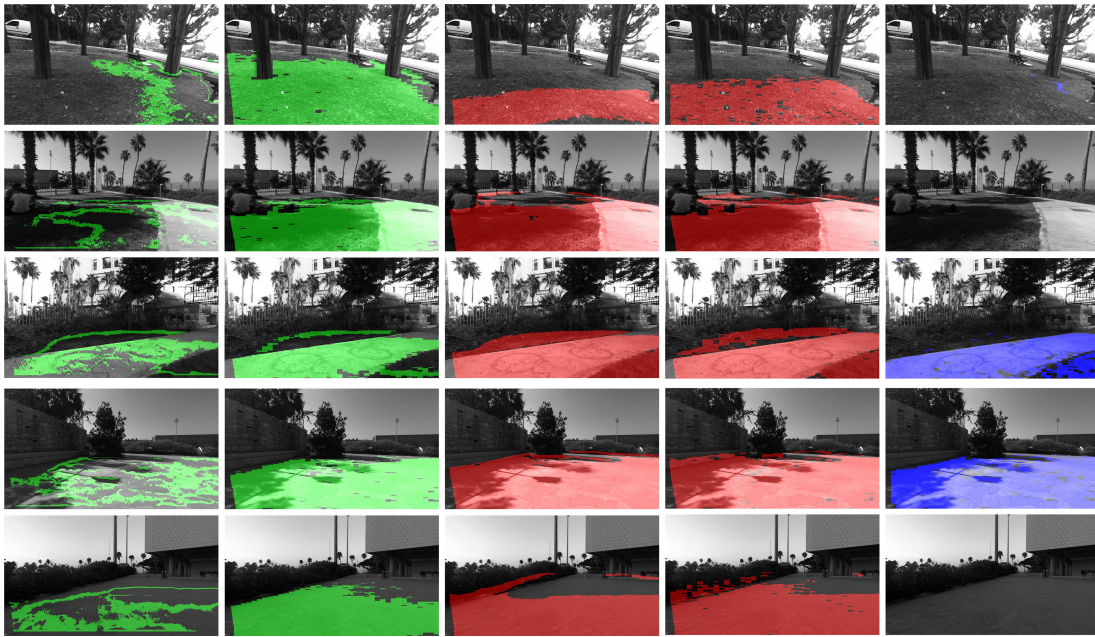


Figure 5.7: Examples of the training data extracted using my algorithm and the plane fitting algorithm (first and third columns respectively), and the final classification results obtained from  $\nu$ -SVC using my algorithm (green), largest fitted plane algorithm (red) and bootstrapping (blue). Bootstrapping does not explicitly extract training data at each frame and thus only results of the final classification are shown.

## 5.5 Free Space Classification and Mapping Results

All experiments were done with the feature vector and classifier parameters held constant across all three datasets. Furthermore, the confidence interval of the training data extraction algorithm is set to 30%. The labels obtained from the two classifiers are compared to ground truth labels to compute three performance criteria, which are the recall, precision, and specificity. Examples of these labels are presented in Fig.5.9 for the PNB classifier and Fig.5.10 for the  $\nu$ -SVC classifier. It has to be noted that the PNB classifier classifies each pixel in the  $720 \times 1280$  images, where as the  $\nu$ -SVC classifies  $5 \times 32$  image blocks.

Fig.5.8 present a comparison between the three performance criteria of the two classifiers. On the easy dataset, the PNB classifier performs better than the  $\nu$ -SVC classifier achieving a mean recall value of 0.9131 with a mean precision of 0.9823 and mean specificity of 0.9936. The  $\nu$ -SVC classifier on the other hand achieved a mean recall of 0.8853, a mean precision of 0.9713, and a mean specificity of 0.9897. I attribute the better performance of the PNB classifier to the linear separability of individual features in the proposed feature vector and to the resemblance of their probability densities to that of the normal distribution as it can be seen in Fig.5.3.

When applied on the medium dataset on the other hand, the  $\nu$ -SVC classifier seems to perform better with a mean recall value of 0.8549, mean precision of 0.9416 and mean specificity of 0.9802 versus a mean recall of 0.8326, mean precision of 0.9540 and mean specificity of 0.9847 for the PNB classifier. The better performance of the  $\nu$ -SVC is primarily due to the complex nature of the scene. The ground class color distribution as it can be seen in Fig.5.4 is multimodal and not linearly separable, and thus it is expected that the  $\nu$ -SVC classifier performs better than the PNB classifier in such scenarios.

Finally, the performance of the  $\nu$ -SVC classifier on the difficult dataset is also better than the PNB classifier, achieving a mean recall of 0.8795 with a mean precision of 0.8561 and specificity of 0.9479. The PNB classifier on the other hand achieved a mean recall of 0.8590 with a mean precision of 0.8872 and mean specificity of 0.9611. The reason behind the better performance is that five out of six features in the difficult dataset are seen not to be linearly separable in Fig.5.5.

The performance of mapping the classifiers is also of importance. The mapping procedure is performed by first performing coordinate transformation to align the camera coordinate frame with the robot coordinate frame. The 3-D point cloud is then projected onto the  $X$  and  $Y$  2D-plane. The mapping procedure follows the description in section 4.5.1. Points with the same  $X, Y$  are handled by addition of their log odds.

Fig.5.11 shows the maps of the environment of the three datasets created by the two classifiers. For the easy dataset, the performance of the two classifiers is relatively close. On the other hand, the  $\nu$ -SVC outperforms the PNB classifier on the medium and difficult datasets. It can be seen that in the medium dataset, the PNB classifier cannot find a path as it wrongly classifies free space as obstacles.

It can be observed from the experiments performed that the  $\nu$ -SVC classifier is

much more conservative than the PNB classifier. This is because the probabilistic output of the  $v$ -SVC (Fig.5.10) is very close to either 0 or 1, while that of the PNB classifier (Fig.5.9) is more spread out on the  $[0, 1]$  interval providing more levels on the occupancy grid.

Computation time should also be taken into account when evaluating the performance of the two classifiers. The computation time is measured as the time required to extract training data using the proposed algorithm, perform the classification, and construct the occupancy grid representation. Fig.5.8 shows the results of the computation time in seconds for both classifiers. It can be clearly seen that the PNB classifier requires less computation time than the  $v$ -SVC classifier. The PNB classifier requires 0.46, 0.5, 0.58 seconds per frame from the easy, medium, and difficult datasets respectively vs 0.56, 0.57, 0.6 seconds per frame for the  $v$ -SVC classifier.

As a final thought, both classifiers manage to map the environment in the three datasets fairly well. The PNB classifier classifies the environment pixel wise, and as such is more susceptible to noise, but is better in detecting boundaries. Furthermore, the PNB classifier is seen to be faster than the  $v$ -SVC classifier, and provides continuous probability values for each pixel. On the other hand, the  $v$ -SVC classifier performs better when features are not linearly separable. As a final recommendation, one should use the PNB classifier in man-made, planar environments, and the  $v$ -SVC classifier in tough non-planar environments.

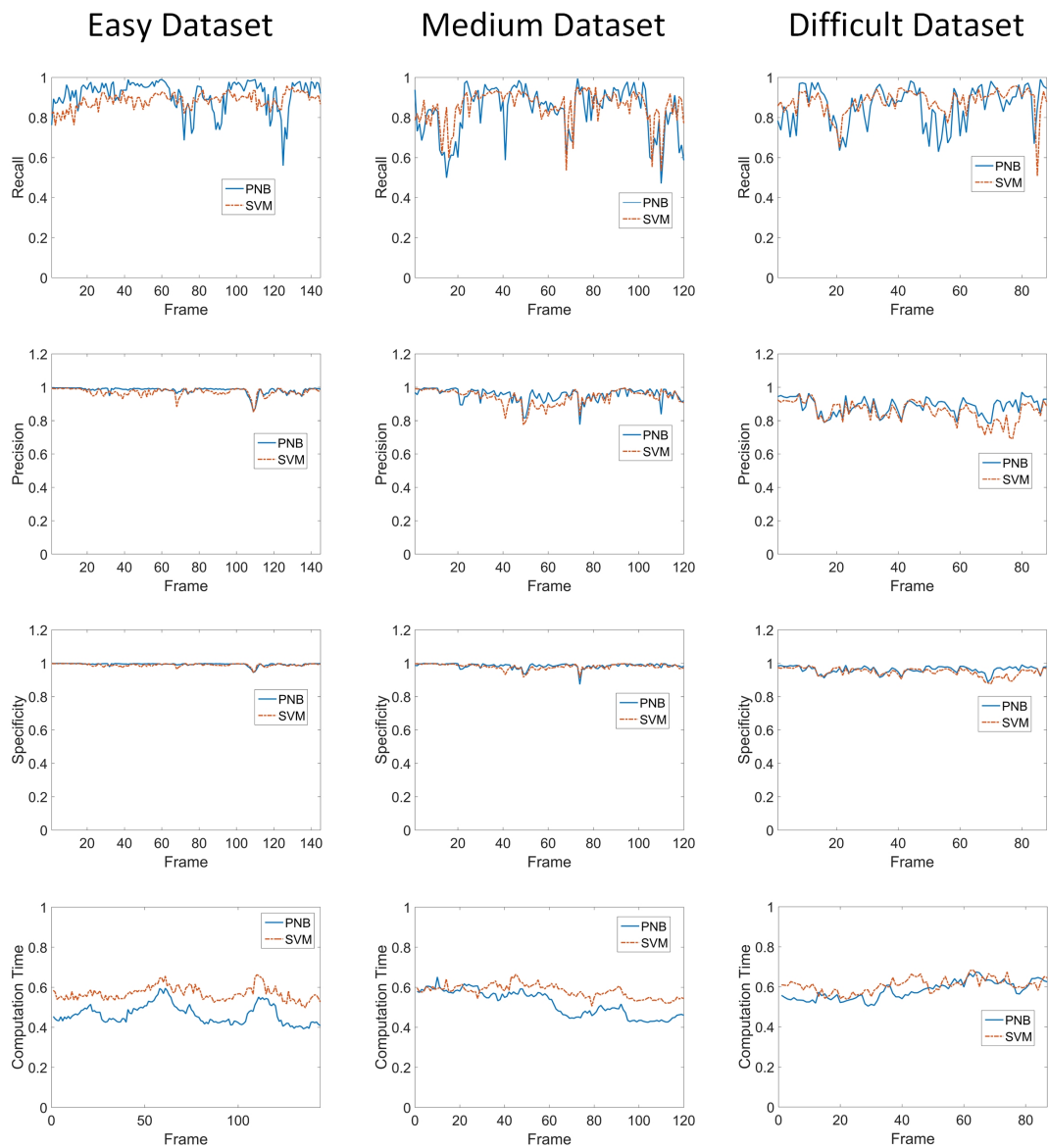


Figure 5.8: A plot of the Recall, Precision, Specificity, and Computation time (in seconds) of the PNB and  $\nu$ -SVC classifiers vs the frames of the three datasets. It is noticed that the two classifiers are similar in terms of classification results, but the PNB classifier is faster.

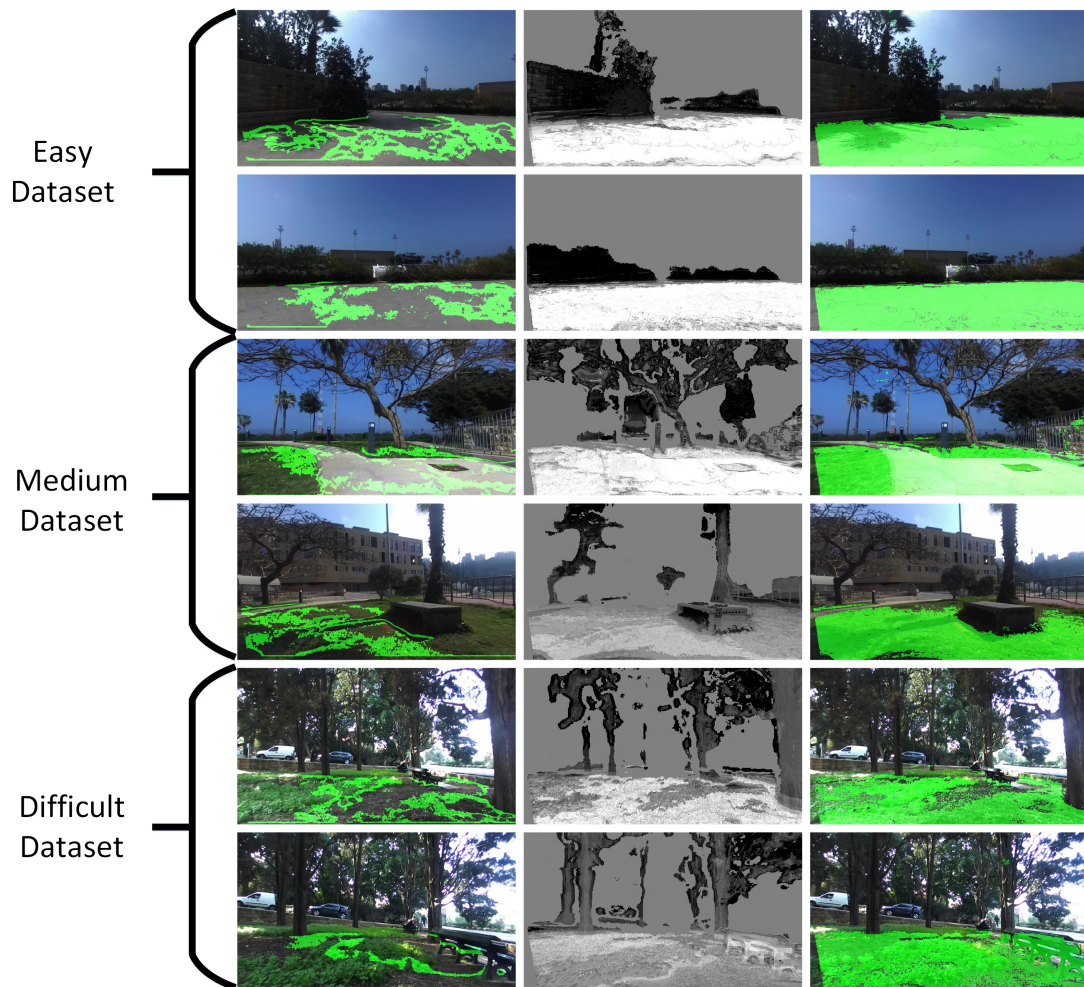


Figure 5.9: The results of training data extraction (first column), the occupancy frame (second column) and the final ground segmentation of the PNB classifier. The first two rows are from the easy dataset, the second two from the medium dataset, and the final two from the difficult dataset

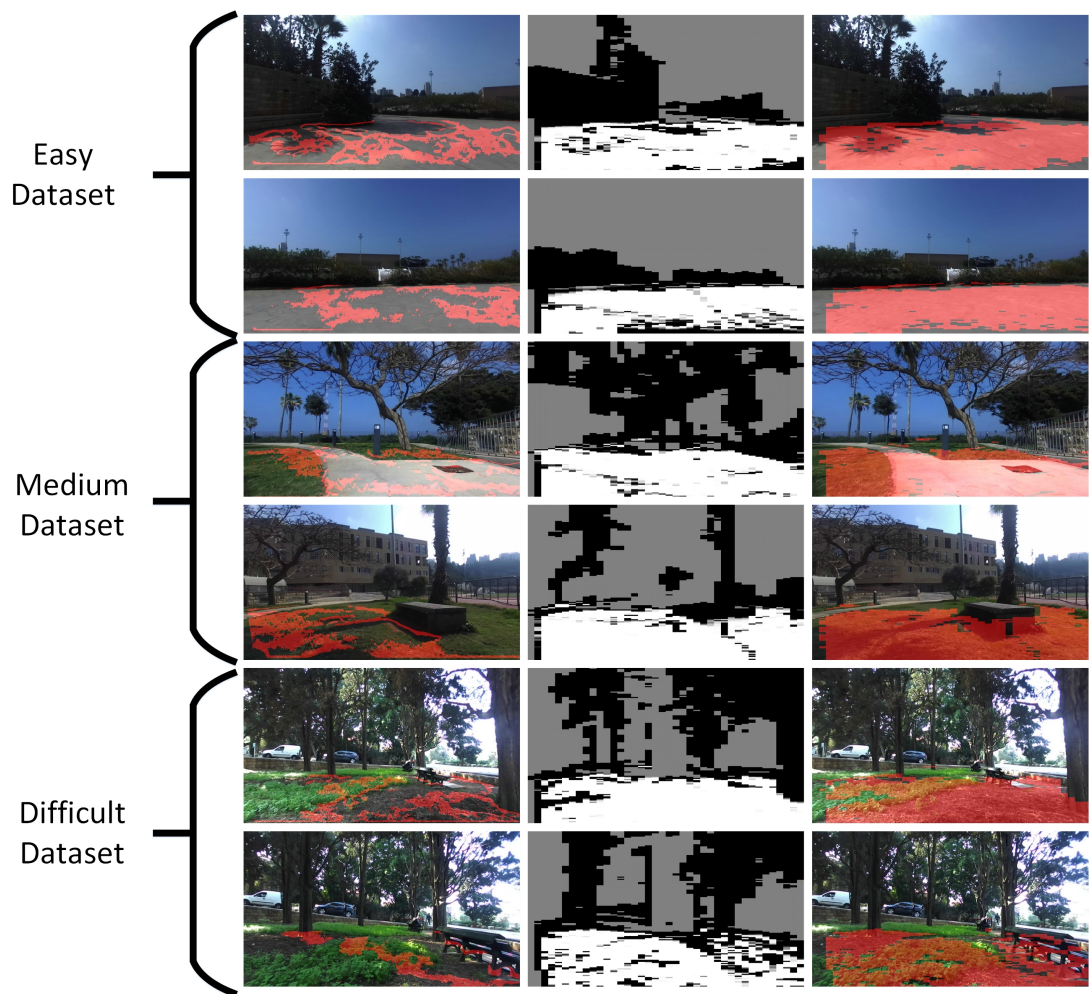


Figure 5.10: The results of training data extraction (first column), the occupancy frame (second column) and the final ground segmentation of the  $\nu$ -SVC classifier. The first two rows are from the easy dataset, the second two from the medium dataset, and the final two from the difficult dataset.

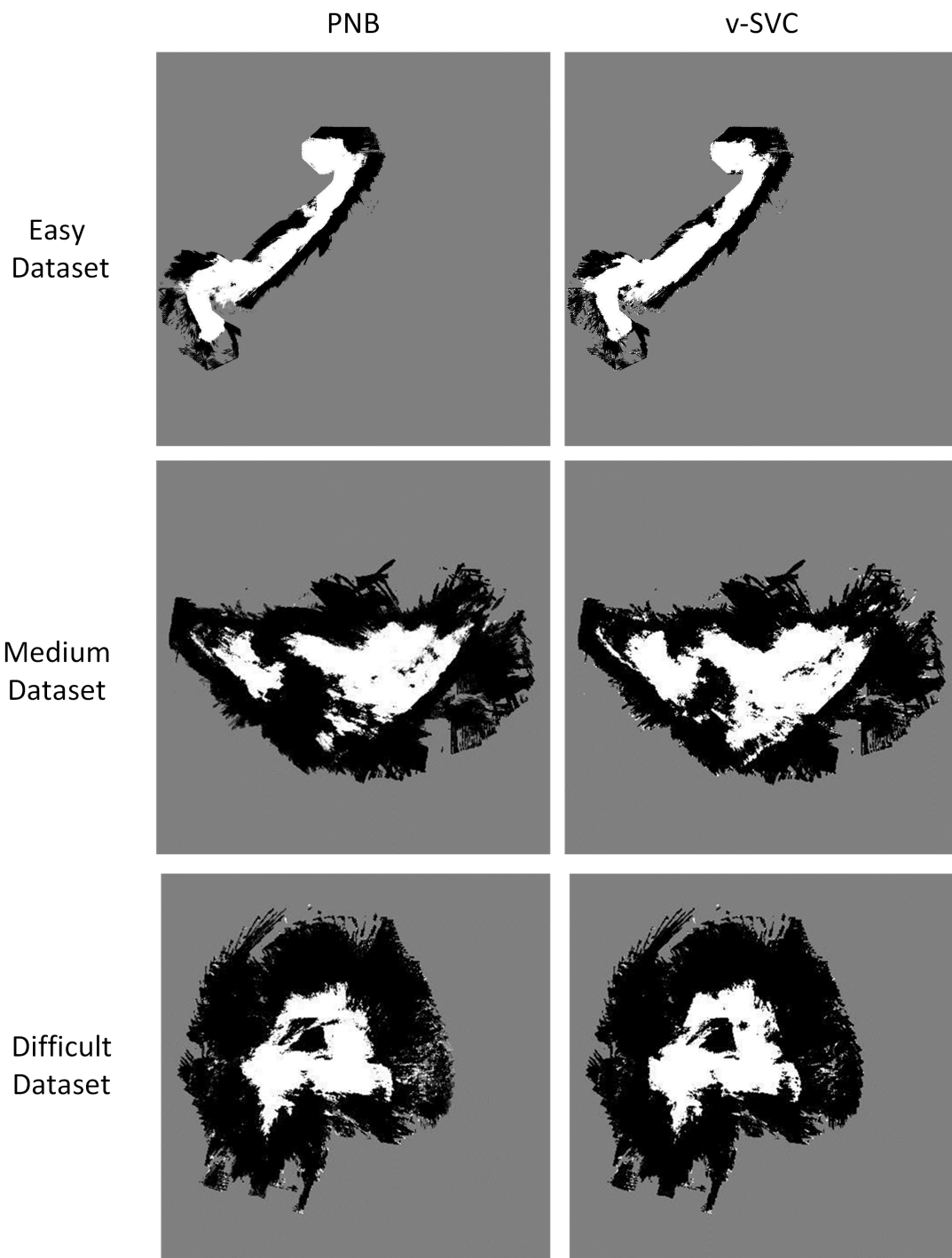


Figure 5.11: The free space mapping results of the PNB and v-SVC for the three datasets. The v-SVC is seen to perform better in free space mapping than the PNB mainly because of its conservative nature.

## Chapter 6

# Conclusion and Future Work

This thesis provides a system that automatically learns, classifies and maps free space in an environment using a stereo sensor. This is done through employing self-supervision by designing an algorithm that extracts training data automatically and reliably from free space in stereo data. The proposed algorithm is shown to be superior to other algorithms in literature by bench-marking on three stereo datasets. As a conclusion, the main contributions of this thesis are :

- A filtering algorithm that increases the reliability of estimating the ground correlation line in the v-disparity space.
- A mathematical proof that measuring the uncertainty in disparity is sufficient to measure the uncertainty in 3D-coordinates of a point in space.
- A Bayesian linear regression frame work to estimate the uncertainty without any prior assumptions on the environment or free variables.
- Using the output PDF from the above framework to extract training data, which are used to supervised two types of classifiers to detect and map free space in an environment.

However, much work remains to be done. The proposed system is memory-less, and throws away precious training data from previous frames. Furthermore, looking into Markov random fields to model inter-pixel dependencies could provide decent boosts in classification performance. Also, switching between the two proposed classifiers would allow the system to benefit from the advantages of both. Finally, it is an interesting idea to perform SLAM using the proposed system to further refine the results of free space mapping.



# Bibliography

- [1] D. Yiruo, W. Wenjia, and K. Yukihiro, “Complex ground plane detection based on v-disparity map in off-road environment,” in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1137–1142.
- [2] B. Suger, B. Steder, and W. Burgard, “Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3d-lidar data,” in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015. [Online]. Available: <http://ais.informatik.uni-freiburg.de/publications/papers/suger15icra.pdf>
- [3] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, “Self-supervised monocular road detection in desert terrain.” in *Robotics: science and systems*. Philadelphia, 2006.
- [4] S. Thrun, M. Montemerlo, and A. Aron, “Probabilistic terrain analysis for high-speed desert driving.” in *Robotics: Science and Systems*, 2006, pp. 16–19.
- [5] A. Milella, G. Reina, J. Underwood, and B. Douillard, “Visual ground segmentation by radar supervision,” *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 696–706, 2014.
- [6] A. Milella, G. Reina, and M. M. Foglia, “A multi-baseline stereo system for scene segmentation in natural environments,” in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–6.
- [7] G. Reina and A. Milella, “Towards autonomous agriculture: automatic ground detection using trinocular stereovision,” *Sensors*, vol. 12, no. 9, pp. 12 405–12 423, 2012.
- [8] P. Vernaza, B. Taskar, and D. D. Lee, “Online, self-supervised terrain classification via discriminatively trained submodular markov random fields,” in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 2750–2757.

- [9] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, “Learning long-range vision for autonomous off-road driving,” *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.
- [10] P. Moghadam and W. S. Wijesoma, “Online, self-supervised vision-based terrain classification in unstructured environments,” in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. IEEE, 2009, pp. 3100–3105.
- [11] D. Kim, S. M. Oh, and J. M. Rehg, “Traversability classification for ugv navigation: A comparison of patch and superpixel representations,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 3166–3173.
- [12] M. Bajracharya, A. Howard, L. H. Matthies, B. Tang, and M. Turmon, “Autonomous off-road navigation with end-to-end learning for the lagr program,” *Journal of Field Robotics*, vol. 26, no. 1, pp. 3–25, 2009.
- [13] C. A. Brooks and K. Iagnemma, “Self-supervised terrain classification for planetary surface exploration rovers,” *Journal of Field Robotics*, vol. 29, no. 3, pp. 445–468, 2012.
- [14] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [15] Z. Hu and K. Uchimura, “Uv-disparity: an efficient algorithm for stereovision based scene analysis,” in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 48–54.
- [16] R. Labayrade, D. Aubert, and J.-P. Tarel, “Real time obstacle detection in stereovision on non flat road geometry through” v-disparity” representation,” in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2. IEEE, 2002, pp. 646–651.
- [17] A. Harakeh, D. Asmar, and E. Shamma, “Ground segmentation and occupancy grid generation using probability fields,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 695–702.
- [18] J. Zhao, J. Katupitiya, and J. Ward, “Global correlation based ground plane estimation using v-disparity image,” in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 529–534.
- [19] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [20] F. Denis, A. Laurent, R. Gilleron, and M. Tommasi, “Text classification and co-training from positive and unlabeled examples,” in *Proceedings of the ICML 2003 workshop: the continuum from labeled to unlabeled data*, 2003, pp. 80–87.

- [21] J. He, Y. Zhang, X. Li, and Y. Wang, “Naive bayes classifier for positive unlabeled learning with uncertainty.” in *SDM*. SIAM, 2010, pp. 361–372.
- [22] B. Schölkopf, R. Williamson, A. Smola, and J. Shawe-Taylor, “Sv estimation of a distributions support,” *Advances in neural information processing systems*, vol. 12, 1999.
- [23] B. Lee, K. Daniilidis, and D. D. Lee, “Online self-supervised monocular visual odometry for ground vehicles,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5232–5238.
- [24] M. J. Procopio, J. Mulligan, and G. Grudic, “Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments,” *Journal of Field Robotics*, vol. 26, no. 2, pp. 145–175, 2009.
- [25] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971–987, 2002.
- [26] M. Häselich, M. Arends, N. Wojke, F. Neuhaus, and D. Paulus, “Probabilistic terrain classification in unstructured environments,” *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1051–1059, 2013.
- [27] S. Labs, <https://www.stereolabs.com/>.