Robotica (2018) volume 36, pp. 1278–1300. © Cambridge University Press 2018
doi:10.1017/S0263574718000371

Self-supervised free space estimation in outdoor terrain

Ali Harakeh, Daniel Asmar* and Elie Shammas

VRL Lab, Department of Mechanical Engineering, American University of Beirut, Riad El-Solh, 1107 2020 Beirut, Lebanon. E-mails: asharakeh@gmail.com, es34@aub.edu.lb

(Accepted April 28, 2018. First published online: June 1, 2018)

SUMMARY
The ability to reliably estimate free space is an essential requirement for efficient and safe robot navigation. This paper presents a novel system, built upon a stochastic framework, which estimates free space quickly from stereo data, using self-supervised learning. The system relies on geometric data in the close range of the robot to train a second-stage appearance-based classifier for long range areas in a scene. Experiments are conducted on board an unmanned ground vehicle, and the results demonstrate the advantages of the proposed technique over other self-supervised systems.

KEYWORDS: Free space estimation; Stereo data; Self-supervised learning; Outdoor terrain.

1. Introduction

Scene understanding and modeling is an essential condition for the success of any unmanned autonomous robot exploration. In its most basic form, this understanding reduces to delineating occupied space from free space and is essential for a system to safely navigate its environment.

The problem of estimating free space in structured and static environments is usually solved by exploiting properties of certain well defined structures, such examples include those of Hedau *et al*.[1] and Labayarde *et al*.[2] While the first exploits the box like geometry of furniture to estimate free space in indoor scenes from a single camera image, the second uses the planar geometry of a road and identifiable lane markings to estimate the free space in urban road scenes. In unstructured or unknown environments such as forest areas, the lack of structure of the scene causes methods relying on static scene properties to fail.

To account for the ever changing properties of free space in unstructured scenes, it is common to resort to learning-based systems, which usually require a training phase in which training data representing free space is used as an input to the learning algorithm. The extraction and classification of training data are usually performed through direct human supervision; unfortunately, this becomes impractical and time consuming as the range of properties to be learned becomes larger. Furthermore, the resulting system cannot extend classification beyond the environments it learned during training, thereby restricting its autonomy.

Recent free space estimation approaches tackle this problem through self-supervision, where one classifier directly supervises input to a second classifier. The first classifier uses data it is confident about to label parts of the environment as free space; this data is then provided as input to the second classifier that extends the labeling over the whole environment. The proposed system in this paper lies within this framework, allowing fully autonomous free space estimation without relying on any rigid assumptions such as a planar ground or bootstrapping methods.

The contribution of this paper is in a novel self-supervised system for the segmentation of outdoor ground terrain of varying morphology, from man-made flat areas to relatively rugged terrain. The system is implemented on board an unmanned ground vehicle (UGV), and we present our results of ground class classification and occupancy grid mapping on three different outdoor environments.

* Corresponding author. E-mail: da20@aub.edu.lb

The remainder of this paper is structured as follows. Section 2 provides a brief summary of previous systems employing self-supervision. Section 3 explains in detail our proposed free space estimation system. Section 4 presents the experiments as well as an in-depth analysis of the results achieved by our proposed system. Section 5 concludes the paper and presents the direction of future work.

2. Related Work

This section presents at first a summary of the state of the art in self-supervised learning algorithms used for free space estimation, and then provides an overview of the v-disparity algorithm used to produce the raw data input to our robust and fast free space estimation, hereafter referred to as RFFSE.

2.1. Self-supervised learning for free space estimation

There are many systems in literature that successfully employed different sensors to estimate free space. This section provides a review on methods that are used for long-term navigation. Systems that use high capacity models on a fixed dataset such as[3] are not within the scope of comparison of this manuscript. Sugar *et al.*[4] used 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.*[5] used a 2-D LIDAR to extract training data belonging to free space using the Probabilistic Terrain Analysis (PTA) algorithm proposed in ref. [6]. 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 those of the training environment. The proposed RFFSE method differs from both methods in that it extracts training pixels belonging to free space 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.*[7] used the echo in a radar image to reliably extract training patches from free space, 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 proposed RFFSE method mitigates 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.*[8] utilize 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 ref. [7]. Reina *et al.*[9] 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. It is worth noting that RFFSE does not need any special initialization and in fact can be launched inside a heavily cluttered scene.

Howard *et al.*[10] introduced two methods for learning to ascribe geometric properties (e.g., occupancy) in distant areas—where stereo information is unreliable—from experience gained in a local area, where geometric and proprioceptive properties are clearly perceived. The system required offline training of a support vector machine (SVM) with a large number of hand-labeled traversable and non-traversable examples. RFFSE does not require offline training and is self-supervised.

Kim *et al.*[11] used 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. RFFSE overcomes this problem by only requiring a single free parameter that determines the reliability versus the number of training points extracted.

Vernaza *et al.*[12] 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.*[13] used 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.*[14] used 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 fail in scenarios where the ground plane is not the largest plane in the image. In contrast, RFFSE 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 class is not planar.

Kostavelis *et al.*[15] proposed a system for traversability estimation based on machine learning. The system uses stereo vision to extract features that train an SVM for classifying traversable and not traversable terrain. The system requires hand-labelling of training data and does not generalize very well to images of scenes that are not included in the training set. Since RFFSE is self-supervised and does not rely on any prior in classifying ground regions, it works well in transitioning between scenes with different ground flatness, without the need for training on a dataset that is comprehensive of the various ground morphologies a robot might encounter outdoors.

Reina *et al.*[16] fused LIDAR and stereo for the assessment of traversable terrain in off-road scenes; their system used two self-learning classifiers, one based on LIDAR and the second on stereo. The disadvantages in their system include the requirement for offline determination of weights using ground truth data, and the requirement for the scene to be free of obstacles during initialization. RFFSE is simpler since it does not require any calibration between LIDAR and stereo camera, and it also does not require any special conditions during initialization.

Proprioceptive sensors such as vibration sensors have also been used to provide labels for free space classification tasks.[17–19] These sensors generate vibration signatures as features and require the robot to have previously driven over a patch to determine if it belongs to the ground class. The vibration-based classifiers usually require manual tuning.[18] RFFSE requires minimal human supervision and does not require manual tuning.

It is noticed that the main weakness in the state of the art lies mainly in the training data extraction methods, where they all impose strong assumptions on the geometry of the ground plane. By using the projection of the ground class onto the v-disparity image coupled with a Bayesian linear regression framework, RFFSE is able to reliably extract training data from the entire image in a robust and quick manner.

2.2. The v-disparity algorithm

The v-disparity algorithm was first proposed by Labayrade *et al.*[2] 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. Figure 1 provides an example of a v-disparity image with its corresponding disparity image and Algorithm 1 lists the construction procedure. For more information on the v-disparity image, we refer the reader to the work of Hu and Uchimura.[20]

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
```

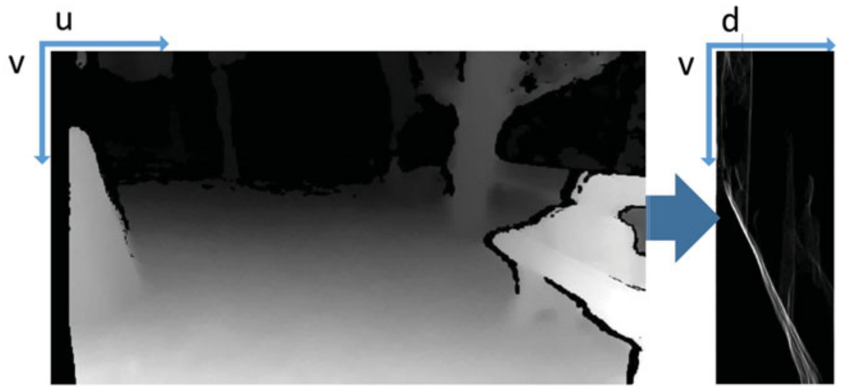

Fig. 1. Left: disparity image. The bright color signifies larger disparity, and hence smaller depth. Right: v-disparity image. The projection of the ground class in the scene is a slanted line, which is visible in the v-disparity image (the image on the right has been manipulated to add contrast and better visualize the vertical lines).

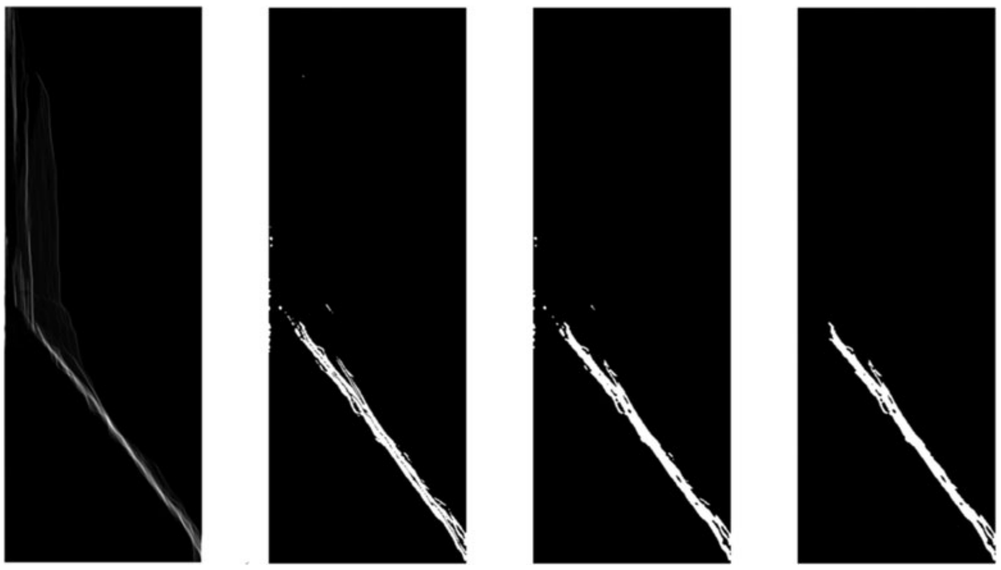

Fig. 2. (a) Original v-disparity image with visible vertical lines representing obstacles. (b) Application of edge detection. (c) Application of a majority black morphological operation. (d) Application of the randomized binary area opening morphological operation. The final filtered version of the v-disparity image after the randomized binary area opening procedure.

3. Proposed RFFSE System

In this section, the details of the proposed RFFSE system are explained, staring from the v-disparity image filtering, to the online learning of the occupancy probabilities, to the second-stage classification.

3.1. v-disparity image filtering

It has been previously proven in ref. [2] that under near-zero roll angle of a camera, horizontal and oblique planes project as slanted lines onto the v-disparity image. This can be seen in Fig. 1, where the ground's projection appears as a prominent slanted line in the v-disparity image. Direct detection of the ground correlation line(1) in the raw v-disparity image is unreliable, especially in cluttered and unstructured scenes.[21–24] We instead use the filtering algorithm proposed in our previous work[21] to get a robust estimate of the ground correlation line (Figure 2 briefly explains the process).

(1)The slanted line projection of the ground class is termed the ground correlation line.[2]

3.2. Online learning of the occupancy probability density function

In scenarios where the ground class is made up of a single plane, the filtered v-disparity image contains a continuous, and well defined ground correlation line, which can be reliably modeled as a first degree polynomial. However, in unstructured or cluttered scenes, the ground correlation line might have a variable width, might contain discontinuities, or might not be a first degree polynomial. To accommodate these cases, Bayesian linear regression is performed by using the v, d pairs in $VD_{filtered}$ as input, to provide as output a learned predictive distribution $P(d/v)$ that describes the probability of a measurement variable d to belong to the ground correlation line (and hence to the ground class), given its row coordinate v. A further comment is warranted here: under extreme camera-roll angles, the projection of a horizontal plane appears not as a line but as a belt centered around that line; the higher the roll of the camera, the larger the width of the belt. We deal with this problem by modeling the projection in the v-disparity space not as a line, but as a second order polynomial, as will be explained below.

3.2.1. 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

$$d = w_0 + w_1 v + w_2 v^2 + \delta = \mathbf{w}^{\mathrm{T}}\phi^*(v) + \delta, \tag{1}$$

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

$$P(d|v, \mathbf{w}, \beta) = \mathcal{N}(d; \mathbf{w}^{\mathrm{T}}\phi^*(v), \beta^{-1}). \tag{2}$$

The vectors $\mathbf{v} = [v_1, \dots, v_N]^{\mathrm{T}}$ and $\mathbf{d} = [d_1, \dots, d_N]^{\mathrm{T}}$ 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 Independent and Identically Distributed (IID) variables drawn from the conditional distribution in (2) and as such, their likelihood function has the expression

$$P(\mathbf{d}|\mathbf{v}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(d_n; \mathbf{w}^{\mathrm{T}}\phi^*(v_n), \beta^{-1}). \tag{3}$$

3.2.2. 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

$$P(\mathbf{w}|\alpha) = P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta) = \mathcal{N}(0, \alpha^{-1}I). \tag{4}$$

The prior is considered to be a zero mean and isotropic Gaussian with a single precision parameter α. This choice of prior reduces the number of its unknown parameters to only α and results in a Gaussian posterior distribution when multiplied with the likelihood function in (3). Having set the prior, the posterior distribution of the parameter vector $\mathbf{w}$ given the training data can be written using Bayes rule as

$$P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \Gamma P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta), \tag{5}$$

where Γ is a normalization coefficient and $P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w})$ is the likelihood function in (3). 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

$$P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \mathcal{N}(\mathbf{w}; \mu_w, \Sigma_w), \tag{6}$$

where μ_w is the mean

$$\mu_w = \beta \Sigma_w \Phi^{\mathrm{T}} \mathbf{d}, \tag{7}$$

and Σ_w is the 3×3 covariance matrix

$$\Sigma_w^{-1} = \alpha I + \beta \Phi^{\mathrm{T}} \Phi. \tag{8}$$

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

$$\Phi = \begin{bmatrix} 1 & v_1 & v_1^2 \\ \cdot & & \\ \cdot & & \\ 1 & v_n & v_n^2 \end{bmatrix}. \tag{9}$$

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

$$P(d|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \int_{\mathbf{w}} P(d|v, \mathbf{v}, \mathbf{d}, \alpha, \beta, \mathbf{w}) P(\mathbf{w}|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) d\mathbf{w}. \tag{10}$$

It is noted that the predictive distribution is the result of a convolution of two Gaussian distributions in (2) and (6). Accordingly, the predictive distribution has the following form:

$$P(d|v, \mathbf{v}, \mathbf{d}, \alpha, \beta) = \mathcal{N}(d; \mu_w^{\mathrm{T}} \phi^*(v), \Sigma_p), \tag{11}$$

where the variance Σ_p can be written as

$$\Sigma_p = \frac{1}{\beta} + \phi^*(v)^{\mathrm{T}} \Sigma_w \phi^*(v). \tag{12}$$

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

As a final thought, the importance of all the assumptions undertaken in this section lies in obtaining a closed-form equation for the mean and variance of the predictive distribution, which can be efficiently computed in with minimal computation time.

3.2.3. 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 α, β, and $\mathbf{w}$. This expansion would have the form

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

which has no closed-form solution due to the lack of knowledge of the conditional joint PDF $P(\alpha, \beta|v, \mathbf{v}, \mathbf{d})$. An approximation of the fully Bayesian treatment of this hierarchical model is computed by setting the hyper-parameters at the highest level of the hierarchy (α and β) to their most likely values instead of integrating them out.[25]

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

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

To estimate the two hyper-parameters, the conditional joint PDF is expanded using Bayes theorem as

$$P(\alpha, \beta|v, \mathbf{v}, \mathbf{d}) \propto P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) P(\alpha, \beta|v, \mathbf{v}). \tag{13}$$

Due to the lack of knowledge of the hyper-parameters α and β 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

$$\begin{aligned}\widehat{\alpha} &= \underset{\alpha}{\operatorname{argmax}}\, P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta),\\ \widehat{\beta} &= \underset{\beta}{\operatorname{argmax}}\, P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta).\end{aligned} \tag{14}$$

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

$$P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta) = \int_w P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta, \mathbf{w})P(\mathbf{w}|v, \mathbf{v}, \alpha, \beta)d\mathbf{w}. \tag{15}$$

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

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

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

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

The derivative equation with respect to α is

$$\frac{\partial \ln P(\mathbf{d}|v, \mathbf{v}, \alpha, \beta)}{\partial\alpha} = \frac{1}{\alpha} - \frac{1}{2}\left[\mu_w\mu_w^{\mathrm{T}} + \frac{\partial \ln|\Sigma_w^{-1}|}{\partial\alpha}\right]. \tag{16}$$

The determinant of the matrix Σ_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:

$$\frac{\partial \ln|\Sigma_w^{-1}|}{\partial\alpha} = \sum_i \frac{1}{\lambda_i + \alpha}. \tag{17}$$

Setting the partial derivative in (16) to zero, the hyper-parameter α will have the form

$$\alpha = \frac{1}{\mu_w\mu_w^{\mathrm{T}}}\sum_i \frac{\lambda_i}{\lambda_i + \alpha}. \tag{18}$$

Similar analysis is done with respect to the hyper-parameter β to obtain

$$\frac{1}{\beta} = \frac{1}{N - \sum_i \frac{\lambda_i}{\lambda_i + \alpha}}\sum_{n=1}^{N}[d_n - \mu_w^{\mathrm{T}}\phi^*(v_n)]^2. \tag{19}$$

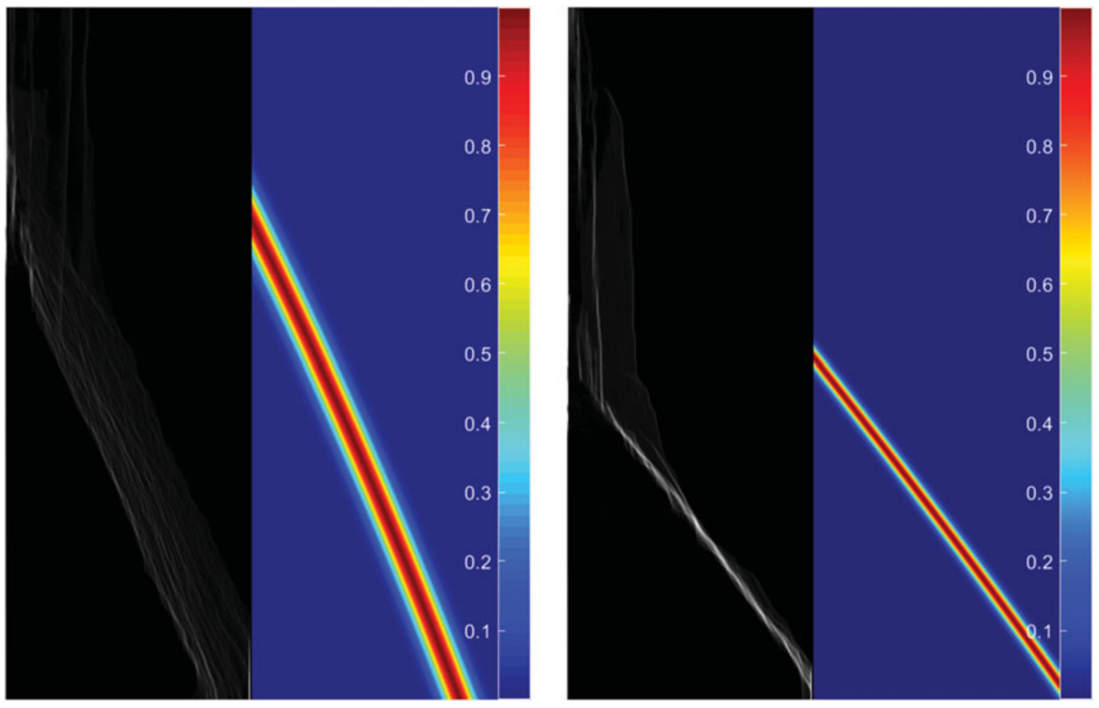

Fig. 3. The original v-disparity image (black and white) and the resulting probability density function in Eq. (20) estimated from Bayesian linear regression (colored). The estimated PDF can be seen to closely resemble the ground correlation line, with an additional probabilistic interpretation. The color of each pixel determines the probability of a point in the v-disparity image to belong to the ground class.

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 μ_w and the sum $\sum_i \frac{\lambda_i}{\lambda_i+\alpha}$ and then compute α and β using (18) and (19), respectively, until convergence, which is determined when the difference between the old and new values 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 hyper-parameter does not affect the end result after convergence.

3.2.4. Using the learned PDF for training pixels extraction. After learning the hyper-parameters $\widehat{\alpha}$ and $\widehat{\beta}$ from (18) and (19), respectively, Σ_p can be computed from (12) resulting in a tractable form of the predictive distribution written as

$$P(d|v, \mathbf{v}, \mathbf{d}, \widehat{\alpha}, \widehat{\beta}) = \mathcal{N}(d; \mu_w^{\mathrm{T}}\phi^*(v), \Sigma_p). \tag{20}$$

A visual representation of the predictive distribution can be seen in Fig. 3. RFFSE uses this predictive distribution (20) to extract training pixels by labeling pixels with a disparity value d belonging to a certain confidence interval as training pixels. This leads to RFFSE having only a single free parameter, the confidence interval, whose effect on the extracted training data will be shown in the following sections.

3.3. The second-stage classifiers

The Bayesian linear regression framework above provides incomplete pixel labels belonging to the positive ground class that contain no negative obstacle samples. It separates N image pixels into two subsets: TP denoting pixels with a label $l_{ground} = 1$, and UP denoting unlabeled pixels with $l = \emptyset$. Furthermore, each pixel in both subsets is assigned an M dimensional feature vector $\bar{f} = (f_1, \ldots, f_m)^{\mathrm{T}}$. The second-stage classification task is thus defined as providing a label $l \in \{ground, obstacle\} = \{1, 0\}$ for each pixel in UP. In what follows, two second-stage classification schemes are proposed to perform this task: the positive naive Bayes (PNB) classifier, and the ν-support vector classifier.

3.3.1. The positive naive Bayes classifier. The PNB classifier as defined by Denis *et al.*[26] 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

$$l = \underset{l\in\{0,1\}}{\operatorname{argmax}}\, P(l|f_1, f_2, \ldots, f_m), \tag{21}$$

$$= \underset{l\in\{0,1\}}{\operatorname{argmax}}\, \eta P(l)P(f_1, f_2, \ldots, f_m|l), \tag{22}$$

where η is a normalization coefficient, and where (22) is derived from (21) through the Bayes rule. Assuming conditional independence of each component of the feature vector, (22) reduces to

$$l = \underset{l\in\{0,1\}}{\operatorname{argmax}}\, \eta P(l) \prod_{i=1}^{M} P(f_i/l). \tag{23}$$

For now, it is assumed that each component of the feature vector lies in a strictly positive discrete feature space such that $f_i \in [0, 2, \ldots, K] \forall i \in [1, M]$, creating a vocabulary V of discrete features. Furthermore, the features are assumed to have multinomial distribution given the class label such that $P(f_i/l) \sim$ 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

$$C(f_i, S) = \sum_{j=1}^{N} \mathbb{1}\{f_i = f_j \wedge l = l_S\}, \tag{24}$$

$$C(S) = \sum_{j=1}^{N} \mathbb{1}\{l = l_S\}, \tag{25}$$

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

$$P(f_i|l = 1) = \frac{\zeta_p + C(f_i, TP)}{\zeta_p Card(V) + C(TP)}, \tag{26}$$

where $Card(V)$ is the cardinality of the vocabulary V, and ζ_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 such that

$$P(f_i) = P(f_i|l = 0)P(l = 0) + P(f_i|l = 1)P(l = 1). \tag{27}$$

The negative probability is then written as

$$P(f_i|l = 0) = \frac{P(f_i) - P(f_i|l = 1)P(l = 1)}{P(l = 0)}. \tag{28}$$

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

$$P(f_i) = \frac{C(f_i, UD)}{C(UD)}. \tag{29}$$

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

$$P(f_i/l=0)=\frac{1+\max(0,C(f_i,UD)-P(l=1)P(f_i/l=1)C(UD))}{Card(V)+(1-P(l=1))C(UD)}, \tag{30}$$

where the max function was used to insure a non-negative probability,[27] 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, (26) and (30) are substituted in (23), and labels can be generated for each pixel in the scene.

3.3.2. The ν-support vector classifier. The ν-SVC was proposed by Scholkopf *et al.*[28] in 1999 and became a popular kernel-based learning algorithm for one class classification problems. The ν-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:

$$\begin{aligned}&\min_{\alpha}\frac{1}{2}\sum_{jk}\alpha_j\alpha_k k(\bar{f}_j,\bar{f}_k)\text{ subject to :}\\ &0\le\alpha_j\le\frac{1}{C(TP)},\\ &\sum_j\alpha_j=\nu,\end{aligned} \tag{31}$$

where $\bar{f}_1,\ldots,\bar{f}_{C(TP)}$ 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

$$l=\max(0,\operatorname{sgn}(\sum_j\alpha_j k(\bar{f}_j,\bar{f})-\rho)), \tag{32}$$

$$\rho=\sum_k\alpha_k k(\bar{f}_j,\bar{f}_k),\ \text{ for any } 0\le\alpha_k\le\frac{1}{C(TP)}. \tag{33}$$

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

$$k(\bar{f}_j,\bar{f}_k)=\exp(\frac{-(\bar{f}_j-\bar{f}_k)^2}{2s^2}), \tag{34}$$

where s is the scale parameter. We refer the reader to the work of ref. [28] for a detailed analysis of the effect of the ν and s on the classification results of the ν-SVC.

3.3.3. Feature selection. The selection of discriminant features to be used by the second-stage classifiers is essential for good classification results. However, if one strives for 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 in this paper required to collect training data at each frame and in a quick manner; accordingly, the sought after features need not be temporally invariant, but instead, we opt to use features that are computationally efficient. We select the raw data features available directly from stereo image data, including the mean height, and the mean of R, G, and B channels. Furthermore, the height variance and the maximum absolute difference in height were added, as they describe geometric textural roughness, and also require little computational time. The PNB classifier requires the feature space to be discrete. Color features chosen above are already discrete, taking values between 0 and

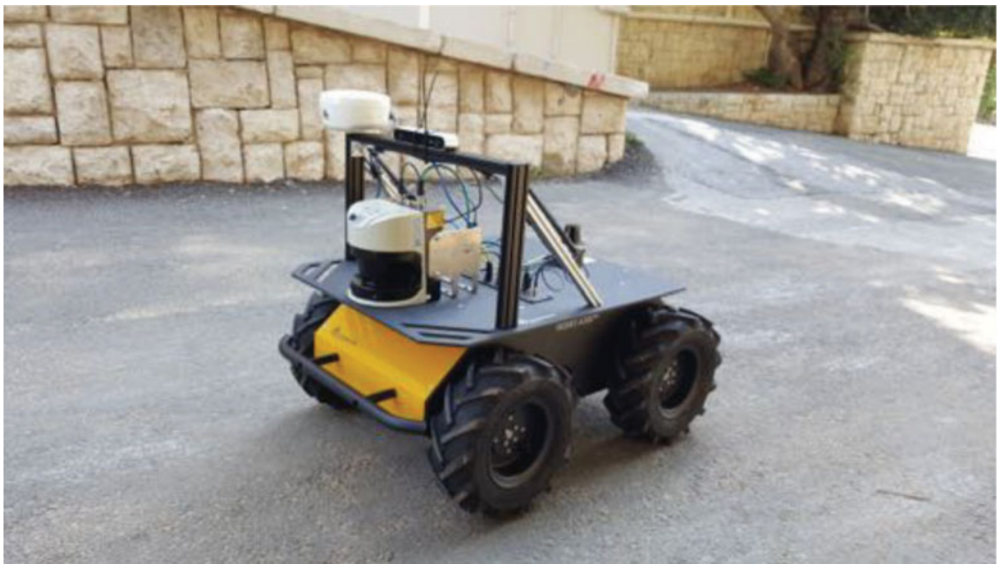

Fig. 4. Deployed sensors and robot. The Zed Stereo Camera is rigidly mounted on top of the Clearpath Husky UGV.

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:

$$f_{new} = 255 \times \frac{f_{old} - \min(f_{old})}{\max(f_{old}) - min(f_{old})}, \tag{35}$$

producing feature values between 0 and 255 for all components of the feature vector. As a final note, the discretization method is a linear transformation, and as such the characteristics of the feature distributions are preserved.

4. Experiments and Results

This section presents the experimental setup used for data acquisition and describes the datasets acquired. It then provides an analysis of the robustness of the representation of feature distributions of the ground class by the extracted training data. Finally, the classification and environment mapping results of the proposed second-stage classifiers using RFFSE for training data extraction are presented and analyzed.

4.1. Hardware

As shown in Fig. 4, Stereo Lab's Zed Camera[29] is used to aquire 720×1280 RGB images as well as 3-D 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® i7™ processor at 3 GHz with 32 GB RAM.

4.2. Datasets

To be able to perform the necessary experiments, three datasets[30] were created with terrains ranging from planar to non-planar ground. Each frame in the dataset is comprised of a stereo pair of 720 × 1280 colored images, their corresponding disparity image, and pixel X, Y, and Z coordinates 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 three datasets include the following:

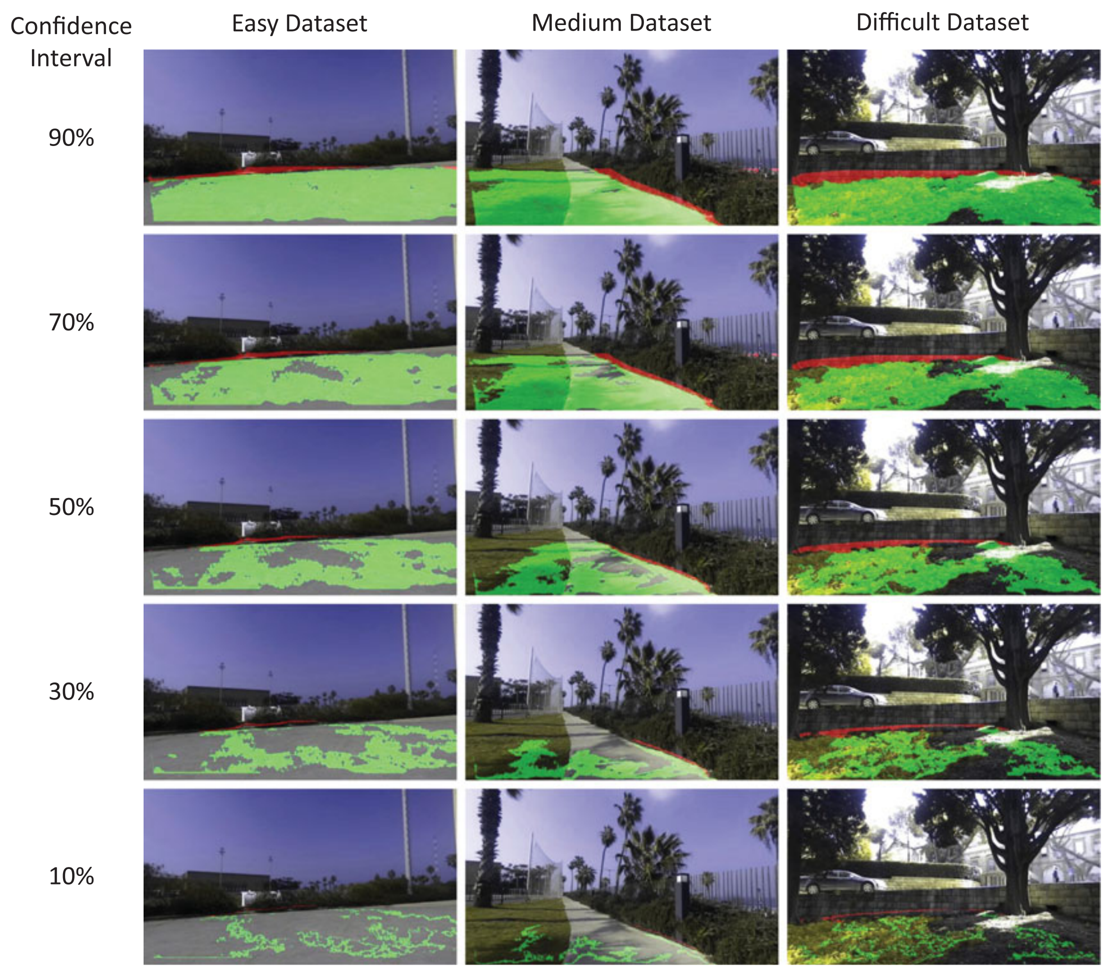

Fig. 5. The effect of varying the confidence interval on the amount and quality of training pixels extracted in the image.

Difficult dataset: A total of 88 images were taken with a stereo camera mounted on a UGV driven on non-planar off-road terrain.

Moderate dataset: A total of 120 images were taken with a stereo camera mounted on a UGV driven on a moderately non-planar park-like terrain.

Easy dataset: A total of 145 images were taken with a stereo camera mounted on a UGV driven on a 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, pixel level ground truth labels are generated manually for every frame of the three datasets.

4.3. The effect of the confidence interval on extracted training pixels

The predictive distribution (20) is used to extract training pixels by labeling pixels with a disparity value d belonging to a given confidence interval as training pixels. Figure 5 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 subsequent lower row, reaching 10% confidence interval for the bottom row. It can be noted that as the width of the confidence interval decreases, the number of correctly labeled training pixels (green) decreases, but at the same time the number of outliers (red) also decreases. In other words, fewer but more accurate training pixels are obtained by tightening the confidence interval.

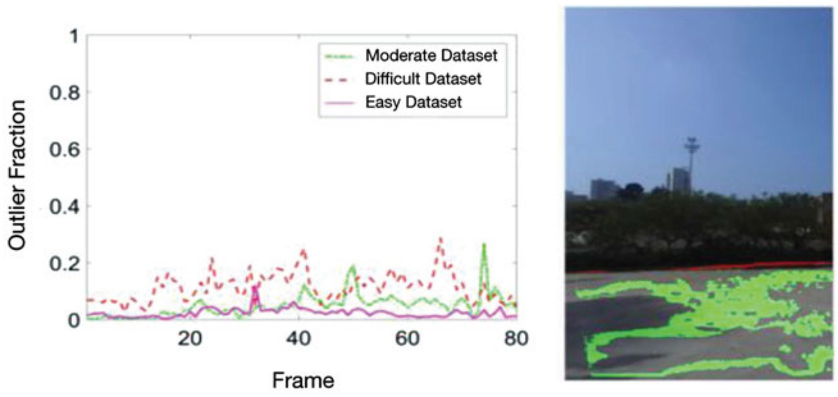

Fig. 6. Left: the fraction outliers computed by running the proposed algorithm over the three datasets. Right: An example of the presence of outliers in the extracted training pixels. Correctly labeled training pixels are shown in green, whereas outliers are shown in red.

4.4. Assessing the quality of extracted training pixels

Although it is established that the proposed algorithm can indeed provide training pixels, the quality of the extracted training pixels remains an unanswered question. To begin with the analysis, goodness criteria need 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 the confidence interval is set to 30% for all the tests performed. This value is seen to provide a suitable compromise between the number of training data and the outlier fraction for the three datasets the system was tested on.

The first goodness criterion to be analyzed is 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. The left sub-figure of Fig. 6 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 labeled data as the environment becomes harsher. On the other hand, the mean outlier fraction produced by the algorithm is 0.0268, 0.0452 and 0.1098 for the easy, moderate and difficult datasets, respectively, which is tolerable and can be handled through second-stage classification. The outlier fraction is highly correlated with the quality of the disparity images, and tends to increase as the quality of the disparity images deteriorates. This is a natural outcome of the dependence of RFFSE on the disparity space for training data extraction and it is anticipated that RFFSE should 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 six features in the selected feature space is performed. The features are assumed to be normally distributed and their mean and standard deviation are compared. Table I presents a comparison between the sufficient statistics of the true versus estimated distributions taken from a randomly selected frame from each dataset. It can be seen that the estimated distribution's 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 all the pixels that belong to ground class, found from hand-labeled pixel level ground truth. The estimated distributions' plots is superposed over that of true distributions' plots in Figs. 7–9 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 RFFSE 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 I. 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.

Easy dataset

Statistic	Red	Blue	Green	Mean Y	Y var.	Y diff.
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

Moderate dataset

Statistic	Red	Blue	Green	Mean Y	Y var.	Y diff.
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 Y	Y var.	Y diff.
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

4.5. 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%.

4.5.1. Free space classification. The labels obtained from the two classifiers are compared to ground truth labels in order to compute three performance criteria, which are the recall, precision, and specificity. Examples of classification labels are presented in Fig. 12 for the PNB classifier and in Fig. 13 for the ν-SVC classifier. It has to be noted that the PNB classifier classifies each pixel in the 720×1280 images, whereas the ν-SVC classifies 5×32 image blocks. In the easy dataset, the PNB classifier performs better than the ν-SVC classifier in terms of recall, achieving a mean recall value of 0.9131, while the mean precision is 0.9823 and mean specificity is 0.9936 (see Fig. 10).

The ν-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. The better performance of the PNB classifier in terms of recall is attributed 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 can be seen in Fig. 7.

When applied on the moderate dataset on the other hand, the ν-SVC classifier seems to perform better with a mean recall value of 0.8549 versus a mean recall of 0.8326 for the PNB classifier. The ν-SVC's precision is 0.9416, about 1% lower than the PNB classifier, while the specificity is approximately the same. This is expected as precision and recall are antithetic, as one increases, the other decreases. The better performance of the ν-SVC is primarily due to the complex nature of the scene. The ground class color distribution as it can be seen in Fig. 8 is multimodal and not linearly separable, and thus it is expected that the ν-SVC classifier performs better than the PNB classifier in such scenarios.

Finally, the performance of the ν-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

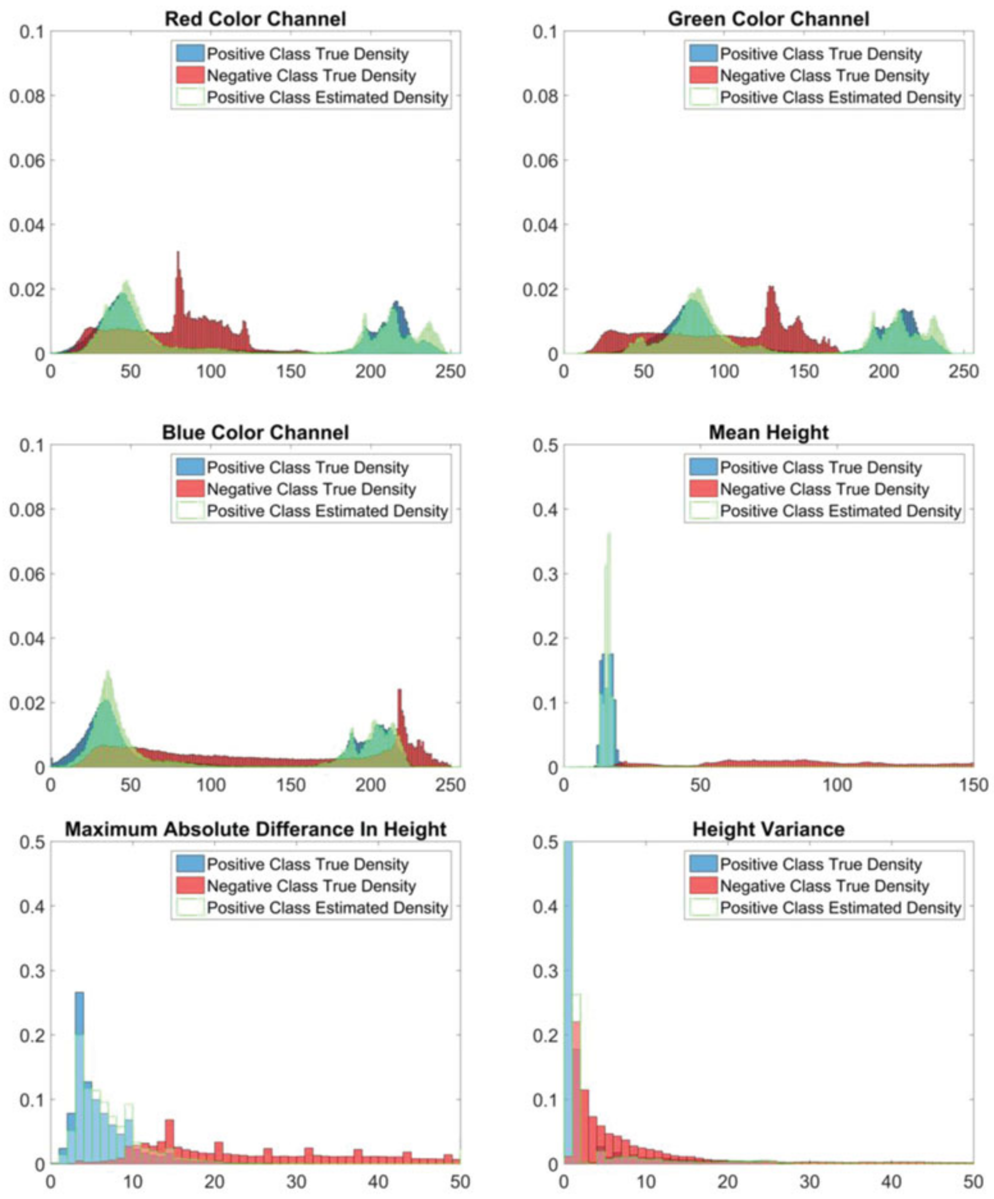

Fig. 7. Easy dataset. The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the easy dataset. The estimated probability distribution for the positive class (green) is found using training pixels extracted via our proposed algorithm. The features are all linearly separable and unimodal, thus approximated well by a normal distribution.

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. 9. As a final conclusion, both classifiers provide suitable free space classification results, with the ν-SVC performing slightly better than the PNB classifier on difficult terrain.

4.5.2. 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 1000×1000 cell grid with each cell representing a 10 cm $\times$ 10 cm world patch.

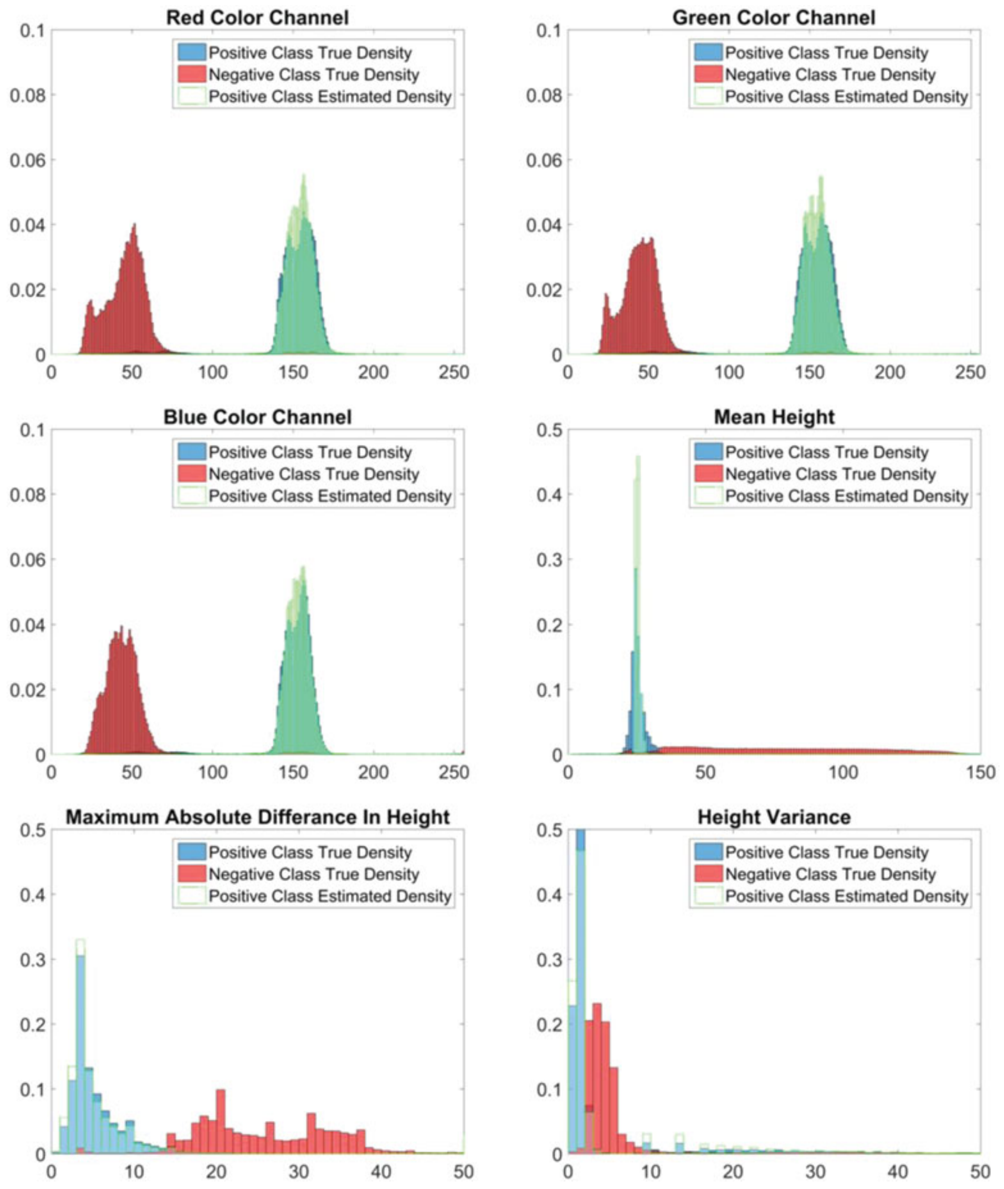

Fig. 8. Moderate dataset. The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the moderate dataset. The estimated probability distribution for the positive class (green) is found using training pixels extracted via RFSFE. The color 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.

Initially, the occupancy probability of each cell is set to 0.5. The occupancy probability is then updated as the UGV explores the environment as follows:

$$O_t = O_{t-1} + O_{sensor}, \tag{36}$$

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

$$O_{sensor} = \log(\eta P(l = 1) \prod_{i=1}^{M} P(f_i/l = 1)) - \log(\eta P(l = 0) \prod_{i=1}^{M} P(f_i/l = 0)), \tag{37}$$

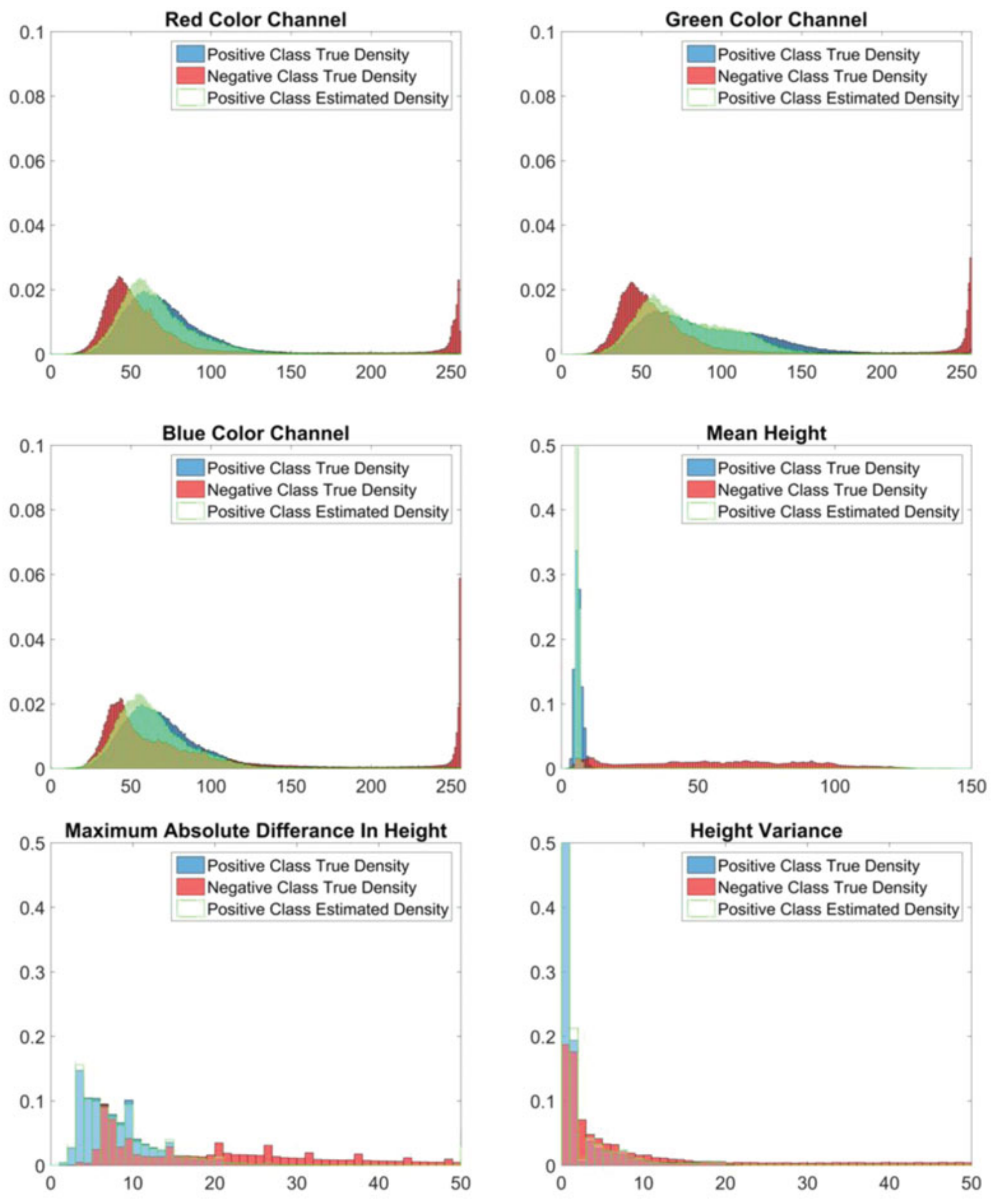

Fig. 9. Difficult dataset. The true probability distributions of the features for the positive (blue) and negative (red) class in a random frame from the difficult dataset. The estimated probability for the positive class (green) is found using training pixels extracted via RFSFE. The color features' 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.

where η is a normalization constant, f_i are features in the feature vector, and l is the label $\in [0, 1]$. This equation suggests that the occupancy probability for each voxel in the local occupancy grid is computed as the difference between the two class posterior probabilities.

Modifying the result of the ν-SVC is more complex. At first, the raw score inside the sgn function in (33) 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

$$O_{sensor} = \frac{2}{1 + \exp(-\sum_j \alpha_j k(\bar{f}_j, \bar{f}) + \rho)} - 1. \tag{38}$$

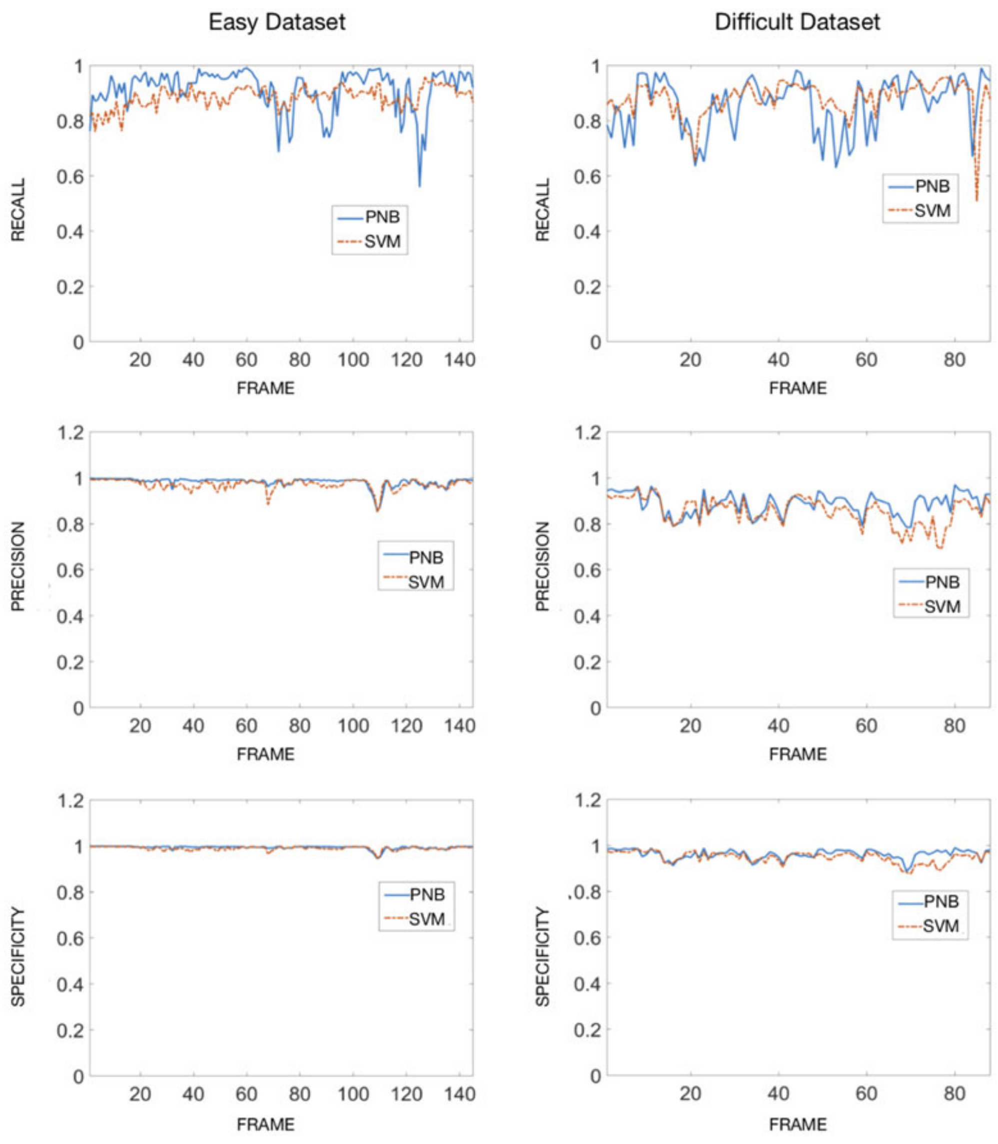

Fig. 10. The free space mapping results of the PNB and ν-SVC for the easy and difficult datasets. The ν-SVC is seen to perform better in free space mapping than the PNB mainly because of its conservative nature.

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. The performance of the mapping done by 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 2-D planes. The mapping procedure follows the description in Section 4.5.2. Points with the same X, Y are handled by addition of their log odds values.

Figure 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 ν-SVC outperforms the PNB classifier on the moderate and difficult datasets. It can be seen that in the moderate 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 ν-SVC classifier is much more conservative than the PNB classifier. This is because the probabilistic output of the ν-SVC (Fig. 13)

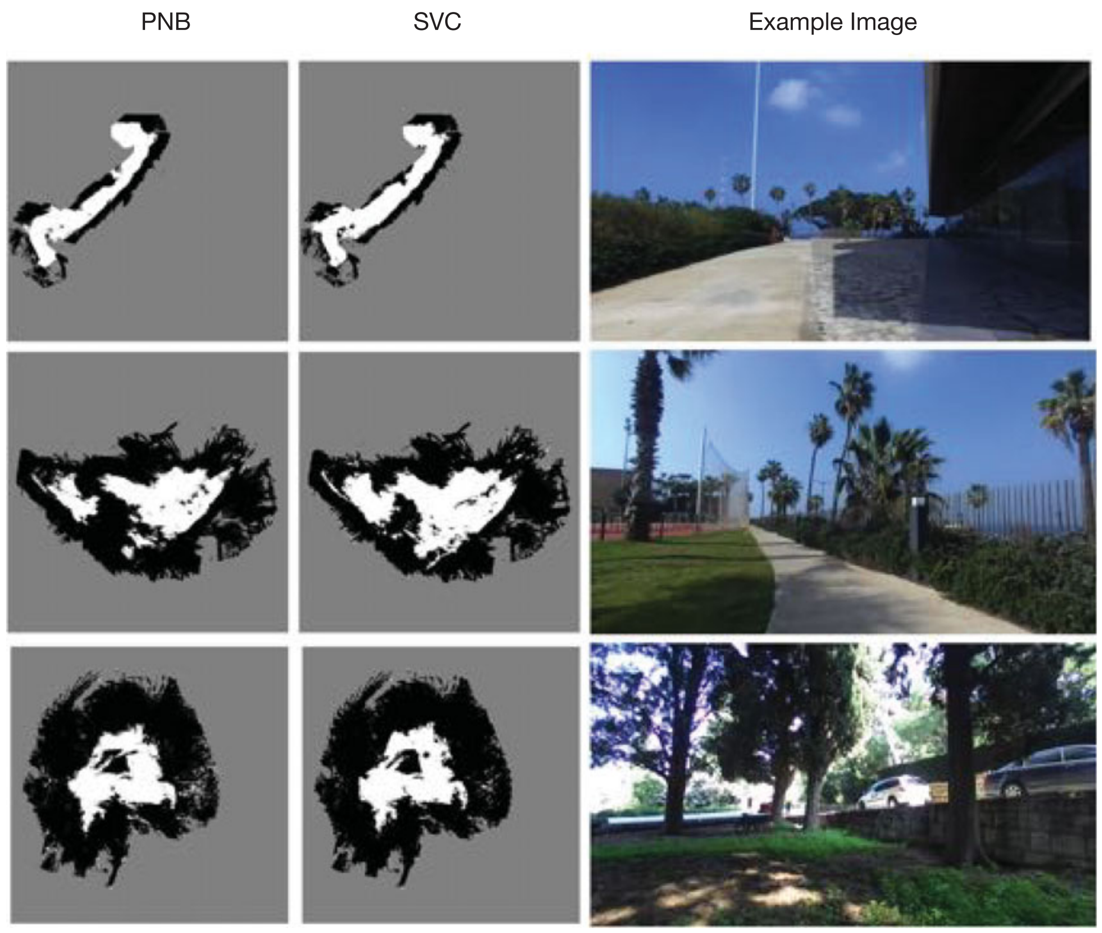

Fig. 11. The free space mapping results of the PNB and ν-SVC for the three datasets (easy at top, moderate at middle, and difficult at bottom). The ν-SVC is seen to perform better in free space mapping than the PNB mainly because of its conservative nature.

is very close to either 0 or 1, while that of the PNB classifier (Fig. 12) 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. Figure 14 shows the results of the computation time in seconds for both classifiers (here, we only show the results of the easy and difficult datasets). It can be clearly seen that the PNB classifier requires less computation time that the ν-SVC classifier. The PNB classifier requires 0.46, 0.5, 0.58 seconds per frame from the easy, moderate and difficult datasets, respectively, versus 0.56, 0.57, 0.6 seconds per frame for the ν-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 ν-SVC classifier, and provides continuous probability values for each pixel. On the other hand, the ν-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 ν-SVC classifier in difficult non-planar environments.

5. Conclusion and Future Work

This paper proposed a fast and robust system that classifies and maps free space in outdoor environment, in which the ground features different levels of flatness. Results show that training

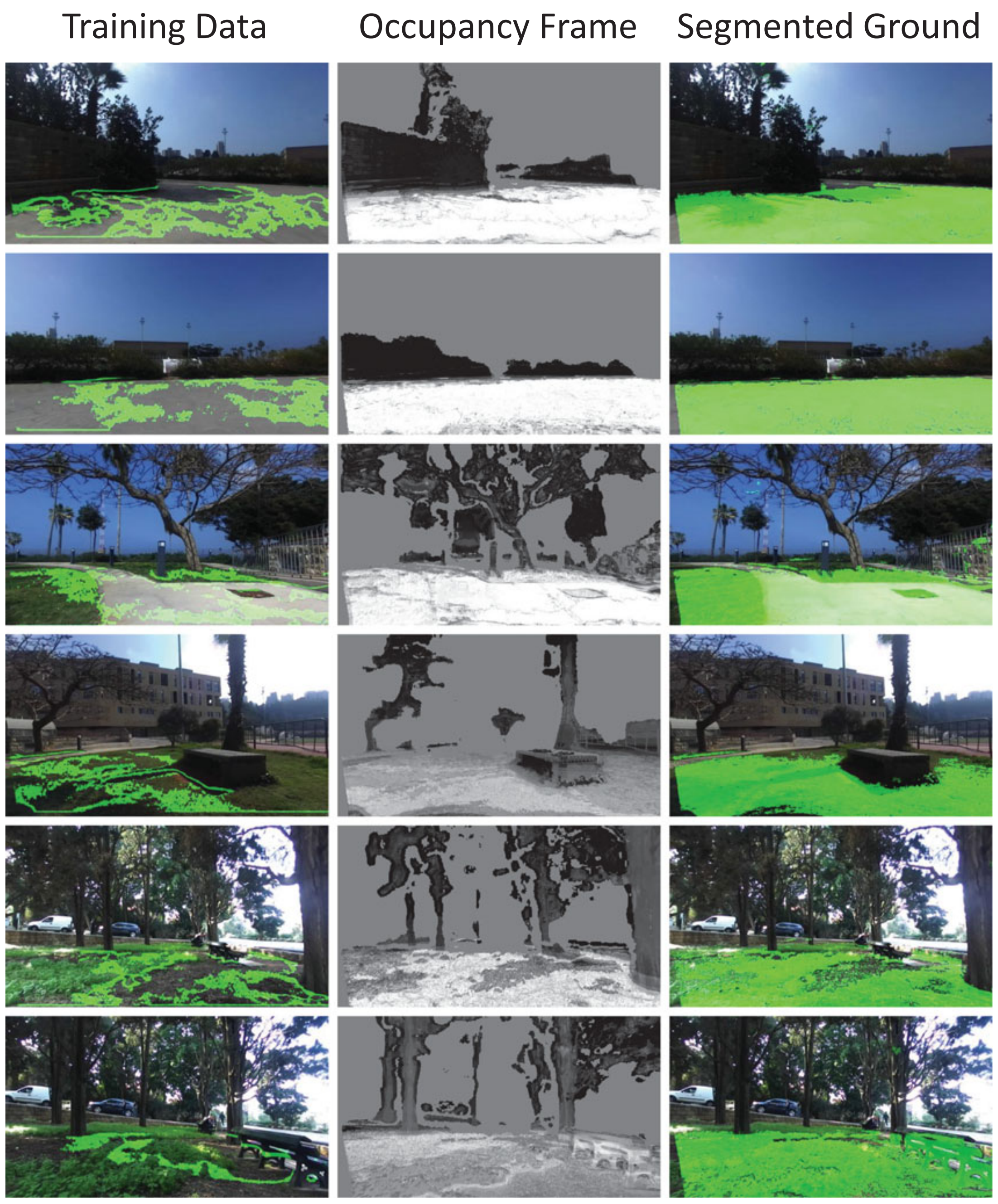

Fig. 12. PNB classifier results: training data extraction (first column), the occupancy grid result (second column), and the final ground segmentation. The first two rows are from the easy dataset, the second two from the moderate dataset, and the final two from the difficult dataset.

data extracted through RFFSE contain a small fraction of outliers and correctly models the true feature distribution of the ground class. Furthermore, RFFSE suggests two different classifiers, each suitable for different environments; while PNB is more suited to man-made structured environments because of its speed, ν-SVC works better in unstructured outdoor scenes.

For future work, we are exploring a sequential model learning system, as the current proposed second-stage classifiers are memory-less, and throw 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 would be an interesting

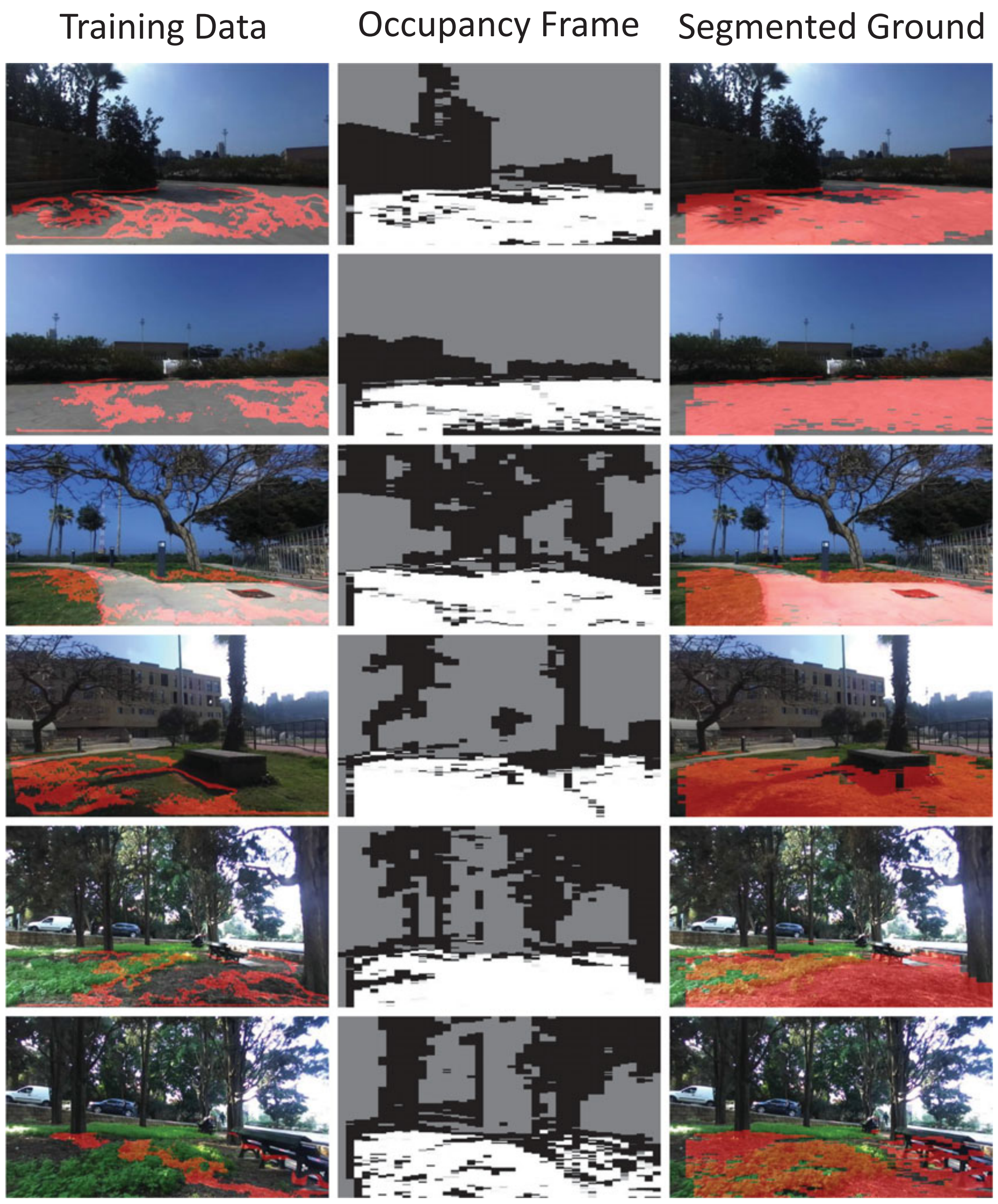

Fig. 13. ν-SVC classifier results: the training data extraction (first column), the occupancy grid results (second column), and the final ground segmentation. The first two rows are from the easy dataset, the second two from the moderate dataset, and the final two from the difficult dataset.

idea to include our ground plane estimation results within a semantic SLAM framework; knowing the location of ground could assist in segmenting and recognizing outdoor objects such as trees and lampposts, and effectively lead to an improved semantic SLAM solution.

Acknowledgments
This work was supported by the University Research Board (URB) at the American University of Beirut, and by the Lebanese National Council for Scientific Research (LNCSR).

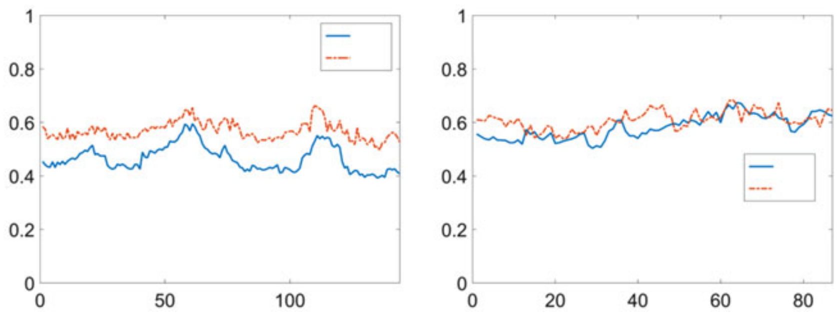

Fig. 14. Computation time for the PNB and ν-SVC for the easy and difficult datasets.

References

1. V. Hedau, D. Hoiem and D. Forsyth, “Recovering Free Space of Indoor Scenes from a Single Image,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2012) pp. 2807–2814.
2. R. Labayrade, D. Aubert and J. P. Tarel, “Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through “v-Disparity” Representation,” *Proceedings of the IEEE Intelligent Vehicle Symposium*, vol. 2, (2002) pp. 646–651.
3. D. Maturana, P. W. Chou, M. Uenoyama and S. Scherer, “Real-time semantic mapping for autonomous off-road navigation,” **In**: *Field and Service Robotics*, Springer, Cham. (2018) pp. 335–350.
4. 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**: *Robotics and Automation (ICRA)*, 2015 IEEE International Conference on (IEEE, May 2015) pp. 3941–3946.
5. H. Dahlkamp *et al.*, “Self-Supervised Monocular Road Detection in Desert Terrain,” *Proceedings of Robotics: Science and Systems*, Philadelphia.
6. S. Thrun, M. Montemerlo and A. Aron, “Probabilistic Terrain Analysis for High-Speed Desert Driving,” *Proceedings of Robotics: Science and Systems*, pp. 16–19.
7. A. Milella *et al.*, “Visual ground segmentation by radar supervision,” *Robot. Auton. Syst.* **62**(5), 696–706 (2014).
8. A. Milella, G. Reina and M. M. Foglia, “A Multi-Baseline Stereo System for Scene Segmentation in Natural Environments,” *Proceedings of the IEEE International Conference Technologies for Practical Robot Applications (TePRA)*, (2013) pp. 1–6.
9. G. Reina and A. Milella, “Towards autonomous agriculture: Automatic ground detection using trinocular stereovision,” *Sensors* **12**(9), 12405–12423 (2012).
10. A. Howard *et al.*, “Towards learned traversability for robot navigation: From underfoot to the far field,” *J. Field Robot.* **23**(11–12), 1005–1017 (2006).
11. D. Kim, S. M. Oh and J. M. Rehg, “Traversability Classification for UGV Navigation: A Comparison of Patch and Superpixel Representations,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* (2007) pp. 3166–3173.
12. P. Vernaza, B. Taskar and D. D. Lee, “Online, Self-Supervised Terrain Classification Via Discriminatively Trained Submodular Markov Random Fields,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (2008) pp. 2750–2757.
13. R. Hadsell *et al.*, “Learning long-range vision for autonomous off-road driving,” *J. Field Robot.* **26**(2), 120–144 (2009).
14. P. Moghadam and W. S. Wijesoma, “Online, Self-Supervised Vision-Based Terrain Classification in Unstructured Environments,” *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, (2009) pp. 3100–3105.
15. I. Kostavelis, L. Nalpantidis and A. Gasteratos, “Collision risk assessment for autonomous robots by offline traversability learning,” *Robot. Auton. Syst.* **60**(11), 1367–1376 (2012).
16. G. Reina, A. Milella and R. Worst, “Lidar and stereo combination for traversability assessment of off-road robotic vehicles,” *Robotica* **34**(12), 2823–2841 (2016).
17. M. Bajracharya *et al.*, “Autonomous off-road navigation with end-to-end learning for the LAGR program,” *J. Field Robot.* **26**(1), 3–25 (2009).
18. C. A. Brooks and K. Iagnemma, “Self-supervised terrain classification for planetary surface exploration rovers,” *J. Field Robot.* **29**(3), 445–468 (2012).
19. K. M. Wurm *et al.*, “Identifying vegetation from laser data in structured outdoor environments,” *Robot. Auton. Syst.* **62**(5), 675–684 (2014).
20. Z. Hu and K. Uchimura, “Uv-Disparity: An Efficient Algorithm for Stereovision Based Scene Analysis,” *Proceedings of the IEEE Intelligent Vehicles Symposium,* (2005) pp. 48–54.

21. A. Harakeh, D. Asmar and E. Shammas, "Ground Segmentation and Occupancy Grid Generation Using Probability Fields," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (2015) pp. 695–702.
22. A. Harakeh, D. Asmar and E. Shammas, "Identifying good training data for self-supervised free space estimation." **In**: *Computer Vision and Pattern Recognition (CVPR)*, 2016 IEEE Conference on (IEEE, June 2016) pp. 3530–3538.
23. D. Yiruo, W. Wenjia and K. Yukihiro, "Complex Ground Plane Detection Based on v-Disparity Map in Off-Road Environment," *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, (2013) pp. 1137–1142.
24. J. Zhao, J. Katupitiya and J. Ward, "Global Correlation Based Ground Plane Estimation Using v-Disparity Image," *Proceedings of the IEEE International Conference on Robotics and Automation*, (2007) pp. 529–534.
25. C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).
26. F. Denis *et al.*, "Text Classification and Co-Training from Positive and Unlabeled Examples," *Proceedings of the ICML Workshop: The Continuum from Labeled to Unlabeled Data* (2003) pp. 80–87.
27. J. He, Y. Zhang, X. Li and Y. Wang, "Naive bayes classifier for positive unlabeled learning with uncertainty," **In**: *Proceedings of the 2010 SIAM International Conference on Data Mining* (Society for Industrial and Applied Mathematics, April 2010) pp. 361–372.
28. B. Scholkopf, R. C. Williamson, A. J. Smola and J. Shawe-Taylor, "SV Estimation of a Distribution's Support," *Proc. 14th Neural Information Processing Systems (NIPS '00)*, (2000) pp. 582–588.
29. Stereo Labs., Zed camera. Available at: `https://www.stereolabs.com/` (2017).
30. AUB stereo dataset. Available at: https://www.dropbox.com/s/qpxyvjdk0yiym9p/aubstereodataset.rar?dl=0 (2016).