Identifying Good Training Data for Self-Supervised Free Space Estimation

Ali Harakeh, Daniel Asmar, Elie Shammas
Department of Mechanical Engineering, VRL Lab
American University of Beirut, Beirut, Lebanon
ash18@mail.aub.edu, da20@aub.edu.lb, es34@aub.edu.lb

Abstract

This paper proposes a novel technique to extract training data from free space in a scene using a stereo camera. The proposed technique exploits the projection of planes in the v-disparity image paired with Bayesian linear regression to reliably identify training image pixels belonging to free space in a scene. Unlike other methods in the literature, the algorithm does not require any prior training, has only one free parameter, and is shown to provide consistent results over a variety of terrains without the need for any manual tuning. The proposed method is compared to two other data extraction methods from the literature. Results of Support Vector classifiers using training data extracted by the proposed technique are superior in terms of quality and consistency of free space estimation. Furthermore, the computation time required by the proposed technique is shown to be smaller and more consistent than that of other training data extraction methods.

1. Introduction

Scene understanding and modeling is a vital condition for the success of any unmanned autonomous system 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. Furthermore, knowledge of freespace could provide significant insight in difficult scene understanding problems [1].

The problem of estimating free space in structured and static environments is usually solved by exploiting properties of certain well defined structures. Two examples of free space estimation solutions are that of Hedau *et al*. [2] and Labayrade *et al*. [3]. 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 natural 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 is 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 long range fully autonomous free space estimation without relying on any rigid assumptions such as a planar ground or bootstrapping methods. The main contribution in this paper is a novel training data extraction method that is fast, accurate, and can be applied in structured and unstructured environments.

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 details the training data extraction algorithm proposed in the paper. Section 4 briefly presents the second classification stage based on a one-class Support Vector Classifier (ν-SVC), which is a necessary tool to assess the goodness of the training data extraction algorithms. Section 5 presents the results and analysis of the ν-SVC using our proposed method for training data extraction versus two other training data extraction methods. Finally, Section 6 concludes the paper.

2. Related Work

The first part of this section presents a sample of previous work on free space estimation with an emphasis on learning-based methods. The second part gives a brief overview of the v-disparity algorithm and v-disparity image filtering, which is used as a first stage in the proposed training data extraction algorithm.

Self-Supervised Learning For Free Space Estimation: a variety of sensors and sensor combinations have previously been employed for free space estimation. Sugar *et al*. [4] 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*. [5] used a 2-D LIDAR to extract training data belonging to free space using the Probabilistic Terrain Analysis (PTA) algorithm proposed in [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 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*. [7] 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 posses 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*. [8] utilizes a stereo camera to extract geometric features that are used to classify voxels in a 3-D point cloud belonging to free space. In order to create the ground model, the system needs to be initialized in an area free of obstacles. The requirement for initialization is problematic when the system fails and the human operator cannot intervene to reinitialize it. Our system does not need any special initialization and in fact can be launched inside a heavily cluttered scene.

Vernaza *et al*. [9] also 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. This training 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 paper 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.

The V-Disparity Algorithm: the v-disparity algorithm was first proposed by Labayrade *et al*. [3] for road 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 them vertically. For example, a 720x1280 disparity image is transformed into a 720x256 v-disparity image. Oblique and horizontal planes in the disparity image are mapped to slanted lines in the v-disparity image. Thus, detecting slanted lines in the v-disparity image is equivalent to the estimation of the ground plane in the disparity image. Fig.1-c shows the constructed v-disparity image, where the slanted line projection of the ground plane termed the ground correlation line can be observed.

The V-Disparity Image Filtering and Stochastic Model: Harakeh *et al*. [10] noted that traditional line detection methods were unreliable when trying to estimate slanted lines in off-road scenes, and proposed a binary filtering algorithm that takes as an input the v-disparity image and provides as an output a filtered v-disparity image containing only slanted lines. This was coupled with a stochastic model, which uses maximum likelihood linear regression with a first order polynomial basis functions to provide point estimates of the parameters of the ground correlation line. The parameters were then used to estimate the mean of a probability distribution that describes the occupancy probability of each pixel. This approach has two drawbacks. First, offline training is required to determine the optimal classification threshold for ground segmentation, which limits the usability of the modeled probability distribution when extending to a new environment. Second, finding point estimates of the line parameters usually results in over fitted solutions that highly depend on the size and position of the (v,d) pairs remaining in filtered v-disparity image. The system presented in this paper handles these problems by directly modeling the occupancy probability distribution through the Bayesian linear regression framework, which is described in the next section.

3. Bayesian Linear Regression For Training Data Extraction

This section describes the training data extraction algorithm that we are proposing. First, for each new image the v-disparity image is extracted, then is filtered by using the

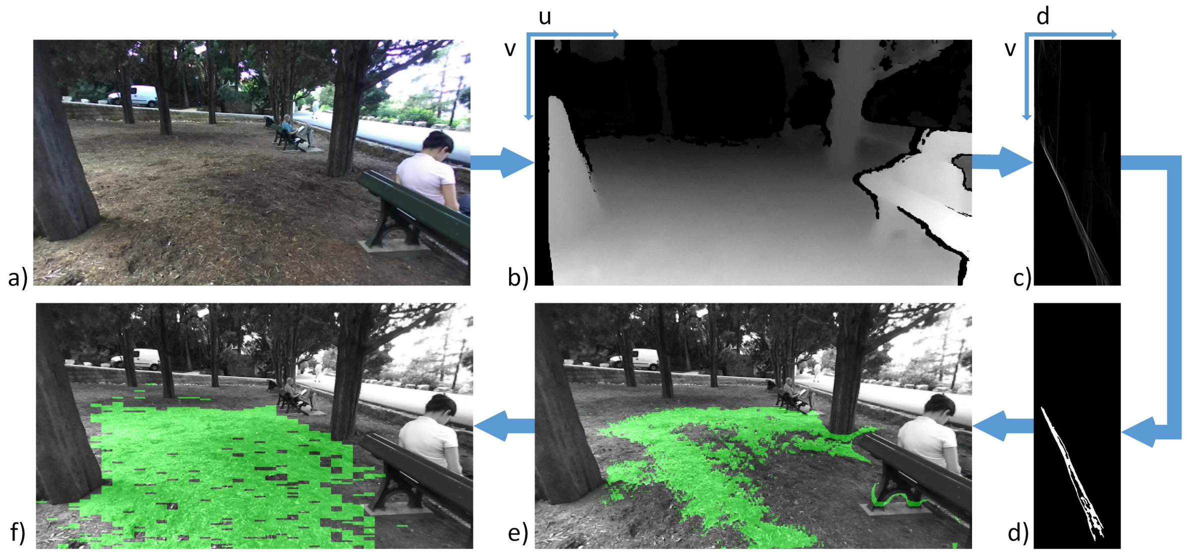

Figure 1: Flowchart of our system. **a)** Stereo Left Image. **b)** Stereo disparity image. **c)** v-disparity image. **d)** Filtered v-disparity image. **e)** Training pixels in green are extracted using our Bayesian linear regression framework, and are used as input to a One-Class Support Vector Classifier. **f)** Results of the Support Vector Classifier with the green area as the estimated free-space in the image.

binary filtering algorithm [10], resulting in an image containing only pixels belonging to the ground correlation line (Fig.1-d). The remaining (v,d) pairs in the image are used as training data input to Bayesian linear regression to learn the predictive probability distribution $P(d|v)$, where v is the input variable and d the target variable. Although the predictive distribution from Bayesian linear regression is usually used to predict new values of d given v, we found that we can use it to compute the probability of a disparity value d in the disparity image to belong to the ground correlation line, which is analogous to the probability of d belonging to the ground plane.

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:

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

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 $\mathbf{w} = [w_0\ w_1\ w_2]$. ε is a zero mean Gaussian random variable with precision β. The conditional distribution of d takes the following form:

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

The vectors $\mathbf{v} = [v_1 ... v_N]$ and $\mathbf{d} = [d_1 ... d_N]$ 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 $[d_1 ... d_N]$ are assumed to be IID variables drawn from the conditional distribution in (2) and as such, their likelihood function has the expression :

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

3.1.1 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:

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

For simplicity, the prior is considered to be zero mean and isotropic Gaussian with a single precision parameter α. This assumption reduces the number of unknown parameters in the prior 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:

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

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:

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

and Σ_w is the 3×3 covariance matrix:

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

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

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

Th 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:

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

where the variance Σ_p can be written as:

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

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 apriori.

3.1.2 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:

$$P(d|v,\mathbf{v},\mathbf{d}) = \int_\alpha\int_\beta\int_{\mathbf{w}} P(d|v,\mathbf{v},\mathbf{d},\alpha,\beta,\mathbf{w}) \, P(\mathbf{w}|v,\mathbf{v},\mathbf{d},\alpha,\beta) \, P(\alpha,\beta|v,\mathbf{v},\mathbf{d})d\mathbf{w}d\beta d\alpha, \tag{13}$$

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 hyperparameters at the highest level of the hierarchy (α and β) to their most likely values instead of integrating them out [11].

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

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

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

$$P(\alpha,\beta|v,\mathbf{v},\mathbf{d}) \propto P(\mathbf{d}|v,\mathbf{v},\alpha,\beta)P(\alpha,\beta|v,\mathbf{v}). \tag{14}$$

Due to the lack of knowledge of the hyperparametes 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:

$$\begin{aligned} \widehat{\alpha} &= \underset{\alpha}{argmax}\, P(\mathbf{d}|v,\mathbf{v},\alpha,\beta), \\ \widehat{\beta} &= \underset{\beta}{argmax}\, P(\mathbf{d}|v,\mathbf{v},\alpha,\beta), \end{aligned} \tag{15}$$

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

$$P(\mathbf{d}|v,\mathbf{v},\alpha,\beta) = \int_w P(\mathbf{d}|v,\mathbf{v},\alpha,\beta,\mathbf{w})P(\mathbf{w}|v,\mathbf{v},\alpha,\beta)d\mathbf{w}, \tag{16}$$

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:

$$\begin{aligned} \ln P(\mathbf{d}|v,\mathbf{v},\alpha,\beta) =& \\ &\ln\alpha + \frac{N}{2}\ln\beta - \frac{\ln|\Sigma_w^{-1}|}{2} \\ &-\frac{N}{2}\ln(2\pi) - \frac{\beta}{2}||\mathbf{d}-\Phi\mu_w||^2 - \frac{\alpha}{2}\mu_w\mu_w^T. \end{aligned} \tag{17}$$

The derivative equation with respect to α is:

$$\frac{\partial \ln P(\mathbf{d}|v,\mathbf{v},\alpha,\beta)}{\partial \alpha} = \frac{1}{\alpha} - \frac{1}{2}\left[\mu_w \mu_w^T + \frac{\partial \ln |\Sigma_w^{-1}|}{\partial \alpha}\right]. \quad (18)$$

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 we get:

$$\frac{\partial \ln |\Sigma_w^{-1}|}{\partial \alpha} = \sum_i \frac{1}{\lambda_i + \alpha}. \quad (19)$$

Setting the partial derivative in (18) to zero, the hyper-parameter α will have the form:

$$\alpha = \frac{1}{\mu_w \mu_w^T} \sum_i \frac{\lambda_i}{\lambda_i + \alpha}. \quad (20)$$

Proceeding with similar analysis with respect to the hyper-parameter β we obtain:

$$\frac{1}{\beta} = \frac{1}{N - \sum_i \frac{\lambda_i}{\lambda_i+\alpha}} \sum_{n=1}^{N} [d_n - \mu_w^T \phi^*(v_n)]^2. \quad (21)$$

We note 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 (20) and (21) until convergence. Convergence is determined when the difference between the old and new value of the hyperparameters is less than a specified tolerance. Initial value selection and the tolerance are discussed in the next section.

3.1.3 Training Pixel Extraction

After learning the hyperparameters $\widehat{\alpha}$ and $\widehat{\beta}$ from (20) and (21) respectively, the final form of the predictive distribution becomes:

$$P(d|v,\mathbf{v},\mathbf{d},\widehat{\alpha},\widehat{\beta}) = \mathcal{N}(d;\mu_w^T\phi^*(v),\Sigma_p), \quad (22)$$

with Σ_p computed from (12). The predictive distribution (22) is usually used to estimate new values of the target variable d given the row coordinate v as the input variable. Here, a confidence interval is specified over the PDF, and disparity values in the disparity image belonging to it are labeled as pixels belonging to free space in the image. An example of training pixel labeling is shown in Fig.1-e.

The algorithm contains only three free parameters which are the initial values of the hyperparameters, the tolerance for convergence, and the confidence interval. Since we are using the confidence interval for selecting training data, the initial value of the hyperparameters has no effect on the selection procedure. The tolerance is set to a very low value of 10^{-10} for both hyper parameters. This leaves the confidence interval to be the only real free parameter in the algorithm controlling the amount of data labeled as training pixels. In our implementation, disparity values laying in the 30% confidence interval are chosen as our training data. Increasing the width of the confidence interval yields more but less precise training data and vice-versa.

4. Second Stage Classification

This section briefly describes the second stage classification phase, which is necessary to evaluate the goodness of the extracted training data. At first, the image is separated to constant sized blocks in order to extract histograms necessary for the creation of the feature vector. Blocks are labeled as training blocks if at least 10% of their pixels include training pixels. Kim *et al*. [12] showed that the quality of free space estimation deteriorates as the size of the blocks increases. In contrast, the computational speed decreases as the size of the blocks increase. We chose 5×32 blocks as a compromise between quality and computation speed.

Each block in the image has associated with it a 20 dimensional feature vector comprised of:
From HSV space:

- An 8 bin histogram of hue.
- A 5 bin histogram of saturation.
- The mean value of hue.
- The mean value of saturation.

From RGB space:

- The mean value of R.
- The mean value of G.
- The mean value of B.

From XYZ space:

- The mean height of each block with respect to the stereo sensor.
- The difference in height between the highest and lowest point in the block.

This feature vector might not be the optimal for free space estimation, but is sufficient for comparing training data extraction methods.

The training data extraction algorithm can only extract data from one of the two classes, and therefore the classification problem is formulated as a one-class classification problem in which all the training data belongs to the positive class and where the negative class is severely under sampled. The ν-Support Vector Classifier proposed by [13] is chosen as the second stage classifier since it requires minimal free parameter selection. A standard off-the-shelf implementation of the ν-SVC is used and thus only the selection of free parameters in this implementation will be discussed. The three free parameters in the ν-SVC algorithm are ν, the kernel scale, and the outlier fraction. ν 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 ν and outlier fraction allows the ν-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.

5. Experiments and Results

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 ν-SVC and comparing the three corresponding output pixel labels.

5.1. 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 720×1280 colored images, their corresponding disparity image, and pixel X,Y,and Z coordinate with respect to the camera's coordinate frame. The images are captured by Stereo Lab's ZED stereo camera [14] and 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: 16 images taken with a hand held stereo camera on highly non-planar terrain(Fig.3-first row).

Medium dataset: 120 images taken with the a stereo camera mounted on UGV driven on a slightly non-planar park-like terrain (Fig.3-second and third rows).

Easy dataset: 145 images taken with the a stereo camera mounted on a UGV driven on a highly planar man-made terrain (Fig.3-fourth and fifth rows).

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. Furthermore, ground truth is generated manually for every frame of the three datasets.

5.2. Baselines

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 [8] 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 [9] and [12] 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 ν-SVC algorithm.

5.3. Analysis Of The Results

All experiments were done with the feature vector and ν-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 ν-SVC using the three algorithms are compared to ground truth labels to compute three performance criteria, which are the recall, precision, and specificity.

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 our case are the obstacles. The proposed algorithm's aim is two-fold, first to maximize all three performance criteria of the ν-SVC classifier and second, to keep its performance relatively the same over all the three types of terrain. Table 1 summarizes the mean recall, precision, and specificity of the ν-SVC classifier using training data from the three algorithms over all the frames of each datasets.

The ν-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.2-left where the recall is plotted as function of frames. Better relative performance of the ν-SVC using Bootstrapping on the Difficult dataset is mainly due to the constant color proper-

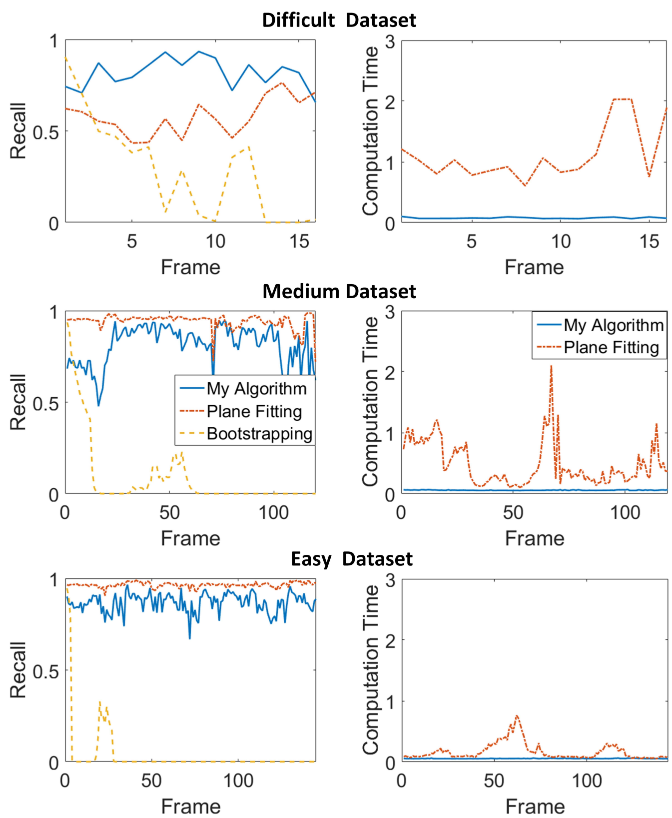

Figure 2: Plots of recall values of the ν-SVC and computational time (in seconds) of the three training data extraction methods per frame of the datasets..

ties of the ground in this dataset. The results of the ν-SVC using Bootstrapping are shown in the fifth column of Fig.3. 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 ν-SVC utilizing plane fitting reaches 0.9646 and 0.9422 recall on the easy and medium dataset respectively. Compared to the ν-SVC using our algorithm, which has a recall of 0.8671 and 0.8114 on the same datasets, the ν-SVC utilizing plane fitting seems to perform better. We 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 ν-SVC using our algorithm achieves a precision of 0.9604 and 0.9326 respectively vs 0.9340 and 0.8931 for the ν-SVC using plane fitting. The specificity of the ν-SVC using our 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 ν-SVC using plane fitting. On the Difficult dataset, a deterioration in the quality of classification of the ν-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.3 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 ν-SVC using our algorithm is able to provide a recall value of 0.8147, providing an increase of 0.2368 over the recall of the ν-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 our 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 1 shows that as the nature of the scene becomes more non-planar, the computation time of plane fitting increases. Furthermore, Fig.2-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 pro-

Table 1: Evaluation of the ν-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, our algorithm proves to produce better results.

ν-SVC using Our 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
ν-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
ν-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

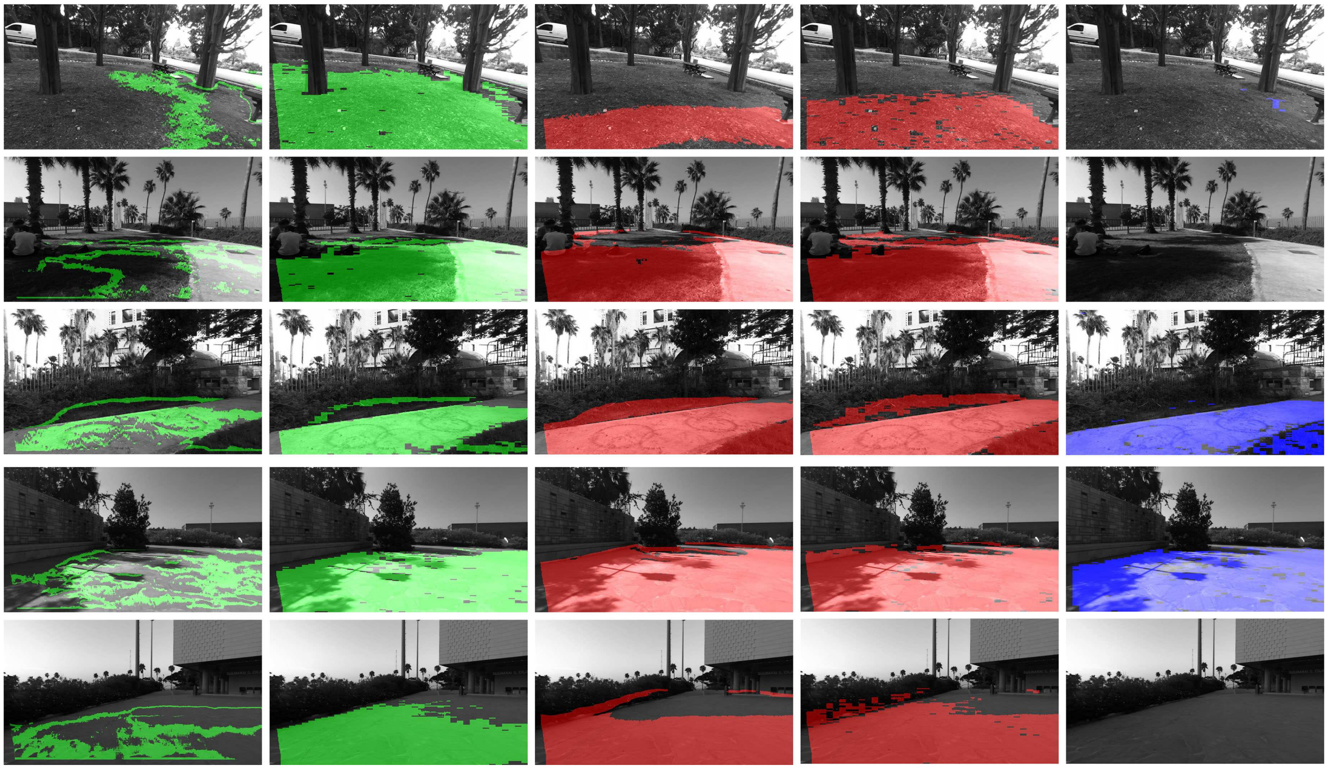

Figure 3: Examples of the training data extracted using our algorithm and the plane fitting algorithm (first and third columns respectively), and the final classification results obtained from ν-SVC using our algorithm (green), largest fitted plane algorithm (red) and bootstrapping (blue). Bootstrapping does not explicitly extract training data at each frame and thus only results of the final classification are shown.

posed algorithm shows a more consistent computation time whether across datasets (Table 1, fifth column) or across frames (Fig.2).

The intuition behind the improved performance provided by the ν-SVC using our 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 us to extract training data over the whole scene even in highly non-planar scenarios (Fig.3-first column, first row) and results in the computational time of our 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.3, where the training data provided by our algorithm results in better classification 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.

6. Conclusion

We presented a novel method to extract training data for free space classification. The proposed algorithm does not require any prior training, has only one free parameter, and is shown to provide consistent results over a variety of terrains, without requiring any terrain-specific tuning. ν-SVC using our algorithm for training data selection is shown to provide comparable results in planar terrain and much better results in highly non-planar ones over other methods in literature. Finally, our algorithm requires less computation time to extract data from environments of varying typologies. The time to extract the data is both low and consistent regardless of the environment being planar or not.

7. 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) .

References

[1] D. C. Asmar, J. S. Zelek, and S. M. Abdallah, "Tree trunks as landmarks for outdoor vision slam," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, 2006, pp. 196–196. 1

[2] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2807–2814. 1

[3] 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. 1, 2

[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 *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/suger15icra.pdf 2

[5] 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. 2

[6] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic terrain analysis for high-speed desert driving." in *Robotics: Science and Systems*, 2006, pp. 16–19. 2

[7] 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. 2

[8] 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. 2, 6

[9] 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. 2, 6

[10] A. Harakeh, D. Asmar, and E. Shammas, "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. 2, 3

[11] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006. 4

[12] 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. 5, 6

[13] 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. 5

[14] S. Labs, https://www.stereolabs.com/. 6