Natural Terrain Classification using Three-Dimensional Ladar Data for Ground Robot Mobility

Jean-François Lalonde, Nicolas Vandapel *, Daniel F. Huber, and Martial Hebert
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, USA

June 7, 2006

Abstract

In recent years, much progress has been made in outdoor autonomous navigation. However, safe navigation is still a daunting challenge in terrain containing vegetation. In this paper, we focus on the segmentation of ladar data into three classes using local three-dimensional point cloud statistics. The classes are: "scatter" to represent porous volumes such as grass and tree canopy, "linear" to capture thin objects like wires or tree branches, and finally "surface" to capture solid objects like ground surface, rocks or large trunks. We present the details of the proposed method, and the modifications we made to implement it on-board an autonomous ground vehicle for real-time data processing. Finally, we present results produced from different stationary laser sensors and from field tests using an unmanned ground vehicle.

1 Introduction

Autonomous robot navigation in terrain containing vegetation remains a considerable challenge because of the difficulty in capturing and representing the variability of the environment. Although it is not a trivial task, it is possible to build reliable models of smooth three-dimensional (3-D) terrain of bare ground as demonstrated in the context of planetary or desert exploration (Goldberg, 2002). It is however much more difficult to cope with areas that cannot be described by piecewise smooth surfaces like grass, bushes, or the tree canopy. Because such areas exhibit porous surfaces that surround scattered data, they are more naturally described by 3-D point clouds rather than by smooth surfaces. Reliable sensing, reconstruction, modelling and interpretation capabilities of such environments are critical for off-road navigation of unmanned ground vehicles (UGV). Tasks include reliably recovering of vegetation-covered ground for mobility analysis, detecting rocks or tree stumps hidden in grass, and discriminating dense foliage against solid obstacles such as trees.

In this paper, we address the problem of 3-D point cloud processing to support ground vehicle mobility. We present an approach that was demonstrated in real-time on-board an unmanned ground vehicle (Bornstein, 2003). This approach is decomposed into three main steps: classification, segmentation and semantic interpretation. Point-wise classification is based on local point cloud geometric analysis. The 3-D point cloud produced incrementally as the robot traverses the environment is segmented into three classes: surfaces (ground surface, rocks, large tree trunk), linear structures (wires, thin branches, small tree trunks) and porous volumes (foliage, grass). Those three classes correspond to three low-level geometric primitives that can capture the variability of natural environments. The features used are based on local spatial statistics extracted over a fixed-size support volume. For each class, the features distribution is learned prior to the robot mission using labelled data from representative terrains. During the mission, Bayesian classification is used to label the incoming ladar data. The second step consists of the extraction of connected components that group together the individual 3-D points based on the local consistency of various features. The third step consists of

*Contact author: vandapel@ri.cmu.edu

the geometric analysis of those components to refine the classification in order to further discriminate similar objects such as wires against branches.

One additional step, high-level geometric primitive modelling, is performed off-line. This final step consists of fitting geometric primitives to the different components extracted, allowing a compact and accurate representation of the environment that preserves the geometric and semantic properties of the scene. This is used for visualization purposes and illustrated in Figure 1.

(a) (b)

Figure 1: Real-time data processing for scene interpretation. (a) Unmanned vehicle used. (b) Automatically extracted geometric model.

The method presented is sensor independent and relies only on a small number of parameters. Almost all parameters are learned during a training phase that implicitly captures the sensor characteristics, the scanning patterns and the influence of the environment. In addition, the efficient data representation developed allows real-time data processing on-board the vehicle, that has been tested extensively.

This paper presents an extension and improvement over the work previously published in (Hebert, 2003) and (Vandapel, 2004a). In (Hebert, 2003), we presented results produced off-line using data coming from different laser radar sensors[1] in a variety of environments. In (Vandapel, 2004a) preliminary results from on-board processing on an unmanned vehicle, as well as additional results from stationary sensors, were presented. Although the core of the method is similar, this paper introduces previously unpublished new developments (scale selection, efficient data structure, filtering, semantic interpretation, high-level geometric modelling), and a new data analysis (evaluation of the classifier against ground truth). Results from a larger corpus of field testings and an in-depth analysis of the limitation of the current approach are also reported.

The state of art of scene interpretation for off-road navigation using 3-D information is presented in Section 2. Then, Section 3 details the approach proposed in this paper and Section 4 evaluates the approach using data from static laser radar. Section 5 focuses on the implementation of our approach for live data processing on-board a ground mobile robot. In Section 6, results from several field tests conducted in Central Pennsylvania are presented. Finally, Section 7, discusses the approach's key limitations and how they can be addressed.

2 State of the art

Because of its importance for obstacle detection and environment modelling, the issue of detecting and segmenting vegetation has been explored in the past. The use of spectral data has received much more attention than the use of geometric information.

In natural environment, numerous hazards threaten the mobility of robots such as terrain obstacles: step-like, negative (ditch), or slopes. Methods based on the convolution of a robot model with a 2D-1/2 grid map or in range images have been successfully applied for outdoor robot navigation. However these approaches are better suited for bare ground-like environment such as planetary terrain, open roads or desert environments, rather than for terrain with vegetation.

[1] In this paper, we use ladar, laser radar, laser scanner interchangeably.

Efforts have been made with some success to cope with vegetation but it still remains in the 2D-1/2 domain, not in 3-D like this paper proposes. Terrains with sparse vegetation cover, that allow perception of the load-bearing surface, can be reduced to the aforementioned class by recovering the ground surface as proposed in (Vandapel, 2003). Nevertheless, such approach cannot handle complex 3-D scenes containing thin structures (branches), overhanging vegetation, or obstacles in vegetation.

Terrain classification for vegetation detection has some success using color images, texture information and ladar data (Belluta, 2000) (Rasmussen, 2001) (Rasmussen, 2002) (Hong, 2002) (Dima, 2004). These techniques perform 2-D classification in image space and results are back-projected in 3-D from a calibrated sensor suite. Vegetation cannot be systematically labelled as obstacle. Otherwise, cross-country navigation, that requires driving over vegetation, will be impossible. On the other hand, this paper proposes a method to handle general 3-D scenes using a purely geometric approach.

A few approaches, pioneered by Huang *et al.* (Huang, 2000), considered single point statistics of range images of natural environments. However, to characterize texture, local statistics on range, derivatives of range and frequency components are needed. Macedo (Macedo, 2000) presented results on single point statistics computed on data from a single point laser range finder that showed differentiation between vegetation and solid surfaces like rocks. In (Castano, 2003), he presented a more geometric approach to this problem. The latter work relies on the use of a narrow beam, low elevation laser looking with a normal angle at short range obstacles for a small scale robot. In our case, we deal with a robot of larger size where the laser is located above ground level, introducing perspective effects, at longer range of up to 50-60 meters. We are interested in classifying the data into three basic classes. Depending on the laser performance and the vegetation permeability, our approach will also recover obstacles within the vegetation.

The idea of using vegetation permeability has been used in several projects. The idea is to keep track of how many times a laser beam can traverse voxels of a 3-D grid representation of the environment. The ratio of traverse versus hits is an indicator of the presence of solid surfaces or permeable volumes such as vegetation. Lacaze presented this approach in (Lacaze, 2002). Along with other methods, a similar approach is used in (Wellington, 2003) and in (Kelly, 2004) to recover the load bearing surface.

In (Anguelov, 2005), the authors use a Markov Random Field (MRF) framework to classify laser data collected from a ground vehicle into four classes (ground, trees, grass and building). The features used are the Principal Components on the local anisotropic points distribution, the point distribution within vertical columns and the maximum local elevation. Processing was performed off-board. Wolf (Wolf, 2005) uses a Hidden Markov Model to classify the terrain as traversable or non-traversable. He then uses a MRF approach as an off-line segmentation step to further refine results. The classification is performed on-board but no timing is reported.

In a recent paper (Manduchi, 2005), Manduchi reports on terrain classification and obstacle detection using stereo, laser and color camera from a large and small unmanned vehicle, without however reporting any timing information.In a related field, a large literature exists on the recovery of the ground terrain surface from airborne laser sensor, see (Sithole, 2004) for a review and comparison of the methods. This includes the filtering of the vegetation and the interpolation of the terrain surface.

3 Approach

Our approach is based on 3-D point cloud statistics used to compute saliency features that capture the spatial distribution of points in a local neighborhood. The saliency features distribution is automatically captured by a Gaussian Mixture Model (GMM) using the Expectation Maximization (EM) algorithm. Given such a model, trained off-line, new data can be classified on-line with a Bayesian classifier (Duda, 2000).

3.1 Local point statistics

The saliency features used are inspired by the tensor voting approach of Medioni (Medioni, 2000). Instead of using the distribution of the estimated normals, the distribution of the 3-D points are used directly. The local spatial point distribution, over some neighboring area, is captured by the decomposition into principal components of the covariance matrix of the 3-D points position. The size of the neighborhood considered, the support region, defines the scale of

the features (see Section 4.1 for experimental evaluation and Section 7 for further discussion on this matter).

The symmetric positive definite covariance matrix for a set of N 3-D points $\{X_i\} = \{(x_i, y_i, z_i)^T\}$ with $\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$ is defined in Equation 1.

$$\frac{1}{N}\sum_{i=i}^{N}(X_i - \overline{X})(X_i - \overline{X})^T \tag{1}$$

The matrix is decomposed into principal components ordered by increasing eigenvalues. $\overrightarrow{e_0}$, $\overrightarrow{e_1}$, $\overrightarrow{e_2}$ are the eigenvectors corresponding to the eigenvalues λ_0, λ_1, λ_2 respectively, where $\lambda_0 \geq \lambda_1 \geq \lambda_2$. Figure 2 illustrates the three features used.

In the case of scattered points, we have $\lambda_0 \backsimeq \lambda_1 \backsimeq \lambda_2$ and no dominant direction can be found. In the case of a linear structure, the principal direction will be the tangent at the curve, with $\lambda_0 \gg \lambda_1, \lambda_2$. Finally, in the case of a solid surface, the principal direction is aligned with the surface normal with $\lambda_0, \lambda_1 \gg \lambda_2$ and $\overrightarrow{e_0}$, $\overrightarrow{e_1}$ span the local plane of observations.

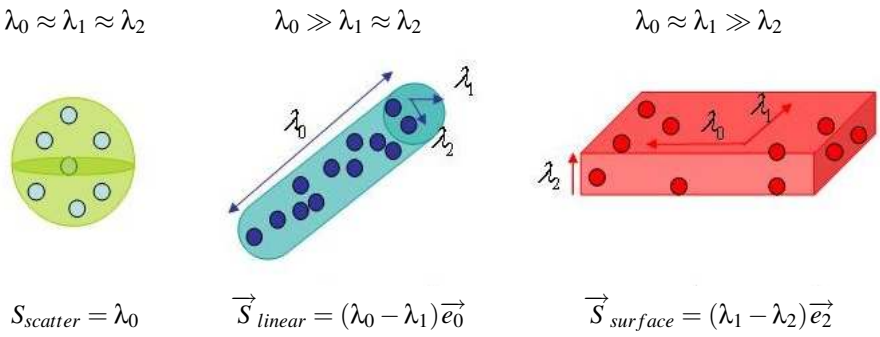


Figure 2: Illustration of the saliency features.

As is shown in Figure 2, the three saliency features, named *scatter-ness*, *linear-ness* and *surface-ness*, are linear combinations of the eigenvalues.

In practice, it is not feasible nor desirable to hand-tune thresholds directly to perform classification because those values may vary considerably depending on the type of terrain, the type of sensor, and the point density. A standard way of automating this process is to train a classifier that maximizes the probability of correct classification on a training data set. This is the object of the next two sections.

3.2 Classification

3.2.1 Training model

A parametric model of the saliency features distribution is learned by fitting a Gaussian Mixture Model (GMM) using the Expectation-Maximization (EM) algorithm on a hand labelled training data set. This approach is retained because its properties are well-known and it is suitable for fast data processing with only three features. See (Bilmes, 1997) for practical details on the EM algorithm and (Duda, 2000) for classification using GMM. The resulting density probability model for each class is the sum of n_g Gaussians with weight, mean and covariance matrices $\{(\omega_i, \mu_i, \Sigma_i)\}_{i=1\ldots n_g}$.

In order to capture the variability of the terrain, the training set must contain data from flat surfaces, rough bare surfaces and short grass terrains, from thin branches and power line wires, from dense and sparse tree canopy and finally tall grass. The labelling process is performed using a graphical interface which allows the selection of 3-D points individually or in groups. We enforce a balanced labelled dataset between the three different classes. In order to capture the influence of surrounding clutter, saliency features are computed only at points for which it is visually easy to confidently determine their class, but the support region includes all the points. This causes saliency features of branches to capture the influence of the leaves, for example.

This labelling and model fitting is performed off-line and only once for each sensor. Once a model is obtained, previously unseen data points can be efficiently labelled. This is the focus of the next section.

3.2.2 Classifier

Let $M^k = \{(\omega_i^k, \mu_i^k, \Sigma_i^k)\}_{i=1\ldots n_g^k}$ be the GMM of the k^{th} class. Let $S = (S_{scatter}, S_{linear}, S_{surface})$ be the saliency features of a new point to be classified. The conditional probability that the point belongs to the k^{th} class is given by

$$p(S|M^k) = \sum_{i=1\ldots n_g^k} \frac{\omega_i^k}{(2\pi)^{d/2}|\Sigma_i^k|^{1/2}} e^{-\frac{1}{2}(S-\mu_i^k)^T \Sigma_i^{k\,-1}(S-\mu_i^k)} \tag{2}$$

where d is the number of saliency features. The class is chosen as

$$k_{max} = \underset{k}{\text{argmax}}\{p(S|M^k)\} \tag{3}$$

The normalized classification confidence is defined as

$$\frac{p(S|M^{k_{max}})}{p(S|M^{k_{max}}) - \max\limits_{k \neq k_{max}} \{p(S|M^k)\}} \tag{4}$$

3.3 Filtering

The features used and the parametric data modelling approach have expected limitations in capturing the scene geometry for classification. This section presents four different basic filtering methods to remove spurious misclassifications that occur in particular circumstances: edge effect, density variation, lack of support data and isolated results. All filters are implemented as point-wise and one-pass local voting scheme, and performed after classification.

For each point initially classified as linear, the *edge* filter determines a neighborhood of fixed size around it. It then computes the most frequent class among all points present in the neighborhood, and changes the interest point's class to this value. This allows the removal of isolated linear points, which often happens at the edges of surfaces. The *isolated surface* filter performs a similar operation, on points that are initially classified as surface.

The *isolated density* filter allows for the removal of isolated points in low density regions. It loops over all the points and removes a point from the dataset if the number of points in its neighborhood is below a fixed threshold. From testing, five neighbors is sufficient to yield satisfying results.

The *ground* filter is somewhat different because it attempts to identify ground points. Based on the observation that the volume below a ground point should be empty, the filter defines the neighborhood region of each point as a cone with its opening pointing downwards, centered at the point of interest. The filter then counts the number of points lying in that cone, and changes the class to "surface" if the number is below a threshold, typically three points. See (Vandapel, 2003) for more details. The four filters are summarized in Table 1. Results of the application of such filters are shown in Section 4.4.

Filter name	Affects	Effect
Edge	linear	Change class to most frequent class among neighbors
Isolated surface	surface	Change class to most frequent class among neighbors
Isolated density	all	Discard point if number of neighbors is smaller than threshold
Ground	all	Downward cone, change class to "surface" if number of neighbors is smaller than threshold

Table 1: Summary of the different point-wise, one-pass local voting filters implemented.

3.4 Grouping

In the next two sections, we show that point-wise classification results can be used to obtain a semantic description of the environment. The first step consists in grouping points that share similar characteristics into connected components. The connected component algorithm employs a one-pass seeded region growing algorithm, akin to standard image processing methods, which exploits class, direction and spatial connectivity attributes of neighboring points.

A point is first chosen at random (the *seed*), and a new empty component is created. The algorithm retrieves all the points in a neighborhood of fixed size around it. All neighbors are analyzed to determine if they have the same class as the seed and if the angular separation between their direction vector ($\overrightarrow{e_0}$ for the "linear" class or $\overrightarrow{e_2}$ for the "surface" class) and the seed's lies below a fixed threshold. This last test is not performed for the "scatter" class because there is no dominant direction (see Figure 2). Neighbors that satisfy these conditions are added to the component and to a temporary queue. The algorithm then loops over all the points in the queue, and performs the same procedure. Once the queue is empty, a point that has not been visited yet is randomly selected as a seed, and the same operations are repeated until all points are visited. Results of the application of the algorithm are shown in Sections 4.3 and 4.4.

3.5 Semantic interpretation

The last step performs a geometric analysis on the components themselves to distinguish between different types of objects such as branches, tree trunks, ground, wires and vegetation for example. To distinguish between wires, branches, and tree trunks, the principal direction $\overrightarrow{d}_{component}$ of the linear object is estimated by computing the covariance matrix over all the points of that component, and by extracting the eigenvector associated to the largest eigenvalue. This is similar to the local point statistics presented in Section 3.1, but it is performed at the component level rather than at the point level. The diameter is also estimated by projecting the point centers to the plane perpendicular to the direction $\overrightarrow{d}_{component}$, and by computing the average distance between each projected points and their centroid. Simple tests based on these values allow us to reliably discriminate between different objects. Depending on their size, some tree trunks might also appear as surface patches. In this case, a simple test measuring the angular deviation from the vertical direction is used to discriminate with the ground. Finally, objects with "scatter" class are identified as vegetation.

In addition, we create a high-level scene model by fitting simple geometric primitives to each of the identified objects (Lukacs, 1998). For instance, wires, branches and tree trunks are represented by cylinders, vegetation is modeled by ellipsoids and the ground is a digital elevation mesh. The resulting scene model is very simple, intuitive to understand and can easily be visualized using traditional 3-D rendering techniques. Examples of such results are shown in Section 6.5.

4 Evaluation

This section presents an evaluation of the approach proposed above using data collected by three stationary ground sensors: an actuated SICK laser, a high resolution and high density Zoller-Fröhlich (Z+F) laser, and a long range Riegl scanner. First the sensors used are presented, followed by classification results for two forest scenes, two scenes containing linear wires, and an open space. Next, the improvement in classification due to the post-processing methods is evaluated, before quantitatively comparing the results against manually labelled data. Throughout this section, we show that our approach is efficient with different sensors and versatile when tested against different kinds of environments.

4.1 Model parameters

The results presented in this section use a discretized space representation made of 10 cm edge voxels (see Section 5.2.1 for more details). The optimal number of Gaussians n_g necessary to capture the saliency features distribution without

[1]The figures in this paper are best viewed in color. Unless indicated otherwise, we use the red/blue/green colors (or dark/darker/light grey) to display information concerning surface/linear/scatter classification results or features. The saturation of colors represents class confidence. Points are enlarged to assure proper rendering in a printed version of the paper.

over-fitting the training data was determined experimentally by cross-validation. This procedure involved testing for values of n_g ranging from one to six in a variety of different training and test sets. The best results were achieved by using $n_g = 3$, and this number is used in the experiments presented throughout this paper. In each of these experiments, the correct convergence of the EM algorithm was observed. A support region size of 45 cm radius has been determined to be optimal using a similar procedure.

4.2 Static sensors characteristics

Some of the data presented in this section was collected using a SICK LMS291 attached to a custom made scanning mount similar to the sensor found in (Wellington, 2003). The laser can be seen in the bottom-right of Figure 4-(a). The laser collects 60,000 points per scan. The angular separation between laser beams is $\frac{1}{4}$ degree over a 100° field of view (FOV). The angular separation between laser sweeps is $\frac{2}{3}$ of a degree over 115°.

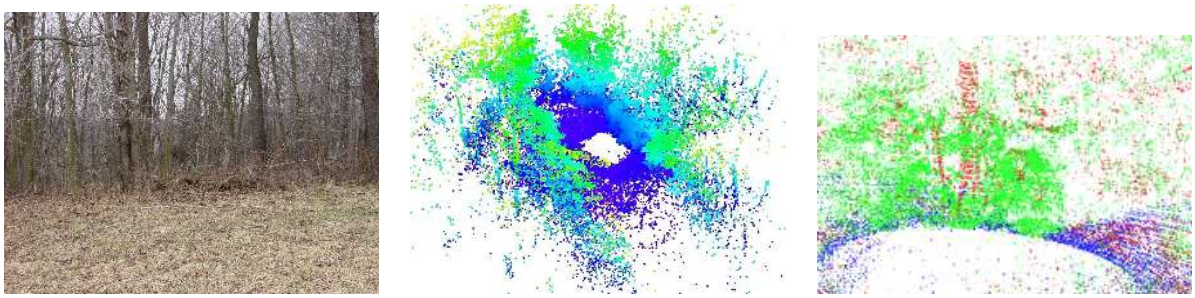
Figure 3 presents results produced using data from a Zoller-Fröhlich (Z+F) LARA 21400 3-D imaging sensor. It has a maximum range of 21.4 m with millimeter accuracy, a $360° \times \pm 35°$ FOV; it produces 8000×1400 range and intensity measurements per scan (Langer, 2000).

Figure 8 presents results produced using data from a Riegl LMS-Z210 3-D imaging sensor. The maximum range is 350 m with 2.5 cm resolution. The laser records the first and last pulse, but only the first pulse is used here. Each scan produces 444 x 740 points.

4.3 Examples of classification results

4.3.1 Forest environment

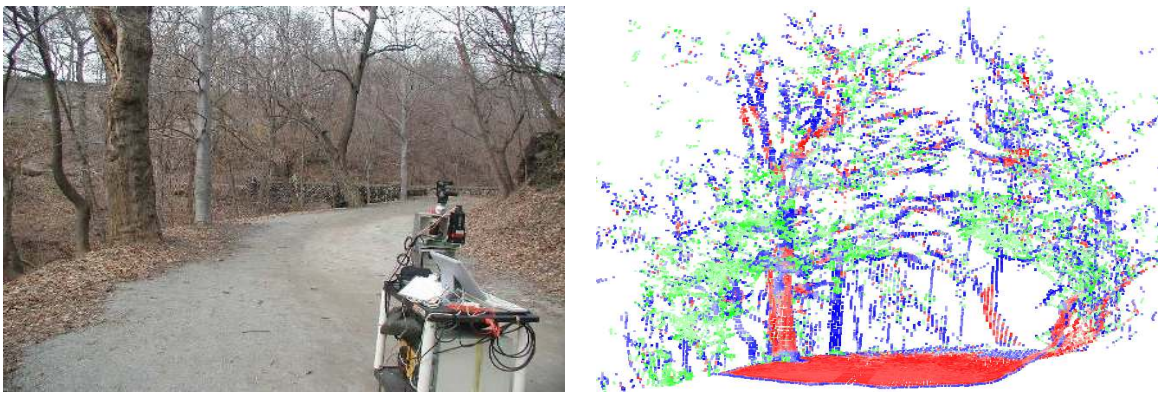
For this first example, data was acquired by positioning the Z+F laser on a trail in a densely vegetated terrain: flat ground was covered by thick vegetation and a trail crossed a densely forested area. Figure 3-(a) shows a picture of the scene. Figure 3-(b) shows the 3-D data where the elevation is color-coded from blue to red for low to high elevation. The circular hole is due to the self-occlusion of the laser. Figure 3-(c) shows a close-up view of the segmentation results of one the scene areas. In this first example, the "scatter" and "linear" class are fused together and the ground surface class points are separated from other surface class points using a geometric method presented in (Vandapel, 2003). One can note that the tree trunk, even though surrounded by vegetation, is correctly classified.



(a) (b) (c)

Figure 3: Example of classification results with the Z+F laser. (a) Scene. (b) 3-D point cloud with the elevation color-coded. (c) Segmented scene. Points in red, blue, green (or dark/darker/light grey) represent non-ground surfaces, ground surface, scatter structures respectively.

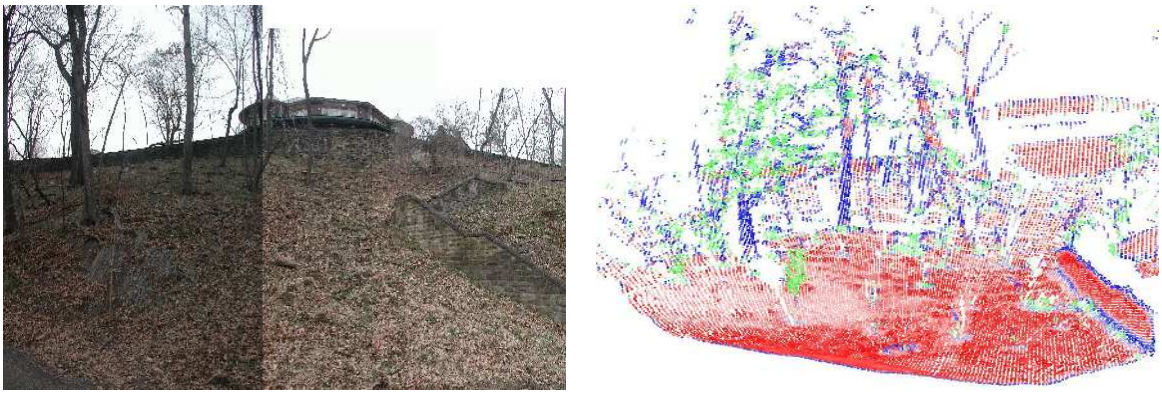
Figure 4 shows an example with data obtained with the actuated SICK laser from a forest scene. The laser was on a flat and bare ground trail in a wooded area with large trees; thus the keyword *flat* will be used to refer to this dataset. The terrain trail and the side slope, as well as large tree trunks, thin branches, and smaller tree trunks, are recovered correctly. In this example, there are three different causes of misclassification: edge effect (at the border of surface areas), presence of several surfaces (junction of tree trunks with ground surface) and fixed-scale neighborhoods (at the end of the trail, due to the slant angle effect of the laser).



(a) (b)

Figure 4: Example of classification results with the actuated SICK laser (*flat* dataset). (a) Scene. (b) Classification results. See footnote page 6 for color code.

Figure 5 shows another example produced using the actuated SICK laser. In this case, the scene has a larger depth than before and it includes a terrain slope with a rough surface containing thinner structures. It is identified by the keyword *rough*. Again, the errors are caused by edges, scale and multiple manifolds.



(a) (b)

Figure 5: Example of classification results with the actuated SICK laser (*rough* dataset). (a) Scene. (b) Classification results. See footnote page 6.

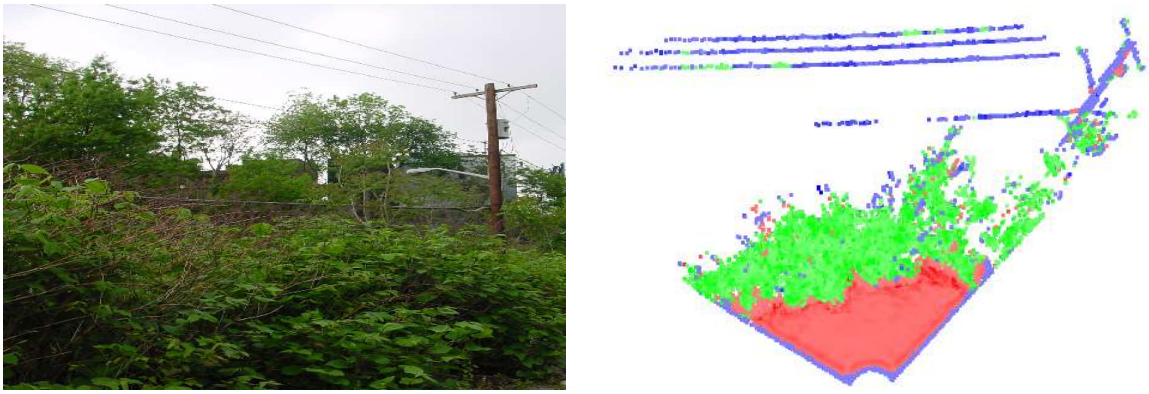
4.3.2 Thin linear structures segmentation

The next two examples show linear structure classification results of wires. Even though they are not natural features, wires can constitute major obstacles for ground robot mobility and also for unmanned aircraft flying at low altitude. In Figure 6, we present a classification result for isolated wires from a power line. The wires are correctly recovered as linear features even though they are close to each other.

The example from Figure 7 is even more challenging because wires are cluttered by surrounding foliage. The power line is located in the vicinity of a tree line. Note that the pole supporting the cables is correctly classified.

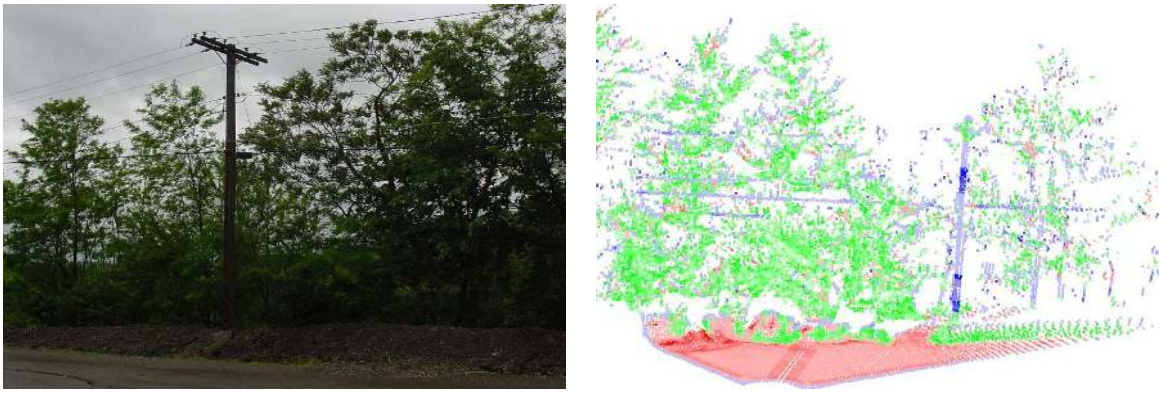
4.3.3 Open space environment

Figure 8-(a)/(b) shows the classification result and the largest connected components for a scene containing a grassy terrain, short trees and a chain-link fence covered by vegetation. Some of the ground points are misclassified as linear



(a) (b)

Figure 6: Example of classification with the SICK laser. Isolated wires. (a) Scene. (b) Classification results. See footnote page 6 for color code.



(a) (b)

Figure 7: Example of classification with the SICK laser. Wires adjacent to clutter. (a) Scene. (b) Classification results. See footnote page 6 for color code.

because of the scanning pattern of the sensor. Specifically, two consecutive scan lines project far from each other in one dimension, but the spacing between laser points within each scanline remains close to each other. Currently, this approach cannot deal with such artifacts, but Section 7 discusses a geometric method we introduced recently based on automatic scale selection to address that problem.

4.4 Post-processing

Section 3.3 introduced several one-pass post-processing filters that were implemented in order to reduce the effects of some sources mentioned. The approach relies on ensuring local classification consistency based on class assignment and geometric relationship between a specific point and its neighbors through a majority voting scheme.

Figures 9 and 10 show classification results for the *flat* and *rough* datasets respectively (see Figures 4 and 5). Each figure includes the raw classification results (a), the edge points filtered (b), the isolated surface point filtered (c), the isolated 3-D points (d). Figure 9-(e) illustrates the ground surface filtering results while Figure 10-(e) shows the connected components.

The next example (Figure 11) shows the use of the saliency direction to connect elements from the same class. This allows the separation of the side terrain slope and tree trunks from the ground, the load bearing surface.

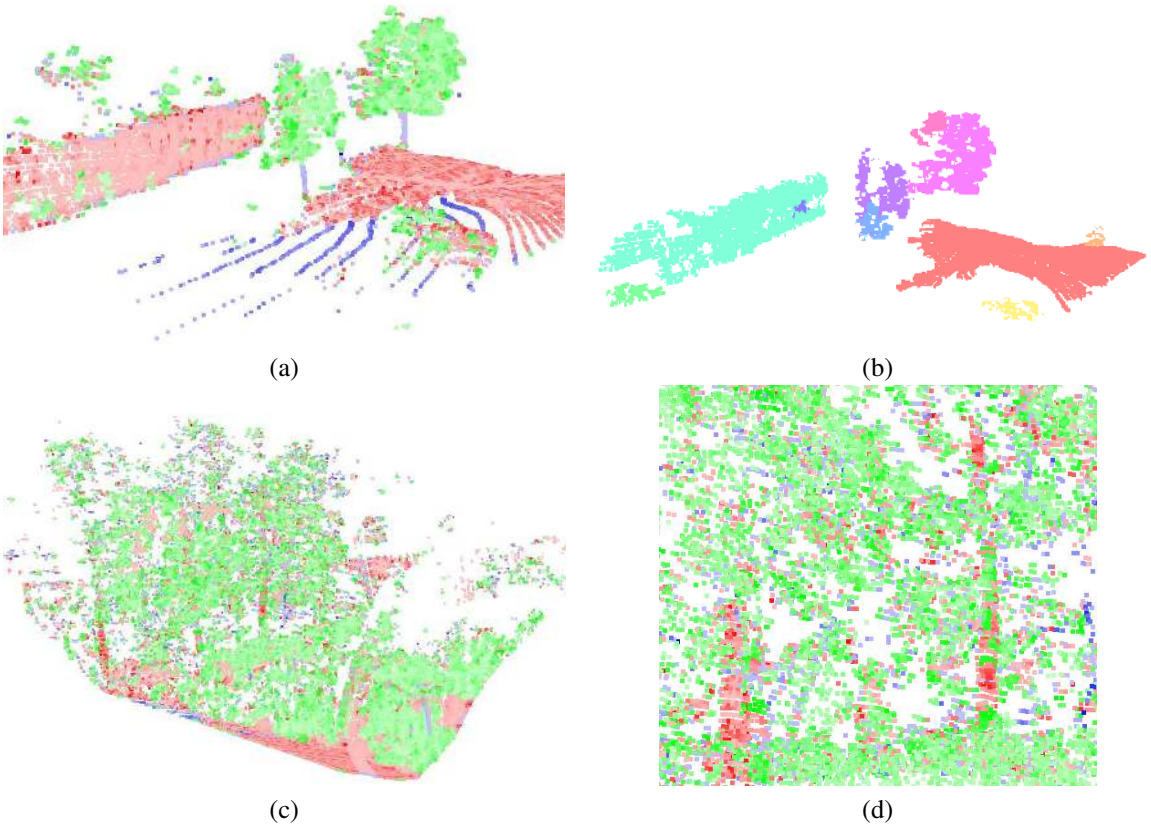


Figure 8: Example of classification results with the Riegl laser. (a)-(b) Open space with chain-link fence:(a) Classification, (b) Connected components, each component is represented by a different color (or grey tone). (c)-(d) Dense vegetation: (c) Classification, (d) Close-up view. See footnote page 6 for color code.

4.5 Classification evaluation

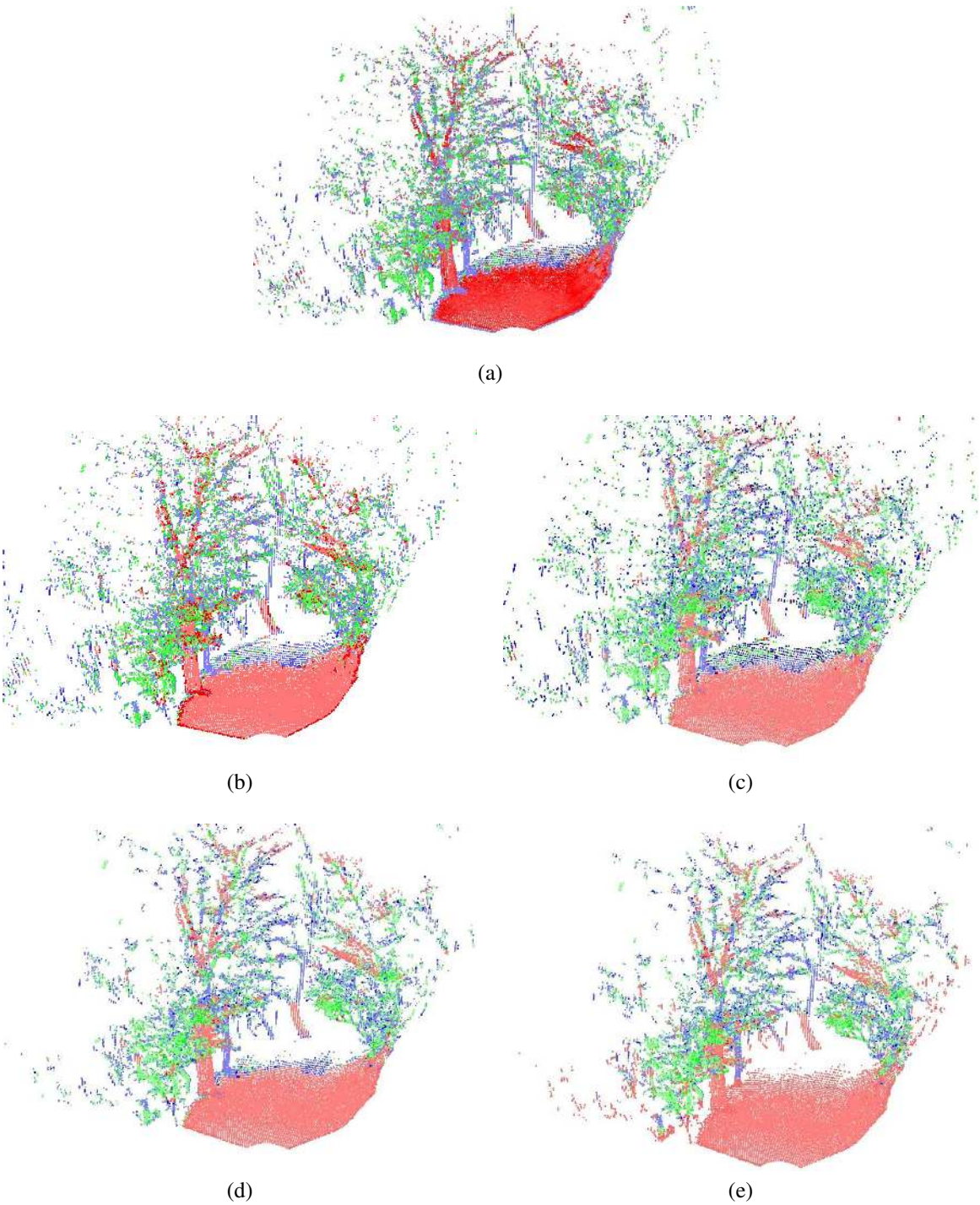
To evaluate the accuracy of the classification results, quantitative results obtained by comparing the output of the classifier with hand-labeled data are presented, for the *flat* and *rough* datasets (see Figures 4 and 5). Those data sets were not part of the training data set.

The labeling method requires manual identification of points that belong to each of the three classes. However, because of varying point density, clutter and occlusions, it is sometimes very hard to determine visually the true nature of a group of points. When the true class is too ambiguous and cannot be identified by a human expert, the corresponding points are set as "undetermined" and are not used in the accuracy computation. For the *rough* dataset, 320 points (0.97%) were undetermined, as opposed to 1997 (5.7%) for the *flat* dataset.

Classification accuracy for the *rough* dataset varies from 71.85% for the raw classification results to 79.38% when applying the isolated density filter (see Table 2). The confusion matrices for each filtering method are shown in Table 3. Generally, classification accuracy is very good for surfaces, with values between 83% and 93%. However, this is not the case for the linear and scatter classes, where the accuracy is lower, especially for scatter. This is because scattered points are very difficult to segment manually, and hidden linear structures or surfaces may appear as clutter to the human expert eye. This explains the high percentage of scatter points classified as linear (around 45%).

Filtering methods improve the accuracy in classification of surfaces and scatter points. However, because of the voting nature of the filtering methods and the fact that linear structures are often surrounded by vegetation or surfaces, filtering decreases classification accuracy for the linear class. Since the number of linear points in a scene is always much lower than the two other classes, a global increase in accuracy is still observed, as shown in Table 2.

Classification accuracy for the *flat* dataset is somewhat lower than in the previous example. Table 5 shows that



(a)

(b)

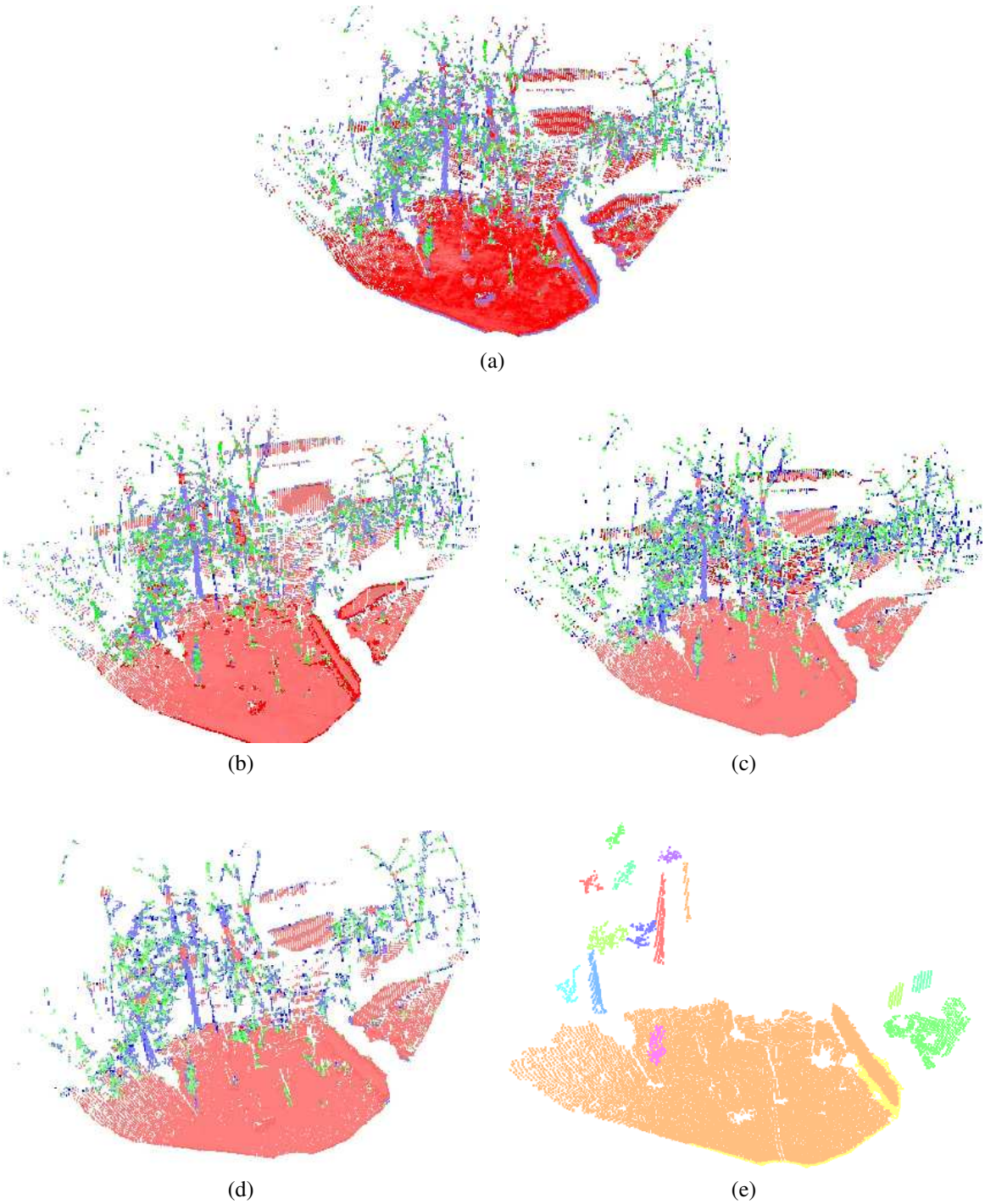
(c)

(d)

(e)

Figure 9: Example of data post-processing for the *flat* dataset. (a) Raw classification. (b) Edge effect filtered. (c) Isolated surface class points filtered. (d) Isolated 3-D points filtered. (e) Ground filtered. See footnote page 6 for color code.

classification accuracy varies from 55.84% for the raw classification results to 61.94% after applying the ground filtering method. However, after closer inspection, the confusion matrices for the *flat* dataset in Table 6 are very similar to the matrices for the *rough* dataset shown above. The difference is that the global percentage of scattered points (55.29%, see Table 7) is much higher than in the previous case (21.95%, see Table 4). Thus the behavior of our approach is similar in both cases, the difference in the number of low accuracy scattered points explains the difference in global accuracy.



(a) (b) (c) (d) (e)

Figure 10: Example of data post-processing for the *rough* dataset. (a) Raw classification. (b) Edge effect filtered. (c) Isolated surface class points filtered. (d) Isolated 3-D points filtered. (e) Connected components. Each color (or grey tone) represents one connected component. See footnote page 6 for color code.

4.6 Off-line performance

To evaluate the performance of the current approach, it is necessary to estimate how fast the data structure can integrate new points coming from the laser into voxels as well as evaluate the two principal operations: saliency computation (computation of the covariance matrix and extraction of its principal components, see Section 3.1) and classification (computation of the conditional probabilities, see Section 3.2.2). Table 8 shows the off-line timing results averaged

(a) (b)

Figure 11: Example of post-processing for the *flat dataset*. Connected components without (a) and with (b) feature direction. Each color (or grey tone) represents one connected component. Only the largest components are displayed.

	Initial	Edge	Isolated surface	Isolated density
total # of pts misclassified	9065	8198	7806	6061
total # of pts verified	32204	32641	32641	29405
misclassification rate (%)	28.14	25.11	23.91	20.61
accuracy (%)	71.85	74.88	76.08	79.38

Table 2: Overall statistics for the *rough* dataset

	Initial			Edge			Isolated surface			Isolated density		
	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface
scatter	38.35	48.76	12.88	38.65	44.57	16.77	46.18	45.60	8.20	45.29	46.11	8.58
linear	20.95	70.76	8.28	32.83	56.35	10.81	35.28	58.35	6.35	35.79	57.45	6.75
surface	3.89	12.78	83.32	4.15	5.98	89.86	5.45	5.77	88.76	3.12	3.93	92.93

Table 3: Confusion matrices (in %) for the *rough* dataset. Rows are ground truth and columns are computed classes.

	Initial	Edge	Isolate surface	Isolated density
scatter	23.05	23.54	23.54	21.98
linear	8.77	8.72	8.72	8.66
surface	68.16	67.72	67.72	69.34

Table 4: Number of points per class (in %) for the *rough* dataset.

	Initial	Edge	Isolated surface	Isolated density	Ground
total # of pts misclassified	14376	14454	12980	11472	11433
total # of pts verified	32555	33011	33011	30046	30046
misclassification rate (%)	44.15	43.78	39.32	38.18	38.05
accuracy (%)	55.84	56.21	60.67	61.81	61.94

Table 5: Overall statistics for the *flat* dataset

	Initial			Edge			Isolated surface			Isolated density			Ground		
	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface	scatter	linear	surface
scatter	39.53	44.65	15.80	40.07	35.48	24.43	47.80	36.84	15.35	47.19	36.49	16.30	45.11	33.70	21.17
linear	23.49	62.91	13.58	32.11	49.11	18.76	35.29	51.54	13.16	35.42	50.44	14.13	31.32	44.26	24.40
surface	2.74	14.25	82.99	2.90	7.18	89.90	3.50	7.37	89.12	2.58	5.63	91.77	0.88	0.84	98.26

Table 6: Confusion matrices (in %) for the *flat* dataset. Rows are ground truth and columns are computed classes.

over 6 datasets of different sizes and composition. On average, our method is able to process 6800 voxels per second on a Pentium IV at 3 GHz. While this measure is suitable for off-line performance evaluation, we show in Section 6.3 a more appropriate method of reporting on-line timing.

	Initial	Edge	Isolated surface	Isolated density	Ground
scatter	56.31	56.67	56.67	55.29	55.29
linear	13.35	13.36	13.36	12.83	12.83
surface	30.33	29.95	29.95	31.87	31.87

Table 7: Number of points per class (in %) for the *flat* dataset.

Insertion	Saliency	Classification	Saliency & classification combined
1.7×10^6 points/sec	9000 voxels/sec	29000 voxels/sec	6800 voxels/sec

Table 8: Timing results for off-line batch processing.

5 Rover implementation

This section discusses computational issues related to the implementation of our approach on-board a mobile robot. Unfortunately, the method presented above cannot be used without modification for that purpose. After justifying this claim and detailing the necessary modifications to achieve fast processing of the information on-board a ground mobile robot, we finally introduce the data flow implemented on the autonomous unmanned vehicle used.

5.1 Issues

Several issues arise with the implementation of the previous approach on-board a mobile robot:

- The mobility laser on the robot has a very high acquisition rate, in excess of 100,000 points per second.
- Because the turret, on which the laser is mounted, and the robot are in motion, the same area of the scene is perceived several times under different viewpoints and at different distances. This is an advantage in term of coverage of the environment, but it implies incorporating the new data continuously in the existing data structure and recomputing the classification for already perceived scene areas.
- If the robot is stationary for some time, there is no need to accumulate a huge amount of data of the scene from the same viewpoint.
- The method requires the use of data in a support region around the point of interest considered, which is a time consuming range search procedure.
- For each new laser data point added, the saliency features and classification need to be recomputed for the prototype point at that location. Furthermore, they must also be recomputed for every other prototype point that has new laser data in its neighborhood.
- As the robot moves in the environment, the data need to be incorporated into a global map or in a smaller map that needs to be scrolled.

In the rest of this section, we describe the solutions that we have implemented to deal with these problems.

5.2 Practical implementation

5.2.1 Data structure

In (Hebert, 2003) we directly use the points produced by the laser stored in a dense data representation. This approach is too expensive for on-board processing because of the number of individual 3-D points to consider for update and classification, and also because of the size of the data structure to maintain while the robot is in motion. Using the data collected during our experiments, it is estimated that the percentage of voxels occupied in a dense data representation varies between 2 % and 12 % when the voxel size varies between 10 cm and 1 m in edge length. As a result, a sparse voxel data representation was chosen. Each basic element, a cube of 10 cm edge, is called a *prototype point*. The 10 cm

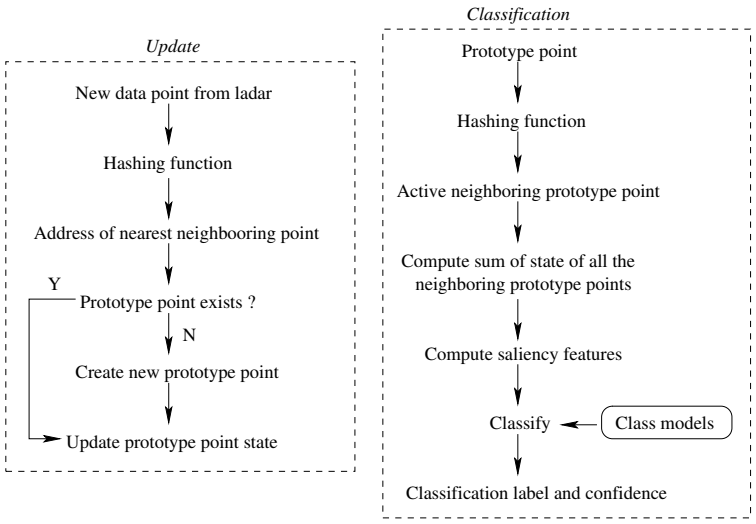


Figure 12: Flowchart of the current system implementation

size was chosen as a compromise between computational efficiency, memory management, and scene reconstruction accuracy. Note that 45 cm is the radius used for the support region to compute the features. Instead of storing all raw 3-D points in memory, each prototype point maintains only the sufficient statistics for the covariance matrix of the raw points that fall inside its bounds. This allows efficient memory usage and exact feature computation. In order to take advantage of the incremental computation of the saliency, the contribution of the points based on their distance to the point of interest is not weighted. Similarly, data aging was considered too costly to implement. The complete set of prototype points is stored in a structure called a *prototype volume*. It allows efficient access to the prototype points via a hashing function to do range searches. The hash key is 64 bits long and made of the concatenated index value of the Z,Y and X coordinates. The length of the key ensures that the global map is large enough and does not need to be scrolled. The X,Y coordinates indices are 20 bits long and the Z component 14 bits long.

5.2.2 Update and classification

The flowchart of the current system is shown in Figure 12. It includes two sequential processes: the update of the intermediate saliency features (performed continuously) and the classification of the data (performed on request or at regular intervals). The update consists in incorporating the new 3-D point either by creating a new prototype point or by updating an existing one. The classification step then computes the local saliency features in the support region by performing a range search operation and combining the information stored in each neighboring prototype point. Classification is done as described in Section 3.2.2. The update and classification steps are currently performed sequentially. Five ladar frames are accumulated, then updated or newly created prototype points are classified.

Raw data points are incorporated into the data structure at a rate of one million points per second. The classification is performed at the rate of 6,600 prototype points per second for a 45 cm radius support region. This approach allows incorporation of the data in real time as it comes from the ladar.

5.2.3 Speed-up

Several additions are necessary to improve classification speed and to allow real-time processing. First, the number of prototype points to be processed is greatly reduced by limiting classification to an interest area around the robot. This is justified because it is typically more important to process the data lying at close range, in front of the robot. Unless otherwise specified, an area of 20×20 m in front of the robot is used.

In addition, a partial update strategy is implemented to avoid re-classifying the same prototype point multiple times. Ideally, one would want to (re)compute the saliency features each time a new prototype point is either created or updated. However, because of the small size of the prototype point, this step can be skipped with an acceptable loss

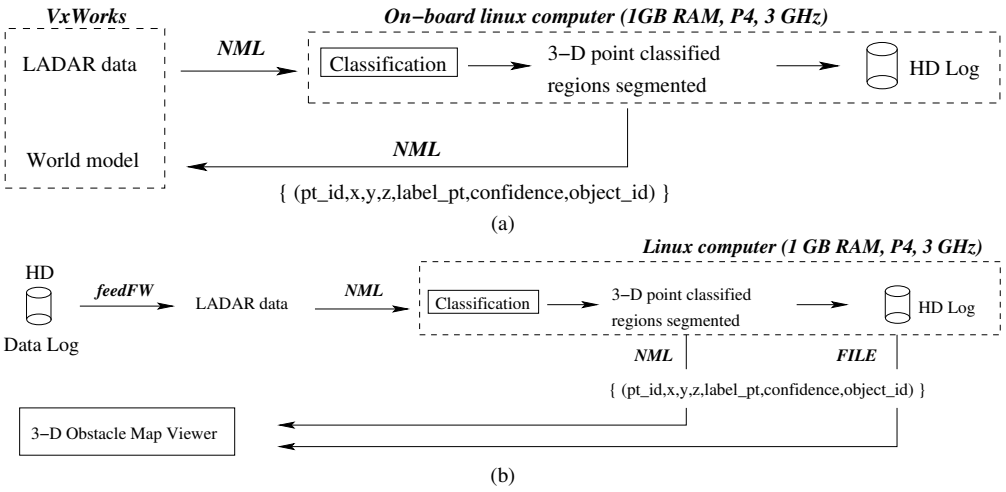


Figure 13: Data flow diagram. (a) Live data processing. (b) Playback data processing

of classification performances compared to the gain in processing time. In the worst case scenario, the classification error rate increases by 15% but the processing time is reduced by a factor of ten.

We also use two other parameters that indicate whether a prototype point should be classified again or not. The following tests are performed after classification for each prototype point:

- m: The number of times each prototype point has been classified. If $m > m_{max}$, the point is never classified again. Typically, a value of $m_{max} = 10$ is used. This prevents performing unnecessary classification if the robot is standing still.
- r: The likelihood ratio. If k_1 satisfies Equation 3 and k_2 represents the second most likely class, then, from Equation 2,

$$r = \frac{p(S|M^{k_1})}{p(S|M^{k_2})}$$

  If $r > r_{max}$, the current classification is considered accurate and the prototype point is marked to prevent further processing, although it is still updated by new points. Intuitively, this corresponds to the moment when the classifier is sufficiently confident about the current classification result. Throughout the experiments, $r_{max} = 2$ unless otherwise reported.

5.3 Interface

The architecture of the robot is the NIST 4D/RCS (Albus, 2002b) and the communication between processes is performed using the Neutral Message Language (NML) (Gazi, 2001). The ladar data is stored and continuously updated in an NML buffer, on one of the robot boards running VxWorks. Our code runs on a Linux computer, either a VME board part of the robot computer boards, or a stand-alone computer that is fitted inside the robot, and it communicates with the robot using NML buffers via Ethernet. Currently, a computer with a Pentium IV processor at 3 GHz, with 1 GB of RAM is used.

Playback tools that read laser data in a native file format and put them into a NML buffer were first developed. The code can then read from the NML buffer, incorporate the data, classify it, and send the results into another NML buffer. This last NML buffer is used by other robot processes to update the obstacle map or perform higher level scene interpretation. There is no additional effort to use the same code on-board of the robot. Only one configuration file, containing information about the shared memory buffer (name, computer host), needs to be updated. Figure 13 illustrates the resemblance between the data flow for the playback and live data processing configuration.

The classification results are sent back to the vehicle navigation systems to be incorporated into an obstacle map and to be used by other sub-systems such as one evaluating the mobility of the terrain based on the ground surface recovered. Such integration is underway.

6 On-line classification field tests results

Since 2003, a series of field tests were performed with a ground mobile robot to demonstrate the feasibility of our approach. This section first presents the vehicle and the test range used for experimentation. It then shows classification and high-level scene modelling results obtained by running the algorithms on the vehicle.

6.1 Unmanned vehicle and test area

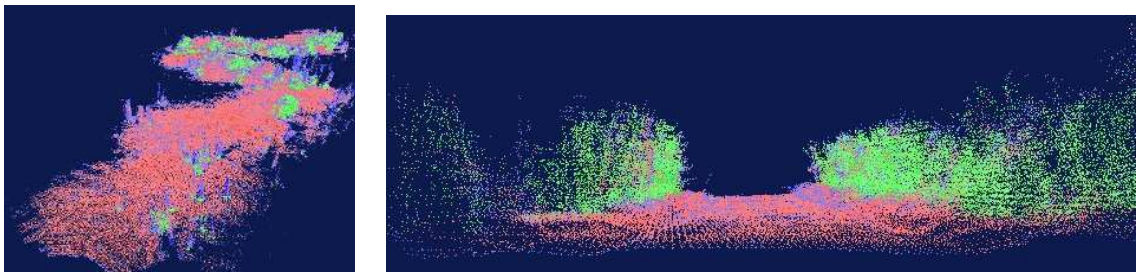
As mentioned earlier, our goal is to have the method presented above running on-board the eXperimental Unmanned Vehicle (XUV) while the robot is navigating in natural environments. A similar platform was used in the Demo-III program (Albus, 2002a). The robot is equipped with the GDRS mobility laser. This rugged laser radar provides range images of 180×32 pixels resolution at 20 Hz, with a maximum range up to 80 m. The laser is mounted on a turret controlled by the navigation system to build a terrain model used for local obstacle avoidance. Additional information on the laser can be found in (Shneier, 2003).

Field tests were conducted in Central Pennsylvania. The terrain is several square kilometers in size and includes various natural features such as open space meadows, wooded areas, rough bare ground and ponds, and is traversed by a network of trails. The terrain elevation varies significantly between different areas. In total, over half a dozen field tests were conducted over a two-year period.

6.2 Terrain classification

In Figure 14, two examples of terrain classification are shown. In the first case, the robot traveled below the tree canopy along a 200 m long trajectory at a speed between 1.5 and 2.0 m/s. In the second case, the robot is on a trail bordered by foliage and moves at a similar speed.

For each prototype point in the data structure, the time between its creation and its classification is recorded, and the results are presented in cumulative histograms (see Section 6.3.2 for a more detailed explanation). Approximately 90% of the prototype points are classified less than 1 second after their creation.



(a) (b)

Figure 14: On-board terrain classification. (a) From a wooded area. (b) From a trail. See footnote page 6 for color code.

6.3 Comparison against batch processing

6.3.1 Classification accuracy

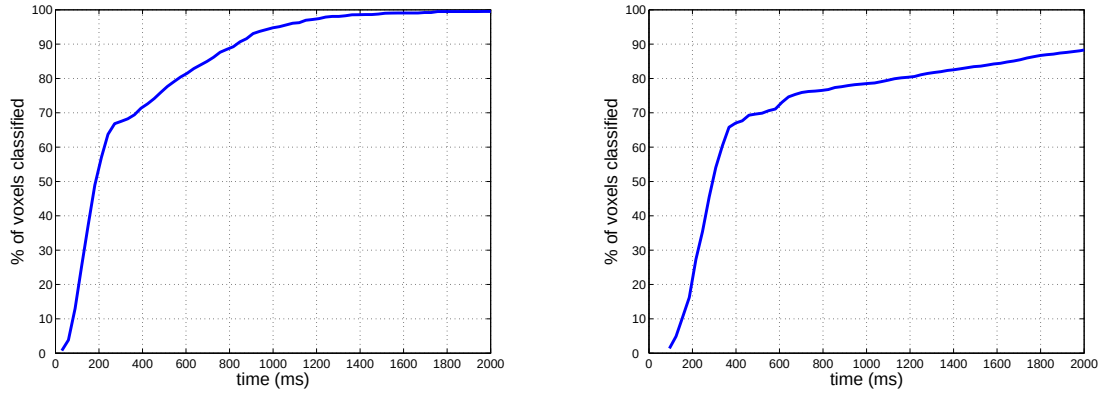
A more complete model can be constructed as data is accumulated over time. For example, previously occluded areas get filled in as they appear in the robot's field of view due to the vehicle motion. As a consequence, the occupancy of any given point varies over time, changing the saliency features. Thus, classification results obtained from partial versus full model might differ. In order to evaluate the difference, data was recorded while the robot was driving through an outdoor scene. We used the playback data processing data flow (see Figure 13) because it allows the execution of different experiments on the same data. By comparing the classification results point-wise, we observed a difference up to 15% between the live and static processing.

6.3.2 On-line Timing

It is generally hard to quantify timing performances of 3-D processing techniques for perception, because it depends on a variety of parameters, such as the sensor's characteristics, the area of interest processed, the speed of the robot, data density and distribution, and others, such as those presented in Section 5.2.3. The traditional way of reporting the number of processed prototype points per second is insufficient, as this does not take the dynamic nature of the process into account.

We introduce a different way of reporting timing performance, which is more appropriate for prototype point-based perception approaches. The idea is to compute, for each prototype point, the delay between the time of its creation (the first time a point in inserted in that prototype point), and the time at which its class is known with sufficient confidence. By analogy, we are computing the delay between the time the robot first "sees" an object and the time it "knows" what type of object it is.

Figure 15 shows an example of cumulative histograms obtained by computing the time between prototype point creation and classification, for two different sets of parameters. The faster the curve reaches 1 (100% of prototype points classified) the better. This data was obtained by driving the robot over 10 m in a rough forest environment, at walking speed. As illustrated in Figure 15-(a), 90% of the prototype points are classified at most 847 ms after their creation. In Figure 15-(b), the increase in m_{max} (maximum number of times a prototype point is to be classified) cause the first 90% of prototype points to be classified within at most 2223 ms after their creation. In the figures, r_{max} is the maximum likelihood ratio.

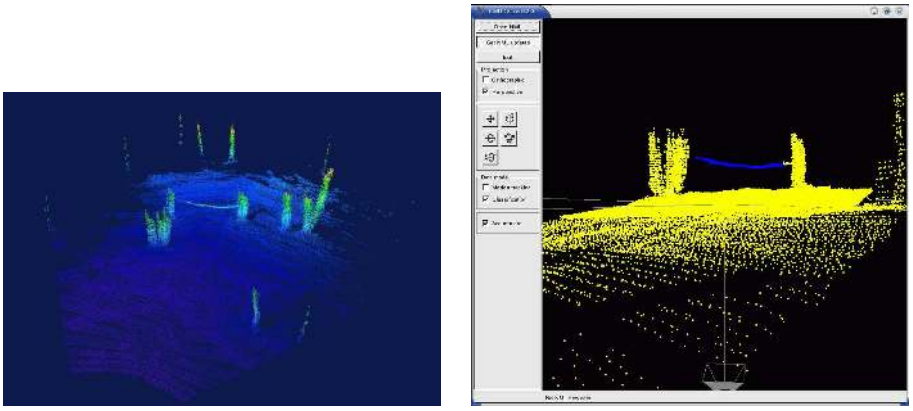


(a) (b)

Figure 15: Cumulative histograms of the time between voxel creation and classification obtained using playback. The same dataset is used in the two figures. (a) Using parameters $r_{max} = 2$ and $m_{max} = 5$ (b) $r_{max} = 2$ and $m_{max} = 10$ (see Section 5.2.3).

6.4 Wire segmentation

As explained in Section 4.3.2, wire segmentation is a useful application and it has been tested on-board a ground vehicle extensively. Figure 16-(a) shows a typical scene used for testing. In this example, a wire is hung between two trees at a height of approximately 2 m. The robot is then teleoperated to drive 10 m towards the wire at a slow walking pace and stop 2 m in front of it. While driving, the classification is performed, and the wire extraction process is executed afterwards. A live display application is used to visualize the reconstructed 3-D points. The points are colored in blue if part of a wire, and in yellow otherwise. Figure 16-(b) shows an actual screen capture of the live display. The wire is correctly segmented from the rest of the scene.



(a) (b)

Figure 16: Wire segmentation. (a) Accumulated 3-D data, color coded by elevation. With in red (or ligh grey) high elevation and in blue (or darker grey) low elevation. (b) Live display screen capture. Wire points are colored in blue (or dark grey), the non-wire points in yellow (or light grey).

6.5 High-level scene modelling

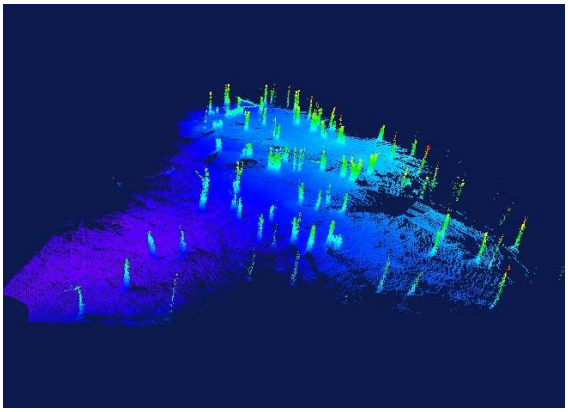
The high-level scene modelling step reported in Section 3.5 has also been implemented and tested on-board the vehicle, and Figure 17 reports results from these experiments. In this test, some wires are hung between trees at an average height of 2 m. Thicker branches are also present at the same height. The robot is teleoperated in an alpha-shape loop in the woods. The classification and high-level scene extraction are performed on-line, as the robot was driving at a slow walking speed. Although all 3-D points are recorded, the processing is done only on a 20×20 m area of interest in front of the robot, which explains the missing data in Figures 17-(b) and (c).

Figure 17-(f) and (g) illustrates how our approach can discriminate between branches and wires based on object diameter. In this example, a wire is hung between two trees (on the right hand-side of the figures), close to a horizontal branch (on the left hand-side). In the model, the branch appears as a horizontal yellow cylinder and the wire is the blue cylinder, which confirms the segmentation is correct. Problems such as overlapping primitives and better geometric fitting are subject of our current research work.

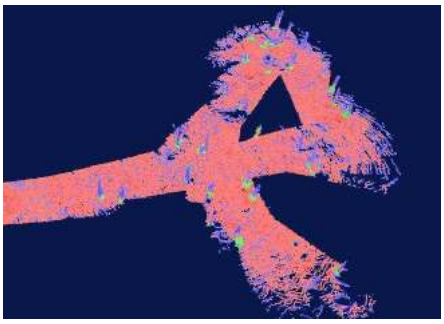
7 Discussion

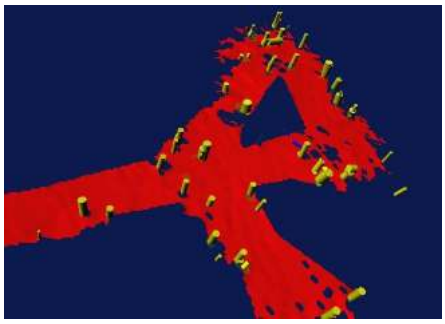
Until now, this paper presented methods that were implemented and tested on-board an unmanned ground vehicle. We now report on many of the on-going efforts to improve some of the fundamental issues previously described.

Data structures As presented in Section 3.1, the classification technique relies on the computation of local features defined over a neighborhood of fixed size. Since data is voxelized and point density is sufficiently high, a large majority of these neighborhoods overlap. To take advantage of this situation, we have introduced in (Lalonde, 2005b) a data structure that allows efficient range search operations by reusing pre-computed partial results. In off-line experiments,

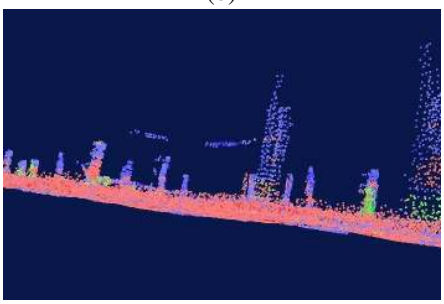


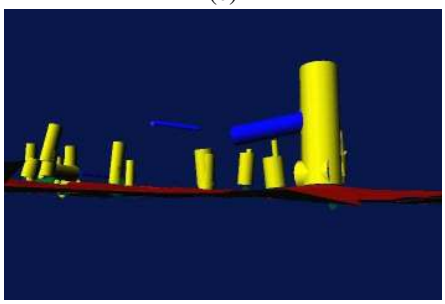
(a)



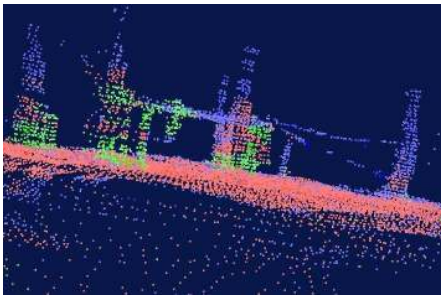


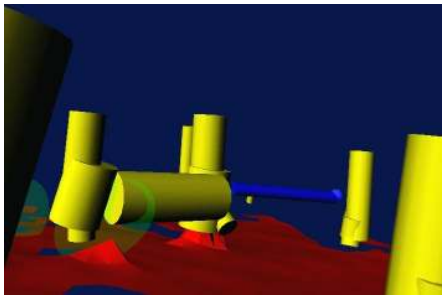
(b) (c)





(d) (e)





(f) (g)

Figure 17: Results of high-level scene modelling, on-board the vehicle. (a) Raw 3-D points, color-coded by elevation. With in red (or ligh grey) high elevation and in blue (or darker grey) low elevation. (b/d/f) Classification results, see footnote page 6 for color code. (c/e/g) Corresponding scene model. Yellow (or light grey) cylinders represent tree trunks, blue (or darker grey) cylinders represent wires, the ground is modeled as a red (or dark grey) mesh, and vegetation as semi-transparent green (light grey) ellipsoids.

we have reported a decrease of up to 400% in saliency computation time over the implementation presented in this paper (see Table 8). This technique is currently being implemented for on-board testing.

Automatic scale selection Although it allows fast computation, our approach suffers from the fact that the size of the support region (scale) is kept to a fixed value for each point in the dataset. The choice of scale is a compromise between noise sensibility (small value) and the ability to capture relevant details in the scene (optimal value), without encompassing too much clutter (too large of a value). The value of 0.4 m for the radius used in the experiments (c.f. Section 3.1) was shown experimentally to yield the best classification results on average over a large data set from a variety of terrains.

In (Lalonde, 2005a), we have introduced a scale selection method that automatically determines the best support region size for each point independently. It iteratively computes local approximations of the curvature, density and noise level, then estimates the scale based on an equation derived from an eigen-analysis of the scatter matrix. Currently, the approach is designed for surfaces only, and we have reported increase in classification accuracy of 30% for the previously misclassified points. We are now interested in extending this approach to handle the scatter and linear classes, and are looking at ways to port such approach on-board the ground vehicle.

Context The ad hoc filtering techniques presented in Section 3.3 were used as a simple and fast way to use context to improve classification results. The advantage is that they can easily be used for on-line computation. However, they are not based on any theoretical guarantees and need the manual adjustment of several thresholds.

We propose to model the relationship between features using the Markov Random Field (MRF) framework (Li, 1995), which enforces spatial consistency, or the preference that neighboring 3-D points have the same label. This technique iteratively builds a graph connecting neighboring points and assign weights to the edges such that the minimum cut will correspond to the minimum of an energy function (Kolmogorov, 2004). This technique is compared against the one introduced in Section 3.3, using the data from the example in Figure 11. The classification accuracy is 55.82% without filtering, 61.81 % with simple filtering, and 71.86 % with MRF filtering.

Although it yields better accuracy and is based on strong theoretical background, MRF filtering is currently unsuitable for on-line processing because it requires time-consuming operations, such as graph building and min-cut, to be run multiple times. We are now addressing this problem.

Features The second-order features used to describe the local neighborhood of each point are sometimes insufficient and cannot model correctly some of the true underlying geometric structure. This results in classification errors, notably at edges and junctions (c.f. Section 4.3). The use of additional features could also help train a more discriminative classifier. For example, first-order features used by Medioni in (Medioni, 2000) are currently being integrated.

Learning algorithm The current approach to perform classification is not scalable to a higher number of features, because EM typically experiences convergence issues as the number of parameters to estimate grows. Alternative methods based on Support-Vector Machine are also being explored.

8 Summary and future work

In this paper, we present a method to perform 3-D data segmentation for terrain classification in vegetated environment. To our knowledge, it is the first time that full 3-D data processing (from data acquisition to results representation) is performed live on-board an unmanned vehicle. Previous approaches have certainly used 3-d sensors (stereo, laser) but have processed the data sequentially in the range image plane and then projected them into a 2D-1/2 representation. Instead, we maintain all the processing in the 3-D space of accumulated data.

To summarize, our method uses local point distribution statistics to produce saliency features that capture the surface-ness, linear-ness and scatter-ness of local area. Statistical classification techniques are used to capture the variability of the scenes and the sensor characteristics (scanning pattern, range resolution, noise level). A Gaussian mixture model is then fitted to a training data set and this parametric model is used to perform Bayesian classification. This approach is validated using data from static Z+F, SICK and Riegl lasers. The implementation of our approach on-board an autonomous mobile robot, the DEMO-III XUV, was presented. Finally, off-line batch and on-board real-time results were used to demonstrate the versatility but also the limitations of the approach and its current implementation.

Some of limitations are currently addressed such as automatic scale selection (Lalonde, 2005a), efficient data structure design to speed-up computations (Lalonde, 2005b) and point cloud filtering to remove laser artifacts (Tuley, 2005). But those approaches have not yet been ported on-board the vehicle. On-going work also includes semantic interpretation of segmented point cloud to discriminate wires from branches, high level scene modelling for efficient environment representation, context modelling to capture more complex scene geometry, and detection and segmentation of complex porous structures (Vandapel, 2004b). Another aspect of the on-going work is to tie the classification results to other existing modules on-board the vehicle such as mobility evaluation or obstacle detection.

Acknowledgments

This work was conducted through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-209912. We would like to thank General Dynamics Robotics System for their support.

Bibliography

(Anguelov, 2005) Anguelov, D., Taskar, B., Chatalashev, V., Koller, D., Gupta, D., Heitz, G. & Ng, A. (2005, June). Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. Paper presented at the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA.

(Albus, 2002a) Albus, J, Murphy K., Lacaze A., Legowik S, Balakirsky S., Hong T., Shneier M. & Messina A. (2002, Oct.). 4D/RCS sensory processing and world modeling on the Demo III experimental unmanned ground vehicles. Paper presented at the International Symposium on Intelligent Control, Vancouver, BC, Canada.

(Albus,2002b) Albus, J. et al. (2002). 4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle System. N.I.S.T (Tech. Rep. 6910). Gaithersburg, MD, USA: National Institute of Standards and Technology.

(Bornstein, 2003) Bornstein, J. & Shoemaker, C. (2003, April). Army ground robotics research program. Paper presented at the SPIE conference on Unmanned Ground Vehicle Technology V, Orlando, FL, USA.

(Bilmes,1997) Bilmes, J. (1997). A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. (Tech. Rep. ICSI-TR-97-021). Berkeley, CA, USA: The International Computer Science Institute, University of Berkeley, Technical Report .

(Bellutta, 2000) Bellutta, P., Matthies, L., Owens, K. & Rankin, A. (2000, Oct.). Terrain Perception for DEMO III. Paper presented at the IEEE Intelligent Vehicle Symposium (IVS'03), Dearborn, MI, USA.

(Castano, 2003) Castano, A. and Matthies, L. (2003, Sept.). Foliage Discrimination using a Rotating Ladar. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'03), Taipei, Taiwan.

(Dima, 2004) Dima, C., Vandapel, N. & Hebert, M. (2004, April). Classifier Fusion for Outdoor Obstacle Detection. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'04), New Orleans, LA, USA.

(Duda, 2000) Duda, R., Hart, P. and Stork, D. (2000). Pattern Classification. Wiley Interscience.

(Gazi, 2001) Gazi, V., Moore, M., Passion, K., Shackleford, W., Proctor, F., & Albus, J. (2001) The RCS Handbook: Tools for Real Time Control Systems Software Development. Wiley-Interscience.

(Goldberg, 2002) Goldberg, S., Maimone, M., & Matthies, L. (2002, March). Stereo Vision and Rover Navigation Software for Planetary Exploration. Paper presented at the IEEE Aerospace Conference, Big Sky, MT, USA.

(Hebert, 2003) Hebert, M. & Vandapel, N. (2003). Terrain Classification Techniques from Ladar Data for Autonomous Navigation. Paper presented at the Collaborative Technology Alliances Conference, Washington, DC, USA.

(Hong, 2002) Hong, T., Rasmussen, C., Chang, T., & Shneier. (2002, May). Fusing Ladar and Color Image Information for Mobile Robot Feature Detection and Tracking. Paper presented at the International Conference on Intelligent Autonomous Systems, Washington, DC, USA.

(Huang, 2000) Huang, J., Lee, A. & Mumford, D. (2000, June). Statistics of Range Images. Paper presented at the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'00), Los Alamitos, CA, USA.

(Kelly, 2004) Kelly, A., Amidi, O., Bode, M., Happold, M. , Herman, H., Pilarski, T., Rander, P., Stentz, A., Vallidis, N. & Warner, R. (2004, June). Toward Reliable Off-Road Autonomous Vehicles Operating In Challenging Environments. Paper presented at the International Symposium on Experimental Robotics (ISER'04), Singapore.

(Kolmogorov, 2004) Kolmogorov, V. & R. Zabih. (2004). What energy functions can be minimized using graph cuts? IEEE Transaction on Pattern Analysis and Machine Intelligence, 2004, 26(2), 147-59.

(Lacaze, 2002) Lacaze, A., Murphy, K. & Delgiornio, M. (2002, July). Autonomous Mobility for the Demo III Experimental Unmanned Vehicles. Paper presented at the AUVSI Conference, Orlando, FL, USA.

(Lalonde, 2005a) Lalonde, J-F., Unnikrishnan, R, Vandapel, N & Hebert, M. (2005, June). Scale Selection for Classification of Point-sampled 3-D Surfaces. Paper presented at the IEEE 3-D Digital Imaging and Modeling (3DIM'05), Ottawa, Ontario Canada .

(Lalonde, 2005b) Lalonde, J-F., Vandapel, N. & Hebert, M. (2005, June). Data Structure for Efficient Processing in 3-D. Paper presented at the Robotics: Science and Systems Conference, Cambridge, MA, USA.

(Langer, 2000) Langer, D., Mettenleiter, M., Hrtl, F. & Frhlich, C. (2000). Imaging Ladar for 3-D Surveying and CAD Modeling of Real World Environments. International Journal of Robotics Research, 19(11), 1075-1088.

(Li, 1995) Li, S. (1995). Markov Random Field Modeling in Computer Vision. Springer-Verlag.

(Lukacs,1998) Lukacs,G., Martin, R., Marshall, D. (1998, June). Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. Paper presented at the European Conference on Computer Vision (ECCV'98), Freiburg, Germany.

(Macedo, 2000) Macedo, J, Manduchi, R. & Matthies, L. (2000, Dec.). Ladar-based discrimination of Grass from Obstacle for autonomous navigation. Paper presented at the International Symposium on Experimental Robotics (ISER'00), Honolulu, Hawaii, USA.

(Manduchi, 2005) Manduchi, R., Castano, A., Talukder, A. & Matthies, L. (2005). Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. Autonomous Robots, 18(1), 81-102.

(Matthies, 2003) Matthies, L., Bellutta, P. & McHenry, M. (2003, April). Detecting water hazards for autonomous off-road navigation. Paper presented at the SPIE Conference on Unmanned Ground Vehicle Technology, Orlando, FL, USA.

(Medioni, 2000) Medioni, G., Lee, M. & Tang, C. (2000). A Computational Framework for Segmentation and Grouping. Elsevier.

(Miller, 2002) Miller, J.R. (2002). A 3D Color Terrain Modeling System for Small Autonomous Helicopters. Ph.D thesis, Carnegie Mellon University, The Robotics Institute, Pittsburgh, PA, USA.

(Rasmussen, 2001) Rasmussen, C. (2001, Dec.). Laser Range-, Color-, and Texture-based Classifiers for Segmenting Marginal Roads. Paper presented at the IEEE International Conference On Computer Vision and Pattern Recognition (CVPR'01), Hawaii, USA.

(Rasmussen, 2002) Rasmussen, C. (2002, May). Combining Laser Range, Color and Texture Cues for Autonomous Road Following. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'02), Washington, DC, USA.

(Sithole, 2004) Sithole, G. & Vosselman, G. (2004). Experimental comparison of filter algorithms for bare-Earth extraction from airborne laser scanning point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 59, 85-101.

(Shneier, 2003) Shneier, N., Chang, T., Hong, T., Cheok, G., Scott, H., Legowik, S. & Lytle, A. (2003,April). A Repository of Sensor Data for Autonomous Driving Research. Paper presented at the SPIE Unmanned Ground Vehicle Technology V, Orlando, FL, USA.

(Tuley, 2005) Tuley, J., Vandapel, N. & Hebert, M. (2005, April). Analysis and Removal of Artifacts in 3-D LADAR Data. . Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona,

Spain.

(Vandapel, 2003) Vandapel, N. & Hebert, M. (2003, July). Experimental Results in Using Aerial LADAR Data for Mobile Robot Navigation. Paper presented at the International Conference on Field and Service Robotics (FSR'03), Lake Yamanakako, Japan.

(Vandapel, 2004a) Vandapel, N., Huber, D.F., Kapuria, A. & Hebert, M. (2004, April). Natural Terrain Classification using 3-D Ladar Data. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'04), New Orleans, LA, USA.

(Vandapel, 2004b) Vandapel, N. & Hebert, M. (2004, Sept.). Finding Organized Structures in 3-D Ladar Data. Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04), Sendai, Japan.

(Vandapel, 2005). Vandapel, N., Kuffner, J. & Amidi, O. (2005, April). Planning 3-D Path Networks in Unstructured Environments. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain.

(Wellington, 2003) Wellington, C. & Stentz, A. (2003, July). Learning prediction of the load-bearing surface for autonomous rough-terrain navigation in vegetation. Paper presented at the International Conference on Field and Service Robotics (FSR'03), Lake Yamanakako, Japan.

(Wolf, 2005) Wolf, D.F., Sukhatme, G., Fox, D. & Burgard, W. (2005, April). Autonomous Terrain Mapping and Classification Using Hidden Markov Models. Paper presented at the IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain.