Terrain classification in complex 3D outdoor environments

Àngel Santamaria-Navarro, Ernesto H. Teniente, Martí Morta and Juan Andrade-Cetto
Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens Artigas 4-6, Barcelona 08028, Spain
{asantamaria, ehomar, mmorta, cetto}@iri.upc.edu

Abstract

This paper presents two techniques to detect and classify navigable terrain in complex 3D environments. The first method is a low level on-line mechanism aimed at detecting obstacles and holes at a fast frame rate using a time-of-flight camera as the main sensor. The second technique is a high-level off-line classification mechanism that learns traversable regions from larger 3D point clouds acquired with a laser range scanner. We approach the problem using Gaussian Processes as a regression tool, in which the terrain parameters are learned, and also for classification, using samples from traversed areas to build the traversable terrain class. The two methods are compared against unsupervised classification, and sample trajectories are generated in the classified areas using a non-holonomic path planner. We show results of both the low-level and the high-level terrain classification approaches in simulations and in real-time navigation experiments using a Segway RMP400 robot.

1 Introduction

A large number of mobile robot applications are tailored to human environments, and the robots used are expected to navigate and move efficiently on them. An elementary capability for efficient navigation in such environments is obstacle avoidance. In planar indoor and outdoor domains, 2D range sensing suffices to guarantee adequate obstacle avoidance, and a large number of deployed systems rely purely on 2D range sensing for safe navigation alongside humans. In less structured 3D settings however, 2D sensing does not suffice to guarantee safe and robust navigation, and 3D sensing alternatives are favored, such as stereo vision or 3D laser scanning.

Time-of-flight (ToF) cameras combine the advantages of image-based sensing and range sensing as they provide registered range and intensity values for each pixel. Moreover, ToF cameras capture range data at high frame rates overcoming the need to aggregate range measurements typical of other 3D scanning devices (Ortega and Andrade-Cetto, 2011). However, since ToF cameras are active infrared illumination devices, they have not been used prominently for outdoor robot navigation due to sun saturation effects. As new more powerful devices enter the marketplace, this situation is expected to change.

In this paper we exploit the advantages of ToF sensing and report the use of a photonic mixer device (PMD) ToF camera for robust obstacle and hole detection for real-time outdoor mobile robot navigation. The technique includes several sensor data processing algorithms combined with a custom-developed short-term planner to account for the non-holonomic constraints of the platform used. The camera is located in the front of the robot and facing downwards as shown in Fig. 1.

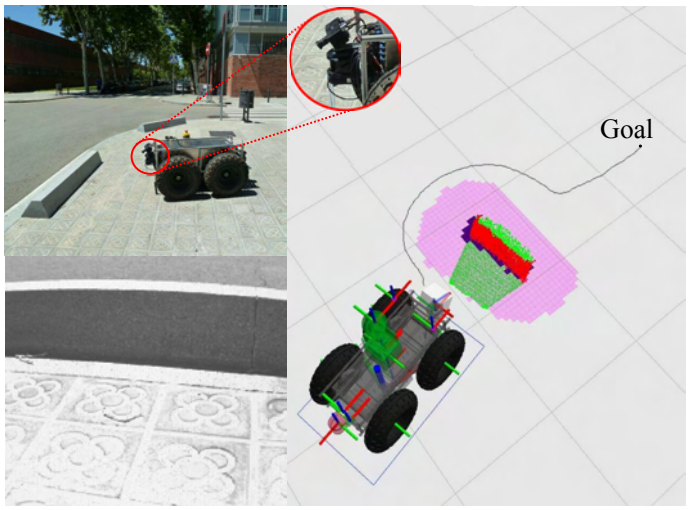


Figure 1: Obstacle detection with a ToF camera. Top left: The ToF camera is located in the frontal part of the robot and facing downwards. Bottom left: Intensity image of the scene as observed by the ToF camera. Right: The detected obstacle is shown in red, and the traversable regions are shown in green. The pink area represents an inflation radius around the detected obstacle, used to compute the local path to the goal as shown in black.

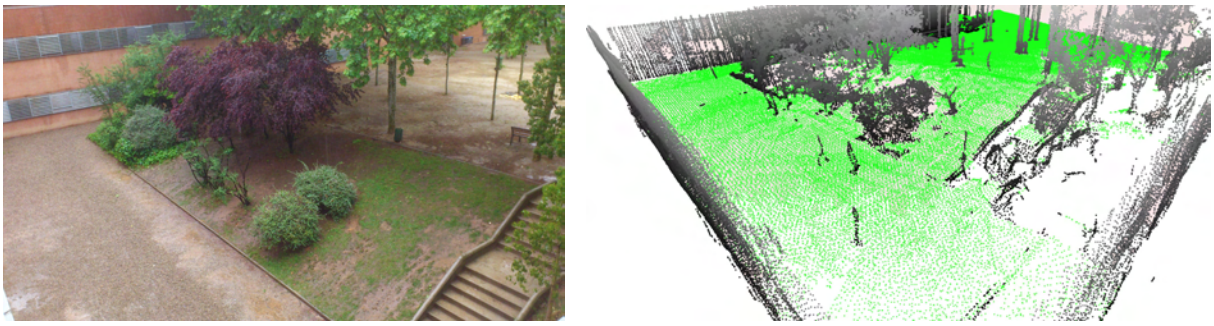
Once a low-level obstacle avoidance mechanism at high frame rate is in place, it is desirable to have a high-level technique that can plan longer paths to a desired goal. This requires not only the acquisition of dense maps, but also a classification strategy on these maps. To build the maps, we use our method in (Valencia et al., 2009). In this paper, we present such classification scheme.

State-of-the art terrain classification methods use 3D laser range finder data at impressively rich point cloud density, and extract features from these point clouds (Douillard et al., 2011). Such features are then used to build traversable and non-traversable classes, usually through parametric classification. Feature thresholding is a difficult task that sometimes requires human intervention (Stoyanov et al., 2010; Triebel et al., 2006; Joho et al., 2007). However, learning algorithms with manual labeling of data (Lalonde et al., 2006) can also be used.

In this paper we present a pair of methods to attain high-level off-line classification of traversable areas, in which training data is acquired automatically from navigation sequences. Our experiments show f-scores –weighted average of precision and recall values– as high as 0.98 for the classification of traversable and non-traversable regions.

To motivate the reader, Fig. 2 shows the type of terrain our method is able to handle. Using only some of the traversed areas as positive training samples, our algorithms successfully classified the rest of the traversable terrain and our robot was able to climb the steep grass incline next to the stairs. This is in contrast to other techniques which search for global planarity constraints or other features not in accordance with the training data.

The originality in this paper is two-fold. On the one hand, we present a study in the use of ToF sensing for outdoor obstacle avoidance, and a technique to smooth the local obstacle avoidance paths produced with an A* planning algorithm using cubic B-splines. Secondly, we present an off-line GP-based classification mechanism to detect traversable regions purely from traversed positive samples. The classification results are then used to plan global paths for the robot.



(a) (b)

Figure 2: Non-parametric classification of complex terrain. a) A picture of the Facultat de Matemàtiques i Estadística (FME) patio. b) Classification results. Traversable areas in green, and obstacles in height-valued gray. The positive samples in the training sequence used to produce this result included the points at the robot footprint that were acquired while climbing the slope next to the stairs. Consequently, the classification technique was able to correctly label such scenario.

2 Related work

In contrast to indoor robot navigation, outdoor mobile robots need to be equipped with 3D sensing capabilities (Sanfeliu et al., 2010). The state of the art in high speed 3D sensing for mobile robots is achieved with the Velodyne sensor. This device, in its HDL32 and HDL64 versions is tailored to scan wide areas at high frame rates, and is commonplace in research in autonomous vehicles (Leonard et al., 2008). The HDL64 device provides point clouds with over 1.3M points per second on a 360 degree horizontal view of the scene with a limited vertical field of view of 26.8 degrees and up to 120 meters (Spinello et al., 2011).

Laser-based 3D sensors are prone to transformation errors at high speed due to the need for motion compensation. A high speed alternative to laser scanning for 3D imaging is the use of CCD or CMOS cameras and stereo triangulation, and more recently, the use of 3D time-of-flight cameras. ToF cameras measure both intensity and range by illuminating the scene with an infrared wave form and by demodulating the amount of light reflected by the scene onto the sensor (Foix et al., 2011). These devices have been modestly used for robot navigation. The first approaches in indoor robot navigation with a prototype of the SR-2 Swissranger (Weingarten et al., 2004; Blanc et al., 2004) motivate their use for obstacle avoidance, and point out that camera calibration and data preprocessing steps were necessary to get stable range data from the sensor. Furthermore, ToF cameras have been integrated with the Player middleware (Hussmann et al., 2009), and their signal cropped to be used as a 2D range sensor for indoor mapping (Almansa-Valverde et al., 2012), or only to enhance the results obtained with laser navigation (Yuan et al., 2009).

Real-time ToF 3D sensing is now commonplace, and also the calibration procedures have been refined (Fuchs and Hirzinger, 2008), to the point that ToF range images are used nowadays in the context of robotics, to create models of 3D rigid (Foix et al., 2010) and flexible objects (Alenyà et al., 2011), to fit conics for pipeline landmark detection (Thielemann et al., 2008) or to create high resolution range maps through fusion with stereo (Gandhi et al., 2012). In this paper we analyze the use of ToF imaging for fast obstacle detection and outdoor navigation.

Aside from reactive local navigation, we want to add to our system global path planning capabilities. To that end, we need to build larger maps, and to classify their terrain into traversable and non-traversable areas. Dense 3D mapping calls for special sensing modalities. The Velodyne is fast, but produces sparse point clouds. ToF cameras on the other hand have limited range. To accommodate for dense long range sensing, we have built instead a series of 3D scanners (Valencia et al., 2009; Teniente and Andrade-Cetto, 2013), and used them to build large and dense maps of the environment.

Terrain classification on 3D environments with non-flat surfaces has been an active research topic in mobile robotics. An early approach to the problem is the use of elevation maps (Hebert et al., 1989; Howard et al., 2006). This is a discrete representation of the terrain that stores surface height in its cells. Neighboring cells with a difference in height above a user-defined threshold are marked as non-traversable. In (Pfaff and Burgard, 2005) an extension of elevation maps is given that allows to deal with vertical and overhanging objects. These were further extended to represent multiple levels with the Muti-Level Surface Maps (MLS) (Triebel et al., 2006). And later, in (Joho et al., 2007), MLS were extended to include slope information to handle possible abysses and holes.

Terrain classification is approached with the aid of Markov random fields in (Häselich et al., 2011; Anguelov et al., 2005; Munoz et al., 2009), either using cameras or range data. All these mapping schemes store the 3D environment information in regularly spaced grids. Our technique does not treat annotated 2D maps, but rather full 3D data, allowing us to handle underpasses or tree shadows with ease. Another version of terrain classification that uses 3D data is through the use of 3D occupancy maps (Martin and Moravec, 1996). This technique however has large memory requirements. To deal with the memory requirements of 3D occupancy maps, one can resort to the use of variable resolution grids, such as octrees (Wurm et al., 2010; Stoyanov et al., 2010). Instead of learning the classification, (Pauling et al., 2009) segmented the point clouds through a graph-based ellipsoidal region-growing process using a minimum spanning-tree and two maximum edge-weight conditions, leading into discrete regions that represent obstacles in the environment. (Morton and Olson, 2011) classifies objects according to height-length-density parameters. The above-mentioned methods to terrain classification are parametric. They offer fast terrain segmentation, but generally it is difficult to find the rules which work for a variety of terrain types and dangerous areas like abyss.

Non-parametric approaches use some learning strategies that include the use of training data. Some recent methods use learning from demonstration (Silver et al., 2010), imitation learning (Silver et al., 2009), learning from proprioceptive measurements (Sofman et al., 2006; Angelova et al., 2007), or coordinate ascent learning to estimate terrain roughness (Stavens and Thrun, 2006). In (Lacroix et al., 1999) for instance, Bayesian classification is performed on elevation maps using stereo vision to compute occupancy maps. In a method similar to ours in spirit, (Kim et al., 2006) produce a grid with traversable regions from an autonomous data collection along the robot path. Their method extracts 13 different features from monocular images, and feeds them to a majority voting process to predict the traversability of each cell. Our method is simpler in that we use less features, and are not constrained to regularly spaced grid settings. Another non-grid based representation for 3D data is presented in (Lalonde et al., 2006). They perform terrain classification directly over the point clouds just as we do, learning the terrain feature distribution by fitting a Gaussian Mixture Model using the Expectation Maximization algorithm on a set of hand labeled training data. We resort instead to the use of Gaussian Processes for classification, and do so over automatically acquired positive training sequences.

Gaussian processes (GPs) (Rasmussen and Williams, 2005) have recently called attention in mobile robotics for classification (Murphy et al., 2012). Compared to a more traditional technique such as Support Vector Machines (SVM) GPs offer several advantages such as learning the kernel and regularization parameters, integrated feature selection, fully probabilistic predictions and interpretability. In comparison with Support Vector Regression (SVR), GPs take into account the uncertainties, predictive variances and the learning of hyper-parameters. Using GPs, (Karumanchi et al., 2010) proposed a mobility representation where in contrast of the traversability criterion, the maximum feasible speed is used to classify the world adding exteroceptive or proprioceptive states augmenting the GP input vector. An approach using GPs on two-dimensional range data to compute occupancy probability is presented in (O'Callaghan and Ramos, 2012). Another approach, (Murphy and Newman, 2010), takes advantage of GPs to build cost maps for path planning from overhead imagery. In (Vasudevan et al., 2009), the authors infer missing data in 3D range maps for open pit mines. And recently, GPs were used to segment point clouds in not one but multiple classes (Paul et al., 2012). In (Douillard et al., 2011), the Gaussian Process Incremental Sample Consensus (GP-INSAC) algorithm is presented, that consists of an iterative approach to probabilistic, continuous ground surface estimation, for sparse 3D data sets that are cluttered by non-ground objects. This approach however does not specifically address traversable region classification as presented in the next sections.

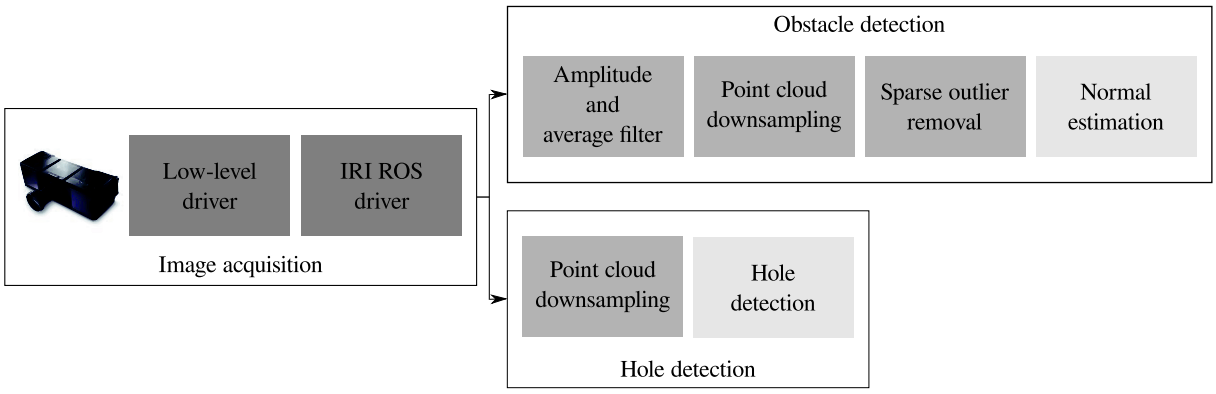


Figure 3: The ToF camera data passes through various low-level filters according to the task to be solved.

3 Obstacle detection with a time-of-flight camera

Time-of-flight cameras use infrared light to illuminate the scene, thus sunlight influences their distance measurements. For this reason, ToF devices require algorithms for suppression of background illumination (SBI) or an equivalent infrared suppression scheme if their use is intended for outdoor applications. (Piatti and Rinaudo, 2012) showed that the PMD CamCube 3.0 camera is more robust to sunlight than other ToF cameras. This camera is based on phase shift measurements and has a PMD 41k-S2 sensor of 200 x 200 pixels, with integrated SBI technology. For this reason, PMD CamCube 3.0 camera has been used for this work.

Our objective is to detect obstacles in front of the robot at a high frame rate and this is done by analyzing the planarity and orientation of patches in the point cloud. Depth measurements coming from the ToF camera are contaminated by both systematic and nonsystematic errors (Foix et al., 2011). Systematic errors can be dealt with calibration. For instance, the PMD camera used in this paper is claimed to work under sunlight mainly thanks to its powerful IR auto-illumination characteristic, but the camera integration time (IT) has to be adjusted for each particular illumination condition. As discussed in (May et al., 2006), the IT affects the amplitude and intensity of the data. As for the random non-systematic errors, these need to be removed with various filtering schemes. In this paper, we develop several low-level and morphological filters depending on the task to be solved. Without filtering, the ToF camera produces a very low signal to noise ratio. For obstacle detection, we implemented amplitude and noise reduction filters, downsampling, outlier removal and normal estimation. For hole detection, downsampling suffices. The source point cloud is passed through the above-mentioned filters ending with the estimation of local orientation at each point. See Fig. 3.

- *Amplitude filtering* is used to remove 3D points which have been read with a lower amplitude than a threshold. The amplitude has no specific units and it represents the amount of near infrared light reflected from the scene. Higher amplitude means more confidence in the measurement and thresholding it discards primarily data resulting from objects with low infrared reflectivity, far away distances, or from objects which are located at the peripheral area of the measurement volume due to inhomogeneous scene illumination. The ToF camera has limitations on surfaces with high or low reflection characteristics that pass undetected, by sensor saturations at sunlight, or by absorbing entirely the IR illuminant.

- *Average filtering* consists of an integration approximation by observing each pixel in the point cloud at consecutive time intervals and keeping the z distance average. This filter smoothes the output averaging a number of consecutive point clouds depending on the buffer size. The use of the average filter improves signal-to-noise ratio and although it removes temporary outliers it introduces a delay to detect obstacles.

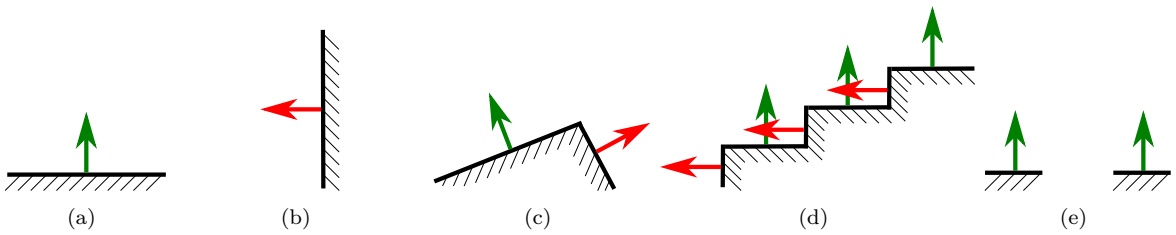


Figure 4: Classification of points into traversable (green) and non-traversable (red) areas according to their locally estimated normal orientation(a) ground, b) obstacle, c) slope, d) stairs, e) hole).

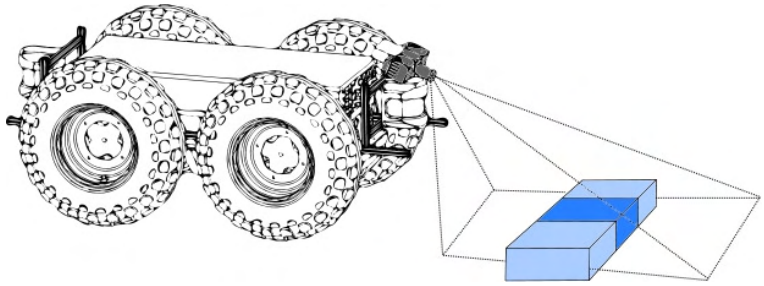


Figure 5: The area in front of the robot is divided into three regions, and the amount of data in each region is analyzed to evaluate the presence of a hole on the road.

- *Point cloud downsampling.* The density of the point cloud produces delays due to computation costs. Once the data has been filtered by amplitude and average, it is interesting to down-sample the point cloud mainly because in our application not all points add extra information to perform obstacle detection. To down-sample, we uniformly reduce the number of points considering an expected output resolution.

- *Sparse outlier removal* is based on the computation of the distribution of point neighbor distances in the input dataset. For each point, its mean squared distance to all its k neighbors is computed. By assuming that the resulting distribution is χ^2, all points whose mean squared distances are outside an interval defined by the point mean and standard deviation can be considered as outliers and trimmed from the dataset. Outlier removal is at the expense of filtering the boundary of step edges, which for our application is inconsequential.

- *Normal estimation.* We estimate the normal orientation of local planar patches at each point with the method in (Ortega et al., 2009). Traversable regions will have a local normal orientation nearly perpendicular to the robot footprint, whereas obstacles will be either non-flat, or have other orientations, see Fig. 4. In the special case of a gradual increase of ground inclination we added a safety condition comparing the absolute orientation of the planar patch with that of the 3D compass onboard the robot.

A robust navigation strategy must not only be able to detect obstacles efficiently, but also to detect apertures or holes in the terrain. No specific low-level processing of the data is required to do this but only to quantify the amount of data gathered from a particular region. To do this we divide the region in front of the robot in three areas as shown in Fig. 5, and analyze the amount of registered data in them. Finer subsampling can be used in this case since the structure of the data is not relevant for this task and the normals need not be computed. To interface with the navigation cost map, once a hole is detected, we generate a virtual point cloud simulating an obstacle in that region.

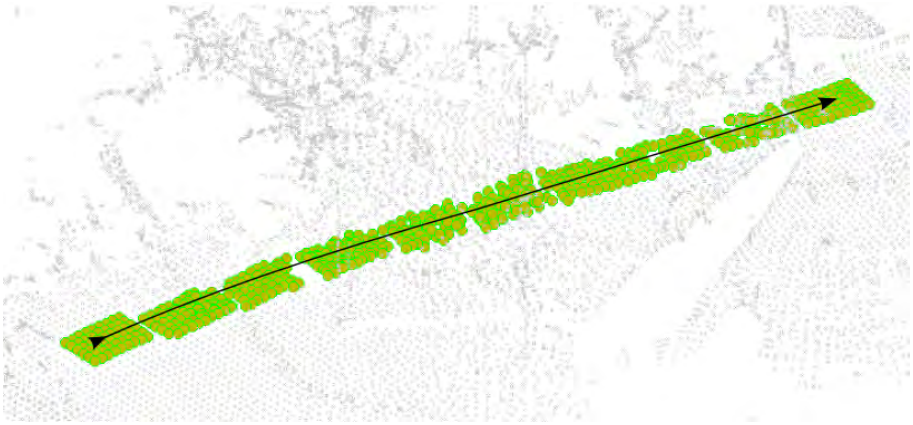


Figure 6: Example of the automatic data collection. In green the positive traversable samples along the robot's path footprint.

4 Gaussian processes for off-line terrain classification

We want to identify traversable areas in very complex terrain. This means that we must be able to deal with many terrain characteristics, such as slopes and bumps, vegetation, different terrain surfaces (e.g., soil, grass, concrete), etc. To build a map of the environment, we drive our mobile robot taking multiple dense 3D point clouds. Pose SLAM (Ila et al., 2010; Valencia et al., 2009) is used to register these point clouds together.

Initially we compute two features for each point in the map, slope and texture, using principal component analysis over a window of points centered at each query point. The eigenvector associated with the smallest eigenvalue defines the orientation of the local plane, and is used to compute plane slope f_s, with regards to the world reference frame. The smallest eigenvalue is used to describe the terrain roughness f_r.

We model the traversable class as a Gaussian process (GPs) (Rasmussen and Williams, 2005) and train a covariance function to estimate its non-parametric underlying distribution. The prior knowledge of the environment is the training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$, where $\mathbf{x}_i$ contains the slope and roughness input values for each point $\mathbf{p}_i$ in the map, and y_i is a scalar target value which is dependent of the problem to solve (regression or classification), e.g., the traversable and non traversable classification class labels. In our case, we take advantage of automatic data collection to build the training set, similar to (Kim et al., 2006). Since our robot trajectories were human-driven, they already satisfy any unmodeled traversability constraint. The points located below the robot footprint at each node in the Pose SLAM path are thus tagged as traversable. See Fig. 6. This collection method alleviates the hard work of manually labeling the training dataset. For each point in this set we compute our slope and texture features, and aggregate them into a positive feature vector $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] : \mathbf{x}_i = [f_r, f_s]$. Ideally, the training dataset would consist of all the observations made. However we subsample the data. Data reduction is important since the GP has worst case computational complexity $O(n^3)$, and memory complexity $O(n^2)$.

The problem of learning in GPs is exactly the problem of finding suitable properties for the covariance function that best models the training data $\mathcal{D}$. These properties, called the hyper-parameters, are adjustable variables that are learned from $\mathcal{D}$ and are used to do predictions for new inputs that we may have not seen in $\mathcal{D}$. There are numerous covariance functions (kernels) that can be used to model the relationship between the random variables corresponding to a given data. In our approach, we have used a neural network covariance defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \arcsin\left(\frac{2\tilde{\mathbf{x}}_i^T \Sigma \tilde{\mathbf{x}}_j}{\sqrt{(1+2\tilde{\mathbf{x}}_i^T \Sigma \tilde{\mathbf{x}}_i)(1+2\tilde{\mathbf{x}}_j^T \Sigma \tilde{\mathbf{x}}_j)}}\right), \quad (1)$$

where $\tilde{\mathbf{x}}_i = (1, \mathbf{x}_i)^T$ is the so-called feature augmented input vector (Rasmussen and Williams, 2005). We have set its hyper-paramteres $\Sigma = diag(\sigma_0^2, \sigma^2)$ as those fitting the squared expected value of the error function on the samples $\mathbf{x}$. σ_f^2 is the hyper-parameter signal variance, used to scale the correlation between points. The use of a neural network covariance kernel has been proved to be effective in handling discontinuous (rapidly changing) data, and this is the main reason why we think it is effective to model complex terrain data (Vasudevan et al., 2009).

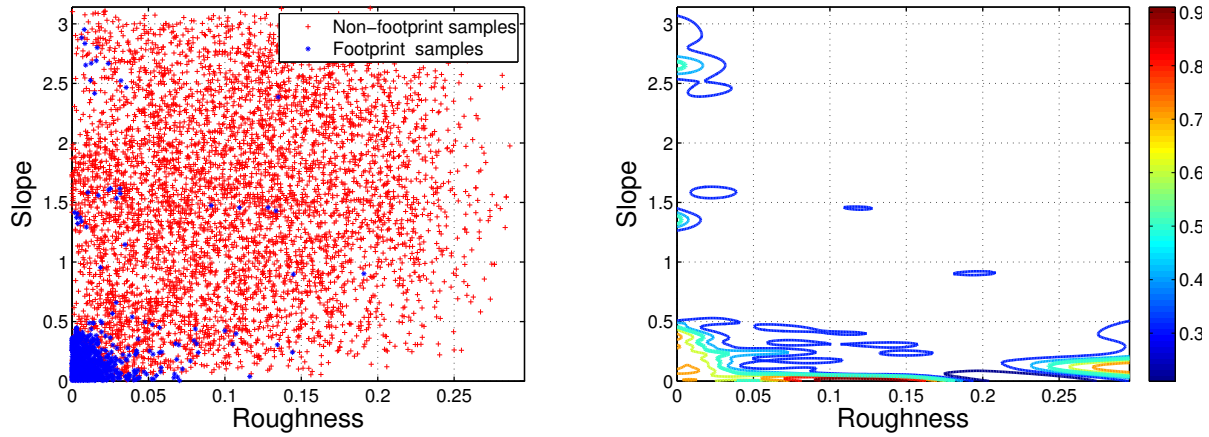
At this point we propose two different approaches to classify the points as traversable P_t or obstacles P_o. The first proposal employs GP regression and only traversable samples. The main advantage of using regression is to have less computational effort for the training and inference than GP classification, but with lower accuracy. In this case, the training dataset $\mathcal{D}$ will consist of a uniformly sampled and filtered version of $\mathbf{x}$ including only traversable points as in Fig. 6. We use a statistical density filter to remove spurious data which may affect the regression. The specification of the prior is important, because it fixes the properties of the functions considered for inference. In this case, we use a zero mean Gaussian prior (Rasmussen and Williams, 2005). The algorithm will produce a classification threshold $y = c$ such that traversable points are $P_t = \{\mathbf{p}_i | y_i \geq c\}$ and obstacle points are $P_o = \{\mathbf{p}_i | y_i < c\}$. We can use regression in our proposal since data dimensionality is small and we can assume continuity, and also because we only have two classes.

A second approach is to perform GP classification, which increases the computational load but we believe it offers a more accurate prediction for this problem, as shown in the experiments section. The training data set is established differently than for regression. Now the training dataset needs two different classes. The first class comprises the same labelled traversable features used during regression, which came from the robot footprint samples during manual robot motion. In this case, we do not further subsample the training set since the prediction of the GP classification is expected to remove spurious data during classification. We call the input data set in this class $\mathbf{x}_+$, and receives the label $y = +1$. The second class with label $y = -1$ is built by randomly sampling the remaining unlabeled points in $\mathbf{x}$. This set is called $\mathbf{x}_-$. Now we build the two-class training data set as $\mathcal{D} = \{(\mathbf{x}_+, \mathbf{x}_-), \mathbf{y}\}$. To obtain the traversable points we create a grid over the slope and roughness features. This grid is used to compute the log predictive probability curves with the learned hyper-parameters, see Fig. 7. It should be noted that the region of high likelihood for high roughness and low slope (on the bottom right of the Fig. 7) is largely underrepresented in the training set, meaning that the GP would classify samples in that region with large uncertainty values. In reality, since no points in the entire dataset satisfied such condition, it is very improbable that during execution, a point would be found to have such feature values and be misclassified. Traversable points will be those with a probability larger than a threshold p_t indicated by the likelihood curve that best fits $\mathbf{x}$, i.e., traversable points are $P_t = \{\mathbf{p}_i | \ln p \geq p_t\}$ and obstacle points are $P_o = \{\mathbf{p}_i | \ln p < p_t\}$.

Note that when no true negatives are available, as in our case, it is still possible to train only with positive samples during regression, or to randomly draw negative samples from the unlabeled data during classification. These two strategies are commonly used in learning, and are not unique to GPs, but are also applicable for SVM classification (Elkan and Nato, 2008) or with the SpyEM algorithm (Liu et al., 2002).

Once the point cloud is either regressed or classified, we propose a filtering step to remove isolated points on each class (Lalonde et al., 2006). The filter uses nearest neighbor information, discarding all points that have less than a selected number of neighbors in a search radius. Moreover, a statistical density filter is also applied to remove points in regions with low density.

Identifying borders is important to protect the robot integrity. Border points may be located near abysses, on stairs and on holes that normally are out of the sensor field of view. We define border points as $P_b = \{\mathbf{p}_i \in P_t | f_{b1} \geq (\mu(f_{b1}) + \kappa * \sigma(f_{b1})) \wedge f_{b2} \geq r \wedge f_{b3} > \beta\}$, where r indicates distance to the border, and β is the maximum angle interval for a no border point. Next in line, we remove P_b from P_t and add these points



(a) (b)

Figure 7: (a) Training dataset. (b) Likelihood curves for GP classification. In the color bar, the red color indicates the maximum likelihood for a point to be traversable.

to the obstacle set.

5 Robot navigation

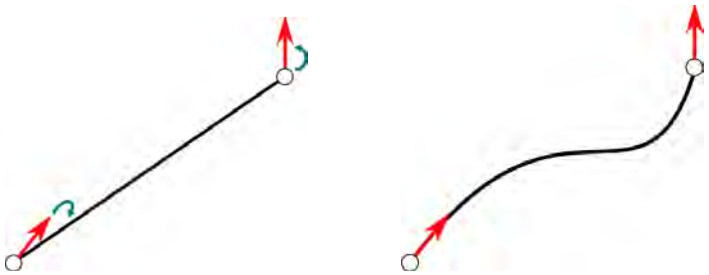
We have a two-level robot navigation scheme, in which a global plan is computed off-line, and a local navigation module is used to drive the robot during path execution. The global path planner is a modified A* algorithm that plans trajectories in the off-line segmented map. The local planner on the contrary is a simpler planer that plans small paths to avoid obstacles locally as observed by the ToF camera. Since the global planner was previously reported in (Teniente and Andrade-Cetto, 2013), it is not fully developed here. The main contribution of the section is on the use of a B-spline smoother in the local planner to account for the non-holonomic constraints of our vehicle. This two-level navigation scheme is explained next.

Once the terrain has been offline classified, we are now in the position to compute a global path to a desired destination that will be fed as waypoints to a lower level online robot navigation module. To this end we resort to a hybrid randomized A* method that can plan safe trajectories in rich complex 3D environments whilst guaranteeing reachability at a desired robot pose with significantly lower computation time than competing alternatives. The method is called HRA* and it has been shown to compare favorably against A*, RRT and RRT*, in terms of computation time, and generates significantly shorter paths than A* and RRT (Teniente and Andrade-Cetto, 2013).

This path planner incrementally builds a tree using the A* algorithm. However, it includes a hybrid cost policy to efficiently expand the search tree. The method combines random sampling from the continuous space of kinematically feasible motion commands with a cost to goal metric that also takes into account the vehicle non-holonomic constraints.

To speed up the node search, the method also includes heuristics to penalize node expansion near obstacles, with a penalty proportional to the inverse distance to collision; and to limit the number of explored nodes, the method book-keeps visited cells in the configuration space, and disallows node expansion at those configurations in the first full iteration of the algorithm.

Once this global plan has been computed, the trajectory is fed to the online navigation module consisting on a local plan to drive the robot from one waypoint to the next, and uses a low-level controller to follow it.



(a) (b)

Figure 8: Comparison between (a) straight and (b) B-spline paths. The path (a) includes in-place rotations which are avoided in (b) considering our skid steer platform.

Within this navigation scheme, obstacle data in the area close to the robot is assembled into a discretized two-dimensional structure, i.e., the cost map. The local planner is in charge of finding a suitable path to the next waypoint whilst avoiding large-valued cells in the cost map. The cost map is a 2D structure because the robot is constrained to drive on surface ground, although the underlying representation of the world actually consists of a 3D voxel grid. This 3D-occupancy grid is projected down into 2D and costs propagate outward as specified by a decay function as in (Marder-Eppstein et al., 2010).

Without the need to consider the specific hybrid cost policy and the heuristics of the underlying method of the global planner, we take advantage of a simpler method for the online navigation module. Our system is integrated to the Robotics Operating System (ROS) navigation stack that uses the Rollout Trajectory Planner (RTP) method as a low level controller. This, in contrast to the more popular Dynamic Window Approach (Fox et al., 1997), samples the control space directly instead of sampling target trajectories in velocity space. The advantage of sampling directly in the space of velocity commands is that sampling can be performed at a higher pace than with DWA for which the robot dynamics needs to be inverted to compute the desired velocity commands that will generate the trajectory. The RTP planner has been reported to be the best performing low-level controller in the DARPA LAGR contest (Gerkey and Konolige, 2008). Unfortunately, the planners included in such ROS navigation stack only implement Dijkstra (Cormen et al., 1992) or A* (Hart et al., 1968) algorithms but do not take into account the initial and final robot orientation, as well as the robot's non-holonomic constraints.

Since the cost function in the navigation stack implementation is only related to the traveled distance, the paths obtained are minimal distance paths made up of straight trajectories that enforce in-place rotation at the waypoints, which are problematic for skid-steer platforms such as ours. These rotations put maximum stress on the motors because of friction as the wheels skid, leading to motor failure. To avoid it, we have modified the planner by replacing straight segments with cubic B-splines as in (Pan et al., 2007), enforcing continuity in the start and goal velocities. See Fig. 8. B-spline curves are minimal support representations with respect to a desired degree and smoothness. They are preferred over Bezier curves because their degree is independent of the number of control points.

The main practical application of the B-spline smoother is to account for the non-holonomic constraints of the platform during path execution. The offline global planer takes such platform constraints into account when planning a path. But, the real time local planner for obstacle avoidance that was present in the online navigation ROS module used did not have such feature. Our B-spline smoother serves that purpose, replacing the motion commands created by the the A* planner in the ROS navigation module so that in plane rotations are avoided for our skid steer vehicle.

Given an initial collision-free piecewise linear trajectory computed by the A* local planner on the cost map, our objective is to modify this cost map to smooth the path to the next waypoint. In contrast to (Pan et al., 2007), in which obstacles are dealt with by recursively splitting the piecewise trajectory in those areas in

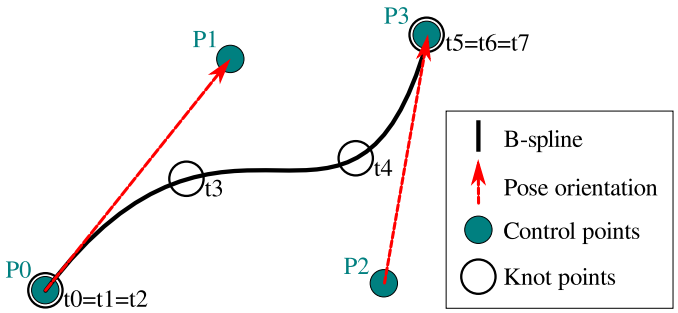


Figure 9: Cubic B-spline with control and knot points that satisfy the velocity constraints.

which a collision is detected, we take a different strategy. We carve a valley on the cost map along a cubic B-spline curve that satisfies continuity of velocity. The curve will erode this valley along collision-free areas but its contribution will not be significant in cells with large obstacle values. The resulting local trajectory will be smoothed while maintaining collision avoidance.

For each linear segment we need $m = 8$ knots, $t_0 \leq \ldots \leq t_{m-1}$, and our B-spline becomes a parametric curve $S(t) : [t_n, t_{m-n-1}] \rightarrow \mathbb{R}^2$, made up of a linear combination of basis B-splines $b_{i,n}(t)$

$$S(t) = \sum_{i=0}^{m-n-2} \mathbf{p}_i \, b_{i,n}(t), \;\; t \, \epsilon \, [t_n, t_{m-n-1}] \tag{2}$$

where $\mathbf{p}_i \, \epsilon \, \mathbb{R}^2$ are the desired control points, and the basis B-splines $b_{i,n}$ obey the Boor recursion. This blending function can be easily precomputed, which put in matrix form becomes

$$S_i(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix} , \tag{3}$$

for $t \, \epsilon \, [0, 1]$.

The control points $\mathbf{p}_i$ are chosen to satisfy linear and angular velocity constraints, and are computed using the robot kinematic equations:

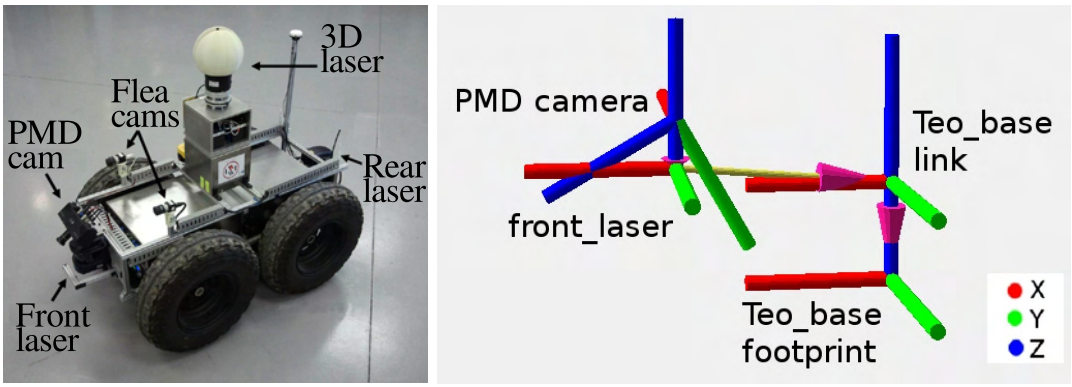
$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + v_{tx} \, dt \\ y_t + v_{ty} \, dt \\ \theta_t + \omega_t \, dt \end{bmatrix} . \tag{4}$$

So, if $\mathbf{p}_0 = [x_s, y_s]^T$ and $\mathbf{p}_3 = [x_g, y_g]^T$ are the start and goal locations, with $\theta_0 = \theta_s$ and $\theta_3 = \theta_g$ the start and goal orientations, the remaining control points, shown in Fig. 9, can be computed by:

$$\mathbf{p}_1 = \mathbf{p}_0 + \mathbf{v}_s \, dt \, , \quad \theta_1 = \theta_0 + \omega_s dt \, , \tag{5}$$

$$\mathbf{p}_2 = \mathbf{p}_3 - \mathbf{v}_g \, dt \, , \quad \theta_2 = \theta_3 - \omega_g dt \, . \tag{6}$$

With this parameterized B-spline we modify the potential propagation wave used by the A* planner, carving on the cells containing the spline path, i.e., subtracting a user-trained value from each cell in the cost map on which the B-spline passes by. Thus, we modify the local A* path with the cubic B-spline and use RTP as a local planner up front to follow the valley in the cost map as closely as possible while taking into account the kinematics of the robot.



(a) (b)

Figure 10: (a) The mobile robot Teo, a skid steer Segway RMP400 platform, and its various sensors, and (b) the relation between its coordinate frames.

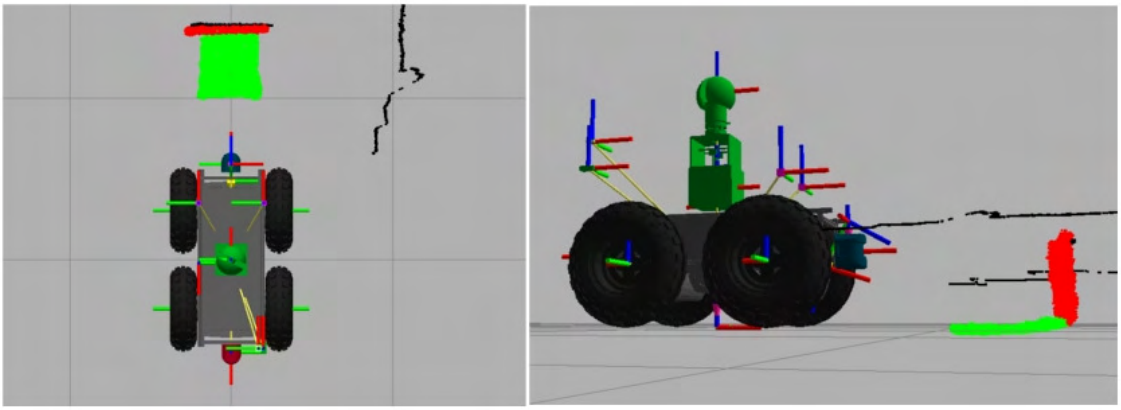


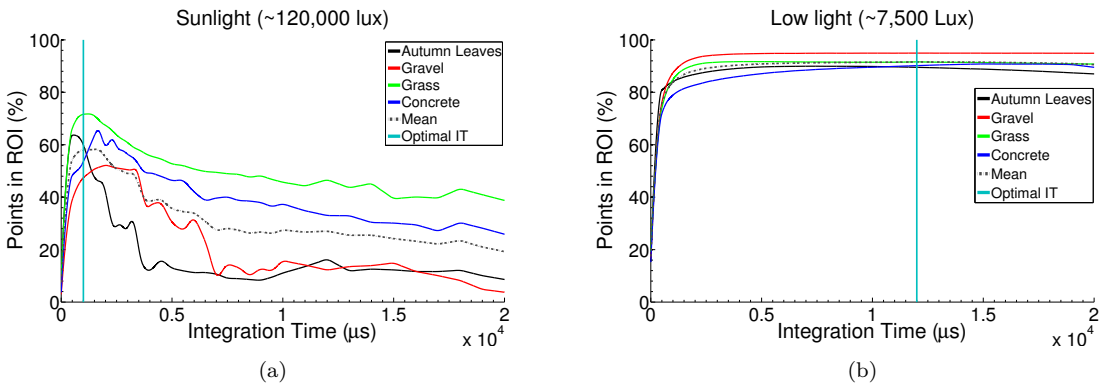
Figure 11: Camera depth calibration. 2D laser shown in black, and ToF range data shown in green and red during the calibration process.

6 Experiments

6.1 Online obstacle avoidance

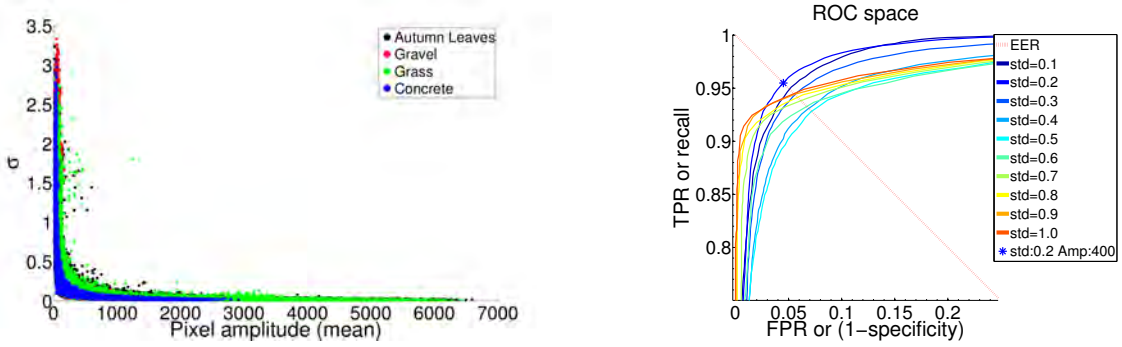
To validate the performance of the low-level online obstacle avoidance method, we conducted a series of experiments using our robot Teo, shown in Fig. 10, a Segway RMP400 unit equipped with a variety of sensors to which we added a PMD Camcube 3.0 ToF camera, working at 20MHz modulation frequency. To obtain data from the ToF camera, we developed a ROS compliant driver and its accompanying ROS nodes, using the SDK libraries from the manufacturer, PMD technologies.

The camera calibration parameters are used by the ROS camera driver to correct monocular distortions and solve camera model errors whereas the front laser in the robot was used as a reference to calibrate the sensor pose calibration with respect to the base reference frame. The robot was placed in front of a wall with the ToF camera facing the wall and the floor, as shown in Fig. 11. The rigid transformation between the ToF camera and the front laser was 130mm, 0mm, 0mm,-2.09rad, 0rad, and -1.5707rad in x, y, z, roll, pitch, and yaw, respectively. Similarly, the front laser is related to the base link by a translation of -590mm along the x axis and 50mm along the z axis; and the base link to the robot footprint by a mere translation of 266.7mm along the z axis.



(a) (b)

Figure 12: Integration time for different scenarios and sunlight illuminances. (a) Bright sunlight (120,000 lux), (b) Low light (7,500 lux).



(a) (b)

Figure 13: Pixel amplitude analysis. (a) Standard deviation vs pixel amplitude, (b) ROC curves

To choose an appropriate integration time (IT) value for the camera, a series of experiments were driven under different illuminations and for various surface types. Fig. 12a shows the percentage of detected points in a region of interest (ROI) corresponding to the floor in front of the robot for varying IT values and heavy sunlight exposure. In contrast, Fig. 12b shows also percentage values of detected points in the ROI under low light conditions, where most of the light reflected to the sensor comes from the camera illumination source. We compute the optimum IT value for each case as the best mean of points in the ROI for the different soil samples in both the high and low sunlight exposure settings. In the bright sunlight case the optimal IT is 1,000 μs and in the case of low light, 12,000 μs is the appropriate IT value. Given that our methods are tailored to outdoor scenarios, we choose as integration time a value of 1,000 μs.

To compute the amplitude and average filter parameters, another series of experiments were conducted. Fig. 13a shows the relationship between the measured standard deviation of point depths read from the sensor and the pixel amplitude mean for different soil types. A suitable way to filter the point cloud is to use a high pass filter on the pixel amplitude. Fig. 13b shows ROC curves at different standard deviations and varying amplitudes. The optimal amplitude is chosen considering the equal error rate (EER) and corresponds to a standard deviation of 0.2 meters and an amplitude of 400.

Considering the maximum expected robot velocity of 1 m/s, a camera frame rate of 15 fps and a camera field of view up to 5 m, we set the average buffer size to 7 frames. This amounts for a delay in obstacle detection of less than half a second.

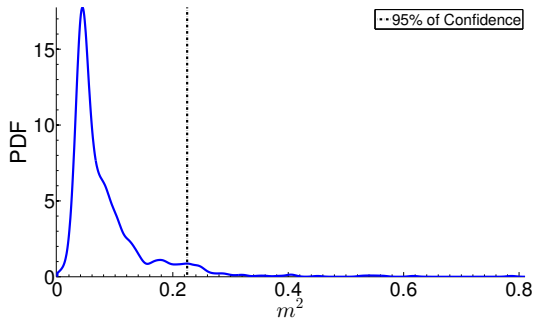
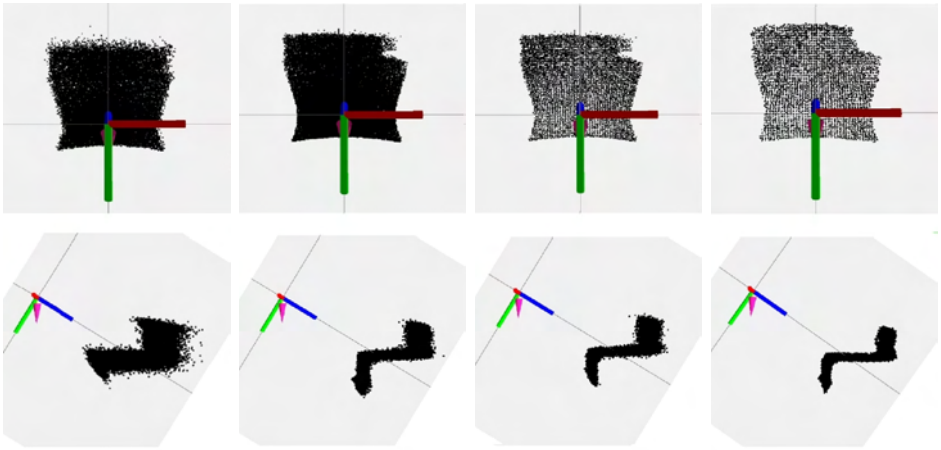


Figure 14: Distribution estimate of neighbor's squared distances.



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

Figure 15: Low-level data processing. (a) Original point-cloud, (b) amplitude and average filters, (c) down-sampling and (d) sparse outlier removal.

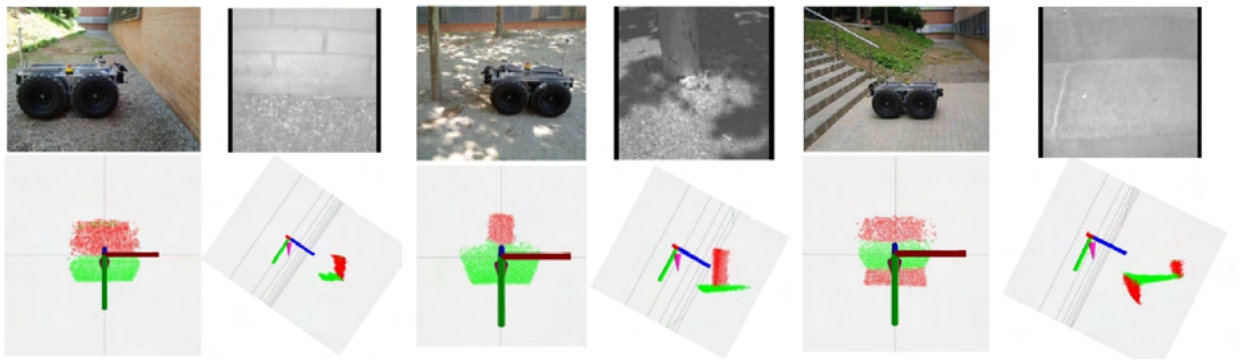
To down-sample the data, we consider the camera resolution (200x200), the focal length ($0.013m$), and the field of view (40°x40°). The desired resolution at which we wish to detect obstacles at $5m$ is $55points/m$, thus we set the downsampling ratio to get one third of the original points with a point each $5.5cm$.

As for sparse outlier removal, if we a compute the squared distance of each point to its k nearest neighbors, and assume a χ^2 distribution for these squared distances, we can easily remove all points laying outside a confidence zone with value of 95% using a χ^2 test. This is shown in Fig. 14. Due to computational limitations, we have set to 20 the total number of k nearest neighbors to compute.

To qualitatively see the effect of each of these filters for the acquisition of the ToF depth maps, Fig. 15 shows the point cloud obtained from the robot facing a set of stairs, and the results of the various low-level filtering steps.

Considering our particular robot platform configuration, obstacle detection was triggered for planar patches of 100 points, due to computational costs, and the reachable slopes with normal orientations smaller than 78.46 degrees from the robot base. Fig. 16 shows obstacle detection for various outdoor scenarios: a wall, a tree trunk, and a set of stairs. In the image, green points mean traversable, whereas red indicate obstacles.

Hole detection is triggered when less than 50 points are detected in any of a set of three boxes of $60 \times 5 \times$



(a) (b) (c)

Figure 16: Obstacle detection on different outdoor scenarios, (a) wall, (b) tree trunk and (c) set of stairs

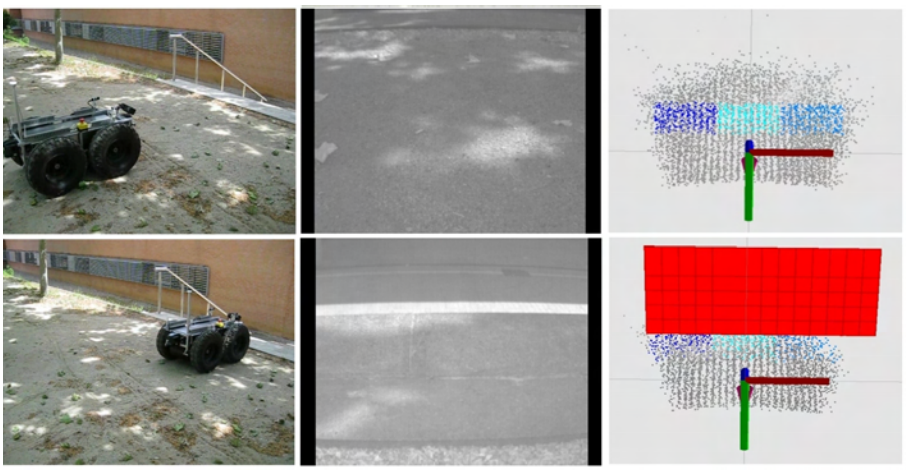


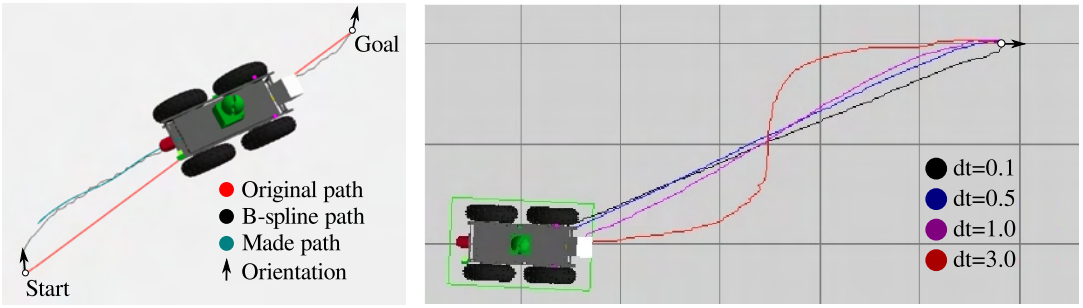
Figure 17: Hole detection.

80 centimeters each, in front of the robot. Fig. 17 shows the three detection regions in three levels of blue, and the detection warning in red once it is triggered.

During local navigation, obstacle information is assembled into a cost map, and our local planner modifies the values in those cells corresponding to the B-spline valley. With these modifications, the planner searches a minimum cost path to the waypoint. Fig. 18a shows the difference between the original non-holonomic unaware planner and our B-spline modification.

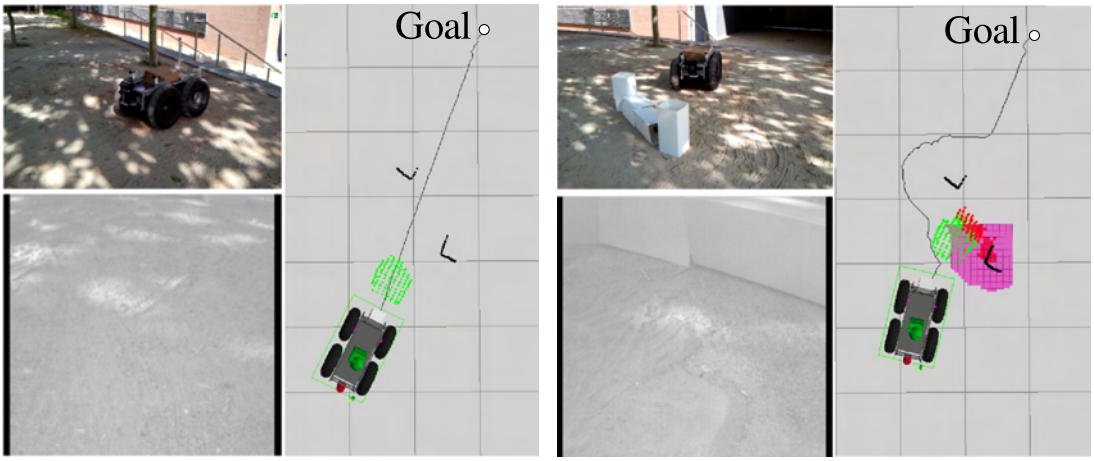
There is one free parameter to choose in our implementation, the B-spline integration time. With this parameter we can effectively control the location of the control points and generate trajectories with varying smoothness. Fig. 18b shows the different forms of the B-spline curve for varying values of dt. Considering a usual distance to the waypoint at 4m and a robot translational speed of 0.2 m/s (although the maximum speed could be at 1m/s, for safety reasons the usual speed is lowered), we get the best results at $dt = 0.5$s.

In our navigation stack the local planner updates in real-time the cost map with obstacles and holes. Thus, the planner updates also in real-time the path to the waypoint. Fig. 19 shows this situation. The figure shows in black the obstacles detected by the front laser and in red the obstacles detected by the ToF camera. The green dots represent the obstacle free areas detected by the ToF camera. Note that our implementation integrates in the cost map information from both the front and rear 2D lasers as well as the ToF camera.



(a) (b)

Figure 18: (a) Comparison of the paths produced by the original and the B-spline local planners in ROS. (b) B-spline trajectories for varying values of the parameter dt.



(a) (b)

Figure 19: Real-time update of the path to a waypoint, (a) before and (b) after obstacle detection.

Finally Fig. 20 shows different frames of an outdoor navigation sequence using the ToF camera integrated with all other sensing devices from Teo with some obstacles in another outdoor scene. In this sequence, the cost map is also plotted. In red, cells in the map with low values, indicating the chosen trajectory. In magenta, the detected obstacles. The lower left frames at each iteration show the robot model together with the path of potentials towards the waypoint as computed with the rollout trajectory planner and the B-spline smoothing.

6.2 Off line terrain classification

Two different datasets were used for the experiments. In both dataset, point clouds were gathered using a custom-built 3D laser with a Hokuyo UTM-30LX scanner mounted in a slip-ring. Each scan has 194,580 points with a resolution of 0.5 deg azimuth and 0.25 deg elevation. The data of the first dataset was acquired in the Barcelona Robot Lab, located at the Campus Nord of the Universitat Politècnica de Catalunya, and consists of a 10,000 sq meters with several levels and underpasses with moderate vegetation. The custom-built 3D laser was mounted on Helena, a Pioneer 3AT mobile robot, acquiring 400 scans. The entire dataset is available in (Teniente et al., 2011). The second dataset was gathered in the Facultat de Matemàtiques i Estadística, located at the Campus Sud of the Universitat Politècnica de Catalunya. The space consists of 2,500 sq meters located in the inner courtyard of the building, which has two main levels together with

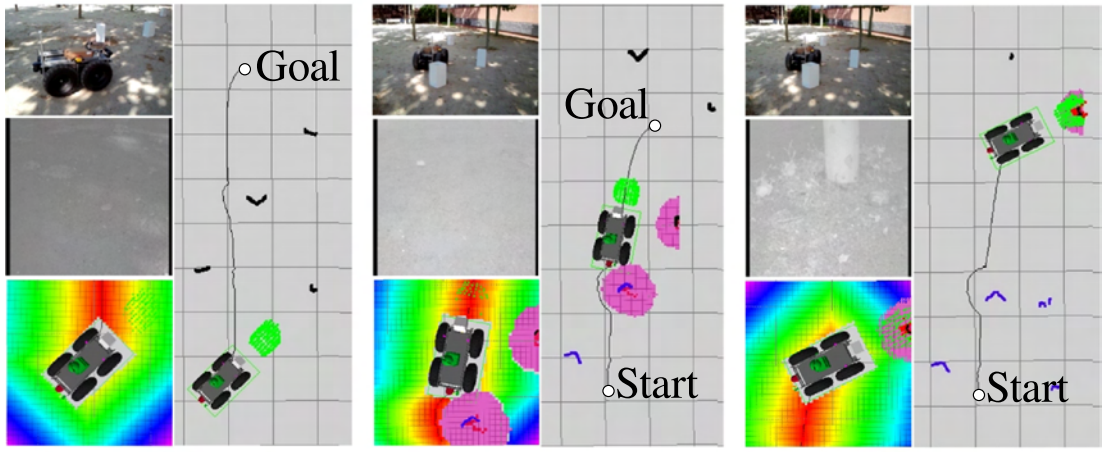


Figure 20: Outdoor navigation using the ToF camera and the front and rear lasers. The figure also plots a color coded version of the cost map.

Method	FME Dataset Recall	BRL Dataset Recall
Naive parametric	0.987	0.978
SVM linear	0.9329	0.9266
SVM quadratic	0.9289	0.9266
SVM polynomial	0.9224	0.9054
SVM RBF	0.9250	0.9112
SVM MLP	0.7039	0.1583
GP regression	0.997	0.998
GP classification	0.995	0.993

Table 1: Traversability segmentation results using only the robot footprint to train the classifier.

some outdoor furniture and some vegetation including trees inside the navigable zone. For this dataset, the sensor was mounted atop Teo, our rough outdoor terrain Segway RPM400 mobile robot. In this case only 39 scans were collected. There are important differences between the two datasets, despite both including uneven terrain, the surfaces that the robots traverse are significantly different. While in the first dataset we only have paving, in the second one we have soil, gravel, grass and concrete. Also Teo is capable of going over bumps as big as 15 cm in height.

The point clouds were uniformly sampled using a box size of $bz = 0.15m$. Then for each sampled point we search the nearest neighbors in a search radius of $1.35bz$ over the raw data, and compute f_r and f_s. Also, we discarded points with local height above $2.5m$, since the points above this height do not add useful information and are a computational burden. Filtering and border detection were performed on the union of the two classified point clouds. In the outlier removal filter we discarded points with less than 7 neighbors within a $3 * h(bz)$ radius, where $h(bz)$ is the box diagonal. In the density filter, the search radius was the same. To compute border features the search radius was set to $2.5 * h(bz)$. The border detection values are: $\kappa = 2$, $r = 1.85$ and $\beta = 35^o$.

For the learning algorithm the training dataset sampling steps were 0.005 for roughness, and 0.5^o for normal orientation. In the GP classification for the first class data we sample $\mathbf{x}$ as stated before. For the second class data we first sampled the unlabeled set $\mathbf{x}_u$ uniformly, and then picked n random samples, where n is 7 times the number of elements in $\mathbf{x}$. We decided to do this because we needed to reduce the training data set dimension to ease computational burden. For the execution times reported, we run the experiments in an Intel Core i7-2720 system @ 2.20 GHZ, with 8 GB of RAM, running Ubuntu 10.04 64 bits, with MATLAB.

Method	FME Dataset			BRL Dataset		
	Precision	Recall	f-score	Precision	Recall	f-score
Naive parametric	0.9212	0.9949	0.9566	0.8832	0.9738	0.9263
SVM linear	0.8422	0.9962	0.9127	0.9015	0.9582	0.9290
SVM quadratic	0.9043	0.4761	0.6238	0.8656	0.9718	0.9266
SVM polynomial	0.8529	0.9962	0.9190	0.9213	0.3822	0.5402
SVM RBF	0.8634	0.9957	0.9248	0.8101	0.9819	0.8877
SVM MLP	0.7582	0.9886	0.8582	0.2156	0.9194	0.3492
GP regression	0.7229	0.9936	0.8369	0.5188	0.9382	0.6681
GP classification	0.9724	0.9916	0.9819	0.9112	0.9617	0.9358

Table 2: Traversability segmentation results using hand-labelled ground truth.

		FME Dataset	BRL Dataset
Total size		16937 points	11518 points
GP regression	Training size	832 points	338 points
	Training time	0.8 sec	0.35 sec
	Segmentation time	0.53 sec	0.19 sec
GP classification	Training size	6080 points	4144 points
	Training time	326 sec	550 sec
	Segmentation time	11.82 sec	8.84 sec

Table 3: Dataset sizes and execution times.

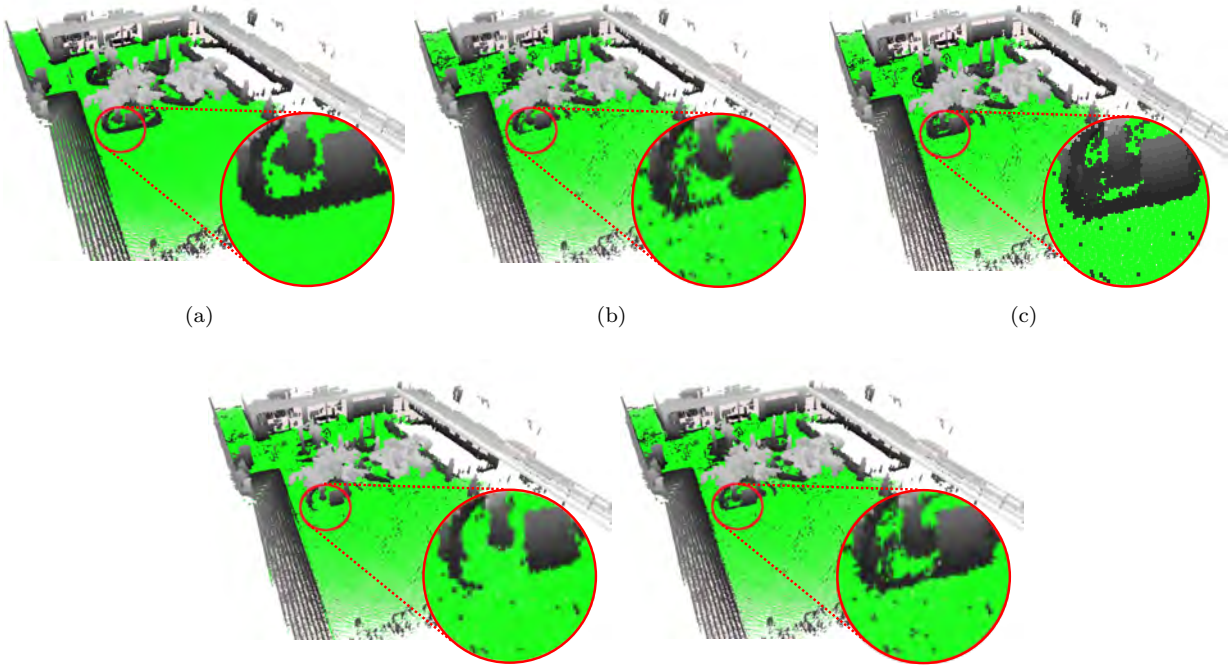
We used the GP toolbox in (Rasmussen and Nickisch, 2010).

We compared the two proposed GP-based segmentation schemes against a naive linear classifier, and also using SVMs with different kernel functions (linear, quadratic, polynomial, RBFs, and multilayer perceptron). To measure classification performance, we employ two different approaches. First we run the regression and classification methods purely on robot footprint points, and use this as ground truth to compute the recall ratios shown in Table 1. We can see that segmentation of the point clouds in both cases, regression and classification, produce slightly better recall values than naive parametric classification and significantly outperform the SVMs. By recall we mean the ratio of true positive samples over the sum of true positive and false negative samples. False negatives are in this case the points in the labeled part of the dataset that were not classified as traversable. Note that since we have only positive samples, only recall as a measure of classification performance can be computed.

The evaluation however is not completely fair. The large recall values obtained for the naive parametric classifier might be misleading since no labeled obstacles are being considered. To come up with an evaluation that takes false positive and true negative classification into account, we hand labeled the point clouds with positive and negative ground truth and computed not only recall but also precision and f-scores. The results are shown in Table 2. By precision we mean the ratio of true positives over the sum of true positives and false positive samples; and the f-score, a common statistic for classification performance, is computed as twice the product of precision and recall over the sum of precision and recall. The table shows how GP classification outperforms all of the other methods on f-scores in both datasets used at the expense of larger computation costs. Database sizes and computation times are given in Table 3.

Quantitatively speaking, GP classification is less resilient to filtering and shows better precision than regression, parametric classification, and SVM classification. In some cases however, it might be sufficient to use GP regression given its significantly smaller computational load. Fig. 21 shows qualitative results of all segmentation methods on the BRL dataset, compared against hand-labelled ground truth.

One reason why GP classification produces slightly better results than GP regression in this particular



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

Figure 21: Qualitative comparison of GP traversability segmentation on the BRL dataset: a) Hand labeled classes, b) Naive parametric classification, c) SVM classification, d) GP regression, and d) GP classification, Traversable areas are shown in green and obstacles are in height-valued gray.

problem might be that whereas function values take fixed values during regression, during classification, the relation from function values to class labels is achieved via class probabilities. The flexibility of using a threshold function to accommodate such class probabilities could be a reason for the improved classification results at the expense of higher computational cost. Although, the statement cannot be generalized to any other GP regression/classification problem.

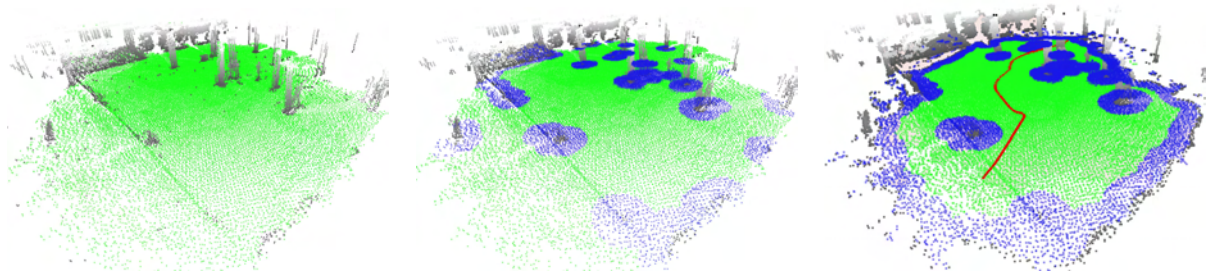
Finally, Fig. 22 shows the results of using the classified point cloud to compute a global path using HRA*.

7 Conclusions

This article presents two systems for 3D terrain classification in outdoor environments. The first, using a TOF sensor, runs in real time and heuristically detects non-traversable areas in front of the robot. The second, using 3D LIDAR point clouds, runs offline and uses a self-supervised classifier to classify regions in a point cloud map as traversable or non-traversable. Trajectory generation and path planning methods for the mobile platform are also addressed.

A first of its kind detailed study in the use of ToF cameras for outdoor mobile robot obstacle avoidance is presented. The technique detects not only obstacles, but also holes, is tightly integrated with the low level controller and is able to produce steering commands to the robot meeting vehicle motion constraints.

An alternative to our method to generate smooth trajectories would be to compute Dubbins curves to the goal (Bonnafous et al., 2001) instead of our smooth B-splines. The substitution of our B-splines for Dubbins curves in the carving of the cost map, as well as the use of intensity images provided by the ToF camera



(a) (b) (c)

Figure 22: Terrain classification on the FME dataset. (a) GP classifier, (b) filtered obstacles, (c) Filtered obstacles and borders, and in red, the resulting HRA* path. Traversable points are in green, obstacles are in height-valued gray, and borders are in blue.

to do the online obstacle detection algorithm more robust, are foreseen additions to our algorithm that we leave for future research.

All our code for this low level module is available for download at `http://www.ros.org/wiki/teo_apps`. The technique includes acquisition and data processing methods to obtain reliable 3D information of a section of terrain in front of a mobile robot. The technique has been designed especially to work on outdoor unstructured environments with a PMD Camcube 3 ToF camera as the main sensing device.

The second method presented is a high-level off-line terrain classification mechanism that processes 3D point clouds to generate traversability maps for the computation of global paths. The method uses Gaussian Processes to classify the terrain as traversable or not, and has the advantage that it can be trained purely from positive samples. These samples can easily be acquired whilst maneuvering the robot in the intended terrain. Using two variants of supervised learning –GP regression and GP classification– we are able to classify dense point clouds acquired with Pose SLAM (Ila et al., 2010).

We showed that with only two features, one for local roughness, and one for slope, we can get classification performance with f-scores better than SVM and naive parametric classification. Collision free regions for path planning were extracted from the filtered classified points, using border and obstacle detection. Finally, we show that the resulting traversability maps can be used for 3D path planning. Given that the GP encodes also the variance of the distribution, we leave also as future work, exploiting such information to also guide the path planning strategy, making the robot navigate along areas with least classification uncertainty, such as in (Valencia et al., 2013).

Acknowledgments

This work has been partially funded by the Spanish Ministry of Economy and Competitiveness under project PAU+ DPI2011-27510 and by the EU projects ARCAS FP7-287617 and Cargo-ANTs FP7-605598.

References

Alenyà, G., Dellen, B., and Torras, C. (2011). 3D modelling of leaves from color and ToF data for robotized plant measuring. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 3408–3414, Shanghai.

Almansa-Valverde, S., Castillo, J., and Fernandez-Caballero, A. (2012). Mobile robot map building from time-of-flight camera. *Expert Syst. with Appl.*, 39(10):8835–8843.

Angelova, A., Matthies, L., Helmick, D., and Perona, P. (2007). Fast terrain classification using variable-

length representation for autonomous navigation. In *Proc. 21st IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–8, Minneapolis.

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Proc. 19th IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 169–176, San Diego.

Blanc, N., Oggier, T., Gruener, G., Weingarten, J., Codourey, A., and Seitz, P. (2004). Miniaturized smart cameras for 3D imaging in real time. In *Proc. IEEE Int. Conf. Sensors*, pages 471–474, Vienna.

Bonnafous, D., Lacroix, S., and Simeon, T. (2001). Motion generation for a rover in rough terrains. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 784–789, Maui.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1992). *Introduction to Algorithms*. MIT Press, Cambridge.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3D lidar point clouds. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2798–2805, Shanghai.

Elkan, C. and Nato, K. (2008). Learning classifiers from only positive and unlabelled data. In *Proc. 14th Int. Conf. Knowl. Discov. Data Min.*, pages 213–220, Las Vegas.

Foix, S., Alenyà, G., Andrade-Cetto, J., and Torras, C. (2010). Object modeling using a ToF camera under an uncertainty reduction approach. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1306–1312, Anchorage.

Foix, S., Alenyà, G., and Torras, C. (2011). Lock-in time-of-flight (ToF) cameras: A survey. *IEEE Sensors J.*, 11(9):1917–1926.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *Robotics Autom. Mag.*, 4(1):23–33.

Fuchs, S. and Hirzinger, G. (2008). Extrinsic and depth calibration of ToF-cameras. In *Proc. 22nd IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1–6, Anchorage.

Gandhi, V., Cech, J., and Horaud, R. (2012). High-resolution depth maps based on ToF-stereo fusion. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 4742–4749, Saint Paul.

Gerkey, B. and Konolige, K. (2008). Planning and control in unstructured terrain. In *Proc. IEEE ICRA Workshop Path Plan. Costmaps*, Pasadena.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernetics*, 4(2):100–107.

Häselich, M., Lang, D., Arends, M., and Paulus, D. (2011). Terrain classification with Markov random fields on fused camera and 3D laser range data. In *Proc. Eur. Conf. Mobile Robots*, pages 153–158, Orebro.

Hebert, M., Caillas, C., Krotkov, E., Kweon, I., and Kanade, T. (1989). Terrain mapping for a roving planetary explorer. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 997–1002, Scottsdale.

Howard, A., Turmon, M., Matthies, L., Tang, B., Angelova, A., and Mjolsness, E. (2006). Towards learned traversability for robot navigation: From underfoot to the far field. *J. Field Robotics*, 23(11-12):1005–1017.

Hussmann, A., Schauer, D., and MacDonald, B. (2009). Integration of a 3D-ToF camera into an autonomous mobile robot system. In *Proc. 26th IEEE Instrum. Meas. Tech. Conf.*, pages 528–533, Singapore.

Ila, V., Porta, J. M., and Andrade-Cetto, J. (2010). Information-based compact Pose SLAM. *IEEE Trans. Robotics*, 26(1):78–93.

Joho, D., Stachniss, C., Pfaff, P., and Burgard, W. (2007). Autonomous exploration for 3D map learning. In Berns, K. and Luksch, T., editors, *Autonome Mobile Systeme*, pages 22–28, Kaiserslautern. Springer.

Karumanchi, S., Allen, T., Bailey, T., and Scheding, S. (2010). Non-parametric learning to aid path planning over slopes. *Int. J. Robotics Res.*, 29(8):997–1018.

Kim, D., Sun, J., Oh, S. M., Rehg, J. M., and Bobick, A. F. (2006). Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 518–525, Orlando.

Lacroix, S., Mallet, A., Chatila, R., and Gallo, L. (1999). Rover self localization in planetary-like environments. In *Proc. 3th Int. Sym. Artif. Intell., Robotics Autom. Space*, pages 433–440, Noordwijk.

Lalonde, J. F., Vandapel, N., Huber, D., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robotics*, 23(1):839–861.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A perception driven autonomous urban vehicle. *J. Field Robotics*, 25(10):727–774.

Liu, B., Lee, W., Yu, P., and Li, X. (2002). Partially supervised classification of text documents. In *Proc. 19th Int. Conf. Mach. Learning*, pages 387–394, Sydney.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 300–307, Anchorage.

Martin, M. and Moravec, H. (1996). Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute. Carnegie Mellon University.

May, S., Werner, B., Surmann, H., and Pervolz, K. (2006). 3D time-of-flight cameras for mobile robotics. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 790–795, Beijing.

Morton, R. and Olson, E. (2011). Positive and negative obstacle detection using the HLD classifier. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1579–1584, San Francisco.

Munoz, D., Vandapel, N., and Hebert, M. (2009). Onboard contextual classification of 3-D point clouds with learned high-order markov random fields. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2009–2016, Kobe.

Murphy, L., Martin, S., and Corke, P. (2012). Creating and using probabilistic costmaps from vehicle experience. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4689–4694, Vilamoura.

Murphy, L. and Newman, P. (2010). Planning most-likely paths from overhead imagery. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 3059–3064, Anchorage.

O'Callaghan, S. T. and Ramos, F. (2012). Gaussian process occupancy maps. *Int. J. Robotics Res.*, 31(1):42–62.

Ortega, A. and Andrade-Cetto, J. (2011). Segmentation of dynamic objects from laser data. In *Proc. Eur. Conf. Mobile Robots*, pages 115–121, Orebro.

Ortega, A., Haddad, I., and Andrade-Cetto, J. (2009). Graph-based segmentation of range data with applications to 3D urban mapping. In *Proc. Eur. Conf. Mobile Robots*, pages 193–198, Dubrovnik.

Pan, J., Zhang, L., and Manocha, D. (2007). Collision-free and curvature-continuous path smoothing in cluttered environments. In *Robotics: Science and Systems III*, pages 1–8, Atlanta.

Paul, R., Triebel, R., Rus, D., and Newman, P. (2012). Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2404–2410, Vilamoura.

Pauling, F., Bosse, M., and Zlot, R. (2009). Automatic segmentation of 3D laser point clouds by ellipsoidal region growing. In *Proc. Australasian Conf. Robotics Autom.*, Sydney.

Pfaff, P. and Burgard, W. (2005). An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. 5th Int. Conf. Field and Service Robotics*, volume 25 of *Springer Tracts Adv. Robotics*, pages 195–206, Port Douglas. Springer.

Piatti, D. and Rinaudo, F. (2012). SR-4000 and CamCube3.0 time of flight (ToF) cameras: Tests and comparison. *Remote Sens.*, 4(4):1069–1089.

Rasmussen, C. and Nickisch, H. (2010). Gaussian processes for machine learning (GPML) toolbox. *J. Mach. Learning Res.*, 11:3011–3015.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press.

Sanfeliu, A., Andrade-Cetto, J., Barbosa, M., Bowden, R., Capitán, J., Murtra, A. C., Gilbert, A., Illingworth, J., Merino, L., Tur, J. M. M., Moreno, P., Ollero, A., Sequeira, J., and Spaan, M. (2010). Decentralized sensor fusion for ubiquitous networking robotics in urban areas. *Sensors*, 10(3):2274–2314.

Silver, D., Bagnell, J. A., and Stentz, A. (2009). Applied imitation learning for autonomous navigation in complex natural terrain. In *Proc. 7th Int. Conf. Field and Service Robotics*, volume 62 of *Springer Tracts Adv. Robotics*, pages 249–259, Cambridge. Springer.

Silver, D., Bagnell, J. A., and Stentz, A. (2010). Learning from demonstration for autonomous navigation in complex unstructured terrain. *Int. J. Robotics Res.*, 29(12):1565–1592.

Sofman, B., Lin, E., Bagnell, J. A., Vandapel, N., and Stentz, A. (2006). Improving robot navigation through self-supervised online learning. In *Robotics: Science and Systems II*, Philadelphia.

Spinello, L., Luber, M., and Arras, K. O. (2011). Tracking people in 3D using a bottom-up top-down detector. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 1304–1310, Shanghai.

Stavens, D. and Thrun, S. (2006). A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proc. 22nd Conf. Uncertain. Artif. Intell.*, pages 13–16, Cambridge.

Stoyanov, T., Magnusson, M., Andreasson, H., and Lilienthal, A. (2010). Path planning in 3d environments using the normal distributions transform. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3263–3268, Taipei.

Teniente, E. and Andrade-Cetto, J. (2013). HRA*: Hybrid randomized path planning for complex 3D environments. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1766–1771, Tokyo.

Teniente, E., Morta, M., Ortega, A., Trulls, E., and Andrade-Cetto, J. (2011). Barcelona Robot Lab data set. [online] http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/php/dataset.php.

Thielemann, J., Breivik, G., and Berge, A. (2008). Pipeline landmark detection for autonomous robot navigation using time-of-flight imagery. In *Proc. IEEE CVPR Workshops*, pages 1–7, Anchorage.

Triebel, R., Pfaff, P., and Burgard, W. (2006). Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2276–2282, Beijing.

Valencia, R., Morta, M., Andrade-Cetto, J., and Porta, J. (2013). Planning reliable paths with Pose SLAM. *IEEE Trans. Robotics*, 29(4):1050–1059.

Valencia, R., Teniente, E., Trulls, E., and Andrade-Cetto, J. (2009). 3D mapping for urban service robots. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 3076–3081, Saint Louis.

Vasudevan, S., Ramos, F., Nettleton, E., and Durrant-Whyte, H. (2009). Gaussian process modeling of large-scale terrain. *J. Field Robotics*, 26(10):812–840.

Weingarten, J., Gruener, G., and Siegwart, R. (2004). A state-of-the-art 3D sensor for robot navigation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2155–2160, Sendei.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., and Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. IEEE ICRA Workshop Best Pract. 3D Percept. Model. Mob. Manip.*, Anchorage.

Yuan, F., Swadzba, A., Philippsen, R., Engin, O., Hanheide, M., and Wachsmuth, S. (2009). Laser-based navigation enhanced with 3D time of flight data. In *Proc. IEEE Int. Conf. Robotics Autom.*, pages 2844–2850, Kobe.