Self-Supervised Learning to Visually Detect Terrain Surfaces for Autonomous Robots Operating in Forested Terrain

Shengyan Zhou and Junqiang Xi
Department of Mechanical Engineering, Beijing Institute of Technology, Haidian, Beijing 100081, China
e-mail: zhousy@mit.edu, xijunqiang@hotmail.com
Matthew W. McDaniel and Takayuki Nishihata
Robotic Mobility Group, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
e-mail: mcdaniel@mit.edu
Phil Salesses
United States Army, Topographic Engineering Center, Alexandria, Virginia, 22315
e-mail: mark.p.salesses@us.army.mil
Karl Iagnemma
Robotic Mobility Group, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139
e-mail: kdi@mit.edu

Received 20 December 2010; accepted 13 November 2011

Autonomous robotic navigation in forested environments is difficult because of the highly variable appearance and geometric properties of the terrain. In most navigation systems, researchers assume a priori knowledge of the terrain appearance properties, geometric properties, or both. In forest environments, vegetation such as trees, shrubs, and bushes has appearance and geometric properties that vary with change of seasons, vegetation age, and vegetation species. In addition, in forested environments the terrain surface is often rough, sloped, and/or covered with a surface layer of grass, vegetation, or snow. The complexity of the forest environment presents difficult challenges for autonomous navigation systems. In this paper, a self-supervised sensing approach is introduced that attempts to robustly identify a drivable terrain surface for robots operating in forested terrain. The sensing system employs both LIDAR and vision sensor data. There are three main stages in the system: feature learning, feature training, and terrain prediction. In the feature learning stage, 3D range points from LIDAR are analyzed to obtain an estimate of the ground surface location. In the feature training stage, the ground surface estimate is used to train a visual classifier to discriminate between ground and nonground regions of the image. In the prediction stage, the ground surface location can be estimated at high frequency solely from vision sensor data. © 2012 Wiley Periodicals, Inc.

1. INTRODUCTION

Unmanned ground vehicles (UGVs) have demonstrated effective autonomous operation in a variety of settings such as deserts, farms, and urban environments (Dahlkamp, Kaehler, Stavens, Thrun, & Bradski, 2006, Manduchi, Castano, Talukder, & Matthies, 2004; Thrun et al., 2006; Wellington, Courville, & Stentz, 2006). Future applications will require UGVs to operate autonomously in forested environments. Robust autonomous operation in a forest depends on the ability of the UGV to distinguish the forest floor from the trees, bushes, shrubs, and other vegetation that can obstruct the UGV's progress. This vegetation can exhibit widely varying geometric properties (size and shape) and appearance properties (color and texture) because of seasonal variation, the age of the vegetation, and the species (type) of vegetation. This is in addition to variation in appearance that can arise because of weather effects and variable illumination. To operate effectively in forested environments, UGVs must identify the drivable terrain surface. In previous research, the authors have developed a LIDAR-based sensing system for forest robots that can robustly detect the terrain surface (McDaniel, Nishihata, & Iagnemma, 2010). However, the LIDAR data acquisition process was found to be time-consuming, preventing real-time operation of the algorithm. In addition, the LIDAR system employed in the work exhibited a relatively short sensing range, less than 30 m, preventing the robot from performing reliable long-range navigation.

In this paper, a novel approach to ground surface detection in forested terrain is presented, which combines LIDAR sensor data with vision sensor data in a self-supervised learning framework. In this system, LIDAR sensor data are acquired infrequently, and are used not only

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE **Self-Supervised Learning to Visually Detect Terrain Surfaces for Autonomous Robots Operating in Forested Terrain**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Robotic Mobility Group,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**21**	

to identify the ground surface (using the technique proposed in McDaniel et al., 2010) but also to automatically supervise the training of a classifier based on visual features (color and texture). The trained visual classifier can operate at a high frequency, and because of the online learning paradigm, can adapt to changes in environment appearance. It can also identify the position of ground surfaces located at a significant distance from the camera. Our experiments show that the resulting sensing system exhibits good performance in several forested environments.

2. STATE OF THE ART

Because of the importance of terrain surface classification for autonomous robots, the issue has been explored in the past. There are three main strategies for this task: vision-based, LIDAR-based, and vision/LIDAR-based.

2.1. Vision-Based

To solve the problem of terrain classification, many researchers rely on stereo cameras for terrain classification (Broggi, Caraffi, Fedriga, & Grisleri, 2005; Kelly & Stentz, 1998; Manduchi et al., 2004). A stereo algorithm finds pixel disparities between two aligned images, calculating a three-dimensional (3D) point cloud. By applying geometrical and statistical heuristics to the 3D point cloud, the terrain surface and obstacles can be classified. Stereo algorithms can generate 3D point clouds at relatively high frequency (several hertz). However, the resulting depth map is typically short range and may be sparse.

Due to the limitation of stereo vision methods, Hadsell et al. (2009) adopt a near-to-far self-supervised architecture. They use a stereo algorithm to produce a 3D point cloud; then ground plane and footline estimation methods are applied to separate these points into ground, obstacle, and footline classes. After projection of these labeled points onto images, the online learning framework extracts the visual features and uses them to train a classifier. The trained classifier can be used to predict long-range visual data from images.

2.2. LIDAR-Based

There are also many researchers using LIDAR sensors to detect the terrain surface for robot navigation (Elmqvist 2002; Hebert & Vandapel, 2003; Lalonde, Vandapel, Huber, & Hebert, 2006; Wellington & Stentz, 2003). In general, LIDAR sensors can return dense 3D point clouds, however, scanning LIDAR sensors often operate at a relatively low frequency (1 Hz or less). Fixed (nonscanning) LIDAR sensors can operate at a high frequency; however, they require an additional algorithm to accumulate data as the robot moves. This algorithm relies on accurate robot pose estimation, which is difficult to achieve in the forest, where the terrain surface is often rough and sloped, and the presence of a tree canopy makes GPS signal reception unavailable.

Elmqvist (2002) separates 3D laser radar points into several sampled grids and adopts an energy function of the active shape models to evaluate the ground surface in each grid. According to this evaluation, an energy map is generated from those 3D grids. The lower the value of the grid, the higher the probability that it belongs to the ground surface.

Wellington & Stentz (2003) rely on multiple LIDARs to collect 3D points and register them into a global map. This mapping is learned by observing actual vehicle motion after driving over a given terrain. Associated with 3D points on the map, four features are extracted from these points and then used to train the robot for predicting the terrain properties in the front of robot. This method has shown to be effective in various environments.

In (Lalonde et al., 2006), the author used LIDAR to collect 3D point clouds. Principal component analysis is applied to extract the first three eigenvalues from a symmetric positive definite covariance matrix of neighboring 3D points. Then a hand-labeled training data set is used to train a terrain model. Last, the trained model is used for terrain classification. In experiments, this method performs well on various scenes.

2.3. LIDAR/Vision-Based

Due to the limitation of both the vision and LIDAR sensors mentioned, (Dahlkamp et al. (2006), Konolige et al., (2009), Rasmussen (2002), Sofman et al. (2007), and Thrun et al. (2006) combine vision and LIDAR into one system, to mitigate the drawbacks of each approach.

Dahlkamp et al. (2006) apply a Kalman filter to estimate the robot pose based on measurements from two differential GPS systems and a six-degree-of-freedom inertial measurement unit (IMU). Based on the pose estimation, 2D LIDAR points are projected into a 3D frame. Using the PTA algorithm, the drivable points can be calculated. Then, because the camera and LIDAR are calibrated and registered with each other, points in the image associated with drivable points in the 3D point cloud are labeled as training data to train a visual classifier. Last, using the self-supervised trained classifier, a prediction of the drivable area in the image is made for robot navigation.

Sofman et al. (2007) apply a learning approach to integrate both general feature-based estimation and self-supervised locale-specific estimation to improve navigation capabilities for unmanned ground vehicles. In their application, they integrate overhead data and far-range sensor data into an online learning framework. Both local information and global information are used to yield a cost map to improve robot navigation. Experiments show dramatic improvement of navigation performance in traversal time, distance traveled, and average speed.

A key difficulty of self-supervised classification methods is ensuring that erroneous training data are not (automatically) inserted into the training process, because this could degrade, rather than improve, classifier performance. This is difficult, because in the self-supervised paradigm, an expert is not available to discriminate between good and bad training examples.

Forested environments often contain obstacles such as trees and shrubs that may be in close proximity to one another. This requires that 3D point clouds be generated at a relatively high frequency, in order to rapidly identify the drivable terrain surface. This requirement makes some LIDAR-based algorithms difficult to apply, because of the relatively low bandwidth of LIDAR 3D point capture. Camera-based navigation methods can operate at high frequency; however, it is challenging to describe a consistent model for the appearance of terrain surface in forest environment (e.g., for approaches that rely on supervised classification to detect obstacles based on appearance). This makes camera-only algorithms difficult to apply in forested environments.

Here, a LIDAR/camera-based method for navigation in forested environments is presented. This work adopts a self-supervised approach that trains a visual classifier based on the output of a LIDAR-based ground plane detection method. This mitigates the drawbacks of LIDAR-based methods (specifically, low-bandwidth data capture) and vision-based methods (specifically, sensitivity to variation in scene appearance). The application of such a scheme in a forested environment represents a novel contribution to the literature. A novel approach to classifier training that relies on morphological operations is introduced, with the goal of minimizing the number of erroneous training examples that are automatically provided to the visual classifier.

3. OVERVIEW OF PROPOSED APPROACH

This paper presents a self-supervised learning approach to identifying drivable terrain surface for a robot operating in forested terrain. This approach has three separate stages: a LIDAR-based ground estimation stage, a camera-based self-supervised learning stage, and a camera-based online learning stage. The first two stages provide training data to train a visual classifier, whereas the third stage runs (during navigation) between LIDAR scans.

In the experiments presented in Section 8 of this paper, the robot remained stationary in the first stage. The LIDAR tilt stage scanned to acquire a 3D point cloud. By applying a feature extraction algorithm to these 3D points, seven features were adopted to describe differences between ground and nonground points and train a classifier to identify the ground surface in those points. Combined with a triangulated irregular network (TIN) approach, a set of 3D training points are generated and projected into the image plane.

In the second stage, given the training set in the image plane, a feature selection method is employed to select five features that best describe differences between ground and nonground appearance. Then an initial visual classifier is learned by applying these selected features to the training pixels.

In the third stage, an online learning algorithm is activated to update the visual classifier in order to adapt to changes in the environment.

This paper is organized as follows: Section 4 provides an introduction of the system architecture and sensor calibration. In Section 5, a ground surface estimation algorithm based on LIDAR sensor data is briefly introduced. Algorithms for visual feature selection and classifier training are presented in Section 6. In Section 7, an online learning algorithm is demonstrated. Experimental results and conclusions are presented in Section 8 and Section 9.

4. SYSTEM ARCHITECTURE AND SENSOR CALIBRATION

In this work, a small robotic platform is employed for experimental sensor data collection purposes (see Figure 1). The robot base is a MobileRobots Pioneer 3-AT, and it is equipped with a Point Grey Bumblebee XB3 camera, a Hokuyo UTM-30LX LIDAR, a servo-controlled tilt stage, and a step motor. The forward-looking camera is fixed in front of the robot, and the LIDAR is mounted on a servo-controlled tilt stage to enable collection of a 3D point cloud. The camera and LIDAR are calibrated with respect to one another, to enable the LIDAR point cloud to be projected into image coordinates.

In the proposed self-supervised learning framework, data from the LIDAR sensor are used to supervise training of a vision-based classifier. This requires that the 3D LIDAR points are mapped to the camera images. Thus, the precise relationship between 3D points and image pixels must be known. For this task the calibration is implemented using the Laser-Camera Calibration Toolbox (Unnikrishnan & Hebert, 2005). A brief description of the robot's coordinate systems and calibration parameters is given here.

Five coordinate frames are used in the following work: world coordinates, robot coordinates, servo-controlled tilt stage coordinates, LIDAR coordinates, and camera coordinates (see Figure 2). World coordinates are represented in an inertial frame and used for global path planning and object localization. Robot coordinates are represented in a robot-fixed frame and are used for local path planning. Because the servo-controlled tilt stage and camera are rigidly mounted on the robot body, they are fixed with respect to the robot coordinates.

To perform (offline) calibration, various calibration parameters and coordinate transformations must be identified, in order to map 3D points into image coordinates.

A 3D point is denoted by $\mathbf{M_L} = [X_L, Y_L, Z_L]$ in servo-controlled tilt stage coordinates. The LIDAR is mounted on the servo-controlled tilt stage, which has just one degree of freedom in the pitch direction. The 3D point $\mathbf{M_L}$ can be

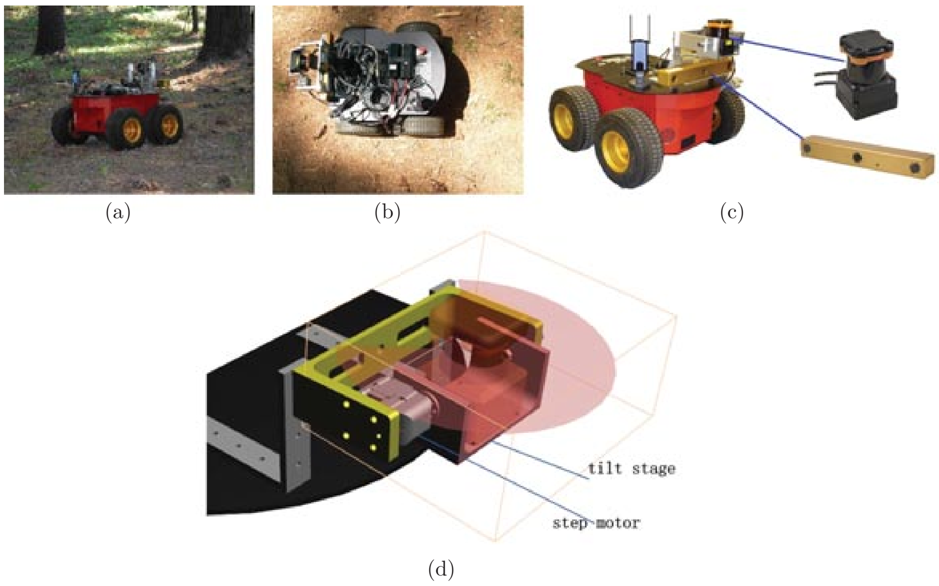


Figure 1. Experimental robot platform, (a) lateral view and (b) top view. (c) Perception sensors: camera and LIDAR. (d) Servo-controlled tilt stage and step motor

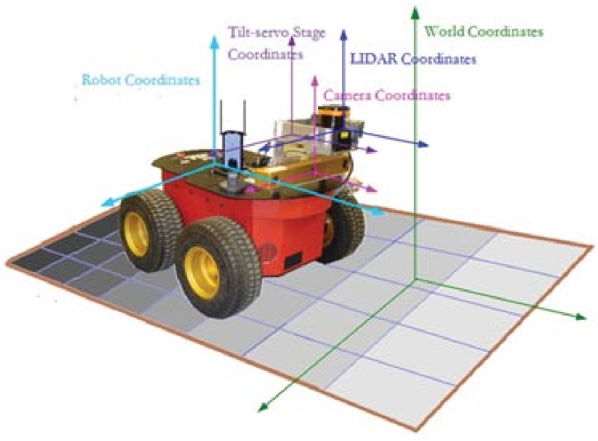


Figure 2. Robot coordinate system.

calculated as follows:

$$\mathbf{M}_L^{'} = [X_L Y_L Z_L]^{'} = \begin{bmatrix} \sin\sigma \\ \cos\sigma\cos\theta \\ \cos\sigma\sin\theta \end{bmatrix} d, \tag{1}$$

where D refers to the distance measured by LIDAR, σ refers to the scanning angle in the LIDAR frame, and θ refers to the pitch angle in the servo-controlled tilt stage frame.

It is assumed that the lens model is a pinhole camera model. A 3D point is denoted by $\mathbf{M_C} = [X_C, Y_C, Z_C]$ in camera coordinates. A rigid transformation between the servo-controlled tilt stage and camera frames is as follows:

$$\mathbf{M}_C^{'} = [\mathbf{R} \quad \mathbf{T}][\mathbf{M}_L \quad 1]^{'}, \tag{2}$$

where $[\mathbf{R}\ \mathbf{T}]$ is the rotation and translation that relate the servo-controlled tilt stage coordinates to the camera coordinates. After this translation, the normalized pinhole projection is given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}, \tag{3}$$

where $(x \ \ y)$ is the pixel projected onto the image frame without distortion. Adjusting for radial lens distortion, the normalized point is calculated as

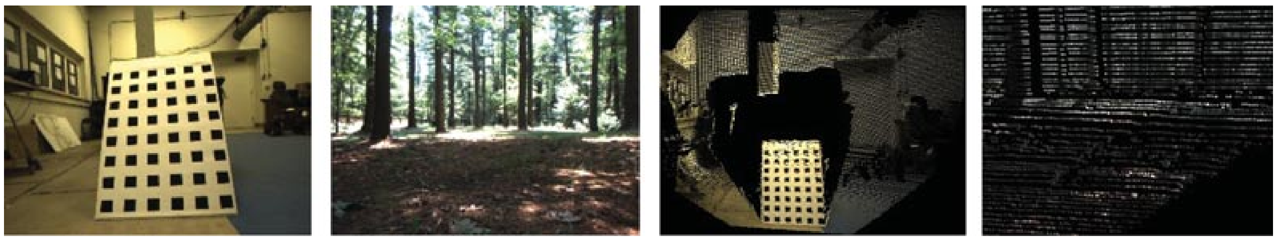
$$\begin{bmatrix} x_r \\ y_r \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + dx, \tag{4}$$

where $r^2 = x^2 + y^2$, and dx is the tangential distortion vector. In our application, we assume dx to be zero. The final pixel coordinates of the point, (c, r), are then given by

$$\begin{bmatrix} c \\ r \\ 1 \end{bmatrix} = A \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} + \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix}, \tag{5}$$

where $\mathbf{A}$ is called the camera intrinsic matrix with (u_0, v_0) the coordinates of the principal point, α and β the scale factors in the image horizontal and vertical axes, and γ the parameter describing the skewness of the two image axes.

Thus, given formulas (1)–(5) and the values for $\mathbf{R}$, $\mathbf{T}$, k_1, k_2, k_3, α, β, and γ, from the calibration, we can associate 3D points from the LIDAR with image pixels from the camera (as shown in Figure 3).



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

Figure 3. Visualized 3D point; (a) and (c) are images captured by camera in laboratory and forest; (b) and (d) are separately colorized 3D points.

5. GROUND ESTIMATION USING LIDAR

In this section, we briefly review our two-stage approach to ground plane estimation from 3D data, which was specifically designed for sloped, rough environments (McDaniel et al., 2010).

5.1. Approach Summary

The approach (McDaniel et al., 2010) divides the task of ground plane identification into two stages. The first stage is a local height-based filter, which is based on the assumption that, in any vertical column (of specified dimensions) containing range data points, only the lowest point can belong to the ground. In practice this eliminates a large percentage of nonground points from further consideration. (In the test data sets presented here, 98.7% of data points were eliminated through this method.) The second stage uses a support vector machine (SVM) classifier (Burgers, 1998), which combines eight heuristically inspired features to determine which of the remaining points belong to the ground plane.

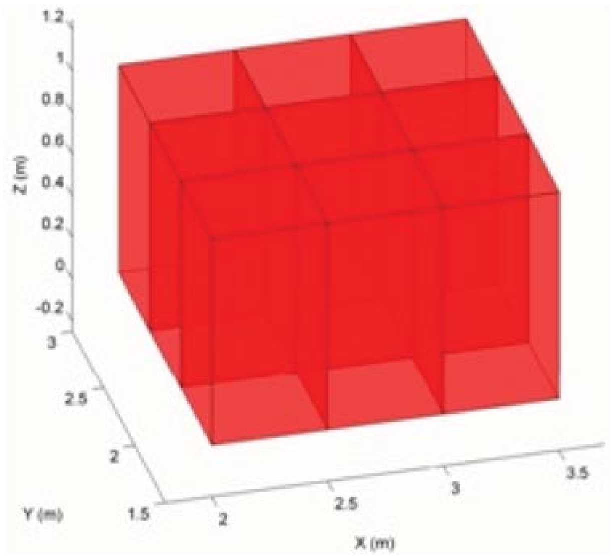


Figure 4. A portion of the Cartesian space partitioned into columns.

5.2. First Stage

Given a set of range data points in Cartesian space, the goal of ground plane identification is to identify which of those points lie on the ground surface. In this work, candidate points are represented in an inertial frame with coordinates (x, y, z).

In the first stage, the points are divided into (0.5×0.5)-m columns based on their x and y values. These columns are identified by indices (i, j), where $i = [x/0.5]$ and $j = [y/0.5]$. In each of these columns, only the lowest point (i.e., the point with minimum z value) is retained as a possible ground point. For simplicity, the lowest point in column (i, j) is hereafter denoted $P_{i,j}$, and its coordinates are referred to as $(x_{i,j}, y_{i,j}, z_{i,j})$. Figure 4 clarifies the concept of dividing the Cartesian space into columns.

5.3. Second Stage

In the second stage, a variety of features are extracted to represent attributes of each point $P_{i,j}$ and the lowest points in each of the eight neighboring columns (i.e., $P_{i-1,j-1}$, $P_{i-1,j}$, $P_{i-1,j+1}$, $P_{i,j-1}$, $P_{i,j+1}$, $P_{i+1,j-1}$, $P_{i+1,j}$, and $P_{i+1,j+1}$, as shown in Figure 5. A set of eight features was defined based on their usefulness in discriminating ground from nonground. These features, denoted $f_1, \ldots, f_8$, are combined into a feature vector $\mathbf{F_{i,j}} = (f_1, \ldots, f_8)$ for each point, which is used by a classifier to identify whether that point belongs to the ground. These features include

- f_1: Number of occupied columns in the neighborhood of column (i, j)
- f_2: Minimum z of all neighbors minus $z_{i,j}$
- f_3: Value of $z_{i,j}$
- f_4: Average of all z values in neighborhood
- f_5: Normal to best fit plane of points in neighborhood
- f_6: Residual sum of squares (RSS) of best fit plane of points in neighborhood
- f_7: Pyramid filter
- f_8: Ray tracing score

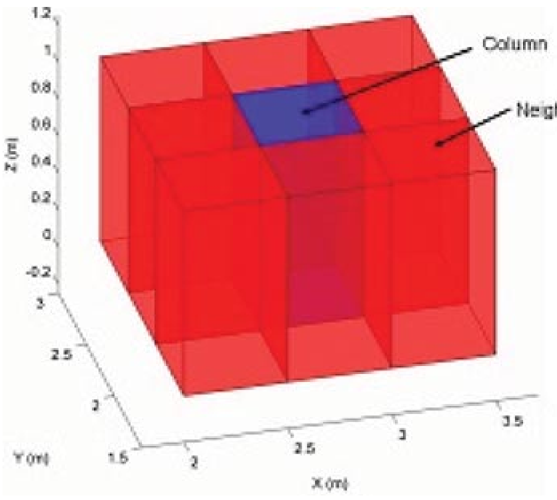


Figure 5. A column and its neighbors.

A description for these eight features is presented below:

1. f_1: *Number of occupied columns in neighborhood*. This feature is used to evaluate the density of points in column (i, j). This feature encodes the fact that bushes, shrubs, and tree trunks typically block the trace of LIDAR scanning and cast "shadows" in point clouds, which reduces the number of points in occupied neighbors. Thus, not every column (i, j) will contain LIDAR points. This feature is computed as the number of occupied columns in the neighborhood of column (i, j):

$$f_1 = N. \tag{6}$$

2. f_2, f_3, f_4: *Features using z height values*. Feature f_2 expresses the smoothness of the terrain around column (i, j). In general, ground can be considered to have a relatively smooth surface compared to the edges of trees or shrubs. Feature f_2 takes the difference between $z_{i,j}$ and the minimum z of all neighboring columns:

$$f_2 = \min(z_{i-1,j-1}, z_{i-1,j}, z_{i-1,j+1}, z_{i,j-1}, z_{i,j+1}, z_{i+1,j-1}, z_{i+1,j}, z_{i+1,j+1}) - z_{i,j}. \tag{7}$$

Features f_3 and f_4 use the z value in each column, encoding the assumption that the ground will generally not be located significantly higher than the robot. This assumption is expected to be true except for cases with highly sloped terrain. In that situation, f_7 and f_8 will compensate for the drawback inherent in this assumption:

$$f_3 = z_{i,j} \tag{8}$$

$$f_4 = (z_{i-1,j-1} + z_{i-1,j} + z_{i-1,j+1} + z_{i,j-1} + z_{i,j+1} + z_{i+1,j-1} + z_{i+1,j} + z_{i+1,j+1}) + z_{i,j}. \tag{9}$$

3. f_5, f_6: *Features using best fit plane*. f_5 and f_6 encode the fact that ground is typically flatter than other objects (e.g., shrubs, tree trunks, and canopy). The plane that minimizes orthogonal distances to candidate points is found using principal components analysis. Thus, given the set of N points in the neighborhood, $P_k = [x_{i,j}, y_{i,j}, z_{i,j}]$, and their mean, $\bar{P} = (1/N)\sum_{k=1}^{N} P_k$, the normal vector n is calculated as

$$n = \arg \min_{n \in R^3, \|n\|^2=1} \sum_{k=1}^{N} ((P_k - \bar{P}) \cdot n)^2. \tag{10}$$

Feature f_5 is the dot product between the normal of the plane and the z axis:

$$f_5 = n \cdot (0, 0, 1). \tag{11}$$

f_6 is another measurement of smoothness of the terrain, and is calculated as

$$f_6 = \frac{1}{N} \sum_{k=1}^{N} ((P_k - \bar{P}) \cdot n)^2. \tag{12}$$

4. f_7: *Pyramid filter*. A ground filter similar to the one in (Lalonde et al., 2006) was also implemented. This filter counts the number of points falling within a downward-facing cone with its vertex located at a candidate point. To improve computational efficiency, this filter is here discretized by forming a pyramid structure [instead of a cone as in (Lalonde et al., 2006)] of cubic voxels under each point $P_{i,j}$. The number of other candidate ground points that fall within the pyramid of $P_{i,j}$ is counted, and this count is used as feature f_7. A representative pyramid of voxels is shown in Figure 6.
5. f_8: *Ray-tracing score*. The last feature is inspired by ray tracing. Intuitively, it is obvious that the ground (or any other structure) cannot lie directly between the LIDAR and any point it observes. Similarly, the ground cannot lie above the line segment formed between the LIDAR and the points it observes. Feature f_8 quantifies this insight using a voxel-based approach. For each $(0.5 \times 0.5 \times 0.5)$-m cubic voxel containing a point $P_{i,j}$, f_8 is a sum of the line segments passing directly under a voxel in column (i, j) that contains point $P_{i,j}$. Thus, points with lower ray tracing scores are more likely to be ground, and points with higher scores are less likely. This concept is illustrated in Figure 7. The black arrow represents a ray traced from the LIDAR to a data point. The voxels that this ray passes through are blue, and the voxels above it are red. Any point in a red voxel would have its ray tracing score incremented by one.

5.4. Training and Classification Description

To train the SVM classifier, a variety of point cloud data sets in different scenes were collected for training purposes. These data sets were labeled by hand to assign each

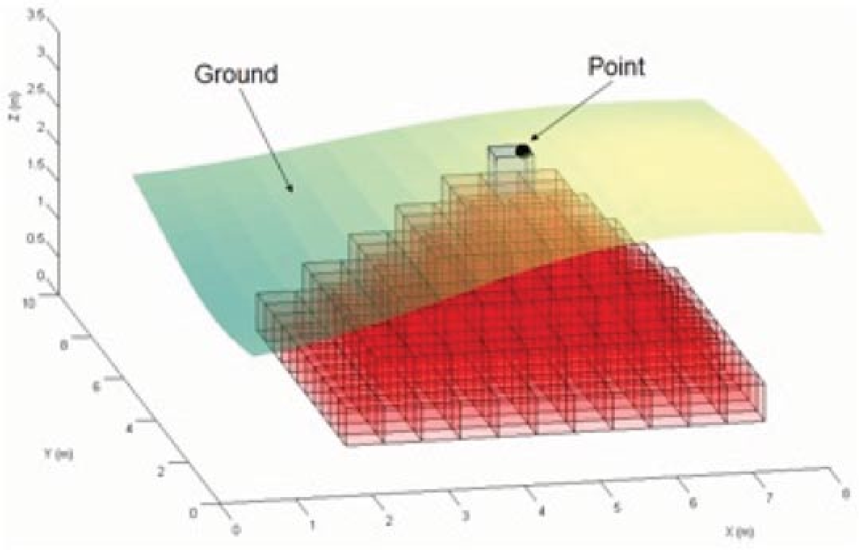


Figure 6. A pyramid of voxels under a candidate data point. The voxel that contains the candidate point is at the pyramid apex.

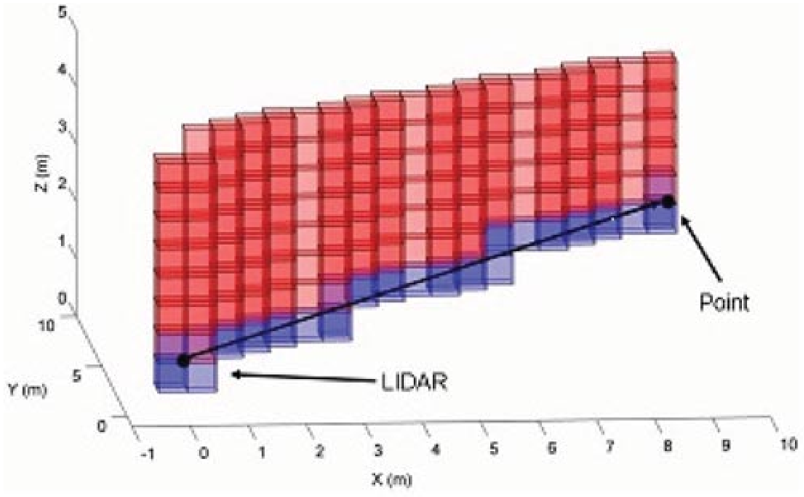


Figure 7. Ray traced from data point back to LIDAR sensor source. Voxels above the traced line segment are red, and voxels along the line segment are blue. (Image best viewed in color.)

point to ground/nonground classes. Then the feature vectors of those labeled points were extracted according to the method above. Given the training points and their feature vectors, a SVM classifier is trained. Based on cross validation within the training data, a set of appropriate kernel parameters can be found. For this work, the SVMclassifier was implemented using LIBSVM (Chang, 2008). For this paper a linear kernel was used with SVM parameter $C = 100$ selected based on cross-validation. Using the trained classifier, points belonging to a previously unlabeled scene can be classified. The experiments are demonstrated in the Section 8.

6. SELF-SUPERVISED VISUAL LEARNING

In the self-supervised learning algorithm proposed here, the ground estimation algorithm is used to identify the locations of 3D points that belong to the ground surface. Then, using the calibrated and registered sensor system described above, 3D ground points can be associated with pixels in the image captured by the camera. During navigation, training examples can thus be provided to a vision-based supervised classifier, in order to form a classifier to discriminate between ground points and nonground points.

Several issues must be addressed before the proposed learning algorithm can be successfully implemented. First, the ground estimation algorithm analyzes only the single lowest point in every column, to determine whether it belongs to the ground surface. This results in a sparse projection of points into the image plane. Second, the ground estimation algorithm discards nonground points during its filtering stage, which results in a sparse number of data associated with features such as leaves or branches, which could be used as negative training data in the visual classifier.

The proposed self-supervised learning algorithm is composed of three elements: training set generation, feature selection, and classifier training. These elements are described here.

6.1. Training Set Generation

In the LIDAR-based ground estimation algorithm, each point in the 3D point cloud is assigned membership in a particular column, and the lowest point in each column is used as a candidate point for ground estimation. As noted above, when these points are associated with pixels, this results in a sparse data set for visual classifier training. Because of this, a training set generation algorithm has been developed to yield additional 3D candidate training points from the 3D point cloud dataset. In this paper, a triangulated irregular network (TIN) is employed to model the ground plane. Additional 3D points are then extracted, which are used for projection into the image frame for visual classifier training.

A TIN is a data structure used for surface representation. There are various ways to create a TIN, but the most frequently used method is Delaunay triangulation. Delaunay triangulation finds triangles from a set of points such that no point in the set is inside the circumcircle (or circumsphere in 3D) of any other triangle in the triangulation. In this paper, Delaunay triangulation is used to model the surface of the ground plane for each scene using the LIDAR data which was classified as ground. The resulting triangulation for each scene is presented in Section 8, in which the color of each triangle is proportional to its height.

The local height-based filter presented in the previous section eliminates approximately 98% of the data points after filtering. When the surface model is built, some of those eliminated points can be used to complement training data. In other words, the TIN-based surface model is calculated by using the lowest 3D points that are not filtered by the local height-based filter, which are classified as ground points. In the training set generation stage, those raw points that were filtered by the local height-based filter are considered as candidates to complement visual training data. For simplicity, we call these points filtered points in the following sections.

The plane of the triangle, $Ax + By + Cz = D$, can be found by calculating the normal to that triangle. Given the three corners of the triangle, $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$, from ground points, where each point is a vector of $\mathbf{p}_t = [x_t, y_t, z_t]$, there exist certain filtered points $(\mathbf{q}_1, \ldots, \mathbf{q}_j)$, where each point is a vector of $\mathbf{q}_i = [x_i^q, y_i^q, z_i^q]$ in the triangle in the X–Y plane. More training data are generated via the following process.

First, the normal to this triangle is computed as

$$n = (\mathbf{p}_2 - \mathbf{p}_1)\times(\mathbf{p}_3 - \mathbf{p}_1) = A\hat{x} + B\hat{y} + C\hat{z}. \qquad (13)$$

The last coefficient can be calculated using one of the three corners $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$:

$$D = Ax_1 + By_1 + Cz_1. \qquad (14)$$

The estimated ground height at (x_i^q, y_i^q) of $\mathbf{q}_i$ can be calculated by

$$z_i^Q = (D - Ax_i^q + By_i^q)/C, \quad (i = 1, \ldots, N). \qquad (15)$$

The decision on whether $\mathbf{q}_i$ belongs to a class composed of ground points or a class composed of nonground points can be made based on the following condition:

$$V_{q_i} = \begin{bmatrix} 1 & z_i^q \le z_i^Q + T_2 \\ -1 & z_i^q \ge z_i^Q + T_1 \\ 0 & z_i^Q + T_1 > z_i^q > z_i^Q + T_2 \end{bmatrix}, \qquad (16)$$

where V_{q_i} denotes the class the point $\mathbf{q}_1$ belongs to. T_1, T_2 are thresholds to measure the height distances of ground points or nonground points to the estimated plane. (For this work, T_1 is 1,000 mm, and T_2 is 10 mm.) Here, "0" means the point is uncertain. "1" means the point belongs to the ground, whereas "−1" means the point does not belong to the ground. Points in the "ground" and "nonground" classes can then be used as positive training points (for points on the ground surface) and negative training points (for points not on the ground surface), respectively. An example of a resulting training set is shown in Figure 8.

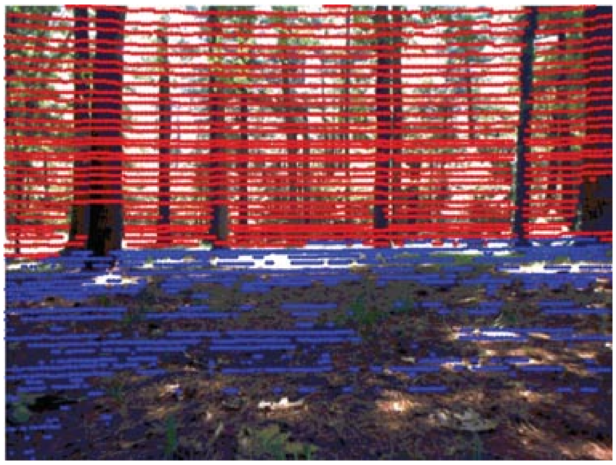


Figure 8. Example of projected training points on the image.

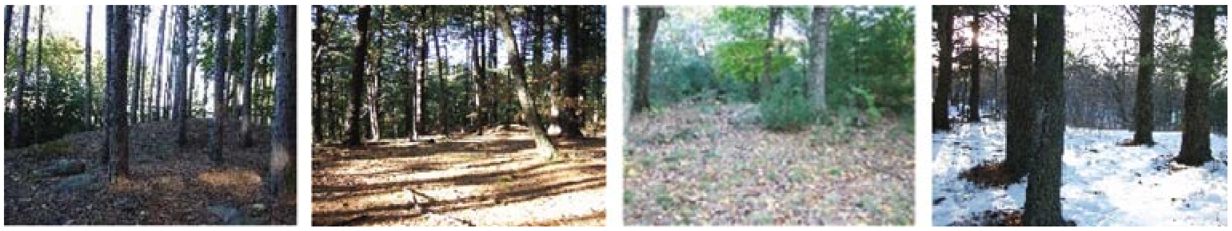


Figure 9. Examples of forest appearances for various data sets.

6.2. Feature Selection and Classifier Training

Given ground and nonground training points in an image, visual features (i.e., color and texture) associated with each point can be used for classifier training. Here, no a priori assumption is made regarding which features will have a high degree of discriminative power in separating ground/nonground classes in the forest environment. This is because the appearances of ground and nonground regions in the forest environment can vary widely from scene to scene and season to season (as shown in Figure 9).

In this paper, the AdaBoost algorithm associated with Fuzzy SVM (Lin & Wang, 2002) is employed for feature selection purposes. Candidate features are divided into color features and texture features. The six candidate color features include both RGB and HSV features [R, G, B, H, S, V]. Nine candidate texture features are Haar features in three directions (vertical, horizontal, and diagonal) and three scales. Haar features are extracted in the gray image channel. Thus, there are 15 candidate features for selection.

The AdaBoost algorithm was first introduced in (Freund & Schapire, 1997). The purpose of this algorithm is to combine several weak classifiers into a strong classifier. Another benefit of this algorithm is that, given many features, the boosting process analyzes individual feature contributions in classifying training data. During the boosting process, the weight on each training sample is updated according to the weak classifier in each boosting round. The weight will be smaller if the sample is rightly classified by the weak classifier, and vice versa. In that case, the weak learner would focus much more on those that were not correctly classified in the next round of boosting. As a result, each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process (Viola & Jones, 2001).

The classic AdaBoost algorithm (Viola & Jones, 2001) uses a stump-based weak classifier, which performs well in feature selection for the final cascade classifier in the application of face detection. However, in the application described here, the task is to select several good features, not to form a final cascade classifier. In that case, the weak classifier should have more capability for evaluating the discriminative power of each feature.

In this paper, we present a novel weak classifier in the AdaBoost algorithm. The Fuzzy SVM, which was first introduced in (Lin & Wang, 2002), can accept as input a weight on each training sample. The weight can make different contributions to forming the decision surface. In another words, higher weights on training samples have more contribution in forming the decision surface. The process of feature selection in this paper is shown as Algorithm 1.

AdaBoost and the Fuzzy SVM algorithm are used to select a good feature set. Then, given the selected features, the support vector machine is implemented to determine a decision hyperplane in the feature space for the visual classifier. Based on cross validation within the training data, a set of appropriate kernel parameters can be found. [For this work, the SVM classifier was implemented using LIBSVM (Chang & Lin, 2008), and a RBF kernel was employed.]

7. ONLINE LEARNING

In the proposed algorithm, the robot uses data from a LIDAR scan to generate ground and nonground points for training of a vision-based classification algorithm. However, because the forest environment's appearance can change frequently because of variable illumination conditions or changes in vegetation type, the visual classifier should be periodically updated (i.e., retrained). Here, an online learning algorithm is employed to allow adaptation of the visual classifier to changing environmental appearance.

The online learning algorithm is based on the assumption that the drivable ground surface is simply connected in the image plane. Given this assumption, a metric to evaluate the quality of previous classification is employed. This metric is analyzed to determine whether to retrain the visual classifier.

7.1. Evaluation

To evaluate the performance of the classifier, a three-step process is used. First, morphological operations are used to identify areas that are likely to be either ground or nonground. Second, the classification output from the SVM is assessed to identify potentially misclassified points. Third, a numerical metric is calculated to determine the predicted classification accuracy. Note that this approach does not rely on hand-labeled training data, but rather on the assumption that the morphological operations will adequately separate ground regions from nonground regions.

Algorithm 1. AdaBoost and Fuzzy SVM for feature selection.

```
Given dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i$ is a feature vector of training point extracted by the
algorithm described in section 6-2 and $y_n \in \{-1,1\}$ refers to nonground and ground classes.

Initialize weights $w_i = \frac{1}{n}$, n is the number of training data.

Define $S=\{(x_1, y_1, w_i), \ldots, (x_n, y_n, w_n)\}$ as Fuzzy SVM training set
Initialize the selected feature set $F \in \emptyset$.
Initialize the candidate feature set C as all 15 features described in section 6.2.
For t=1,...,5
  1. For each feature j in feature set C, using a Gaussian radial basis function (RBF) kernel Fuzzy
     SVM to train a classifier $h_j$ on training set S. The error is evaluated with respect to $w_j$,
     $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  2. Choose the classifier, $h_t$, with the lowest error $\epsilon_t$, eliminate the corresponding feature from
     feature set C and add the corresponding feature into F.
  3. Update the weights:
         $w_i \leftarrow w_i \beta_t^{1-e_i}$
     Where $e_i = 0$ if example $x_i$ is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
  4. Normalize the weights,
         $w_i \leftarrow \frac{w_i}{\sum_{j=1}^{n} w_j}$,
```

7.1.1. Morphological Operation

Morphological operations (Soille, 1999) are commonly used to understand the structure of an image. Morphological operations play a key role in applications such as machine vision and automated object detection.

In this paper, the main morphological operations are flood filling and hole filling. These operations are implemented to determine the largest connected ground area and erode all the holes (i.e., nonground pixels) in that connected ground area. Then the largest connected ground region is labeled as ground and all the other regions are labeled as nonground regions. The process of morphological operation is illustrated in Figure 10. The process separates a classification result into two regions: a ground region and a nonground region.

It should be noted that the morphological operation has the risk of eroding small obstacles in front of the robot. To mitigate this problem in practice, we place the camera as low as possible. As a result, small obstacles that were erroneously eroded could easily be traversed by the robot.

7.1.2. Potentially Misclassified Points Detection

Classification results are then compared with the outputs of the morphological operation (illustrated in Figure 10). Points classified as ground by the classification stage that lie in the nonground regions of the morphological result are assumed to be potentially misclassified points. Also, points classified as nonground by the classification stage that lie in the ground regions of the morphological result are assumed to be potentially misclassified points. These potentially misclassified points are labeled as a new training set.

7.1.3. Classification Evaluation

An evaluation function is proposed to evaluate the performance of the classification result described in the previous section, and determine if the visual classifier should be retrained online. This evaluation function relies on the assumption mentioned in the previous section that the ground region is simply connected. The function is computed as

$$E_{\mathrm{AFP}} = \sum_{r=1}^{H}\sum_{c=1}^{W} V_1(r,c) \Big/ \sum_{r=1}^{H}\sum_{c=1}^{W} R_1^M(r,c) \tag{17}$$

$$E_{\mathrm{AFN}} = \sum_{j=2}^{N}\sum_{r=1}^{H}\sum_{c=1}^{W} V_j(r,c) \Big/ \sum_{j=2}^{N}\sum_{r=1}^{H}\sum_{c=1}^{W} R_j^M(r,c) \tag{18}$$

$$E_{\mathrm{AF}} = \sum_{j=1}^{N}\sum_{r=1}^{H}\sum_{c=1}^{W} V_j(r,c) \Big/ \sum_{j=1}^{N}\sum_{r=1}^{H}\sum_{c=1}^{W} R_j^M(r,c) \tag{19}$$

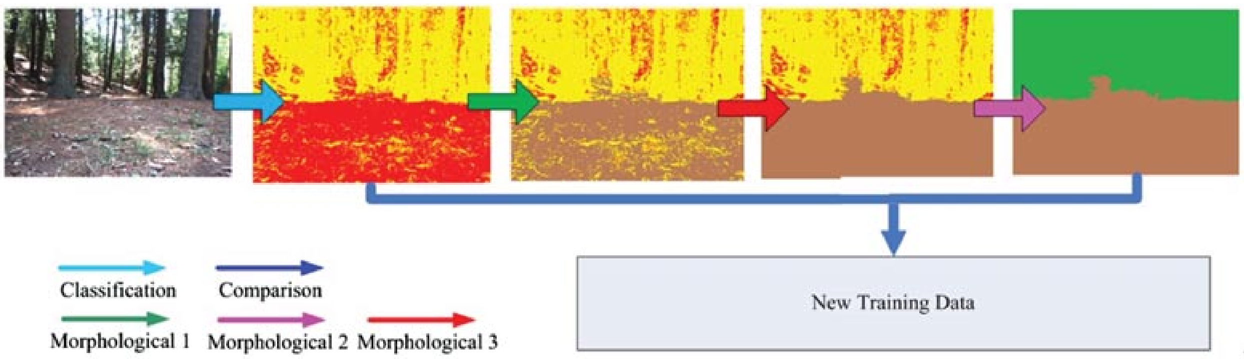


Figure 10. Morphological operation process. "Classification" shows the result obtained directly from the SVM classifier. The other four images illustrate various stages of morphological operations, such as flood filling and hole filling.

$$V_j(r,c) = \begin{bmatrix} 1, & R_j^C(r,c) \neq R_j^M(r,c) \\ 0, & R_j^C(r,c) = R_j^M(r,c) \end{bmatrix}, \quad r = 1,\ldots,H, c = 1,\ldots,W, j = 1,\ldots,N \Bigg], \quad (20)$$

where E_{AFP} refers to error rate of assumption-based false positive, E_{AFN} refers to error rate of assumption-based false negative, and E_{AF} refers to error rate of assumption-based classification false. N is the number of regions after the morphological operations. V_1 is the ground region, whereas $V_j, (j = 2,\ldots,N)$ refers to nonground region, (r,c) denotes a pixel position in the image and $R_j^C(r,c)$ is the value of the classification result at (r,c), whereas $R_j^M(r,c)$ is the morphological operation result at (r,c). $V_j(r,c)$ indicates whether $R_j^C(r,c)$ and $R_j^M(r,c)$ belong to the same class (ground/nonground). So the performance of the ground detection classifier is analyzed by computing the values of E_{AFP}, E_{AFN}, and E_{AF} derived from the formulas (17)–(20).

The values of E_{AFP}, E_{AFN}, and E_{AF}, can be compared with three user-defined threshold values, T_{AFP}, T_{AFN}, and T_{AF}. These threshold values determine the update rate for the online learning process. The baseline evaluation values E^0_{AFP}, E^0_{AFN}, and E^0_{AF} are calculated in an initial frame in which the visual classifier is trained directly by the LIDAR, and represent the best performance the system can be expected to reach. In this work, parameter values were chosen as $T_{\text{AFP}} = E^0_{\text{AFP}} + 0.02$, $T_{\text{AFN}} = E^0_{\text{AFN}} + 0.02$, and $T_{\text{AF}} = E^0_{\text{AF}} + 0.02$). We called the value of 0.02 the margin of error tolerance. In later frames, if any of E_{AFP}, E_{AFN}, and E_{AF} was larger than its corresponding threshold, the retraining process was triggered.

7.2. Online Learning

Given the comparison between the classification result and the morphological operation, potentially misclassified points can be extracted. Those points can be considered as a new training set and can be inserted into the training set database for online retraining. To bound the size of the training database, old training points may be discarded, with an equal number of new training data added from the set of potentially misclassified points. There are three key questions related to this process: how to reduce existing training data, how to add new training data, and how to filer ambiguous points.

1. *Reducing the old training data*. An SVM decision function can generally be described as follows (Burgers, 1998):

$$f(x) = \sum a_i y_i K(x_i, x) + b \quad (21)$$

$$D(x) = f(x)/y, \quad (22)$$

where K is the kernel function, x_i is the support vector and y_i refers to its corresponding class, and b is the offset. The decision function is used to determine which class x belongs to. The value of $D(x)$ reflects the distance from x to the decision hyperplane. Using (21) and (22) to compute the values of all training data, we can find their distances to the decision hyperplane. Because the decision hyperplane is composed of support vectors that are not far from the decision hyperplane, the greater the distance, the less contribution it has to determining the decision hyperplane. Given this, we discard the old training points according to their contributions.
2. *Adding new training data*. In each frame, many potentially misclassified points are considered as candidates. It is obvious that all the distances of these points to the decision hyperplane are negative, because they are misclassified points (based on the morphological assumptions). The smaller the value of the distance, the more contribution it has to determining the new decision hyperplane. In that case, the decision on how to select new

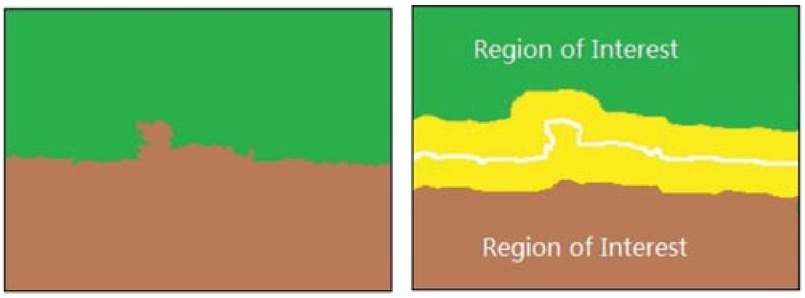


Figure 11. Illustration of regions of interest for learning.

training data is based on the contributions of those candidate points.

3. *Filtering ambiguous points*. As mentioned above, potentially misclassified points are generated by comparing classification results and morphological results. This heuristic is least reliable near the boundary separating the ground and background regions. To minimize the error caused by training with mislabeled data, points nearer to the morphological operation boundary than a specified threshold are not considered as new training data (see Figure 11). In this paper, the threshold is set as 20 pixels from the boundary.

8. EXPERIMENTAL RESULTS

The proposed method for visual detection of the terrain surface has three-stages: ground detection from 3D LIDAR data, self-supervised visual learning based on the result of LIDAR data points, and online learning.

To characterize the performance of this approach, the algorithm was applied to numerous experimental data sets. Experimental results of the first stage are presented in Section 8.1. Results from the second and third stages are demonstrated in Sections 8.2 and 8.3. Results from two additional forest scenes are presented in Section 8.4.

8.1. Experiments of Ground Detection from LIDAR Data

LIDAR data were acquired for five different forest environments in the greater Boston area. In this paper, these five data sets will be used to demonstrate the effectiveness of the classification and modeling techniques that are presented. The first set of data was collected in a simple and relatively flat forest environment. These data are meant to be a baseline set that will be compared with the other, more complex scenes. The data set contains several deciduous trees and shrubs, but is largely open. The moderate scene is more cluttered with numerous deciduous trees and shrubs and significant ground cover. The remaining two data sets, dense1 and dense2, contain significantly more deciduous trees than the previous data sets, and also have considerable sloped terrain. In the dense1scene, there is a total elevation change of 4.2 m and the LIDAR is scanning down the hill. In the dense2, the total change is 3.1 m and the LIDAR is scanning up the hill.

There are two stages in LIDAR-based classification. The results in each stage are presented below.

1. *Results of Stage 1*. Stage 1 applies a local minimum z filter. This filter drastically reduces the number of points to be analyzed. It is important to note that each column in Cartesian space may contain many ground points, but only one minimum z point. Therefore, many ground points may be removed. Also, sometimes a higher fraction of ground points are removed than of nonground points, due to the fact that the ground near the LIDAR is scanned much more densely than trees and shrubs in the distance.

   To assess the accuracy of this method, it is useful to analyze the types of points present in each column. For example, columns that are farther away from the LIDAR may not have any ground points because the ground is occluded by trees and shrubs. However, if a column contains both ground and non-ground points, a ground point should be selected if the algorithm is robust. These results are summarized in Table I.

   As expected, there are many columns in each scene where there are only nonground points. However, in the columns with both ground and nonground points, this filter stage selects ground points an average of 98.65% of the time.

2. *Results of Stage 2*. The second stage of LIDAR-based classification uses a SVM classifier that is trained using hand-labeled point cloud data sets to assign each of the minimum z points as being ground or nonground. The results are demonstrated in Table II.

   The results suggest that this classifier can effectively and consistently identify points belonging to the ground class. The average accuracy over the five data sets is 86.28%.

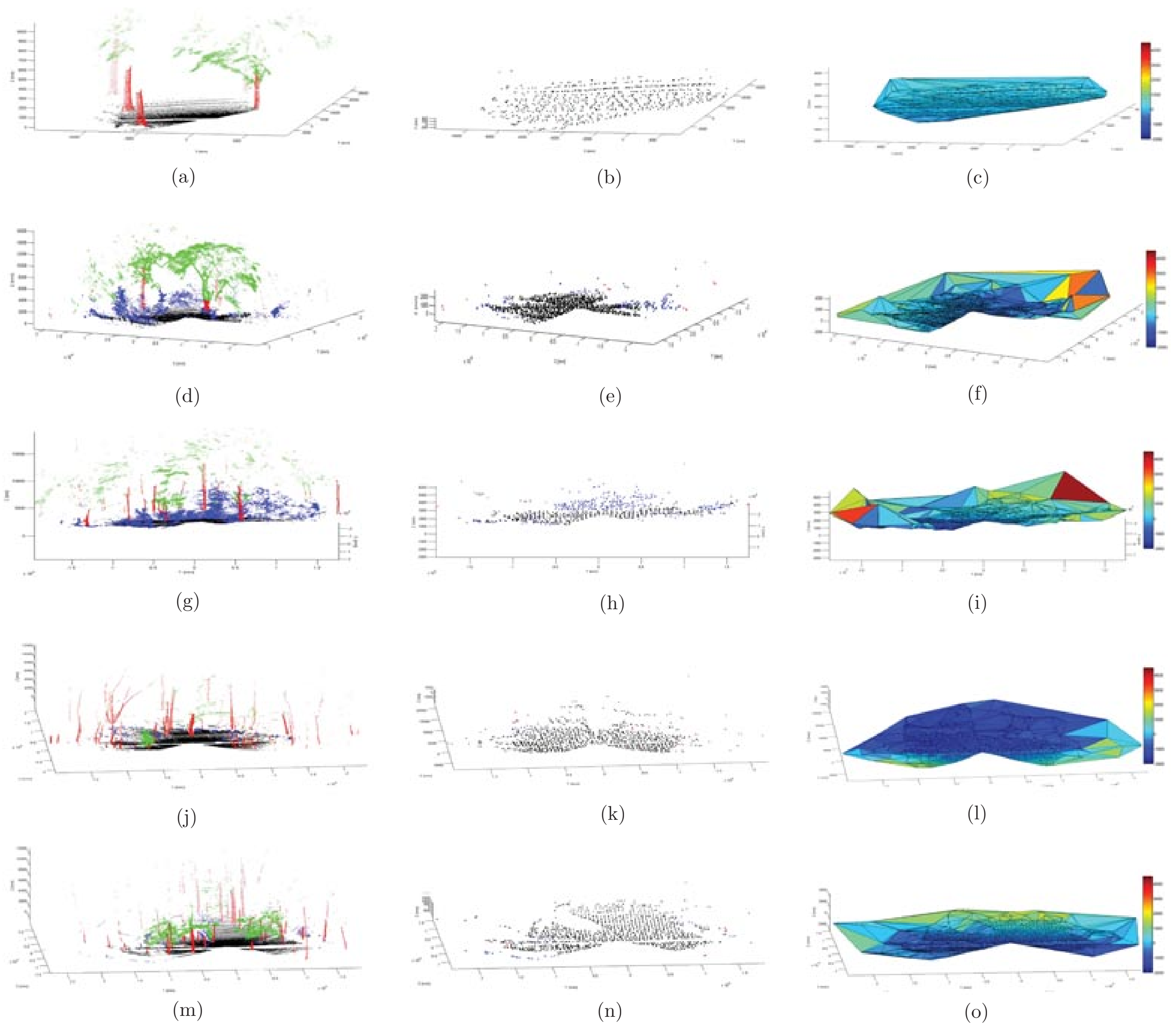


Figure 12. Hand-labeled point clouds of (a) baseline scene, (d) sparse scene, (g) moderate scene, (j) dense1 scene and (m) dense2 scene. Black points indicate ground, blue points indicate bushes/shrubs, red points indicate trunks, and green points indicate canopy. (b), (e) ,(h), (k) and (n) are ground point estimation results in each scene. Each includes true positive and false positive points. (c), (f) ,(i), (l) and (o) are ground surface of TIN in different scenes. (Image best viewed in color.)

It is helpful to visualize these results using point clouds. Figures 12(c, d, g, j, m) demonstrate the hand-labeled data sets that are used as ground truth. Figures 12(b, e, h, k, n) show the LIDAR data classified as ground by the SVM. This includes both true positives (ground points correctly classified as ground) and false positives (nonground points incorrectly classified as ground). Figure 13 shows the receiver operating characteristic (ROC) curves for the five data sets. As shown in Figure 13, fairly low false positive rates can be achieved while relatively high true positive rates are maintained.

8.2. Experiments of Self-Supervised Visual Learning

As mentioned in Section 6, there are three stages in self-supervised visual learning: training data generation, visual feature selection, and visual classifier training. The results in each stage are presented below.

Table I. Stage 1 local minimum z filter: Column analysis.

	No. of initially Sensed 3D points	No. of points after filtering	Columns with ground data	Columns with nonground data	Columns with both and ground selected	Columns with both and nonground selected	Accuracy of columns with both
baseline	69,598	938	542	261	135	0	100%
sparse	141,785	2,309	509	1,174	622	4	99.35%
moderate	154,403	1,625	251	1,176	193	5	97.41%
dense1	112,988	1,521	815	313	384	9	97.66%
dense2	159,023	1,574	694	352	521	7	98.66%
Total	637,797	7,967	2,811	3,276	1,855	25	98.65%

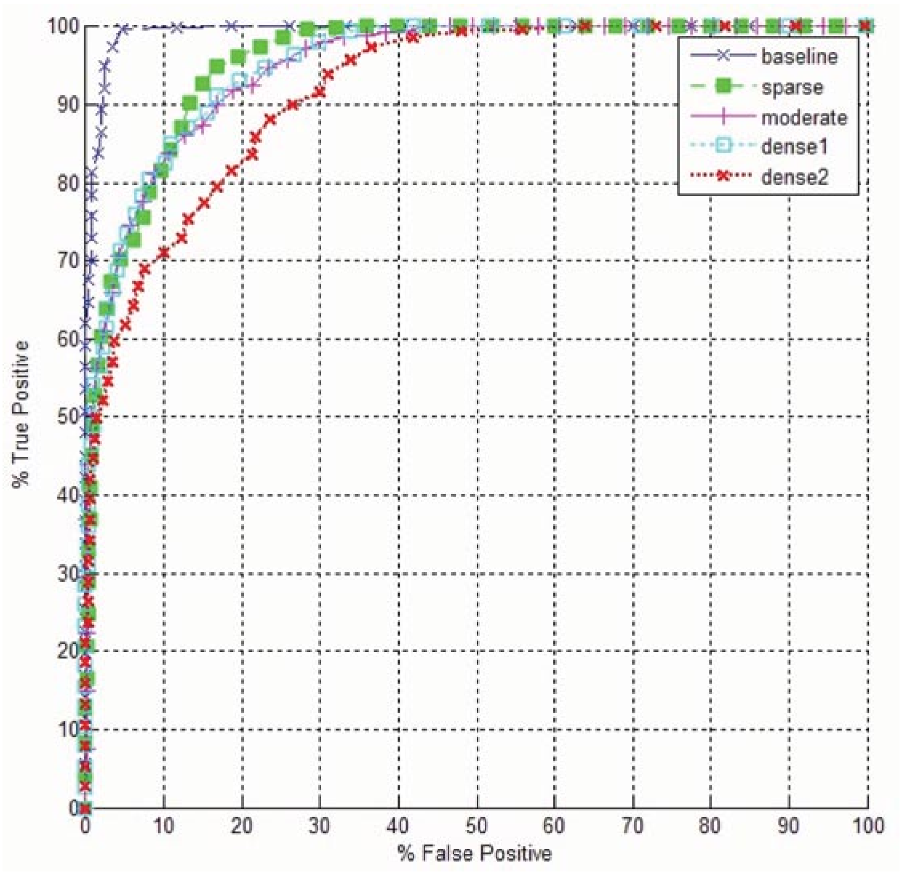


Figure 13. ROC curves for baseline, sparse, moderate, dense1, and dense2 scenes.

1. *Results of training data generation*. In Section 6.1, Delaunay triangulation is introduced to model the surface of the ground plane. In the experiments, each scene was modeled by TIN to create the surface of the ground plane from a set of points. Figures 12 (c, p, i, l, o) show the resulting triangulation for each scene, in which the color of each triangle is proportional to its height.

   Additional training data are generated by comparing the height of the ground plane with the actual sensed 3D LIDAR points. The points that are lower than the plane are selected as ground training data, whereas

Table II. Stage 2 results of classification.

	True ground classified as ground	True bushes and shrubs classified as ground	True tree stem classified as ground	true tree canopy classified as ground	Accuracy of columns with both
baseline	677/666	0/0	26/10	232/0	97.75%
sparse	1,131/1,032	463/149	47/12	664/2	88.63%
moderate	444/441	752/420	50/17	379/6	72.55%
dense1	1,199/1,131	56/39	198/30	68/2	90.86%
dense2	1,215/1,077	164/67	105/19	89/0	98.66%
Total	4,666/4,347	1,435/675	426/88	1,432/10	86.26%

the points that are higher than the plane are selected as nonground training data.

2. *Results of feature selection and classification of visual classifier*. A comparison is presented here to demonstrate the efficacy of the AdaBoost and Fuzzy SVM-based feature selection algorithm in visual classification. In this experiment, the training set is generated by the algorithm described in Sections 5 and 5.1. Given a training set, three individual classifiers are trained for ground prediction. Those results are compared to a hand-labeled ground truth for comparison. The results are shown in Figure 14.

   Here, the following SVM classifier formulations were compared, for use in vision-based ground detection.

   - Selected features. The classifier was trained with the five best features selected by the AdaBoost-based algorithm.
   - All features. The classifier was trained with all 15 of the candidate visual features.
   - Remaining features. The classifier was trained with the 10 features that were not selected by the AdaBoost-based algorithm.

   Figures 14 (a, b, c, d) show the classification results of the classifier trained by the five best selected features. The selected features classifier (using 5 features) was observed to perform well compared to the all-features classifier that employed 15 features (see Figures 14 (e, f, g, h)). However, the selected features classifier exhibits improved computational efficiency because it requires fewer feature extraction operations than the all features classifier. The remaining features classifier (trained on the 10 features that were not selected) performs more poorly than the other two classifiers.

   Note that in the second and fourth scenes, the selected features classifier performs slightly better than the all features classifier in terms of detection rate. We expect that this is because the additional features introduce confounding information for separating ground/nonground classes. Also, it should be noted that the features selected vary from scene to scene, based on the specific visual properties of each individual environment.

8.3. Experiments with Online Learning

The proposed approach to ground detection by visual sensors includes two phases: an initial training phase and an online learning phase. In the initial phase, the visual classifier is trained based on LIDAR-identified training data. In the online phase, visual classifier training data are augmented to include results from online learning based on morphological analysis. The approach was applied to experimental data sets that were taken in the Breakheart Reservation, which is located south of Boston, Massachusetts. Numerous data sets were collected.

For each set, the frame rate was 1 Hz. The robot moved at 1 m/s and traveled approximately 130 m in the forest. Figure 15 shows the performance of the online learning approach as compared to a visual classifier trained based solely on LIDAR data captured in the first frame. The appearance of the environment changed because of changes in illumination during the experiment. As can be seen in the figure, the accuracy of the fixed classifier, shown in red, deteriorated steadily. In contrast, the accuracy of the classifier using online learning remained high throughout the experiment, demonstrating that it is capable of adapting to the changing environment.

In the first forty frames, the average detection accuracy of the online learning algorithm was 85.62%, whereas the accuracy without the online learning algorithm was 84.73%. Both algorithms performed relatively well, because the environment appearance did not change dramatically during this initial period.

During frames 41–126, the environment illumination changed significantly. During this period, the performance of the nononline learning algorithm degraded, whereas the online learning algorithm maintained good performance. Table III compares the results of the two algorithms.

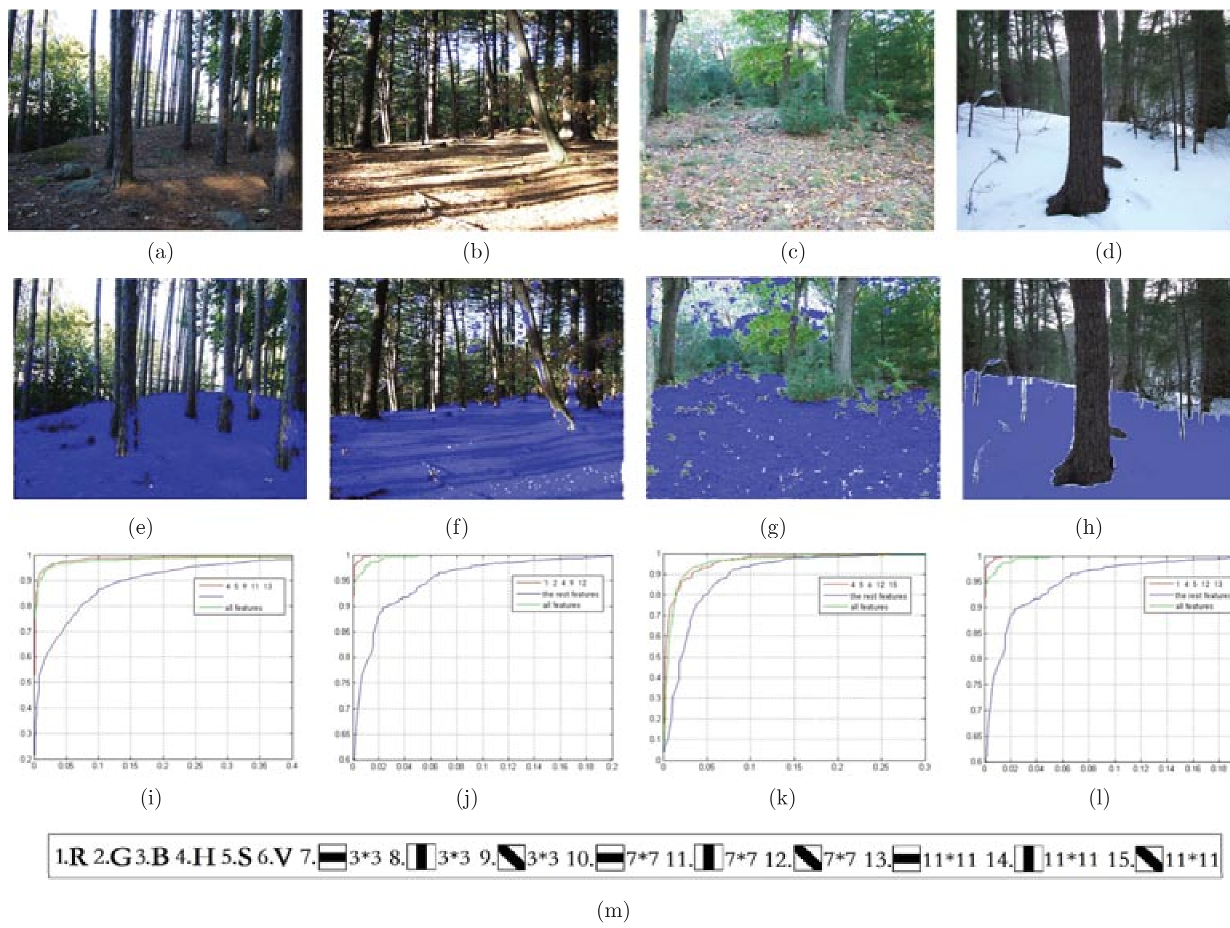


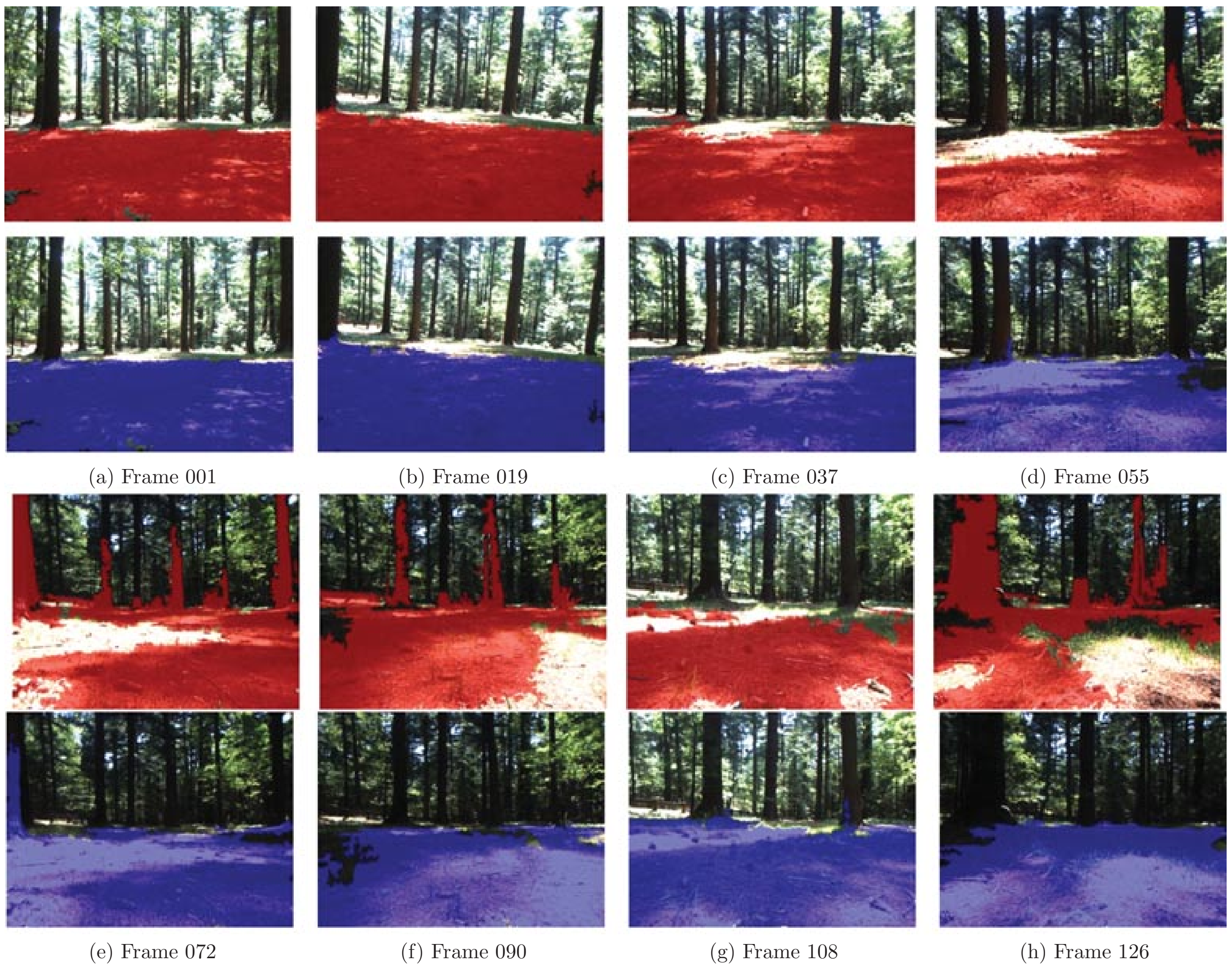
Figure 14. Classification result and classification comparison: (a), (b), (c), (d) are raw image data captured from four different scenes. (e), (f), (g), (h) are results classified by the "selected features" classifier. (i), (j), (k), (l) are ROCs of the three classification results. (The red line refers to the "selected features" classifier. The number refers to the selected feature number. The green line refers to the "all features" classifier. The blue line refers to the "remaining features" classifier. (m) is the feature map, where the first six features are (red, green, and blue) in the RGB color space, then (hue, saturation, and value) in HSV color space. The final nine features are Haar features in vertical, horizontal, and diagonal directions at 3 × 3, 7 × 7 and 11 × 11 scales.

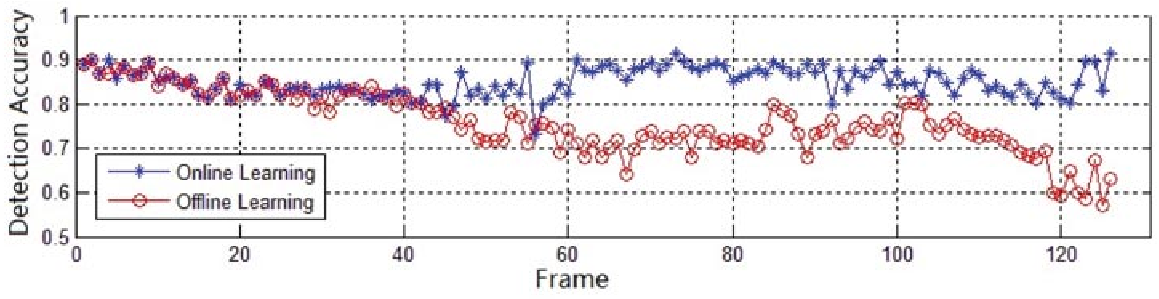
Table III. Comparison results of online learning algorithm and nononline learning algorithm.

	Online learning	Nononline learning
First frames	90.12%	90.12%
1–40 frames	85.62%	84.73%
41–126 frames	83.21%	73.62%
Online learning times	21	0
Average accuracy	83.96%	77.15%

Experiments from two additional traverses are presented here. These experiments were performed in two separate regions in forest with increased terrain slope, compared to the experiment described in the previous section.

In the first scenario (as shown in Figure 16), the robot moved from relatively flat terrain to sloped terrain (approximately 10 deg). During robot motion, illumination and vegetation properties changed. In Figure 16(a) the blue points are raw LIDAR points that fall outside of the camera view. The green points are raw LIDAR points that can be projected into the image. The red points are LIDAR points that are classified as ground points using the LIDAR-based classification and training data generation presented in Section 5 and Section 6.1. The right image of Figure 16(a) shows the training samples projected from LIDAR points into image pixels. Given these training data, the 1st, 2nd, 6th, 11th, and 15th features were selected. Using these selected features and training samples, the online learning results are shown in Figure 16(b). Raw and classified images with index numbers are also shown in Figure 16(b).

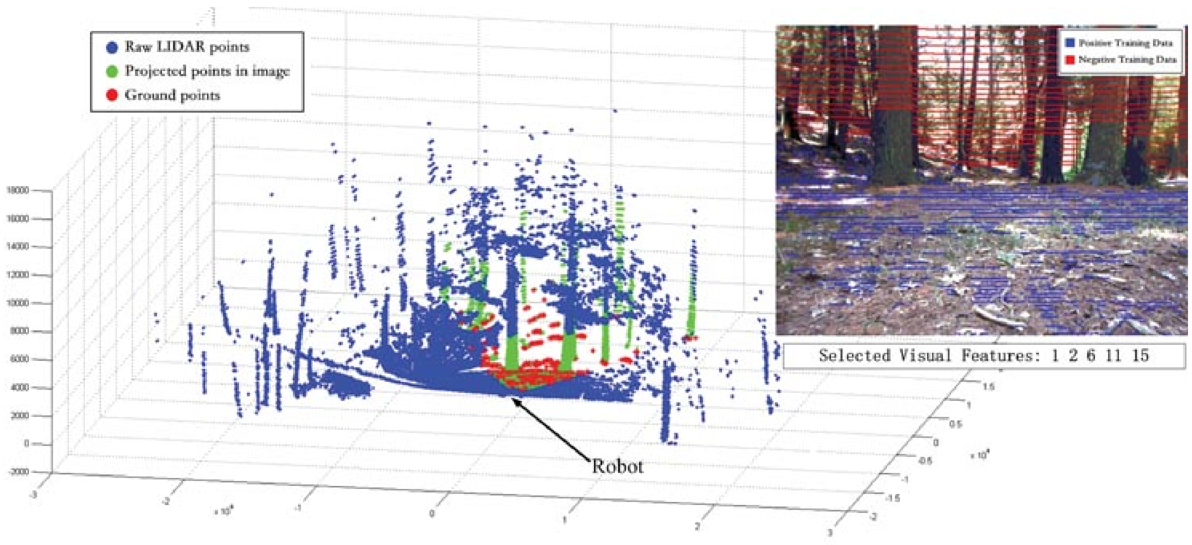




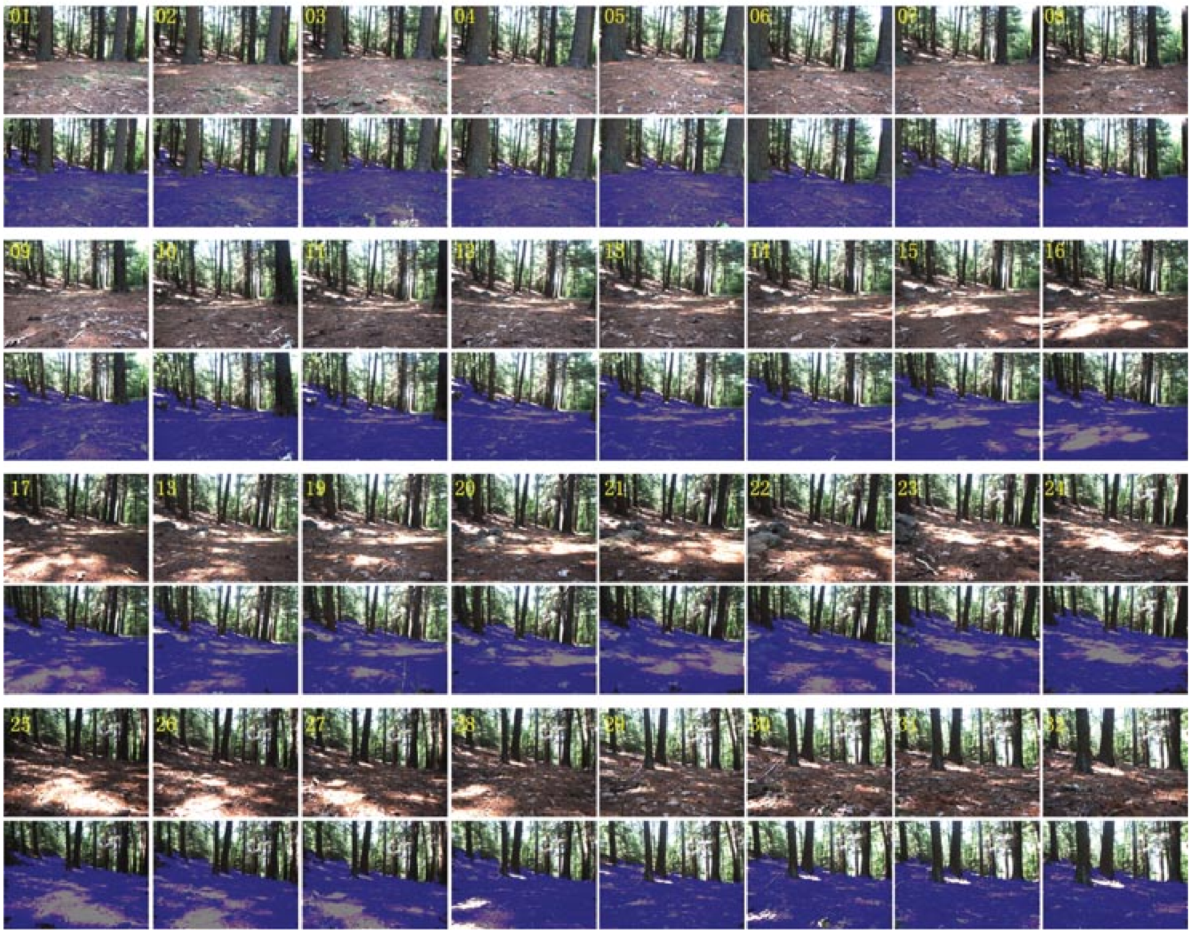
(i)

Figure 15. (a–h) The selected visual detection results from consecutive frames, (Red results are generated from the initial classifier trained by 3D LIDAR points mentioned in Section 6 without online learning, whereas blue refers to online learning.) (i) The detection accuracy of those results.

For this data set, the self-supervised learning method exhibited an average accuracy of 84.36%, whereas the results for the method including morphological operations exhibited an average accuracy of 94.83%. A traditional supervised classifier (trained once, at the initial frame, by LIDAR data) exhibited an average accuracy of 72.54%. Performance of the traditional supervised classifier was observed to degrade after the 13th frames, likely because of

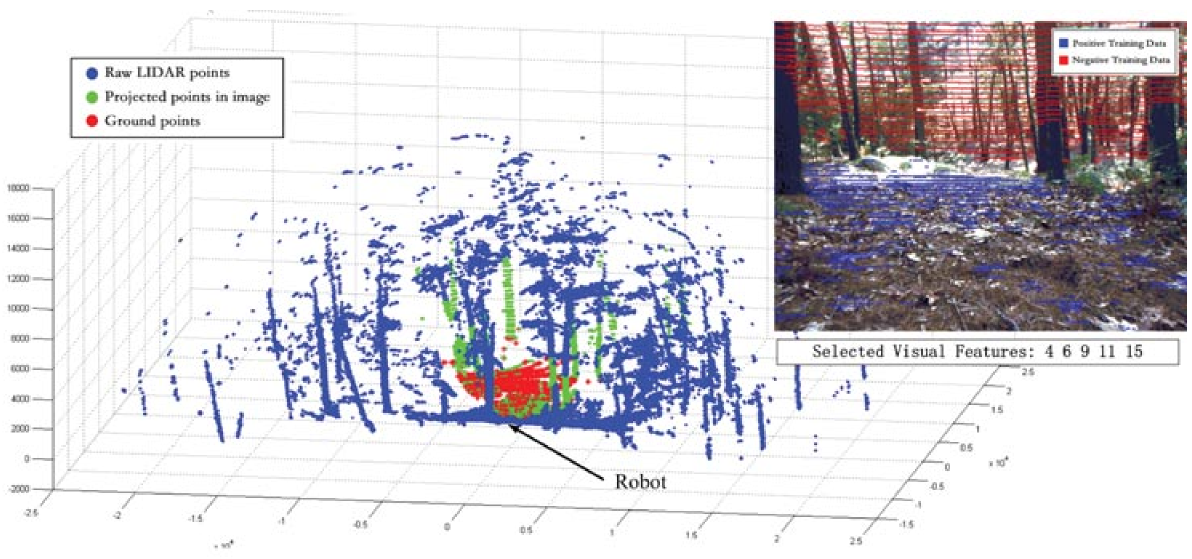


(a)

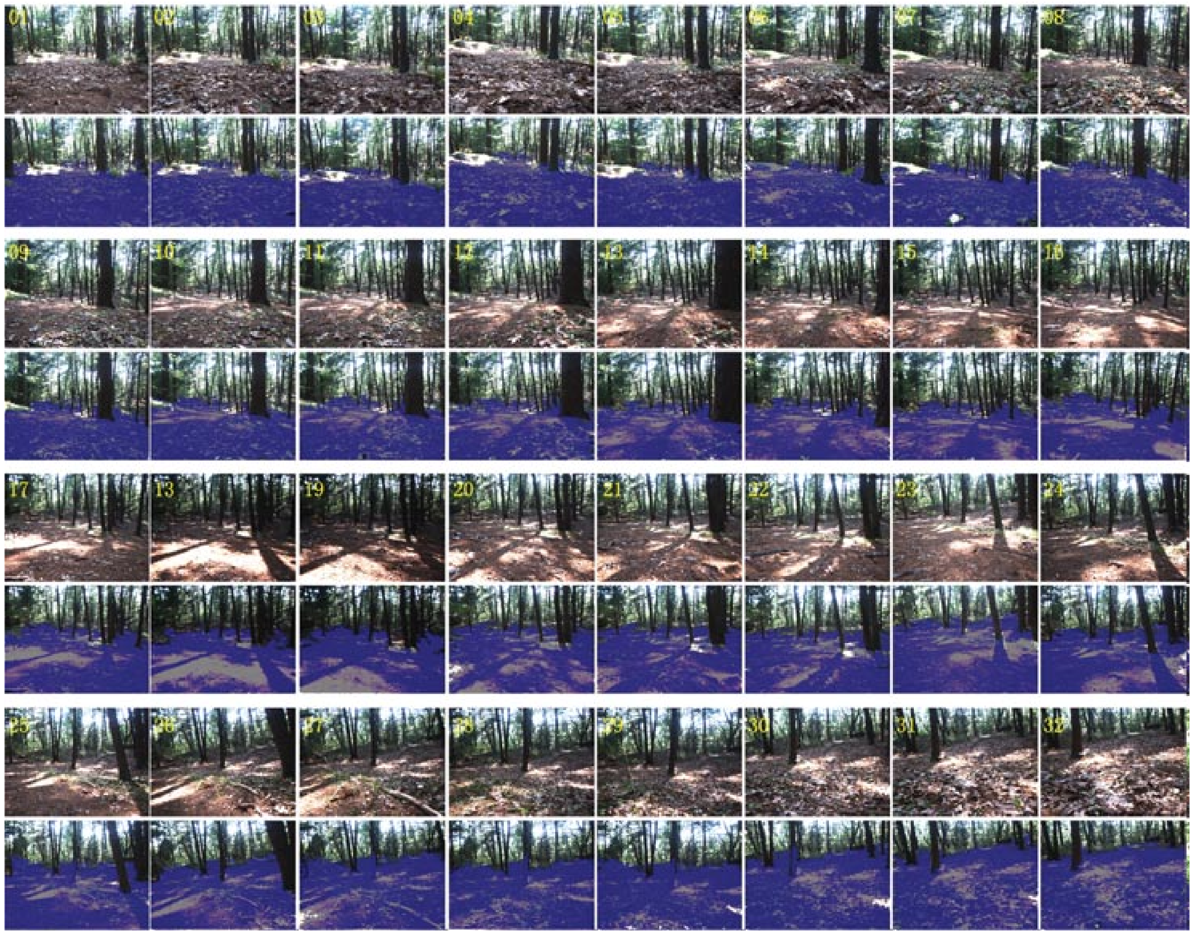


(b)

Figure 16. Experiment in sloped terrain in forst: (a) labeled data from LIDAR sensor and projection into image; (b) indexed raw and classifed images.



(a)



(b)

Figure 17. Experiment in dense and bumpy environment in forest; (a) labeled data from LIDAR sensor and projection into image; (b) indexed raw and classified images.

significant changes in illumination (i.e., direct sunlight passing through the canopy to illuminate the forest floor).

In the second scenario (shown in Figure 17), the robot traveled through a region with denser vegetation and a significantly rougher ground surface. Again, illumination changed significantly during the experiment. In Figure 17(a) the blue points are raw LIDAR points that fall outside of the camera view. The green points are raw LIDAR points that can be projected into the image. The red points are LIDAR points that are classified as ground points using LIDAR-based classification and training data generation presented in Section 5 and Section 6.1. The right image of Figure 17(a) shows the training samples projected from LIDAR points into image pixels. Given these training data, the 4th, 6th, 9th, 11th, and 15th features were selected. Using these selected features and training online learning results are shown in Figure 17(b). Raw and classified images with index numbers are also shown in Figure 17(b).

For this data set, the self-supervised learning method exhibited an average accuracy of 81.46%, whereas the method using morphological operators exhibited an average accuracy of 95.21%. A traditional supervised classifier (trained once, at the initial frame, with LIDAR data) exhibited an average accuracy of 74.27%. Performance of the traditional supervised classifier was observed to degrade after the 12th frame, likely because of the combined effects of changes in illumination and vegetation properties.

Collectively, these experiments demonstrate the effectiveness of the proposed methodology. Specifically, they show that the online learning algorithm is able to maintain good performance despite changing environmental conditions. This is in contrast to a traditional supervised classifier, which exhibits poor robustness to changing conditions. Also, these experiments demonstrate the utility of employing morphological operations to improve the performance of online learning methods. This is an important issue for systems that employ learning, or automatic training, because presenting poor training examples to a learning system can substantially degrade performance.

9. CONCLUSION

The paper has presented an approach to self-supervised visual learning for terrain surface detection in forest environments. By the application of feature extraction on a 3D point cloud, the ground surface can be assigned using a traditional supervised SVM classifier. Then ground plane modeling using Delaunay triangulation is used to generate adequate training samples and initialize a self-supervised visual classifier. The method exploits the accuracy of LIDAR sensors to accurately generate 3D point clouds, and the high frame rate of a visual sensor for classifying terrain. A novel approach to morphological operations is implemented to evaluate the performance of the current visual classifier and to activate an online learning process for retraining the visual classifier. This allows the robot to adapt to environment changes, and improves the performance of vision-based navigation in forest environments. Also, in the feature selection algorithm, the paper applies the Fuzzy SVM classifier as a weak learner in the AdaBoost process, which improves the capabilities of AdaBoost in feature selection.

APPENDIX: INDEX TO MULTIMEDIA EXTENSIONS

The videos shown in the table are available as Supporting Information in the online version of this article.

Extension	Media type	Description
1	Video	Sensor system experiment video
2	Video	Experimental results for terrain surface detection

ACKNOWLEDGMENTS

This work was supported by the U.S. Army Research Office (Contract W912HZ-08-C-0060) and the U.S. Army Topographic Engineering Center (contract W911NF-07-1-0540).

REFERENCES

Broggi, Alberto, Caraffi, Claudio, Fedriga, Rean Isabella, & Grisleri, Paolo. (2005, June). Obstacle detection with stereo vision for off-road vehicle navigation. In Proceedingss. International. IEEE Workdings on Machine Vision for Intelligent Vehicles, San Diego, USA.

Burgers Christopher, J. C. (1998). A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2, 121–167.

Chang, C., & Lin, C. (2008, October). LIBSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm/. Accessed on October 2008.

Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., & Bradski, G. (2006, August). Self-supervised monocular road detection in desert terrain. In Proceedings of robotics: Science and systems, Philadelphia, USA.

Elmqvist, M. (2002). Ground surface estimation from airborne laser scanner data using active shape models. International Archives of Photogrammetry and Remote Sensing, 34, 114–118.

Freund, Y., & Schapire, R. E. (1997, August). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119–139.

Hadsell, Raia, Sermanet, Pierre, Ben, Jan, Erkan, Ayse, Scoffier, Marco, & Kavukcuoglu, Koray. (2009). Learning long-range vision for autonomous off-road driving. Journal of Field Robotics, 26(2), 120–144.

Hebert, M., & Vandapel, N. (2003, May). Terrain classification techniques from ladar data for autonomous navigation. Paper presented at the Collaborative Technology Alliances Conference, Marryland, USA.

Kelly, Alonzo, & Stentz, Anthony (1998, May). Stereo vision enhancements for low-cost outdoor autonomous vehicles. Paper presented at the International Conference on Robotics and Automation, Workshop WS-7, Navigation of Outdoor Autonomous Vehicles (ICRA'98).

Kong, H., Audibert, J.-Y., & Ponce, J. (2009, June). Vanishing point detection for road detection. Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), Florida, USA.

Konolige, K., Agrawal, M., Blas, M. R., Bolles, R. C., Gerkey, B., Sola, J., & Sundaresan, A. (2009). Mapping, navigation, and learning for off-road traversal. Journal of Field Robotics, 26(1), 88–113.

Lalonde, J.-F., Vandapel, N., Huber, D., & Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. Journal of Field Robotics, 23(10), 839–861.

Lin, C. F., & Wang, S. D. (2002, May). Fuzzy support vector machines. IEEE Transactions on Neural Networks, 13(2), 464–471.

Manduchi, R., Castano, A., Talukder, A., & Matthies, L. (2004). Obstacle detection and terrain classification for autonomous off-road navigation.autonomous Robots. 18, 81–102.

McCall, Joel, C., & Trivedi, Mohan, M. (2006). Video based lane estimation and tracking for driver assistance: survey, system, and evaluation. IEEE Transactions on Intelligent Transportation Systems, 7(1) 20–37.

McDaniel, M. W., Nishihata, T., Brooks, C. A., & Iagnemma, K., (2010, May). Ground plane identification using LIDAR in forested environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK.

Rasmussen, C. (2002, May). Combining laser range, color, and texture cues for autonomous road following. In Proceedings IEEE International Conference on Robotics and Automation, Washington, DC.

Sofman, B., Lin, E., Bagnell, J., Cole, J., Vandapel, N., & Stentz, A. (2007). Improving robot navigation through self-supervised online learning. Journal of Field Robotics, 23(12), 1059–1075.

Soille, P. (1999). Morphological image analysis: Principles and applications. Springer-Verlag, California, USA.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L., Koelen, C., Markey, C., Rummel, C., Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., & Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(1), 661–692.

Unnikrishnan, R., & Hebert, M. (2005, July). Fast extrinsic calibration of a laser rangefinder to a camera. (Tech. Rep. CMU-RI-TR-05-09) Carnegie Mellon University, Robotics Institute, Pittsburgh, USA.

Viola, Paul, & Jones, Michael, J. (2001). Rapid object detection using a boosted cascade of simple features. Paper presented at the IEEE International Conference on Computer Vision and Pattern Recognition.

Wellington, Carl, Courville, Aaron, C., & Stentz, Anthony. (2006, December). A generative model of terrain for autonomous navigation in vegetation. International Journal of Robotics Research, 24, 83–92.

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

Zhou, S., & Iagnemma, K. (2010, October). Self-supervised learning method for unstructured road detection using fuzzy support vector machines, Paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan.