Auton Robot (2008) 24: 69–86
DOI 10.1007/s10514-007-9063-6

Learning traversability models for autonomous mobile vehicles

Michael Shneier · Tommy Chang · Tsai Hong · Will Shackleford · Roger Bostelman · James S. Albus

Received: 13 December 2005 / Accepted: 15 October 2007 / Published online: 21 November 2007

Abstract Autonomous mobile robots need to adapt their behavior to the terrain over which they drive, and to predict the traversability of the terrain so that they can effectively plan their paths. Such robots usually make use of a set of sensors to investigate the terrain around them and build up an internal representation that enables them to navigate. This paper addresses the question of how to use sensor data to learn properties of the environment and use this knowledge to predict which regions of the environment are traversable. The approach makes use of sensed information from range sensors (stereo or ladar), color cameras, and the vehicle's navigation sensors. Models of terrain regions are learned from subsets of pixels that are selected by projection into a local occupancy grid. The models include color and texture as well as traversability information obtained from an analysis of the range data associated with the pixels. The models are learned without supervision, deriving their properties from the geometry and the appearance of the scene.

The models are used to classify color images and assign traversability costs to regions. The classification does not use the range or position information, but only color images. Traversability determined during the model-building phase is stored in the models. This enables classification of regions beyond the range of stereo or ladar using the information in the color images. The paper describes how the models are constructed and maintained, how they are used to classify image regions, and how the system adapts to changing environments. Examples are shown from the implementation of this algorithm in the DARPA Learning Applied to Ground Robots (LAGR) program, and an evaluation of the algorithm against human-provided ground truth is presented.

Keywords Learning · Traversability · Classification · Color models · Texture · Range · Mobile robotics

M. Shneier (✉) · T. Chang · T. Hong · W. Shackleford · R. Bostelman · J.S. Albus
National Institute of Standards and Technology, Gaithersburg, MD 20899, USA
e-mail: Michael.Shneier@nist.gov

T. Chang
e-mail: Tommy.Chang@nist.gov

T. Hong
e-mail: Tsai.Hong@nist.gov

W. Shackleford
e-mail: Will.Shackleford@nist.gov

R. Bostelman
e-mail: roger.bostelman@nist.gov

J.S. Albus
e-mail: james.albus@nist.gov

1 Introduction

If autonomous mobile robots are to become more generally useful, they must be able to adapt to new environments and learn from experience. To do so, they need a way to store pertinent information about the environment, recall the information at appropriate times, and reliably match stored information with newly-sensed data. They also must be able to modify the stored information to account for systematic changes in the environment.

The Defense Advanced Research Projects Agency's (DARPA) Learning Applied to Ground Robots (LAGR) program (Jackel et al. 2006) aims to develop algorithms that will let a robotic vehicle travel through complex terrain without having to rely on hand-tuned algorithms that only apply in limited environments. The goal is to enable the

Fig. 1 The small robot used in the DARPA LAGR program

control system of the vehicle to learn which areas are traversable and how to avoid areas that are impassable or that limit the mobility of the vehicle. To accomplish this goal, the program provided small robotic vehicles to each of the participants (Fig. 1). The vehicles are used by the teams to develop software. A separate LAGR Government Team, with an identical vehicle, conducts tests of the software each month. This paper describes the approach taken by the NIST team to address one part of the problem—learning how appearance relates to traversability.

An overview of the NIST approach is given in Albus et al. (2006). It makes use of data from range sensors, color cameras, and position sensors to describe regions in the environment around the vehicle and to associate a cost of traversing each region with its description. Models of the terrain are learned using a scheme that makes use of both geometric and appearance information. The vehicle runs using a control hierarchy called 4D/RCS (Albus and Meystel 2001; Albus et al. 2002). 4D/RCS provides a hierarchical organization of control nodes, each of which divides the system into sensory perception (SP), world modeling (WM) and behavior generation (BG) subsystems. Each 4D/RCS node is designed to carry out specific duties and responsibilities. Each node is assigned a specified span of control, both in terms of supervision of subordinates, and in terms of range and resolution in space and time. Interactions between SP, WM, and BG give rise to perception, cognition, and reasoning. At lower levels in the hierarchy, representations of space and time are short-range and high-resolution. At nodes higher in the hierarchy, representations of space and time are long-range and low-resolution. This enables high-precision fast-action response from the low level control nodes, while higher level nodes are generating long-range plans and abstract concepts over broad regions of time and space. Typically, planning horizons expand by an order of magnitude in time and space at each higher level in the hierarchy. Within the WM of each node, a knowledge database provides a model of the external world at a range and resolution that is appropriate for the behavioral decisions that are the responsibility of that node.

This paper is concerned with the sensory processing and world modeling aspects of the hierarchy. It discusses the processing of multiple sensor inputs to generate models of terrain, and construction of traversability maps which are sent to the world model. There they provide input to path planners that generate trajectories to take the vehicle to its goal.

The vehicle provided by DARPA is a small but very capable robot with substantial on-board processing capacity and a rich set of sensors. The sensors include two pairs of color cameras mounted on a turret on the front of the vehicle, a pair of infra-red range sensors (non-contact bumpers) on the front of the vehicle, and a physical bumper centered on the front wheels of the vehicle. For position sensing, the vehicle has a Global Positioning System (GPS) sensor, wheel encoders, and an inertial navigation system (INS). In addition, there are sensors for motor current, battery level, and temperature. There are four single-board computers on the vehicle, one for low-level vehicle control, one for each of the stereo camera pairs, and one for overall control of the vehicle. All processors use the Linux operating system. The vehicle has an internal Ethernet network connecting the processors, and a wireless Ethernet link to external processors.

The availability of range information enables a robot to navigate largely using the geometry of a scene. Another viable approach is to use topology of the surrounding space (DeSouza and Kak 2002). In this approach, the relationships between features are used to describe the world rather than their precise geometry or quantitative measurements. We chose to use geometry to describe the world because it fits well with the 4D/RCS control architecture and it makes map-based path planning straightforward. Sensor processing is aimed at determining where the vehicle is and what parts of the world around it are traversable. The robot can then plan a path over the traversable region to get to its goal. Where range information is missing or unreliable, navigation is not so straightforward because it is less clear what constitutes clear ground. A typical range sensor will not be able to provide reliable range very far in front of the vehicle, and it is part of the aim of this work to extend the traversability analysis beyond the range sensing limit. This is done by associating traversability with appearance, under the assumption that regions that look similar will have similar traversability. Because there is no direct relationship between traversability and appearance, the system must learn the correspondences from experience.

The appearance of regions in an image has been described in many ways, but most frequently in terms of color and/or texture. Ulrich and Nourbakhsh (2000b) used color imagery to learn the appearance of a set of locations to

enable a robot to recognize where it is. A set of images was recorded at each location and served as a description of that location. Images were represented by a set of one-dimensional histograms in both HLS (hue, luminance, saturation) and normalized Red, Green, and Blue (RGB) color spaces. When the robot needed to recognize its location, it compared its current image with the set of images associated with locations. To compare histograms when matching images, the Jeffrey divergence was used (Puzicha et al. 1997). The location was recognized as that associated with the best-matching stored image.

In Ulrich and Nourbakhsh (2000a) the authors also addressed the issue of appearance-based obstacle detection using a single color camera and no range information. Their approach makes the assumptions that the ground is flat and that the region directly in front of the robot is ground. This region is characterized by color histograms and used as a model for ground. In the domain of road detection, a related approach is described in Tan et al. (2006). In principle, the method could be extended to deal with more classes, and our algorithm can be seen as one such extension that does not need to make the assumptions because of the availability of range information for regions close to the vehicle.

Learning has been applied to computer vision for a variety of applications, including traversability prediction. Wellington and Stentz (2003) predicted the load-bearing surface under vegetation by extracting features from range data and associating them with the actual surface height measured when the vehicle drove over the corresponding terrain. The system learned a mapping from terrain features to surface height using a technique called locally weighted regression. Learning was done in a map domain. We also use a map in the current work, although it is a two dimensional (2D) rather than a three dimensional (3D) map, and we also make use of the information gained when driving over terrain to update traversability estimates, although not as the primary source of traversability information. The models we construct are not based on range information, however, since this would prevent the extrapolation of the traversability prediction to regions where range is not available.

Howard et al. (2001) presented a learning approach to determining terrain traversability based on fuzzy logic. A human expert was used to train a fuzzy terrain classifier based on terrain roughness and slope measures computed from stereo imagery. The fuzzy logic approach was also adopted by Shirkhodaie et al. (2004), who applied a set of texture measures to windows of an image followed by a fuzzy classifier and region growing to locate traversable parts of the image.

Talukder et al. (2002) describe a system that attempts to classify terrain based on color and texture. Terrain is segmented using labels generated from a 3D obstacle detection algorithm. Each segment is described in terms of Gabor texture measures and color distributions. Based on color and texture, the segments are assigned to pre-existing classes. Each class is associated with an *a priori* traversability measure represented by a spring with known spring constant. We also make use of 3D obstacle detection in our work, but don't explicitly segment the data into regions. We model both background and obstacle classes using color and texture, but all models are created as the vehicle senses the world (although we have made use of known models for some man-made features used in the LAGR program, such as orange plastic fencing material). Given that we have no prior knowledge of the type of terrain that may be encountered, it is usually not possible to pre-specify the classes. Similarly, the vehicle learns the traversability of the terrain by interacting with it, either by driving over it or generating a bumper hit.

The contributions of this paper include a learning algorithm that uses range data to provide traversability information and color and texture from images to provide appearance information. It learns associations between appearance and traversability from small samples and represents them using a histogram-based representation of models that provides a well-defined way of comparing the models and matching them with sensed data. The models are described in terms of color and texture features that do not rely on range data. This enables them to be used to classify regions for which no range data are available. The models are learned from data selected to be close together in 3D space, making it more likely that they are from the same physical region. These modules extend the 4D/RCS architecture by including learning of entities both in the maps kept by the World Model and as symbolically represented objects.

The rest of the paper is organized as follows. First we introduce the problem to be addressed in Sect. 2. Next in Sect. 3, we explain the algorithm and discuss how models are learned and how the classification is carried out. We then describe how the results are represented and present some examples to further explain how the system performs in Sect. 4. We then present an evaluation of the learning algorithm compared to ground truth in Sect. 5, and conclude in Sect. 6 with a discussion.

2 Learning traversability

Many robotic vehicles can navigate successfully in open terrain or on highly constrained roads. Frequently, this capability is due to a careful provision of relevant information about the domain in which the vehicle will operate. The problem we address in this paper, in the context of the DARPA LAGR program, is to determine how to introduce a learning capability to the robot that will enable it to decide for itself the traversability of the terrain around it, based on input from its sensors and its experience of traveling over similar terrain in the past.

Evaluation in the LAGR program takes the form of navigating the vehicle from a defined start point to a fixed goal point. This requires avoiding obstacles such as trees, fences, or various objects introduced into the environment by the LAGR administrators conducting the tests. The vehicle uses its sensors to build a model of the world around it and plans a path from the start to the goal. In many cases, obstacles are placed along the path in such a way as to ensure that a straight-line path to the goal is not traversable. Also, the course may be set up in such a way that by the time the stereo sensors or bumpers detect an obstacle, the vehicle has entered a region that requires a long detour to reach the goal. Teams are given three chances to reach the goal. The idea is that early runs will enable the robot to learn which regions to avoid and which to seek out, so that by the third run it has determined the most efficient path. The vehicle has no a priori knowledge of the kind of terrain it will traverse, so it must learn as it goes along by observing the geometry and appearance of the terrain.

Learning may include remembering the path the vehicle took in previous runs or the regions seen by the sensors during those runs. In our approach, both of these types of learning are included but, as described in this paper, we also try to learn a relationship between the appearance of the terrain and its observed traversability. An advantage of this kind of learning is that regions that are too far away for reliable stereo (and hence reliable obstacle detection) can be identified as either desirable or undesirable for the vehicle to traverse. This enables the vehicle to plan further ahead and avoid entering traps that prevent it from reaching the goal. Remembering the learned models also allows the vehicle to navigate when stereo is not available, as was the case in some of the LAGR evaluations.

3 The algorithm

The autonomous vehicle relies on its sensors to describe the terrain over which it is traveling. Sensor processing must interpret the raw data and extract from it information useful for planning. This includes topographic information, such as slopes and ditches, and feature-based information, such as obstacles and ground cover. While some of the topographic information can be extracted from the range data fairly easily, other features are harder to identify and their properties are not usually obvious from analysis of the sensory data. For example, the traversability of tall grass cannot be determined from range and color information alone, so additional information must be provided through some other means. Often, this is part of the a priori information built in to the system, meaning that the vehicle only has to recognize regions as tall grass to be able to associate a traversability value with them. In this paper, we develop a method that enables the vehicle to learn the traversability of different regions from experience.

We develop an algorithm that first analyzes the range data to locate regions corresponding to ground and to obstacles. Next, this information is used, along with the range and color data, to construct models of the appearance of regions. These models include an estimate of the cost of traversing the regions. Finally, the models are used to segment and classify regions in the color images. Associating regions with models enables traversability costs to be assigned to areas where there is no range data and thus no directly measurable obstacles. As the vehicle traverses the terrain, more direct information is gathered to refine the traversability costs. This includes noting which regions are actually traversed and adjusting the traversability of the associated models. It also involves adjusting the traversability of regions where the vehicle's mechanical bumper is triggered.

3.1 Building the models

First, we construct a local occupancy map (Fig. 2). The map consists of a grid of cells, each of which represents a fixed square region in the world projected onto a nominal ground plane. That is, the grid is fixed to the earth, with each cell centered on a particular latitude and longitude or UTM coordinate. Currently we use an array of 201×201 cells, each of size 0.2 m square, giving a map of size 40 m on a side. The map is always oriented with one axis pointing north and the other east. The map scrolls under the vehicle as the vehicle moves, and cells that scroll off the end of the map are forgotten. Cells that move onto the map are cleared and made ready for new information. Note that if the vehicle moves far enough, the entire map will change. If it then returns to a place it has previously traversed, the information known about that location will be lost. In principle, it would be straightforward to remember all information learned as the vehicle moves about. Alternatively, we could use maps of different resolutions and a strategy for storing abstracted information for later use into a global map as the occupancy grid moves out of a region. Both of these alternatives are used in the World Model component of 4D/RCS, but not in the occupancy map built for our algorithm. If it is available, the cells that the occupancy grid moves into can be filled in using the pre-stored abstract information. Such a strategy was found to be highly effective in learning control in a feedback system (Kwong and Passino 1996). However, for our application the storage of all information or even abstracted versions of it is not generally useful because of errors in the navigation system, which grow as the vehicle moves. This means that when it comes time to restore the contents of a cell it may be hard to decide which stored cell should be used.

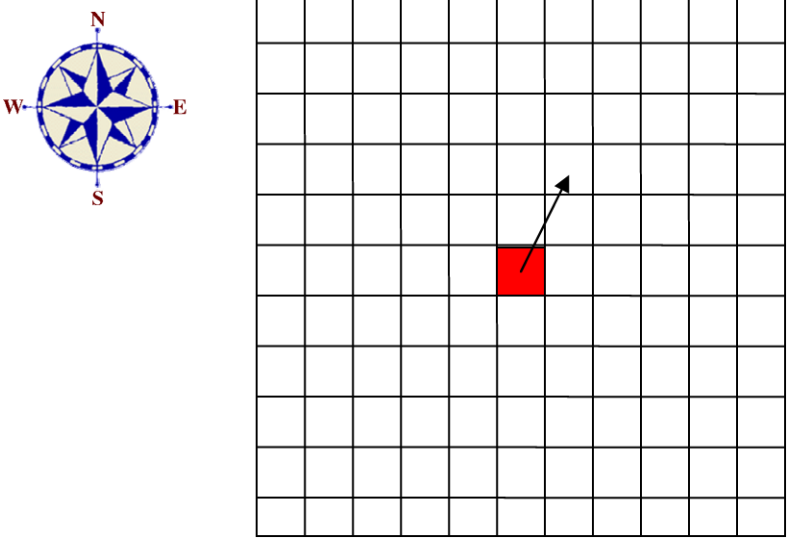

Fig. 2 An occupancy grid with the vehicle in the center

Given the occupancy grid, the algorithm processes the range data and locates obstacles and ground regions. Obstacles are defined as objects that extend more than some distance d above or below the ground (d is determined by the maximum obstacle height that the vehicle can negotiate). Obstacles are detected in the range images (Chang et al. 1999). The algorithm scans column by column in the image, starting with a point known to be on the ground. An initial ground value is assigned at the location where the front wheels of the vehicle touch the ground, known from Inertial Navigational System (INS) and GPS sensors. A pixel is labeled an obstacle if the surface patch to which it corresponds rises high enough and abruptly enough from the ground plane.

The model-building algorithm takes as input the color image, the associated and registered range data (x, y, z points), and the labels (GROUND and OBSTACLE) computed by the obstacle-detection step. It builds models by segmenting the color image into regions with uniform properties. Only points that have associated range values are used. The process works as follows:

When a data set becomes available for processing, the map is scrolled so that the vehicle occupies the center cell of the map. Each point of the data set consists of a vector containing three color values, red (R), green (G), and blue (B). The vector also contains the 3D position of the point (x, y, z), and a label from the obstacle detection step. Currently we consider only OBSTACLE and GROUND labels, although the obstacle detection algorithm identifies other regions, such as overhanging objects. Each point is processed as follows.

1. If the point is not labeled as GROUND or OBSTACLE, it is skipped (other labels can be treated without significant changes to the algorithm). Points that do not have associated range values are also skipped.
2. Points that pass step 1 are projected into the map. This is possible because the x, y, and z values of the point are known as is the pose of the vehicle. If a point projects outside the map it is skipped. Each cell receives all points that fall within the square region in the world determined by the location of the cell, regardless of the height of the point above the ground. The cell to which the point projects accumulates information that summarizes the characteristics of all points seen by this cell. This includes color, texture, and contrast properties of the projected points, as well as the number of OBSTACLE and GROUND points that have projected into the cell and an estimate of the cell's traversability.

Color is represented by ratios R/G, G/B, and intensity (computed as $0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}$) rather than directly using R, G, and B. This provides a small amount of protection from the color of ambient illumination. The color ratios and intensity distribution are represented by 8-bin histograms, representing values from 0 to 255. The values are stored in a normalized form, meaning that the values can be viewed as probabilities of the occurrence of each ratio. Texture and contrast are computed using Local Binary Patterns (LBP) (Ojala et al. 1996). These patterns represent the relationships between pixels in a 3×3 neighborhood in the image, and their values range from 0 to 255. Similarly to the color ratios, the texture measure is represented by a histogram with 8 bins, also normalized. Contrast is computed as a part of computing the LBP measure and is represented by a single number ranging from 0 to 1.

Local Binary Patterns are computed on 3×3 windows (Fig. 3). First, the center pixel value is used to threshold the other pixels in the window (Fig. 3b). Pixels in the neighborhood are set to 1 if they are larger than the center pixel and to 0 otherwise. Then a weighted

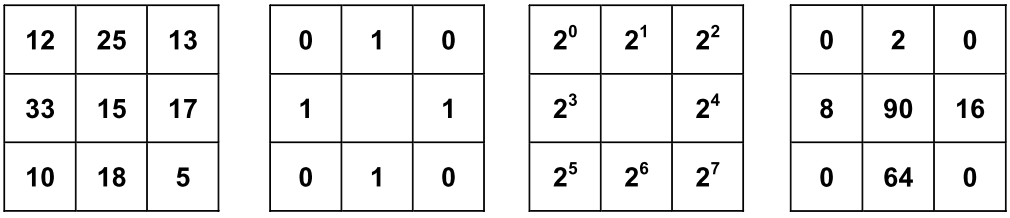

Fig. 3 **a** A 3 × 3 neighborhood. **b** Result of thresholding by middle value. **c** Weights applied to each thresholded pixel. **d** Resulting value in the center cell is the sum of the weighted thresholded values ($0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + \cdots = 90$ or 01011010 binary)

sum is computed of the eight surrounding thresholded points (Fig. 3d). The weights are assigned as powers of 2 (Fig. 3c), so that each location has a unique weight. The weighted sum of the ones and zeroes surrounding the central pixel thus gives rise to a unique number between 0 and 255 representing the texture measure for that location. Given that there are eight surround pixels, and each has value 0 or 1 after thresholding, the final value assigned by the operator to the central pixel can be represented by an eight-bit byte, making the implementation very efficient. The LBP values are combined with a contrast measure at each point, computed over the same window. Contrast is computed as the mean of the values in the window that were above the threshold subtracted from the mean of those below the threshold. For the example of Fig. 3, the contrast is $(25+33+18+17)/4-(12+13+10+5)/4 = 13.25$.

3. When a cell accumulates enough points it is ready to be considered as a model. We determine the sample size by requiring 95% confidence that the sample represents the true distribution and 95% confidence in the proportions in each bin. This determines (from Table 1 in Chakravarty 1999) the sample size of 510 points. Many cells will accumulate more than this number in the very first frame, so learning is fast. In order to build a model, we require that a minimum percentage (currently 95%) of the points projected into a cell have the same label. Given the small region in space the cells represent, this is mostly the case. If a cell is the first to accumulate enough points, its values are simply copied to instantiate the first model. Models have exactly the same structure as cells, so this is trivial. Note that the models include the number of OBSTACLE and GROUND points that project into all the cells that match them. This enables the traversability of the model to be computed as in step 4 below.

   If there are already defined models, the cell must be matched to the existing models to see if it can be merged or if a new model must be created. Matching is done by computing a score, *Dist*, as a weighted sum of the elements of the model m, and the cell c. That is,

$$Dist(c,m) = \sum w_i f_i(c,m),$$

where f_i is either a measure of the similarity of two histograms or, in the case of contrast, is the absolute value of the difference of the two contrast values, $f_{contrast} = |contrast_m - contrast_c|$. The histograms are always stored normalized by the number of points. In the LAGR tests, all w_i were set to 1. Various measures f_h of the similarity of two histograms can be used, such as a Chi Squared test or Kullback-Liebler divergence. After trying these (plus others) we found them too expensive for our real-time application. A sum of squared differences worked as well and is cheaper to compute. Thus, for each model histogram h_m and the corresponding cell histogram h_c,

$$f_h = \sum_{i=1}^{8} (h_{mi} - h_{ci})^2.$$

Cells that are similar enough are merged into existing models; otherwise, new models are constructed. A cell can only merge with a model that has the same type of traversability (i.e., if the cell is considered traversable using the computations in step 4 below, then it can only match with models that are considered traversable). When using a test like Chi Squared, there are standard tables to determine confidence in similarity of distributions. We didn't have those for our measure, so set the threshold experimentally. The same threshold was used for all experiments and all test runs in the LAGR program. If the number of models exceeds a limit, merging of the most similar models is forced, although it might be better to replace the oldest or least-used model with the new one. Merging is a straightforward summation of histograms, each normalized by its number of points. The merged contrast measure is computed as the weighted average of the two contrasts being merged. Figure 4 shows the histograms representing three different models. For efficiency and to prevent overlearning, we set a limit on the number of points that are merged into a model. For 99% confidence, we see from Chakravarty (1999) that about 5000 points are needed. We currently incorporate 10,000 points before we freeze the model.

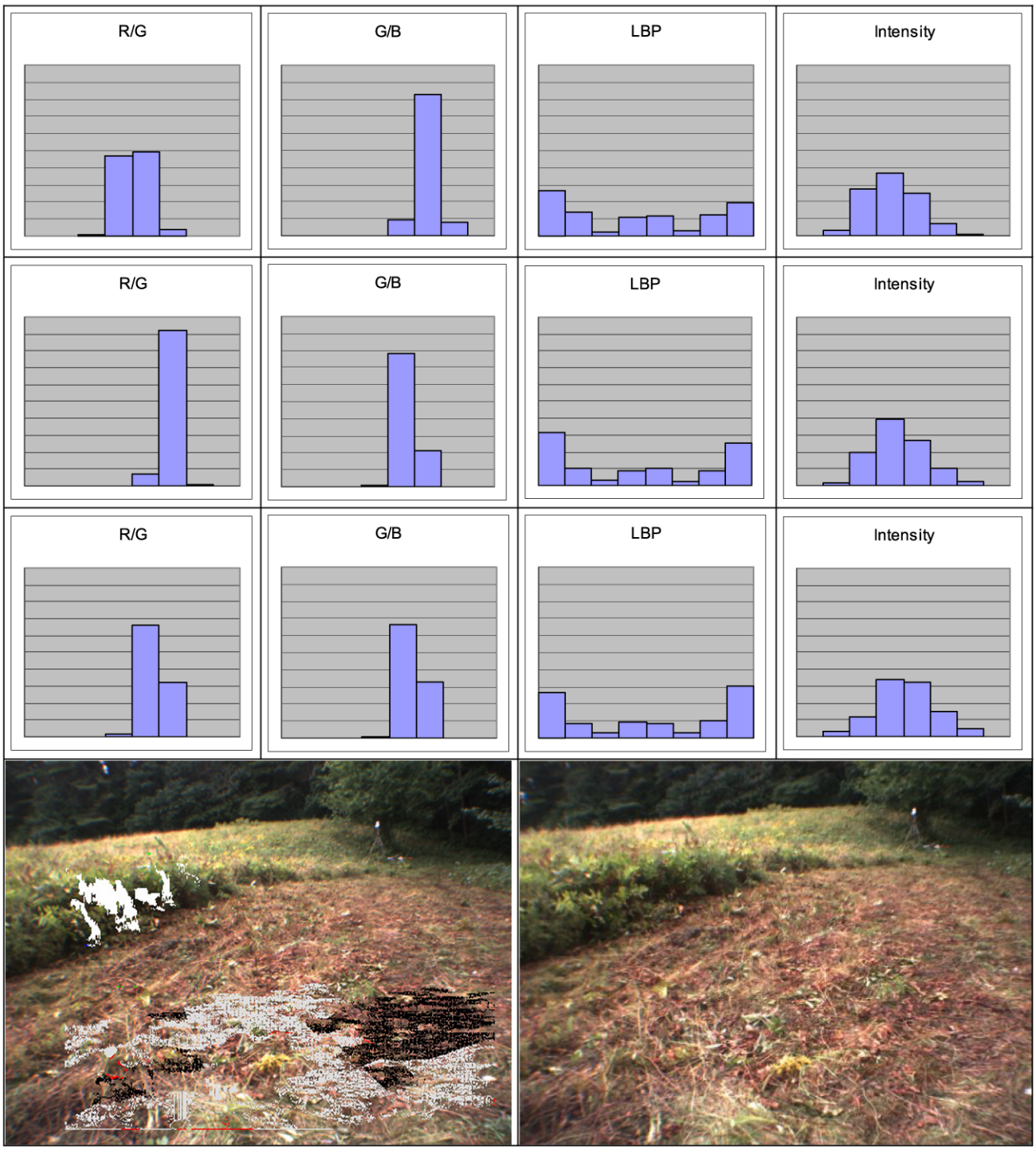

Fig. 4 Examples of histograms used to construct models. *Top row* corresponds to the *white regions* in the *left image*. *Middle row* corresponds to the *black regions*. *Bottom row* corresponds to the *gray regions*. The *black region* is not traversable, while the other two regions are traversable

4. At this stage, there is a set of models whose appearance is distinct. Our interest is not so much in the appearance of the models, but in the traversability of the regions associated with them. Traversability is computed using three types of information. First, when a point is projected into a cell, it brings with it a label, either GROUND or OBSTACLE. Each cell accumulates a count of the number of GROUND and OBSTACLE points that have been projected into it. Second, the vehicle itself occupies a region of space that maps into some neighborhood of cells. These cells and their associated models are given an increased traversability because the vehicle is traversing them. If the bumper on the vehicle is triggered, the cell that corresponds to the bumper location and its model, if any, are given a decreased traversability. Cells and models that don't have known traversability from bumper hits or from being traversed are given traversability values computed as $numOBSTACLE/(numOBSTACLE + numGROUND)$. We plan to further modify the traversability weights by observing when the wheels on the vehicle slip or the engine has to work harder to traverse a cell.
5. When all the points in the input data have been processed, the occupancy map is sent to the World Model (WM) as follows. First, only cells that have values that have changed are sent. If a cell does not have an associated model, its local traversability measure is sent. If it does have a model, the traversability computed from the model is sent. This means that information learned in one region is propagated to other, similar regions through the model matching process. Note that the WM has no knowledge of the local models, and receives only traversability information rather than region identity. The WM stores the traversability information in its own maps which are used to plan a path to the goal.
6. As each new set of data comes in, the process is repeated starting at step 1. Periodically, a sweep is made of all the models. Each model is compared to all the others. If two models are similar enough, they are merged and the number of models is reduced accordingly.

A question arises of what to do when points continue to map into a cell after it has matched with a model. One option is to increment the information in the matching model with the new data. As multiple cells that match the model each update the model's distributions, the individual cells may become poorer matches for the model. This could be the case, for example, because the appearance of a region on the ground can be different when viewed from far away than from close up. An alternative option was adopted instead. This is to link the cell with the model and then zero out the local distributions stored in the cell. Any points that now project into the cell are used to update the cell's local distributions. When the threshold number of points is reached, the cell is matched with all the models again and is either merged with the best match (which may be different from its original model) or cause the creation of a new model. This ensures that the model associated with a cell is always as good a representation as possible.

Another issue concerns the possibility that the traversability of regions matching a model may change as the vehicle moves into new territory. That is, a region with the same appearance as a traversable model in one part of the scene may actually not be traversable in another part of the scene. The approach taken here is to construct "shadow" distributions. That is, when a cell matches with an existing model but has an opposite traversability, a new model is constructed. This model does not participate in model matching until the number of points associated with it exceeds that of the existing model. At this point, the new model replaces the old as the prime model, and the new traversability becomes associated with all matching regions of the image.

3.2 Classifying scenes

So far, we have dealt with model building and computing traversability for cells in the occupancy grid. The information extracted in this way is useful for creating the internal world model and enabling path planning, but it is limited in that it is only applicable to points in the sensed data for which both color and range information are available. Typically, this is a subset of the points in the color images. The next step is to use the models to classify entire scenes. Here only color information is available, with the traversability being inferred from that stored in the models. The assumption is that regions that look similar will have similar traversability.

The approach is to pass a window over the image and compute the color and texture measures at each window location. The matching between the windows and the models operates exactly as it does when a cell is matched to a model in the learning stage. Windows do not have to be large, however. They can be as small as a single pixel and the matching will still determine the closest model, although with low confidence. In the implementation the window size is a parameter, typically set to 16×16. If the best match has an acceptable score, the window is labeled with the matching model. If not, the window is not classified. Windows that match with models inherit the traversability associated with the model. In this way large portions of the image are classified. The choice of window size is a compromise between making it large enough to ensure high confidence that the sample represents the true distribution and small enough that it does not cover too much terrain. Several window sizes were tried with very little difference in results. The 16×16 pixel window size was used in all the test runs.

A problem arises in sending the results to the World Model, which requires a 3D location to be associated with

Fig. 5 The plane to which objects beyond stereo range are projected. The *gray region* is the part of the image that lies on the plane. There are different plane for each of the stereo pairs of cameras

each point. For the part of the image used for model creation, only points that have associated range values are processed, so the problem doesn't arise. For the rest of the image our approach is to make two assumptions. One is that the ground is flat, i.e., that the pose of the vehicle defines a plane through the wheels (see Fig. 5, where the gray region is the part of the image that lies on the assumed ground plane). This allows windows that match with models that are created from ground points to be mapped to 3D locations. The second assumption is that all obstacles are normal to the ground plane and touch the ground. This allows obstacle windows to be projected into the ground plane and thus to acquire 3D locations. We address the issue of where to map a tall obstacle by processing the image from the bottom up (that is, from close to the vehicle to farther away). When an obstacle region first appears, it should thus represent the bottom of the object, and will get mapped to the ground at this point. All contiguous obstacle regions in the same column of the image are mapped to the same ground point, so a tall obstacle will map to the closest ground rather than being spread over the plane. The value of observing these regions from far away is that a low-resolution, long range path planner can decide to avoid obstacles or traverse ground regions well before a finely detailed local plan needs to decide exactly how the vehicle will move (see Fig. 7).

4 Experimental results

The algorithm has been implemented on the DARPA LAGR platform and used to provide part of the information for navigation. We present examples below of how the algorithm works in practice. All data for the examples were collected during tests run by the LAGR Government team. These tests were conducted each month during the first phase of the LAGR program, for a total of 14 tests. A number of different locations were used, with a range of terrain, vegetation, and hazards designed to test various aspects of learning in the systems. The tests required developers to send their control software on flash memory cards to the test facility. The software was then loaded onto a vehicle identical to that used to develop the software. This vehicle was commanded to travel from a start waypoint to a goal waypoint through an obstacle-rich environment. The environment was not seen in advance by the development teams. The Government team measured the performance of the system on multiple runs. To demonstrate learning, performance should improve from run to run as the systems become familiar with the course. Some of the tests also required learning from examples.

In the examples below, the performance of the NIST system is a result of a number of processes that learn about the environment in different ways. These include learning by example and by experience using the algorithm described in this paper. They also include learning pixel-based color models of the terrain and assigning traversability on the assumption that the terrain directly in front of the vehicle is traversable (Tan et al. 2006). Non-image based learning is also used, including learning traversability maps constructed from previous runs, and remembering and optimizing paths taken in past runs (Albus et al. 2006). It is difficult to determine the contributions of each of the learning methods to the final actions, but the examples visually illustrate the traversabilities learned by the algorithm and show them overlaid on images taken during different tests. Across the examples, the values of all algorithm parameters are fixed. The next section provides an evaluation of the learning algorithm in isolation.

4.1 Learning in dry vegetation

The LAGR tests were conducted in a variety of environments, including the Southwest Research Institute's Small Robot Testbed in San Antonio, Texas. Test 9 was held there in January, 2006 in a vegetated area with both woodland and grassland features. The vegetation was dry with only small color variations between different types of vegetation and the ground surface. For this test, color was intended by the Government team not to be a very good feature for building the models. The course began on a lane mown through a field with trees, bushes, brush, and tall grass and traveled along the lane until it reached an open field, where it turned left and continued through the field to the goal. Shortly after leaving the start point, the lane divided into two parallel channels, which merged after approximately 10 m. Further along the course the lane intersected another lane. From the intersection, the straight-line path to the goal traveled through brush and tall grass that was not traversable by the robot, although that was not evident from the intersection point. Further along the course the lane reached

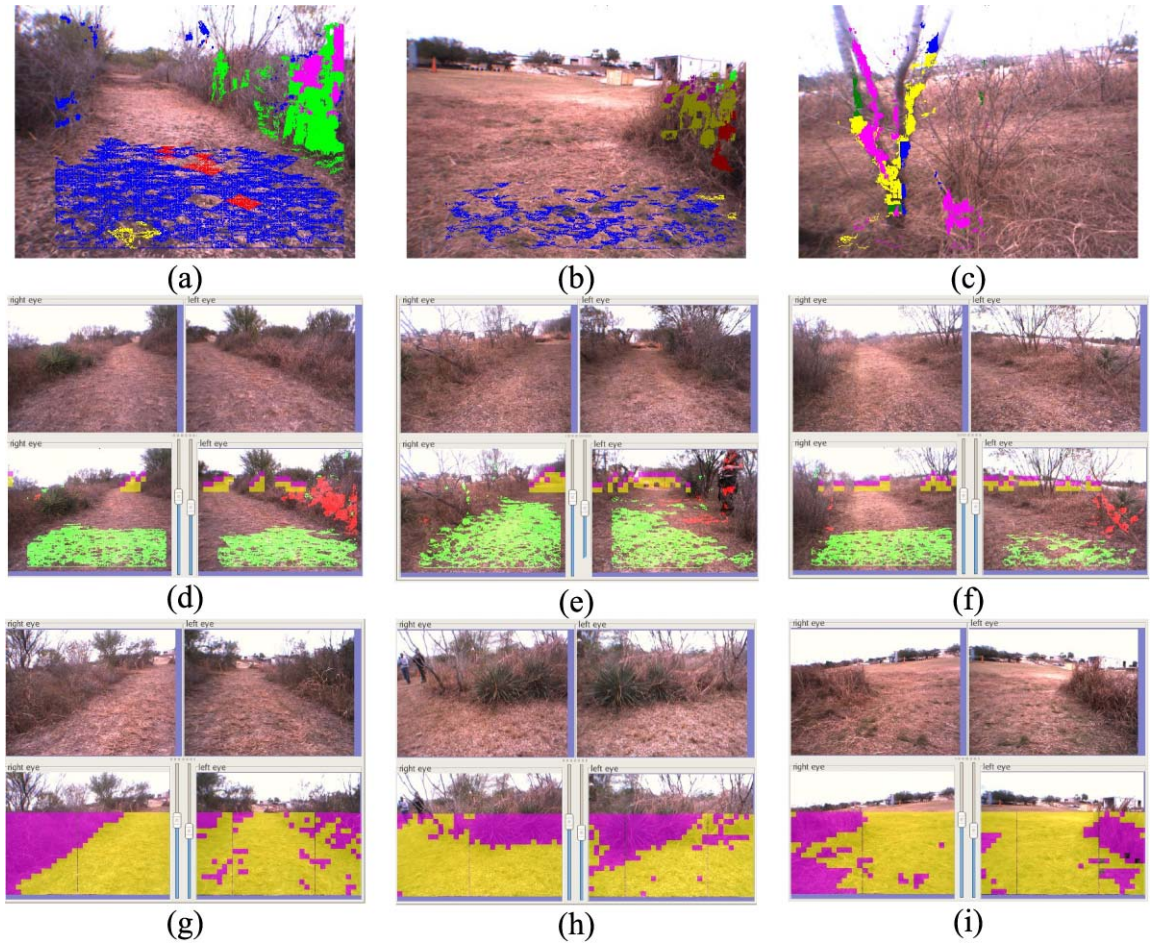

Fig. 6 (Color online) Examples of model learning and classification. **a–c** Models constructed as the vehicle traverses the course. Each color corresponds to a separate model. **d–f** Traversability computed from the models. *Green* is traversable, *red* is not. Classification computed from the models in the range beyond stereo is shown as *yellow* for traversable and *magenta* for not traversable. **g–i** Result of classifying the terrain using the models. *Yellow* corresponds to traversable, *magenta* to obstacles

the open field. A patch of tall grass that was not traversable by the robot grew on the left-hand side of the lane at this point.

The Government team wanted the robot to follow the lane from start to goal. They expected that the robot would be tempted to turn left off the lane at one of the intersections to pursue the straight-line path to the goal. They hoped that in earlier runs, the robot would explore the left-hand turns, and that in later runs, the system would stay on the lane because it had learned that the left-hand turns were unproductive. The NIST system did what the Government team wanted, in that it explored the side path on its first run, learned that it was not productive, and avoided it on the second run. On the third run the system started to explore the non-productive path again, but quickly gave up and returned to the favored path. The time for the first run was 4:05 minutes. For the second and third runs times were 2:21 and 2:32.

Figure 6 shows results of building models and classifying terrain during a test run. The first row (a–c) shows three images taken during the first run. Models are shown overlaid on the images in colors, with each color representing a different model but not necessarily a different traversability. Blue and red models represent traversable regions, while the other colors represent non-traversable regions. The colors are overlaid on the images at locations where points from the stereo data projected into the local map. Each map cell can match a single model, but not all points in the cell are included in the model (only those that have a compatible label). The rectangular shape of the cells explains why some of the regions, such as the red regions in Fig. 6a, appear rectangular. Parts of the image that are not colored with a model can arise if the stereo did not label those points as either obstacle or ground, if there was no model that matched the points, or if the points are beyond the range of stereo. As the vehicle moves, the sensor data sweeps over the terrain and points that are not colored in one image will most likely be labeled in a subsequent image.

The second row of Fig. 6(d–f) shows interpretations of the scene in terms of traversability, regardless of the models that matched each image region. Points shown in green are considered traversable, while those in red are not traversable. The green and red points are marked where points were used to update a model or create a new one. The yellow and magenta points correspond to classifications using a 16×16 window in the region beyond stereo to construct a distribution that matched a model, with yellow representing traversable regions and magenta representing obstacles. The algorithm does not work as well close to the horizon because the size of ground covered by a pixel becomes large and may encompass more than one region. Hence, models may not match, leading to sparse classification. Nonetheless, the algorithm works well enough to provide the directions to open regions and blocked regions, and, as illustrated by the planned path in Fig. 7, this can make the crucial difference between getting drawn into a bad region because of the short range of the vision system and being able to avoid it.

The last row of Fig. 6(g–i) shows the results of using the models to match the terrain from directly in front of the vehicle to the horizon. These results were obtained after the first run created the models. A 16×16 window is passed over the image and each block is matched with a model. If the matching model is for a traversable region, that block of the image is colored yellow. If it is for a non-traversable region, the window is colored magenta. If there is no match, the region is left uncolored. Note that the traversability continues to be modified by driving over or bumping into objects even when stereo is not available. Thus, the models that gave rise to the small obstacle regions in Fig. 6(g–i) will have their traversability increased if the vehicle drives over the corresponding terrain.

4.2 Learning for long range planning

One of the requirements for LAGR was the ability to see beyond the distance at which reliable stereo range data could be acquired. The approach we took was to use the region in which good stereo could be obtained to learn models of the terrain. These models were then matched with regions in the images that corresponded to terrain between the end of reliable stereo and the horizon. As soon as the first model was created, it was applied to classify this region. The example is taken from the seventh test, conducted in October 2005 at Fort Belvoir, Virginia. The course began in an open field. The straight-line path from start to goal led through bushes on the left-hand side of the field. The only traversable path through the bushes was circuitous and difficult, and drastically increased the time taken to complete the course. The path to the right was mainly through open terrain and a dirt road. The Government team placed an artificial obstacle between the beginning of the bushes and the road. The obstacle was low enough to allow the robot to easily see the road, yet tall and dense enough to serve as an obstacle. It divided the course into a region to the left through which it was difficult to reach the goal, and one to the right through which it was easy to reach the goal.

The Government team was hoping that the vehicles would either immediately see and understand the superior traversability of the route to the right of the obstacle, or would take the left side through the bushes on the first run, learn that it was bad, and then take the right hand route on subsequent runs, improving its speed from run to run. The NIST system matched these expectations. In the first run, having no knowledge of the course and no models of the ground and obstacles, it took the left route, negotiated the bushes, and eventually reached the goal after 4 minutes 37 seconds. During this run, the system learned models of the ground, the artificial barrier, and the vegetation. It also constructed and remembered a map of the terrain and the path it actually traveled. This allowed it to improve its performance substantially in the second run, as described below.

Figure 7 shows the sequence of events in the first and second runs of the course. Each subfigure shows a view from the Operator Control Unit (OCU) of the vehicle. The view includes a raw image from each stereo camera (top) and the result of classifying the image using the current set of traversability models (bottom). Note that the left camera image is shown on the right of the OCU and the right image is on the left. This is because the left stereo system points right, and the right one points left.

In Fig. 7a the vehicle is at the start position and has just started creating models. The first model is of the ground in front of the vehicle, shown in green (traversable). As soon as a model has been constructed, the system attempts to classify the region between the end of stereo and the horizon. The left eye was able to classify part of this region, shown in yellow, but the corresponding region in the right eye did not match the initial model well enough to be classified.

Figure 7b shows the situation a little later when the barrier is in the field of view of the left camera pair. A new model is constructed for part of the barrier, shown in red (not traversable). Part of the barrier was classified as ground (green) because the obstacle detection algorithm labeled it as ground instead of as an obstacle. Enough of the barrier was correctly classified to enable the system to behave in the right way. At this stage, the right eye has still only seen the ground class. The left eye has started to label the region beyond stereo with both traversable and non-traversable classes. The right eye will not be able to do so until it sees some of the obstacles in the region where stereo is trusted.

Figure 7c shows a view of the barrier taken after the entire first run. This figure was created by first running the system

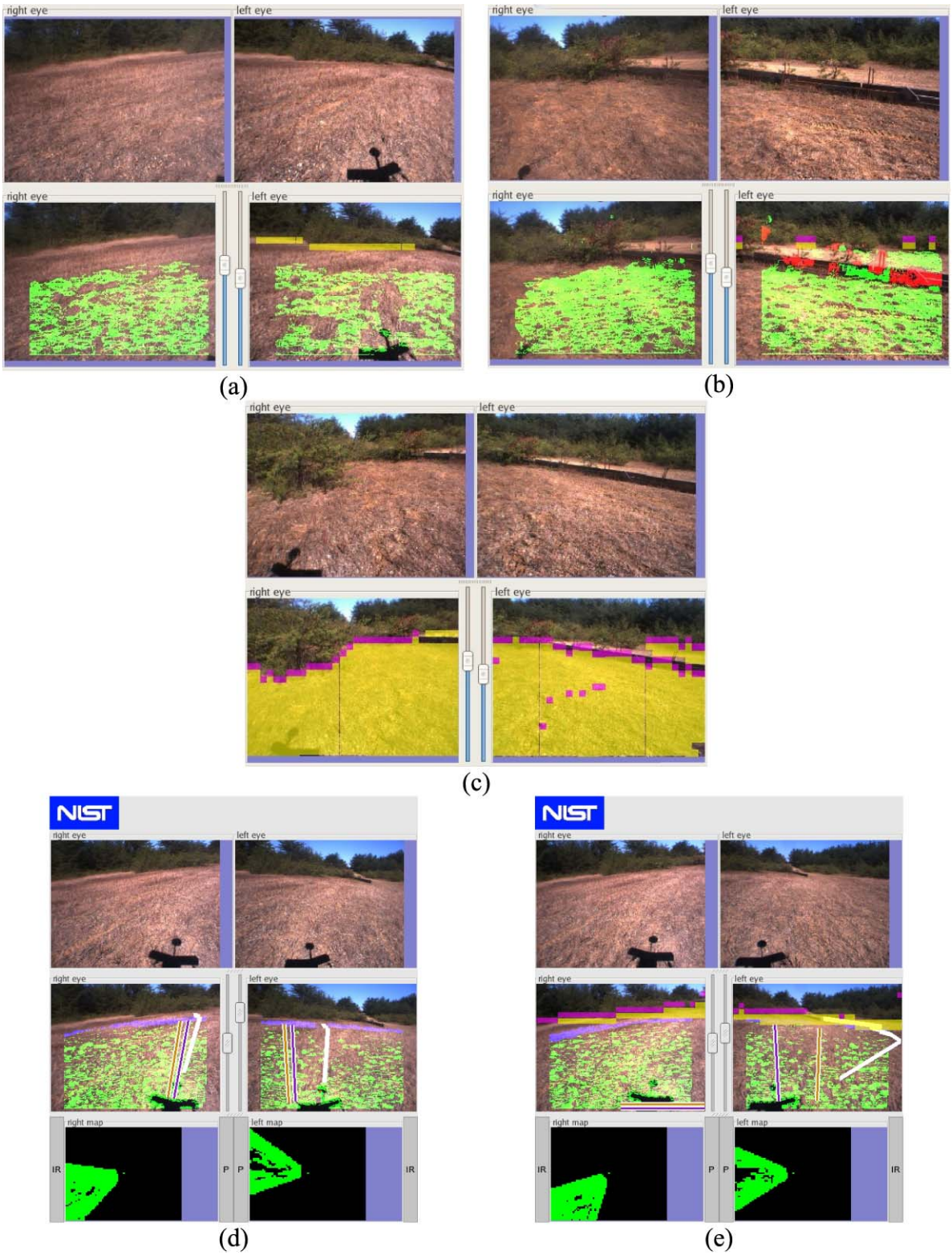

Fig. 7 (Color online) Learning about obstacles to improve path planning. **a** View at the beginning of the first run. Only models of the ground have been created (*green*). **b** View when the barrier is seen. A model for an obstacle is created (*red*). **c** Result of classifying the scene out to the horizon using the learned models (ground is *yellow*, obstacles are *magenta*). **d** *White line* shows planned path at start of first run. **e** *White line* shows planned path at start of second run

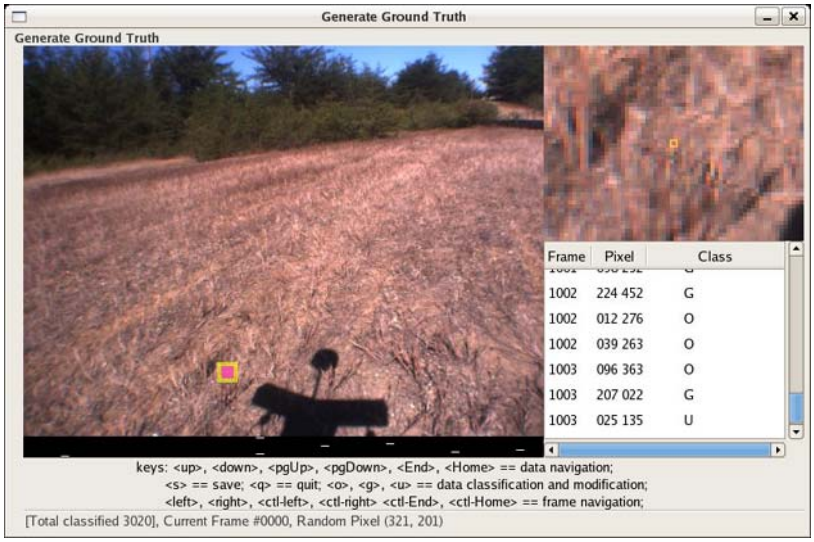

Fig. 8 The GUI for generating ground truth showing a frame from Test 7

on the data set of the entire first run, then turning off stereo processing and running through the first run again. No new models were constructed since there was no stereo to provide the labels. The region from in front of the vehicle to the horizon was classified. Traversable regions are shown in yellow and non-traversable regions in magenta. The obstacle regions have been projected into the ground plane and only the bottoms of obstacles (the parts closest to the vehicle) have been colored magenta.

Figure 7d shows a different view from the OCU. Here there are three rows of images. The top pair shows the OCU view taken from the position where the vehicle started its first run. The middle images show the result of stereo obstacle detection out to 6.5 m in front of the vehicle (green for ground, red for obstacles, and blue for regions that were seen by stereo but were considered too far away to be reliable). The bottom shows the local stereo maps built during obstacle detection. The white line overlaid on the middle pair of images shows the planned path, which points directly to the goal in the absence of any obstacles or prior information.

Figure 7e shows roughly the same location as that in Fig. 7d, but at the start of the second run. In the first run, the vehicle explored the terrain and learned models of the ground and the obstacles. It applies these models to the region of the image beyond the stereo reliability range. In the OCU, the results show up as the yellow (ground) and magenta (obstacle) bars in the middle images. As can be seen, this information starts where the green region corresponding to the stereo processing ends. The white line in Fig. 7e shows the planned path for the second run. The learned information enabled the planner to avoid entering the non-productive region to the left of the barrier and make it directly to the goal in 1 minute and 32 seconds.

5 Performance evaluation

The above results give an overview of how the whole LAGR system works, but do not give a clear indication of how the learning algorithm contributes to performance. To do this, a method was developed of evaluating the algorithm relative to human performance (Shneier et al. 2006). The method is applicable to any algorithm that labels regions of an image with class labels. Evaluating the algorithm requires determining how well the learned models classify the degree of traversability of the terrain around the vehicle. The evaluation uses ground truth generated by one or more human observers. Data sets used for the evaluation consisted of log files generated during the tests. Log files contain the sequence of images collected by the two pairs of stereo cameras on the LAGR vehicle and information from the other sensors, including the navigation (GPS and INS) sensors and bumper sensors (physical and IR bumpers). The NIST system performs exactly the same when playing back a log file as it did when it first ran the course. Therefore, logged data is a good source for performance testing.

The ground truth is collected by a human stepping sequentially through the log file, and classifying one or more points from each image. A graphical tool is used to display the image and randomly select a point (Fig. 8). The point is highlighted for the user, who selects one of the labels Ground (G), Obstacle (O), or Unknown (U). The tool then writes a record to a file containing the frame number, coordinates of the selected point, and the label provided by the user. When ground truth collection is complete, the file is available for evaluating the performance of the learning algorithm or any other algorithm that assigns traversability labels to regions.

Table 1 Results for Test 6

Test 6, 2513 ground truth points			
No. correct	No. incorrect	% correct	% incorrect
2197	317	87.4%	12.6%
Error distribution across label types			
Not classified (Unknown)	Obstacle instead of Ground	Ground instead of Obstacle	
30%	52%	17%	

Table 2 Results for Test 7, User 1

Test 7, 702 ground truth points			
No. correct	No. incorrect	% correct	% incorrect
592	110	84.5%	15.5%
Error distribution across label types			
Not classified (Unknown)	Obstacle instead of Ground	Ground instead of Obstacle	
47%	34%	19%	

Table 3 Results for Test 7, User 2

Test 7, 2195 ground truth points			
No. correct	No. incorrect	% correct	% incorrect
1884	312	85.8%	14.2%
Error distribution across label types			
Not classified (Unknown)	Obstacle instead of Ground	Ground instead of Obstacle	
71%	4%	25%	

To generate data for performance analysis, the learning algorithm reads the ground truth file and the log file. It processes the log file as it usually does when running on the vehicle. Each time it comes to an image frame for which ground truth is available, it classifies the points selected in the frame and writes out a file containing the ground truth it read in plus an entry giving the learned classification of the pixel in the ground truth file. When the entire log file has been processed, the output file contains an entry for each ground truth point that gives both the human's classification and the system's classification. Under the assumption that the human's classification is correct, an analysis can be conducted of the errors committed by the learning algorithm.

The evaluation was applied to a number of examples taken from data gathered by the LAGR evaluation team at locations in Virginia and Texas. In the evaluations, the learning system starts out with no models. This is how the system typically starts, at least for the first test run at each location. As it reads the log file and the ground truth data, the learning program both creates the models and classifies the ground truth points. This means that early in the sequence of images, only a small number of models are available for classification. As more of the terrain is seen, more models are constructed, and the range of regions that can be classified increases. The algorithm learns very fast, however, often creating the first few models from the first frame or two of data. Since the terrain doesn't usually change abruptly, classification performs well from the start, particularly for points close to the vehicle.

Four sets of ground truth data were created by three different people using the GUI in Fig. 8. The data were taken from log files of three different tests: Test 6, Test 7, and Test 9. The ground truth created for Test 6 consisted of 3 points per frame, using the log file of the first test run. Because the human sometimes labeled a point as Unknown, and because some of the points randomly selected for ground truth were in the sky, the actual number of usable points was closer to 2 per frame (there were 1,270 frames). Table 1 shows a summary of the results of the evaluation. As can be seen, the algorithm performed well, labeling 87% of the points the same as the human. Of the incorrect labels, 30% arose from situations where the algorithm did not find a match with any model and labeled the points Unknown, 52% came from incorrectly labeling points as Obstacle instead of Ground, and 17% from labeling points as Ground instead of Obstacle.

The ground truth for Test 7 was created from the log file of the first test run. Two different people generated ground truth files. One selected 1 point per frame, resulting in a usable count of 702 points, while the other selected 3 points per frame, resulting in a usable count of 2195 points, where usable points are determined as described above for Test 6. Having different selections of points for the same data set enabled us to see if there was significant variation between people's selection of labels and also let us see if a smaller number of points was as effective as a larger one. As can be seen in Tables 2 and 3, the results for both the small sample size and the large one are very similar, indicating that it is not necessary to label large numbers of points. What was surprising was that the distribution of the errors was different. For the smaller set, the percentage of errors due to the learning algorithm not being able to identify the class of the point was 46%, whereas the corresponding percentage for the larger set was 71%. In the tests we have done, the distributions of errors with different random sets of points have not shown any obvious pattern.

The ground truth for Test 9 was created from the log file of the first run, using a single point from each frame and a total of only 176 frames. There were a total of 290 points

to be classified. As can be seen in Table 4, the system performed a little worse in this low-color environment, but still respectably.

The results of all the performance evaluations are accumulated in Table 5. As can be seen, 86% of the time the algorithm assigns the same labels to regions as those assigned by human observers.

Since the learning algorithm depends on the labels assigned by the stereo obstacle detection algorithm, a similar performance evaluation was done on that algorithm. For each set of data, the log file was read together with the ground truth. For each ground truth point in a frame, the classification assigned by the program was compared to that assigned by the human. Since stereo has limited range, only about 30% of ground truth points could be classified by the obstacle detection algorithm. The percent correct and incorrect are computed for the subset of points that had labels from both the human and the program. The obstacle detection performed as shown in Table 6, agreeing with a human 91% of the time. Accounting for this upper limit on the effectiveness of the learning algorithm, it can be seen that learning performs 95% optimally.

Another way of using the ground truth data is to investigate the effects of the model parameters. We use five parameters to describe the models, and here we discuss the effects of selecting subsets of these parameters. We explored using only color (no intensity or texture), using color plus intensity with no texture, and not using color. There are two color components, R/G and G/B. We did not explore removing only one of them. Nor did we look at the effects of contrast. Some of the results were surprising.

Table 7 shows the classification success of the algorithm when it learns models with one or more features removed. It appears that removing texture has hardly any effect. The percentage of correct classifications for Test 7 goes down marginally (just over 2%), but the correct classification for Test 9 goes up (about 2%)! This is very surprising. Since the data for Test 9 showed little color variation, we assumed that the texture was providing most of the discrimination. It probably means that the texture measure we used is not suitable for this application (perhaps because it uses such a small neighborhood). On the other hand, taking color out of the model features has a big impact, dropping the classification accuracy in Test 7 from about 86% to 53%. For Test 9 the accuracy also drops, but only from 80% to 76%. This is reasonable, since the data showed so little color variation.

Finally, if only color is used, the performance on Test 9 degrades considerably, from 80% to 56%. The performance on Test 7 actually goes up marginally, although probably not significantly. We can conclude that intensity plays a significant role in classification, especially in Test 9. Color is clearly important, but the use of the Local Binary Pattern operator is questionable. We plan to explore alternative tex-

Table 4 Results for Test 9

Test 9, 290 ground truth points			
No. correct	No. incorrect	% correct	% incorrect
232	58	80.3%	20.1%
Error distribution across label types			
Not classified (Unknown)	Obstacle instead of Ground		Ground instead of Obstacle
19%	21%		60%

Table 5 Cumulative results

Tests 6, 7, and 9, 5701 ground truth points	
Number of points classified	5701
Number correct	4905
Number incorrect	797
Percentage correct	86%
Percentage incorrect	14%

Table 6 Cumulative results of stereo-based obstacle detection

Cumulative, Tests 6, 7, and 9	
Number of points classified	1663
Number correct	1512
Number incorrect	151
Percentage correct	91%
Percentage incorrect	9%

Table 7 Effects on classification of changing model parameters

No texture		No color		Only color	
% correct	% incorrect	% correct	% incorrect	% correct	% incorrect
Test 7 model parameter variation					
83.52%	16.48%	53.26%	46.79%	86.25%	13.75%
Test 9 model parameter variation					
82.35%	17.99%	76.12%	24.22%	56.40%	43.94%

ture measures based on multiresolution Gabor filters as in Talukder et al. (2002) to see if they perform better.

Overall, the results show that the algorithm for learning traversability works well, with a high degree of agreement (86%) between its classifications and those of a human observer. This provides confidence that the algorithm will enhance the performance of the LAGR control system as a whole.

6 Discussion and conclusions

We have presented a system that learns to predict the traversability of regions based on the assumption that regions that look similar will have similar traversability. The models that are constructed to represent regions are robust, in the sense that they apply across a wide set of ranges. Objects may look substantially different from close up than they do from far away, and the color and texture measures are computed using only data from the closest parts of the image. Nonetheless, the models continue to match well when applied to the middle range of the image. This is probably due to the coarse binning in the models which mimics the smoothing effect of increasing range. Because we apply the texture measure to regions beyond the range of stereo, it would perhaps be better to use a multiscale texture measure (e.g., based on Gabor filters (Talukder et al. 2002)). Alternatively we could adopt the approach described by Hadsell et al. (2006), in which a transform is applied to the image to scale the rows by distance.

The color model used to represent the appearance of the different terrain models is a descendent of the histogram intersection approach developed by Swain and Ballard (1991). Instead of three-dimensional histograms, we use two one-dimensional histograms, and instead of their histogram intersection algorithm for comparing histograms, we use a sum of squared difference measure (which is very similar to the sum of absolute differences used in histogram intersection). The size of the histograms we use is substantially smaller also, but, as expected from Swain and Ballard's analysis, this has little impact on the accuracy of the matching. Pietikainen et al. (1996) showed that three one-dimensional histograms perform almost as well as one three-dimensional histogram, although they did not use color ratio histograms in their experiments.

The number of models constructed depends on the complexity of the environment and on the similarity measure used to compare models. Even with a fairly strict similarity measure we have not found that the number of models becomes large. A typical scenario, such as driving through the woods, will generate less than twenty models. It is also not very important to know exactly what each model represents. The main concern is that the models correctly predict traversability, not that they encode a semantically meaningful object. Nonetheless, regions are typically represented by a single model or perhaps two if there is a significant lighting change.

Using the range information to build the models ensures that points that are spatially close in the three-dimensional world provide the data for the models. This makes them more likely to belong to the same physical region, especially since they are required to have the same label from the obstacle detection algorithm. The processed range information, which provides labels for obstacle and ground based on the geometry of the region, provides strong evidence of the traversability of the region. This serves as a good starting point for traversability, which is modified by the behavior of the vehicle in driving over some regions and registering bumper hits in others. In some cases, however, range information is not available. In this case, we can still build models under the initial assumption that the region in front of the vehicle is traversable and that regions that do not match are not traversable (Ulrich and Nourbakhsh 2000a; Tan et al. 2006). Together with observing what actually happens as the vehicle moves, this should enable the algorithm to work in a similar manner to how it does when range is available.

Incorporating learning into the operating system of a robotic vehicle requires many compromises. The vehicle places limits on processing time because it must have the information it needs fast enough for control to remain stable and for the vehicle to avoid obstacles. A balance must be maintained between learning and other processing in each module so that each can operate effectively. The overall cycle time of the processing of each component must be fast enough that other modules that use its information are not forced to wait for it. These issues are addressed in our system partly by the use of the 4D/RCS architecture and partly by careful implementation of the algorithms.

We have presented in this paper an algorithm that learns to associate traversability of a region with a description of its color and texture. It represents the associations using a histogram-based representation of models that enables easy comparisons between models and sensed data. The features used to describe the models do not rely on range data. This lets them be used to classify regions for which no range data are available. The models are learned from data selected to be close together in space, making it more likely that they are from the same physical region. We have shown through a number of examples taken from the DARPA LAGR test data that the algorithm performs satisfactorily by classifying the traversability of regions either across an entire image or only in the region beyond the range of stereo. The algorithm performs as part of a larger control system that includes other types of learning, and is able to perform its actions fast enough to ensure that the entire control system remains stable.

Acknowledgements The work described in this paper was conducted under grants from the DARPA LAGR program and the Army Research Laboratory. We are grateful for their support. We are also indebted to Kevin Passino, whose careful reading of the manuscript and knowledge of the machine learning literature greatly improved the paper.

References

Albus, J. S., & Meystel, A. (2001). *Engineering of mind: an introduction to the science of intelligent systems*. Somerset: Wiley.

Albus, J. S., Huang, H.-M., Messina, E., Murphy, K., Juberts, M., Lacaze, A., Balakirsky, S., Shneier, M. O., Hong, T., Scott, H., Horst, J., Proctor, F., Shackleford, W., Szabo, S., & Finkelstein, R. (2002). *4D/RCS Version 2.0: A reference model architecture for unmanned vehicle systems* (NISTIR 6912). Gaithersburg, MD: National Institute of Standards and Technology.

Albus, J., Bostelman, R., Chang, T., Hong, T., Shackleford, W., & Shneier, M. (2006). Learning in a hierarchical control system: 4D/RCS in the DARPA LAGR program. *Journal of Field Robotics*, *23*(11/12), 975–1003.

Chakravarty, S. (1999). Sample size determination for multinomial population. In *National association for welfare research and statistics 39th annual workshop*, Cleveland, Ohio. http://www.nawrs.org/ClevelandPDF/papers/Page_2x.html.

Chang, T., Hong, T., Legowik, S., & Abrams, M. (1999). Concealment and obstacle detection for autonomous driving. In *Proceedings of the robotics & applications conference* (pp. 147–152). Santa Barbara, CA.

DeSouza, G. N., & Kak, A. C. (2002). Vision for mobile robot navigation: a survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *24*(2), 237–267.

Hadsell, R., Sermanet, P., Ben, J., Han, J., Chopra, S., Ranzato, M., Sulsky, Y., Flepp, B., Muller, U., & LeCun, Y. (2006). On-line learning of long-range obstacle detection for off-road robots. In *The learning workshop*, Snowbird, UT.

Howard, A., Tunstel, E., Edwards, D., & Carlson, A. (2001). Enhancing fuzzy robot navigation systems by mimicking human visual perception of natural terrain traversability. In *Joint 9th IFSA world congress and 20th NAFIPS international conference* (pp. 7–12).

Jackel, L. D., Krotkov, E., Perschbacher, M., Pippine, J., & Sullivan, C. (2006). The DARPA LAGR program: goals, challenges, methodology and phase I results. *Journal of Field Robotics*, *23*(11/12), 945–973.

Kwong, W. A., & Passino, K. M. (1996). Dynamically focused fuzzy learning control, Part B. *IEEE Transactions on Systems, Man and Cybernetics*, *26*(1), 53–74.

Ojala, T., Pietikainen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, *29*, 51–59.

Pietikainen, M., Nieminen, S., Marszalec, E., & Ojala, T. (1996). Accurate color discrimination with classification based on feature distributions. In *13th international conference on pattern recognition (ICPR'96)* (Vol. 3, pp. 833–838).

Puzicha, J., Hofmann, T., & Buhmann, J.M. (1997). Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. In *IEEE computer society conference on computer vision and pattern recognition (CVPR'97)* (pp. 267–272). San Juan, Puerto Rico.

Shirkhodaie, A., Amrani, R., Chawla, N., & Vicks, T. (2004). Traversable terrain modeling and performance measurement of mobile robots. In *Performance metrics for intelligent systems, PerMIS'04*, Gaithersburg, MD.

Shneier, M., Shackleford, W., Hong, T., & Chang, T. (2006). Performance evaluation of a terrain traversability learning algorithm in the DARPA LAGR program. In *Performance metrics for intelligent systems, PerMIS 2006*.

Swain, M. J., & Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, *7*(1), 11–32.

Talukder, A., Manduchi, R., Castano, R., Matthies, L., Castano, A., & Hogg, R. (2002). Autonomous terrain characterisation and modelling for dynamic control of unmanned vehicles. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 708–713).

Tan, C., Hong, T., Shneier, M., & Chang, T. (2006). Color model-based real-time learning for road following. In *IEEE intelligent transportation systems conference (ITSC'06)* (pp. 939–944). Toronto, Canada.

Ulrich, I., & Nourbakhsh, I. (2000a). Appearance-based obstacle detection with monocular color vision. In *Proceedings of the AAAI national conference on artificial intelligence*. Austin, TX.

Ulrich, I., & Nourbakhsh, I. (2000b). Appearance-based place recognition for topological localization. In *IEEE international conference on robotics and automation* (pp. 1023–1029). San Francisco, CA.

Wellington, C., & Stentz, A. (2003). Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetation. In *International conference on field and service robotics* (pp. 49–54).

Michael Shneier is group leader for perception systems at NIST. He received his Ph.D. in Artificial Intelligence at the University of Edinburgh, Scotland. He manages and conducts research in sensing and world modeling in the areas of manufacturing and autonomous vehicle navigation. Efforts are also directed towards sensor evaluation, calibration, and fusion, and reference data collection. Current research includes road and road sign detection, terrain classification, learning, 6DOF dynamic metrology for manufacturing, and sensing for robot safety.

Tommy Chang received his Bachelors degrees in both Electrical Engineering and Computer Sciences from the University of Maryland, College Park. He is currently pursuing a Masters degree in Computer Sciences from Johns Hopkins University. He is also a researcher in the Intelligent Systems Division at the National Institute of Standards and Technology. His research interests are computer vision, image processing and robotics.

Tsai Hong received her Ph.D. from the University of Maryland in 1982. Dr. Hong conducts research in perception for vehicle navigation and for manufacturing, and develops performance evaluation methods for perception systems. Dr. Hong develops and supervises research projects in realtime vision, world modeling, multi-sensor fusion, and temporal fusion in dynamic environments. She has served as a doctoral thesis advisor and committee member for various students and has published over 100 articles on the above research areas.

Will Shackleford graduated in 1993 with a B.S. in Electrical Engineering from the University of Maryland. He works in the Intelligent Systems Division of the Manufacturing Engineering Lab at the National Institute of Standards and Technology. Projects include the Real-Time Control Systems (RCS) Library, Neutral Messaging Language (NML), Enhanced Machine Controller (EMC), Aerial Multi-axis Platform (AMP) used for Air Force airplane depainting, and various autonomous ground vehicle applications.

Roger Bostelman is Program Manager for the Intelligent Control of Mobility Systems program at NIST. He has been with NIST for 29 years. Roger has designed, built, and tested mechanical systems and their interface electronics on robot arms and vehicles. He holds a B.S. in Electrical Engineering from the George Washington University and an M.S. in Technical Management from the University of Maryland University College. He has over 45 publications in journals and conference proceedings and he holds 5 patents on RoboCrane systems with one pending.

James S. Albus is a Senior NIST Fellow working on theoretical and experimental studies of intelligent behavior in artificial and biological systems. He received his Ph.D. in Electrical Engineering from the University of Maryland. He serves on the editorial board of six journals and one scientific book series in the field of intelligent systems He has published more than 200 papers in the field of intelligent systems, and has authored, co-authored, or edited six books.