Integrating Learning into a Hierarchical Vehicle Control System

James Albus, Roger Bostelman*, Tsai Hong, Tommy Chang, Will Shackleford, Michael Shneier
National Institute of Standards and Technology
100 Bureau Drive, Gaithersburg, MD 20899, USA
james.albus, roger.bostelman, tsai.hong, tommy.chang, will.shackleford, or michael.shneier@nist.gov

Abstract

The National Institute of Standards and Technology's (NIST) Intelligent Systems Division (ISD) is a participant in the Defense Advanced Research Projects Agency (DARPA) LAGR (Learning Applied to Ground Robots) Program. The NIST team's objective for the LAGR Program is to embed learning algorithms into the modules that make up the 4D/RCS (Four Dimensional/Real-Time Control System), the standard reference model architecture that ISD has applied to many intelligent systems. This enables the vehicle to learn to navigate in complex, off-road terrain. The vehicle learns in several ways. These include learning by example, learning by experience, and learning how to optimize traversal. Learning takes place in the sensory processing, world modeling, and behavior generation parts of the control system. This paper describes the 4D/RCS structure, its application to the LAGR program, and the learning and mobility control methods used by the NIST team's vehicle. Results are

* Corresponding Author:

Roger Bostelman
Intelligent Systems Division
National Institute of Standards and Technology
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899
Tel: (301) 975-3426
Fax: (301) 990-9688
Email: Roger.Bostelman@nist.gov

shown from the series of tests conducted by an independent evaluation team, and the performance of one of the learning algorithms is evaluated.

1. Introduction

The National Institute of Standards and Technology's (NIST) Intelligent Systems Division (ISD) has been developing the RCS [1,3] reference model architecture for over 30 years. 4D/RCS is the most recent version of RCS developed for the Army Research Lab Experimental Unmanned Ground Vehicle program. ISD is currently applying 4D/RCS to the DARPA LAGR program [5].

The DARPA LAGR program aims to develop algorithms that enable a robotic vehicle to travel through complex terrain without having to rely on hand-tuned algorithms that only apply in limited environments. The goal is to enable the 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 (90 kg) robotic vehicles to each of the participants (Fig. 1). The vehicles are used by the teams to develop software. A separate DARPA team, with an identical vehicle, conducts tests of the software each month. Operators load the software onto their vehicle and command it to travel from a start waypoint to a goal waypoint through an obstacle-rich environment. They measure the performance of the system on multiple runs with the expectation that improvements will be made through learning.

Learning has been applied to computer vision for a variety of applications, including traversability prediction. Wellington and Stentz [14] 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) map with multiple attributes at each map grid point 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 do not include range or 3D shape information, however, since this would prevent the extrapolation of the traversability predictions to regions where range is not available.

Howard et al. [4] 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. [10], 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 and his colleagues [12] 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. 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.

2. 4D/RCS REFERENCE MODEL ARCHITECTURE

The 4D/RCS architecture is characterized by a generic control node at all the hierarchical control levels. Each node within the hierarchy functions as a goal-driven, model-based, closed-loop controller. Each node is capable of accepting and decomposing task commands with goals into actions that accomplish task goals despite unexpected conditions and dynamic perturbations in the world. At the heart of the control loop through each node is the world model, which provides the node with an internal model of the external world (Fig. 2). The world model provides a site for data fusion, acts as a buffer between perception and behavior, and supports both sensory processing and behavior generation.

The nature of the world model distinguishes 4D/RCS from conventional artificial intelligence (AI) architectures. Most AI world models are purely symbolic. In 4D/RCS, the world model is a combination of instantaneous signal values from sensors, state variables, images, and maps that are linked to symbolic representations of entities, events, objects, classes, situations, and relationships in a composite of immediate experience, short-term memory, and long-term memory. Real-time performance is achieved by restricting the range and resolution of maps and data structures to what is required by the behavior generation module at each level. Short range, high resolution maps are implemented in the lower levels, with longer range, lower resolution maps at the higher levels

A diagram of the internal structure of the world model and value judgment system is shown in Fig. 2. Within the knowledge database, iconic information (images and maps) is linked together and to symbolic information (entities and events). Situations and relation-

ships between entities, events, images, and maps are represented by pointers. Pointers linking symbolic data structures to regions in images and maps provide symbol grounding and let the world model project its understanding of reality onto the physical world.

3. 4D/RCS Applied to LAGR

NIST was challenged by DARPA to show that the 4D/RCS architecture was able to include learning. Thus, the control system delivered with the LAGR vehicle was replaced with one based on the 4D/RCS architecture. The 4D/RCS architecture for LAGR (Fig. 3) consists of two levels, enabling plans to be generated at the low and high levels out to approximately 20 m and 60 m, respectively, in front of the vehicle. This is all that is needed because the LAGR test courses are small (about 100 m), and the test missions are short (typically less than 10 minutes.) For controlling a battalion of autonomous vehicles, five or more 4D/RCS hierarchical levels would be needed. The following sub-sections describe the algorithms implemented in sensor processing, world modeling, and behavior generation, and the learning algorithms. More detail can be found in [2].

3.1 Sensory Processing

The sensor processing column in the 4D/RCS hierarchy for LAGR starts with the sensors on board the LAGR vehicle. Sensors include two pairs of stereo color cameras, a physical bumper and infra-red bumper sensors, the motor current sensor (for terrain resistance), and the navigation sensors (GPS, wheel encoder, and IMU). Sensory processing modules include a stereo obstacle detection module, a bumper obstacle detection module, an infrared obstacle detection module, an image classification module, and a terrain traction detection module. Sensory processing is responsible for interpreting the sensed data at each level and reporting what it senses to the world model.

The vehicle has two ways of detecting very close obstacles. The first is a physical bumper that is activated by hitting an object hard enough to trigger a switch. The second is a pair of infra-red sensors that return a signal if an object is within their range. The infra-red sensors were found to be very noisy and unreliable and were seldom used. The physical bumper was reliable and locations that activated it were given the highest possible traversability cost.

3.2 Stereo Vision

Stereo vision is primarily used for detecting obstacles. We use the SRI Stereo Vision Engine [6] to process pairs of images from the two stereo camera pairs. For each newly acquired stereo image pair, the obstacle detection algorithm processes each vertical scan line in the reference image independently and classifies each pixel as ground, obstacle, short_obstacle, cover or invalid. Pixels that are not in the 3D point cloud are marked invalid. Pixels corresponding to obstacles that are shorter than 5 cm high are marked as short_obstacle. The obstacle height threshold was chosen because the LAGR vehicle can ignore and drive over smaller pebbles and rocks. cover corresponds to obstacles that are taller than 1.5 m, a safe clearance height for the LAGR vehicle.

Within each reference image, the corresponding 3D points are accumulated onto a 2D cost map of 20 cm by 20 cm cell resolution. Each cell has a cost value representing the percentage of obstacle pixels in the cell. In addition to cost value, color and elevation statistics are also kept and updated in each cell. This map is sent the world model at the current level and to the sensory processing module at the level above in the 4D/RCS hierarchy. Fig. 6 shows a view of obstacle detection from the operator control unit (OCU). The top row of the Figure shows an original image from each of the stereo pairs. The middle row shows the re-

sults of obstacle detection. The white regions are classified as obstacles and black represents ground. The mid-grey regions represent points for which the stereo algorithm was able to compute a range but the confidence was considered too low for reliable classification. The bottom row of Fig. 6 shows the 2D cost map for each of the stereo pairs.

3.3 World Modelling

The world model is the system's internal representation of the external world. It acts as a bridge between sensory processing and behavior generation in the 4D/RCS hierarchy by providing a central repository for storing sensory data in a unified representation. It decouples the real-time sensory updates from the rest of the system. For the LAGR program, two world model levels have been built (WM1 and WM2). Each world model process builds a two dimensional map, but at different resolutions. These are used to temporally fuse information from sensory processing. Currently the lower level of sensory processing (SP1) is fused into both WM1 and WM2 because the learning module in SP2 does not yet send its models to the WM. Fig. 4 shows WM1 and WM2 maps constructed from the stereo obstacle detection module in SP1. The maps contain traversal costs for each cell in the map. The position of the vehicle is shown as an overlay on the map. The colors represent cost values, and black represents unknown areas. Each map cell represents an area on the ground of a fixed size and is marked with the time it was last updated. The length and width of the map is 40 m for WM1 and 120 m for WM2. The information stored in each cell includes the average ground and obstacle elevations, the variance, minimum and maximum elevation, and a confidence measure reflecting the quality of the data. The cell also contains its traversability cost and confidence, which is updated by the stereo obstacle detection module, image classification module, bumper module, and infrared sensor module. The map updating algorithm

is based on confidence-based mapping as described in [8].

In the absence of a bumper hit, the costs are combined to determine the relative safety of traversing the grid using the following equation:

$$Cost_{cell} = W_s \times Cost_{stereo} + W_l \times Cost_{Learn} + W_c \times Cost_{classification}$$

where $Cost_{cell}$ is the cost to traverse each grid cell. $Cost_{Learn}$, $Cost_{classification}$, and $Cost_{stereo}$ are the fused costs in the world model based on the learning module, classification module, and stereo obstacle detection module. W_s, W_l, and W_c are weighting constants for each cost. If there is a bumper hit in the cell, $Cost_{cell}$ is assigned a fixed, high cost. The cost in each map cell represents the best estimate of traversability in the region represented by that cell, based on information fused over time. Each cost has a confidence associated with it and the map grid selects the label with the highest confidence. The final cost maps are constructed by taking the fused cost from all the sensory processing modules.

Except in one test, the weights W_s, W_l, and W_c were determined empirically by visually evaluating the cost maps on a large number of test runs. In Test 11, the weights were set by treating the results of the stereo obstacle detection as ground truth. If the learning algorithms reported results different from the obstacle detection, their weights (W_s, W_l,) were reduced. While the vehicle performed well in Test 11, there was no time to validate this approach, so it was not carried forward to Tests 12 and 13. An evaluation was later carried out using the same method and data sets as described in Section 5.1. It showed that the obstacle detection algorithm agrees 91% of the time with human classification. This justifies its use in setting the weights of the learning results.

The maps are updated with new sensor data and scrolled to keep the vehicle centered. When the vehicle moves out of the center grid cell of the map, a scrolling function is en-

abled. The map is vehicle-centered, so only the borders need to be initialized. Initialization information may be obtained from remembered maps saved from previous test runs. Remembering maps is a very effective way of learning about terrain, and has proved important in optimizing successive runs over the same terrain.

3.4 Behavior Generation

Top level input to Behavior Generation (BG) (Fig. 5) is the final goal point in UTM (Universal Transverse Mercator) coordinates. At the bottom of the 4D/RCS hierarchy, BG produces a speed for each of the two drive wheels updated every 20 ms, which is input to the low-level controller. The low-level controller returns status to BG, including motor currents, position estimate, physical bumper switch state, raw GPS and encoder feedback, etc.

Two position estimates are used in the system. Global position is strongly affected by the GPS antenna output and received signal strength and is more accurate over long ranges, but can be noisy. Local position uses only the wheel encoders and inertial measurement unit (IMU). It is less noisy than GPS but drifts significantly as the vehicle moves, and even more if the wheels slip.

The system consists of five separate executables. Each sleeps until the beginning of its cycle, reads its inputs, does some planning, writes its outputs and starts the cycle again. Processes communicate using the Neutral Message Language (NML) [9]. Each module also regularly outputs a status message that can be used by the supervising process and by developers via a diagnostics tool to monitor the process.

The LAGR Supervisor is the highest level BG module, responsible for starting and stopping the system. It reads the final goal and sends it to the waypoint generator. The waypoint generator chooses a series of waypoints for the lowest-cost traversable path to the goal using

global position and translates the points into local coordinates. It generates a list of waypoints using either the output of an A* Planner or a previously learned route to the goal. The planner takes a terrain grid (map) from WM and translates it into a grid of costs. In most cases the costs are simply looked up in a small table from the corresponding element of the input map. However, cells close to obstacles have their costs increased to provide clearance for safe vehicle motion.

The waypoint follower receives a series of waypoints, spaced approximately 0.6 m apart, which could be used to drive blindly without a map. However, there are some features of the path that make this less than optimal. When the path contains a turn, it is either at a 45° or 90° angle with respect to the previous heading. The waypoint follower could smooth the path, but it would at least partially enter cells that were not covered by the path chosen at the higher level. The A* planner might also plan through a cell that was partially blocked by an obstacle. The waypoint follower is responsible for avoiding the obstacle. The first step in creating a short range plan is to choose a goal point from the list provided by the A* Planner. One option would be to use the point where the path intersects the edge of the map. However, due to the differences between local and global positioning, this point might be on one side of an obstacle in the long range map and on the other side in the short range map. To avoid this, the first major turning point in the high level plan is selected. The waypoint follower searches a preset list of possible paths starting at the current position and chooses the one with the best score. The score represents a compromise between getting close to the turning point, staying far away from obstacles and higher cost areas, and keeping the speed up by avoiding turns.

The behavior generation module includes behavioral responses specific to different cir-

cumstances, including the following, implemented in the waypoint follower.

- AGGRESSIVE MODE—Ignore obstacles except those detected with the bumper and drive in the direction of the final goal. Terrain such as tall grass causes the vehicle to wander. Short bursts of aggressive mode help to get out of these situations.
- HILL CLIMB—Wheel motor currents and roll and pitch angle sensors are used to sense a hill. The vehicle attempts to drive up hills without stopping to avoid momentum loss and slipping.
- NARROW CORRIDOR/CLOSE TO OBSTACLE—In tight spaces the system slows down, builds a detailed world model, and considers a larger number of alternative paths to get around tight corners than in open areas.
- HIGH MOTOR AMPS/SLIPPING—When motor currents are high or the system thinks the wheels are slipping it tries to reverse direction and then tries a random series of speeds and directions, searching for a path where the wheels can move without slipping.
- REVERSE FROM BUMPER HIT—Immediately after a bumper hit the vehicle backs up and rotates to avoid the obstacle.

The lowest level module, the LAGR Comms Interface, takes a desired heading and direction from the waypoint follower, determines a vehicle-specific set of wheel speeds, and handles all communications between the controller and vehicle hardware.

3.5 Learning algorithms

Learning is a basic part of the LAGR program. Kinds of learning that have been addressed include learning by example, learning from experience, and learning of maps and paths. In the LAGR program, learning from sensor data has mainly focused on learning the traversability of terrain. This includes learning by seeing examples of the terrain and learning

from the experience of driving over (or attempting to drive over) the terrain.

3.5.1 Learning Classification in Color Vision

A color-based image classification module [13] runs independently from the obstacle detection module in SP1. It learns to classify regions in the scene by their colors, represented by color histograms. This enables it to provide information about obstacles and ground points even when stereo is not available. Pixels near the vehicle that fall into a 1 m wide by 2 m long rectangular area in front of the vehicle are used to construct and update the ground color histogram. The construction of the background model initially randomly samples the area above the horizon. Once the algorithm is running, the algorithm randomly samples pixels in the current frame that the previous result identified as background. These samples are used to update the background color model using temporal fusion.

The cost for each pixel is determined by a color voting method and the degree of belief that a point is background is calculated from the two histograms as

$$\frac{N_{background}}{N_{background} + N_{ground}}$$

where $N_{background}$ and N_{ground} are the number of hits in the corresponding histogram bin. In the case of multiple ground color histograms, the minimum cost is used. The cost image is sent to the world model. Fig. 7 illustrates color classification on the images shown in Fig. 6.

3.5.2 Learning using stereo-based labels

Another model-based learning process occurs in the SP2 module of the 4D/RCS architecture, which takes input from SP1 in the form of labeled pixels with associated *(x, y, z)* positions. Classification takes place in SP1 and uses the models to label the traversability of

image regions based only on data from the color cameras (without stereo).

An assumption is made that terrain regions that look similar will have similar traversability. The learning works as follows [11]. The system constructs a map of the terrain surrounding the vehicle, with map cells 0.2 m by 0.2 m. Each pixel passed up from SP1 has an associated red (R), green (G), and blue (B) color value in addition to its (x, y, z) position and label (obstacle or ground). Points are projected into the map using their (x, y, z) positions. Each map cell accumulates descriptions of the color, texture, intensity, and contrast of each point that projects into it. Color is represented by 8-bin histograms of ratios *R/G* and *G/B*. This provides some protection from the color of ambient illumination. Texture and contrast are computed using Local Binary Patterns (LBP) [7]. Texture is represented by a histogram with 8 bins. Intensity is represented by an 8-bin histogram, while contrast is a single number ranging from 0 to 1.

When a cell accumulates enough points, it is ready to be considered as a model. To build a model we require that 95% of the points projected into a cell have the same label (obstacle or ground). If a cell is the first to accumulate enough points, its values are used to create the first model. If other models already exist, the cell is matched to these models to see if it can be merged or if a new model must be created. Matching is done by computing a weighted sum of the elements of the model and the cell. Each model has an associated traversability, computed from the number of obstacle and ground points that were used to create the model. These models correspond to regions learned by example. Learning by experience is used to modify the models. As the vehicle travels, it moves from cell to cell in the map. If it is able to traverse a cell that has an associated model, the traversability of that model is increased. If it hits an obstacle in a cell, the traversability is decreased.

To classify a scene, only the color image is used. A window is passed over the image and color, texture, and intensity histograms and a contrast value are computed as in model building. A comparison is made with the set of models, and the window is classified with the best matching model if a sufficiently good match value is found. Regions that do not find good matches are left unclassified.

Fig. 8(a) shows an image taken during learning. The pixels contributing to the learning are shown in white for obstacle points and black for ground points. Fig. 8(b) shows a scene labeled with traversability values computed from the models built from previous data.

4. Results

Testing in the LAGR program is done by an independent Government team using a vehicle functionally identical to the vehicles on which the software is developed. Tests occur about once a month, with a total of thirteen tests in the first phase of the program. Developers send their control software on flash memory cards to the test facility. The software is loaded onto a vehicle, which is commanded to travel from a start waypoint to a goal waypoint through an obstacle-rich environment. The environment is not seen in advance by the development teams. The Government team measures 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 require learning from examples.

The NIST team got off to a slow start, not managing to complete the courses for the first few tests. This was partly due to the effort required to replace the LAGR-supplied control system with the 4D/RCS architecture. Once this was in place and we had gained some experience in what was expected of the vehicle, performance improved significantly.

We briefly discuss the results of the tests, starting with Test 4.

Test 4 was conducted at Fort Belvoir, VA in June, 2005. The course traveled through a field containing both natural and artificial barriers. The course started in a wooded area, crossed an open field with fences and boxes as obstacles, and ended in another wooded area. One of the fences was placed in the path of the vehicle in the form of a cul de sac, in a way that ensured that the vehicle would have to enter the dead end and find its way out. The Government team expected the vehicles to learn that objects that looked like bright orange fences were obstacles. The NIST control system was able to complete the course twice, failing on the first run when it got caught in some bushes. On the second and third runs it used a behavior called wall following to negotiate the fences, get out of the cul de sac, and make its way to the goal. The learning used in this test was to remember the path from run to run, and both successful runs took about 4.5 minutes.

Test 5 took place in New Hampshire in August, 2005. There were two courses on this test, each of which was attempted three times. The first course started in a meadow, went through some woods with dense foliage, and ended in another meadow. For this course, the NIST vehicle was defeated not by an inability to navigate, but by not being able to climb a slope in the woods. The vehicle was not able to maintain a constant velocity when climbing, and kept slipping on the pine needles on the ground. It was following this test that the hill climbing behavior was developed. The second course started in an open meadow with tall grass and vegetation. It traveled down a grassy trail through the woods and into another field. It then turned right and ran along the edge of the woods, climbed a small embankment ended in the middle of the clearing. The only traversable path from start to goal followed the trail.

This was the first course on which the NIST vehicle demonstrated learning effectively. The maps were saved from run to run and the path remembered after being optimized to remove loops. In the first run, the vehicle suffered from wheel slip. This had the unfortunate consequence of causing the map to scroll because the vehicle thought it was moving. As a result, obstacles appeared in the wrong places. This caused the vehicle to take a non-optimal path, but it was still able to reach the goal in 6:47 minutes. Given a successful first run, the saved map and path were used in the second two runs, enabling the vehicle to make two straightforward runs to the goal, in 2:48 and 2:50 minutes, respectively.

Test 6 returned to the site at Fort Belvoir, VA, using the same location as Tests 1-4. The test followed a path through a slightly wooded area, ending in an open field. Two orange plastic fences were placed in the path of the vehicle, with the goal being to learn that the first fence represented an obstacle and to use that knowledge to avoid the second fence. The three runs are shown graphically in Fig. 9. The NIST system was able to clearly show learning in this test. In the first run, it had to search to find a gap in the fence, and was also delayed by wandering through some tall grass. Fig. 10 shows graphically how the path was optimized and remembered for later runs. The white line is the path taken in the first run, showing the exploration of unfruitful areas. The grey line is from the second run. The loops and excursions have been eliminated enabling the vehicle to substantially reduce the distance it has to travel. In this test, the NIST vehicle reduced its time between each successive run, from 5:03 in the first run, to 4:34 in the second, and 3:20 in the third.

Test 7 was conducted in October 2005 at Fort Belvoir. 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 divided the course into a region to the left that made it difficult to reach the goal, and a region to the right that made it easy to reach the goal.

The Government team expected the vehicles to either immediately see and understand the superior traversability of the route to the right of the obstacle, or 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 their speed from run to run. Test 7 introduced a new expectation for the LAGR vehicles: the ability to see beyond the distance at which reliable stereo range data could be acquired. The approach the NIST team took to address this was to use the region in which good stereo could be obtained to learn models of the terrain as described in Section 3.5.2. These models were then matched with regions in the images that corresponded to terrain between the end of reliable stereo and the horizon.

The NIST system matched the expectations of the Government team by learning from the first run and using the learned information to improve on subsequent runs. In the first run, having no knowledge of the course and no models of the ground and obstacles, the vehicle 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. Fig. 11 shows the sequence of events in the first and second runs. Each subfigure shows a view from the Operator Control Unit (OCU) of the vehicle. The OCU displays a raw

image from each stereo camera (top) and the result of classifying the images using the current set of traversability models (bottom).

In Fig. 11(a) 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 black (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 black, but the corresponding region in the right eye did not match the initial model well enough to be classified. Fig. 11(b) shows the situation a little later when the barrier is in the field of view. A new model is constructed for part of the barrier, shown in white (not traversable). Part of the barrier was classified as ground (black) 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 correctly. 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.

Fig. 11© shows a view of the barrier taken after models have been built over the entire first run. The region from in front of the vehicle to the horizon is shown classified. Traversable regions are shown in black and non-traversable regions in white. Fig. 11(d) 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 (black for ground, white for obstacles, and mid-grey 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 light grey 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.

Fig. 11(e) shows roughly the same location as that in Fig. 11(d), 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. In the second run, it applied these models to the region of the image beyond the stereo reliability range. In the OCU, the results show up as the black (ground) and white (obstacle) bars in the middle images. As can be seen, this information starts where the black region corresponding to the stereo processing ends. The light grey line in Fig. 11(e) 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 travel directly to the goal in 1 minute and 32 seconds.

Test 8 was targeted at learning from examples. A set of training data was provided in advance, and involved a path made of white lime laid down across a grassy field to show the vehicle where it was expected to go. The vehicle was supposed to learn from this example how to recognize the path. For this test, NIST made use of the color-based learning described in Section 03.5.1. This was originally developed for road following, so was naturally suited for following the white path. On the first run, the NIST vehicle started following the path, but lost sight of it when it approached a tree. The vehicle then turned left and got into a maze area made of hay bales and did not complete the course. Run 2 began like Run 1, but the system got back onto the lime path, and followed it to the goal. Run 3 followed the lime path all the way, setting the course speed record.

Test 9 was held in San Antonio, TX 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 models, so the intensity and texture measures in the models were expected to provide most of the discrimination. The course began on a lane mown through a field with trees, bushes, brush, and tall grass and traveled along the lane to 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. Where the lane reached the open field, a non-traversable patch of tall grass grew on the left-hand side of the lane.

The Government team expected the robot to follow the lane from start to goal. They hoped that in earlier runs, the robot would explore the left-hand turns, but that later 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 expected, 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.

The first row of Fig. 12 shows three images taken during the first run. Models are shown overlaid on the images in shades of grey, with each shade representing a different model but not necessarily a different traversability. The grey pixels are overlaid on the images at locations where points from the stereo data projected into the local map. The

rectangular shape of the cells explains why some of the regions, such as the white region in Fig. 12.(a), appear rectangular. Parts of the image that are not colored with a model can arise if 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 typically be labeled in subsequent images.

The second row of Fig. 12 shows interpretations of the scene in terms of traversability. Points shown in black are considered traversable, while those in white are not traversable. The black and white blocks correspond to classifications using 16x16 windows in the region beyond stereo, with black representing traversable regions and white representing obstacles. Each window is used to construct a set of color and texture distributions that is matched with the models to provide the classifications.

The last row of Fig. 12 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 16x16 window is passed over the image and each block is matched with a model. If the matching model is traversable, that block of the image is colored black. If it is not traversable, the window is colored white. 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. 12(g-i) will have their traversability increased if the vehicle drives over the corresponding terrain.

Test 10 took place in San Antonio, TX in January, 2006 and was divided into multiple sub-tests. The goal of the first sub-test was to be able to cope without stereo. The first run of

the vehicle had stereo available, but then one of the cameras in each stereo pair was covered, giving only monocular vision cues to the system. The idea was that the system would learn to recognize traversable and non-traversable regions in the first run, and then apply that knowledge in later runs. The test took place in an open field of mowed grass. A rectangular area was defined where a total of 28 narrow green posts were randomly placed. The Government team expected the robot to learn from the training run with stereo that green posts were obstacles. They expected the robot to follow a path through the posts and across the open field to the goal.

In the first run, with stereo, the vehicle using the NIST controller maneuvered through the post field without hitting anything. During this run, it learned stereo-based models of the ground and the obstacles as described in 3.5.2. In the next run, without stereo, the vehicle hit two of the posts and placed them in its map as obstacles. In the third run, it hit one post. In both cases it navigated to the goal. The learned models were able to identify traversable and non-traversable regions most of the time based only on information from color and texture. The second sub-test in Test 10 was a replay of Test 9, using the same terrain except that the vegetation was drier and more beaten down from use. The NIST vehicle was one of two that completed the course on all three runs and it had the best times of all the teams.

Test 11 was also conducted in Texas. The course ran through a sparsely wooded area with three potential paths. The straight-line path was not traversable. A path was laid out by the Government team as the best choice, and there was another route that left the optimal path after a short distance, was slower to navigate, but did lead to the goal. The optimal path traveled down a lane that did not lead directly to the goal. It was marked with mulch in an

effort to provide clear color and texture cues. The alternative path branched off from the optimal path and led more directly towards the goal point. In taking this path, the vehicle had to navigate through trees, bushes, hay bale barriers, and a cul-de-sac. The Government team hoped that the vehicle would take the optimal path, but even though there was mulch laid down, the vehicles couldn't know just from image data that it was a good bet to follow the mulch rather than turn off on a path that was more direct. The NIST vehicle always took the right hand turn onto the more complex course. In the first run, it explored the cul de sac, but learned that it was not productive. In the two later runs, it removed the exploration loop from its stored path and went straight to the goal.

Test 12 was another test of learning from example. A training course was laid out, consisting of a path defined by brown mulch, which took an "S" turn through two sets of three hay bales. The vehicle was driven manually along the path collecting data for learning the appearance of the path. The actual test took place in an open field. One path marked with brown mulch followed a path across the field and through a gap between two pine trees. Another unmarked path traveled through a maze of hay bales. The expectation was that the vehicles would follow the mulch path, since that is what they were trained on. The NIST vehicle did not follow the path. It took the alternate path to the goal each time. This was because of continuous model updating during the run (using the learning algorithm described in Section 3.5.1), which invalided the pre-trained model when the vehicle overshot the path. Because several other teams also failed to follow the path, a retest was done. This time, the continuous model updating was removed and the vehicle was successfully able to recognize the mulch path.

The last test, Test 13, took place in Virginia in May, 2006. In this test there were many possible paths to the goal, but four that looked like reasonable routes to follow. The Government team expected the vehicles to follow the third or fourth of these routes, which were the most traversable. In this test, the NIST vehicle used a number of variations for learning. In all the runs the maps and paths were saved, but each run used a different variation of image-based traversability learning. In the first run, the color model classifier based on the ground in front of the vehicle (Section 3.5.1) was adjusted to learn for only the initial 3 seconds of the run. This classifier was modified to use normalized hue, saturation, value (HSV) information as well as RGB to construct the models. It then used the learned models to classify the images for the rest of the run. On this run, the vehicle explored for a while, but then found the most desirable path and took it to the goal. On the second run, the vehicle learned throughout the run using the color and texture models with traversability labels from stereo (Section 3.5.2). On this run, the vehicle found the desirable path and went directly to the goal. On the third run, only the saved map was used and the algorithm performed as in run two.

5. Performance Evaluation

The LAGR tests evaluated the control systems of the LAGR teams as a whole, and particularly the learning components. The performance of the vehicles was evaluated by comparison with a baseline system developed by Carnegie Mellon University. At the start of the program, it was considered state of the art for vehicle control, but it did not have any learning capability. The LAGR systems were required to beat the baseline system by 10% by the end of the first phase of the program. While the baseline scored well in the first few tests, by the end it was routinely beaten by all the participants, even when some of them were not

using learning. While this form of evaluation was effective, it did not give a good indication of which components of the control systems were providing the greatest value. To explore this issue, NIST has started evaluating individual learning components of the system.

5.1 Evaluating Terrain Traversability Learning

A method was developed of evaluating the algorithm described in Section 3.5.2. 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 evaluation consisted of log files generated during the tests. Log files contain the sequence of images collected by the cameras on the LAGR vehicle and information from the other sensors, including the navigation and bumper sensors. The NIST system performs exactly the same when playing back a log file as it did when it 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. 13). 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.

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 tests conducted at locations in Virginia and Texas. Results are shown for these examples and an overall evaluation is given of the performance of the algorithm across all the data sets. 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. 13. 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 approximately 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 any 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.

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.

5.2 Evaluating Algorithm Parameters

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 6 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 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 texture measures based on multi-resolution Gabor filters as in [12] 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. Summary and Conclusions

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.

The role of 4D/RCS in balancing the system comes from the breakdown of processing components into units, each of which considers a small part of the problem. In the LAGR system, this is most evident in the world model and planner, since most of the sensory processing is carried out in only one of the SP levels. The learning algorithms are spread over the modules and work on the information available at their level in the hierarchy. For example, there are two planners, one that uses information in WM1 and makes plans that look out about 20 m from the vehicle. The other planner uses information in WM2 and makes lower resolution plans to reach the goal or to the edge of the map, where it chooses the point on the boundary closest to the goal.

Another issue is assigning processing to computing resources on the vehicle. There are four processors, one of which runs the low-level control of the vehicle and is not

available for other use. In our implementation, two of the processors are dedicated to sensory processing. One processor is assigned to each of the stereo cameras, and the third is used for both planning and world modeling. Sensory processing and the associated learning algorithms run independently in each SP processor. Both SP1 and SP2 processes run simultaneously in the processors for the left and right eyes and communicate using shared memory. They send their output to the WM processes over the Ethernet. Because of the substantial processing load of stereo processing and obstacle detection, the learning and classification processes have limited resources and can't be too complicated. Both the world model (WM1 and WM2) and the behavior generation (BG1 and BG2) run on the third processor. This has proved to be a good distribution of tasks to resources, but the processors in the system run at close to their limits. Sometimes the vehicle has to slow down to ensure that obstacles are detected, inserted into the world model, and seen by the planner in time to be avoided.

The learning methods developed for the LAGR program have constantly been expanded as the program has progressed. The LAGR Government team provided general guidelines for what must be learned, but have introduced challenges including requiring the vehicle to learn to follow a path, to learn to recognize obstacles in the distance so as to avoid entering a region with no outlet, and to operate without stereo. This has led to the requirement that learning be flexible. It has also led us to incorporate learning capability throughout the system because the different modules can learn different aspects of the environment. This enables quick response to the challenges.

Participating in the LAGR program involves developing a lot more than the learning methods highlighted in its name. Before learning can be accomplished, the vehicle itself

must be controlled and the sensor data processed to provide feedback. One of the goals for NIST was to demonstrate the capabilities of the 4D/RCS architecture to control the vehicle, and to show how learning could be embedded into 4D/RCS. To do this, the controller provided with the vehicle was replaced as completely as possible with a 4D/RCS two-level hierarchy. The series of tests conducted under the program showed that the goal was achieved. It also showed how the modularity of the control system enabled individual learning components to be added over time to improve the system's performance. For example, the two learning methods in the sensory processing modules were developed independently and added at different times, as were the algorithms for saving the maps and for optimizing the planned paths. As new versions were developed, they could be swapped in or out to determine their value. When the LAGR Government Team devised tests requiring new capabilities, these could be implemented and added to the control system without requiring changes in unrelated parts of the system. This made it possible to manage the time pressure imposed by the monthly test schedule.

The learning in each of the modules is not simply added on to the process that implements the module. It is embedded as part of the module, and operates in accordance with its location in the hierarchy. Thus, model learning in SP1 uses simpler features than that in SP2, maps learned in WM1 and WM2 have different sizes and attributes, and plans in BG have different resolution and range. The LAGR program has provided the opportunity to develop learning in 4D/RCS, which promises to substantially enrich the demonstrated capabilities of the architecture.

Acknowledgements

We are grateful to Larry Jackel at DARPA for supporting this work and to the LAGR

Government team, whose constructive interaction with us during the tests has been both enjoyable and helpful.

References

[1] J. S. Albus, H-M Huang, E. Messina, K Murphy, M Juberts, A. Lacaze, S. Balakirsky, M. O Shneier, T. Hong, H. Scott, J. Horst, F Proctor, W. Shackleford, S. Szabo, and R. Finkelstein, "4D/RCS Version 2.0: A Reference Model Architecture for Unmanned Vehicle Systems," National Institute of Standards and Technology, Gaithersburg, MD, NISTIR 6912, 2002.

[2] J. Albus, R. Bostelman, T. Chang, T. Hong, W. Shackleford, and M. Shneier, "Learning in a Hierarchical Control System: 4D/RCS in the DARPA LAGR Program," *Journal of Field Robotics, Special Issue on Learning in Unstructured Environments (in press)*, 2006.

[3] S. Balakirsky, E. Messina, and J. S. Albus, "Architecting a Simulation and Development Environment for Multi-Robot Teams," Proceedings of the International Workshop on Multi Robot Systems, 2002.

[4] Ayanna Howard, Edward Tunstel, Dean Edwards, and Alan Carlson, "Enhancing fuzzy robot navigation systems by mimicking human visual perception of natural terrain traversability," Joint 9th IFSA World Congress and 20th NAFIPS International Conference, 2001, pp. 7-12.

[5] Jackel, L. Learning Applied to Ground Robots (LAGR). http://www.darpa.mil/ipto/programs/lagr/ . 2005.

[6] Konolige, K. SRI Stereo Engine, http://www.ai.sri.com/~konolige/svs/. 2006. SRI International, Menlo Park, CA.

[7] T. Ojala, M. Pietikainen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29 1996, pp. 51-59.

[8] D. N. Oskard, T. Hong, and C. A. Shaffer, "Real-Time Algorithms and Data Structures for Underwater Mapping," Proceedings of the SPIE Advances in Intelligent Robotics Systems Conference, 1988.

[9] W. P. Shackleford, F. M. Proctor, and J. L. Michaloski, "The Neutral Message Language: A Model and Method for Message Passing in Heterogeneous Environments," Proceedings of the 2000 World Automation Conference, 2000.

[10] A. Shirkhodaie, R. Amrani, N. Chawla, and T. Vicks, "Traversable Terrain Modeling and Performance Measurement of Mobile Robots," Performance Metrics for Intelligent Systems, PerMIS '04, 2004.

[11] M. Shneier, T. Chang, T. Hong, and W. Shackleford, "Learning Traversability Models for Autonomous Mobile Vehicles," *Autonomous Robots (submitted)*, 2006.

[12] A. Talukder, R Manduchi, R. Castano, L. Matthies, A. Castano, and R. Hogg, "Autonomous Terrain Characterisation and Modelling for Dynamic Control of Unmanned Vehicles," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2002, pp. 708-713.

[13] Ceryen Tan, Tsai Hong, Michael Shneier, and Tommy Chang, "Color Model-Based Real-Time Learning for Road Following," Proceedings of the IEEE Intelligent Transportation Systems Conference, 2006.

[14] Carl Wellington and Anthony Stentz, "Learning Predictions of the Load-Bearing Surface for Autonomous Rough-Terrain Navigation in Vegetation," International Conference on Field and Service Robotics, 2003, pp. 49-54.

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 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%	

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
Effects on Classification of Changing Model Parameters

Test 7 Model Parameter Variation					
No Texture		No Color		Only Color	
% Correct	% Incorrect	% Correct	% Incorrect	% Correct	% Incorrect
83.52%	16.48%	53.26%	46.79%	86.25%	13.75%
Test 9 Model Parameter Variation					
No Texture		No Color		Only Color	
% Correct	% Incorrect	% Correct	% Incorrect	% Correct	% Incorrect
82.35%	17.99%	76.12%	24.22%	56.40%	43.94%

Figure captions

Fig. 1. The DARPA LAGR vehicle

Fig. 2. The basic internal structure of a 4D/RCS control loop

Fig. 3. Two-level instantition of the 4D/RCS hierarchy for LAGR.

Fig. 4. OCU display of the World Model cost maps built from sensor processing data. WM1 builds a 0.2 m resolution cost map (left) and WM2 builds a 0.6 m resolution cost map (right).

Fig. 5. Behavior Generation High Level Data Flow Diagram.

Fig. 6. OCU display showing original images (top), results of obstacle detection (middle), and cost maps (bottom). White represents obstacles, black is ground, and mid-grey represents obstacles too far away to classify.

Fig. 7. OCU display showing original images (top) and cost images (bottom). The 1 m wide by 2 m long rectangular areas assumed to be ground (white boxes) are overlaid on the cost images.

Fig. 8. Learning by example images. (a) is an image taken during learning and overlaid with (white) obstacles and (black) ground, (b) is the same image overlaid with traversability information as obstacles (white) and ground (black).

Fig. 9. Tracks from three runs of the NIST vehicle in Test 6. Run 1 is shown in black, Run 2 in white, and Run 3 in grey.

Fig. 10.Graph showing path optimization. The white line shows the path taken in the first run. The grey line is the optimized path used on the second run.

Fig. 11. Learning about obstacles to improve path planning. (a) View at the beginning of the first run. Only models of the ground have been created (black). (b) View when the barrier is seen. A model for an obstacle is created (white). (c) Result of classifying the scene out to the horizon using the learned models (ground is black, obstacles are white). (d) Light grey line shows planned path at start of first run. (e) Light grey line shows planned path at start of second run.

Fig. 12. Examples of model learning and classification. (a-c) Models constructed as the vehicle traverses the course. Each shade of grey corresponds to a separate model. (d-f) Traversability computed from the models. Black is traversable, white is not. Classification computed from the models in the range beyond stereo is shown as black for traversable and white for not traversable. (g-i) Result of classifying the terrain using the models. Black corresponds to traversable, white to obstacles.

Fig. 13. The GUI for generating ground truth showing a frame from Test 7.

Figures

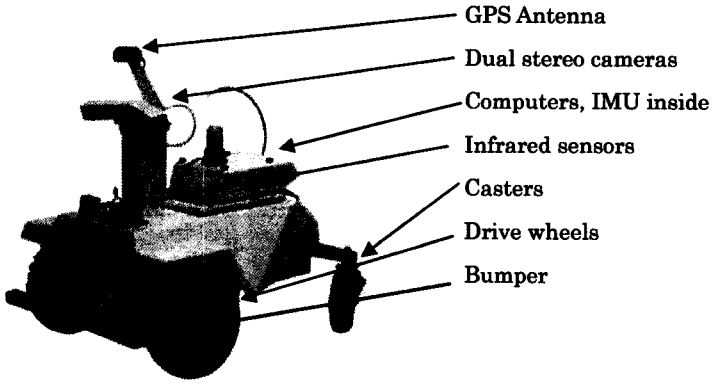


Fig. 1.

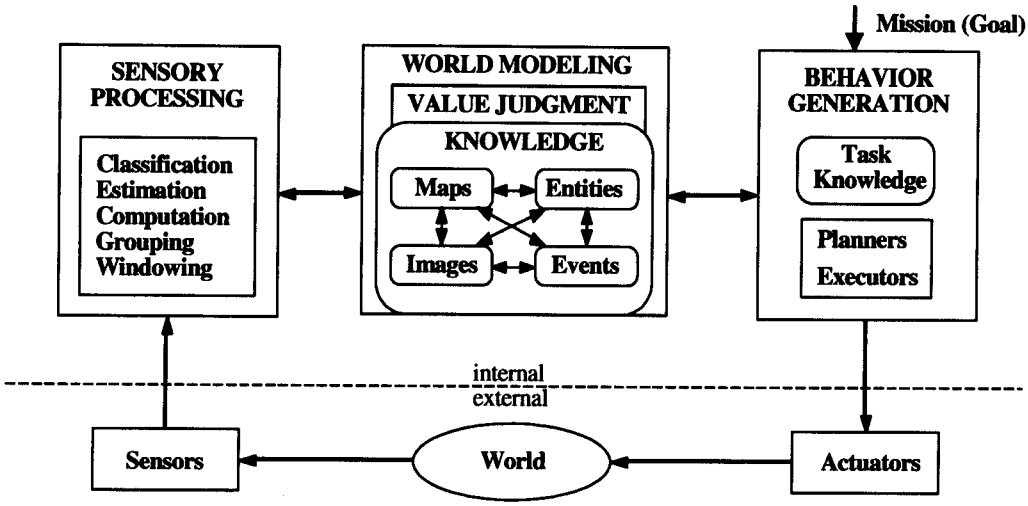

Mission (Goal)
SENSORY PROCESSING
Classification
Estimation
Computation
Grouping
Windowing
WORLD MODELING
VALUE JUDGMENT
KNOWLEDGE
Maps
Entities
Images
Events
BEHAVIOR GENERATION
Task Knowledge
Planners
Executors
internal
external
Sensors
World
Actuators

Fig. 2.

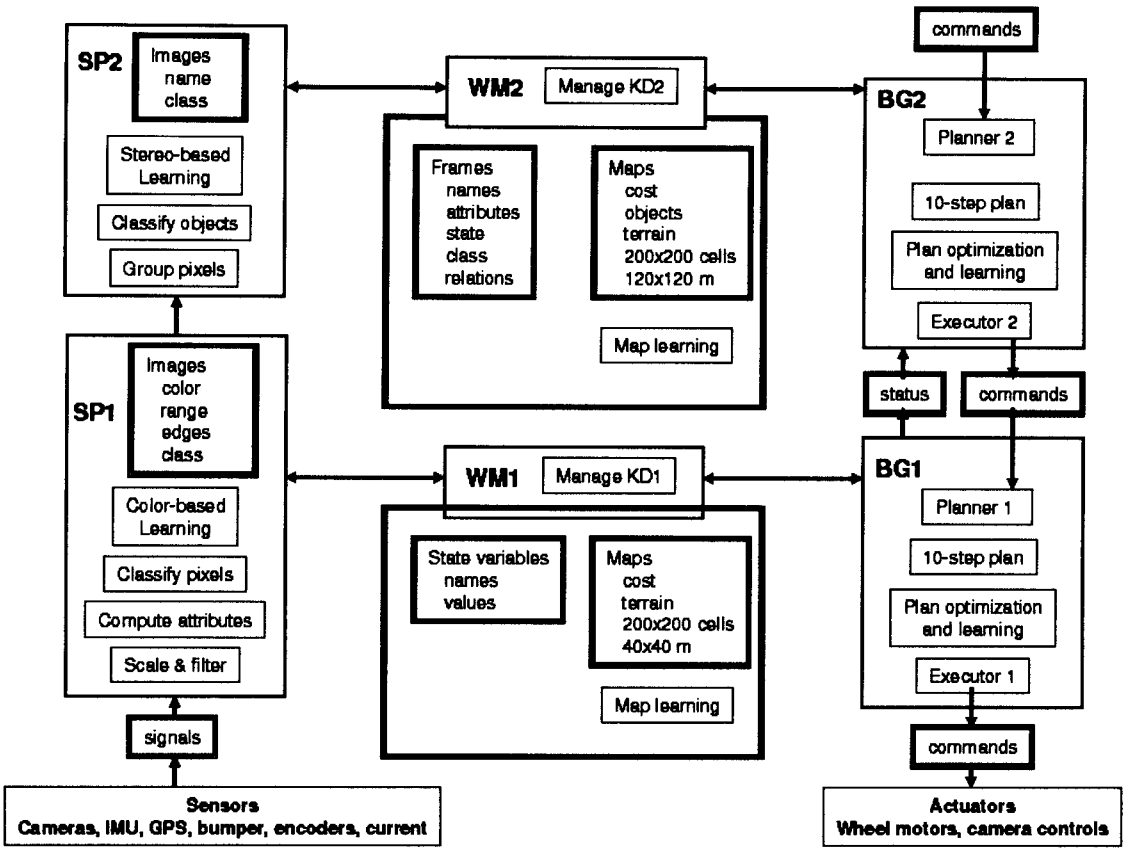

SP2
Images
name
class
Stereo-based
Learning
Classify objects
Group pixels
SP1
Images
color
range
edges
class
Color-based
Learning
Classify pixels
Compute attributes
Scale & filter
signals
Sensors
Cameras, IMU, GPS, bumper, encoders, current
WM2
Manage KD2
Frames
names
attributes
state
class
relations
Maps
cost
objects
terrain
200x200 cells
120x120 m
Map learning
WM1
Manage KD1
State variables
names
values
Maps
cost
terrain
200x200 cells
40x40 m
Map learning
commands
BG2
Planner 2
10-step plan
Plan optimization
and learning
Executor 2
status
commands
BG1
Planner 1
10-step plan
Plan optimization
and learning
Executor 1
commands
Actuators
Wheel motors, camera controls

Fig.3.

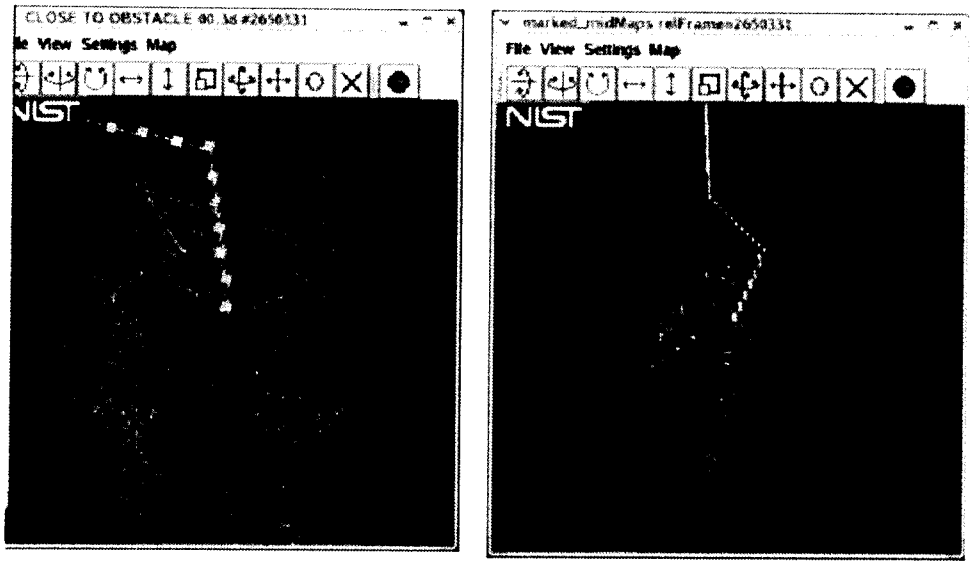


Fig. 4.

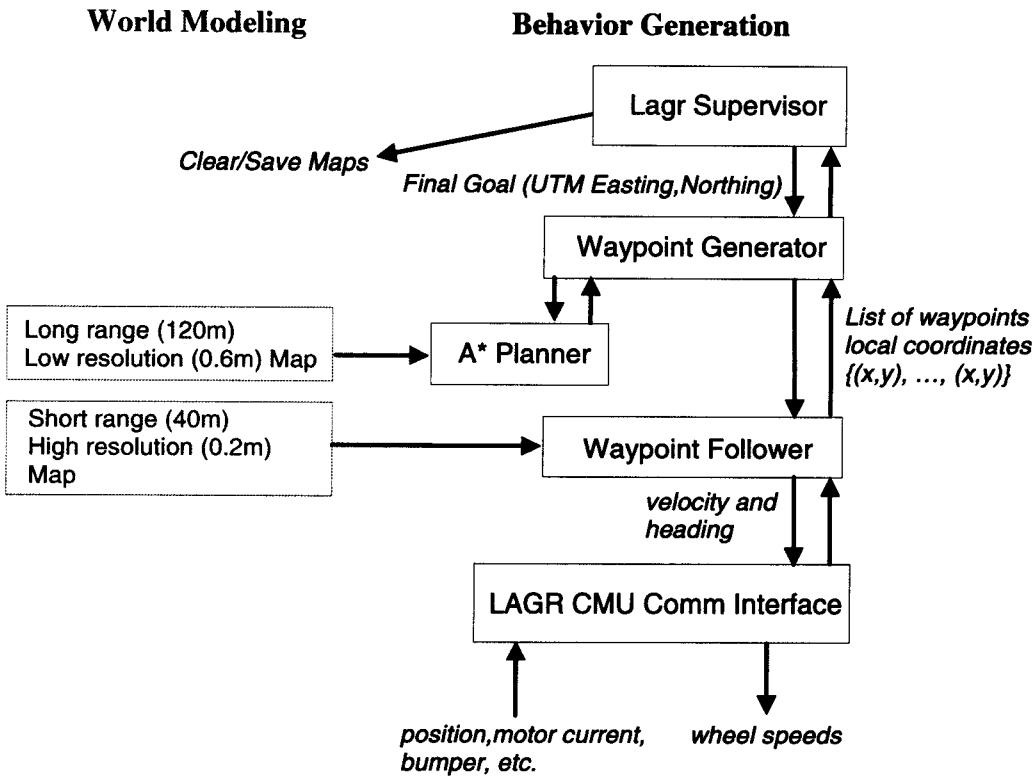

World Modeling
Behavior Generation
Lagr Supervisor
Clear/Save Maps
Final Goal (UTM Easting,Northing)
Waypoint Generator
Long range (120m)
Low resolution (0.6m) Map
A* Planner
List of waypoints
local coordinates
{(x,y), ..., (x,y)}
Short range (40m)
High resolution (0.2m)
Map
Waypoint Follower
velocity and
heading
LAGR CMU Comm Interface
position,motor current,
bumper, etc.
wheel speeds

Fig. 5.

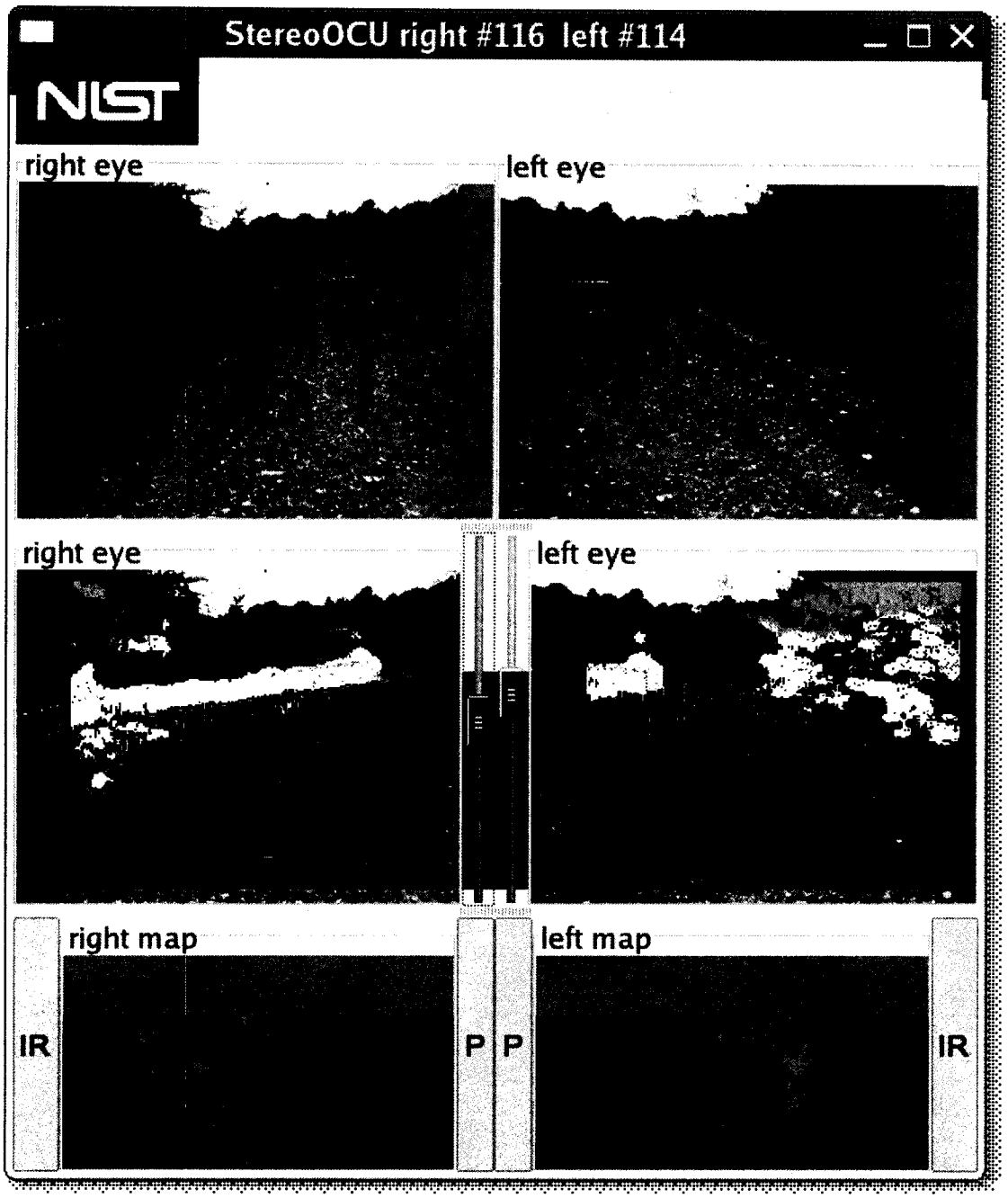

StereoOCU right #116 left #114
NIST
right eye
left eye
right eye
left eye
right map
left map
IR
P
P
IR

Fig. 6.

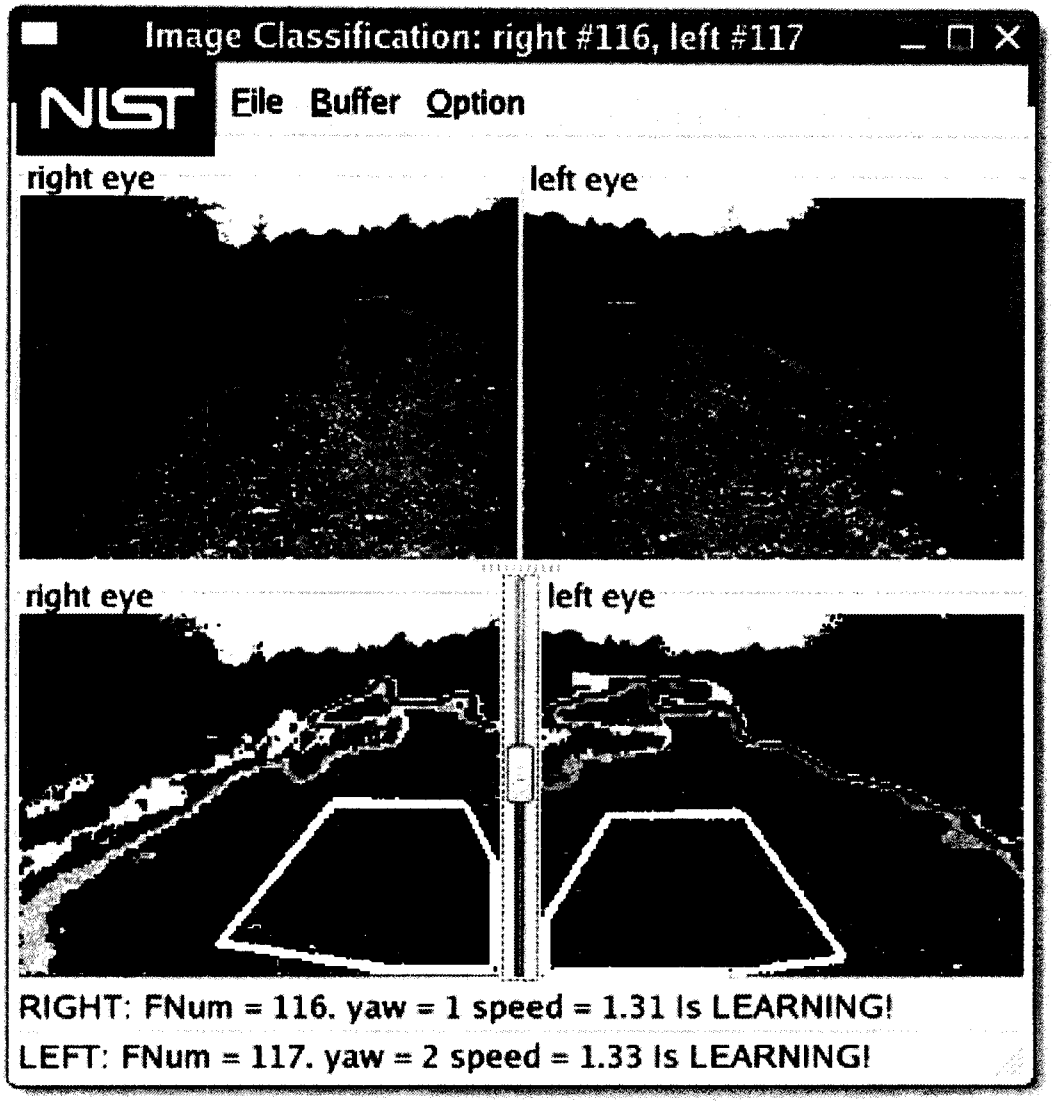
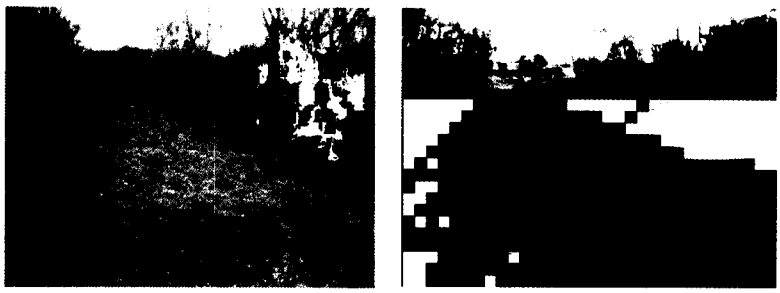


Image Classification: right #116, left #117
NIST
File Buffer Option
right eye
left eye
right eye
left eye
RIGHT: FNum = 116. yaw = 1 speed = 1.31 Is LEARNING!
LEFT: FNum = 117. yaw = 2 speed = 1.33 Is LEARNING!

Fig. 7.



(a) (b)

Fig. 8.

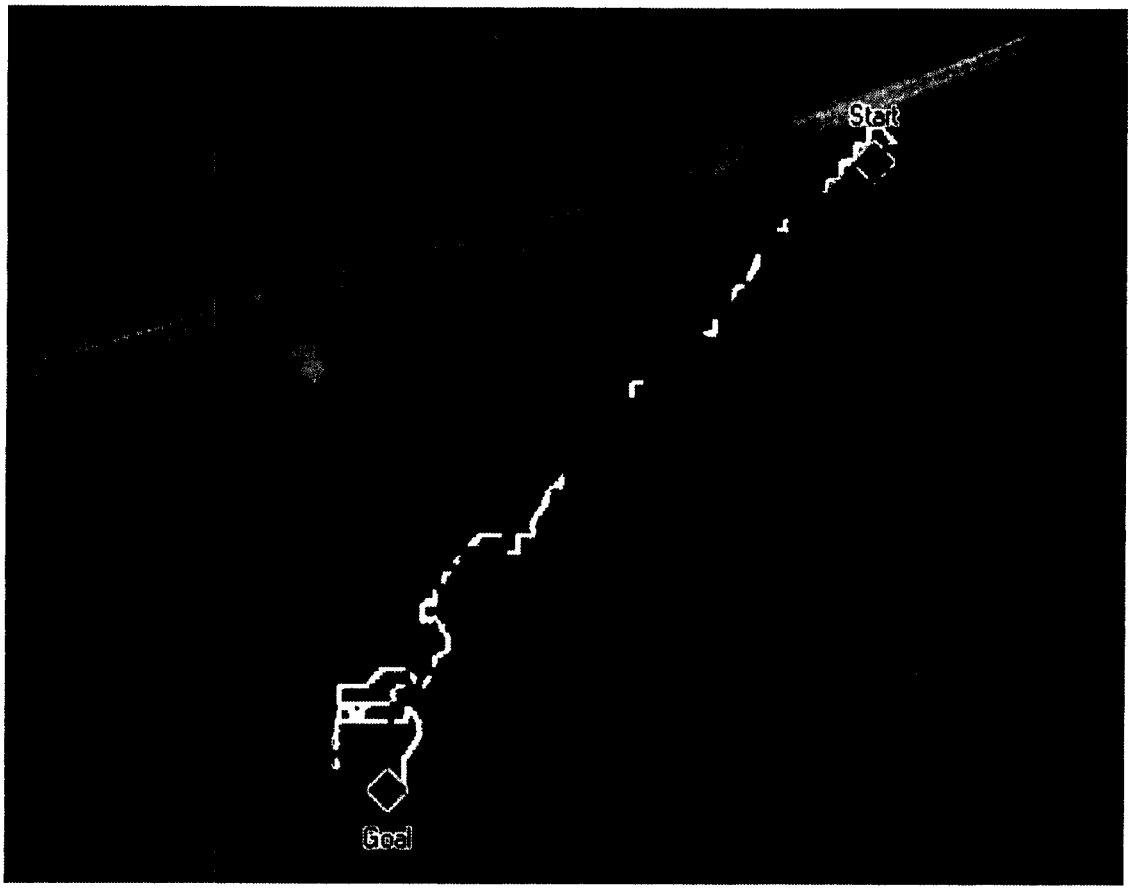


Fig.9.

Fig. 10.

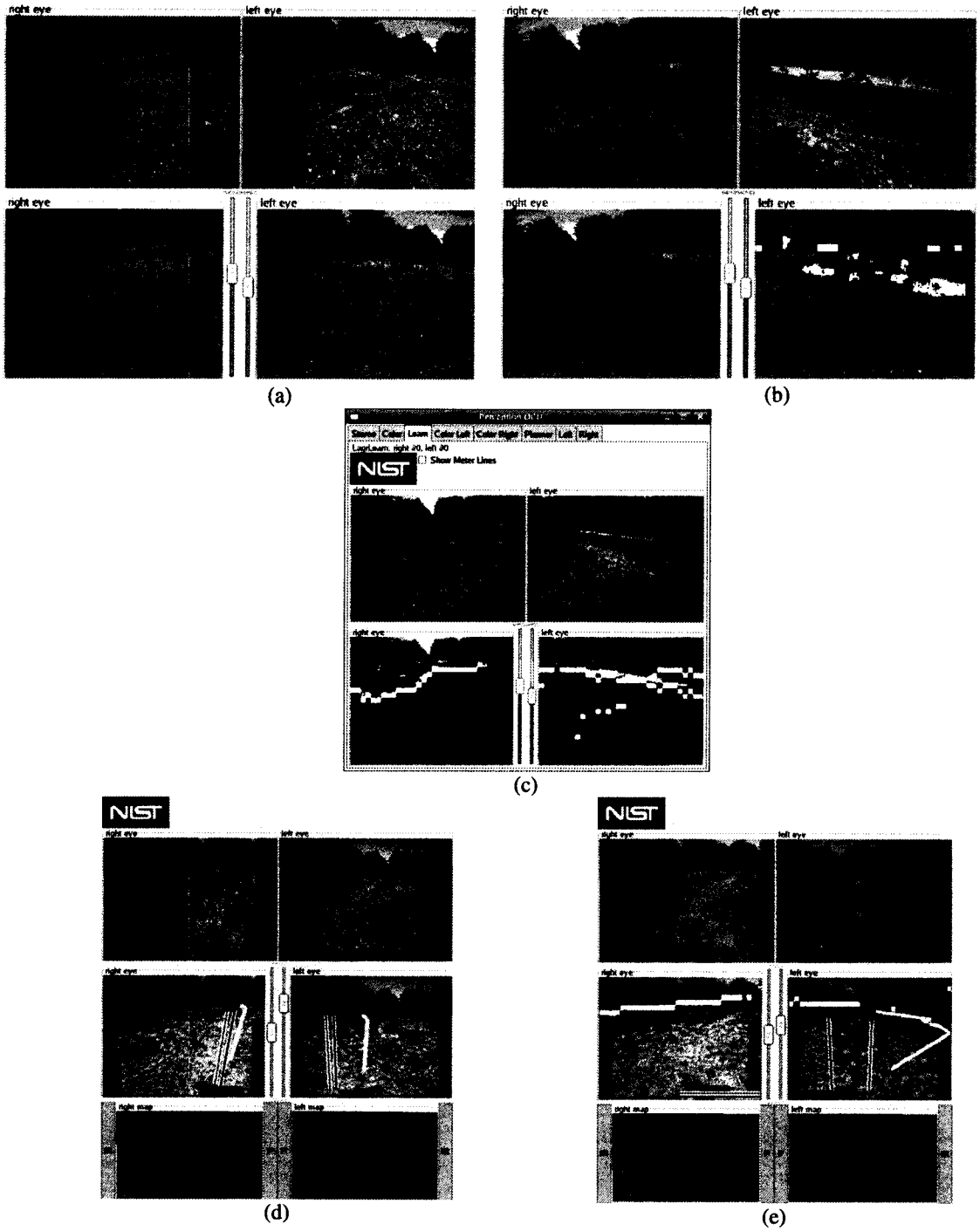

right eye
left eye
right eye
left eye
right eye
left eye
right eye
left eye
(a)
(b)
NIST
right eye
left eye
right eye
left eye
(c)
NIST
NIST
(d)
(e)

Fig. 11.

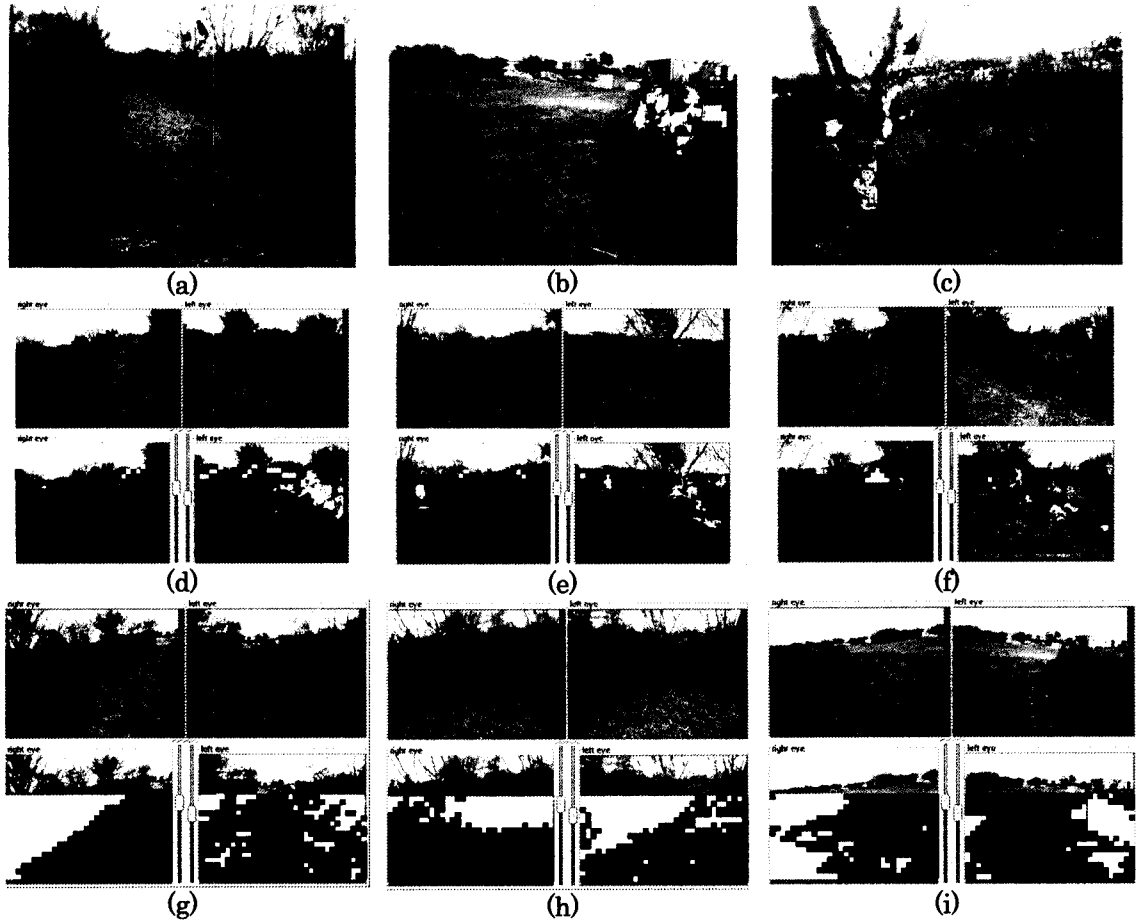


Fig. 12.

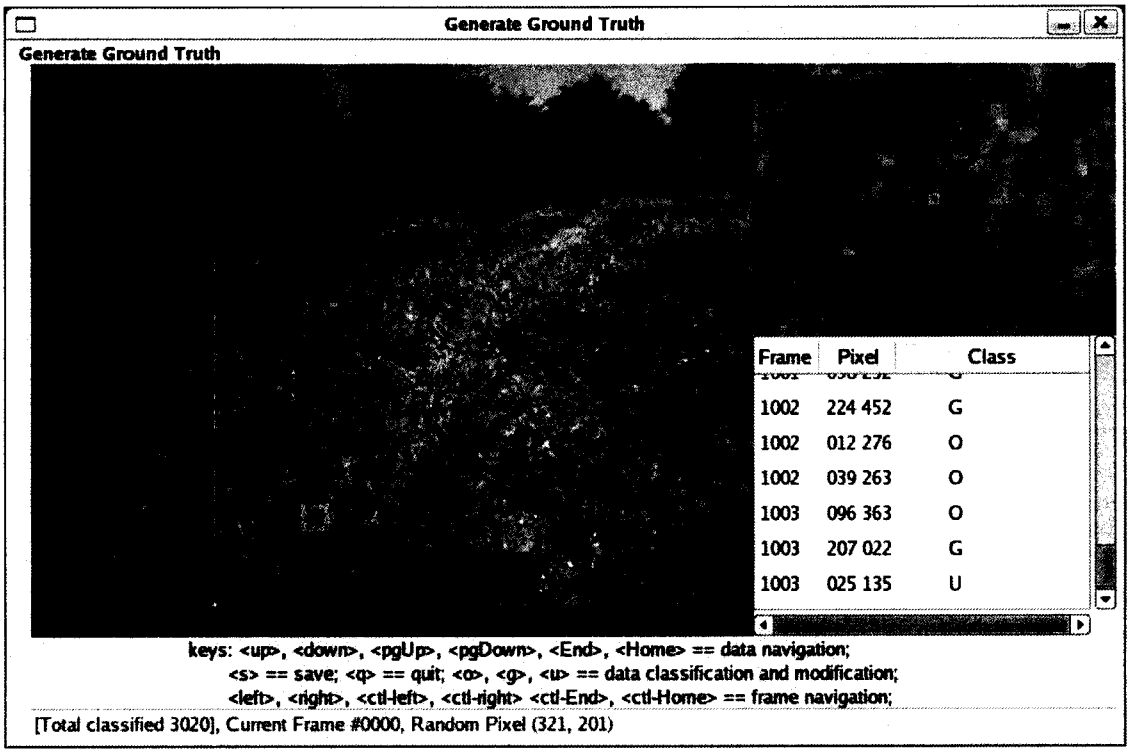

Generate Ground Truth
Generate Ground Truth
Frame
Pixel
Class
1002 224 452 G
1002 012 276 O
1002 039 263 O
1003 096 363 O
1003 207 022 G
1003 025 135 U
keys: <up>, <down>, <pgUp>, <pgDown>, <End>, <Home> == data navigation;
<s> == save; <q> == quit; <o>, <g>, <u> == data classification and modification;
<left>, <right>, <ctl-left>, <ctl-right> <ctl-End>, <ctl-Home> == frame navigation;
[Total classified 3020], Current Frame #0000, Random Pixel (321, 201)

Fig. 13.