International Journal of Computer Vision 74(3), 287–302, 2007

DOI: 10.1007/s11263-006-0024-x

Reverse Optical Flow for Self-Supervised Adaptive Autonomous Robot Navigation

A. LOOKINGBILL, J. ROGERS, D. LIEB, J. CURRY AND S. THRUN
Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, CA 94305-9010, USA
apml@stanford.edu
jgrogers@stanford.edu
curryj@stanford.edu
dlieb@stanford.edu
thrun@stanford.edu

Received November 1, 2005; Accepted December 11, 2006

First online version published in January, 2007

Abstract. Autonomous mobile robot navigation, either off-road or on ill-structured roads, presents unique challenges for machine perception. A successful terrain or roadway classifier must be able to learn in a self-supervised manner and adapt to inter- and intra-run changes in the local environment.

This paper demonstrates the improvements achieved by augmenting an existing self-supervised image segmentation procedure with an additional supervisory input. Obstacles and roads may differ in appearance at distance because of illumination and texture frequency properties. Reverse optical flow is added as an input to the image segmentation technique to find examples of a region of interest at previous times in the past. This provides representations of this region at multiple scales and allows the robot to better determine where more examples of this class appear in the image.

Keywords: off-road navigation, terrain classification, optical flow, mobile robots, computer vision

1. Introduction

Autonomous mobile robot navigation in challenging, real-world environments is currently an area of great excitement in the fields of computer vision and robotics. While competition-style events such as the DARPA Grand Challenge have drawn public attention to the difficulties of following roads in ill-structured environments (DARPA, DGC, http://www.grandchallenge.org.,), government-funded research projects such as the DARPA Learning Applied to Ground Robots (LAGR) program and the US Department of Defense DEMO I, II, and III programs (DARPA, LAGR, http://www.darpa.mil/ipto/programs/lagr/vision.htm., ; Shoemaker and Bornstein, 1998) have emphasized navigation in completely off-road environments. Robust autonomous perception and navigation algorithms benefit both the military and the private sector. Automakers will be able to extend the usefulness of current research in driver assistance technologies such as lane-departure detection (Pilutti and Ulsoy, 1998) to more varied environments. Robotic exploration and mapping expeditions will also become possible in a wider variety of environments.

Many current approaches to autonomous robot navigation use 3D sensors such as laser range-finders or stereo vision as the basis for their long range perception systems (Murray and Little, 2000; DeSouza and Kak, 2002; Moorehead et al., 1999; Asensio et al., 1999). These sensors have several disadvantages. While they provide accurate measurements within a certain distance, their perceptive range is limited. This limits the top speed of the robotic platform. Also, active sensors such as lasers tend to cost more, consume more power, and broadcast the position of the robot when it may be advantageous for it to remain undetected (Durrant-Whyte, 2001). A monocular camera can mitigate these problems by capturing scene information out to the horizon. Color or texture information can be used to classify pixels into groups corresponding to traversable terrain or obstacles. Color dissimilarity and color gradient changes have been used to perform this classification (Ulrich and Nourbakhsh, 2000; Lorigo et al., 1997) while the use of texture was pioneered by MIT's Polly tour guide for indoor navigation (Horswill, 1993) and has since progressed to the point where natural terrain classifiers that use texture cues can be used on planetary rovers (Chirkhodaie and Amrani, 2004).

Monocular vision-based techniques have been applied with success to road-following in ill-structured environments (Rasmussen, 2002). Similarly, techniques that use stereo camera information to tag obstacles in the field of view and then use that information paired with a visual appearance-based classifier to determine terrain traversability (Bellutta et al., 2000; Manduchi et al., 2005; Iagnemma and Dubowsky, 2002) have met with success in off-road navigation tests. One of the first uses for self-supervised learning was road-following in unstructured environments (Crisman and Thorpe, 1991) and this technique has also been successfully applied more recently to robot navigation in off-road environments (Sofman et al., 2006).

Navigation methods that utilize only a single monocular camera for terrain classification are subject to limitations. The sensor resolution per terrain area decreases in a monocular image as the distance from the robot increases due to perspective effects. To make intelligent navigation decisions about distant objects the pixels corresponding to these objects must be correctly classified as early as possible. This requires that the robot infer class information using only a small patch of the image plane.

The specularity of an observed object depends on the viewing angle of the observer with respect to the surface normal of the object, which in turn is dependent on the distance between the observer and the object. This effect, combined with the periodic nature of textures, means that the visual appearance of an object at a great distance may be different than its appearance when the robot is close enough to detect it with local sensors. Finally, the automatic gain control operations necessary to mitigate the large dynamic range of outdoor scenes will create differences between the appearance of an object when it occupies a large portion of the robot's field of view and its appearance as a small part of a larger scene.

These difficulties can be overcome with the use of self-supervised learning and optical flow techniques. Self-supervised learning in this context refers to a kernel of supervised learning which operates in an unsupervised manner on a stream of unlabeled data. The exact nature of the supervised kernel varies according to the application. In the case of the adaptive road following, the supervision is the assumption that the vehicle is currently on the roadway. In the case of autonomous navigation, the supervision is the assertion that objects or terrain that trigger near-range sensors such as physical bumpers are hazardous and are to be avoided in the future. The self-supervised algorithm takes this information and monitors the sensor inputs and incoming video stream to label a data set without human assistance.

By storing a history of observed optical flow vectors in the scene, the robot's perception module is able to trace pixels belonging to an object in the current image back to their positions in a given image in the past. In this way, obstacles and terrain types that the robot has interacted with using its short range sensors can be correlated with their appearance when the robot first perceived them at long range. Then the robot learns the visual characteristics of obstacles and easily traversable terrain at distances useful for making early navigation decisions and segmenting a 2D image accurately. This information allows long-range planning and higher traversal speeds. This approach, while applying a pipeline of established techniques, is novel in its use of reverse optical flow for image region correlation and results in improved navigation results.

This paper addresses the implementation of such a self-supervised learning system and its advantages and limitations. The algorithms presented here have been developed by the Stanford Artificial Intelligence Lab as part of work on the DARPA LAGR project. The paper is organized as follows. Section 2 discusses our optical flow approach in detail as well as its application to adaptive road following (Section 2.2) and autonomous off-road navigation (Section 2.3). Section 3 discusses the experimental results from both of these applications and explains limitations of the approach and open issues. Section 4 presents the conclusions from this work.

2. Optical Flow and Self-Supervised Learning

2.1. Reverse Optical Flow

Optical flow techniques have been widely used in mobile robot navigation. Image flow divergence has been used to orient a robot within indoor hallways by estimating time to collision (Coombs et al., 1998) and differences in optical flow magnitude have been used to classify objects moving at different speeds to simplify road navigation in traffic (Giachetti et al., 1998).

We define reverse optical flow as the use of stored optical flow vectors between each video frame and its preceding frame to establish correspondences between pixels corresponding to objects in the current frame and their locations in frames from the past. Our approach uses optical flow information to track features on objects from the time they appear on screen until they interact with the local sensors of the robot. Classification and segmentation algorithms are then trained using the appearance of these features at large distances from the robot. The approach is motivated by the example shown

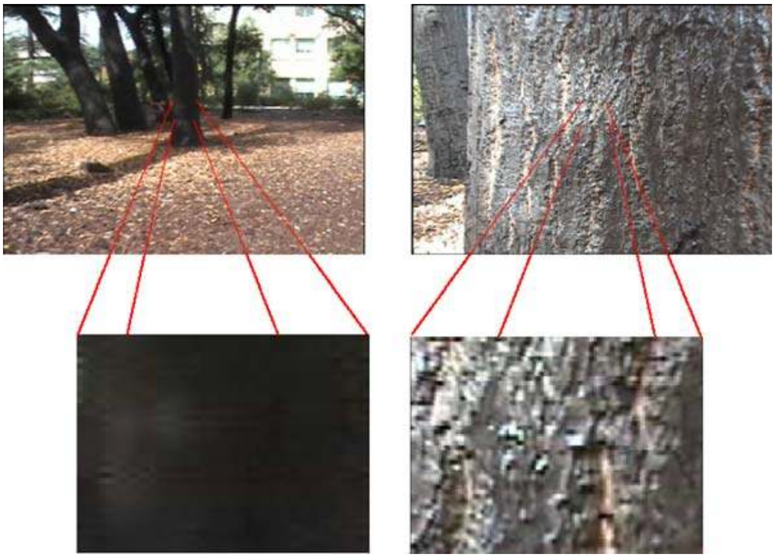


Figure 1. Changes in texture and color appearance with distance.

in Fig. 1. Where traditional monocular image segmentation approaches use the visual characteristics of the tree at short range after a bumper strike shown in the inset on the right side of the figure, our approach uses the characteristics of the tree at a much greater distance shown in the inset on the left side of the figure. As a result, our approach will more accurately classify trees at large distances from the robot, which aides in navigation.

Our approach uses standard optical flow procedures to assemble a history of inter-frame optical flow in real time as the robot navigates to combat the distance-dependent changes in visual appearance and still extract useful terrain classification information from monocular images. This information is then used to trace features on any object in the current frame back to their positions in a previous frame. This optical flow field is populated in the following manner. First, the optical flow between adjacent video frames is calculated. Unique, easily trackable features are identified in the current frame using the Shi-Tomasi algorithm (Shi and Tomasi, 1994). These are unambiguous features that correspond to regions in the image that have significant spatial image gradients in two orthogonal directions. Feature tracking between the current frame and the preceding frame is then performed using a pyramidal implementation of the Lucas-Kanade tracker (Bouguet, 2000). The original images are subsampled and filtered to construct image pyramids. The displacement vectors between the features in the current frame and preceding frame are then calculated by iteratively maximizing a correlation measure over a small window in each level of the pyramid, from the coarsest down to the original. A typical optical flow field captured in the manner is shown overlaid on the original frame from a dataset taken in the Mojave Desert in Fig. 2.

The optical flow field for each consecutive pair of video frames is then subdivided and coarsened by dividing the 720 × 480 image into a 12 × 8 grid and averaging the optical flow vectors in grid cells after removing outliers. The resulting grid, with a mean vector for each cell, is then stored in a ring buffer, a simplified version of which is pictured in Fig. 4(a). A point in the current frame can be traced back to its previous location in any frame in the history buffer or to the frame where it exits the robot's field of view. The diagram shown in Fig. 3 illustrates how this is done for a 200-frame traceback. Zero flow is assumed when an optical flow grid cell is empty. Figure 2 gives an idea of the relative density of the optical flow field. For a representative 7000-frame video sequence, an average of 20% of the grid cells below the horizon are empty. Figure 4(b) shows a set of points in an input video frame denoted by white circles, while Fig. 4(c) shows the location of the points in a frame 200 frames earlier in the sequence calculated with this technique.

The following two subsections discuss in detail the implementation of this technique for two different applications: adaptive road following and autonomous off-road navigation. In adaptive road following the optical flow information is used to assemble a set of templates based on the appearance of the patch of roadway currently in front of the robot. These templates are used in matching the road at different distances from the robot in the current frame. In off-road navigation, the optical flow is used to correlate the appearance of objects and terrain in the past with their traversability as determined by the local sensors of the robot.

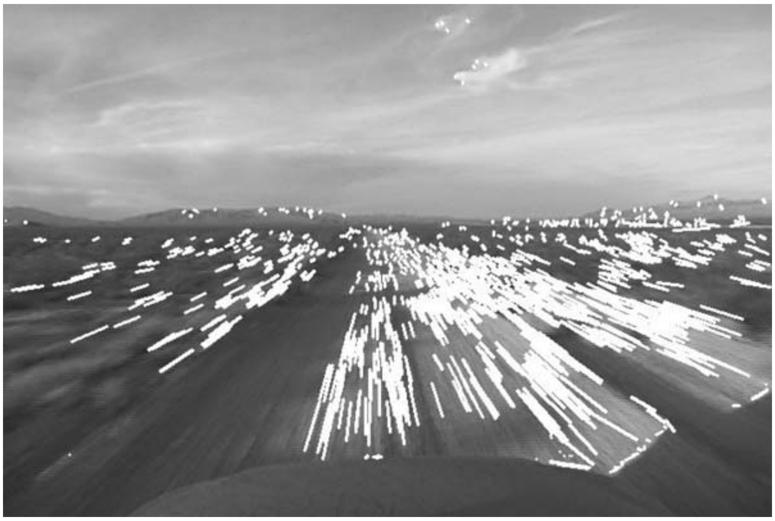


Figure 2. White lines represent the optical flow field in a typical desert driving scene (the length of flow vectors has been scaled by a factor of 2 for clarity).

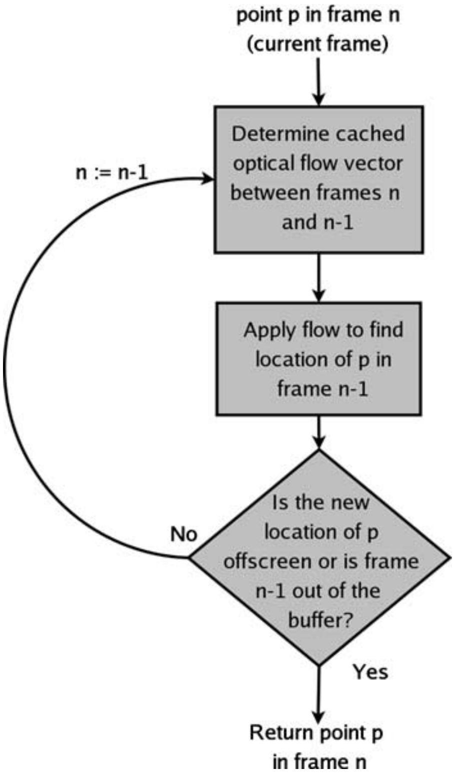

point p in frame n
(current frame)
Determine cached optical flow vector between frames n and n-1
n := n-1
Apply flow to find location of p in frame n-1
Is the new location of p offscreen or is frame n-1 out of the buffer?
No
Yes
Return point p in frame n

Figure 3. Operations for tracing the location of a feature backwards in time.

2.2. *Adaptive Road Following*

The approach discussed here was described in an earlier form in Lieb et al. (2005) and is composed of the following steps: reverse optical flow, horizontal 1D template matching, and dynamic programming. Similar template matching techniques have been used to determine lateral offset in previous work for a lane departure warning system (Pomerleau, 1995). The algorithmic flow is depicted in Fig. 5.

This approach is designed to deal with ill-structured desert roads where traditional highway road following cues such as lane markings and sharp image gradients associated with the shoulder of the road are absent. Regions off the roadway may be similar enough in texture and color to the roadway itself that traditional segmentation techniques are unable to differentiate between them. This approach requires that the vehicle is currently traveling on the road and then tracks regions similar to the area directly in front of the vehicle. This region, typically twenty pixels high, which we call the *definition region* is shown in Fig. 6(a). If the vehicle is currently on the road, the pixels directly in front of the vehicle in the image are representative of roadway. Since the appearance of the definition region at different times in the past is important for the template matching procedure described below, the requirement for the correct functioning of the algorithm is that the vehicle has been traveling and recording video for at least 7 seconds (not necessarily on the road) and is currently on the road.

Template Matching. To determine the location of the road at different heights in the current image, a set of horizontal templates is collected that reflects the best guess about the appearance of the roadway at the current time. These templates are formed by taking the location of the definition region in the current frame, and then using optical flow to find the location of this region in images increasingly far in the past. This approach is effective

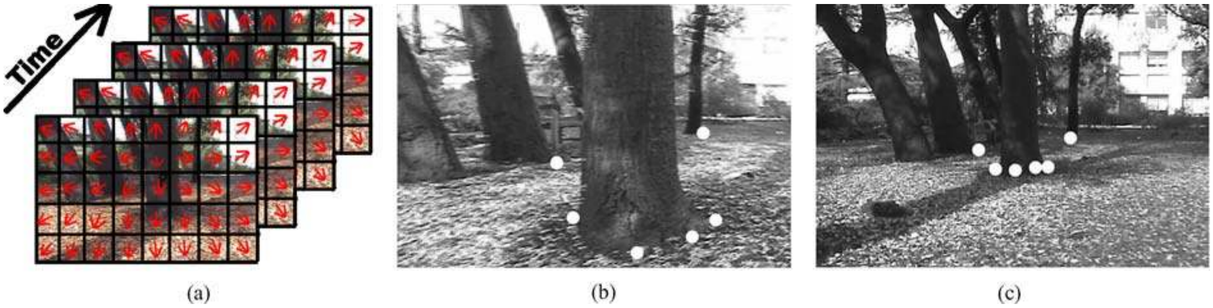


Figure 4. (a) Optical flow compressed and stored for a number of frames in the past (b) Points selected in initial video frame (c) Origin of points 200 frames in the past.

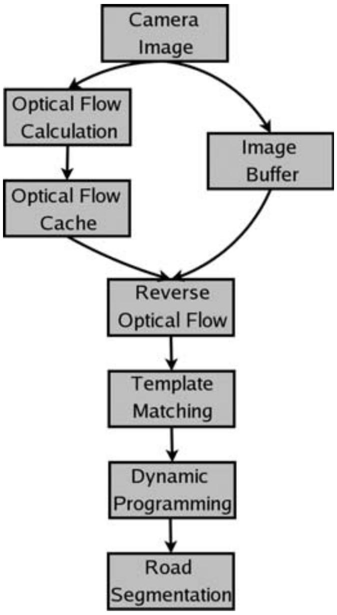


Figure 5. Adaptive road following algorithm.

because of the difference in appearance of objects as a function of their distance from the robot discussed earlier in this paper. Figures 6(b)–(d) show the locations of the templates that result at different distances from the robot as a result of using optical flow to find the location of the definition region in past frames.

Using the set of templates collected for the current definition region, the most likely location of the road at various heights in the image can be estimated by using a horizontal template matching algorithm. The vertical search height for a given template in the current image is determined by scaling the the vertical location of the template in the image in the past from which it was taken. The scaling factor is the difference between the height of the horizon in the current frame and in the original frame. The location of the horizon in each frame is determined using a horizon detector based on the work of Ettinger et al. (2002). A 2-D search space parameterized by the height and angle of the horizon in the image is searched using a multi-resolution approach to minimize a criterion. This criterion is the sum of the variances in the blue channel of the pixels labeled as sky and those labeled as ground. An example of the detected horizon is shown in Fig. 7.

This scaling was necessary to mitigate the effect of changes in the pitch of the vehicle on the performance of the algorithm. It is worth noting that while this approach was developed using a video stream without vehicle telemetry, the addition of accurate pitch information would obviate the need for a horizon detector.

Both the templates and the search space are horizontal slices of the image and templates taken from curved roads therefore appear similar to those taken from straight roads. However, templates taken from curved portions of roadway will be artificially wide. The same effect occurs if the vehicle is undergoing a moderate amount of roll.

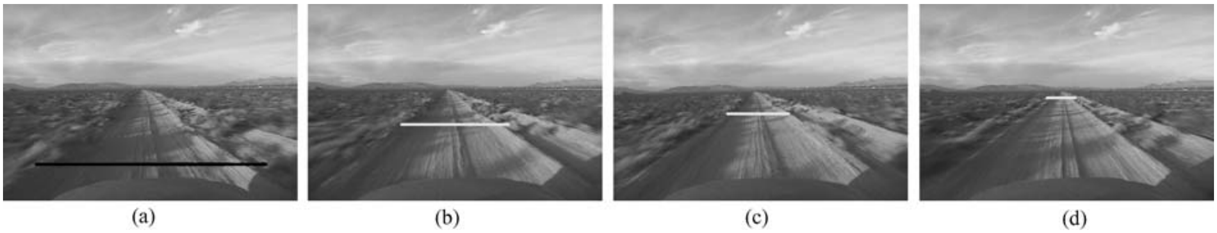


Figure 6. (a) Dark line shows the definition region used in our algorithm. (b)–(d) White lines show the locations in previous frames to which optical flow has traced the definition region.

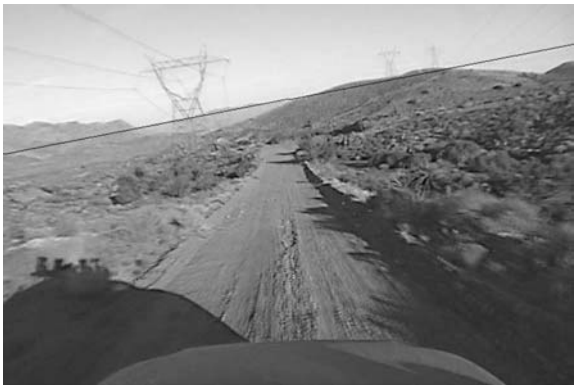


Figure 7. Output of horizon detector is the dark line.

The template matching measure combined with the dynamic programming approach described below mitigates these problems.

The template matching method used is a normalized sum of squared differences (SSD). This computes the strength of the template match along each horizontal search line. Because the search space for each template is only a single horizontal line and the height of the template is typically 20 pixels, this matching measure can be computed very quickly. Figure 8(b) shows the output of the matching measure for a set of 10 template search lines in the image shown in Fig. 8(a). The responses in the SSD image have been widened vertically for display purposes, though they appear at the height in the image at which each individual template was checked. White regions indicate stronger matches, while dark regions indicate weaker matches. Strong responses can also be seen in the upper right portions of the scene where the lack of vegetation and shadows combine to make template matches to the right of the roadway attractive.

Dynamic Programming. Figure 9 depicts the location of the maximum SSD response along each horizontal search line with dark circles. Sometimes the location of maximum response does not lie on the roadway due to similarities in visual characteristics between the roadway and areas off the road, and because of illumination differences between the search areas and the templates.

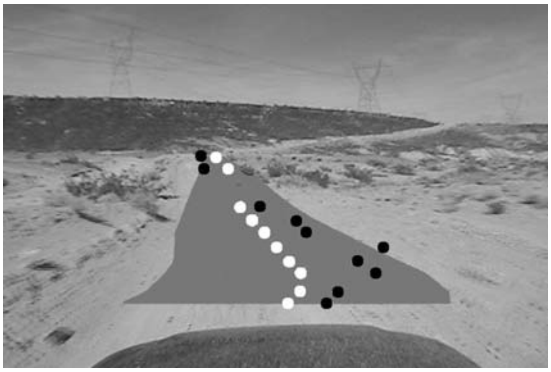
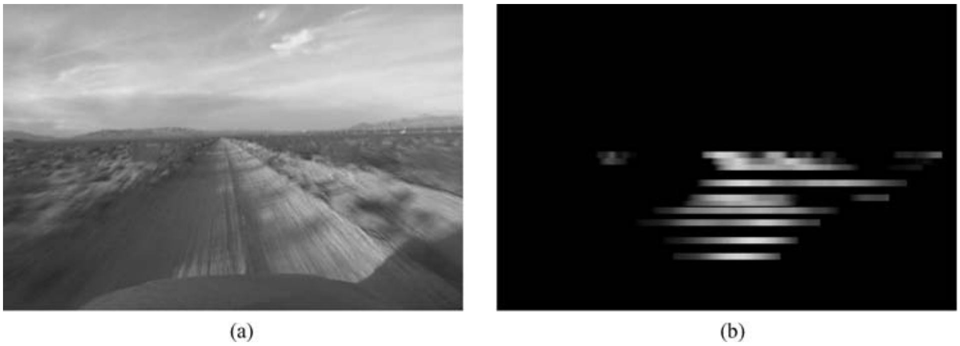


Figure 9. Dark circles represent locations of maximum SSD response along each horizontal search line. Light circles are the output of the dynamic programming routine. The gray region is the final output of the algorithm and is calculated using the dynamic programming output. The width of this region is linearly interpolated from the horizontal template widths.

The need to find the globally optimal set of estimated road positions while satisfying the constraint that some configurations are physically impossible for actual roads suggests the use of dynamic programming. Dynamic programming has already been used for road detection, both aerial (Dal Poz and do Vale, 2003) and ground-based (Redmill et al., 2001; Kim et al., 2002; Kang and Jung, 2003). The purpose of dynamic programming in our approach is to calculate the estimated position of the road at each search line in the image so that when the positions are taken together they minimize some global cost function. The cost function chosen in this case is the SSD response for each horizontal search line, summed over all search lines. The search lines are processed from the topmost downward, with the cost at each horizontal position computed as the SSD cost at that location plus the minimum cost within a window around the current horizontal position in the search line above. The horizontal position of this minimum cost is also stored as a link.



(a) (b)

Figure 8. (a) Input video frame (b) Visualization of SSD matching response for 10 horizontal templates for this frame.

Once the bottommost search line has been processed in this way, the globally optimal solution is found by following the path of stored links, each of which point the minimum cost position in the search line above. The path traversed represents the center of the estimated position of the road.

Given the assumption that the vehicle is currently on the roadway, the finite window restriction serves to enforce a constraint on the maximum expected relative angle between the heading of the vehicle and the road and reduces the computation time of the optimization. To compute a reasonable window size for a given road heading, the equations below can be used.

$$\begin{pmatrix} P_x \\ P_y \\ 1 \end{pmatrix} = I \cdot E \cdot \begin{pmatrix} 1 \\ \frac{\Delta Y}{\Delta X} \\ 0 \\ 1 \end{pmatrix} \tag{1}$$

$$W \geq \frac{(\min\Delta_H) \cdot P_x}{P_y} \tag{2}$$

In Eq. (1), P_x and P_y represent the image coordinates of a point along the road of relative slope $\frac{\Delta Y}{\Delta X}$ viewed through the intrinsic transformation I and extrinsic transformation E. In Eq. (2), the dynamic programming window W must be greater than or equal to the minimum separation between template match regions ($\min\Delta_H$) divided by the road slope in the image $\frac{P_y}{P_x}$.

The output of the dynamic programming module is depicted by the light circles in Fig. 9. The white region in Fig. 9 indicates road segmentation. The location of this region was determined by the dynamic programming output while the width of the segmented region was linearly interpolated from the widths of the horizontal templates.

2.3. *Off-Road Navigation*

The algorithm described here is illustrated in Fig. 10 and consists of four major parts: initial pixel clustering, image segmentation, projecting image space information into an occupancy grid assuming a flat ground plane and vertical obstacles, and handling training events. This approach was designed to provide long-range terrain classification information for a mobile robot. By placing that information into an occupancy grid, other sensors such as stereo vision and infrared sensors can be used to augment this grid. A D* global path planning algorithm (Stentz, 1994) is run on this map to perform rapid navigation.

The initial pixel clustering step takes an RGB input image and runs K-means on it with K set to 16 clusters. A set of sample points are taken at random from each of these 16 clusters and used to compute a multi-variate Gaussian in RGB color space for each initial cluster. Each of these Gaussians is represented by a mean vector and a covariance matrix.

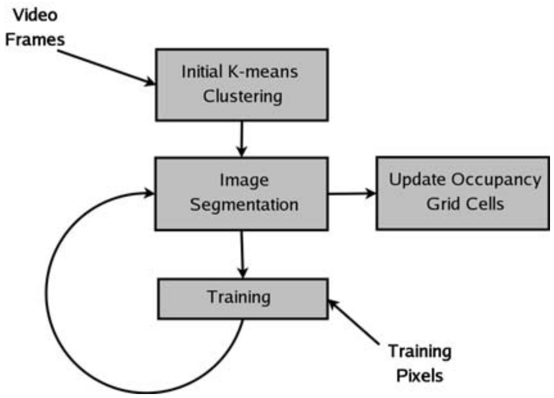


Figure 10. Off-road navigation algorithm.

After the initial input image is processed, each subsequent image is segmented based on these Gaussians according to a Maximum Likelihood (ML) strategy. A class label l is chosen for each pixel in order to maximize the probability $f(c \mid l)$. The class label l is one of $\{good, bad, lethal\}$. Three classes were chosen instead of two to allow the global D* planner to discriminate between terrain that was passable but potentially hazardous and lethal obstacles through which the robot could not navigate. The choice of near-range sensor suite used for training did not provide enough information to warrant the use of more than three classes. The conditional pdf for a color c, given that it belongs to class l is

$$f(c \mid l) = \sum_{j=1}^{M(l)} \frac{1}{M(l)} \cdot G(c; \mu_{l,j}, \Sigma_{l,j}) \tag{3}$$

where the Gs are the Gaussian pdfs with means μ and covariance matrix Σ, and $M(l)$ is the number of classes. This approach is similar to the one used by Manduchi et al. (2005), and is interesting in this context as a good approach which can be improved with the use of optical flow.

Note that the class *unknown* corresponds to the case when a pixel's color falls more than a maximum distance from any of the Gaussians in RGB space. We use a Mahalanobis metric for this distance.

The critical portion of this approach is the training of the model. When the robot gathers data from the environment through its local sensors, information about which pixels in an image correspond to obstacles or different types of terrain is used to train the Mixture of Gaussians model. The addition of the optical flow method described earlier makes it possible to do this training based on the appearance of obstacles and terrain at

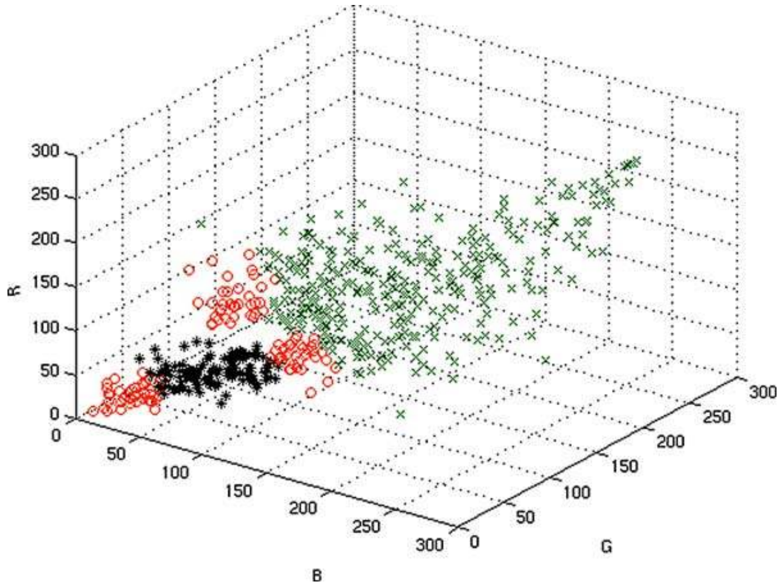


Figure 11. Points corresponding to the good class are depicted by x's, bad class by circles, and lethal class by stars.

greater distances from the robot. This improves the performance of classification of image regions corresponding to those obstacles and terrain types at distance and enables higher traversal speeds. For the self-supervised learning, the supervisory inputs are the physical bumpers and the infrared range sensors. Blame assignment, while not perfect, is made easier by the fact that cameras on the platform (LAGR robot, see Fig. 21) point downward and inward. The pixels which correspond to the top of a traffic cone-sized obstacle that triggered either the right or left physical bumper switch have been determined via calibration. In the same manner, when the infrared range sensors register an obstacle at a certain range, the pixel correspondences have been determined ahead of time.

When a local sensor such as a physical bumper or an infrared range sensor registers an obstacle the optical flow procedure is called. The pixels that correspond to where that object lies in the current image are traced back to the point where the they first entered the field of view of the robot (or the location in the oldest frame for which information exists in the optical flow history buffer if a full traceback is possible). Pixels corresponding to the object from this frame in the past are then used to train the Mixture of Gaussians classifier. These pixels are incorporated in the Gaussian mixture component whose mean and covariance yield the minimum distance in the Mahalanobis sense. If the Mahalanobis distance to the nearest mean is greater than 0.9 then a new mixture component is created to capture any future examples of this obstacle. Since these Gaussian mixture components are initially trained on K-Means output, their covariances model the underlying variability of the color of the object being recognized. This means that the classification is insensitive to the specific value of the threshold Mahalanobis distance. A value of 0.9 empirically allows various types of trees to be classified together, while still permitting new types of objects to be recognized as being different. A point cloud representation of what the different classes look like for data gathered during a test run is shown in Fig. 11. The image points used to train the Gaussians are shown in RGB space classed into their good, bad, and lethal classes.

Finally, the terrain classification information present in the current segmented video frame must be processed in such a way that the robot can make useful navigation decisions. In this particular pipeline we use look-up tables with information about the point on an assumed flat ground plane which corresponds to every pixel in the image. These tables were constructed using the intrinsic and extrinsic parameters of the robot's cameras determined from a camera calibration. They allow a ray to be calculated for each pixel in the image, and the location at which that ray intersects the ground plane is then a simple trigonometric calculation.

The information about the location on the ground plane to which each pixel corresponds is used to cast votes for what class each grid cell in the occupancy grid will be assigned. To locate vertical obstacles in space under the flat ground plane assumption, we scan vertically from the bottom of the image to find the first obstacle classified pixels. Since we are assuming that all good terrain lies flat on the ground plane, the first pixels which do not conform to the good class when processed in this manner locate the intersection of the obstacle with the ground. All additional pixels above this intersection point will

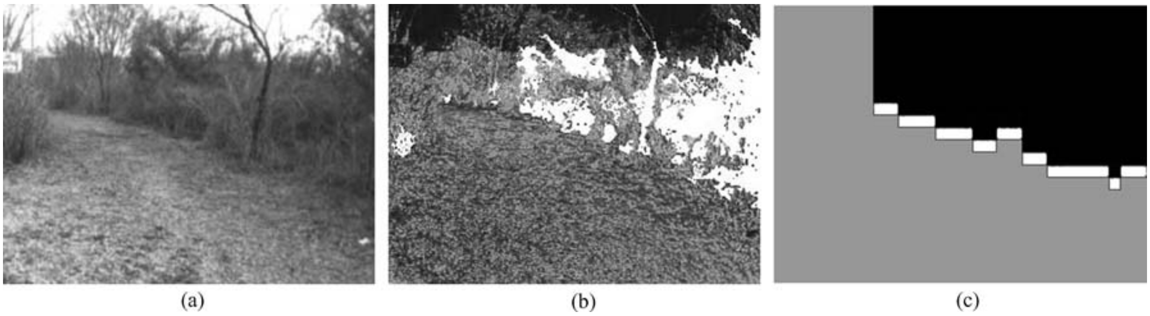


Figure 12. (a) Input frame (b) Raw segmentation output (c) Output of 'bottom finder'.

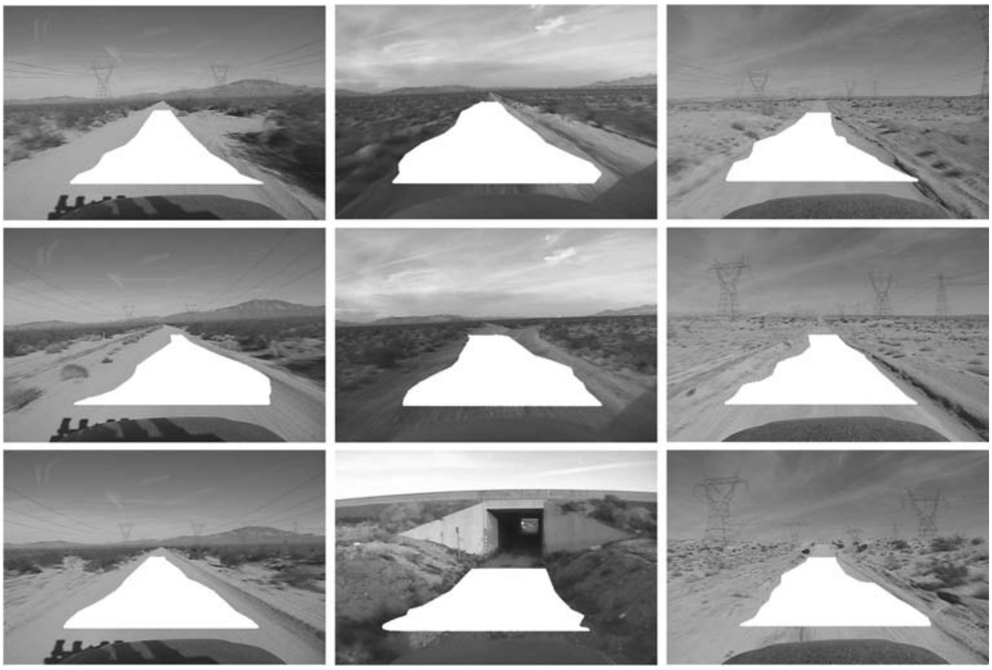


Figure 13. Single frame algorithm output for three Mojave Desert data sets. Each column contains results from one of the three video sequences.

be improperly projected with the ground plane table, so instead their collective influence is represented by voting all of their influence at this intersection. For an example of this process see Fig. 12.

3. Results

These are the results of applying self-supervised learning and the reverse optical flow technique to adaptive road following and mobile robot navigation.

3.1. Adaptive Road Following

The approach works well in environments where the road is ill-structured and makes no prior assumptions about the visual appearance of the roadway being followed. Single frame results taken from three different 720 × 480 pixel video sequences shot in the Mojave desert are shown in Fig. 13. Each column of the figure contains images from a different test video sequence. The first video sequence contains footage from a straight dirt road where scrub brush lines the sides of the road and was taken in direct sun. The second sequence comes from a straight road with less vegetation along the sides. It was taken late in the afternoon, with long shadows stretching across the road. The third sequence is from a trip through terrain with changes in elevation and gravel coloration. Between the three sequences there is more than 12 minutes of video.

The details of our implementation of the algorithm discussed in Section 2.1 are as follows. The road position estimates are the result of 1D template matching of a set of 10 horizontal templates found using the optical flow procedure. These templates are samples at different points in time of the visual characteristics of the roadway area currently in front of the robot. These templates were taken from the past ranging from 1 frame to 200 frames prior to the current frame. The spacing of the temporal samples was chosen to provide an even vertical spacing in the image plane. The templates were 20 pixels high, and the definition region and templates were refreshed every 10 frames in order to adapt to gradual changes in the appearance of the roadway. 3000 feature correspondences were used to calculate the optical flow fields, and the mean flow vectors were stored in a grid of 96 square cells covering the entire image plane. To quantify the overall performance of the algorithm in this domain, the results of running it on the three 7200-frame data sets described above were evaluated using the two performance metrics described below. The data sets were taken prior to the development of this algorithm, and do not reflect the algorithm exerting any control over the path of the vehicle.

For comparison purposes, each of these data sets has also been run through an image segmentation program which uses Markov Random Fields (MRF) with the Metropolis algorithm for classification. The software was written by Berthod et al. (1996), Kato et al. (1992), and Kato (1994). This software is publicly available at Zoltan Kato's website, but it has been modified to take fixed training regions and upgraded to use full color information and covariance matrices. The training regions have been permanently set for every test frame to be a rectangular region in front of the vehicle which corresponds to the definition region used for our algorithm and a square region to the left of where the road should appear in the image. These regions were chosen because they provide the MRF with a good example of what the road looks like versus the rest of the scene. The Metropolis algorithm works by iteratively refining a labeling assignment over pixels according to a Single and Double potential to minimize an energy function. The Double potential component of the energy function is minimized when neighboring pixels have the same labeling. The Single potential is minimized when a pixel is labeled as belonging to the most similar class. This procedure produces an image classification which will be compared with the results generated by the optical flow technique using the following two metrics.

Pixel Coverage Metric. The first metric compares pixel overlap between the algorithm output and ground truth images in which the road has been segmented by a human operator, as shown in Fig. 14. The number of pixels in the frame that have been incorrectly labeled as roadway is subtracted from the number of correctly labeled roadway pixels. This number is then divided by the total number of pixels labeled as road by the human operator for that frame. Using the metric proposed here, a score of 1.0 would correspond to correctly identifying all the road pixels as lying in the roadway, while not labeling any pixels outside the roadway as road pixels. A score of 0.0 would occur when the number of actual road pixels labeled as roadway is equal to the number of non-roadway pixels incorrectly identified as being in the road. If more pixels were incorrectly labeled as roadway than actual road pixels correctly identified, negative scores would result. This measure is computed once per second and averaged over the entire video sequence. While this pixel coverage metric is easily visualized and simple to compute, it must be recognized that, due to perspective effects, it is strongly weighted towards regions close to the vehicle.

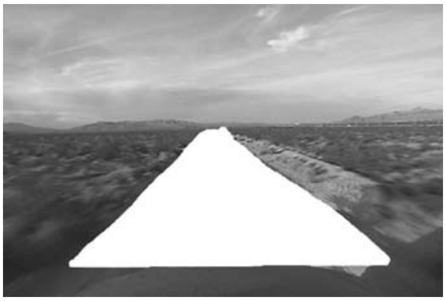


Figure 14. Typical human-labeled ground-truth image.

Line Coverage Metric. The second metric mitigates the distance-related bias of the first metric by comparing pixel overlap separately along a set of horizontal lines in the images. Five evenly spaced horizontal lines are chosen ranging in vertical position between the road vanishing point and the vehicle hood in the ground-truth image. Success scores are calculated just as in the first metric, except they are reported individually for each of the five lines. The metric returns five sets of success scores computed once per second and averaged over the entire video sequence.

Figure 15 shows the performance of the algorithm proposed in this paper on the three different video sequences,

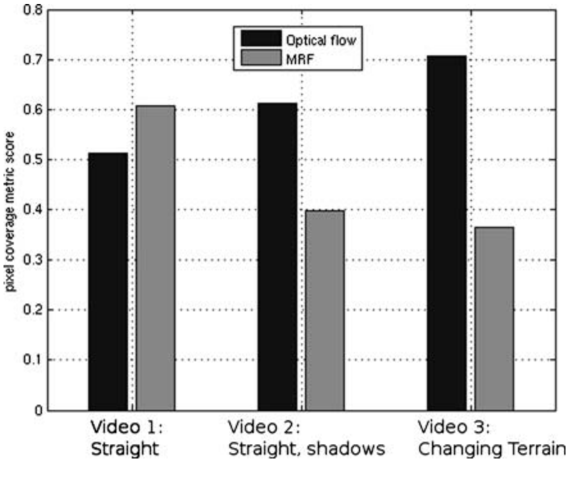


Figure 15. Pixel coverage results on the three test video sequences.

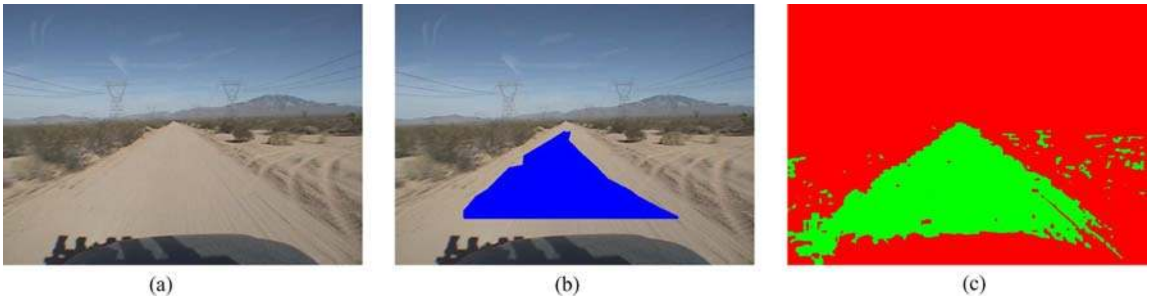


Figure 16. (a) Input frame (b) Optical flow technique output (c) MRF classifier output.

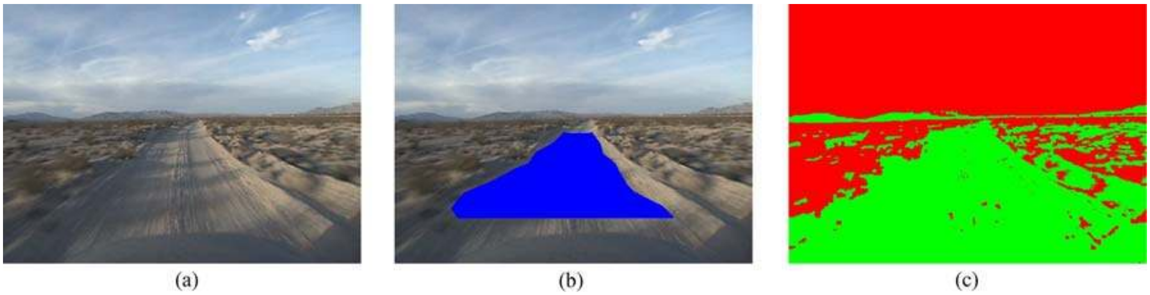


Figure 17. (a) Input frame (b) Optical flow technique output (c) MRF classifier output.

evaluated using the pixel coverage metric. These results are compared against the output from the Markov Random Field segmentation program. The MRF classifier assumes that the region directly in front of the vehicle is on the roadway but it does not use optical flow methods. It is worth noting again here this approach performs road following only when the vehicle is currently on the roadway. It is important to keep in mind that the scores for both our approach and the MRF-based classification reflect only the ability to follow the road, not to find it.

The MRF classifier outperforms the optical flow technique slightly on the first video sequence. In this video, the areas off the roadway are covered with sage brush which differs from the roadway in both color and texture. A representative frame of output from both the MRF classifier and the optical flow technique, and the corresponding input frame are shown in Fig. 16.

In the second video sequence, the long shadows and sparser vegetation combine to make areas off the road visually more similar to the roadway. Consequently, the performance of the MRF classifier was adversely affected while the optical flow technique was unaffected. Figure 17 shows the test output on a representative frame from this video. An interesting limitation of the approach is its vulnerability to intermittent shadows. If a template taken from the past comes from a region in shadow, and is being matched against a region in the current image which is not in shadow, the resulting SSD response will be biased towards darker regions. This effect is shown in the SSD response and corresponding input frame in Fig. 18. The dynamic programming step alleviates the severity of this effect. In the third video sequence, the almost complete absence of vegetation makes the areas off the road visually very similar to the roadway. The performance of the MRF classifier suffers as a result, while the strong vertical texture components in the roadway improve the results of the template matching done during optical flow traceback. This can be seen in the test output in Fig. 19. Figure 20shows the performance of the algorithm on the same three data sets, now evaluated using the line coverage metric. Scores are graphed for a set of five evaluation lines increasingly distant from the vehicle. The performance of the algorithm generally declines as the distance from the vehicle increases. The proposed algorithm achieves very low false positive rates by making no assumptions about the general appearance of the road and classifying regions that adhere to its learned roadway information.

Videos of the three 7200-frame test sets, showing the results of tracking with the proposed algorithm as well as with the MRF classifier are available at *http://cs.stanford.edu/group/lagr/road_following/*. The algorithm runs at 3 Hz on a 3.2 GHz PC at 720 × 480 pixel resolution.

3.2. *Off-Road Navigation*

This section discusses the results from applying the technique discussed in Section 2.2. This work was done as

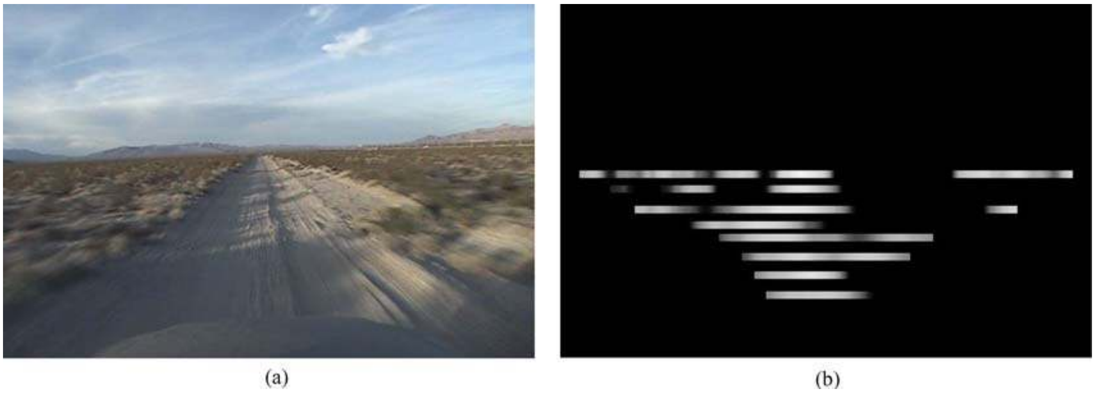


Figure 18. (a) Input frame (b) SSD response.

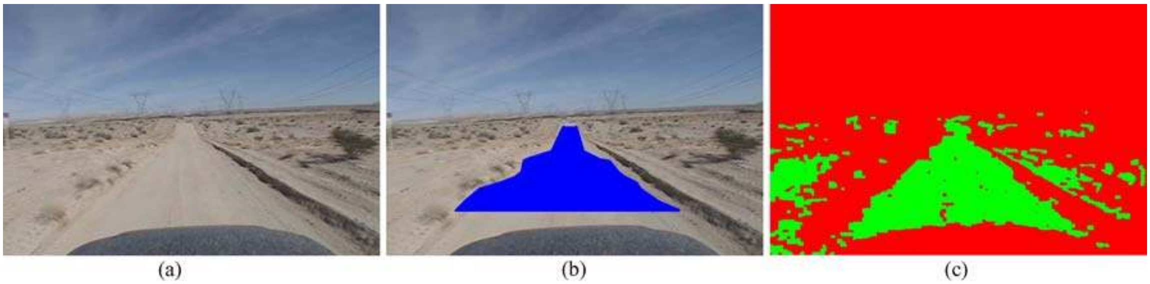


Figure 19. (a) Input frame (b) Optical flow technique output (c) MRF classifier output.

part of the Stanford Artificial Intelligence Lab work on the DARPA LAGR program. The robot platform for the program is shown in Fig. 21. The goal of this program is autonomous off-road navigation between two GPS waypoints while avoiding obstacles. This is to be done using computer vision as the only long-range sensor. Since the visual obstacles at the remote test sites where the races are run may vary drastically from obstacles used to test the vehicle on our test courses, self-supervised learning is an attractive approach.

The choice of metric to quantitatively test the monocular 2D image segmentation portion of our architecture is an important one. A frame-by-frame pixel data metric is desirable. The metric must also compare algorithm-labeled images to operator-labeled images. There is a great research benefit in using standardized benchmarks for natural object recognition algorithms such as MINERVA (Singh and Sharma, 2001). However, since our approach requires the entire video stream up until an object interacts with the local sensors a static database of benchmark images is not appropriate for evaluating this work. The traditional percent incorrectly classified pixel error metric fails to capture the importance of correctly classifying objects at a greater distance from the robot that subtend fewer pixels and the pixel distance error metric (Yasnoff et al., 1977) is not well suited to natural scenes with large depth of field.

Scene Segmentation. When evaluating our algorithm, we found the following two-part metric useful. If the image segmentation section of our algorithm is set to return a binary cost map (hazard or non-hazard) then the false negatives are reflected in a percentage of discrete obstacles correctly matched. The number of pixels correctly classified by the algorithm as obstacle lying in a human-labeled obstacle are divided by the total number of human-labeled pixels for that obstacle. This percentage is then averaged for all the obstacles in the scene and corrects for the smaller total number of pixels present in obstacles farther from the robot which we consider equally important for successful long-range, high-speed navigation. The false positives are reflected in a percentage of incorrectly classified square meters of ground within 25 meters of the robot in a given image divided by the total number of square meters which the algorithm was given the task of segmenting. The number of square meters corresponding to a given pixel in the image was calculated using the ground-plane tables discussed earlier in this paper. Incorrect classifications of terrain at large distances from the robot have a large effect on this score.

This process is illustrated in Fig. 22. Figure 22(a) shows a sample frame from the video sequence. Figure 22(b) shows the hand-labeled obstacles, Fig. 22(c) shows the results of segmentation done without reverse optical flow, and Fig. 22(d) shows the results of segmentation done with reverse optical flow.

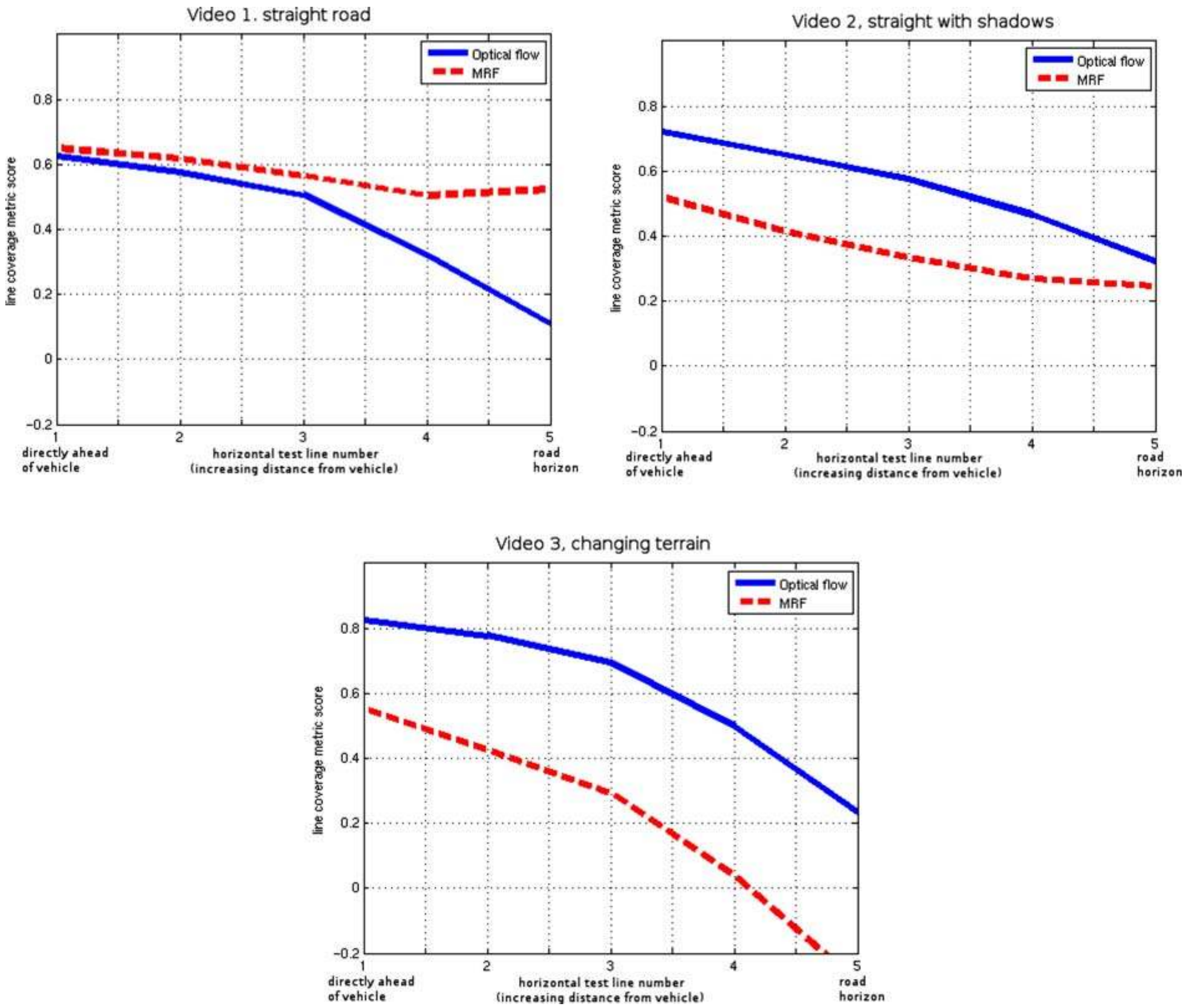


Figure 20. Pixel coverage results are shown at different distances from the front of the vehicle towards the horizon for the three video sequences.

The results shown here come from a set of three test runs in a forest in flat lighting with no sharp shadow where the only obstacles were trees. Two instances of our segmentation module were running at the same time. One was using optical flow techniques to determine the characteristics of obstacles the robot interacted with, while the other was not. During each run the robot was guided by an operator at normal traversal speeds towards three or four of the trees in the scene for a local-sensor interaction. After the initial interactions, the robot was driven through the forest without any additional obstacle interactions at close range. During the runs, video frames along with their segmented counterparts were logged on the robot at a rate of 10 Hz. After the run these logs were parsed, and every 10th frame was analyzed using the metrics described above. The three runs, taken together, totaled over 3,000 frames, and 304 of these frames were used to compute the results. Without optical flow, the percentage coverage of each obstacle was 81.65% while the percentage of area incorrectly classified as belonging to an obstacle was 53.75%. With optical flow, the percentage coverage of each obstacle was 88.98% and the percentage of area incorrectly classified was 36.53%. The data show that the percentage of each obstacle in the scene

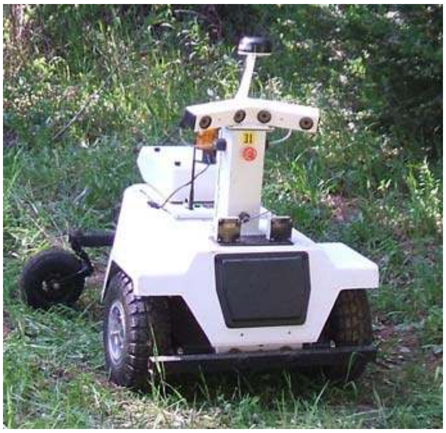


Figure 21. The LAGR robot platform.

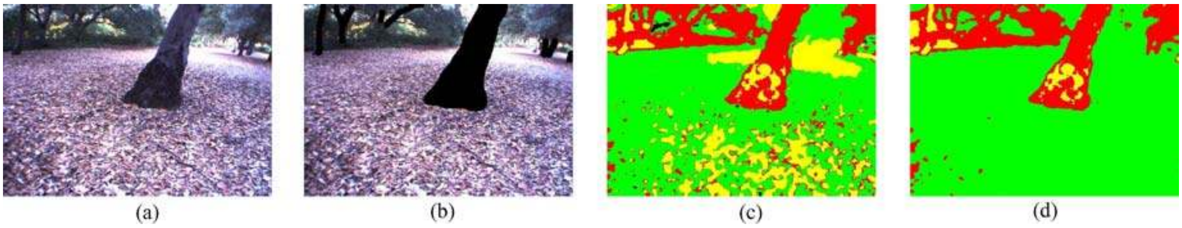


Figure 22. (a) Video frame (b) Hand-labeled obstacle image (c) Segmentation without optical flow (d) Segmentation with optical flow.

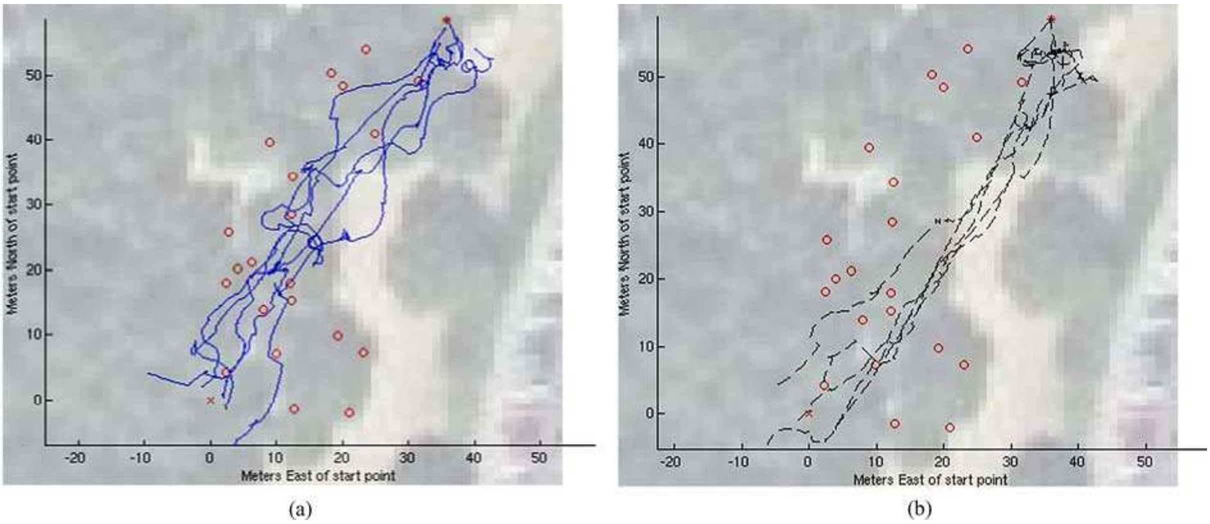

Meters North of start point
50
40
30
20
10
0
-20
-10
0
10
20
30
40
50
Meters East of start point
(a)
Meters North of start point
50
40
30
20
10
0
-20
-10
0
10
20
30
40
50
Meters East of start point
(b)

Figure 23. (a) Paths taken using data collected without optical flow (b) Paths taken using data collected with optical flow (map overlays courtesy of Google Maps).

correctly identified by the classifier is higher with the optical flow traceback than without. More significant is the decrease in the percentage of area incorrectly classified as belonging to an obstacle class.

Autonomous Navigation. To test the applicability of the scene segmentation results discussed above, we evaluated the autonomous performance of the robot with the Gaussian databases collected from running with and without optical flow traceback during the last data collection run discussed above. Running all of our planning and sensing modules except stereo vision, we tested the robot on a 70 m course running through a different part of the forest containing trees with which the robot had not previously interacted. Learning was turned off during these runs. A total of 10 runs were made, in each of which the robot achieved the goal. Half of the runs were made using the Gaussian database collected without optical flow traceback while the other half were made using the Gaussian database collected with optical flow traceback. The starting position of the robot was randomly varied within a radius of 5 meters for each pair of traceback/no traceback runs. The paths taken by the robot during these ten tests, as well as the locations of the trees on the test course are shown in Fig. 23. The number of interactions with trees which caused the robot to stop and back up (physical and IR bumper hits) was noted for each run, along with the total run time. The average time and the number of trees the robot interacted with at close range during runs using the data collected with the optical flow approach were lower than the corresponding numbers for runs using the data collected without optical flow. The results, with 95% confidence ellipses, are shown in Fig. 24.

The 2D monocular image terrain classifier works well normally, and performs even better when optical flow techniques are used to correlate the appearance of objects close to the robot with those at greater distances. Videos illustrating the use of optical flow techniques on the robot to trace features corresponding to specific objects back in time as well as the performance of the segmentation algorithm can be found at *http://cs.stanford.edu/group/lagr/IJCV_IJRR/*.

4. Conclusions

In this article we have presented an algorithm that uses optical flow techniques to integrate long-range data from monocular camera images with short-range sensor

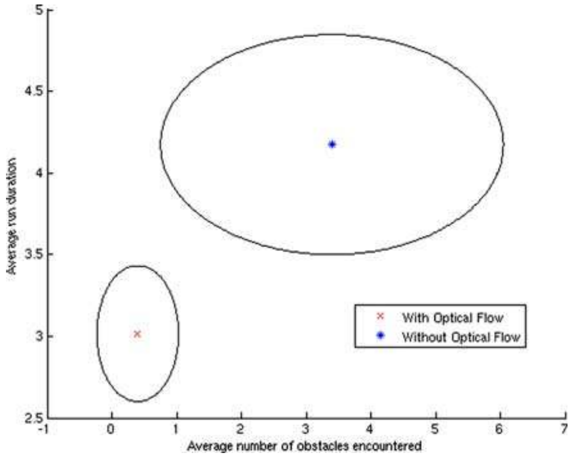


Figure 24. Autonomous navigation results with 95% confidence ellipses.

information to improve the quality and utility of terrain classification. In our approach, the optical flow field between successive input frames is coarsened and saved in a ring-buffer to allow the location of features on an object in the current frame to be traced back to their locations in a previous frame. This allows information about the traversability and danger of objects within range of the robots physical and IR bumpers to be used to classify similar objects reliably at large distances from the robot. This, in turn, allows more efficient navigation and higher speeds.

There are domains in which the optical flow techniques upon which this approach is based do poorly. Unusually smooth image regions or regions of images which are completely saturated or desaturated will not contain trackable features. This causes the optical flow history grid cells in those areas to be empty or filled with noise, which will cause errors when objects are traced back to their original locations. Objects which are in the field of view of the robot, and then leave the image, but return (such as objects seen while the robot is executing an S-turn) are currently only tracked back to their locations in the image where they most recently entered the field of view of the robot. Handling these cases using template matching or particle filter tracking might allow a more complete traceback. Finally, changing illuminant conditions can result in unacceptable rates of misclassification.

This algorithm has been applied to two currently compelling problems in the intersection of the fields of mobile robotics and computer vision: autonomous offroad navigation and road-following in ill-structured environments. In road-following, the approach was shown to be effective when MRF based or texture-based approaches may break down. In autonomous off-road navigation, the approach resulted in an increase in the percentage of each obstacle correctly segmented in the input image and a decrease in the percentage of traversable terrain incorrectly classified. Finally, using this technique in a situation where monocular vision was the only long-range sensor on a robot platform proved to be advantageous.

Extensions of this work will include the use of more sophisticated machine learning and computer vision techniques for doing feature selection and scene segmentation once objects have been traced back to their origins in earlier frames. This approach is also applicable to tracking, recognition, and surveillance problems where correlating the appearance of objects or people at close range with the corresponding visual characteristics at long range increases the accuracy of classification.

Acknowledgments

This research has been financially supported through the DARPA LAGR program. The views and conclusions contained in this document are those of the authors, and should not be interpreted as necessarily representing policies or endorsements of the US Government or any of the sponsoring agencies. The authors would like to acknowledge the Stanford Racing Team for providing the test videos used in the preparation of this paper and would also like to thank Itai Katz, Philip Engstrom, and Paul Fontes. The fourth author expresses his gratitude to Sandia National Laboratories. Without Sandia's generous support of his graduate studies, his participation in this work would not have been possible.

References

1. Asensio, J.R., Montiel, J.M.M., and Montano, L. 1999. Goal directed reactive robot navigation with relocation using laser and vision. In *IEEE Proc. of International Conference on Robotics and Automation*, 4:2905–2910.
2. Bellutta, P., Manduchi, R., Matthies, L., and Rankin, A. 2000. Terrain perception for DEMO III. In *Proc. Intelligent Vehicles Conf.*, Dearborn, MI.
3. Berthod, M., Kato, Z., Yu, S., and Zerubia, J. 1996. Bayesian image classification using markov random fields. *Image and Vision Computing*, (14):285–295.
4. Bouguet, J.Y. 2000. Pyramidal Implementation of the Lucas Kanade Feature Tracker, Intel Corporation, Microprocessor Research Labs, http://www.intel.com/research/mrl/research/opencv/
5. Chirkhodaie, A. and Amrani, R. 2004. Visual terrain mapping for traversable path planning of mobile robots. *Proceedings of SPIE*, 5608:118–127.
6. Crisman, J. and Thorpe, C. 1991. UNSCARF, A color vision system for the detection of unstructured roads. In *Proceedings of International Conference on Robotics and Automation*, Sacramento, CA.
7. Coombs, D., Herman, M., Hong, T., and Nashman, M. 1998. Real-time obstacle avoidance using central flow divergence, and peripheral flow. *IEEE Trans. on Robotics and Automation*, 14(1):49–59.
8. Dal Poz, A.P., and do Vale, G.M. 2003. Dynamic programming approach for semi-automated road extraction from medium- and high-resolution images. *ISPRS Archives*, XXXIV(part 3/W8).

9. Defense Advanced Research Projects Agency (DARPA), Darpa Grand Challenge (DGC). Online source: http://www.grandchallenge.org.
10. Defense Advanced Research Projects Agency (DARPA), Learning Applied to Ground Robots (LAGR). Online source: http://www.darpa.mil/ipto/programs/lagr/vision.htm.
11. DeSouza, G. and Kak, A. 2002. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267.
12. Durrant-Whyte, H. 2001. A critical review of the state-of-the-art in autonomous land vehicle systems and technology, *Sandia Report SAND2001-3685*, Sandia National Laboratories, Albuquerque, NM.
13. Ettinger, S.M., Nechyba, M.C., Ifju, P.G., and Waszak, M. 2002. Towards flight autonomy: Vision-based horizon detection for micro air vehicles. *Florida Conference on Recent Advances in Robotics*.
14. Giachetti, A., Campani, M., and Torre, V. 1998. The use of optical flow for road navigation. *IEEE Trans. on Robotics and Automation*, 14(1):34–48.
15. Horswill, I. 1993. Polly: A vision-based artificial agent. *Proceedings of AAAI Conference*, 824–829.
16. Iagnemma, K. and Dubowsky, S. 2002. Terrain estimation for high-speed rough-terrain autonomous vehicle navigation. *Proceedings of the SPIE Conference on Unmanned Ground Vehicle Technology IV*.
17. Kang, D. and Jung, M. 2003. Road lane segmentation using dynamic programming for active safety vehicles. *Elsevier Pattern Recognition Letters*, 24(16):3177–3185.
18. Kato, Z., Zerubia, J., and Berthod, M. 1992. Satellite image classification using a modified metropolis dynamics. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 3:573–576.
19. Kato, Z. 1994. Modelisations markoviennes multiresolutions en vision par ordinateur. Application a' la segmentation d'images SPOT. PhD Thesis, INRIA, Sophia Antipolis, France.
20. Kim, H., Hong, S., Oh, T., and Lee, J. 2002. High speed road boundary detection with CNN-based dynamic programming. *Advances in Multimedia Information Processing—PCM 2002: Third IEEE Pacific Rim Conference on Multimedia*, 806–813.
21. Lieb, D., Lookingbill, A., and Thrun, S. 2005. Adaptive road following using self-supervised learning and reverse optical flow. *Proceedings of Robotics: Science and Systems*.
22. Lorigo, L.M., Brooks, R.A., and Grimson, W.E.L. 1997. Visually-guided obstacle avoidance in unstructured environments. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 373–379.
23. Manduchi, R., Castano, A., Talukder, A., and Matthies, L. 2005. Obstacle detection and terrain classification for autonomous off-road navigation. *Auton. Robots*, 18(1):81–102.
24. Moorehead, S., Simmons, R., Apostolopoulos, D., and Whittaker, W.L. 1999. Autonomous navigation field results of a planetary analog robot in Antarctica. *International Symposium on Artificial Intelligence, Robotics and Automation in Space*.
25. Murray, D. and Little, J. 2000. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171.
26. Pilutti, T. and Ulsoy, A.G. 1998. Decision making for road departure warning systems. In *Proceedings of the American Control Conference*.
27. Pomerleau, D. 1995. RALPH: Rapidly adapting lateral position handler. *IEEE Symposium on Intelligent Vehicles*, Detroit, Michigan, USA.
28. Redmill, K., Upadhya, S., Krishnamurthy, A., and Ozguner, U. 2001. A lane tracking system for intelligent vehicle applications. *Proc. IEEE Intelligent Transportation Systems Conference*.
29. Rasmussen, C. 2002. Combining laser range, color, and texture cues for autonomous road following. In *Proc. IEEE Inter. Conf. On Robotics and Automation*, Washington, DC.
30. Shi, J. and Tomasi, C. 1994. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, 593–600.
31. Shoemaker, C.M. and Bornstein, J.A. 1998. Overview of the demo III UGV program. *Proc. of the SPIE Robotic and Semi-Robotic Ground Vehicle Technology*, 3366:202–211.
32. Singh, S. and Sharma, M. 2001. Minerva scene analysis benchmark. *Proc 7th Australian and New Zealand Intelligent Information Systems Conference*, Perth.
33. Sofman, B., Lin, E., Bagnell, J., Vandapel, N., and Stentz, A. 2006. Improving robot navigation through self-supervised online learning. *To Appear in Proceedings of Robotics: Science and Systems*.
34. Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. *Proceedings of IEEE International Conference on Robotics and Automation*, 4:3310–3317.
35. Ulrich, I. and Nourbakhsh, I. 2000. Appearance-based obstacle detection with monocular color vision. In *Proceedings of AAAI Conference*, pp. 866–871.
36. Yasnoff, W.A., Mui, J.K., and Bacus, J.W. 1977. Error measures for scene segmentation. *Pattern Recognition*, 9(4):217–231.