Off-road Path Following using Region Classification and Geometric Projection Constraints[*]

Yaniv Alon
Mobileye Ltd.
Jerusalem, Israel

Andras Ferencz
Mobileye Ltd.
Jerusalem, Israel

Amnon Shashua
School of Eng. and CS
The Hebrew University of Jerusalem

Abstract

We describe a realtime system for finding and tracking unstructured paths in off-road conditions. The system was designed as part of the recent Darpa Grand Challenge and was tested over hundreds of miles of off-road driving. The unique feature of our approach is to combine geometric projection used for recovering Pitch and Yaw with Learning approaches for identifying familiar "drivable" regions in the scene. The region-based component segments the image to "path" and "non-path" regions based on texture analysis borne out of a learning-by-examples principle. The boundary-based component looks for the path bounding lines assuming a geometric model of a planar pathway bounded by parallel edges taken by a perspective camera. The combined effect of both sub-systems forms a robust system capable of finding the path even in situations where the vehicle is positioned out of the path — a situation which is not common for human drivers but is relevant for autonomous driving where the vehicle may find itself occasionally veering out of the path.

1. Introduction

The interest in paved and unstructured road/path following has attracted interest for more than two decades with early systems focused on paved road following [3, 12] (see also survey in [2]) and more recent attempts to handle off-road conditions — some of it triggered by the races of the Darpa Grand Challenge off-road autonomous driving competitions [4] held in 2004 and 2005 in the Mojave desert — [10, 1, 9, 13, 15].

Paved road following is largely considered a "solved" problem as a growing number of automotive manufacturers are offering lane departure warning systems. However, off-road path detection and following poses several interesting and new challenges due to the unstructured nature of the scene. Fig. 1 illustrates some of the challenges one typically faces: (i) path boundaries are somewhat ill-defined in cases where strong non-boundary longitudinal edges are present due to ditches, ruts, and tire marks of other vehicles, (ii) the boundary (when present) may be defined by various different texture properties such as due to vegetation, change of path material or change of geometry (in mountainous driving) (iii) the type of terrain or path material may change considerably from one area to the next, and (iv) unlike human driving, where the vehicle is always *on path*, in autonomous driving the assumption that the "drivable" area is straight ahead does not always hold.

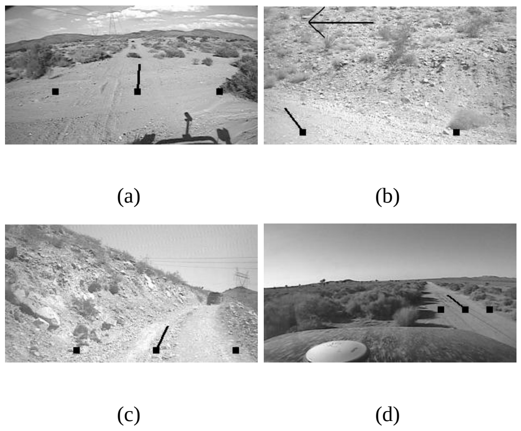


Figure 1. Scenes from autonomous driving in the Mojave desert taken from the 2004 DGC race. System output includes the left and right path boundary point, the center point and the heading orientation. Our system can detect the path even when the boundaries are somewhat ill-structured (a), the vehicle is in a sharp curve (b), the path texture is not uniform (c), and the vehicle is off the path.

The existing approaches for off-road path following fall into two categories: those that attempt to define the forward "drivable" image region by combination of clustering and texture analysis [9, 10] and those that seek the vanishing point (determined by the Pitch and Yaw angles of the camera relative to the road surface) as a way of defining the path

[*]This work is part of Yaniv Alon's MSc dissertation completed at the Hebrew University. Andras Ferencz was at UC Bekeley during the time when this work was completed.

boundaries [15]. The first approach lacks a geometric model of the path which can be described by a small number of parameters assuming that the path ahead is planar and viewed by a perspective projection, whereas the second approach focuses only on detecting boundaries (which when behaved correctly could vote towards the position of the vanishing point) and ignores the ”region-based” nature of the path. Both approaches have their pros and cons but neither single approach can handle the full extent of the required off-road challenge.

In our work we propose combining a region-based classification subsystem of image texture to ”path” and ”non-path” regions together with a geometric subsystem consisting of a flat world assumption taken from a perspective projection for finding the path boundary lines. Both subsystems complement each other thus combining the strengths of both the region-based and boundary-based approaches. Specifically, we use a variety of texture filters together with a learning-by-examples Adaboost [6] classification engine to form an initial image segmentation into Path and Non-path image blocks. Independently, we use the same filters to define candidate texture boundaries and use a projection-warp search over the space of possible Pitch and Yaw parameters in order to select a pair of boundary lines which are consistent with both the texture gradients and the geometric model. The area-based and boundary-based modules are then weighted by their confidence values to form a final path model for each frame.

The system was implemented on a Power-PC PPC7467 1GHZ running at 20 frames per second. We have run the system successfully on 6 hours of recorded data of the entire 2004 DGC race and completed successfully around 50 miles of the 2005 race — both held in the Mojave desert [4]. The 2005 GDC race was in cooperation with the Golem/UCLA group — the vehicle platform can be seen in Fig. 2.

2. Feature Measurements for Classification and Boundary Detection

As mentioned above, our system combines two modules — region-based and boundary based — working in tandem. The region based module classifies image blocks into Path and Non-path labels based on a learning training set and the boundary-based module fits a geometric camera constraint on the boundaries of the path. Both modules require means for measuring texture features — either to be later fed into a classifier or as a basis for deciding where the boundaries between the path and non-path regions may reside. In this section we briefly describe three different feature extraction schemes we used during research including oriented filters, Walsh-Hadamard kernels and Moments.

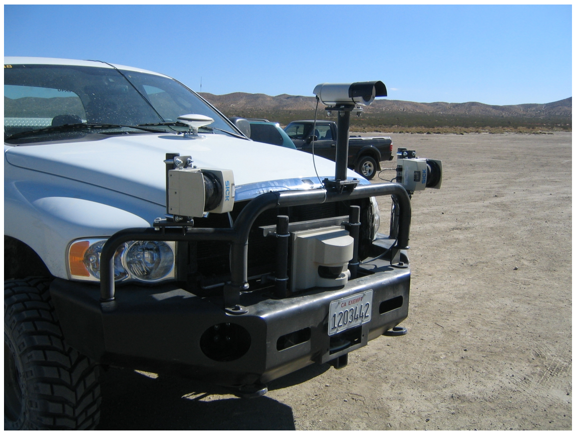


Figure 2. The Golem/UCLA group vehicle used as our test platform. In this installation the camera is mounted inside a housing on top of a pole connected to the front bumper. Other installations had the camera mounted on the windshield.

Oriented Gaussian Derivatives Filters We considered an oriented Gaussian Derivatives filter bank similar to the one used by Malik *et al*. [11] for partitioning grayscale images into disjoint regions with coherent brightness and texture. For a given scale σ, the filter bank contains even and odd-symmetric filters

$$\hat{g}_{odd}(x,y,\sigma) = \frac{d^2}{dy^2}(exp(\frac{y^2}{2\sigma^2})\exp(\frac{x^2}{l^2 2\sigma^2})) \quad (1)$$

$$\hat{g}_{even}(x,y,\sigma) = Hilbert(g_{odd}(x,y,\sigma)) \quad (2)$$

and one center-surround filter

$$\begin{aligned}\hat{cs}(x,y,\sigma) &= \exp(\frac{y^2}{1.5^2\sigma^2})\exp(\frac{x^2}{1.5^2\sigma^2}) \\ &\quad -exp(\frac{y^2}{0.5^2\sigma^2})\exp(\frac{x^2}{0.5^2\sigma^2})\end{aligned}$$

The coefficient $l = 3$ is the aspect ratio between the two 1D Gaussians that form a quadrature pair ([5]) $\hat{g}_{odd}$ and $\hat{g}_{even}$, i.e., having the same frequency response but differ in phase. The actual filter bank contains zero-mean versions g_{odd}, g_{even}, cs normalized to unit L_1 norm. For a given scale σ, the filter bank contains rotated versions of g_{odd} and g_{even} in four equally spaced orientations, and one cs filter which makes it a total of nine filters per scale. The filter response around each filter is calculated with two scales adjusted to the row position due to foreshortening (total of 18 responses per pixel).

Walsh-Hadamard Kernels The Gaussian derivatives filter bank forms a computationally expensive approach which could be prohibitive for a realtime system. We considered using as an alternative a fast convolution approach based on

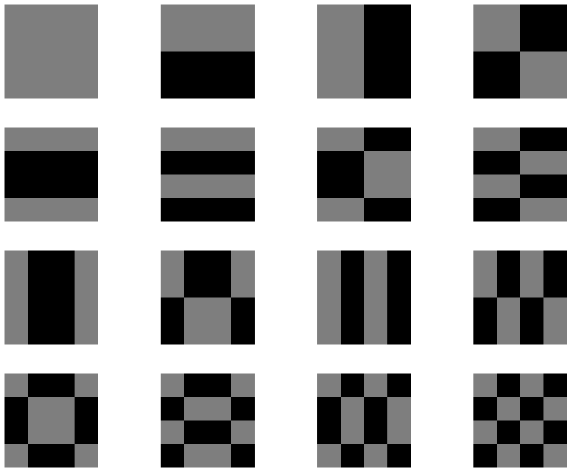


Figure 3. The 16 Walsh-Hadamard 4×4 binary kernels used for texture analysis. Fast execution of convolution with the kernels (see text) requires that the filters be used in a specific order (top-bootm, left-right).

the Walsh-Hadamard filter bank which guarantees an $O(1)$ cost per pixel over the entire collection of filters. The 1D filter bank of 2^k orthogonal filters proposed by [7] can be described recursively by a binary tree whose leaves are the filters:

$$\begin{aligned} U_s^{(0)} &= 1 \\ U_s^{(k)} &= \{[u_s^{(k-1)}, \alpha_i^{(k)} u_s^{(k-1)}]\} \\ &\quad \text{s.t. } u_s^{(k-1)} \in U_s^{(k-1)}, \alpha_i^{(k)} \in \{+1, -1\} \end{aligned}$$

where α is a binary vector. The 2D filter bank is constructed by creating rank-1 matrices using again a binary α vector. The crucial property of the WH kernels is that (i) they form a basis, and (ii) they are efficiently convolved, and (iii) only bit-shifts and integer additions and subtractions are required (for more details see [7]). Fig. 3 displays the 16 4×4 filter kernels we used in our experiments.

Moments Energy The third feature extraction approach is based on the computations of high-order moments. The discrete $(p+q)^{th}$ order moment $\hat{F}_{pq}$ over a square window of size $w \times w$ is defined by (see [8, 16]):

$$\hat{F}_{pq} = \sum_{m=-w/2}^{w/2} \sum_{n=-w/2}^{w/2} I(m,n) x_m^p y_n^q \qquad (3)$$

where $x_m = \frac{m-i}{w/2}$ and $y_m = \frac{n-j}{w/2}$. We used $p, q \in \mathbb{N}$ such that $p + q \leq 2$ and w=3. Convolution of the input image $I(x,y)$ with $\hat{F}_{pq}$ results in six filter response images $\hat{I}_{pq}$. The filter response images are then normalized:

$$I_{pq}(x,y) = \max(1, \sigma * |\hat{I}_{pq}(x,y) - \psi(\hat{I}_{pq})|) \qquad (4)$$

where $\psi(\hat{I}_{pq})$ is the mean value of $\hat{I}_{pq}$ and σ is a normalization factor that was set to 0.001 in our implementation. Example of the filter responses over a warped image can be seen in Fig. 6(c).

3. Path Detection using Classification

The feature extraction methods described above are used as a representation for a block-wise path classification procedure using an Adaboost classifier over a training set. The input image $\hat{I}$ is normalized:

$$I = \frac{\hat{I} - \sigma(\hat{I})}{\mu(\hat{I})} \qquad (5)$$

where $\sigma(\hat{I})$ and $\mu(\hat{I})$ are the mean and standard deviation values. For efficiency, we divide the image into partially overlapping blocks of size 16×16 (six pixel overlap in each dimension). The filter banks described above are applied to all image pixels and a descriptor vector is generated per block B in the following way.

- For the Gaussian filters we have 18 responses per pixel over all the pixels of a block B and the descriptor vector is $V(B) = (hist(m), \mu_f, \sigma_f, \mu_B, \sigma_B)$ where $hist(m)$ is the histogram over the filter IDs corresponding to the maximal response filter per pixel, μ_f, σ_f are the mean and std of the maximal filter responses per pixel and μ_B, σ_B are the mean and std of the (normalized) pixel values in B.

- For the Walsh-Hadamard filter bank, we have 16 filter responses per pixel. The descriptor vector of a block B is $V(B) = (\mu_1, .., \mu_{16}, \sigma_1, ..., \sigma_{16})$ where μ_i, σ_i are the mean and std of the i'th filter response over all the pixels of B.

- For the moments filter bank, we have 6 filter responses per pixel. The descriptor vector of a block B is $V(B) = (\mu_1, .., \mu_6, \sigma_1, ..., \sigma_6)$ where μ_i, σ_i are the median and std of the i'th filter response over all the pixels of B.

The feature vectors per training blocks B_i with labels $y_i \in \{-1, 1\}$ were fed into an Adaboost classifier engine. Each entry of the feature descriptor vector can be considered as a "weak" learner in the sense that it forms a class discrimination. The main idea of AdaBoost is to assign each example of the training set a weight. At the beginning all weights are equal, but in every round the weak learner returns a hypothesis, and the weights of all the miss-classified examples by that hypothesis are increased. That way the weak learner is forced to focus on the difficult examples of the training set. The final hypothesis is a combination of the hypotheses of all rounds, namely a weighted majority vote, where hypotheses with lower classification error have higher weight.

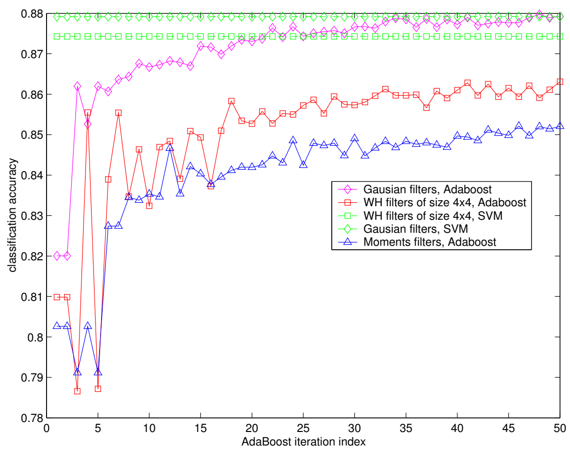


Figure 4. Accuracy of the strong hypothesis during the first 50 AdaBoost iterations over a text set of 150 images picked from the 2004 DGC race. The filter banks include the Gaussian derivatives (best performance, worst computation time), Walsh-Hadamard gray-code filters (best computation time, reasonable performance) and Moment filters (worst performance, medium level computation time). For comparison, we ran also SVM classifier on the Gaussian and WH filters — the added accuracy is relatively small whereas the additional computation time is very significant.

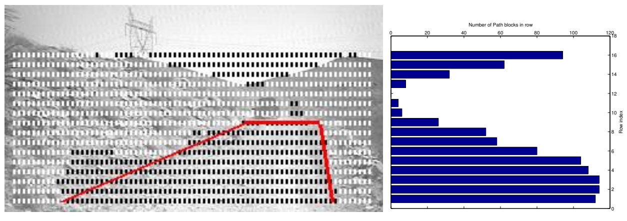
To test the accuracy of the classification we selected 150 test images from 6 hours recordings we had from the 2004 DGC race. We compared the accuracy of the Path detection against the different feature extraction schemes and against an SVM classifier trained on the WH and Gaussian filter banks (see Fig. 4). The best accuracy was achieved with the Adaboost and SVM over the Gaussian derivatives filter bank and the SVM over the WH filter bank. The Adaboost over WH filters was slightly behind (86% compared to 88%) and was chosen as the preferred feature representation for the Adaboost engine due to its computational efficiency.

Once the blocks were assigned the labels of Path and Non-path from the Adaboost classifier the system underwent the following steps for defining the path region:

1. Sky: Since the sky region can deceptively appear similar to a Path region we look for a minima over a row-sum of Path labeled regions. The minimal row-sum point defines the end-of-path mark which all Path regions above that mark are ignored. (see Fig. 5b).

2. Top Bounding Line: removal of spurious false positives near the horizon is done by defining for each column x the top row y_x containing Path labeled regions:

$$y_x = \underset{\bar{y} \in Y_b(x)}{\text{argmax}} \sum_{y=min(Y_b(x))}^{\bar{y}} C(x,y),$$



(a) (b)

Figure 5. Texture Classification Module: (a) Centers of blocks that were classified as Path are marked in Black and Non-path blocks shown in White. The overlaid straight lines are the path boundaries and the top line marks the visibility path edge. (b) sum of the Path classified blocks in each row. There is a local minimum near the horizon — this is used to remove skyline blocks.

where $C(x,y) \in \{-1,1\}$ are the block labels and $Y_b(x)$ is the set of center blocks coordinates. We sort y_x in descending order and pick the 5th from the top as a virtual line, the "top line" y_o, which marks the last visible row of the path. The top-line is used for an approximate range to the farthest point on the path and also for deriving a confidence value for the region-based path finding process for the frame in question.

3. Left/Right Bounding Lines: the left and right bounding lines of the path are derived via a minimal error separating line on each side of the path, as follows. A center line is drawn as a least-squares fit to the geometric centers defined by the positive blocks in each row. Two separator lines are derived, one from each side of the center line, by searching over the 2D space of position and orientation of the line so that a separation error is minimized. The separation error $e(l)$ is minimizes the chances that Non-path blocks are contained inside the Path region: $e(l) = \alpha|F_p(l)| + |F_N(l)|$ where $F_p(l)$ is the set of the Path classified blocks that are to the left of the separator l, and $F_N(l)$ is the set of Non-pathblocks that are to the right of l (assuming left bounding line — the description is reversed for the right-hand line). Fig. 10 (see section 5) displays the precision-recall curve as we change the parameter α. For our final system we chose $\alpha = 4.0$.

Fig. 5a shows an example of a frame with the classified blocks and the top, left and right separator lines which bound the path region. The scheme described above works well under various situations, but as mentioned in the introduction, lacks the geometric notion of a path borne by the assumption of a perspective camera, flat world assumption and parallel path bounding lines in the scene. In the next section we describe the projection-warp path boundary scheme which looks for the bounding lines via a search for the road's vanishing point followed by a combination of the two schemes to form a robust path finding system.

4. Path Boundaries via Projection-Warp iterations

The techniques introduced is Section 3 work well in most of the scenes which includes paths of similar properties to those presented in the training set. However, the system may encounter unfamiliar path textures, or path texture that is very similar to non-path areas in the training set, and thus report that it is unable to find a drivable area ahead.

In this section, we suggest a second technique, inspired by the RALPH system for paved roads [12], which does not rely on prior learned texture information. Instead, it makes the assumption that the area immediately in-front of the vehicle is on a path whose texture properties are different from the surrounding non-drivable areas. For this cue to be reliable, we have to constrain the solution to a strict geometric model where the path boundary lies on straight parallel edges. This allows us to reduce the drivable path problem to 4 degrees of freedom: (x, y) position of the vanishing point, and left and right distance to the edge of the path.

The geometric constraint borne out of the flat world assumption, perspective camera and parallel path boundaries in the world suggest the following projection-warp scheme [12] per frame: given a hypothesis of Pitch and Yaw angles of the camera, the image is warped to form a top view in world coordinates. In the warped image the path boundaries are supposed to be *parallel* vertical lines if indeed the Pitch and Yaw angles are correct. A projection of the image texture edges onto the horizontal axis will produce a 1D profile whose peaks correspond to vertical texture edges in the warped image. We look for a pair of dominant peaks in the 1D profile and generate a score value which is then maximized by search over the Pitch and Yaw angles.

In a nutshell, we start with the Pitch and Yaw angle estimates of the previous frame followed by an incremental Pitch and Yaw estimation using optic-flow and a small motion model. The incremental estimation is based on the solution of the motion equation per image point (x, y):

$$xw_x + yw_y = yu - xv,$$

where (u, v) are the flow (displacements) of the point (x, y) and w_x, w_y are the Pitch and Yaw angles. We use point tracking in the vicinity of the horizon and the close areas in front of the camera in order to recover w_x, w_y in a least-squares fashion. The incremental Pitch and yaw angles are added to the previous frame estimate to form the current frame starting point for Pitch/Yaw finding. The image is then warped with the current Pitch/Yaw estimate.

The warped image is divided into overlapping 10×10 blocks with each pixel forming a block center. Using the Walsh-Hadamard filter bank described above we estimate the likelihood $e^{-\Delta}$ that the vertical line passing through the block center forms a texture gradient where Δ is the L_1 distance between the WH vector descriptors of the two respective halves of the block. Horizontal texture gradients are projected vertically onto the x-axis for the purpose of evaluating a Pitch/Yaw hypothesis. A strong hypothesis is backed by a dominant pair of peaks in the projection (see Fig. 6(d)). The path boundaries and other vertical elements in the image create high values in the projection, while low values are most likely caused by non-continuous vertical texture gradients typically associated with bushes, rocks, and so forth. The peaks in this projection are maximized when the vanishing point hypothesis is correct and lines up with the path edges (and possibly other parallel features). To find high and narrow peaks, we convolve the projection with a box-shaped template. To score our vanishing point hypothesis, we remove low peaks and then sum the remaining ones. Maximum values for this score suggest a strong hypothesis. By finding the highest peaks for these hypotheses, our system is able to find the lateral position of the left and right boundaries. Fig. 6(e) shows the "cleaned up" 1D projection profile and the associated pair of peaks corresponding to the path boundary lines.

Our projection-warp iterative scheme is summarized as follows:

Algorithm 1 (Projection-Warp)

```
1) Calculate incremental Pitch and Yaw angles using optic-flow and the small motion model. Using the previous frame and incremental estimates, generate the current estimate of the vanishing point.
2) Create a warped based on the vanishing point.
3) Calculate vertical texture gradients within the warped image using the WH filter bank on $10 \times 10$ overlapping blocks.
4) Define a reasonable range of Pitch and Yaw angles around the current estimate.
5) For each vanishing point hypothesis, perform a warping and project the gradient edges onto a 1D profile.
6) Clean-up the 1D profile by convolution with a box filter and remove low peaks. Sum the remaining values and obtain a score.
7) Repeat the project-warp steps and set the Pitch and Yaw angles to those that maximize the projection score.
```

5. Results

We describe below a number of performance measures we have collected of the system. The ultimate performance test is how well it performs on the task of aiding the navigational unit of an autonomous vehicle to safely traverse a terrain at high speeds. Our system was used for this purpose by the Golem/UCLA team competing in the 2005 DARPA Grand Challenge. Our output was combined with other sensors to allow the vehicle to stay within the path while driv-

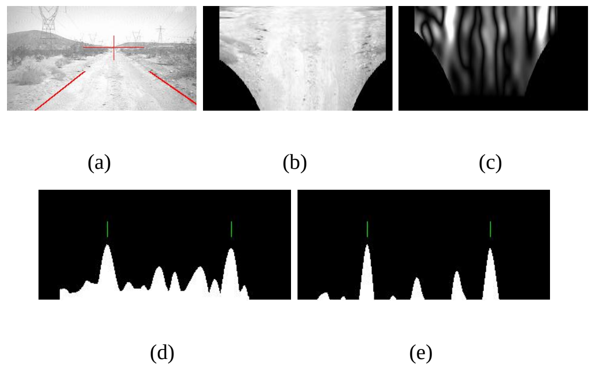


Figure 6. Projection-Warp Search: (a) original image with the overlaid path boundary and vanishing point results. (b) the warped image. (c) texture gradients magnitude. (d) projection: vertical sum of gradients. (e) projection profile followed by convolution with a box filter. The two lines on top of the histogram marks the path boundaries.

ing at speeds up to 50 mph on straight segments as well as to navigate mountainous trails.

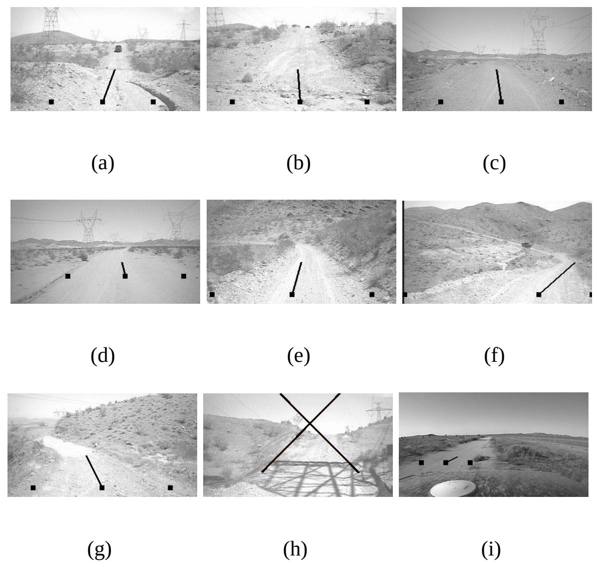


Figure 7. Sample images and system output from 6 hours of driving in the Mojave desert (2004 DGC race). The path is marked by two left-right boundary points and a center point with heading orientation. The "X" mark in (h) coincides with Zero confidence (i.e., system deactivates momentarily) due to confusion with multiple shadows. In (i) the path is detected even though the vehicle is not centered on the path (a situation which is common for autonomous driving).

Our system was trained over 200 randomly selected images from 6 hours of video over 140 miles of trails in the Mojave desert covering the course of DARPA Grand Challenge 2004. The trails cover a large variety of terrain types

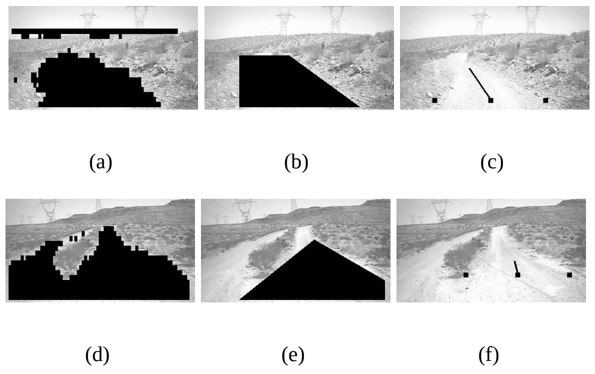


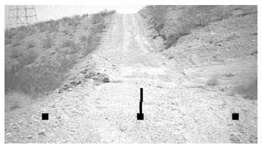
Figure 8. The Path /Non-path texture classification (a),(d) is followed by sky blocks removal and by fitting three lines (arranged as a trapezoid) to the Path blocks. In (b),(e) all the blocks within the trapezoid are labeled as Path, and all the others as Non-path. Finally, the path boundaries at a given distance is calculated (c),(f) together with the path center and heading angle.

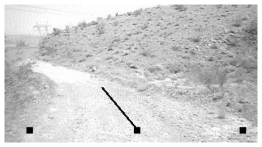
including sandy straight paths, gravel covered winding trails and rocky mountain passes. The video was recorded with a camera located inside the cabin and mounted on the windshield near the rearview mirror. A field of view of 45 degrees was used during training but during the 2005 DGC race a wider field of view of 80 degrees was adopted. Following the training phase, we tested the system performance on the original 6 hours of video with a 45 degrees FOV and with one hour of video recorded with an 80 degrees FOV. Sample images of the system output during these tests are shown in Fig. 7, while Fig. 8 shows the results of each of the main parts of the classification based algorithm.

Overall performance. The most meaningful overall system performance measure is to count how often (what fraction of frames) the system produced correct path edge positions and, where appropriate, heading angles. Furthermore, it is crucial for the system to know when it can not determine the path accurately, so that the vehicle can slow down and rely more on information from other sensors. Our results are broken down to different terrain types. For each, representative challenging clips of 1000 frames were selected (see Fig. 9) and the system performance scored on those sequences by a human observer. The path edge distance accuracy was computed by observing the position of the road edge marks approximately 6 meters in front of the vehicle. A frame was labeled incorrect if the path edge marker at that location appeared to be more then 30 cm ($\approx$ 18 pixels) away from the actual path boundary. For straight paths the perceived vanishing point of the path was also marked, and our algorithm's heading indicator was compared to the lateral position of that point.

A. On relatively straight segments with a comfortably wide path, the navigation system allows the vehicle to drive at speeds up to 50 mph. For those type of scenes (clip (a) in

(a)


(b)


(c)

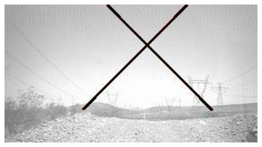

(d)

Figure 9. Sample images from the three clips used for numerical performance measures. (a) loosely marked path boundaries. (b) crossing dry river bed with geometric elevation ahead. (c) mountain pass. (d) Reaching the crest of the hill, short segment of the road is visible, and the system reports low confidence detection.

Fig. 9), our system reported availability (high system confidence) **100%** of the time while producing accurate path boundary locations **99.5%** of the time. The mean angular deviation of the heading angle from the manually marked vanishing point was $1.7\,\mathrm{deg}$.

B. This clip is an example of an uneven terrain with elevation changes. The vehicle passes through a dry river ditch (Fig. 9(b)) where both the path texture and scene geometry is difficult. When the vehicle reaches the crest of the hill (Fig. 9(d)) only a short segment of road is visible. In this case, the system reported low confidence (was unavailable) **8%** of the time. When available, however, the accuracy in boundary locations was **98%**.

C. This clip contains a winding mountain pass (Fig. 9(c)), which is difficult due to path curvature as well as texture variation. Nevertheless, our system was available throughout the clip and achieved an accuracy of **96%** in detecting the path boundary.

Method Comparison. Of the 2 methods presented in the paper, we set the system to primarily rely on the texture classification (section 3) subsystem with the geometric subsystem (section 4) used as a fall back when the *in-path* and *out-of-path* texture could not be well classified. In scene **(A)**, due to the vegetation on the side, the texture could be well classified while the exact location of the texture boundary of the path is quite fuzzy. This heavily favors the classification approach: in fact, the geometric subsystem was active less than **2%** of the time. On the other hand, clip **(B)** had regions with non-typical texture: in this case, the geometric subsystem was dominant between **9** and **20%** of the frames, depending on the exact parameter tuning.

Pixel-wise Performance. Another way to measure the performance of the path classification is to count the number of path pixels that were correctly and incorrectly labeled. We randomly selected 60 frames from the 6 hours of video, hand labeled the Path and Non-pathpixels, and computed the precision-recall curve for different choices of α, the factor from section 3. The precision-recall curve is presented in Fig. 10. Only pixels below the manually labeled limit line were included in those results.

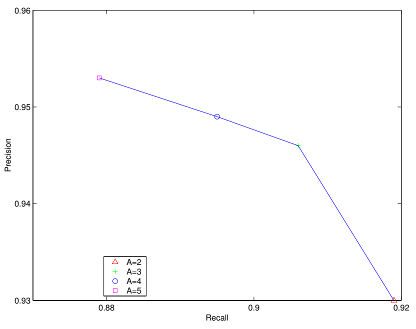


Figure 10. Precision-recall curve.

We repeated this test for the 3 clips above (50 frames each) and compared our results against that of [14] and [10]. While a direct comparison is impossible without running the algorithms on the same clips, we attempted to convert our statistics to the performance metrics that were employed in those studies (“classification accuracy” and “pixel coverage” respectively). The comparison is summarized in Tables 1 and 2.

Weaknesses. As our algorithm relies on texture, our system seems to perform less well (reporting low confidence) where there are significant shadows present in the scene. Unfortunately we currently do not have enough training or testing data available to quantify or to attempt to overcome this problem, therefore as a result the system outputs ”No Path Visible” (an ”X” marking as seen in Fig. 7h) in such situations.

Supplemental Videos: We have posted in http://www.cs.huji.ac.il/∼shashua/cvpr06/ three video clips of our system at work under different terrain type conditions from the 2004 race. The videos show the original footage overlaid with graphics marking the path left and right boundary points and the vehicle orientation as has been recovered form the video (only). The clip ”mountain.mpg” shows a mountain passage terrain, the clip ”texture.mpg” shows a typical desert type of terrain with vegetation material on the sides of the path and the clip ”elevationChanges.mpg” shows the situation of hill climbing (system deactivates momentarily at the crest of the hill).

Method	Classification accuracy
Ours (random set of images)	90.0
Rasmussen	88.6

Table 1. Results: Classification accuracy metric.

Road type	Pixel coverage
ill-structured boundaries	0.790
Ditches, large pitch changes	0.732
Winding mountain path	0.881
Random set of images	0.822
Lieb *et al*.	0.697

Table 2. Results: Pixel coverage metric.

6. Summary

We have described an off-road path-finding algorithm that was designed as an aide for autonomous driving as part of the recent 2005 DGC race. The unique feature of the system is that it incorporates two different detection modalities one based on texture classification of image into ”Path” and ”Non-path” regions followed by cleaning-up processes for turning the classification result into a path with straight-line boundaries and orientations. The second module (projection-warp search) is based on a geometric approach governed by camera parameters (Pitch and Yaw), flat world assumption and parallel path boundaries in the scene. The two modules run in parallel and the system output is governed by the module with the highest confidence. The texture classification module uses a Walsh-Hadamard filter bank followed by an Adaboost trained classifier. The projection-warp search combines motion estimation for an initial camera parameters estimation followed by a search maximizing the sharpness of peaks in the 1D projection of the vertical texture gradients.

Acknowledgement

We thank Stefano Soatto and the entire UCLA and Golem teams for the fruitful cooperation during the 2005 DGC race. This work was partly funded by a grant No. 0397545 from the Israeli defense ministry.

References

[1] J. Crisman and C. Thorpe. Unscarf, a color vision system for the detection of unstructured roads. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2496 – 2501, April 1991. 1

[2] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(2):237–267, 2002. 1

[3] E. D. Dickmanns and A. Zapp. Autonomous high speed road vehicle guidance by computer vision. In R. Isermann, editor, *Automatic Control—World Congress, 1987: Selected Papers from the 10th Triennial World Congress of the International Federation of Automatic Control*, pages 221–226, Munich, Germany, jul 1987. 1

[4] http://www.darpa.mil/grandchallenge/. 1, 2

[5] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. 2

[6] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996. 2

[7] Y. Hel-Or and H. Hel-Or. Real-time pattern matching using projection kernels. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1430–1445, September 2005. 3

[8] M. Hu. Visual pattern recognition by moment invariants. *IEEE Trans. Inform. Theory*, 8(2):179–187, February 1962. 3

[9] D. Kuan, G. Phipps, and A.-C. Hsueh. Autonomous robotic vehicle road following. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):648–658, 1988. 1

[10] D. Lieb, A. Lookingbill, and S. Thrun. Adaptive road following using self-supervised learning and reverse optical flow. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005. 1, 7

[11] J. Malik, S. Belongie, T. K. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001. 2

[12] D. Pomerleau. Ralph: Rapidly adapting lateral position handler. In *IEEE Symposium on Intelligent Vehicles*, pages 506 – 511, September 1995. 1, 5

[13] C. Rasmussen. Laser range-, color-, and texture-based classifiers for segmenting marginal roads. In *IEEE Conference on Computer Vision and Pattern Recognition Technical Sketches*, 2001. 1

[14] C. Rasmussen. Combining laser range, color, and texture cues for autonomous road following. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 4320 – 4325, 2002. 7

[15] C. Rasmussen. Grouping dominant orientations for ill-structured road following. In *CVPR (1)*, pages 470–477, 2004. 1, 2

[16] M. Tucceryan. Moment-based texture segmentation. *Pattern Recogn. Lett.*, 15(7):659–668, 1994. 3