Auton Robot (2009) 26: 103–122
DOI 10.1007/s10514-009-9113-3

Finding multiple lanes in urban road networks with vision and lidar

Albert S. Huang · David Moore · Matthew Antone · Edwin Olson · Seth Teller

Received: 7 November 2008 / Accepted: 6 March 2009 / Published online: 24 March 2009
© Springer Science+Business Media, LLC 2009

Abstract This paper describes a system for detecting and estimating the properties of multiple travel lanes in an urban road network from calibrated video imagery and laser range data acquired by a moving vehicle. The system operates in real-time in several stages on multiple processors, fusing detected road markings, obstacles, and curbs into a stable non-parametric estimate of nearby travel lanes. The system incorporates elements of a provided piecewise-linear road network as a weak prior.

Our method is notable in several respects: it detects and estimates multiple travel lanes; it fuses asynchronous, heterogeneous sensor streams; it handles high-curvature roads; and it makes no assumption about the position or orientation of the vehicle with respect to the road.

We analyze the system's performance in the context of the 2007 DARPA Urban Challenge. With five cameras and thirteen lidars, our method was incorporated into a closed-loop controller to successfully guide an autonomous vehicle through a 90 km urban course at speeds up to 40 km/h amidst moving traffic.

Keywords Lane estimation · Vision · Lidar · Lane-finding

1 Introduction

The road systems of developed countries include millions of kilometers of paved roads, of which a large fraction include painted lane boundaries separating travel lanes from each other or from the road shoulder. For human drivers, these markings form important perceptual cues, making driving both safer and more time-efficient (Miller 1993). In mist, heavy fog or when a driver is blinded by the headlights of an oncoming car, lane markings may be the principal or only cue enabling the driver to advance safely. Moreover, roadway designers use the number, color and type of lane markings to encode spatially-varying traffic rules, for example no-passing regions, opportunities for left turns across oncoming traffic, regions in which one may (or may not) change lanes, and preferred paths through complex intersections.

Even the most optimistic scenarios for autonomous vehicle deployment assume the presence of large numbers of human drivers for the next several decades. Given the centrality of lane markings to public safety, it is clear that they will continue to be maintained indefinitely. Thus autonomous vehicle researchers, as they design self-driving cars, may assume that lane markings will be commonly encountered.

We define the lane-finding problem as divining, from live sensor data and (when available) prior information, the presence of one or more travel lanes in the vehicle's vicinity, and the semantic, topological, and geometric properties of each lane. By semantic properties, we mean the lane's travel sense and the color (white, yellow) and type (single, double, solid, dashed) of each of its boundaries. By topological properties, we mean the connectivity of multiple lanes in regions where lanes start, split, merge, or terminate. We use the term geometric properties to mean the centerline location and lateral extent of the lane. This paper focuses on detecting lanes where they exist, and determining geometric information for each detected lane (Fig. 1). We infer semantic and topological information in a limited sense, by matching detected

This research was sponsored by DARPA, Program: Urban Challenge, ARPA Order No. W369/00, Program Code: DIRO, issued by DARPA/CMO under Contract No. HR0011-06-C-0149.

A.S. Huang (✉) · D. Moore · M. Antone · E. Olson · S. Teller
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: albert@csail.mit.edu

Fig. 1 Our system uses many asynchronous heterogeneous sensor streams to detect road paint and road edges (*yellow*) and estimate the centerlines of multiple travel lanes (*cyan*)

lanes to edges in an annotated input digraph representing the road network.

1.1 Related work

Aspects of the lane-finding problem have been studied for decades in the context of autonomous land vehicle development (Dickmanns and Mysliwetz 1992; Thorpe et al. 1988) and driver-assistance technologies (Bertozzi and Broggi 1998; Bertozzi et al. 2000; Apostoloff and Zelinsky 2004; Fletcher and Zelinsky 2006; Kim 2008). McCall and Trivedi provide an excellent survey (McCall and Trivedi 2006). Lane-finding systems intended to support autonomous operation have typically focused on highway driving (Dickmanns and Mysliwetz 1992; Thorpe et al. 1988), where roads have low curvature and prominent lane markings, rather than on urban environments. Previous autonomous driving systems have exhibited limited autonomy in the sense that they required a human driver to "stage" the vehicle into a valid lane before enabling autonomous operation, and to take control whenever the system could not handle the required task, for example during highway entrance or exit maneuvers (Thorpe et al. 1988).

Driver-assistance technologies, by contrast, are intended as continuous adjuncts to human driving. Commonly deployed Lane Departure Warning (LDW) systems are designed to alert the human driver to an imminent (unsignaled) lane departure (Mobileye 2009; Iteris 2009; Pomerleau and Jochem 1996). These systems typically assume that a vehicle is in a highway driving situation and that a human driver is controlling the vehicle correctly, or nearly so. Highways exhibit lower curvature than lower-speed roads, and do not contain intersections. In vehicles with LDW systems, the human driver is responsible for selecting an appropriate travel lane, is assumed to spend the majority of driving time within such a lane, is responsible for identifying possible alternative travel lanes, and only occasionally changes into such a lane. Because LDW systems are essentially limited to providing cues that assist the driver in staying within the current lane, achieving fully automatic lane detection and tracking is not simply a matter of porting an LDW system into the front end of an autonomous vehicle.

Clearly, in order to exhibit safe, human-like driving, an autonomous vehicle must have good awareness of all nearby travel lanes. In contrast to prior lane-keeping and LDW systems, the lane finding system presented here aims to guide a fully autonomous land vehicle through an urban road network. In particular, our system is distinct from previous efforts in several respects: it attempts to detect and classify all observable lanes, rather than just the single lane occupied by the vehicle; it operates in the presence of complex road geometry, static hazards and obstacles, and moving vehicles; and it uses prior information (in the form of a topological road network with sparse geometric information) when available.

The apparent difficulty of matching human performance on sensing and perception tasks has led some researchers to investigate the use of augmenting roadways with a *physical infrastructure* amenable to autonomous driving, such as magnetic markers embedded under the road surface (Zhang 1991). While this approach has been demonstrated in limited settings, it has yet to achieve widespread adoption and faces a number of drawbacks. First, the cost of updating and maintaining millions of kilometers of roadway is prohibitive. Second, the danger of autonomous vehicles perceiving and acting upon a different infrastructure than human drivers do (magnets vs. visible markings) becomes very real when one is modified and the other is not, whether through accident, delay, or malicious behavior.

Advances in computer networking and data storage technology in recent years have brought the possibility of a *data infrastructure* within reach. In addition to semantic and topological information, such an infrastructure might also contain fine-grained road maps registered in a global reference frame; advocates of these maps argue that they could be used to guide autonomous vehicles. We propose that a data infrastructure is useful for topological information and sparse geometry, but reject relying upon it for dense geometric information.

While easier to maintain than a physical infrastructure, a data infrastructure with fine-grained road maps might still become "stale" with respect to actual visual road markings. Even for human drivers, mapping staleness, errors, and incompleteness have already been implicated in accidents in which drivers trusted too closely their satellite navigation systems, literally favoring them over information from their own senses (CNN 2008; Lyall 2007). Static fine-grained maps are clearly not sufficient for safe driving; to operate safely, in our view, an autonomous vehicle must be able to use local sensors to perceive and understand the environment.

The primary contributions of this paper are:

- A method for estimating multiple travel lanes within a typical urban road network using only information from local sensors;
- A method for fusing these estimates with a weak prior, such as a topological road map with sparse metrical information;
- Methods for using monocular cameras to detect road markings; and
- Multi-sensor fusion algorithms combining information from video and lidar sensors.

We also provide a quantitative analysis of our method's operation, describe its failure modes, and discuss several possible directions for future work.

2 Approach

Our approach to lane-finding involves three stages. In the first stage, the system detects and localizes painted road markings in each video frame, using lidar data to reduce the false positive detection rate. A second stage processes road paint detections along with lidar-detected curbs to estimate centerlines of nearby travel lanes. Finally, any detected centerlines are filtered, tracked, and fused with a weak prior to produce one or more non-parametric lane outputs. The data flow of our approach is illustrated in Fig. 2.

Separation of the three stages provides simplicity, modularity, and scalability, allowing us to experiment with each stage independently of the others and to easily substitute different algorithms at each stage. For example, we evaluated and ultimately utilized two separate algorithms in parallel for detecting road paint, both of which are described below. By introducing sensor-independent abstractions of environmental features, our system is able to scale to many heterogeneous sensors.

2.1 Road boundary detection

This section describes two vision algorithms used for detecting painted lines on the road based on matched filters (Sect. 2.1.2) and spatial gradients (Sect. 2.1.3), respectively, as well as a technique for detecting curbs using 3D laser scan data (Sect. 2.1.5). As input, the vision algorithms receive grayscale images from a single camera, pose information from the vehicle's IMU (Sect. 2.1.1), and 3D obstacles detected from lidar if available (Sect. 2.1.4). As output, all detection algorithms produce a list of curves, represented as polylines in a local coordinate frame (Moore et al. 2009), that correspond to painted line markings or physical road boundaries estimated from the sensor data.

The road paint detection algorithms operate independently on each camera and on each temporal image frame.

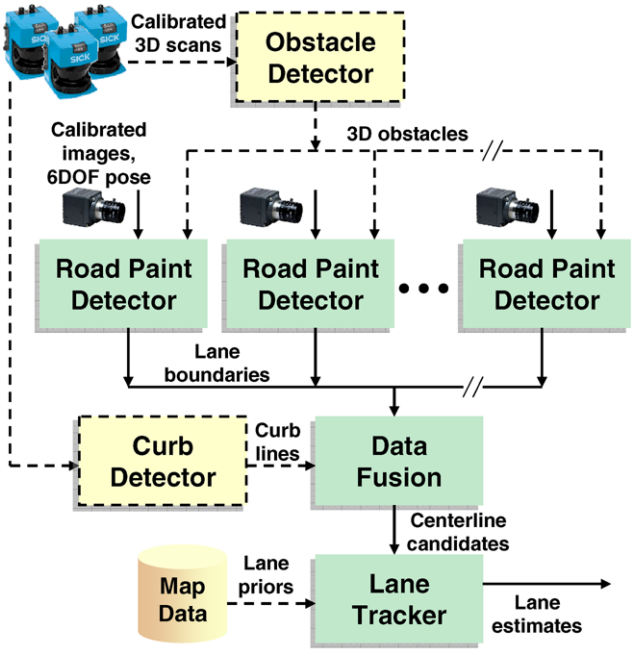


Fig. 2 Raw images acquired by a set of cameras are processed independently and asynchronously to produce lane boundary detections, assisted by real-time vehicle pose estimates and (optionally) obstacles detected from lidar data. Next, spatial/temporal data fusion combines all visual detections, along with curb boundaries (optionally) obtained from lidar data, and outputs high-confidence lane candidates. Finally, lanes are estimated and tracked over time, influenced by curvature constraints and priors generated from map data if available

Although state information could be tracked over time and transferred between frames to assist extraction and curve fitting, we kept the approach stateless since the higher-level lane fusion and tracking stages perform both spatial and temporal filtering in local coordinates. We also eliminate substantial computational expense by operating directly on raw camera images rather than on inverse-perspective corrected images (McCall and Trivedi 2006; Kim 2008), while still reaping the benefits of calibrated cameras and real-world measurements.

We will describe each detection algorithm in greater detail below. First we discuss the importance of accurately instrumented sensor data.

2.1.1 Absolute sensor calibration

Our detection algorithms assume that GPS and IMU navigation data are available, and of sufficient quality to correct for short-term variations in vehicle heading, pitch, and roll during image and laser processing. In addition, we assume that the intrinsic lens parameters (focal length, optical center, and distortion) for each camera and the extrinsic parameters (vehicle-relative pose) for each sensor have been determined in advance. This "absolute calibration" allows sensor data preprocessing in several ways, some of which are illustrated in Fig. 3:

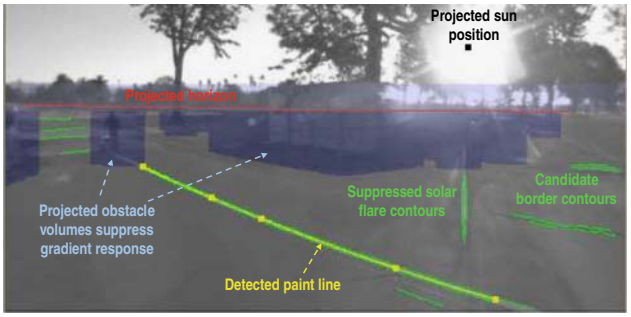


Fig. 3 Use of absolute camera calibration to project real-world quantities, such as sun position and horizon line, into a video image

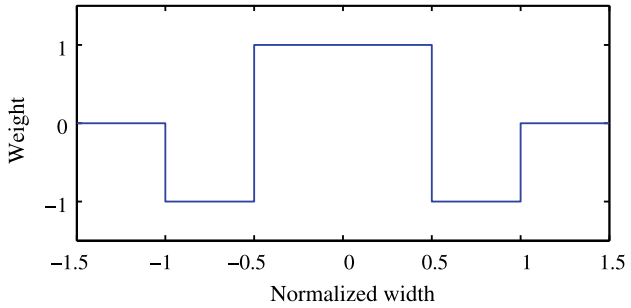


Fig. 4 The shape of the one-dimensional kernel used for matching road paint. By applying this kernel horizontally we detect vertical lines and vice versa. The kernel is scaled to the expected width of a line marking at a given image row and sampled according to the pixel grid

– The horizon line is projected into each image frame. Only pixel rows below the projected horizon are considered for further processing, thus both enhancing runtime efficiency and suppressing potential false positives arising from sky texture.
– Our lidar-based obstacle detector supplies real-time information about the locations of large objects near the vehicle. The detector makes use of relative sensor and vehicle poses to aggregate 3D point data into a common coordinate system, and to produce final measurements in the local reference frame.
– Detected obstacles are projected into the image, and their extents masked, as part of the paint detection algorithms, an important step in reducing false positives.
– Inertial data allows us to project the expected location of the ground plane into the image, providing useful priors and real-world (rather than image-relative) parameters for the vision-based paint detection algorithms.
– Precise knowledge of the date, time, and Earth-relative vehicle pose allow computation of the solar ephemeris; line estimates that point toward the sun in image coordinates are then removed. Others have used a similar approach for shadow prediction (Rasmussen 2008); we have found it successful for preventing spurious paint detections arising from lens flare.
– All detections can be situated in a common local reference frame for meaningful fusion by the higher-level lane centerline estimation stage.

2.1.2 Road paint from matched filters

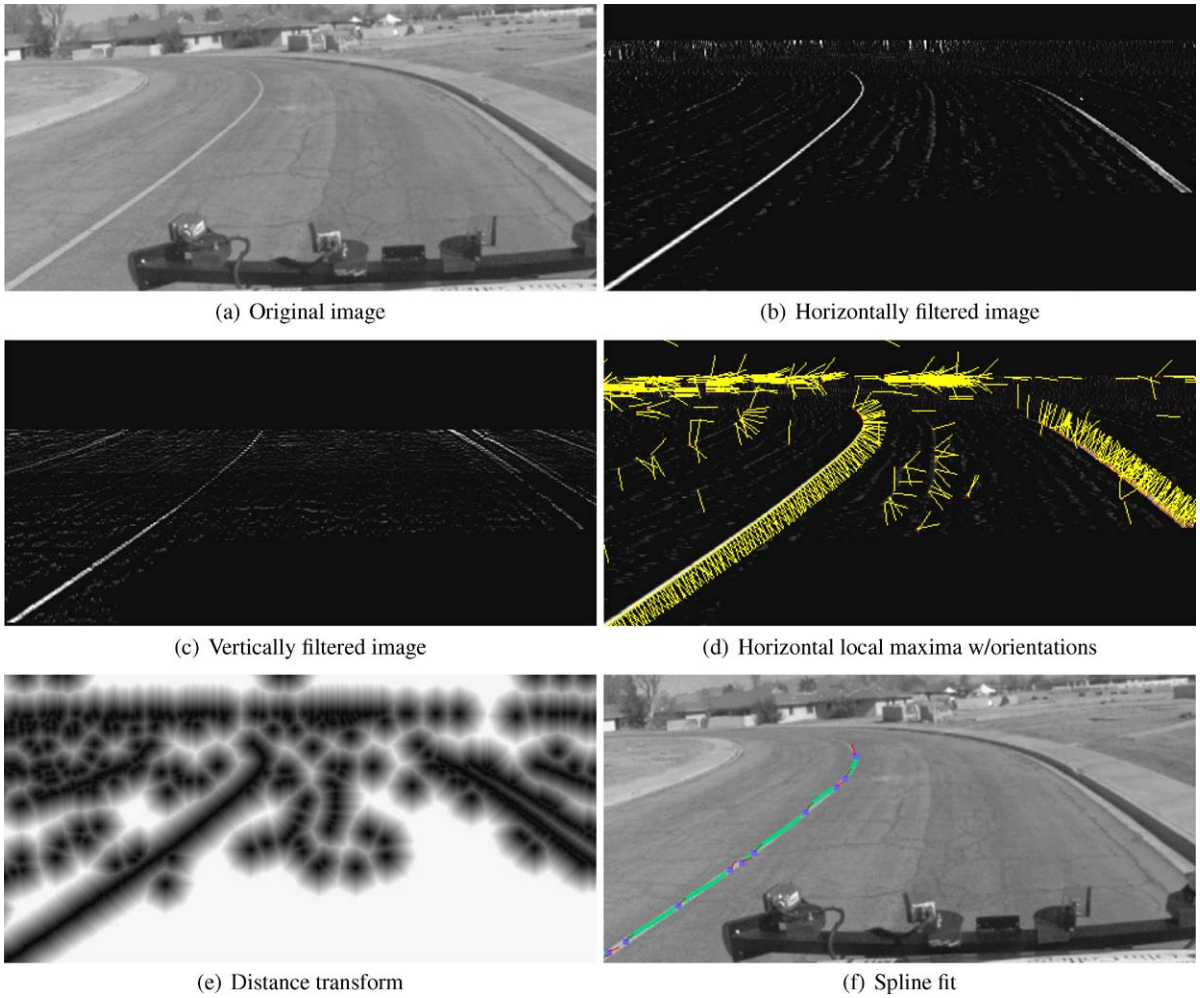
Our first image processing pipeline begins by constructing a one-dimensional matched filter for each row of the input image, such that the width of the filter is the expected width of a painted line marking (e.g. 10 cm) when projected into image coordinates. Each row must have its own filter width because line widths appear smaller as they come closer to the horizon. In addition, horizontal and vertical lines in the image are detected by constructing separate kernels, one of which convolves across the vertical dimension of the image and one across the horizontal dimension. The shape of each kernel is shown in Fig. 4. The width of the kernel's support (the portion of the kernel with non-zero weight) is a trade-off between noise tolerance and the ability to detect closely spaced lines. We chose the support so that double-yellow lines in the center of the road are detected successfully.

Our matched filter approach is similar to that of McCall and Trivedi, who used steerable filters (McCall and Trivedi 2006). Our fixed vertical and horizontal kernels are approximations that have the advantage of executing faster and the disadvantage of being less sensitive to certain line orientations.

For each video frame, the kernel is sampled along the pixel grid at each row according to the projection of the ground plane inferred from live IMU data. The kernels are then convolved with the image data from each row to produce the output of the matched filter. Convolution computation is suppressed where the kernel width is less than one pixel. As shown in Fig. 5, this operation successfully discards most of the clutter in the scene and produces a strong response along line-like features. This is done separately for the vertical and horizontal kernels, giving two output images (Figs. 5b, c).

Next, we iterate over each row of the horizontal filter output and each column of the vertical filter output to build a list of one-dimensional local maxima which will serve as features. Ideally, these maxima occur at the center of any painted lines, although they also occur due to noise and other spurious detections. We reject maxima with a magnitude less than 4% of the maximum possible magnitude, a threshold that was tuned manually to reject maxima occurring in low-contrast image regions.

For each feature, we compute the orientation of the underlying line by finding the direction of principal curvature. At the center of a road paint line, the second derivative of filter response will be large and positive in the direction perpendicular to the line. Parallel to the line, the second deriv-



(a) Original image (b) Horizontally filtered image

(c) Vertically filtered image (d) Horizontal local maxima w/orientations

(e) Distance transform (f) Spline fit

Fig. 5 Our first road paint detector: (**a**) The original image is (**b**) convolved with a horizontal matched filter at each row and (**c**) a vertical filter. (**d**) Local maxima in the horizontal filter response are enumerated and their dominant orientations computed, depicted by the perpendicular to each maximum. (**e**) A distance transform describing the shortest distance from each pixel to the local maxima is used to guide a spline search that (**f**) connects nearby maxima into cubic Hermite splines

ative will be near zero. Thus, we first compute the Hessian, the 2×2 matrix of second derivatives

$$H = \begin{bmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy} \end{bmatrix} \tag{1}$$

where F is the image of filter responses. The second derivatives are computed with 3×3 Sobel kernels. The largest eigenvalue of H is the principal curvature, and its corresponding eigenvector is the direction of that curvature. We attach that direction to the feature as the perpendicular of the underlying line (Fig. 5d).

Once the list of features is generated, we compute a distance transform of the image, such that the intensity at each pixel of the distance transform is proportional to the Euclidean distance from that pixel to the nearest feature (Fig. 5e).

We use cubic Hermite splines to connect the features into continuous curves that represent the underlying lane markings. The goal is to construct splines with approximately 50 pixels between control points. This spacing allows the splines to have relatively few parameters yet still follow the sometimes erratic curves present in urban driving situations. A cubic Hermite spline is parameterized as

$$\begin{aligned} \mathbf{p}(t) = {} & (2t^3 - 3t^2 + 1)\mathbf{p}_0 + (t^3 - 2t^2 + t)h\mathbf{m}_0 \\ & + (-2t^3 + 3t^2)\mathbf{p}_1 + (t^3 - t^2)h\mathbf{m}_1 \end{aligned} \tag{2}$$

where $t \in [0, 1]$ and $\mathbf{p}_0$ and $\mathbf{p}_1$ are a pair of neighboring control points (Bartels and Beatty 1987). This parameterization ensures that the tangents $\mathbf{m}_0$ and $\mathbf{m}_1$ are continuous between pairs of control points. The scale factor h is used to scale the tangent vectors to the appropriate magnitude. We use $h = \|\mathbf{p}_0 - \mathbf{p}_1\|$. When generating splines, we use features extracted above directly as control points and the extracted perpendicular vectors as the tangents (after rotat-

ing them 90 degrees to orient them in the “forward" spline direction).

We now describe our algorithm for fitting splines to the features. First, the algorithm selects 100 “seed” features near the bottom of the image, since features near the bottom are closer to the camera and more well-defined. We then consider every feature further than 50 pixels but closer than 60 pixels away from the starting feature. Any features in this annular region are candidates for the second control point of a spline that starts at the seed feature. For each candidate, we sample a spline from the seed point to the candidate point using equation (2) and sum the squared values of the distance transform along the sampled spline. The candidate with the smallest sum is selected as the next control point. This candidate is now the new seed and the search continues with a new annulus centered at that point. This “greedy” search for an extension of the spline terminates when the average value of the distance transform along the new portion of the spline is larger than 3 pixels. Additional splines are found in the same way until either a pre-defined number of splines is reached (we use 20) or no additional splines can be found. After each spline is found, its constituent features are removed from the global feature list and the distance transform recomputed so that the same spline is not matched twice.

The sensitivity of the spline finder is tuned using a threshold on spline score. A spline’s score is computed as the average squared distance of the spline from features along its path, with smaller scores indicating better matches. A bonus is also assigned to longer splines with more control points. This bonus encourages splines that extend toward the horizon, where line features are weaker and splines might otherwise be rejected. In our system, we tuned this threshold toward extra false positives so that higher layers would have more true positives with which to work. If this road paint detector were to be used directly by a navigation system, it could be tuned to instead reduce false positives at the cost of reduced sensitivity.

2.1.3 *Road paint from symmetric contours*

The second road paint detection mechanism employed in our system relies on more traditional low-level image processing. In order to maximize frame throughput, and thus reduce the time between successive inputs to the lane fusion and tracking components, we designed the module to utilize fairly simple and easily-vectorized image operations.

The central observation behind this detector is that image features of interest—namely lines corresponding to road paint—typically consist of well-defined, elongated, continuous regions that are brighter than their surround. While this characterization excludes circular reflectors and dark-on-light road markings, it does encompass solid and dashed lane boundaries, stop lines, crosswalks, white and yellow

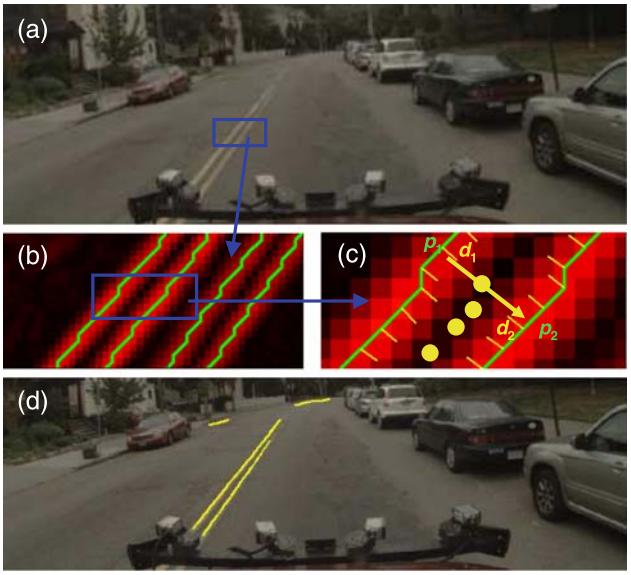


Fig. 6 Progression from (**a**) original image through (**b**) smoothed gradients (*red*), border contours (*green*), and symmetric contour pairs (*yellow*) to form (**c**) a paint line candidate. Final line detections are shown in (**d**) the bottom image

paint on road pavements of various types, and markings seen through cast shadows across the road surface. Thus, our strategy is to first detect the potential boundaries of road paint using spatial gradient operators, then estimate the desired line centers by searching for boundaries that enclose a brighter region; that is, boundary pairs which are proximal and roughly parallel in world space and whose local gradients point toward each other (Fig. 6).

Our approach is quite flexible and robust to many conditions, including several potential shortcomings identified in other road paint extraction algorithms (McCall and Trivedi 2006). Most extraneous image lines are rejected by the symmetric dark-light-dark assumption, metric width and length thresholds, and curvature constraints; straight and curved segments observed from any perspective are handled uniformly, unlike template-based (Taylor et al. 1999; Pomerleau and Jochem 1996) or frequency-based (Kreucher and Lakshmanan 1999) techniques; and features are reliably extracted even under variations in road texture and scene illumination, unlike intensity analysis techniques (Arbib and Pomerleau 1995; Baluja 1996).

The contour-based road line detector consists of three steps: low-level image processing to detect raw features; contour extraction to produce initial line candidates; and contour post-processing for smoothing and false positive reduction. The first step applies local lowpass and derivative operators to produce the noise-suppressed direction and magnitude of the raw grayscale image’s spatial gradients. A loose threshold is applied to the gradient magnitude to reject extremely weak, unreliable edge responses aris-

ing from low-contrast regions while preserving all potential edges of interest. The resulting image undergoes non-maximal suppression in the gradient direction to dramatically reduce extraneous pixels without explicit thresholds; the result is a sparse feature mask image, with a gradient direction and magnitude associated with every valid pixel. As with other edge-based methods (Dickmanns and Mysliwetz 1992; Kang and Jung 2003; Kreucher et al. 1998), the use of spatial gradients and data-relative local acceptance thresholds provides a degree of robustness to commonly observed conditions such as shadowing, low contrast road paint, and variable road pavement texture.

In the second step, a connected components algorithm iteratively walks the feature mask to generate smooth contours of ordered points, broken at discontinuities in location and gradient direction. This results in a new image whose pixel values indicate the identities and positions of the detected contours, which in turn represent candidate road paint boundaries. While the downstream fusion algorithm could make direct use of these raw boundaries, two problems immediately become apparent: true road paint markings will exhibit separate "double" contours, one on either side of a given painted line, representing the dark-to-light and light-to-dark transitions; and many detected contours may correspond to undesirable intensity edges observed, for example, due to hard shadow lines or changes in road material. Therefore, at this stage we enforce the constraint that lines of interest are thin, elongated, light-on-dark regions whose boundaries are parallel in metric coordinates. This constraint precludes detection of dark-on-light road markings and small features such as circular reflectors, and substantially reduces false detections and confusion conditions arising from commonly occurring visual phenomena (McCall and Trivedi 2006).

In order to localize the desired centerlines between detected double-boundaries, we apply a second iterative walk to the contour image described above. At each boundary pixel p_i, traversed in contour order, the algorithm extends a virtual line in the direction of the local gradient d_i until it meets a distinct contour at p_j (Fig. 6c). If the gradient of the second contour d_j points in a direction opposite d_i, and if the metric distance between p_i and p_j is within pre-defined limits corresponding to the expected width of painted lines, then the midpoint of p_i and p_j is added to a growing centerline curve. Many non-paint contours (e.g., those with only one edge or wrong width) are thus removed from consideration.

At this stage our detection algorithm has constructed a set of road paint line candidates, each of which is brighter than its surround; however, this candidate set may be corrupted by undesirable line fragments and outliers. The third and final step of the algorithm therefore applies a series of higher level post-processing operations to produce smooth, high-confidence line estimates for consumption by subsequent data fusion and lane estimation stages. We first merge any contour fragments whose union produces a smooth curve (i.e. does not introduce discontinuities or high curvature); unlike other methods (Lee 2002; Lee and Yi 2005; Kang and Jung 2003), we do not enforce straight line constraints. Next, we fit parabolic arcs to the merged curves and recursively break them at points of high deviation. Finally, all curves shorter than a given threshold length (in pixels and in metric units) are removed before the final image-relative road paint lines are produced. As with the first road paint detection algorithm, these are inverse-perspective mapped and projected onto the ground plane before further processing.

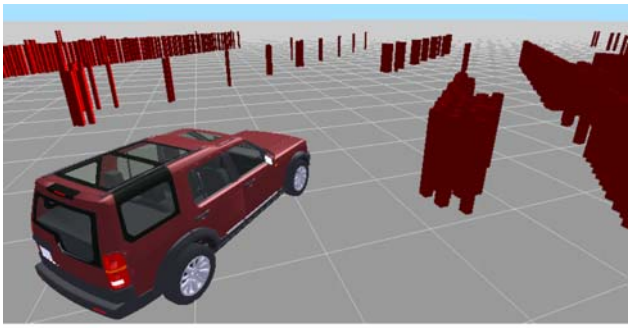
In practice, the primary differences between the two road paint detection algorithms we employ lie in sensitivity and speed. The contour-based detector tends to estimate smoother curves due to its parabolic curve model; the gradient-based detector is able to more accurately capture the geometry of non-parabolic curves. The variable width filter kernels of the gradient-based detector give it a range advantage, allow it to more reliably detect road paint in image regions where the road paint spans only a few image pixels. Lastly, fitting parabolic arcs was faster than the spline search, which allowed the contour-based detector to operate at a higher framerate.

2.1.4 Reducing false positives with obstacle masking

Many world objects exhibit striped appearances that mimic the painted lane boundaries of interest, leading to incorrect detection candidates at early stages of processing. Many such candidates are eliminated via subsequent curvature constraints, spline fitting, and projected length filters, but even with these measures in place, some problematic false positives can still occur due to objects such as fences, guard rails, side trim, vehicle fenders, and buildings with horizontal patterns.

For our vehicle, we developed a lidar-based obstacle detection system whose primary purpose was to ensure that our vehicle avoided collisions with other cars and obstacles. Many objects that can generate false lane detections (such as guard rails) are easily detected by this lidar-based obstacle detector. Since true road paint markings occur only on flat (obstacle-free) ground, the detection of an obstacle implies that any lane detections directly under or near that obstacle are incorrect. Further, since the body-relative 6-DOF poses of all our sensors are known, we can project all 3D obstacle detections into the 2D pixel coordinates of each camera (Fig. 7b). These projections are used to mask corresponding regions in the camera images, explicitly suppressing road lines detected in those regions (Fig. 7c).

The lidar-based obstacle detector is described in detail in a separate paper (Leonard et al. 2008). Briefly, the obstacle detection system relied on a heterogeneous collection of



(a) Lidar-identified obstacles

(b) Obstacles projected into image

(c) Exclusion mask

Fig. 7 (**a**) Lidar identified obstacles. (**b**) Obstacles projected into an image. (**c**) Mask (*grey*) created from horizon line and obstacles. Road paint detections within this mask are discarded

lidars affording a 360-degree field of view. A Velodyne lidar, containing 64 independent lasers, served as the primary obstacle detection sensor. The Velodyne produces a million point samples per second, providing nearly full 3D coverage. Obstacles were detected by grouping lidar returns over a polar grid aligned with the ground plane. If the heights of lidar returns within a single grid cell exhibited significant variation (allowing for outliers), a vertical obstacle was reported within that cell. Additionally, seven SICK lidars formed a horizontal "skirt" around the vehicle, augmenting the obstacles detected by the Velodyne. The SICK sensors served two roles: they filled in the Velodyne's blind spots and served as redundant sensors in the event that the Velodyne failed.

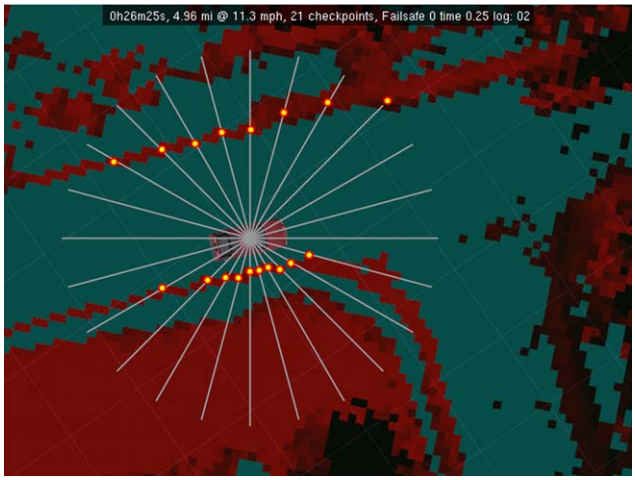


Fig. 8 Road boundaries from lidar. From lidar data, our algorithms are able to detect berms and curbs that typically indicate road boundaries. These boundaries are found by casting rays from the vehicle position: the first transition from smooth surface to rough surface serves as a point detection of the road boundary. Splines are then fit through the detected points in order to yield the road-boundary estimate

2.1.5 Road boundaries from lidar

In addition to detecting large obstacles like guard rails and other cars, the lidar subsystem can detect smaller hazards such as the berms and curbs that often delineate the boundaries of a road. These detections provide evidence that can be fused into the lane centerline estimator to better localize lanes, and in fact represent complementary features that can be used to identify the shape of the road even in the absence of painted lines. We briefly summarize the road boundary detection system here; more detail can be found in a separate publication (Leonard et al. 2008).

Both the Velodyne lidar and the SICK lidars are used to detect road boundaries. The "roughness" of a particular patch of terrain can be estimated by looking for large changes in elevation over small translational changes. These slopes are collected into a 2D array, such that the value of each cell in the array corresponds to the observed roughness of some patch of ground. This resulting roughness map is illustrated by the red areas in Fig. 8. A complication arises due to moving obstacles: the presence of a large vertical discontinuity, perhaps arising from the car ahead, does not indicate a road boundary. We reject these false positives using short-term memory: if a given cell is ever observed to be "smooth" (i.e., part of the road), then any detections of a vertical discontinuity in that cell are ignored. The entire ground surface will thus eventually be observed as transient objects move to uncover it.

From the "roughness map", we detect road boundaries by casting rays from a point near the vehicle, which we assume is on the road. The first transition from smooth to rough is recorded along each ray, forming a set of road-boundary

point detections (see Fig. 8). Much like maximal filter responses in the camera road paint detectors, these point detections are prone to false positives. However, by fitting splines through the points and rejecting those splines that do not match a model of road boundaries, the false positive rate is reduced to an acceptable (very low) level.

The resulting road boundary detections are used as evidence and incorporated into an evidence grid, a process discussed in the following section. When road paint detection fails (due to absent or difficult-to-detect road paint), our road-tracking system relies solely on lidar-derived road boundaries in order to stay on the road.

2.2 Lane centerline estimation

The second stage of lane finding estimates the geometry of nearby lanes using a weighted set of recent road paint and curb detections, both of which are represented as piecewise linear curves. To simplify the process, we estimate only lane centerlines, which we model as locally parabolic segments. While urban roads are not designed to be parabolic, this representation is generally accurate for stretches of road that lie within sensor range (about 50 m in our case).

Lanes centerlines are estimated in two steps. First, a centerline evidence image D is constructed, where the value $D(\mathbf{p})$ of each image pixel corresponds to the evidence that a point $\mathbf{p} = [p_x, p_y]$ in the local coordinate frame lies on a lane center. Second, parabolic segments are fit to the ridges in D and evaluated as lane centerline candidates.

2.2.1 Centerline evidence image

To construct D, road paint and curb detections are used to increase or decrease the values of pixels in the image, and are weighted according to their age (older detections are given less weight). The value of D at a pixel corresponding to the point $\mathbf{p}$ is computed as the weighted sum of the influences of each road paint and curb detection d_i at the point $\mathbf{p}$:

$$D(\mathbf{p}) = \sum_i e^{-a(d_i)\lambda} g(d_i, \mathbf{p})$$

where $a(d_i)$ denotes how much time has passed since d_i was first detected, λ is a decay constant, and $g(d_i, \mathbf{p})$ is the influence of d_i at $\mathbf{p}$. We chose $\lambda = 0.7$.

Before describing how the influence is determined, we make three observations. First, a lane is most likely to be centered half a lane width from a strip of road paint or a curb. Second, 88% of federally managed lanes in the US are between 3.05 m and 3.66 m wide (Highway Statistics 2005). Third, a curb gives us different information about the presence of a lane than does road paint. From these observations and the characteristics of our road paint and curb detectors, we define two functions $f_{rp}(x)$ and $f_{cb}(x)$, where x is the Euclidean distance from d_i to $\mathbf{p}$:

$$f_{rp}(x) = -e^{-\frac{x^2}{0.42}} + e^{-\frac{(x-1.83)^2}{0.14}} \tag{3}$$

$$f_{cb}(x) = -e^{-\frac{x^2}{0.42}} \tag{4}$$

The functions f_{rp} and f_{cb} are intermediate functions used to compute the influence of road paint and curb detections, respectively, on D. f_{rp} is chosen to have a minimum at $x = 0$, and a maximum at one half lane width (1.83 m). f_{cb} is always negative, indicating that curb detections are used only to decrease the evidence for a lane centerline. We elected this policy due to our curb detector's occasional detection of curb-like features where no curbs were present. Let $\mathbf{c}$ indicate the closest point on d_i to $\mathbf{p}$. The actual influence of a detection is computed as:

$$g(d_i, \mathbf{p}) = \begin{cases} 0 & \text{if } \mathbf{c} \text{ is an endpoint of } d_i, \text{ else} \\ f_{rp}(\|\mathbf{p} - \mathbf{c}\|) & \text{if } d_i \text{ is road paint, else} \\ f_{cb}(\|\mathbf{p} - \mathbf{c}\|) & \text{if } d_i \text{ is a curb} \end{cases}$$

This last condition is introduced because road paint and curbs are observed only in short sections. The effect is that a detection influences only those centerline evidence values immediately next to it, and not in front of or behind it.

In practice, D can be initialized once and updated incrementally by adding the influences of newly received detections and applying an exponential time decay at each update. Additionally, we improve the system's ability to detect lanes with dashed boundaries by injecting imaginary road paint detections connecting two separate road paint detections when they are physically proximate and collinear.

2.2.2 Parabola fitting

Once the centerline evidence image D has been constructed, the set R of ridge points is identified by scanning D for points that are local maxima along either a row or a column, and whose value exceeds a minimum threshold. Next, a random sample consensus (RANSAC) algorithm (Fischler and Bolles 1981) fits parabolic segments to the ridge points. At each RANSAC iteration, three ridge points are randomly selected for a three-point parabola fit. The directrix of the parabola is chosen to be the first principal component of the three points.

To determine the set of inliers for a parabola, we first compute its conic coefficient matrix $\mathbf{C}$ (Hartley and Zisserman 2001), and define the set of candidate inliers L to contain the ridge points within some algebraic distance α of $\mathbf{C}$.

$$L = \{\mathbf{p} \in R : \mathbf{p}^T \mathbf{C} \mathbf{p} < \alpha\}$$

For our experiments, we chose $\alpha = 1$ m. The parabola is then re-fit to L using a linear least squares method, and

Fig. 9 The second stage of our system constructs a centerline evidence image. Lane centerline candidates (*blue*) are identified by fitting parabolic segments to the ridges of the image. Front-center camera view is shown in *top left* for context

a new set of candidate inliers is computed. Next, the candidate inliers are partitioned into connected components, where a ridge point is connected to all neighboring ridge points within a 1 meter radius. The set of ridge points in the largest component is chosen as the set of actual inliers for the parabola. The purpose of this partitioning step is to ensure that a parabola cannot be fit across multiple ridges, and requires that an entire identified ridge be connected. Finally, a score s for the entire parabola is computed.

$$s = \sum_{\mathbf{p} \in L} \frac{1}{1 + \mathbf{p}^T \mathbf{C} \mathbf{p}}$$

The contribution of an inlier to the total parabola score is inversely related to the inlier's algebraic distance, with each inlier contributing a minimum amount to the score. The overall result is that parabolas with many good inliers have the greatest score. If the score of a parabola is below some threshold, it is discarded. Experimentation with different values yielded a useful score threshold of 140.

After a number of RANSAC iterations (we found 200 to be sufficient), the parabola with greatest score is selected as a candidate lane centerline. Its inliers are removed from the set of ridge points, and all remaining parabolas are re-fit and re-scored using the reduced set of ridge points. The next best-scoring parabola is chosen, and this process is repeated to produce at most 5 candidate lane centerlines (Fig. 9). Each candidate lane centerline is then discretized into a piecewise linear curve and transmitted to the lane tracker for further processing. Figure 10b shows three such candidates.

2.3 Lane tracking

The primary purpose of the lane tracker is to maintain a stateful, smoothly time-varying estimate of the nearby lanes of travel. To do so, it uses both the candidate lane centerlines produced by the centerline estimator and an *a priori* estimate derived from the provided road map.

(a) Two RNDF-derived lane centerline priors

(b) Candidate lane centerlines estimated from sensor data

(c) Filtered and tracked lane centerlines

Fig. 10 (**a**) The RNDF provides weak *a priori* lane centerline estimates (*white*) that may go off-road, e.g. through trees and bushes. (**b**) On-board sensors are used to detect obstacles, road paint, and curbs, which are in turn used to estimate lanes of travel, modeled as parabolic segments (*blue*). (**c**) The sensor-derived estimates are then filtered, tracked, and fused with the RNDF priors

In the context of the Urban Challenge, the road map was provided as a Route Network Description File (RNDF) (2007). The RNDF can be thought of as a directed graph, where each node is a waypoint in the center of a lane, and edges represent intersections and lanes of travel. Waypoints are given as GPS coordinates, and can be separated by arbitrary distances; a simple linear interpolation of connected waypoints may go off road, e.g. through trees and houses. In our system, the RNDF was treated as a strong prior on the number of lanes, and a weak prior on lane geometry.

As our vehicle travels, it constructs and maintains representations of all portions of all lanes within a fixed radius of 75 m. When the vehicle nears an RNDF waypoint and does not already have an estimate for the waypoint's lane, a new lane estimate is instantiated and extended to the immediate neighbors of the waypoint. The lane estimate is extended and truncated as the vehicle approaches and withdraws from waypoints in the lane.

The centerline of each lane is modeled as a piecewise linear curve, with control points spaced approximately every 2 m. Each control point is given a scalar confidence value

indicating the certainty of the lane tracker's estimate at that point. The lane tracker decays the confidence of a control point as the vehicle travels, and increases it either by detecting proximity to an RNDF waypoint or by updating control points with centerline estimates from the second stage.

As centerline candidates are generated, the lane tracker attempts to match each candidate with a tracked lane. If matching is successful, the centerline candidate is used to update the lane estimate. To determine if a candidate c is a good match for a tracked lane l, the longest segment s_c of the candidate is identified such that every point on s_c is within some maximum distance τ of l. We then define the match score $m(c, l)$ as:

$$m(c,l) = \int_{s_c} 1 + \frac{\tau - d(s_c(x), l)}{\tau} dx$$

where $d(\mathbf{p}, l)$ is the distance from a point $\mathbf{p}$ to the lane l. Intuitively, if s_c is sufficiently long and close to this estimate, it is considered a good match. We designed the matching function to rely only on the closest segment of the candidate, and not on the entire candidate, based on the premise that as the vehicle travels, the portions of a lane that it observes vary smoothly over time, and previously unobserved portions should not adversely affect the matching provided that sufficient overlap is observed elsewhere.

Once a centerline candidate has been matched to a tracked lane, it is used to update the lane estimates by mapping control points on the tracked lane to the candidate, with an exponential moving average applied for temporal smoothing. Figure 10 illustrates this process. After a centerline candidate has been used to update a tracked lane estimate, it is not re-used. At each update, the confidence values of control points updated from a matching are increased, and others are decreased. If the confidence value of a control point decreases below some threshold, its position is discarded and recomputed as a linear interpolation of its closest surrounding confident control points.

3 Urban challenge results

The most difficult part of evaluating a lane detection and tracking system for autonomous vehicle operation often lies in finding a suitable test environment. Legal, financial, and logistical constraints proved to be a significant hurdle in this process. We were fortunate to have the opportunity to conduct an extensive test in the 2007 DARPA Urban Challenge, which provided a large-scale real-world environment with a wide variety of roads. Both the type and quality of roads varied significantly across the race, from well-marked urban streets, to steep unpaved dirt roads, to a 1.6 km stretch of highway. Throughout the race, approximately 50 human-driven and autonomous vehicles were simultaneously active, thus providing realistic traffic scenarios.

Our most significant result is that our lane detection and tracking system successfully guided our vehicle through a 90 km course in a single day, at speeds up to 40 km/h, with an average speed of 16 km/h. A post-race inspection of our log files revealed that at almost no time did our vehicle have a lane centerline estimate more than half a lane width from the actual lane centerline, and at no time did it unintentionally enter or exit a travel lane. We note that the output of the lane tracking system was used directly to guide the navigation and motion planning systems; had the lane tracking system yielded an incorrect estimate, our vehicle would have traveled along that estimate, possibly into an oncoming traffic lane or off-road.

3.1 System confidence

We wished to determine how much our system relied on perceptually-derived lane estimates, and how much it relied on prior knowledge of the road as given in the RNDF. To answer this, we examined the distance the vehicle traveled with high confidence visually-derived lane estimates, excluding control points where high confidence resulted from proximity to an RNDF waypoint.

At any instant, our system can either have no confidence in its visual estimates of the current travel lane, or confidence out to a certain distance a in front of the vehicle. If the vehicle then travels d meters while maintaining the same confidence in its visual estimates, then we say that the system had a high-confidence estimate a meters in front of the vehicle for d meters of travel. Computing a for all 90 km of the race allows us to answer the question of how far out our system could typically "see" (Table 1). From this, we see that our vehicle maintained high confidence visual estimates 1 m or more ahead for 56.8 km, or 65.2% of the total distance traveled. In the remaining portion, the lane tracker relied on an interpolation of its most recent high-confidence estimates.

Table 1 Distance traveled with high-confidence visual estimates in current lane of travel (total distance = 87 km)

Visual range (m)	Distance traveled (km)	
0	30.3	(34.8%)
1–10	10.8	(12.4%)
11–20	24.6	(28.2%)
21–30	15.7	(18.0%)
31–40	4.2	(4.8%)
41–50	1.3	(1.5%)
>50	0.2	(0.2%)

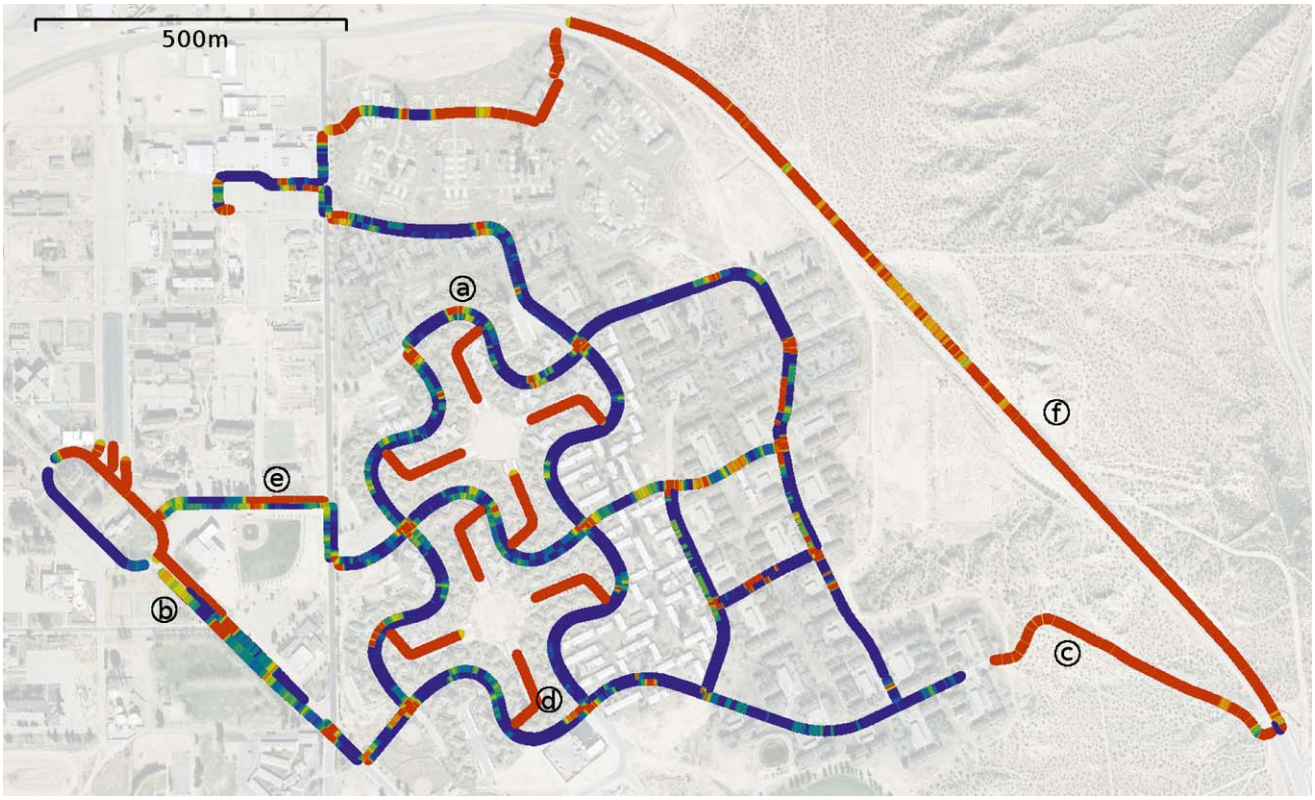


Fig. 11 Aerial view of the Urban Challenge race course in Victorville, CA. Autonomously traversed roads are colored blue in areas where the lane tracking system reported high confidence, and red in areas of low confidence. Some low-confidence cases are expected, such as at intersections and areas with no clear lane markings. Failure modes occurring at the *circled letters* are described in Fig. 12

A second way of assessing the system's performance is by examining its estimates as a function of location within the course. Figure 11 shows an aerial view of areas visited by our vehicle, colored according to whether or not the vehicle had a high confidence estimate at each point. We note that our system had high-confidence lane estimates throughout the majority of the high-curvature and urban portions of the course. Some of the low-confidence cases are expected, such as when the vehicle is traveling through intersections or along roads with no discernible lane boundaries. In other cases, our system was unable to obtain a high-confidence estimate whereas a human would have had little trouble doing so.

Images from our logged camera images at typical failure cases are shown in Fig. 12 (with locations at which these failures occurred marked in Fig. 11). A common failure mode was an inability to detect road paint in the presence of dramatic lighting variation such as that caused by cast tree shadows. However, we note that in virtually all of these cases our system reported no confidence in its estimates and did not falsely report the presence of a lane.

Another significant failure occurred on the eastern part of the course, with a 0.5 km dirt road followed by a 1.6 km stretch of highway. Our vehicle traversed this path four times, for a total of 8.4 km. The highway was an unexpected failure. The travel lane happened to be very wide; its width did not fit the 3.66 m prior in the centerline estimator, which had trouble constructing a stable centerline evidence image. In addition, most of the highway was uphill and road paint detection projected onto an assumed level ground plane had so much projection error that no stable centerline evidence image was created. Mean vehicle pitch can be seen in Fig. 16. This last problem could have been addressed by actively modeling the ground surface with either vision or LIDAR data.

The final common failure mode occurred in areas with faint or no road paint, such as the dirt road and roads with well defined curbs but no paint markings. Since our system uses road paint as its primary information source, in the absence of road paint it is no surprise that no lane estimate ensues. Other environmental cues such as color and texture may be useful in such situations (Dahlkamp et al. 2006).

3.2 Human-annotated ground truth

For a more objective and quantitative assessment of our system, we compared its output to a human-annotated data set of observable lanes. This data set provides, at every moment of the race, the geometry of nearby travel lanes registered in

Fig. 12 Common failure cases (cf. Fig. 11). The most common failure was in areas with strong tree shadows, as in (**a**) and (**b**). Dirt roads, and those with faint or no road paint (**c**–**e**) were also common causes of failure. In (**f**), a very wide lane and widely-spaced dashed markings were a challenge due to our strong prior on lane width. In each of these failure situations, the system reported no confidence in its visual estimates

the vehicle's local reference frame. We briefly describe its creation here.

Perhaps the simplest method for a person to annotate ground truth is to examine a visualization of the vehicle and its sensor data at each desired moment and mark the nearby travel lanes. While straightforward, this might take several minutes per instance labeled, and would thus not be an efficient or even feasible way to densely label many hours of data.

Instead, we note that over time scales spanning several hours to several days, ground truth lane geometry does not typically change relative to a global reference frame. Our approach is to first produce ground truth lane geometry in a global frame by annotating geo-registered ortho-rectified imagery available on Google Maps. We then use our vehicle's GPS estimates to project ground truth into the vehicle's local reference frame for further analysis. This projection suffers from GPS error, so a manual correction is made when necessary to align the ground truth with observable cues in the sensor data. These corrections are linearly interpolated over time under the premise that the GPS error is fairly continuous. In the course of annotating the 90 km of vehicle travel in our Urban Challenge data set, an average of one correction was made every 45 m.

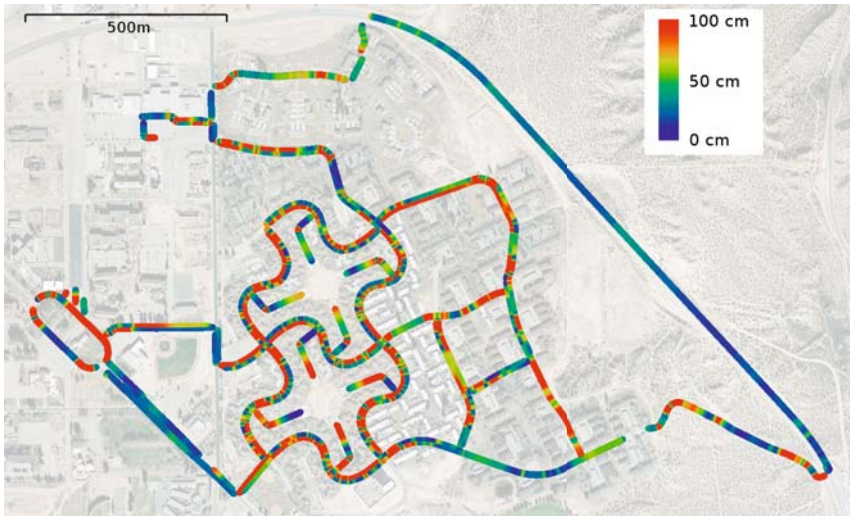
Generating ground truth lane geometry in this manner allows us to conduct a number of experiments that would otherwise be impossible. We can assess the performance of our system under a number of different measures, and see how using it compares to using the RNDF alone without any local perception. Most importantly, ground truth enables use of a quantitative metric with which we can improve and assess future systems.

Lastly, we note that what we are calling "ground truth lane geometry" is perhaps more accurately described as how a human would describe nearby lanes, given a visualization of the vehicle's sensor data. As such, it may be subject to hidden or unknown experimental bias, but we believe it is nevertheless highly useful as is.

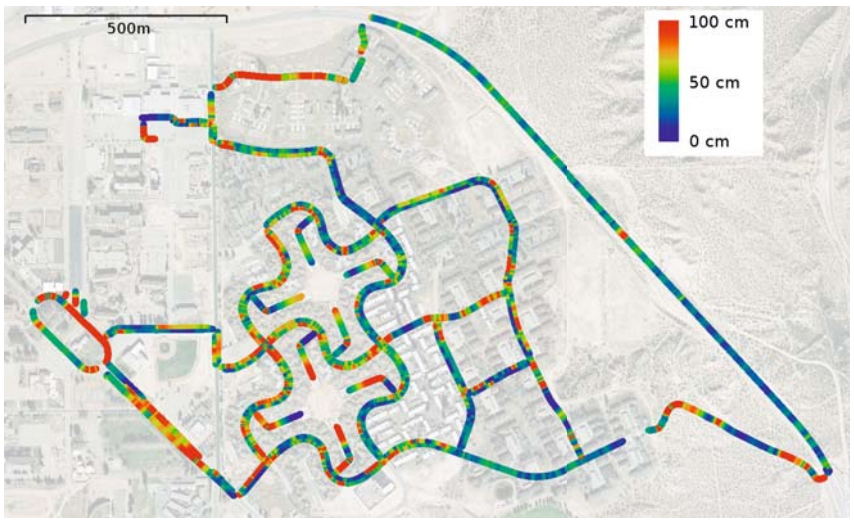
3.3 Centerline error

At each point along the centerline line of a lane estimate, we define its *centerline error* to be the lateral distance from that point to the ground truth lane centerline. In the absence of other obstacles, the vehicle's motion planner typically attempts to track the lane centerline, so the centerline error is a fair measure of the overall system accuracy.

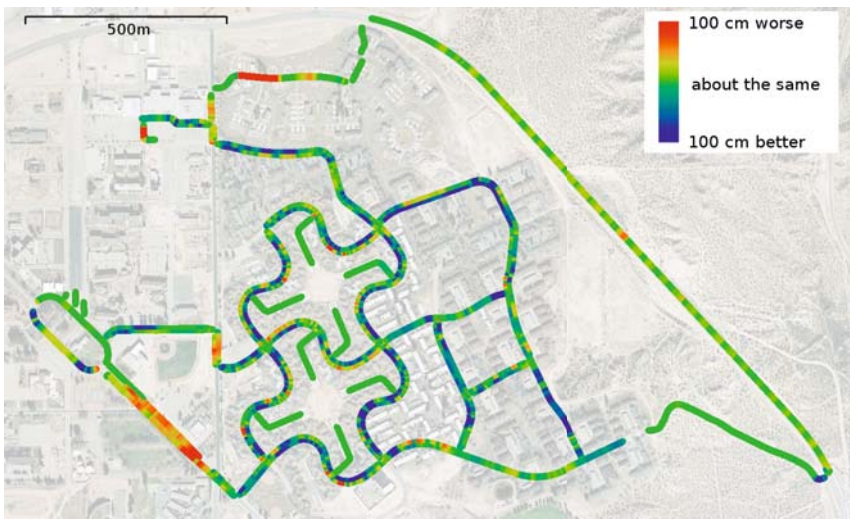
Given that the resolution of our sensors decreases with distance from the vehicle, we expect the accuracy of the lane estimates to decrease the farther away they are. To confirm this, lane estimates were evaluated approximately once per meter of vehicle travel. Centerline error was computed at 1 m intervals on each lane estimate evaluated, starting from 1 m in front of the vehicle up to a distance of 50 m. Lane estimates behind the vehicle were not evaluated. While those estimates tended to be much more accurate, they were no longer useful for forward motion planning.



(a) RNDF Centerline Error



(b) Lane Tracker Centerline Error



(c) Lane Tracker Improvement over RNDF alone

Fig. 13 Mean centerline error 10 m in front of vehicle. (**a**) RNDF error is also dependent on GPS receiver error. (**b**) Lane tracker fuses vision and laser range data to improve RNDF estimates. (**c**) In most cases, our system is as good as or better than using the RNDF alone

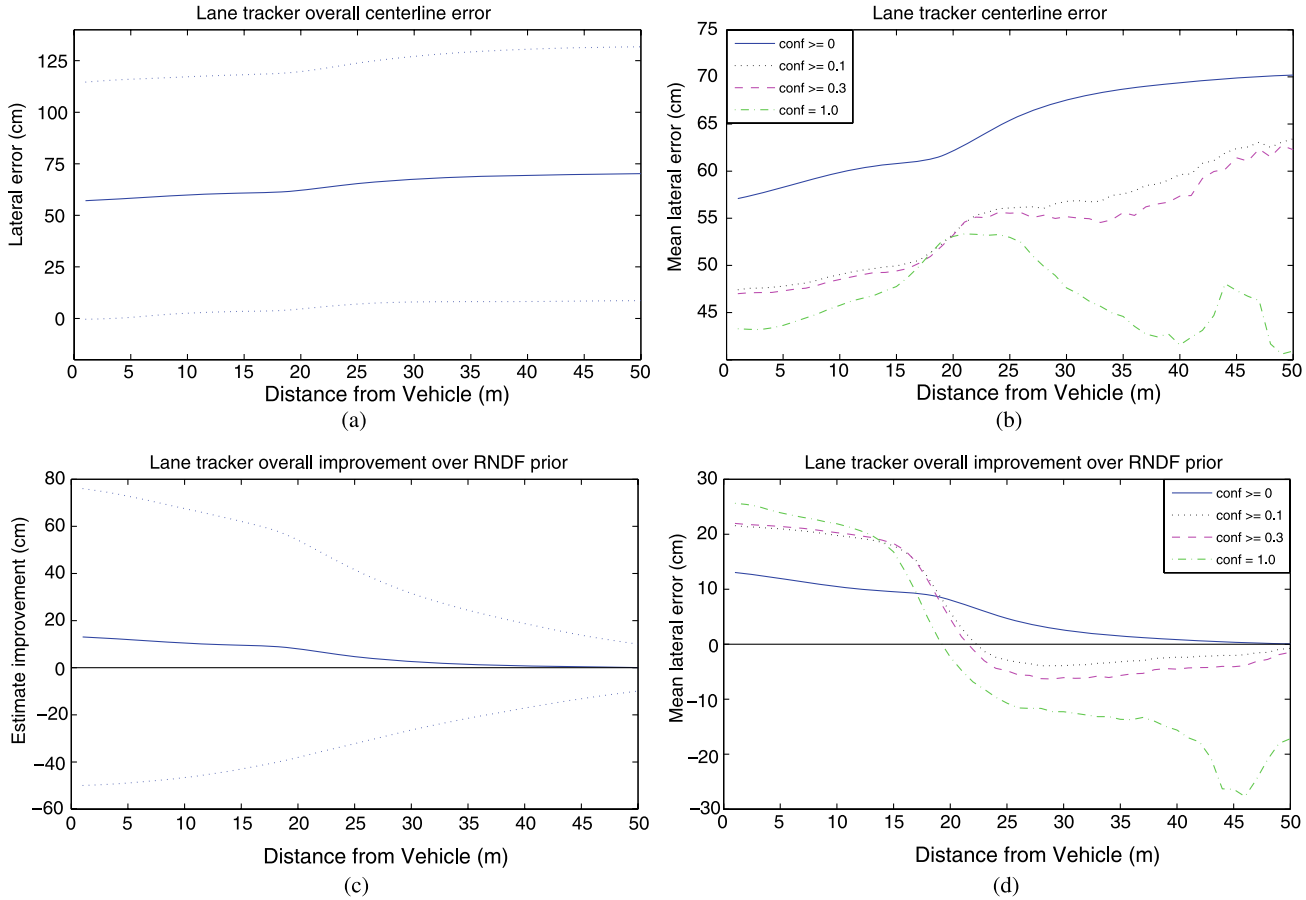


Fig. 14 Centerline error as a function of distance along lane. (**a**) Mean error with 1-σ bounds. (**b**) As system confidence increases, its accuracy improves. (**c**) Mean improvement over the RNDF with 1-σ bounds. (**d**) Mean improvement over the RNDF by system confidence. For high confidence estimates our system outperformed the RNDF only at close ranges

The results of this analysis are shown in Fig. 14. Immediately in front of the vehicle, the mean centerline error was 57 cm, which gradually increased to 70 cm at a distance of 50 m from the vehicle. This reflects all lane estimates produced by our system, including cases where it reported no confidence and was simply interpolating RNDF waypoints. When evaluated in areas of high confidence, we see that the mean error decreased to 40 cm.

To reconcile these error statistics with our earlier claim that our vehicle always remained in its lane, we can examine the width of our vehicle, the centerline error, and the width of lanes on the course. Most lanes on the course were between 4 m and 5 m wide. The roads were built to accommodate vehicles parked on the side, which would normally result in narrower lanes, but in the absence of parked cars, the lanes were effectively much wider. Our vehicle measured 2 m in width, so if it were perfectly centered in a lane, it would have 1–2 m of space to the lane boundary. A mean centerline error of 57 cm reduces this margin to 0.4–1.4 m, which still allows for an imperfect controller. Finally, we also note that the strong prior our system had on lane width gave it a tendency to "lock on" to one lane boundary, and thus produce a centerline estimate that was consistently 1.83 m away from one boundary (our prior on half a lane width), but that did not coincide with the true lane centerline.

The lane centerline error also allows us to answer the question, "Is our system better than simple GPS waypoint interpolation, and by how much?" This is shown in Figs. 14c and d. Overall, the system is modestly better than using the RNDF alone, with a mean improvement of 10 cm at a distance of 10 m from the vehicle. If we again consider only areas with a high confidence level, the mean improvement at 10 m from the vehicle increases to 22 cm. Curiously, at higher confidence levels, the performance of our system relative to the RNDF decreases with distance. While we do expect less improvement at greater distances, we would not normally expect our system to perform worse than the RNDF. To understand why this happens, we next examine the effects of lane curvature on centerline error.

Our method of interpolating GPS waypoints given in the RNDF was simple linear interpolation. As shown in

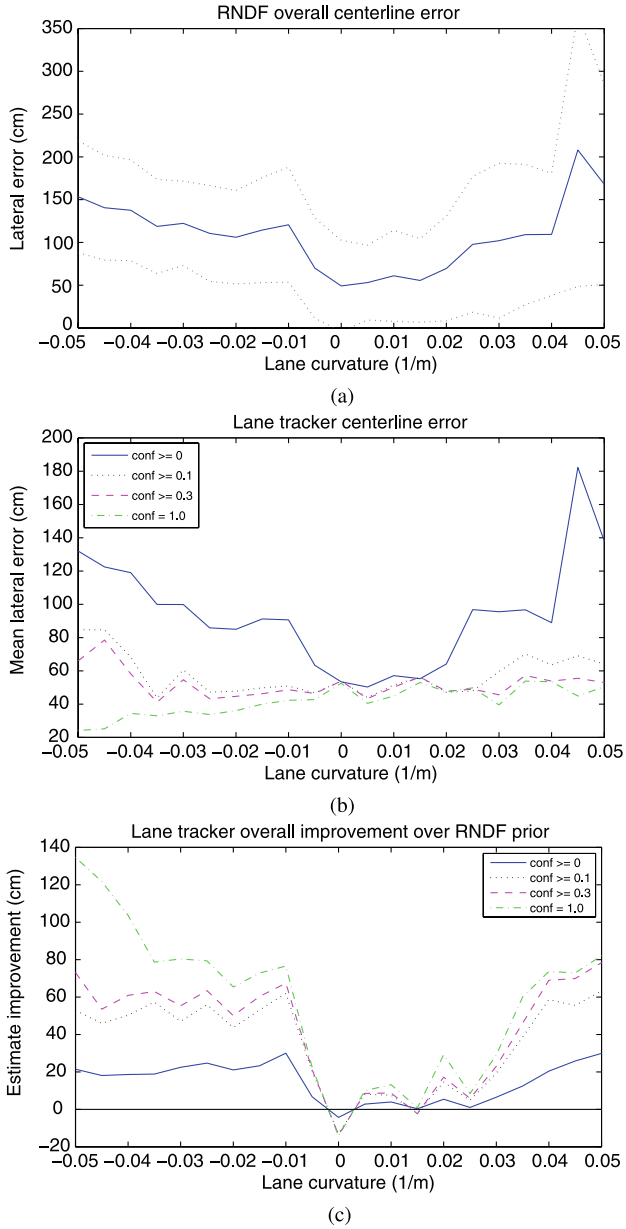


Fig. 15 Mean centerline error as a function of lane curvature. (**a**) The RNDF experiences greater error in high-curvature areas. 1-σ bounds are shown. (**b**) Higher confidence lane estimates are largely able to account for curvature. (**c**) Our system is able to significantly improve RNDF estimates in high-curvature areas

Fig. 15a, the centerline error of this interpolation grows with the magnitude of road curvature. On examining this error, we noticed that the RNDF was significantly more accurate in areas of low curvature, and in some cases had almost no error for long stretches of straight road (cf. Fig. 13a). The distance to which our system reports a confident estimate is also related to lane curvature, as more of a lane is visible for low-curvature roads. Thus, in low-curvature areas, our system served mostly to add noise to already good estimates at distance.

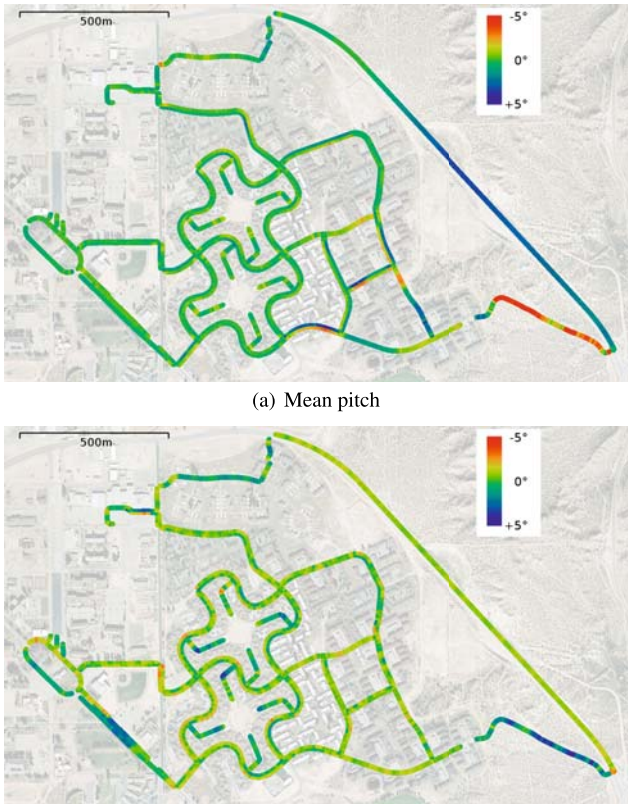
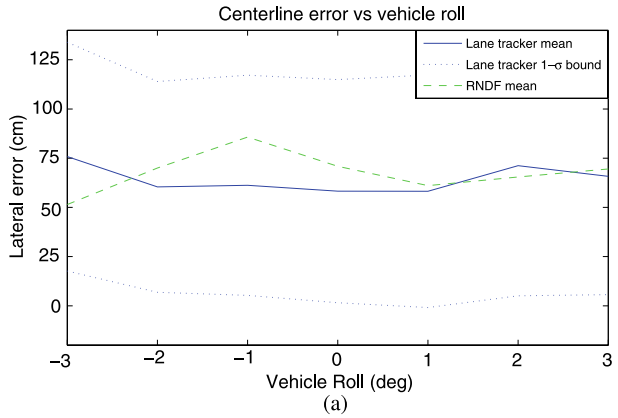


Fig. 16 Mean vehicle pitch and roll throughout the Urban Challenge. Positive pitch and roll values correspond to vehicle nose up and left side up, respectively

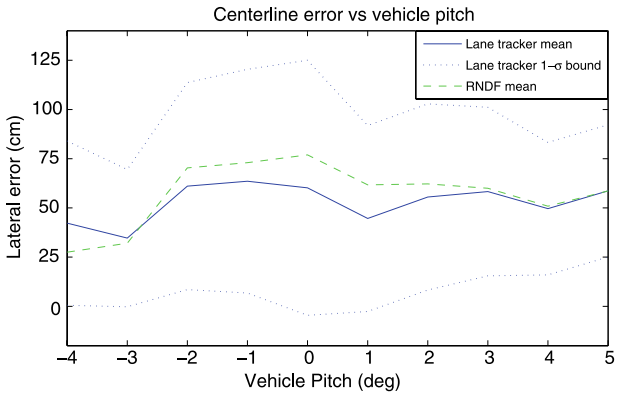
In contrast, our system excelled in areas of high curvature, as shown in Figs. 15b and c. Most of these regions were sufficiently locally parabolic that our model was able to accurately capture the curvature of the roads and significantly improve the RNDF interpolation. For curvatures of 0.05 m^{-1}, mean improvements over the RNDF for high-confidence estimates were almost 1 m.

Lastly, we evaluate lane centerline error against vehicle roll and pitch. Mean vehicle roll and pitch are shown in Fig. 16, and error plots for these factors are given in Fig. 17. Sustained roll and pitch are good indicators of non-level terrain, and we expect worse performance in these cases in light of our level ground-plane assumption. As mentioned earlier, our implementation did not account for these terrain scenarios, but doing so even crudely by estimating a non-level ground plane should help.

Figures 17 show a slight increase in overall centerline error as roll increases, but not so for pitch. However, we note that the RNDF error is inversely correlated with pitch and roll, suggesting that the course was more accurately marked in areas with non-level terrain. When compared together, we can see that the performance of our system relative to the RNDF decreases moderately as roll and pitch increase.

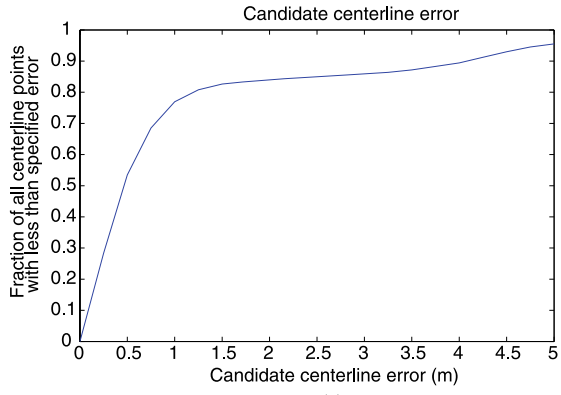


(a)

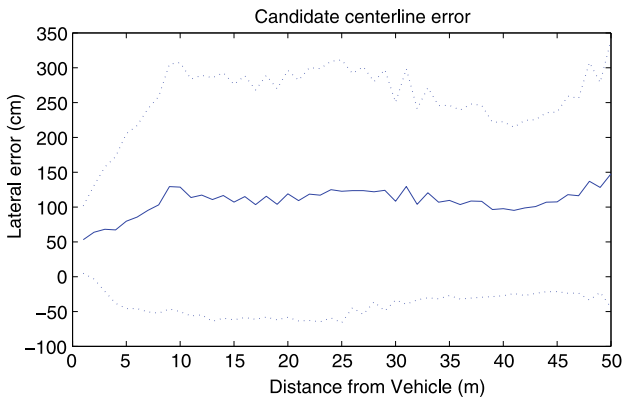


(b)

Fig. 17 Centerline error of our system and the RNDF prior as a function of vehicle roll and pitch, with 1-σ bounds



(a)



(b)

Fig. 18 (**a**) Error distribution for centerline candidates indicating the frequency and magnitude of candidate centerline error. The ideal distribution is a single point in the *top left corner* of the graph. (**b**) Candidate centerline error as a function of distance from the vehicle, with 1-σ bounds

3.4 Centerline candidates

Although our system relied on a road map prior for end-to-end operation, our hope is that it will eventually be able to provide highly accurate estimates of nearby travel lanes without any prior at all. To assess its potential for this task, we can evaluate the accuracy of the centerline candidates produced by the second stage of the system. These estimates are made purely from sensor data alone, and require no GPS or road map prior.

Similar to the filtered and tracked centerline estimates produced as a final output of the system, the centerline candidates can also be evaluated in terms of their centerline error. We sampled centerline candidates generated approximately every meter of travel, and then computed the centerline error for various points on each candidate. In each case, centerline error was computed by comparison to the nearest ground truth lane.

Figure 18a shows the error distribution for candidate centerlines. 53.5% of points on all candidate centerlines were within 50 cm of a true lane centerline, and 4.5% were more than 5 m from any true lane centerline. A common case resulting in large error was when the system generated a candidate centerline on the wrong side of a lane boundary. This could happen when the top of a curb was detected as road paint, or when road paint appeared as a lane boundary with a corresponding curb.

Candidate centerline error as a function of distance from the vehicle is shown in Fig. 18b. Centerline candidates generated very close to the vehicle had the least error, with a mean error of 53 cm at distances of 1–2 m. At a distance of 10–11 m, mean centerline error was 129 cm. Data for this plot was taken only when the vehicle was actually in a travel lane; centerline candidates generated in parking areas were not considered.

After a lane centerline candidate was generated, it underwent a data association step to either match it with an existing lane estimate or to reject it as an outlier. Our system relied on a topological prior to provide correct information about the presence or absence of nearby lanes, and used the candidates to refine the geometric estimates. By applying techniques from vision-based object detection and tracking, our system could be adapted to both detect and track multiple travel lanes.

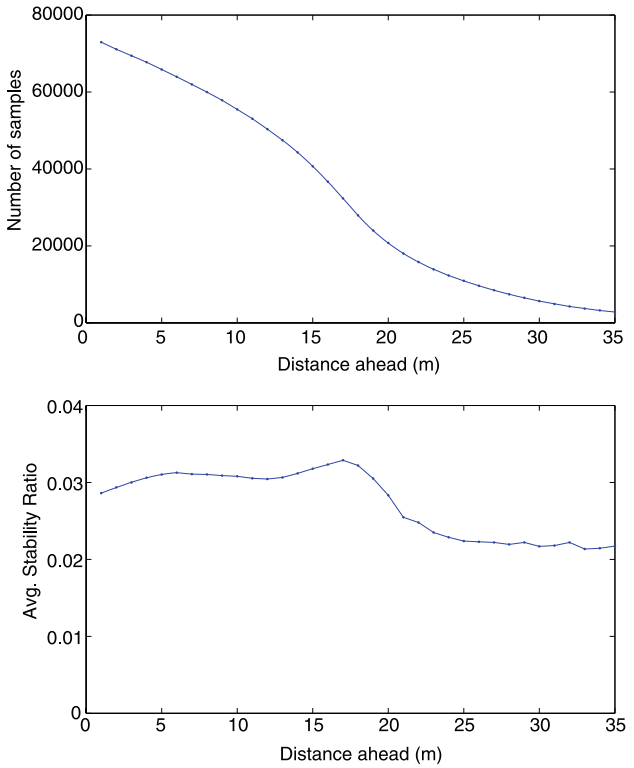


Fig. 19 (*Top*) The average stability ratio. (*Bottom*) The number of samples used to compute the stability ratio varies with r, as only control points with visually-derived high-confidence are used

3.5 Stability

The output of our system is used for high-speed motion planning; thus we desire that its estimates remain relatively stable. Specifically, we desire that once the system produces a high-confidence estimate, that the estimate does not change significantly. To assess the suitability of our system for this purpose, we can compute a *stability ratio* that measures how much its high-confidence lane estimates change over time in the lateral direction.

Consider a circle of radius r centered at the current position of the rear axle center. We can find the intersection $\mathbf{p}_0$ of this circle with the current lane estimate that extends ahead of the vehicle. When the lane estimate is updated at the next time step (10 Hz in this case) we can compute $\mathbf{p}_1$, the intersection of the same circle with the new lane estimate. We define the lane estimator's stability ratio as:

$$R = \frac{\|\mathbf{p}_0 - \mathbf{p}_1\|}{d_v} \tag{5}$$

where d_v is distance traveled by our vehicle in that time step.

Intuitively, the stability ratio is the ratio of the transverse movement of the lane estimate to the distance traveled by the car in that time, for some r. We can also compute an *average stability ratio* for some r by averaging the stability ratios for every time step of the vehicle's trip through the course (Fig. 19). From this figure, we see that the average stability ratio remains small and relatively constant, but still nonzero, indicating that high-confidence lane estimates can be expected to shift slightly as the vehicle moves.

4 Data and software

The interprocess communications framework and software infrastructure we used during the development of our vehicle allowed us to log most of the sensor data collected by the vehicle throughout the Urban Challenge Event, and virtually all of its internal state estimates. In the hope that this data will be useful to others, we have made it available online along with software for parsing and visualizing the data. Due to disk bandwidth constraints, we were able to log only one camera at the source 22.8 Hz sample rate, with the other four cameras spatially and temporally decimated. Data from every other sensor, including the lidar and navigation data, were timestamped and logged at the sensors' maximum sample rate.

In addition to the original sensor data, these log files also contain all of the internal state estimates of our system, such as the road paint detections, centerline candidates, and lane estimates. We encourage interested readers to use the software to better understand our system. The log files and software are at: http://dgc.mit.edu/public.

5 Conclusion and future work

We have attempted to extend, to urban environments, the scope of lane detection and tracking for autonomous vehicles. This paper presented a modular, scalable, perception-centric lane detection and tracking system that fuses asynchronous heterogeneous sensor streams with a weak prior to estimate multiple travel lanes in real-time. The system makes no assumptions about the position or orientation of the vehicle with respect to the road, enabling it to operate when changing lanes, at intersections, and when exiting driveways and parking lots. The vehicle using our system was, to our knowledge, the only vehicle in the final stage of the DARPA Urban Challenge to employ vision-based lane finding for real-time motion planning and control.

Our system works in three stages. In the first stage, calibrated cameras and lidars are used to detect road paint and curbs. A second stage combines the detections in a voting process to construct a centerline evidence image, which is then used to form estimates of potential lane centerlines. A final stage filters and tracks these candidates into lane estimates, while also incorporating a weak prior derived from a road map.

We have provided a quantitative analysis of our vehicle that evaluates its performance under a number of different measures. Our evaluation framework allows us to accurately explain failure modes and determine which aspects would benefit the most from improvement. Additionally, it allows us to objectively evaluate future improvements and compare their successes and failures with those of existing systems.

Despite these advances, our method is not yet suitable for real-world deployment. As with most vision-based systems, it is susceptible to strong lighting variations such as cast shadows, and cannot handle adverse conditions such as rain and snow. Moreover, we have not used all available sources of sensor data, and our analysis has revealed cases in which simplifying assumptions adversely affected system performance.

We are investigating a number of improvements. For example, using lidar intensity data to detect road paint should improve performance in difficult lighting conditions. While we used the solar ephemeris to reject false positives due to solar flare, we could also use it during daylight hours to predict and detect false positives from shadows. Jointly estimating lane width, lane geometry, and elevation gradients should all improve detection accuracy. Finally, since many roads do not use paint as boundary markers, we are extending our method to incorporate other environmental cues.

Acknowledgements This work was conducted on the MIT 2007 DARPA Urban Challenge race vehicle. We give special thanks to Luke Fletcher, Olivier Koch, and John Leonard for numerous discussions and help given throughout the course of this work. We additionally thank the MIT DARPA Urban Challenge Team for their efforts in building a fully-operational vehicle. Besides the authors and the three just mentioned, these team members include David Barrett, Jonathan How, Troy Jones, Mitch Berger, Bryt Bradley, Ryan Buckley, Stefan Campbell, Alexander Epstein, Gaston Fiore, Emilio Frazzoli, Sertac Karaman, Yoshiaki Kuwata, Keoni Maheloni, Katy Moyer, Steve Peters, Justin Teo, Robert Truax, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams.

References

Apostoloff, N., & Zelinsky, A. (2004). Vision in and out of vehicles: Integrated driver and road scene monitoring. *International Journal of Robotics Research*, *23*(4–5), 513–538.

Baluja, S. (1996). Evolution of an artificial neural network based autonomous land vehicle controller. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, *26*(3), 450–463.

Bartels, R. H., & Beatty, J. C. (1987). *An introduction to splines for use in computer graphics and geometric modeling*. Los Altos: Morgan Kaufmann.

Bertozzi, M., & Broggi, A. (1998). GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, *7*(1), 62–80.

Bertozzi, M., Broggi, A., & Fascioli, A. (2000). Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous Systems*, *1*, 1–16.

CNN (2008). Doh! Man follows GPS onto train tracks—when train coming. http://www.cnn.com/2008/US/01/03/gps.traincrash.ap/index.html, Jan. 2008.

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

Dickmanns, E., & Mysliwetz, B. (1992). Recursive 3-d road and ego-state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 199–213.

Leonard, J. et al. (2008). A perception-driven autonomous vehicle. *Journal of Field Robotics*, *25*(10), 727–774.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fletcher, L., & Zelinsky, A. (2006). Context sensitive driver assistance based on gaze–road scene correlation. In *Int. symposium on experimental robotics* (pp. 287–296), Rio De Janeiro, Brazil, Jul. 2006.

Hartley, R. I., & Zisserman, A. (2001). *Multiple view geometry in computer vision*. Cambridge: Cambridge University Press. ISBN: 0521623049.

Iteris, Inc. (2009). http://www.iteris.com.

Kang, D.-J., & Jung, M.-H. (2003). Road lane segmentation using dynamic programming for active safety vehicles. *Pattern Recognition Letters*, *24*(16), 3177–3185.

Kim, Z. (2008). Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, *9*(1), 16–26.

Kreucher, C., & Lakshmanan, S. (1999). LANA: A lane extraction algorithm that uses frequency domain features. *IEEE Transactions on Robotics and Automation*, *15*(2), 343–350.

Kreucher, C., Lakshmanan, S., & Kluge, K. (1998). A driver warning system based on the LOIS lane detection algorithm. In *The Proceedings of the IEEE international conference on intelligent vehicles* (Vol. 1, pp. 17–22), Oct. 1998.

Lyall, S. (2007). Turn back. Exit village. Truck shortcut hitting barrier, New York Times. http://www.nytimes.com/2007/12/04/world/europe/04gps.html, Dec. 2007.

Lee, J. W. (2002). A machine vision system for lane-departure detection. *Computer Vision and Image Understanding*, *86*(1), 52–78.

Lee, J. W., & Yi, U. K. (2005). A lane-departure identification based on LBPE, Hough transform, and linear regression. *Computer Vision and Image Understanding*, *99*(3), 359–383.

McCall, J. C., & Trivedi, M. M. (2006). Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation. *IEEE Transactions on Intelligent Transport Systems*, *7*(1), 20–37.

Miller, T. R. (1993). Benefit-cost analysis of lane marking. *Public Roads*, *56*(4), 153–163.

Mobileye (2009). Mobileye camera based system for: collision warning, lane departure, headway monitoring. http://www.mobileye.com.

Moore, D., Huang, A. S., Walter, M., Olson, E., Fletcher, L., Leonard, J., & Teller, S. (2009). Simultaneous local and global state estimation for robotic navigation. In *Proc. IEEE int. conf. robotics and automation*, May 2009.

US Department of Transportation, Federal Highway Administration, Office of Information Management (2005). *Highway Statistics 2005*. US Government Printing Office, Washington, DC, 2005.

Pomerleau, D. (1995). Neural network vision for robot driving. In M. Arbib (Ed.), *The handbook of brain theory and neural networks*. Cambridge: MIT Press.

Pomerleau, D., & Jochem, T. (1996). Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert: Special Issue on Intelligent System and their Applications*, *11*(2), 19–27. See also IEEE Intelligent Systems.

Rasmussen, C. (2008). RoadCompass: following rural roads with vision + ladar using vanishing point tracking. *Autonomous Robots*, *25*(3), 205–229.

Route network definition file (RNDF) and mission data file (MDF) formats. *Defense Advanced Research Projects Agency*, Mar. 2007.

Taylor, C. J., Kosecká, J., Blasi, R., & Malik, J. (1999). A comparative study of vision-based lateral control strategies for autonomous highway driving. *International Journal of Robotics Research*, *18*(5), 442–453.

Thorpe, C., Hebert, M., Kanade, T., & Shafer, S. (1988). Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, *10*(3), 362–373.

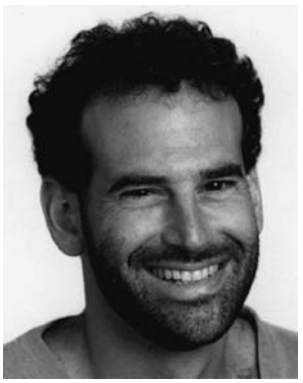
Zhang, W.-B. (1991). A roadway information system for vehicle guidance/control. In *Vehicle navigation and information systems* (Vol. 2, pp. 1111–1116), Oct. 1991.

Albert S. Huang received his Sc.B./A.B. from Brown University in 2003, and his M.S. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT) in 2005. He is is a Ph.D. student at MIT, and has studied mobile robot perception since 2005.

David Moore was a student technical lead for MIT's entry to the 2007 DARPA Urban Challenge while a Ph.D. student at MIT. He is currently a software engineer at DreamWorks Animation SKG, Inc. in Glendale, California. He is also active in the open source community, where he has contributed to the Linux kernel and worked on a number of imaging-related projects.

Matthew Antone received his B.Sc. and M.Eng. in Electrical Engineering from MIT in 1996, and his Ph.D. in Electrical Engineering and Computer Science from MIT in 2001. He is currently a Principal Scientist in the Computer Vision Group at BAE Systems Advanced Information Technologies, where his research interests include object recognition, active tracking, sensor calibration, and scene perception from 2D and 3D data.

Edwin Olson received his Ph.D. in Electrical Engineering and Computer Science from MIT in 2008. He is currently an Assistant Professor at the University of Michigan.



Seth Teller is a Professor of Computer Science and Engineering in the Department of Electrical Engineering and Computer Science at MIT. He led the perception efforts of the MIT Urban Challenge Team.