Lane Estimation for Autonomous Vehicles using Vision and LIDAR

by

Albert S. Huang

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author .....
Department of Electrical Engineering and Computer Science
November 30, 2009

Certified by .....
Seth Teller
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Lane Estimation for Autonomous Vehicles using Vision and LIDAR

by

Albert S. Huang

Submitted to the Department of Electrical Engineering and Computer Science
on November 30, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Autonomous ground vehicles, or self-driving cars, require a high level of situational awareness in order to operate safely and efficiently in real-world conditions. A system able to quickly and reliably estimate the location of the roadway and its lanes based upon local sensor data would be a valuable asset both to fully autonomous vehicles as well as driver assistance technologies. To be most useful, the system must accommodate a variety of roadways, a range of weather and lighting conditions, and highly dynamic scenes with other vehicles and moving objects.

Lane estimation can be modeled as a curve estimation problem, where sensor data provides partial and noisy observations of curves. The number of curves to estimate may be initially unknown and many of the observations may be outliers and false detections (e.g., due to tree shadows or lens flare). The challenge is to detect lanes when and where they exist, and to update the lane estimates as new observations are received.

This thesis describes algorithms for feature detection and curve estimation, as well as a novel curve representation that permits fast and efficient estimation while rejecting outliers. Locally observed road paint and curb features are fused together in a lane estimation framework that detects and estimates all nearby travel lanes. The system handles roads with complex geometries and makes no assumptions about the position and orientation of the vehicle with respect to the roadway.

Early versions of these algorithms successfully guided a fully autonomous Land Rover LR3 through the 2007 DARPA Urban Challenge, a 90 km urban race course, at speeds up to 40 km/h amidst moving traffic. We evaluate these and subsequent versions with a ground truth dataset containing manually labeled lane geometries for every moment of vehicle travel in two large and diverse datasets that include more than 300,000 images and 44 km of roadway. The results illustrate the capabilities of our algorithms for robust lane estimation in the face of challenging conditions and unknown roadways.

Thesis Supervisor: Seth Teller
Title: Professor of Computer Science and Engineering

Contents

Chapter 1

Introduction

The road networks of countries around the world carry countless numbers of people and goods to their destinations each day. To facilitate safe and efficient transport, roadways are typically marked with painted and physical boundaries that define the safe and legal regions of travel. The exact nature and appearance of these markings vary from region to region, but all serve to delineate lanes within which a single file of vehicles is intended to travel.

Fundamentally, lanes divide a travel corridor into multiple, narrower corridors. These subdivisions organize and orient flows of traffic such that many vehicles (e.g., passenger cars, motorcycles, cargo vehicles, etc.) making use of the same physical roadway can do so safely and efficiently. By arranging vehicles into groups traveling in similar directions and speeds, lanes play a critical role in increasing traffic flow and decreasing accidents [49].

A system able to automatically and reliably estimate the roadway and its lanes from a moving vehicle using on-board sensor data would have enormous benefits for land-based travel. It could be used for tasks ranging from wide-scale road and lane quality assessment, to providing inputs to a driver assistance safety or navigation system, to serving as a situational awareness component in a fully autonomous vehicle.

We define the lane estimation 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. Semantic properties refer to information about the lane's intended usage, such as the travel direction, speed limit, and types of vehicle's allowed on a lane (e.g., all vehicles, bicycles, high-occupancy, no trucks, etc.). 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.

Each of the semantic, topological, and geometric properties of a lane introduces difficult estimation challenges. Inferring travel direction, speed limits, lane connectivity, and other usage restrictions, all require some prior knowledge of a region's driving rules in addition to the ability to interpret road signs and associate them with lanes. The perceptual cues that provide this information can vary dramatically across regions; environmental conditions and roadways change appearance over time.

Figure 1-1: Vehicles encounter roadways and lanes in a variety of situations, from well-marked roads on a clear and sunny day, to adverse lighting and environmental conditions. Accurately detecting, localizing, and classifying all travel lanes in the face of these challenges is the goal of a lane estimation system. Top left, bottom left, and bottom right photos courtesy of Flickr users albertoog, rexroof, and fredosan via Creative Commons, respectively.

Roadways and the situations in which they are commonly found can range from well-marked surfaces on clear and sunny days, to poorly marked lanes obscured by snow, rain, shadow, strong sunlight, and a host of other factors (Figure 1-1). Algorithms designed for one scenario may not work well in others; the ability to robustly handle new and unfamiliar situations is critically important. Faded road paint lines still visible after re-painting must be distinguished from fresh road paint, and a functional system must accurately model the width, curvature, and other geometric aspects of the road network required for the task at hand.

For these reasons, accurately detecting and estimating all the different properties of a road network in a real-world setting is a challenge beyond the reach of existing artificial intelligence techniques. Instead of attempting to solve all aspects of the lane estimation problem at once, previous approaches have studied limited aspects, and made simplifying assumptions when relevant. Limits on factors such as the parts of the road network being estimated, the type of road networks considered, the number of lanes present, and the initial conditions surrounding an estimation procedure all help cast the lane estimation problem into a more approachable form.

This thesis focuses on algorithms for detecting lanes where they exist, and determining geometric information for each detected lane. To do so, the system must utilize information from its on-board sensors, and any other a priori information it has available. For example, forward-facing cameras could be used to detect road paint markings, laser range scanners (LIDAR) could provide accurate measurements of local scene geometry, and Global Positioning System (GPS) devices can localize the vehicle within a previously acquired road map.

1.1 Overview

Lane estimation can be decomposed into several major sub-topics (Fig. 1-2).

- Lane model
- Feature detection
- Lane detection from features
- Lane tracking

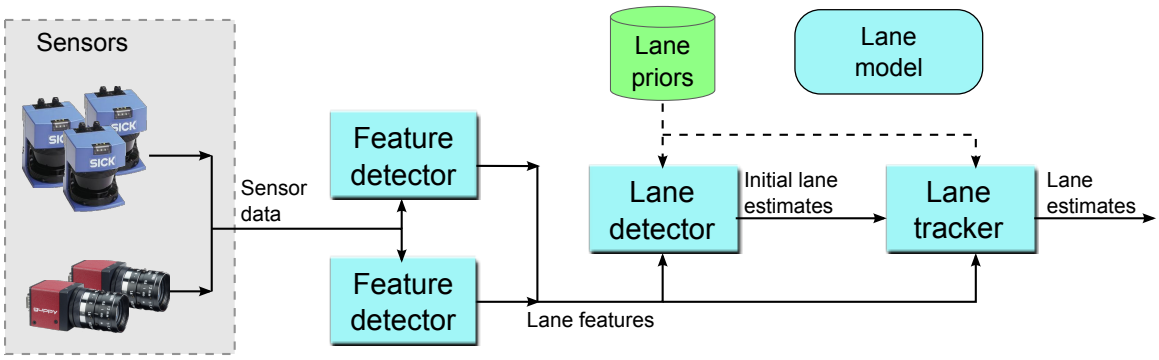


Figure 1-2: Lane estimation can be divided into several major subtasks (blue boxes). Feature detectors extract information relevant to lane estimation from sensor data. Features are interpreted by a lane detector, while accounting for any domain-specific priors, to produce initial lane estimates. Lane tracking algorithms then exploit spatial and temporal continuity to maintain lane estimates over time. The choice of lane model greatly affects all of these steps.

The choice of lane model greatly affects which types of lanes can be accurately represented, the estimation algorithms that can be used, and the possible applications of the estimation system. Simple models such as lines and parabolas are computationally efficient, but are unable to express a substantial class of road geometries. As such, they may be appropriate for some applications such as highway driver assistance, but are unsuitable for fully autonomous driving.

Given a lane model, the first step in a lane estimation algorithm is to process its sensor data to extract features useful for the actual estimation, such as road paint and sections of physical curbs. As with all feature detection algorithms, the goal is to maximize the number of correctly detected road features while minimizing spurious detections and false alarms triggered by phenomena such as lens flare, strong shadows, or road cracks.

The lane detection problem consists of using detected features to infer the presence of one or more lanes, and to generate an initial estimate of lane geometry. Even with perfect feature detection, it can often be difficult to assess the number and geometry of lanes present in a situation. Occlusions, incorrect markings, and other adverse observing conditions can complicate this process.

Once a lane estimate has been initialized, a lane estimation algorithm can exploit spatial and temporal continuity constraints (since lanes usually change gradually) to track the lane as the vehicle moves, and update its estimates as new feature observations are acquired. Since feature detectors are rarely perfect, for robustness any tracking algorithm must reject outliers and account for noise in the raw sensor data.

The difficulty of lane estimation, and each of these topics, varies with the scope of the intended application. Applications that can restrict the class of lanes (e.g., highway only), the number of lanes to estimate, and even the initial estimates (e.g., vehicle starts in the middle of a lane), can choose simpler lane models and estimation methods. For example, driver assistance, lane keeping, and fully autonomous driving systems present dramatically different lane estimation requirements.

The remainder of this chapter presents a more detailed introduction to these topics and describes the primary contributions of this thesis. Chapter 2 presents feature detection algorithms that extract curves representing potential lane boundaries, such as road paint and raised curbs, from video and LIDAR data. Chapter 3 describes a lane estimation algorithm that fuses these features with a road map in order to guide a fully autonomous vehicle through the 2007 DARPA Urban Challenge. While successful for the Urban Challenge, this algorithm suffers from a number of limitations. Chapter 4 presents a novel curve representation that applies well-studied estimation and tracking techniques to the curve estimation problem. Chapter 5 describes lane estimation using this new representation, and compares its performance to that of the algorithms developed earlier for the Urban Challenge.

1.2 Lane keeping and departure warning

Aspects of the lane estimation problem have been studied for decades in the context of autonomous land vehicle development [16, 68, 48] and driver-assistance technologies [7, 8, 3, 22, 36]. The majority of lane estimation systems have been developed for lane keeping and lane departure warning (LDW) in highway environments [48].

The goal of a lane keeping system is to automatically control the vehicle so that it stays in the current travel lane, whereas a lane departure warning system uses its lane estimates to simply emit an audible or visible warning if the human driver makes an unexpected lane change. Lane keeping systems exhibit some autonomy in

the sense that they use estimates of the current lane to center the vehicle within the lane. However, they typically require a human driver to "stage" the vehicle into a valid lane before enabling autonomous operation, and to take control whenever the system can not handle the required task, for example during highway entrance or exit maneuvers [68]. Lane departure warning systems such as MobilEye and Iteris have seen commercial success [50, 31], in part due to the tolerance of human drivers to both false alarms and missed alarms. Originally used to assist long-haul truck drivers on cross country drives, LDW systems have recently begun to appear as safety systems in luxury passenger vehicles.

By focusing on estimating the vehicle's current lane of travel, and assuming that the vehicle is already in a lane and oriented accordingly, the lane keeping and departure warning problems are greatly simplified. The space of possible lane configurations that must be searched is significantly reduced, as the estimation algorithms need not consider multiple lanes or lanes perpendicular to the vehicle's direction of travel. Additionally, since LDW systems are used primarily in highway environments, they additionally constrain the space of possible lane configurations by assuming low road curvature (due to high speed) and few, if any, roadway inflection points.

Because of these simplifications, lane keeping and departure warning systems are essentially limited to providing cues to assist a human or computer driver with staying in the current lane. While this capability has proved valuable for driver assistance, fully autonomous driving requires much more situational awareness than an estimate of the current lane.

1.3 Curve representation

A natural way to represent a lane is as a set of curves running along the ground surface. The left and right boundaries of a single lane each form curves, as does the typically invisible centerline of a lane. All three of these curves represent semantically meaningful structures, and are directly useful for algorithms for both autonomous operation and driver assistance. For example, a motion planning algorithm might search for trajectories that minimize the vehicle distance from the centerline curve, and a warning system could alert a non-signaling driver whenever a wheel is about to cross a boundary curve.

Although lanes themselves have a straightforward mapping to curves, choosing how to represent the curve itself presents computational challenges. In its most general form, a 2D curve $\mathbf{f}$ can be described as a mapping from an interval of real numbers to $\Re^2$:

$$\mathbf{f}(s) = \begin{bmatrix} f_x(s) \\ f_y(s) \end{bmatrix}, \quad s \in [s_1, s_n] \tag{1.1}$$

where s_1 and s_n denote the lower and upper bounds of some interval, and $f_x(s)$ and $f_y(s)$ denote the x and y coordinates of the points through which $\mathbf{f}$ passes. The parameter s can take on different meanings, most commonly time or distance along the curve, or it can simply be an artifact of the curve parameterization method. It is almost always the case that $f_x(s)$ and $f_y(s)$ are continuous functions, and we assume

this to be true for the remainder of this thesis. Curves that can be described only by this general form are computationally difficult to represent, as it is not possible to explicitly store an infinite number of values. Instead, most algorithms working with curves choose a subspace of all possible curves in which a desired curve can be described with a small number of parameters. Curves with piecewise linear curvature, also known as piecewise clothoid, are often used in highway design, and consequently have been a popular way to represent road geometry in lane keeping and LDW systems [17, 53, 66]. Other models have included straight lines [2], parabolas [48], cubic splines [70], and piecewise linear contours [44].

The suitability of a curve representation for lane estimation is often affected by the problem formulation. In particular, restrictions on which lanes to estimate and assumptions on how the vehicle is positioned and oriented with respect to these lanes are commonly used to reduce the space of possible lane configurations. For example, lane keeping and lane departure warning systems assume the vehicle is roughly in line with the current lane of travel, and attempt to estimate only this lane. With the additional assumption of low curvature roads, simple representations are often sufficient. In vision-based systems, lanes are typically represented as curves in the image plane. As the cameras in these cases are fixed to the vehicle, these systems make an implicit assumption about the orientation of the vehicle with respect to the lanes.

The choice of curve representation can be roughly described as a trade-off between representational power and computational simplicity. Simple representations such as lines and parabolas are computationally efficient to estimate and fit to data, but are unable to accurately describe a variety of useful real-world geometries. However, more expressive representations that can describe these geometries require more carefully considered estimation procedures in order to achieve efficient estimation.

1.3.1 Coordinate Frame

The curves defining a lane's boundaries and centerline are fixed to the world, so it might seem obvious to represent the curves in a coordinate frame that is also fixed to the world. Unfortunately, this is not as easy a task as it seems. Data received from a vehicle's onboard sensors is typically given in the sensor's own internal coordinate frame, such as range measurements and angles with respect to a LIDAR sensor body, or light intensity measured for a camera's image pixel. Projecting this data into the world frame requires a known mapping from sensor coordinates to world coordinates, and it is precisely this mapping that is often difficult to obtain. Using a GPS receiver by itself or combined with an inertial navigation system (INS) to estimate the sensor-to-world transformation is appealing, until the realities of GPS bias, drift, dropouts, and multipath effects set in.

One solution to this problem is to avoid solving it, and represent curves in a different coordinate frame. In the case of a vehicle with a single forward-pointing camera, curves could be represented in image space. If the extrinsic calibration of the camera or sensor is known, then curves could be represented in a Cartesian frame fixed to the vehicle (the body frame). Because monocular cameras do not report

depth, features detected in image space must then be projected into the body frame. This is typically accomplished using an inverse perspective mapping onto an assumed or estimated ground surface. Stereo can also be used to estimate feature depth [7, 53].

Estimating curves in a body frame or sensor frame provides some computational convenience, but propagating estimates forward through time as the vehicle moves becomes a challenge of its own. Accurate propagation requires a reliable estimate of the vehicle egomotion – its motion relative to a previous pose. One option is to model vehicle motion as white noise, and simply increase the estimate uncertainty over time. This approach is especially popular for Kalman filtering in the body or image frame. Another option is to simply not propagate estimates, and to treat each time step independently from previous time steps. While simple, this approach discards valuable information provided by previous estimates.

If vehicle egomotion is accurately known, then curve estimates can also be propagated forward through time in a body or sensor frame. More simply, curve estimates and sensor measurements can be represented in a coordinate frame fixed to the local environment. With sufficiently accurate egomotion estimates, drift in this coordinate frame can be ignored over short distances and time scales.

Throughout this thesis, we represent curves in such a coordinate frame, which we call the local frame [52]. The local frame is defined such that the vehicle pose is the integrated result of its previous egomotion estimates, with zero uncertainty about its current pose. Although there is uncertainty in the egomotion estimates, the local frame differs from a world frame initialized at the vehicle starting point in that it shifts the aggregated uncertainty to previous poses, rather than accumulating it forward in time. An analysis of the local frame's properties is given elsewhere [52].

Sensor measurements can be projected into the local frame easily, and remain valid over short distances and time scales. The measurements can be used to construct locally valid lane estimates and other maps, and have the property that accurate propagation of these estimates forward through time can be reasonably approximated with the identity transformation. Egomotion estimates for our vehicle are provided by an inertial navigation system and wheel odometry, but could also be provided by other methods such as visual odometry [55] or laser scan matching [57, 9].

1.4 Feature and Lane Detection

The primary goal of a lane estimation algorithm is to estimate the curves representing the lanes of interest, using available sensor data, and to update these estimates over time as the vehicle observes new parts of the roadway. The first step in this process, arriving at an initial estimate, is the lane detection problem.

Detecting lanes in arbitrary configurations from camera or LIDAR data is not a well-studied problem; previous systems have typically assumed a limited space of possible arrangements. The most common approach is to assume that a fixed number of lanes exist in the sensor field of view, then use regression techniques to estimate the initial lane parameters. Our work relaxes this constraint and attempts to detect an initially unknown number of lanes, with a wider variety of possible configurations.

In virtually all cases, lane detection proceeds by first searching the sensor data for features, then using resulting features to estimate the lanes. In this context, a feature is any subset or transformation of a subset of the sensor data that is specifically useful for lane detection (Fig. 1-3). This process arises from the fact that the sensors used for lane estimation are not specific to lane estimation, and so the goal of feature detection is to extract the most useful parts of the sensor data. For example, from camera images, useful features might be road paint, straight lines, a vanishing point, etc. From LIDAR data, curbs and the road surface form good features for road and lane estimation. Additional features that would be useful if they could be robustly detected might include traffic signs, traffic lights, sidewalks, road shoulders, and even other vehicles.

Figure 1-3: Feature detectors process sensor data to extract information useful for lane detection. The features considered in this thesis are painted markings and physical curbs. Other useful features may also include vehicles, traffic signs, and traffic lights.

Most vision-based methods attempt to extract road paint from image data by searching for bright contours on a dark background. Techniques used for this step include using a Canny or Sobel edge detector [34], convolving a match filter with the image and searching for response peaks [48, 66], and using color-based segmentation [37].

Features can be used to detect lanes in a variety of ways. They can be used to directly regress lane parameters, e.g., via a Hough transform [53, 67, 42], template matching [61, 44], or least-squares fitting [37]. Alternatively, features can be aggregated and fused together using clustering methods such as connected-components [34]. The primary challenge in estimating lane parameters from features is to do so in a way that is robust to noise and outliers. Feature detectors typically use fairly low-level operators, and often have a high false-positive rate. Any reliable fitting procedure must use some form of outlier rejection and fitting criteria in order to avoid detecting lanes where none actually exist.

While the vast majority of lane estimation systems approach the problem in several explicit steps corresponding to feature detection, lane detection, and then lane tracking, a notable exception is the ALVINN system [60, 4]. Designed for lane keeping, this system trains a neural network to directly map image intensity measurements to steering commands, entirely avoiding the construction of an explicit road model.

This results in good performance on the trained road network, but overfitting and the lack of a generalizable road model results in poor performance on other roads with significantly different appearance and geometry.

1.5 Lane tracking

Once an initial estimate is formed, lane estimates must be propagated forward through time and updated as the vehicle moves and makes new observations. Lanes do not change over short time scales, and often have highly predictable geometry, so information about a lane at one point in time and space should provide significant information about the lane at nearby points (Figure 1-4).

Correctly propagating estimates forward and updating them with new information is referred to as the lane tracking problem. One approach to this problem is to avoid it completely, and perform no tracking at all. Instead, lanes could simply be re-detected at each time step independently from all previous estimates [34, 64, 7]. While simple, this approach discards highly useful information, as lane geometry is typically continuous and smooth over both space and time. Algorithms that involve an iterative refinement procedure based on gradient descent or a spring-mass model may use previous estimates to initialize estimation on new sensor data [70]. This exploits spatial and temporal continuity, but does not account for vehicle motion or a scene change. Other approaches include well studied techniques in estimation and tracking, such as the ubiquitous Kalman filter [33].

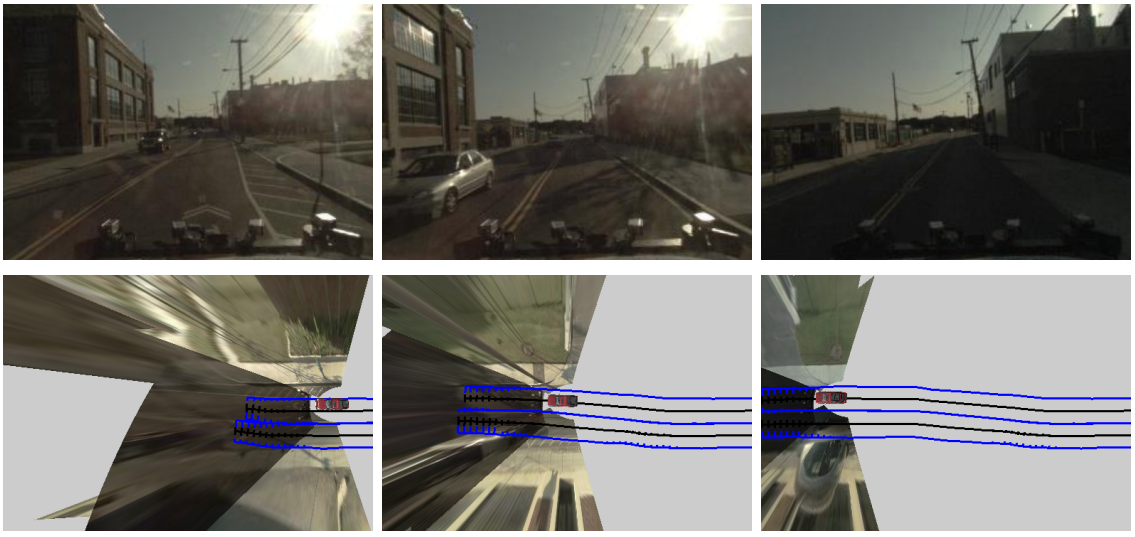


Figure 1-4: Lanes exhibit spatial and temporal smoothness. Estimates at one point (left) provide valuable information about geographically and temporally similar locations (middle, right). Black and blue lines indicate estimated lane centerlines and boundaries, respectively. Front camera images (top row) are projected onto a ground plane along with left and right camera images in the bottom row.

Given an initial lane estimate, the tracking task poses two primary challenge: data

association, and estimate update. The first concerns determining how each feature affects each lane estimate, if at all. Feature detectors are inherently noisy: a road paint detector may misfire on tree shadows, lens flare, cracks, or bright elongated objects; a curb detector may misfire on speed bumps, damaged road surfaces, other vehicles, etc. Feature detectors may also miss important features due to occlusion or conservative thresholds (e.g., road paint not appearing bright enough). Determining which detected features should be associated with which estimates can be approached using a variety of tests, such as template matching [7], statistical tests [35], or random sample consensus methods [36]. This problem is not exclusive to lane estimation; it is well studied in the context of robotic mapping and tracking literature as the data association problem [11, 58].

The second major challenge in tracking involves the actual update procedures. Once the data associations of features to lane estimates have been made, the features must be used to update the lane estimates in a way that takes into account both previous estimates and newly observed information. In doing so, the update procedure must preserve the curve representation, account for noisy observations, and utilize constraints inherent to road networks. For example, constraints on lane width, curvature, and overlapping lane geometries should all be reflected in the update process. As the geometries of adjacent lanes are correlated, features used to update one lane estimate may also provide valuable information about nearby lanes. Thus, robustly incorporating all available information while respecting the model and various constraints presents a challenge not easily formulated using standard optimization and estimation techniques.

Dickmanns's early work in lane estimation used an extended Kalman filter on a curve space with only two curvature parameters [17]. Despite its limited representational power, this system was able to successfully operate in real time on computers with scant computational resources by today's standards. Since then, a variety of tracking techniques have been proposed using variants of the Kalman filter [53, 39, 67], differing primarily in what aspects of the road network are estimated, and in how features are incorporated into the filter.

The Kalman filter has well known limitations, most notably its inability to accurately represent multimodal distributions. To track multiple hypotheses in these situations, researchers have experimented with particle filters, where the primary difference is again in the road model and how features are used to update particles [66, 2, 21, 47]. Although particle filters are able to represent multimodal distributions, they suffer from limitations of their own such as significantly increased computational complexity and particle depletion.

1.6 Evaluation

Crucial to the effective evaluation of any estimation algorithm is an objective way to assess its performance. In many estimation problems, this is done by first compiling a data set where the values to be estimated are manually labeled by a human expert. The expert labels are then referred to as "ground truth." The algorithm is tested

on the data set and its output is compared with ground truth. This is especially useful when comparing multiple algorithms, as it provides an objective and repeatable method to assess the performance of each algorithm in a controlled setting.

Obtaining a useful ground truth dataset for the lane estimation problem is an elusive and difficult endeavor. While a true test of an autonomous system would be to analyze its performance while deployed in autonomous driving scenarios over a variety of real-world road networks, environments, and traffic conditions, such a test is not typically feasible for obvious logistical and safety reasons.

In the absence of closed-loop testing, one could collect data sets and evaluate a method's lane estimates against a labeled ground truth data set. The challenge then becomes how to obtain a data set representative of the variety of roads and conditions in which the method is likely to be fielded. Perhaps the simplest way 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 is extraordinarily time consuming, and would thus not be an efficient or even feasible way to densely label many hours of data.

Evaluations of lane estimation systems in the existing literature have ranged from simply presenting lane estimates superimposed onto camera images [64, 32, 53, 66], to providing summary statistics on closed-loop experiments [7, 61], to rough analysis of short data sequences [70, 36]. In the limited cases where comparisons are made against a set of ground truth images, there are typically only a few hundred labeled images [48]. While these are certainly useful, a more comprehensive data set would be desirable.

1.7 Contributions

Lane estimation can be decomposed into a number of sub-problems, each with its own challenges. Choosing a curve model requires consideration of representational power and the computational costs of reasoning about curves. Feature and lane detectors must have high precision and recall in the face of real world conditions. Lane tracking algorithms must be able to incorporate new information into existing estimates fast enough to be useful, while also accounting for noisy data and constraints typical to lane estimation. Finally, progress measures should be defined that allow for meaningful and objective assessments of a proposed system.

The scope of each of these problems can vary with the intended application, and most approaches in the past have been limited to driver assistance and lane keeping. The work presented in this thesis aims to extend the scope of lane estimation to autonomous driving in a wide variety of environments. Its contributions include:

- Vision- and LIDAR-based feature detectors that extract road paint and curbs from sensor data in real-time. False positives are greatly reduced by combining information from multiple sensors.

- A lane estimation algorithm that fuses detected features with a rough prior road map. This algorithm was used to guide a fully autonomous passenger vehicle

through a 90 km race course in a suburban-like environment, amidst both robot- and human-driven traffic.

- A new curve representation that can express geometrically complex curves, while also permitting the use of efficient estimation and update methods. When applied to lane estimation, this allows for joint estimation of multiple curves, robust outlier rejection, and lane tracking when large parts of the lane boundaries are occluded or otherwise unobserved by the feature detectors.

- A ground truth dataset spanning more than 40 km and 4 hours of travel. Manually labeled lane geometry is available for every moment of travel in this dataset, including more than 300,000 images. This dataset allows for a detailed, quantitative analysis of lane estimation algorithms, and an objective comparison of different methods.

In order for autonomous vehicles to be realistically fielded in real world environments, they must be able to operate in the presence of many other vehicles and moving objects. Lanes are a natural way to organize and shape the flow of traffic for many travelers using the same physical pathways, and the ability to detect and accurately estimate lane geometry is a valuable asset for an autonomous vehicle. Our work demonstrably advances the state of the art in lane estimation, and addresses a variety of existing problems while suggesting new ways to approach others.

While our work addresses a number of standing issues in autonomous vehicles, it also leaves many topics untouched. We do not estimate lane intersections, forks, merges, or other topological aspects of road networks. Neither do our methods produce a semantic understanding of road networks such as travel direction, speed limits, rights-of-way, and vehicle restrictions (e.g., carpools only). A fully autonomous vehicle able to traverse an unknown road network requires these capabilities, in addition to those proposed by this thesis. Nevertheless, our work can still be of immediate use in systems that demonstrate partial autonomy, or in any other vehicular application that can benefit from a geometric understanding of the travel lanes near the vehicle.

Chapter 2

Feature Detection

This chapter describes two vision algorithms used for detecting painted lines on the ground, as well as a technique for detecting curbs using 3D laser scan data. As input, the vision algorithms receive grayscale images from a single camera, pose information from the vehicle's IMU, and 3D obstacles detected from LIDAR if available. As output, all detection algorithms produce a list of curves, represented as polylines lying on the ground model in a local coordinate frame [52], 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. 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 sensor 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 (as in [48, 36]), while still reaping the benefits of calibrated cameras and real-world measurements.

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

2.1 Absolute Sensor Calibration

Our detection algorithms assume that GPS and IMU navigation data are available, and of sufficient quality to provide an earth-relative 6-DOF pose estimate and 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 Figure 2-1:

- The horizon line is projected into each image frame. Only pixel rows below the projected horizon are considered for further processing, thus both enhancing

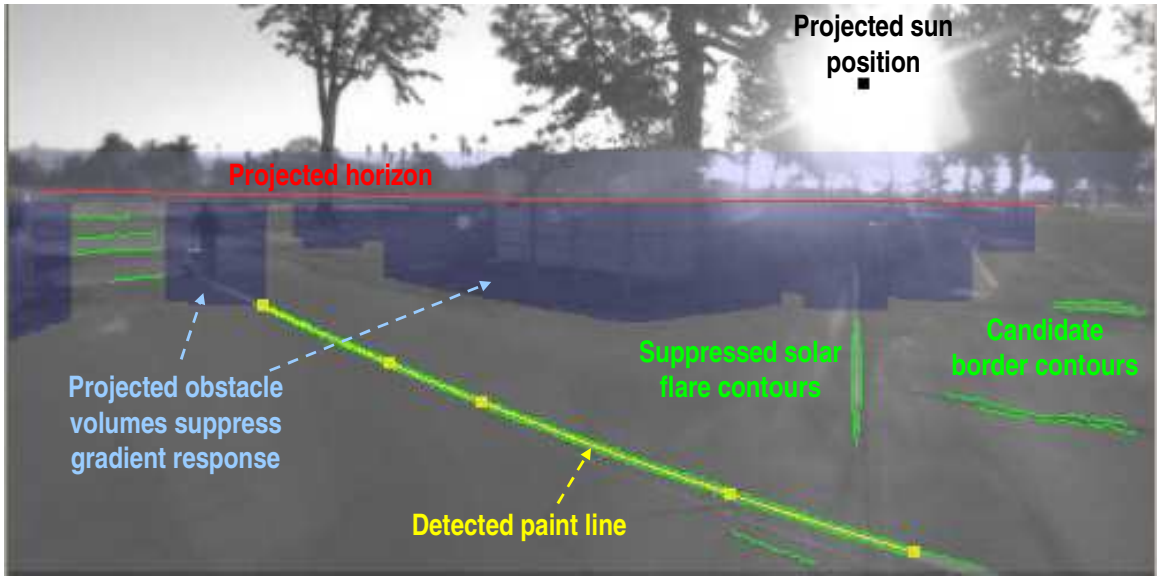


Figure 2-1: Use of absolute camera calibration to project real-world quantities, such as sun position and horizon line, into a video image.

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 express obstacle locations in the local reference frame.

- Detected obstacles are projected into the image, and their spatial 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 [63]; line estimates that point toward the sun in image coordinates are then removed. Others have used a similar approach for shadow prediction [62]; we have found it successful for preventing spurious paint detections arising from lens flare.

- All detections can be transformed into in a common local reference frame for meaningful fusion by the higher-level lane centerline estimation stage.

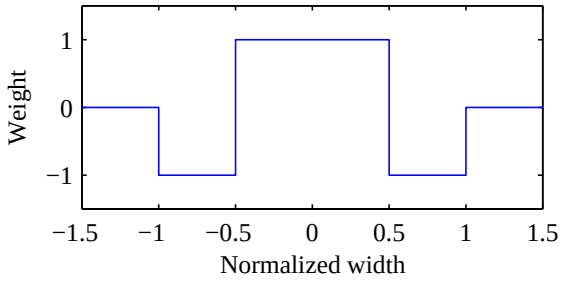


Figure 2-2: The shape of the one-dimensional kernel used for matching road paint, tuned to find light with dark on either side. By applying this kernel horizontally we detect vertical lines and vice versa. The kernel is scaled to the expected width of a line marking at a given image row and sampled according to the pixel grid.

2.2 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 using 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 Figure 2-2. 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 [48]. 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 Figure 2-3, this operation successfully discards most of the scene clutter, and produces a strong response along line-like features. This is done separately for the vertical and horizontal kernels, giving two output images (Figures 2-3b, 2-3c).

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 interest points. 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 interest point, 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 derivative will be near zero. Thus, we first compute the Hessian, the 2×2 matrix of second derivatives

$$H = \begin{bmatrix} F_{xx} & F_{xy} \\ F_{xy} & F_{yy} \end{bmatrix} \tag{2.1}$$

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 interest point as the perpendicular of the underlying line (Figure 2-3d).

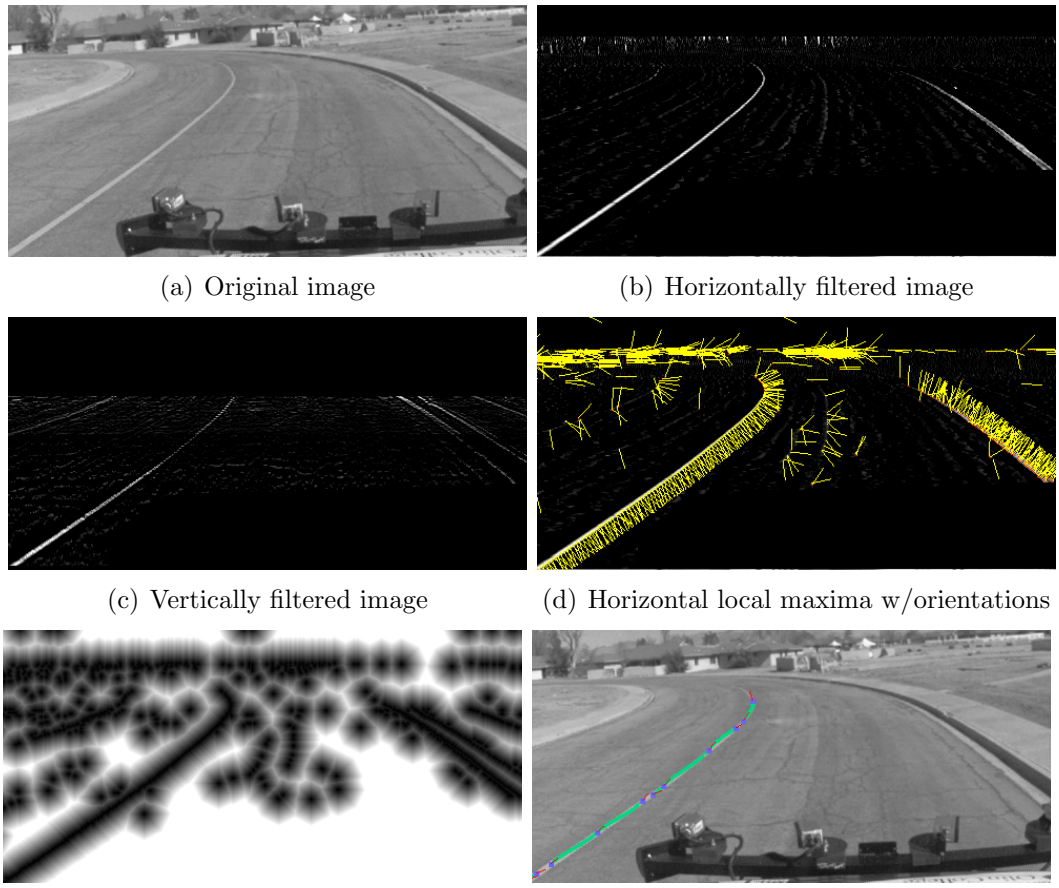
Once the list of interest points is generated, we compute a distance transform of the image, such that the intensity at each pixel of the distance transform is proportional to the Euclidean distance from that pixel to the nearest interest point (Figure 2-3e).

We use cubic Hermite splines to connect the interest points 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

$$\begin{aligned} \mathbf{p}(t) = \quad & (2t^3 - 3t^2 + 1)\mathbf{p}_0 + (t^3 - 2t^2 + t)h\mathbf{m}_0 \\ & +(-2t^3 + 3t^2)\mathbf{p}_1 + (t^3 - t^2)h\mathbf{m}_1 \end{aligned} \tag{2.2}$$

where $t \in [0, 1]$ and $\mathbf{p}_0$ and $\mathbf{p}_1$ are a pair of neighboring control points [6]. 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 interest points extracted above directly as control points and the extracted perpendicular vectors as the tangents (after rotating them 90 degrees to orient them in the "forward" spline direction).

We now describe our algorithm for fitting splines to interest points. First, the algorithm selects 100 "seed" points near the bottom of the image, since points near the bottom are closer to the camera and more well-defined. We then consider every interest point further than 50 pixels but closer than 60 pixels away from some starting interest point. Any interest points in this annular region are candidates for the second control point of a spline that starts at the seed interest point. For each candidate, we sample a spline from the seed point to the candidate point using Equation 2.2 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.



(a) Original image (b) Horizontally filtered image

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

(e) Distance transform (f) Spline fit

Figure 2-3: Our first road paint detector: (a) The original image (b) Original image convolved with a horizontal matched filter at each row (c) Original image convolved with a vertical filter at each row. (d) Local maxima in the horizontal filter response are enumerated and their computed dominant orientations, depicted by the perpendicular to each maximum. (e) A distance transform describing the shortest distance from each pixel to some local maximum. (f) Detected road paint curves.

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 interest points are removed from the global interest point 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 each spline's score, 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.

2.3 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 use 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 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 (Figure 2-4).

Our approach is quite flexible and robust to many conditions, including several potential shortcomings identified in other road paint extraction algorithms [48]. 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 [67, 61] or frequency-based [38] techniques; and features are reliably extracted even under variations in road texture and scene illumination, unlike intensity analysis techniques [60, 4].

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 weak, unreliable edge responses arising from low-contrast regions while preserving 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 [16, 34, 39], 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

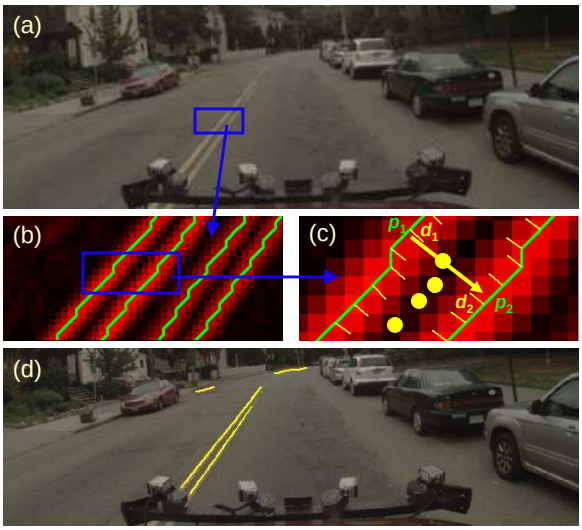


Figure 2-4: Progression from (a) original image through (b) smoothed gradients (red), border contours (green), and symmetric contour pairs (yellow) to form (c) a paint line candidate. Final line detections are shown in (d) the bottom image.

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 nearly 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 [48].

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 (Figure 2-4c). 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 predefined 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 having only one edge or the 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 [41, 43, 34], we do not enforce straight line constraints. Next, we fit parabolic arcs to the merged curves and recursively break them and refit 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.

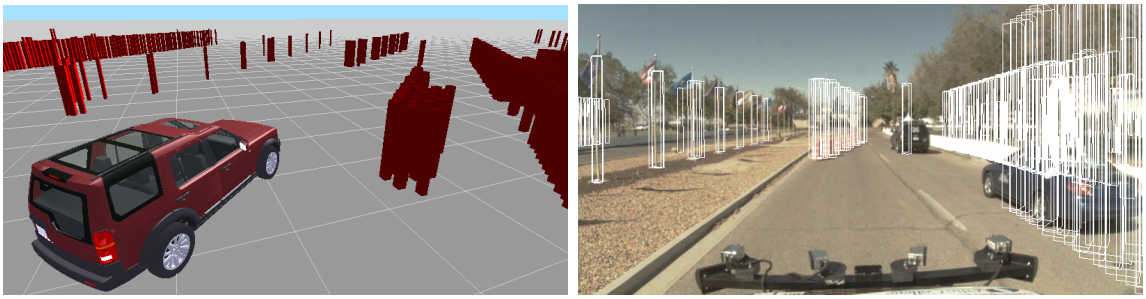
In practice, the primary differences between the two road paint detection algorithms we employ lie in sensitivity and speed. The contour-based detector tends to estimate smoother curves due to its parabolic curve model; the gradient-based detector more accurately captures the geometry of non-parabolic curves. The variable width filter kernels of the gradient-based detector give it a range advantage, allowing 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.

As the road paint detection algorithms extract contours from the camera images, the contours are projected into a local Cartesian coordinate frame so that they can be fused with features from other sensors and used as input to a lane estimation algorithm. Projecting features into the local frame requires a depth estimate for each curve point, which is not reported by the cameras. A straightforward approach, used by the lane estimation implementation described in Chapter 3, is to assume a locally flat and level ground plane, and to project features onto this assumed plane. Estimating the ground surface from local sensor data can improve the projection in areas with non-level terrain. The LIDAR-based terrain estimation algorithm described in Appendix B was used by the lane estimation algorithm described in Chapter 5 for this purpose.

2.4 Obstacle Masking to Reduce False Positives

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 exhibiting horizontal patterns near the ground plane.

For our vehicle, we developed a LIDAR-based obstacle detection system whose primary purpose was to ensure that the vehicle avoided collisions with fixed obstacles and other cars. 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 spurious. Further, since the body-relative 6-DOF poses of all our sensors are known, we can project 3D



(a) LIDAR-identified obstacles

(b) Obstacles projected into front image

(c) Exclusion mask

Figure 2-5: (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.

obstacle detections into the 2D pixel coordinates of each camera (Figure 2.4b). These projections are used to mask corresponding regions in the camera images, explicitly suppressing road lines detected in those regions (Figure 2.4c).

The LIDAR-based obstacle detector is described in detail in a separate paper [45]. Briefly, the obstacle detection system relies on a heterogeneous collection of 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 range and bearing 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.

2.5 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 road boundaries. These detections provide evidence that can be fused into the

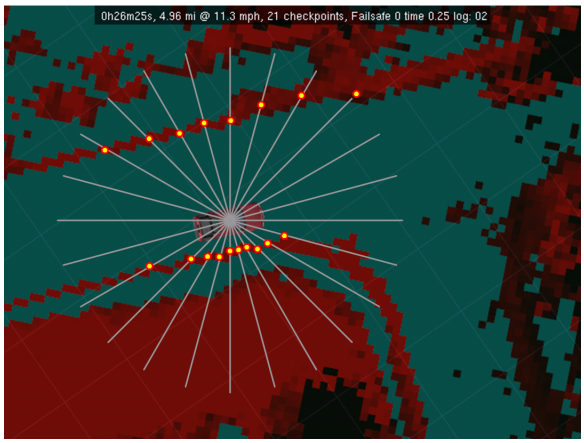


Figure 2-6: Road boundaries from LIDAR. From LIDAR data, our algorithms are able to detect berms and curbs that typically indicate road boundaries. These boundaries are found by casting rays (white line segments) from the vehicle position: the first transition (yellow dots) from smooth surface (blue) to rough surface (red) 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.

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 [45].

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. 2-6. 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 do not mask road paint while the cell remains in sensor range. 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. 2-6). 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 acceptably 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 find the road.

2.6 Summary

Accurately detecting all nearby features useful to lane estimation, while minimizing the number of false alarms, is the primary goal of a feature detector, and the performance of a feature detector greatly impacts the performance of the overall system. Even an optimal estimator would be unable to produce useful estimates if it does not receive useful inputs.

Painted road lines and physical road boundaries such as curbs and berms are the most common environmental features used to delineate roads and lanes. This chapter described two vision algorithms for detecting road paint, and a LIDAR-based algorithm for detecting curbs. False positives in the vision algorithms are greatly reduced by masking out LIDAR-identified obstacles, and by using solar ephemeris computations to account for spurious paint detections caused by sun flare. All of these algorithms operate in real-time.

Other types of features can also be useful for lane estimation. In particular, the presence and location of other vehicles on the roadway often provides strong cues about lane geometry, and in some cases are the most reliable (consider that it is much easier to follow a car on an unlit road on a dark stormy night, than it is to drive alone). Feature detectors that can reliably detect vehicles and other objects such as street lights and traffic signs are an active area of research, and would be useful inputs to a lane estimation algorithm in addition to those described here.

Chapter 3

Lane Estimation in the DARPA Urban Challenge

In this chapter, we present an algorithm that takes as input the road paint and curb features described in the previous chapter, and estimates the number and geometry of nearby travel lanes. Using GPS positioning estimates, it then fuses the locally sensed lane estimates with a weak prior on lane geometry derived from a globally referenced road map.

This algorithm 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. With five cameras and thirteen LIDARs, this algorithm was incorporated into a closed-loop controller to successfully guide an autonomous vehicle through a 90 km suburban-like course at speeds up to 40 km/h amidst moving traffic.

Lanes are estimated and tracked in two stages. In the first stage, road paint and curb features (Chapter 2) are combined in a voting process to detect and estimate lanes of travel near the vehicle. Intuitively, each feature detection votes for lane geometries that may have resulted in the feature, and votes are accumulated in an evidence grid. Lanes are then fit to the evidence grid using a robust random sampling method and a locally parabolic lane model. The output of a first stage is a set of curves representing detected lane centerlines.

In the second stage, the detected lane centerlines are fused with previous lane detections and lane geometries derived from a globally referenced road map. This discards most false detections and allows the system to guide an autonomous vehicle through an established road network while allowing for inaccuracies and sparse information in the road map.

Section 3.1 describes how lanes are detected and estimated from the evidence grid. Section 3.2 describes how new estimates are fused with previous estimates and the road map. A detailed analysis of the system's performance during the DARPA Urban Challenge is then given in Section 3.3.

3.1 Lane Centerline Estimation

The geometry of nearby lanes is estimated 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 accurate to within sensor uncertainty for stretches of road that lie within sensor range (about 50 m in our case).

Lane 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.

3.1.1 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 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 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 U.S. are between 3.05 m and 3.66 m wide [56]. 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 $\mathbf{p}$:

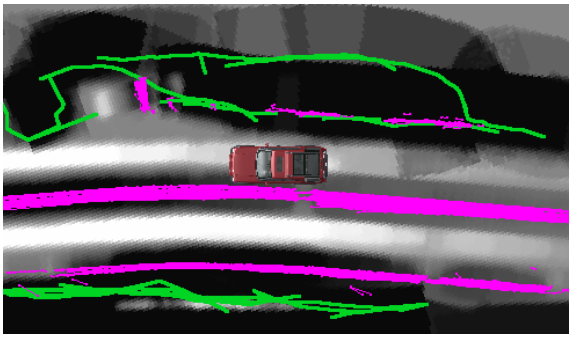
$$f_{rp}(x) = -e^{-\frac{x^2}{0.42}} + e^{-\frac{(x-1.83)^2}{0.14}} \quad (3.1)$$

$$f_{cb}(x) = -e^{-\frac{x^2}{0.42}}. \quad (3.2)$$

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 lane centerlines. 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

(a) Images from left, front, and right on-board cameras.

(b) Road paint and curb features projected into the local frame.



(c) Evidence image constructed from features.

Figure 3-1: Raw sensor data from (a) cameras and (not shown) LIDARs are used to (b) detect road paint (magenta) and curb (green) features. Each of these features votes on potential lane geometries in a (c) centerline evidence image.

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 small sections. The effect is that a detection influences only those centerline evidence values immediately next to it, 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 inserting imaginary road paint detections connecting two separate road paint detections when they are physically proximate and nearly collinear.

Construction of the evidence image, shown in Figure 3-1, is similar to a Hough transform [19], in that each observed feature is used to vote on centerline geometries. However, instead of accumulating votes in the parameter space of a curve, as is done in a Hough transform, this process accumulates votes in the space of points that a curve is likely to pass through. Centerline curves are then fit to this image using a randomized sampling procedure described in the following section.

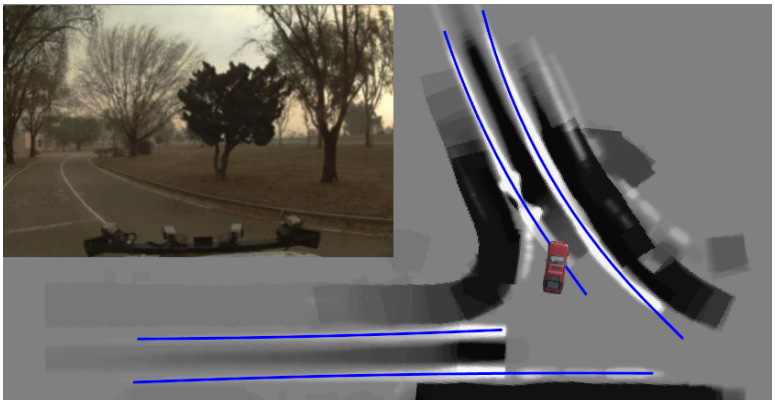


Figure 3-2: Lane centerline candidates (blue) are identified by fitting parabolic segments to the ridges of the evidence image (white). Front-center camera view is shown in top left for context.

3.1.2 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 [20] fits parabolic segments to the ridge points (Figure 3-2). 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 $\mathbf{C}$ [25], 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 $\alpha = 1\text{m}$. The parabola is then re-fit to L using a linear least squares method, and 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. Each candidate lane centerline is then discretized into a piecewise linear curve and transmitted to the lane tracker for further processing.

3.2 Lane Tracking

The primary purpose of the lane tracker is to maintain a stateful, smoothly time-varying estimate of the nearby travel lanes. 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. In the context of the Urban Challenge, the road map was provided as a Route Network Description File (RNDF) [1]. The RNDF can be treated as a directed graph, where each node is a waypoint in the center of some 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 the 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. Each lane estimate is extended and truncated as the vehicle approaches and withdraws from waypoints in that 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 fit to the evidence image.

As centerline candidates are generated, the lane tracker attempts to match each candidate with a tracked lane. Successfully matched centerline candidates are used to update the lane estimates. 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. The match score $m(c, l)$ is defined 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 matching so long as sufficient overlap is observed elsewhere.

Centerline candidates matched to a tracked lane are 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 3-3). 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 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.

(a) Two lane centerline priors derived from a road map

(b) Candidate lane centerlines estimated from sensor data

(c) Filtered and tracked lane centerlines

Figure 3-3: (a) The road map provides weak *a priori* lane centerline estimates (white) that may go off-road. (b) Lane centerline estimates (blue) fit to locally observed road-paint and curb features. (c) Sensor-derived estimates are filtered, tracked, and fused with the road map priors to produce the final lane centerline estimates (white).

3.3 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, or unintentionally enter or exit a travel lane [28, 29]. We note that the output of the lane tracking system was used directly to guide the vehicle's motion planning systems. Had lane tracking system yielded an incorrect estimate, our vehicle would have traveled along that estimate, possibly into an oncoming traffic lane or off-road.

3.3.1 System Confidence

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

At any instant, our system can either have no confidence in its estimate 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 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 3.1).

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%)

Table 3.1: Distance traveled with high-confidence estimates in current travel lane.

From this, we see that our vehicle maintained some high-confidence estimate 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.

A second way of assessing the system's performance is by examining its estimates as a function of location within the course. Figure 3-4 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.

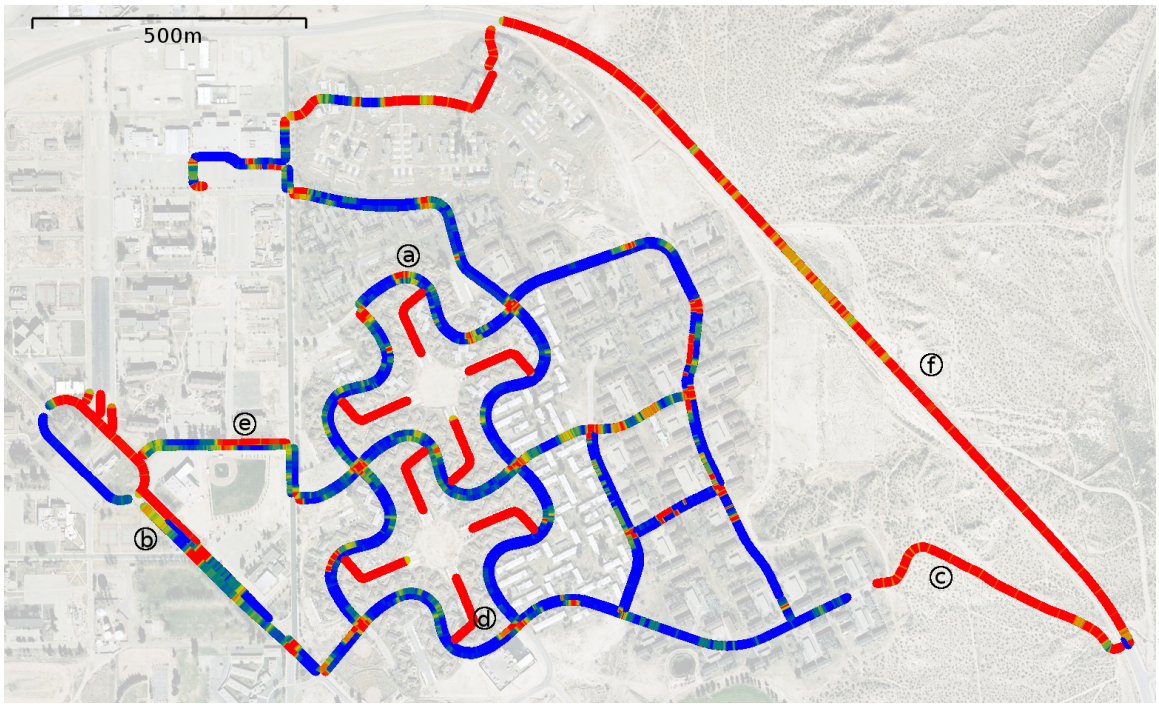


Figure 3-4: Aerial view of the Urban Challenge race course in Victorville, CA. Autonomously traversed roads are colored blue in areas where the lane tracking system reported high confidence, and red in areas of low confidence. Some low-confidence cases are expected, such as at intersections and areas with no clear lane markings. Failure modes occurring at the circled letters are detailed further in Fig. 3-5.

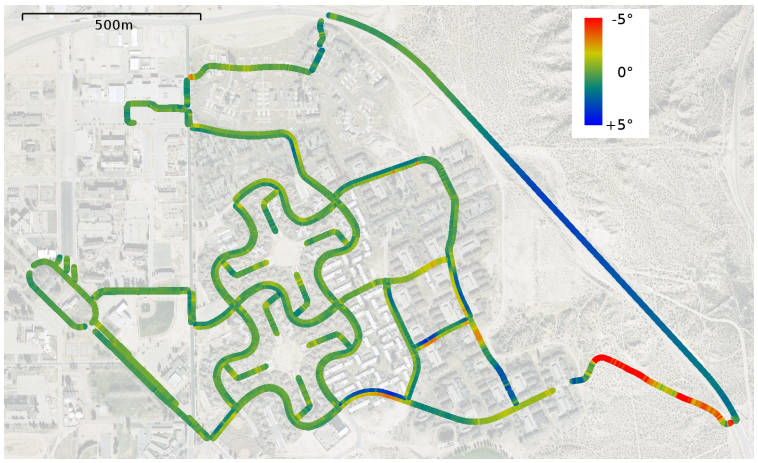
Images from our logged camera images at typical failure cases are shown in Figure 3-5 (with locations at which these failures occurred marked in Figure 3-4). A common failure mode was an inability to detect road paint in the presence of dramatic lighting variation such as that caused by cast tree shadows. 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

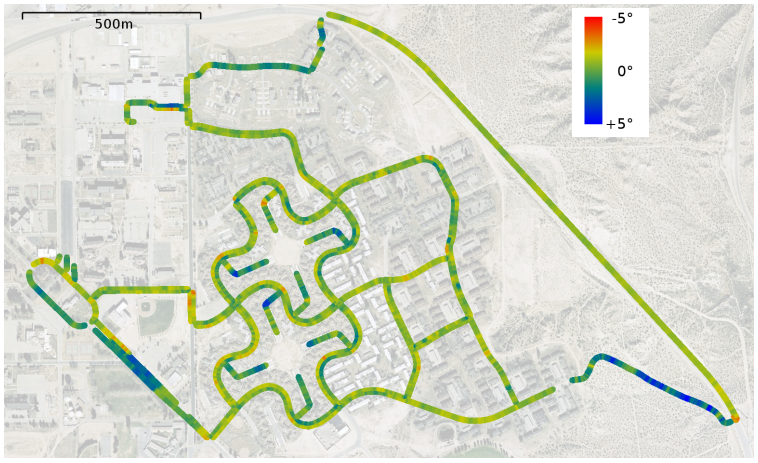
Figure 3-5: Common failure cases (cf. Fig. 3-4). The most common failure was in areas with strong tree shadows, as in (a) and (b). Dirt roads, and those with faint or no road paint (c-e) were also common causes of failure. In (f), a very wide lane and widely-spaced dashed markings were a challenge due to our strong prior on lane width. In each of these failure situations, the system reported no confidence in its visual estimates.

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 poorly 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 the road paint detection projected onto an assumed level ground plane had sufficient projection error that no stable centerline evidence images were created. Mean vehicle pitch and roll can be seen in Figure 3-6. 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 [15].



(a) Mean pitch



(b) Mean roll

Figure 3-6: Mean vehicle pitch and roll throughout the Urban Challenge. Positive pitch and roll values correspond to vehicle nose up and left side up, respectively.

3.3.2 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, for every moment of the race, the geometry of nearby travel lanes expressed in the vehicle's local reference frame. Appendix A gives a detailed description of how the ground truth data set was created; we provide a brief overview here.

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. We then use the vehicle's GPS estimates to provide an initial guess as to the vehicle's pose within the ground truth. 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.

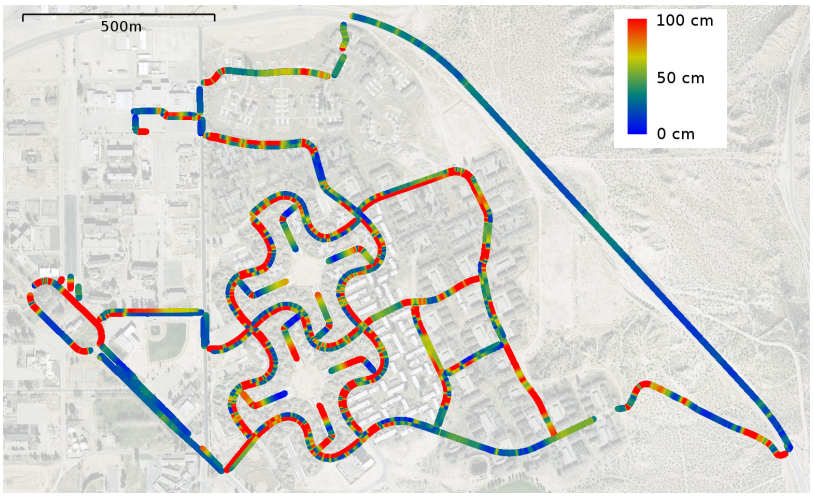
Generating ground truth lane geometry in this manner allows us to conduct a number of experiments that would otherwise be impossible. We can assess the performance of our system against a number of different variables, and see how using it compares to using the RNDF alone without any local perception. 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, hidden or unknown experimental bias may affect the ground truth labels. In some cases the true lane centerline geometry may be have some ambiguity (e.g., one when one lane boundary is unmarked), but we believe it is nevertheless highly useful.

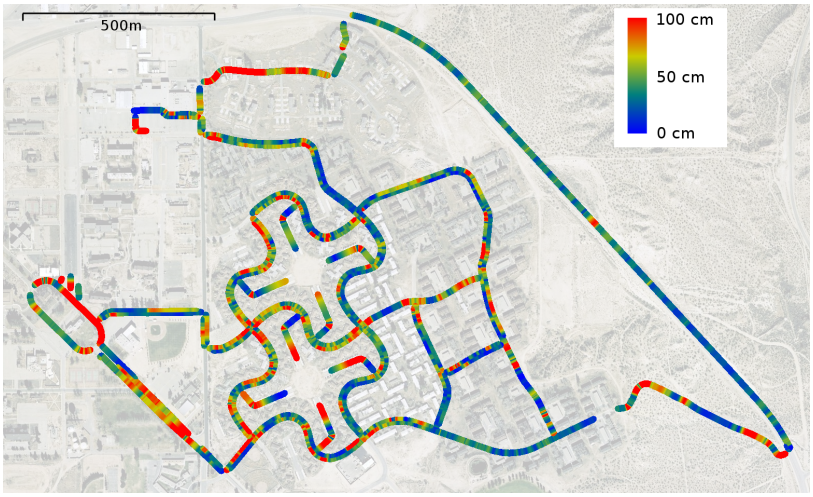
3.3.3 Centerline Error

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 attempts to track the lane centerline [45], so the centerline error is a fair measure of the overall system accuracy.

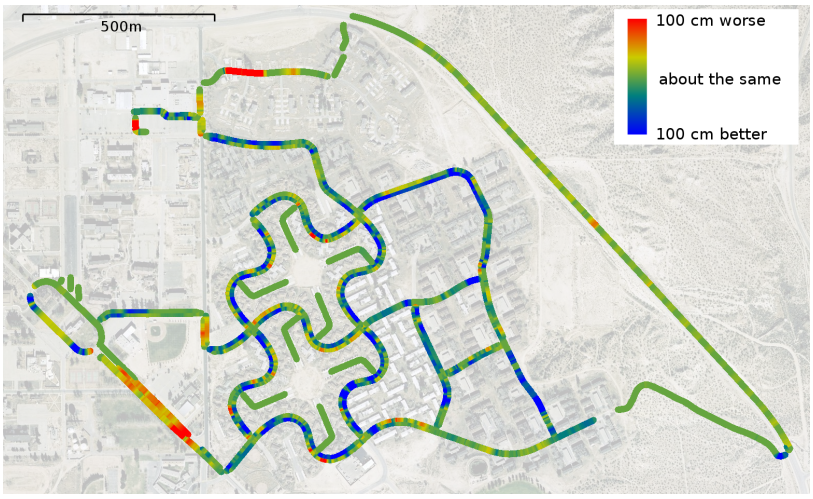
Given that the resolution of our sensors decreases with distance from the vehicle, we expect the accuracy of the lane estimates to decrease the farther away they are. Slight errors in time synchronization and projective errors resulting from small angular errors also contribute to this error. To confirm 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

Figure 3-7: Mean centerline error 10 m in front of vehicle. (a) RNDF error is also dependent on GPS receiver error. (b) Lane tracker fuses vision + laser range data to improve RNDF estimates. (c) In most cases, our system is as good as or better than using the RNDF alone.

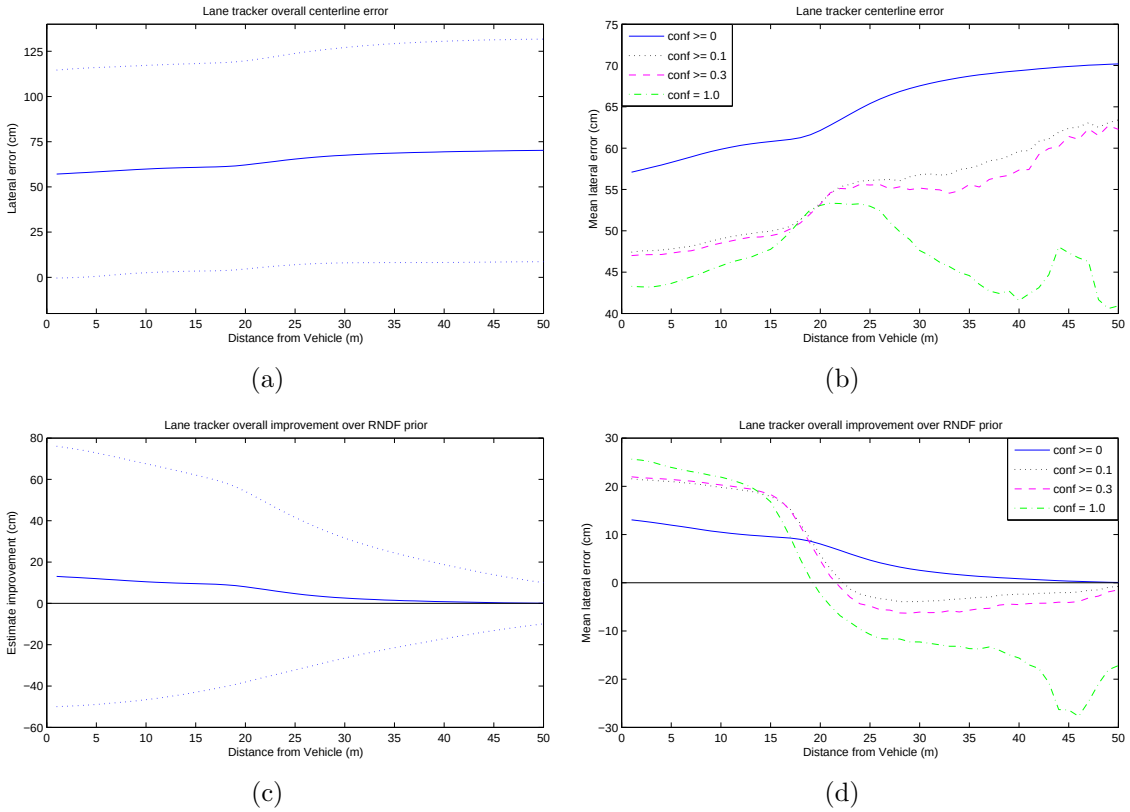


Figure 3-8: Centerline error as a function of distance along lane, averaged along DUC course. (a) Mean error (blue) with 1-σ bounds (dashed). (b) As system confidence increases, its accuracy improves. (c) Mean improvement over the RNDF with 1-σ bounds. (d) Mean improvement over the RNDF by system confidence. For high-confidence estimates our system outperformed the RNDF only at close ranges.

The results of this analysis are shown in in Figure 3-8. Immediately in front of the vehicle, the mean centerline error was 57cm, which gradually increased to 70cm 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 decreases to 43-53 cm (Figure 3-8b).

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 each lane boundary. A mean centerline error of 57cm 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, producing a

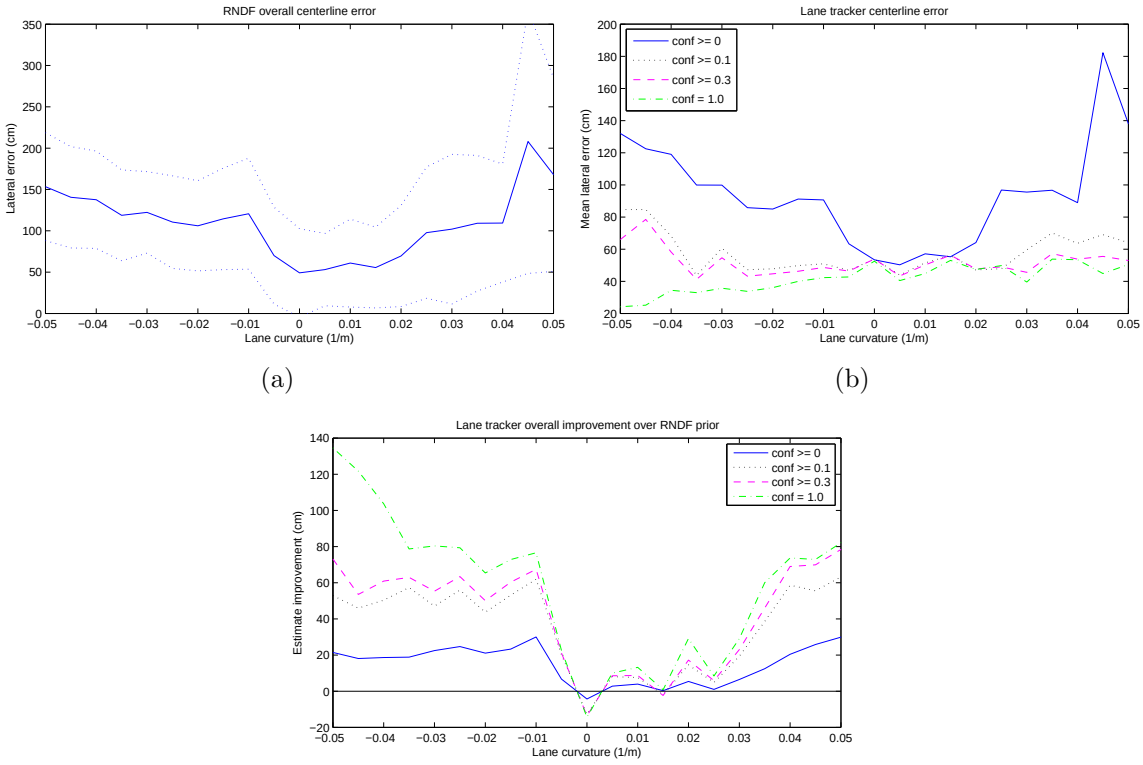
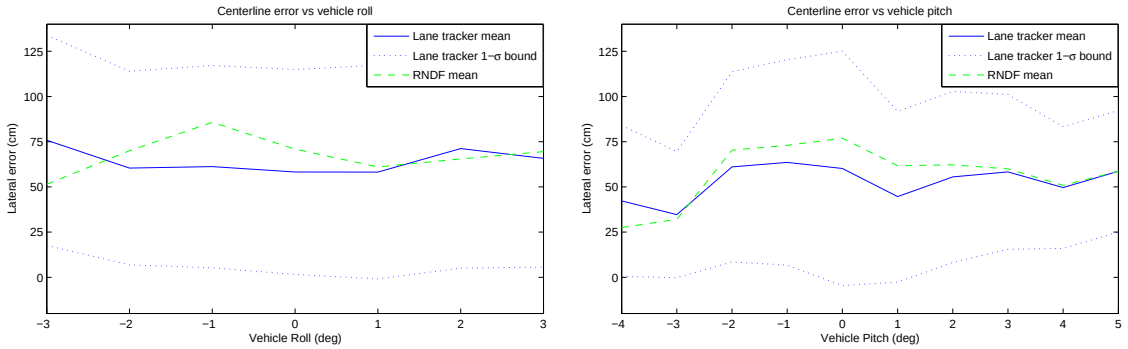


Figure 3-9: Mean centerline error as a function of lane curvature. (a) The RNDF experiences greater error (blue) in high-curvature areas. 1-σ bounds are shown (dashed). (b) Lane tracker centerline error for several confidence thresholds. (c) Our system is able to significantly improve RNDF estimates in high-curvature areas.

centerline estimate that was consistently 1.83 m away from one boundary (our prior on half a lane width), but which 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 Figures 3-8c and 3-8d. Overall, the system is modestly better than using the RNDF alone, with a mean improvement of 10cm 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 22cm. 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 Figure 3-9a, 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. Figure 3-7a). The distance



(a) (b)

Figure 3-10: Centerline error of our system and the RNDF prior as a function of vehicle roll and pitch, with 1-σ bounds.

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.

In contrast, our system excelled in areas of high curvature, as shown in in Figures 3-9b and 3-9c. 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 road curvatures of $0.05\,\mathrm{m}^{-1}$, mean improvements over the RNDF for high-confidence estimates were almost 1 m.

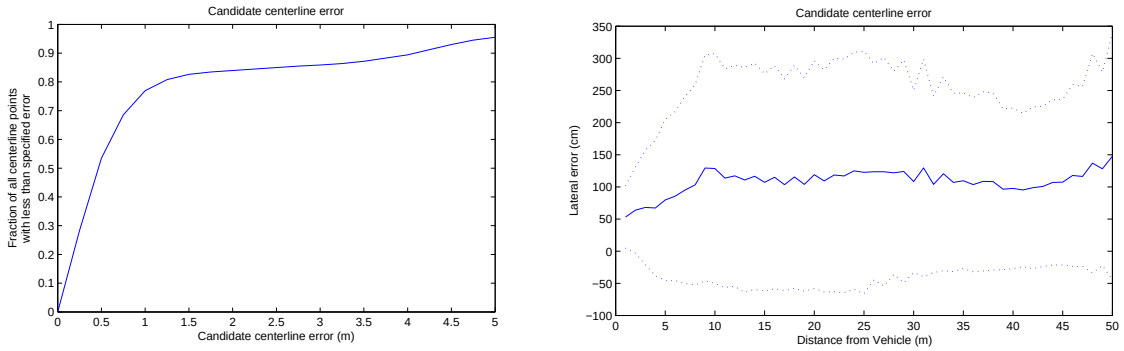
Lastly, we evaluate lane centerline error against vehicle roll and pitch. Mean vehicle roll and pitch are shown in Figure 3-6, and error plots for these factors are given in Figure 3-10. Sustained roll and pitch are good indicators of non-level terrain, and we expect worse performance in these cases in light of our level ground-plane assumption. As mentioned earlier, our implementation did not account for these scenarios, but doing so even crudely by estimating a non-level ground plane should help.

Figures 3-10 shows 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 as roll and pitch increase.

3.3.4 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 out-



(a) (b)

Figure 3-11: (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.

put 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 3-11a shows the error distribution for candidate centerlines: 53.5% of points on all candidate centerlines were within 50cm 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 Figure 3-11b. Centerline candidates generated very close to the vehicle had the least error, with a mean error of 53cm at distances of 1-2 m. At a distance of 10-11 m, mean centerline error was 129cm. 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 for 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.

3.4 Summary

This chapter presented a modular, scalable, perception-centric lane detection and tracking system that fuses asynchronous heterogeneous sensor streams with a weak geometric 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.

Despite these properties, this 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. Nor can it handle adverse conditions such as rain and snow that wreak havoc with the feature detectors. Our analysis has revealed cases in which simplifying assumptions on lane width and ground surface adversely affected system performance. Although the evidence image approach can be used to robustly estimate lane geometry, it fails in situations where many false detections prevent construction of a stable evidence image.

Chapter 4

Basis Curves

The estimation of road and lane geometry can be characterized as a curve estimation problem, where feature detectors produce partial observations of the curves constituting the centerline and boundary curves. While many features are true observations of the road and lanes, some features correspond to tree shadows or lens flare, and others may simply be painted road lines that do not correspond to lane boundaries (e.g., crosswalk stripes, parking spot delimiters, or stop lines).

The general task of estimating and tracking the hidden state of one or more objects using a sequence of noisy observations, possibly containing outliers, has been well studied in other domains using probabilistic techniques. In particular, the ability of probabilistic methods to robustly handle noisy data in a well-modeled estimation problem has contributed greatly to their success in estimation and tracking problems [69, 5]. Even when the objects in question do not actually conform to the distributions used the model them, probabilistic approaches often provide sufficiently accurate approximations to be used with great success. This has not been overlooked in lane estimation [53, 39, 67]; there, the most popular lane tracking technique is the Kalman filter [33].

This chapter describes a novel representation for open 2D curves, a probability distribution over curves using this representation, and two Bayesian algorithms for estimating curves from partial observations. This approach differs substantially from the evidence-image approach described in the previous chapter, and improves upon it in a number of ways. This chapter presents the algorithms as techniques for 2D curve estimation. Chapter 5 shows their application to the lane estimation problem.

In the following sections, we develop the concept of basis curves and describe a probability distribution over curves. The key innovation is the description of many curves as variations of a single basis curve. When formulated in this way, observations of a curve (not necessarily the basis curve) can be expressed as linear transformations of the true curve, and Gaussian distributions can robustly model much of the noise resulting from the sensing processes described in previous chapters. With this model, χ^2 tests can be used for data association and outlier rejection, and the Kalman filter update steps become appropriate for curve estimation and tracking. We justify the approximations that allow this formulation, and explore their consequences. Finally, we show how to estimate "bands," curves with variable width, using basis curves.

4.1 2D Curves

We represent a continuous parametric 2D curve $\mathbf{f}$ as:

$$\mathbf{f}(s) = \begin{bmatrix} f_x(s) \\ f_y(s) \end{bmatrix}, \quad s \in [s_1, s_n] \tag{4.1}$$

where $f_x(s)$ and $f_y(s)$ are the x and y coordinates of the curve, parameterized by a continuous scalar s, defined on the interval $s \in [s_1, s_n]$. The parameter s may represent a semantically meaningful value such as elapsed time or curve distance, or it may simply be an artifact of the curve model. Occasionally, it is useful to reparameterize the curve by defining a new function $\mathbf{f}'$ as:

$$\mathbf{f}'(t) = \mathbf{f}(h(t)), \quad t \in [t_1, t_m] \tag{4.2}$$

where $h(t)$ is a reparameterization function and $s_1 \leq h(t_1) \leq h(t_m) \leq s_n$, such that the new curve $\mathbf{f}'$ is spatially identical to or a subset of the original curve $\mathbf{f}$.

In the context of lane estimation, several objects of interest can be represented as parametric curves. Painted lane boundaries, physical road edges such as curbs, and the invisible lane or road centerlines can all be expressed as curves (Fig. 4-1). Curves can be on the order of meters for short streets or merge lanes, or hundreds of meters for stretches of highway when traveling at speed.

Figure 4-1: Left: Curves of interest in the context of lane estimation include painted lane boundaries (a), raised curbs (b), and (typically invisible) lane centerlines (c). Right: The spatial extent of curves can be much larger than a vehicle's sensor range; in many cases estimates of the entire curve are not necessary.

4.1.1 Curve Representation

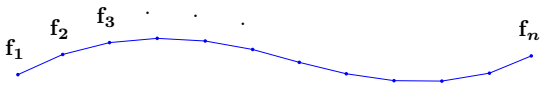
The general curve definition shown in Eq. (4.1) is computationally awkward, in the sense that representing a curve in this form requires storing an uncountably infinite number of values for $f_x(s)$ and $f_y(s)$ on the defined interval. Instead, most algorithms that operate on spatial curves choose a finite-dimensional space in which to represent the most useful set of curves for a particular application.

By far, the most common way to represent curves computationally is with some type of spline, where $f_x(s)$ and $f_y(s)$ are piecewise polynomial functions. The polynomials themselves are typically factored into a vector of control points and a set of basis polynomials, such that the entire spline can be parameterized by the control point positions and the polynomial coefficients. A variety of ways to express control points and basis polynomials gives rise to families of commonly used splines such as Bézier, B-Spline, Catmull-Rom, and Hermite splines [6]. While each of these splines can be defined by polynomials having arbitrary degree, quadratic and cubic splines are the most popular. Of particular note is the quadratic B-Spline, which was pioneered by Blake and Isard for tracking contours in computer vision using Kalman and particle filters [10].

A major advantage of quadratic and cubic splines is that they can express complex curves with a small number of control points. This has natural computational benefits, as it is often sufficient to work directly with the control points without evaluating the curve at every point. Additionally, a variety of spline representations allow for local control, where modification of one control point affects only a local region of the curve near the control point. Unfortunately, this compact representation also introduces complications in estimation and tracking settings.

The primary complication in working with quadratic and cubic splines is precisely that the mapping from control points to the actual curve geometry is nonlinear. As we reason about curve geometry, we will find it useful to express beliefs about the curve in the form of probability distributions over the curve geometry. Mapping these beliefs into the space of control points for quadratic and cubic splines is no longer analytically possible, and doing so numerically would forfeit the computational advantages gained by working with fewer control points.

Curves defined by a piecewise linear or piecewise constant curvature are analytically more attractive than splines [26], and especially attractive for highway modeling and architecture, but are also difficult to use computationally. Because they are defined by spatial derivatives, it is difficult to choose a representation in these forms that allows modification of local curve geometry.



$$\mathrm{F} = \begin{bmatrix} \mathbf{f}_1^\top \\ \mathbf{f}_2^\top \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{f}_n^\top \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_n & y_n \end{bmatrix}$$

Figure 4-2: We represent a piecewise linear curve of $n-1$ segments with an $n \times 2$ matrix of control points.

For this reason, we consider only piecewise linear curves (polylines), and reason directly about the control points defining the curve's many line segments. Computationally, this requires operating on many more control points than if we had used a quadratic or cubic spline, but the algorithmic simplicity that results from using this representation by far justifies this additional burden. Polylines can closely approximate all other curves with a sufficiently dense control point spacing, and changes to individual control points affects only local curve geometry.

Through the rest of this chapter, we represent a piecewise linear curve $\mathbf{f}$ by its $n \times 2$ matrix[1] of control points $\mathrm{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^\top$. A point $\mathbf{f}(s)$ on the curve can then be determined by simple linear interpolation of two neighboring control points (Fig. 4-2). We will refer to the curve alternately as either $\mathbf{f}$ or F, depending on context. Additionally, we assume some method of inferring first and second order spatial derivatives from the control points, such as by numerical approximation, or by fitting a quadratic or cubic spline.[2]

The goal of our method is to produce a useful estimate $\hat{\mathbf{f}}$ of a subset of $\mathbf{f}$, given a set of noisy partial observations. In the context of autonomous vehicles, we seek to estimate the region of $\mathbf{f}$ that lies within sensor range of our vehicle. We note that it is typically the case that the interval $[r_1, r_n]$ on which $\hat{\mathbf{f}}$ is defined is a subset of the interval on which $\mathbf{f}$ is defined. Thus, we define our estimate $\hat{\mathbf{f}}$ as:

$$\hat{\mathbf{f}}(s) = \begin{bmatrix} \hat{f}_x(s) \\ \hat{f}_y(s) \end{bmatrix}, \quad s \in [r_1, r_n] \tag{4.3}$$

and note that $s_1 \leq r_1 < r_n \leq s_n$.

4.1.2 Curve probability distributions

It is often useful to define a probability distribution over curve geometry. In estimation and tracking, the true form of a curve is unknown, and a probability distribution can be used to represent the system's belief about possible geometries of the curve. Noisy observations of a curve can also be modeled as curves drawn from a distribution.

Curve estimation algorithms that represent curves using a series of control points (e.g., the control points of a polyline or a B-Spline) typically define a distribution over control point positions as a way to represent distributions over curve geometry. The most straightforward approach is to model the control points as jointly Gaussian. For example, a distribution over polylines with n control points may be defined with a $2n \times 1$ mean vector and $2n \times 2n$ covariance matrix.

The major drawback of this method is that allowing the control points of the curve to vary in all directions usually provides too many degrees of freedom. Specifically, given a distribution defined in this manner, it is often possible to draw two curves from the distribution that have identical shapes but different probability densities

[1] The conventional way to represent a variable with $2n$ dimensions is with an $2n$-dimensional column vector. We deviate from this convention and choose a $n \times 2$ matrix for mathematical simplicity.

[2] We use, $\bar{\mathbf{b}}_i = \frac{\mathbf{b}_{i+1} - \mathbf{b}_i}{||\mathbf{b}_{i+1} - \mathbf{b}_i||}$, $i \in \{1, 2, \ldots, n-1\}$ and $\bar{\mathbf{b}}_n = \bar{\mathbf{b}}_{n-1}$.

(Fig. 4-3). In the case of a polyline with three collinear points, moving the middle control point longitudinally does not change the curve shape at all, but can change the probability density. Similar scenarios occur with quadratic and cubic splines.

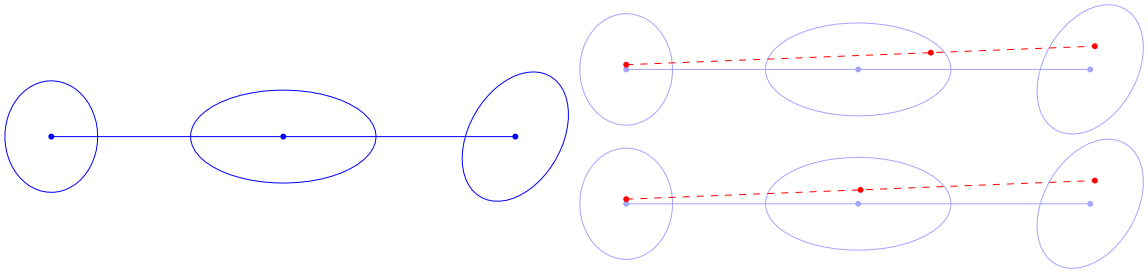


Figure 4-3: (left) A distribution over curves can be defined by a mean and covariance on control points. 1-σ ellipses shown. (right) Two samples (red dots) drawn from this distribution can have the same shape (dashed lines), but different probability densities.

The underlying cause of this difficulty is that given a curve model, there are often many ways to choose the control points such that the resulting curves have the same geometry. A consequence of this property is that evaluating the likelihood that an arbitrary curve is drawn from the distribution becomes a difficult process, as there can be many ways to represent the curve, each with a different likelihood. Thus, distributions over curve control points are not always appropriate for representing distributions over curve shape.

4.2 Basis Curves

In this section, we introduce the notion of basis curves for curve representation, and use it to define a probability distribution over curves. The insight behind this representation is that when reasoning about several curves together, it is often the case that the curves share a common structure. Factoring out the common structure from the curves leaves a set of residuals which are computationally convenient. To do so, we represent each curve in a set by its variation from the set's basis curve.

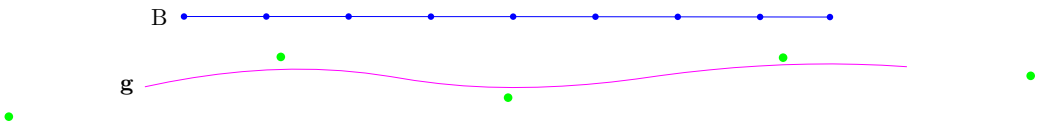
Consider an arbitrary piecewise linear curve $\mathbf{b}$, represented by the matrix of control points $\mathrm{B} = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)^\top$. Denote the unit normal vectors to the curve at each of the control points by the matrix $\bar{\mathrm{B}} = (\bar{\mathbf{b}}_1, \bar{\mathbf{b}}_2, \ldots, \bar{\mathbf{b}}_n)^\top$. We consider a curve $\mathbf{g}$, represented as $\mathrm{G} = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n)^\top$, to be a variation of $\mathbf{b}$ if each $\mathbf{g}_i, i \in \{1 \ldots n\}$, can be expressed as:

$$\mathbf{g}_i = \mathbf{b}_i + g_i^{\mathrm{B}} \bar{\mathbf{b}}_i \tag{4.4}$$

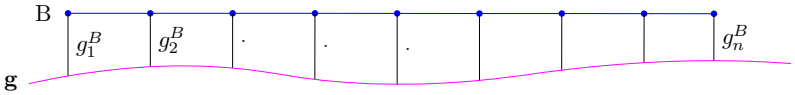
where each g_i^{B} is a signed scalar that indicates how much $\mathbf{g}$ varies from $\mathbf{b}$ along the normal vector at a particular control point. We refer to $\mathbf{b}$ in this context as a *basis curve*, and call the vector $\mathbf{g}^{\mathrm{B}} = (g_1^{\mathrm{B}}, g_2^{\mathrm{B}}, \ldots, g_n^{\mathrm{B}})^\top$ the *offset vector* of $\mathbf{g}$ from $\mathbf{b}$.

Here, we have defined $\mathbf{g}$ as a variation of the basis curve $\mathbf{b}$. Suppose $\mathbf{g}$ is not given to us in this form, and is instead expressed using a different curve representation, possibly a spline model or even a polyline with a different number of control points.

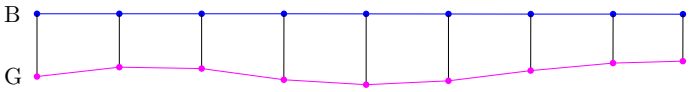
If **g** is geometrically similar to **b**, then we can choose a polyline approximation G of **g** that in turn approximates **g** as a variation of **b**. When the matrix of control points G is defined in this manner, we refer to the resulting offset vector $\mathbf{g}^{\mathrm{B}}$ as the *projection of* **g** *onto* **b**. Figure 4-4 illustrates the projection of a uniform quadratic B-Spline onto a basis curve.



(a) A basis curve (blue) represented by polyline B, and a uniform quadratic B-Spline **g**.



(b) The projection of **g** onto **b**, producing the g_i^{B}.



(c) Approximation of curve **g** as a variation of the basis curve.

Figure 4-4: A uniform quadratic B-Spline (magenta) defined by the green control points can be projected onto a basis curve (blue). Curves using other representations can be projected using the same intersection procedures.

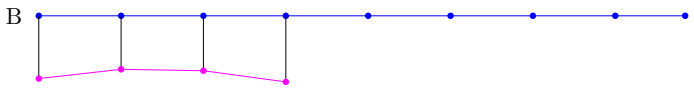
Although **g** itself is not a variation of **b**, the polyline approximation G is. The error of this approximation (e.g., as measured by the area between the polyline approximation and the original curve) is small if **g**'s geometry is similar to **b**, and if the control point spacing of **b** is small relative to the curvatures of the two curves. If the curves are sufficiently different (e.g., nearly orthogonal), it may not be meaningful to project one onto the other.

To review, thus far we have defined:

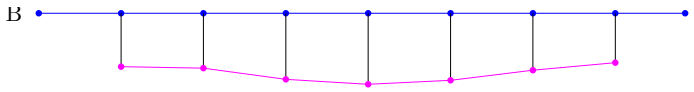
b	A basis curve.
B	The $n \times 2$ matrix of control points representing **b**.
$\mathbf{b}_i$	The i-th control point of B.
$\bar{\mathrm{B}}$	The $n \times 2$ matrix of unit normal vectors of **b** at each control point.
$\bar{\mathbf{b}}_i$	The i-th unit normal vector of $\bar{\mathrm{B}}$.
g	A curve geometrically similar to **b**.
G	The $n \times 2$ matrix of control points approximating or representing **g**.
$\mathbf{g}_i$	The i-th control point of G.
$\mathbf{g}^{\mathrm{B}}$	The projection of **g** onto **b**, also referred to as the offset of **g** from **b**.

We can now use $\mathbf{b}$ to represent common structure shared by many curves, and use the offset vectors to reason about the differences among curves. One interpretation of this is that $\mathbf{b}$ induces an n-dimensional vector space, into which we can project any curve. We call this space the *basis curve space* of $\mathbf{b}$. When projected into a basis curve space, a curve can be represented by a single point – its offset vector.

In general, any curve that spans the length of $\mathbf{b}$ can be projected onto $\mathbf{b}$. It is also useful to reason about curves that do not run the length of $\mathbf{b}$, but do have some "longitudinal overlap" in that they intersect with one or more of the normal vectors of $\mathbf{b}$. In this case, each of the $\frac{n(n-1)}{2}$ consecutive subsets of the control points of $\mathbf{b}$ can be treated as an individual basis curve with its own basis curve space. These smaller basis curve spaces are subspaces of the basis curve space of $\mathbf{b}$ (Figure 4-5).



(a) Variation of the first four control points of a basis curve



(b) Variation of a mid-section of a basis curve

Figure 4-5: Variations (magenta) of a subset of the basis curve (blue) can be described using subspaces of the basis curve vector space of the full basis curve.

4.2.1 Basis curve normal distributions

Next, we consider probability distributions on curves. Specifically, consider a random variation of a basis curve $\mathbf{b}$, where the offset vector $\mathbf{g}^{\mathrm{B}}$ is normally distributed according to $\mathbf{g}^{\mathrm{B}} \sim N(\boldsymbol{\mu}, \Sigma)$. Figure 4-6 shows a basis curve and several hundred variations, where the offset vectors of each variation are drawn from a multivariate Gaussian.

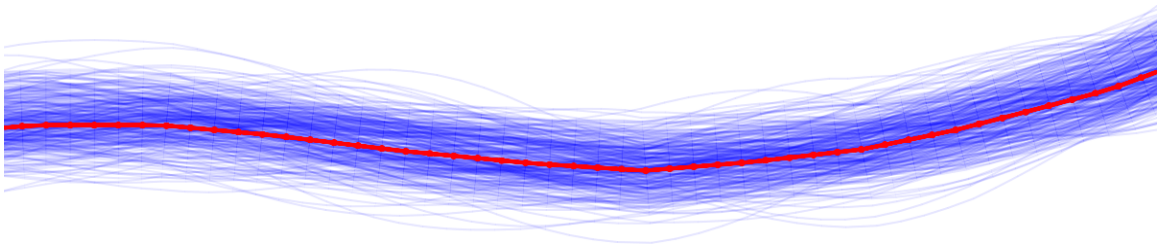


Figure 4-6: Several hundred normally distributed variations of a basis curve.

Together, the basis curve $\mathbf{b}$, mean vector $\boldsymbol{\mu}$, and covariance matrix Σ define a distribution over curves. We refer to a distribution of this form as a basis curve normal distribution, and represent it with the term $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$. The probability density of a curve $\mathbf{g}$ is then defined as:

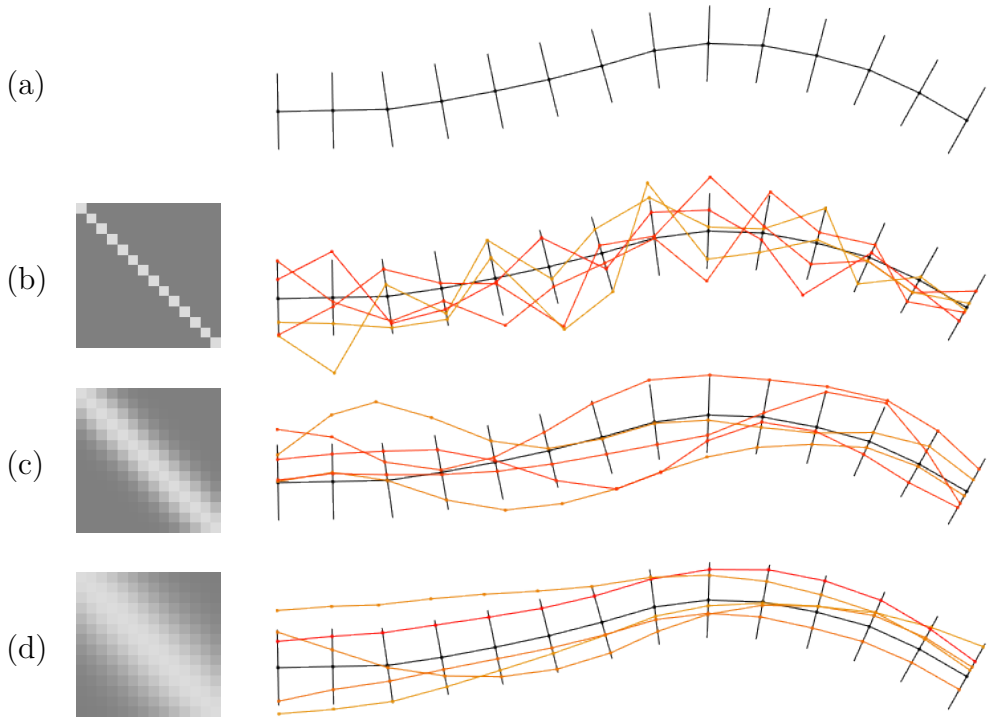


Figure 4-7: (a) A basis curve, with normal vectors shown at each control point. (b-d) Three covariance matrices, and five samples drawn from the distributions defined by the basis curve, $\boldsymbol{\mu} = \mathbf{0}$, and each of the covariances. Lighter colors indicate higher correlation.

$$P_{\mathbf{g}}(\mathbf{g}; \mathrm{B}, \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{g}^{\mathrm{B}} - \boldsymbol{\mu})^{\top}\Sigma^{-1}(\mathbf{g}^{\mathrm{B}} - \boldsymbol{\mu})\right) \tag{4.5}$$

An alternative intuition behind this formulation is that control point uncertainty is expressed only along the curve normals of the basis curve. Thus, each control point has one degree of freedom instead of two; even though G has $2n$ distinct components, it has only n degrees of freedom. The covariance matrix Σ represents the degree and structure of variation between curves drawn from the distribution. Fig. 4-7 illustrates this with samples drawn from three different distributions.

Since each control point has only one degree of freedom, two random curves drawn from a basis curve normal distribution with probability densities will also exhibit different geometries. A corollary to this is that evaluating the likelihood that an arbitrary curve is drawn from a distribution becomes a simple process of computing curve and ray intersections. There are some exceptions to this that may arise when the covariance at a control point is very large relative to the curvature, but this has not had a noticeable impact in our usage for lane estimation.

4.2.2 Changing basis curves

So far we have not discussed how to choose a basis curve, or why one particular basis curve is better than any other. In fact, it may often be the case that a curve distribution is defined using one basis curve, after which another basis curve becomes more appropriate to use than the original. In this case, we would like to switch the basis curves while changing the actual distribution as little as possible.

In general, it is not possible to match the original distribution exactly. In the degenerate case where the new basis curve does not have a projection onto the original basis curve, then there may not be a sensible reparameterization. However, if the new basis curve is similar to the original, then reasonable choices are possible. Here we consider two scenarios: 1) the new basis curve is a re-sampling of the original basis curve, and 2) the new basis curve is a variation of the original basis curve. In both cases, the approximation error is directly related to the amount by which the basis curve normal vectors change: approximation error is smallest when the new basis curve is locally parallel to the original basis curve.

Variation of the original basis curve

If $\mathbf{b}'$ is a variation of the original basis curve $\mathbf{b}$, then B′ can be described by:

$$\mathrm{B}' = \mathrm{B} + \mathrm{diag}(\mathbf{b'}^{\mathrm{B}})\bar{\mathrm{B}} \tag{4.6}$$

where $\mathbf{b'}^{\mathrm{B}}$ is the projection of $\mathbf{b}'$ onto $\mathbf{b}$. In this case, a new mean can be defined by simply subtracting the projection of the new basis curve, and the covariance can remain unchanged:

$$\begin{aligned} \boldsymbol{\mu}' &= \boldsymbol{\mu} - \mathbf{b'}^{\mathrm{B}} \\ \Sigma' &= \Sigma \end{aligned} \tag{4.7}$$

Re-sampling of the original basis curve

If $\mathbf{b}'$ is defined by a re-sampling of the control points of $\mathbf{b}$, then each of the m control points of its polyline representation B′ is a convex linear combination of two neighboring control points on the original curve B. Thus, B′ can be related to B by:

$$\mathrm{B}' = \mathrm{HB} \tag{4.8}$$

where H is a $m \times n$ matrix satisfying the condition that each row has at most two non-zero entries that are adjacent and sum to unity (representing the convex linear combination of neighboring control points). Additionally, $\mathrm{H}_{i,j}$ may be non-zero only if $\mathrm{H}_{i-1,k}$ is zero for all $k > j$ (control points may not be re-sampled out of order). Note that B′ has the same geometry as B, subject to truncation and re-sampling approximation.

Since every point on a polyline curve is either a control point or an interpolation of neighboring control points, it follows that we can define a new mean and covariance by applying the re-sampling transformation. Given a re-sampled basis curve control

matrix B′ defined above, the new mean and covariance are given by:

$$\begin{aligned} \boldsymbol{\mu}' &= \mathrm{H}\boldsymbol{\mu} \\ \Sigma' &= \mathrm{H}\Sigma\mathrm{H}^\top \end{aligned} \tag{4.9}$$

4.3 Estimation of 2D Curves

In this section, we develop a recursive algorithm for estimating an unknown curve $\mathbf{f}$. The algorithm takes as input an initial curve estimate and a noisy observation of the curve, and produces an updated estimate that reflects the novel information provided by the observation. As suggested by previous sections, the uncertainty surrounding the estimates and observations are expressed using basis curve normal distributions. Using this representation, we show how the Kalman filter can be used to update the curve estimates.

We refer to our initial belief about $\mathbf{f}$ using the basis curve normal distribution $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$. The mean estimate of our algorithm, which we refer to as $\hat{\mathbf{f}}$ and describe by the matrix of control points $\hat{\mathrm{F}} = (\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \ldots, \hat{\mathbf{f}}_n)^\top$, can be obtained from B and $\boldsymbol{\mu}$:

$$\hat{\mathbf{f}}_i = \mathbf{b}_i + \mu_i \bar{\mathbf{b}}_i \tag{4.10}$$

The true curve $\mathbf{f}$ has a projection onto $\mathbf{b}$, denoted by $\mathbf{f}^{\mathrm{B}} = (f_1^{\mathrm{B}}, f_2^{\mathrm{B}}, \ldots, f_n^{\mathrm{B}})^\top$. Given a fixed basis curve $\mathbf{b}$, the vector $\boldsymbol{\mu}$ is effectively an estimate of $\mathbf{f}^{\mathrm{B}}$. In this sense, there are two estimates: the estimate $\boldsymbol{\mu}$ of $\mathbf{f}^{\mathrm{B}}$, and the estimate $\hat{\mathbf{f}}$ of $\mathbf{f}$, where $\hat{\mathbf{f}}$ is defined in terms of $\boldsymbol{\mu}$ and $\mathbf{b}$ as shown in Eq. (4.10).

Our estimates are themselves random variables, and if our model is correct, the expected values of these estimates are the true values. As is standard practice in estimation, we often use the estimates in place of the true values.

4.3.1 Initial Estimate

We assume that an initial estimate of a curve is available in the form of a basis curve normal distribution. Since the output of this algorithm is itself a basis curve normal distribution, the output can be used as input to a subsequent update step. However, the first estimate (prior to any updates) must still be obtained.

In our usage, a single road paint or curb feature as produced by a feature detector Chapter 2 is used to initialize a curve estimate. The covariance could be set as a function of sensor resolution, time synchronization confidence, and other characteristics that may affect the error of the initial feature detections. Chapter 5 describes our implementation in greater detail. Other methods for acquiring an initial estimate may include using a GPS prior to load curves from georeferenced map, or having a human driver could press a button indicating the vehicle is in a particular position with respect to a lane.

To reduce approximation errors resulting from projection onto the basis curve, the basis curve $\mathbf{b}$ chosen for estimation should be geometrically similar to the true curve. In this sense, $\mathbf{b}$ can itself be considered as a form of the estimate. If this

is the case, then why use a basis curve at all? The basis curve can be treated as a reference frame to use *during* an update cycle. In this respect, projection of curves onto a basis curve is comparable (but not identical) to the linearization process of an Extended Kalman Filter, whereby the process and observations are linearized about the mean estimates, and then reasoned about in the linearized frame until the end of the update cycle. Instead of linearizing via a Taylor series approximation, curves are approximated by projection into the basis curve space.

4.3.2 Observations

Working with the general parametric form of a curve, we define a noisy observation $\mathbf{z}$ of $\mathbf{f}$ as:

$$\mathbf{z}(u) = \mathbf{f}(u) + v(u)\bar{\mathbf{f}}(u), \quad u \in [u_{z0}, u_{z1}] \tag{4.11}$$

where $v(u)$ is a zero-mean scalar noise term and $\bar{\mathbf{f}}(u)$ is the unit normal vector of $\mathbf{f}$ at u. Thus, $\mathbf{z}$ is a random curve whose probability distribution is determined by $\mathbf{f}$ and $v(u)$. It is typically the case that $\mathbf{z}$ is only a partial observation of $\mathbf{f}$, such that $s_1 \leq u_{z0} < u_{z1} \leq s_n$. We additionally assume that the noise term $v(u)$ is independent of the noise terms for other observations of the curve.

If we project $\mathbf{z}$ onto $\mathbf{b}$ and refer to the piecewise linear approximation of $\mathbf{z}$ as $\mathrm{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)^\top$, then each $\mathbf{z}_i$ can be expressed as:

$$\mathbf{z}_i = \mathbf{b}_i + (f_i^{\mathrm{B}} + v_i)\bar{\mathbf{b}}_i \tag{4.12}$$

where we model the noise terms $\mathbf{v} = (v_1, v_2, \ldots, v_n)^\top$ as jointly Gaussian with zero mean and covariance R. Since the projection $\mathbf{z}^{\mathrm{B}} = (z_1^{\mathrm{B}}, z_2^{\mathrm{B}}, \ldots, z_n^{\mathrm{B}})^\top$ of $\mathbf{z}$ onto $\mathbf{b}$ is defined such that:

$$\mathbf{z}_i = \mathbf{b}_i + z_i^{\mathrm{B}}\bar{\mathbf{b}}_i \tag{4.13}$$

it follows that each z_i^{B} has the form:

$$z_i^{\mathrm{B}} = f_i^{\mathrm{B}} + v_i \tag{4.14}$$

and we can express $\mathbf{z}^{\mathrm{B}}$ as:

$$\begin{aligned} \mathbf{z}^{\mathrm{B}} &= \mathbf{f}^{\mathrm{B}} + \mathbf{v} \\ &= \mathrm{A}\mathbf{f}^{\mathrm{B}} + \mathbf{v}, \quad \mathbf{v} \sim N(\mathbf{0}, \mathrm{R}) \end{aligned} \tag{4.15}$$

where the observation matrix $\mathrm{A} = \mathrm{I}_{n\times n}$ is a trivial linear transformation. For the moment, we have assumed that $\mathbf{z}$ is a full observation of $\mathbf{f}$, such that every normal vector of B intersects the observed curve. We address partial observations shortly, but first discuss how an observation can be parameterized in the form of Eq. (4.15).

Typically, an observation will not be generated as a variation of an existing basis curve with a neatly specified distribution, but will simply be provided as an independent distribution over curves with its own basis curve. For example, a road paint detector may report a strip of road paint with some distribution over its geometry. Other "black box" feature detectors may also produce observation curves, each

characterized with its own basis curve normal distribution.

In this case, the distribution for the observation curve can be reparameterized such that its basis curve coincides with the basis curve $\mathbf{b}$, and may then be reasoned about as an observation of $\mathbf{f}$. Reparameterizing an observation in this way introduces some approximation errors, since the curve distribution almost always changes as the result of reparameterization. However, if $\mathbf{z}$ is actually an observation of $\mathbf{f}$, then the approximation error is small, since $\mathbf{z}$ is simply a noise-corrupted version of $\mathbf{f}$.

Finally, we consider the case of partial observations. It is often the case that the observation $\mathbf{z}$ does not span the full length of the curve $\mathbf{f}$. Consequently, there will be normal vectors of the basis curve $\mathbf{b}$ that do not intersect the observation. A partial observation of this sort can still be related to $\mathbf{f}$ by choosing an observation matrix A of size $m \times n$, where $m \leq n$ and A has a $m \times m$ identity sub-matrix, and is zero everywhere else. For example, if the basis curve has three control points and the observation curve spans only the first 2 of these control points, then the corresponding observation matrix $\mathrm{A}_{2\times 3}$ is:

$$\mathrm{A}_{2\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.16}$$

4.3.3 Data Association

Given a belief over $\mathbf{f}$ parameterized by $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$, and an observation $\mathbf{z}$ described by $\tilde{N}(\mathbf{b}, \mathbf{z}^{\mathrm{B}}, \mathrm{R})$, we would like to determine whether $\mathbf{z}$ is an observation of $\mathbf{f}$, or an observation of a different curve.

To do so, we use a standard outlier rejection approach consisting of a statistical test, where the null hypothesis is that $\mathbf{z}$ does indeed correspond to an observation of $\mathbf{f}$. Define the scalar random variable y as the following Mahalanobis distance [5]:

$$y = (\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu})^{\top}(\mathrm{R} + \mathrm{A}\Sigma\mathrm{A}^{\top})^{-1}(\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu}) \tag{4.17}$$

If the null hypothesis is true (i.e., $\mathbf{z}$ is an observation of the curve $\mathbf{f}$), then $\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu}$ has zero mean, and y obeys a χ^2 distribution with m degrees of freedom. By computing a p-value on y and applying a standard goodness-of-fit test, we can determine if $\mathbf{z}$ is an observation of $\mathbf{f}$, or if it is an observation of a different curve.

When simultaneously estimating and tracking multiple curves, we can apply a greedy matching procedure, in which each observation is associated with the curve that best "explains" that observation. Other data association techniques, such as joint-compatibility branch and bound (JCBB), may also be used [54]. If no tracked curve is likely to have generated the observation according to the current estimates, then the observation could be discarded, used to initialize a new curve for tracking, or incorporated into a higher-level reasoning system.

Algorithm `data_association_test` summarizes the basic data association test for a single observation and a single curve estimate. As with most outlier rejection procedures based on a χ^2 test, this procedure may fail when an outlier is highly similar to an inlier. For example, if a false detection on a tree shadow runs parallel to the road, then its geometry may be highly similar to the true road paint. In this and

similar situations, more sophisticated outlier rejection based on other criteria such as feature appearance may be useful.

Algorithm `data_association_test`

Input: an initial belief $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$
Input: an observation $\tilde{N}(\mathbf{b}, \mathbf{z}^{\mathrm{B}}, \mathrm{R})$
Input: a transformation matrix A relating the observation to the initial estimate
Output: `accept` or `reject` the observation

1: Let m be the dimensionality of $\mathbf{z}^{\mathrm{B}}$
2: Let p be the outlier threshold (e.g., 0.05)
3: Let k be the $1 - p$ percentile of the χ^2_m distribution
4: **if** $(\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu})^\top (\mathrm{R} + \mathrm{A}\Sigma\mathrm{A}^\top)^{-1} (\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu}) > k$ **then**
5: **return** `reject`
6: **else**
7: **return** `accept`

4.3.4 Curvature Prediction

The data association procedure described above is applicable only when the observation and tracked curve have some longitudinal overlap. It is often the case that a true observation $\mathbf{z}$ has no overlap with the current estimate $\hat{\mathbf{f}}$. Consider the case of tracking a lane boundary marked with a dashed line; when a new dash is observed, we would like to associate it with the existing curve estimate.

Intuitively, if we have observed one portion of a curve, we can reliably predict the shape of the nearby unobserved parts of the curve. The direction and curvature of lane boundaries do not change very rapidly, and are governed by the physical limitations of the vehicles for which they were designed. In order to predict the geometry of one part of a curve given an estimate of another part, we must first have a model of how the curve evolves over space.

One way of obtaining this model for the task of road and lane estimation is to utilize existing databases containing the geometry of known road networks. For example, the Commonwealth of Massachusetts publishes a dataset containing the geometry of more than 61,000 km of public roads in the state, produced by manual annotation of ortho-rectified aerial imagery [13]. We fit a simple first-order Markov model to this data set to produce a generative model of road curvature. Specifically, given the signed curvature at one point on a lane boundary, this model predicts the signed curvature of the lane boundary one meter farther along the curve (Fig. 4-8). Higher-order models can be expected to improve prediction accuracy and reduce prediction variance, but risk overfitting.

Using this curve model, we can extend both our belief over $\mathbf{f}$ and the observation $\mathbf{z}$. If the original observation $\mathbf{z}$ was reasonably close to the original curve estimate, but did not actually have any longitudinal overlap, then the extensions may have

enough overlap to robustly determine if the two correspond to the same underlying curve (Fig. 4-9).

Curve prediction can be used to extend the existing curve estimate even when there is already overlap between the observation and the curve estimate. As a vehicle travels and observes new portions of the road and its lanes, it can augment its curve estimates using a curvature prediction model, then use the augmented curve estimates when incorporating new observations. This is similar to state augmentation methods used in simultaneous localization and mapping techniques [65], where the state vector is augmented as new features are added to the map.

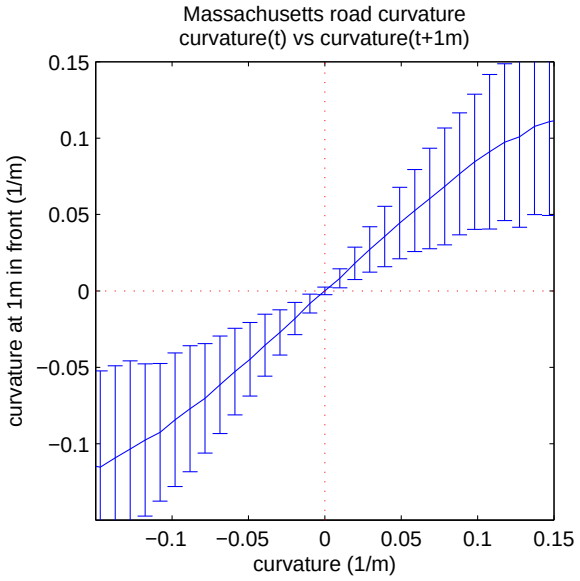


Figure 4-8: A road curvature prediction model, fit to MassGIS data on public roads in the Commonwealth of Massachusetts. 1-σ bounds are shown.

4.3.5 Update

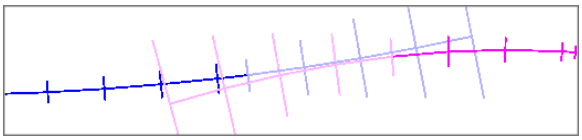
Once an observation $\mathbf{z}$ has been associated with an existing curve estimate, we use $\mathbf{z}$ to update our estimate, according to the update step of a standard Kalman filter [5]. The updated mean and covariance, which we denote as $\tilde{\boldsymbol{\mu}}$ and $\tilde{\Sigma}$, are given by:

$$\begin{aligned} \mathrm{K} &= \Sigma \mathrm{A}^\top (\mathrm{A}\Sigma\mathrm{A}^\top + \mathrm{R})^{-1} \\ \tilde{\boldsymbol{\mu}} &= \boldsymbol{\mu} + \mathrm{K}(\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu}) \\ \tilde{\Sigma} &= (\mathrm{I} - \mathrm{K}\mathrm{A})\Sigma \end{aligned} \tag{4.18}$$

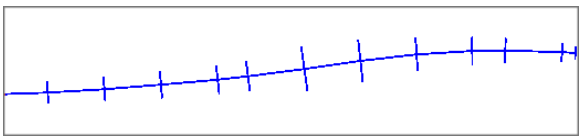
Intuitively, this update allows our estimate of $\mathbf{f}$ to shift a control point only along the basis curve curve normal at that control point. The amount a control point changes is determined by the uncertainties of the original curve and of the observation.

As mentioned earlier, approximation errors resulting from projection onto a basis curve are minimized when the basis curve geometry matches the true curve geometry. Therefore, we reparameterize the basis curve normal distribution defining the curve

(a) Curve estimate (left) and a non-overlapping observation (right)



(b) Predicted extensions of both curves



(c) Final curve estimate

Figure 4-9: A curvature prediction model allows us to associate observations with existing curve estimates when there is no overlap. Short line segments perpendicular to each curve indicate 1-σ uncertainty.

estimate, so that the basis curve coincides with the newly updated maximum likelihood curve estimate. Since this estimate is a variation of the current basis curve, reparameterization consists of offsetting the basis curve by the mean vector, then setting the mean vector to zero, as described in Sec. 4.2.2.

Algorithm `update_curve_estimate` shows the full estimation algorithm. It takes as input the initial belief $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$ over a single curve, and an observation. If the observation passes the data association test, then it is used to generate the updated curve estimate $\tilde{N}(\mathbf{b}^+, \boldsymbol{\mu}^+, \Sigma^+)$.

4.4 Bands – curves with width

All the curves described up to this point have been considered independently of their width, or as if they had no width. In some cases, we may find it useful to reason about the width of a curve in addition to its skeletal structure. For example, if we consider a road or a lane to be a curve with nonzero width, then an autonomous vehicle may find it very useful to maintain an estimate of the curve width in addition to its centerline, so that it can deduce the lane's left and right boundaries.

In this section, we extend our curve estimation algorithm to estimate and track the width of a curve. We now define a *band* $\mathbf{f}$ as a parametric curve with a width term that varies along the curve:

$$\mathbf{f}(s) = \begin{bmatrix} f_x(s) \\ f_y(s) \\ f_h(s) \end{bmatrix}, \quad s \in [s_1, s_n] \tag{4.19}$$

Algorithm `update_curve_estimate`

Input: an initial belief $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$
Input: an observation $\tilde{N}(\mathbf{b}, \mathbf{z}^{\mathrm{B}}, \mathrm{R})$
Input: a transformation matrix A relating the observation to the initial estimate
Output: the new belief $\tilde{N}(\mathbf{b}^{+}, \boldsymbol{\mu}^{+}, \Sigma^{+})$

Augment the initial estimate and observation as necessary via curve prediction
if `data_association_test`$(\mathbf{b}, \boldsymbol{\mu}, \Sigma, \mathbf{z}^{\mathrm{B}}, \mathrm{R}, \mathrm{A})$ is `reject` **then**
  Reject observation as an outlier.
  return $\tilde{N}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$
$\mathrm{K} \leftarrow \Sigma\mathrm{A}^{\top}(\mathrm{A}\Sigma\mathrm{A}^{\top} + \mathrm{R})^{-1}$
$\tilde{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu} + \mathrm{K}(\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu})$
$\tilde{\Sigma} \leftarrow (\mathrm{I} - \mathrm{KA})\Sigma$
$\mathbf{b}^{+} \leftarrow \mathrm{B} + \mathrm{diag}(\tilde{\boldsymbol{\mu}})\bar{\mathrm{B}}$
$\boldsymbol{\mu}^{+} \leftarrow \mathbf{0}$
$\Sigma^{+} \leftarrow \tilde{\Sigma}$
return $\tilde{N}(\mathbf{b}^{+}, \boldsymbol{\mu}^{+}, \Sigma^{+})$

where $f_x(s)$ and $f_y(s)$ denote the centerline of the band, and $f_h(s) > 0$ indicates the *half-width* of $\mathbf{f}$ at s. Defining $\bar{\mathbf{f}}(s) = (\bar{f}_x(s), \bar{f}_y(s))^{\top}$ as the unit normal vector to the centerline at s, the *left boundary*[3] of the band can be expressed as:

$$\mathbf{f}_l(s) = \begin{bmatrix} f_x(s) + f_h(s)\bar{f}_x(s) \\ f_y(s) + f_h(s)\bar{f}_y(s) \end{bmatrix} \tag{4.20}$$

and the *right boundary* as:

$$\mathbf{f}_r(s) = \begin{bmatrix} f_x(s) - f_h(s)\bar{f}_x(s) \\ f_y(s) - f_h(s)\bar{f}_y(s) \end{bmatrix} \tag{4.21}$$

The shape defined by joining the boundary curves at their endpoints encloses the area swept out by the band, and every point inside this region is considered to be inside the band. As before, we use a piecewise linear representation of $\mathbf{f}$ (Fig. 4-10), described by the $n \times 3$ matrix of control points $\mathrm{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n)^{\top}$, where each control point $\mathbf{f}_i$ is defined as $\mathbf{f}_i = (f_{xi}, f_{yi}, f_{hi})^{\top}$.

4.5 Band distributions

We now define a probability distribution over bands using the previously introduced concept of basis curves. We continue to use three parameters: a basis curve $\mathbf{b}$, a mean $\boldsymbol{\mu}$, and a covariance Σ. We note that many bands of interest share a common structure, and that the centerline of similar bands can be described as variations on

[3]Here, we arrange that the unit normal vector points "left" of the curve.

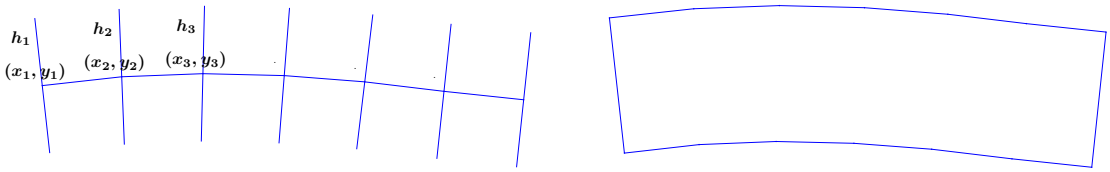


Figure 4-10: Left: We represent a band as a piecewise linear centerline curve and associate a scalar width with each centerline control point. Right: The left and right boundaries of the band can be generated from the centerline and widths.

a basis curve. The primary difference from earlier sections is the addition of a new variable to account for curve width.

More formally, we represent the basis curve $\mathbf{b}$ by its $n \times 2$ matrix of control points:

$$\mathrm{B} = \begin{bmatrix} \mathbf{b}_1^\top \\ \mathbf{b}_2^\top \\ \vdots \\ \mathbf{b}_n^\top \end{bmatrix} = \begin{bmatrix} b_{x1} & b_{y1} \\ b_{x2} & b_{y2} \\ \vdots & \vdots \\ b_{xn} & b_{yn} \end{bmatrix} \tag{4.22}$$

The mean vector $\boldsymbol{\mu}$ is a $2n$-dimensional vector of the form:

$$\boldsymbol{\mu} = (\mu_{l1}, \mu_{h1}, \mu_{l2}, \mu_{h2}, \ldots, \mu_{ln}, \mu_{hn})^\top \tag{4.23}$$

where each μ_{li} describes the mean offset of a centerline control point, and each μ_{hi} describes the mean half-width of the band at the control point. The covariance Σ is a $2n \times 2n$ matrix of the form:

$$\Sigma = \begin{bmatrix} \sigma_{l_1}^2 & \sigma_{l_1 h_1} & \sigma_{l_1 l_2} & \sigma_{l_1 h_2} & \cdots \\ \sigma_{l_1 h_1} & \sigma_{h_1}^2 & \sigma_{l_2 h_1} & \sigma_{h_2 h_1} & \cdots \\ \sigma_{l_1 l_2} & \sigma_{l_2 h_1} & \sigma_{l_2}^2 & \sigma_{l_2 h_2} & \cdots \\ \sigma_{l_1 h_2} & \sigma_{h_2 h_1} & \sigma_{l_2 h_2} & \sigma_{h_2}^2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{12}^\top & \Sigma_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \\ \Sigma_{1n}^\top & \cdots & & \Sigma_{nn} \end{bmatrix} \tag{4.24}$$

As before, we use $\bar{\mathbf{b}}_i = (\bar{b}_{xi}, \bar{b}_{yi})^\top$ to denote the unit normal vector to $\mathbf{b}$ at control point i. We refer to a distribution over bands parameterized in this form as a *basis band normal distribution*, and represent it as $\tilde{B}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$. A random band $\mathbf{g}$ drawn from the distribution can be represented by a matrix of control points:

$$\mathrm{G} = \begin{bmatrix} g_{x1} & g_{y1} & g_{h1} \\ g_{x2} & g_{y2} & g_{h2} \\ \vdots & \vdots & \vdots \\ g_{xn} & g_{yn} & g_{hn} \end{bmatrix} \tag{4.25}$$

where each $\mathbf{g}_i = (g_{xi}, g_{yi}, g_{hi})^\top$ can be expressed as:

$$\mathbf{g_i} = \begin{bmatrix} g_{xi} \\ g_{yi} \\ g_{hi} \end{bmatrix} = \begin{bmatrix} b_{xi} + \bar{b}_{xi} g_{li}^{\mathrm{B}} \\ b_{yi} + \bar{b}_{yi} g_{li}^{\mathrm{B}} \\ g_{hi} \end{bmatrix} \tag{4.26}$$

and we use the vector $\mathbf{g}^{\mathrm{B}} = (g_{l1}^{\mathrm{B}}, g_{h1}, g_{l2}^{\mathrm{B}}, g_{h2}, \ldots, g_{ln}^{\mathrm{B}}, g_{hn})^\top$ to denote the projection of $\mathbf{g}$ onto $\mathbf{b}$. Practically speaking, this is simply the projection of the centerline of $\mathbf{g}$ onto $\mathbf{b}$, augmented by the width of $\mathbf{g}$ at the polyline control points. The probability density of $\mathbf{g}$ is then defined as:

$$P_{\mathbf{g}}(\mathbf{g}; \mathbf{b}, \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{g}^{\mathrm{B}} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{g}^{\mathrm{B}} - \boldsymbol{\mu})\right) \tag{4.27}$$

Note that Eq. (4.27) is identical in form to Eq. (4.5); the difference is in the structure of the mean and covariances. As with curve distributions, it is sometimes useful to change the basis curve upon which a band distribution has been defined, while incurring minimal changes the actual distribution. Choosing a new mean and covariance in the cases of re-sampled and offset basis curves follows the same procedure as in Sec. 4.2.2, with minor modifications. When the new basis curve is a variation of the original basis curve, the width components of the mean band do not change. When the new basis curve is a re-sampling of the original basis curve, the re-sampling matrix H must account for re-sampling the width components in addition to the centerline offset values.

4.6 Band estimation

We now describe an algorithm for estimating a band $\mathbf{f}$ given a partial observation. In particular, we are interested in observations of the band boundary curves, which the algorithm uses to update its belief about the width and centerline geometry of $\mathbf{f}$.

We represent the system's belief about the band with the distribution $\tilde{B}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$. The projection of $\mathbf{f}$ onto $\mathbf{b}$ is given by the vector $\mathbf{f}^{\mathrm{B}} = (f_{l1}^{\mathrm{B}}, f_{h1}, f_{l2}^{\mathrm{B}}, f_{h2}, \ldots, f_{ln}^{\mathrm{B}}, f_{hn})^\top$, $\boldsymbol{\mu}$ can be considered to be an estimate of $\mathbf{f}^{\mathrm{B}}$, and:

$$\hat{\mathrm{F}} = \mathrm{B} + \mathrm{diag}(\boldsymbol{\mu})\bar{\mathrm{B}} \tag{4.28}$$

is the piecewise linear representation of the estimate $\hat{\mathbf{f}}$ of $\mathbf{f}$.

Note that curve estimation can be treated as a special case of band estimation, where the estimated half-width values and their associated covariance entries are zero, indicating absolute certainty of zero width.

4.6.1 Boundary observations and data association

We define a noisy observation $\mathbf{z}(u) = (z_x(u), z_y(u))^\top$ of the left boundary of $\mathbf{f}$ as:

$$\mathbf{z}(u) = \begin{bmatrix} z_x(u) \\ z_y(u) \end{bmatrix} = \begin{bmatrix} f_x(u) + (f_h(u) + v(u))\bar{f}_x(u) \\ f_y(u) + (f_h(u) + v(u))\bar{f}_y(u) \end{bmatrix}, \quad u \in [u_{z0}, u_{z1}] \tag{4.29}$$

where $v(u)$ is a scalar noise term and $\bar{\mathbf{f}}(u) = (f_x(u), f_y(u))^\top$ is the unit normal vector. Thus, $\mathbf{z}$ is a random curve whose probability distribution is determined by $\mathbf{f}$ and $v(u)$. It is typically the case that $\mathbf{z}$ is only a partial observation of $\mathbf{f}$, such that $s_1 \leq u_{z0} < u_{z1} \leq s_n$. An observation of the right boundary can similarly be described as:

$$\mathbf{z}(u) = \begin{bmatrix} z_x(u) \\ z_y(u) \end{bmatrix} = \begin{bmatrix} f_x(u) + (-f_h(u) + v(u))\bar{f}_x(u) \\ f_y(u) + (-f_h(u) + v(u))\bar{f}_y(u) \end{bmatrix}, \quad u \in [u_{z0}, u_{z1}] \tag{4.30}$$

and we describe an observation of either boundary as:

$$\mathbf{z}(u) = \begin{bmatrix} z_x(u) \\ z_y(u) \end{bmatrix} = \begin{bmatrix} f_x(u) + (af_h(u) + v(u))\bar{f}_x(u) \\ f_y(u) + (af_h(u) + v(u))\bar{f}_y(u) \end{bmatrix}, \quad u \in [u_{z0}, u_{z1}] \tag{4.31}$$

where a has value 1 or -1 if the observation is of the left or right boundary, respectively.

As before, the true band $\mathbf{f}$ is not known or directly observable so we approximate the observation distribution using the current state estimates. Since $\mathbf{z}$ is an observation of a boundary, it is drawn from a distribution over bands, and we can parameterize it in the same way as before, using the basis curve $\mathbf{b}$, offset vector $\mathbf{z}^{\mathrm{B}} = (z_1^{\mathrm{B}}, z_2^{\mathrm{B}}, \ldots, z_n^{\mathrm{B}})^\top$, and covariance matrix R. Each control point $\mathbf{z}_i$ can be written:

$$\mathbf{z}_i = \mathbf{b}_i + z_i^{\mathrm{B}}\bar{\mathbf{b}}_i \tag{4.32}$$

The offset vector $\mathbf{z}^{\mathrm{B}}$ also represents the projection of $\mathbf{z}$ onto B, and each z_i^{B} can be expressed as:

$$z_i^{\mathrm{B}} = f_{li}^{\mathrm{B}} + af_{hi} + v_i \tag{4.33}$$

so that $\mathbf{z}^{\mathrm{B}}$ itself can be written as:

$$\begin{aligned} \mathbf{z}^{\mathrm{B}} &= \begin{bmatrix} 1 & a & 0 & 0 & 0 & \ldots & & 0 \\ 0 & 0 & 1 & a & 0 & \ldots & & \vdots \\ \vdots & \vdots & & & \ddots & \ddots & & \\ 0 & \ldots & & & & & 1 & a \end{bmatrix} \mathbf{f}^{\mathrm{B}} + \mathbf{v} \\ &= \mathrm{A}\mathbf{f}^{\mathrm{B}} + \mathbf{v} \end{aligned} \tag{4.34}$$

where $\mathbf{v}$ is a zero-mean noise term distributed according to $\mathbf{v} \sim N(\mathbf{0}, \mathrm{R})$. Thus, when the true band and the observation are projected onto the basis curve $\mathbf{b}$, the observation can be related to the true band by a linear transformation A. If $\mathbf{z}$ is a partial observation of the boundary, such that it only projects onto m control points

of B, then A is a $2m \times 2n$ matrix instead of $2n \times 2n$ matrix, as described earlier for zero-width curves.

Data Association

Data association for bands can be approached in the same way as for curves. Given a belief over $\mathbf{f}$ described by $\tilde{B}(\mathbf{b}, \boldsymbol{\mu}, \Sigma)$, and an observation $\mathbf{z}$ described by $\tilde{B}(\mathbf{b}, \mathbf{z}^{\mathrm{B}}, \mathrm{R})$, we can apply a χ^2 test on the Mahalanobis distance to determine if $\mathbf{z}$ is an observation of $\mathbf{f}$.

4.6.2 Update

The update step for estimating bands is identical to the update step for curve. Figure 4-11 shows a full update cycle, where an observation of a band boundary is used to both update and extend the band.

The updated mean and covariance, which we denote as $\tilde{\boldsymbol{\mu}}$ and $\tilde{\Sigma}$, are given by:

$$\begin{aligned} \mathrm{K} &= \Sigma \mathrm{A}^\top (\mathrm{A}\Sigma\mathrm{A}^\top + \mathrm{R})^{-1} \\ \tilde{\boldsymbol{\mu}} &= \boldsymbol{\mu} + \mathrm{K}(\mathbf{z}^{\mathrm{B}} - \mathrm{A}\boldsymbol{\mu}) \\ \tilde{\Sigma} &= (\mathrm{I} - \mathrm{K}\mathrm{A})\Sigma \end{aligned} \tag{4.35}$$

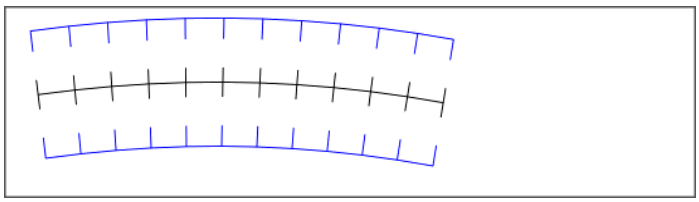
After the updated estimates have been computed, we once again reparameterize the distribution such that the basis curve coincides with the updated maximum likelihood estimate, to minimize approximation error in future update steps.

4.7 Summary

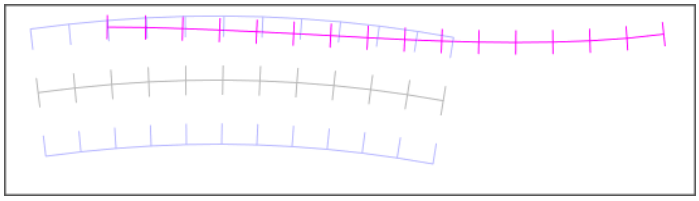
This chapter has described two algorithms for the estimation of spatial curves from partial observations. The first algorithm provides a way to estimate curves, and is shown to be a special case of the second algorithm, which estimates bands from partial observations of the band's boundaries. Cure estimation using basis curves is closely related to the "lateral uncertainty" method we described in an earlier publication [30].

The key innovation behind these algorithms is to express curves as lateral variations of a basis curve. By projecting arbitrary curves into a space defined by the basis curve, observations become linear transformations of the true curves, and Gaussian distributions can be used to model uncertainty. These two properties then allow us to apply Kalman filtering techniques to robustly estimate and track curve and bands.

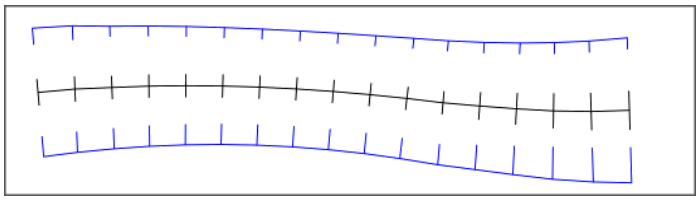
Extending these estimation algorithms to the case of many bands is simple if all bands are considered independently of each other. Many separate filters can be maintained, one for each band, and observations can be assigned to bands as they are generated. This is the technique we use for lane estimation, and is described in the following chapter. In practice, it is often the case that some or all of the curves are spatially correlated. This raises interesting challenges in terms of how to model the correlation.



(a) A band estimate



(b) An observation of the band boundary



(c) The updated estimate

Figure 4-11: An observation is made of a band boundary, and used to update the estimate. The black curve marks the band centerline, and the blue curves mark the left and right boundary marginal distributions. Short line segments along the curves mark control points, and the length of these segments indicate 1-σ uncertainty.

The band estimation algorithm can be viewed as a method for jointly estimating two curves (the left and right boundary curves), where two curves share a basis curve. It is easy to extend this algorithm to jointly estimate three or more curves, as long as each of the curves can be accurately modeled as a variation of an underlying basis curve. Joint estimation of multiple curves with separate basis curves remains to be studied.

Chapter 5

Lane Estimation with Basis Curves

This chapter describes how we apply the curve estimation algorithms from Chapter 4 to the lane estimation problem. The application is generally straightforward, although some additional assumptions are made to simplify the implementation. At the end of the chapter, we provide a quantitative evaluation of this approach.

First, we provide an intuitive description. A lane can be modeled as a single band, where lane boundaries are typically marked by painted lines or physical curbs. In some situations, one or both boundaries may be unobserved due to occlusions or poorly marked roads. The road paint and curb features detected by the algorithms from Chapter 2 serve as observations of the lanes, and are used to update the estimates accordingly.

This raises the question of how the initial lane estimates are obtained. Given two curve estimates that plausibly correspond to the opposite boundaries of a single lane (e.g., they are long, roughly parallel locally, and separated by an appropriate distance), the curves can be paired an "upgraded" into a lane estimate. Thus, curves are first estimated and tracked independently of lanes, and subsequently joined together to form lane estimates.

In turn, we must also initialize the lane boundary estimates. This is simpler, as a single road paint or curb feature can serve as the initial estimate for a lane boundary. If a feature is not matched to either a lane or a boundary estimate, then a new boundary estimate is created and tracked. We also propose a strategy for discarding and trimming estimates to maintain computational efficiency.

This chapter is structured as follows. Section 5.1 introduces the concept of curve orientation, which will be useful for data association. Section 5.2 describes how boundary curves are estimated. Section 5.3 describes how lanes are initialized from boundary curves and subsequently tracked. Section 5.4 provides a quantitative evaluation of the overall method.

5.1 Curve Orientation

While a painted road line may in some sense be interpreted as a single curve, it often serves as the boundary for two separate lanes. Additionally, a LIDAR-detected

physical curb may serve as the boundary for a single lane, but typically on only one side of the curve – the physically lower side. To express these constraints, we augment our definition of a curve to include orientation. Specifically, we arrange the parameterizations of all potentially boundary curves such that the left side of a boundary curve is possibly a lane. Formally, if we represent a boundary curve $\mathbf{f}(s)$ as:

$$\mathbf{f}(s) = \begin{bmatrix} f_x(s) \\ f_y(s) \end{bmatrix} \tag{5.1}$$

and its tangent as:

$$\mathbf{f'}(s) = \begin{bmatrix} f'_x(s) \\ f'_y(s) \end{bmatrix} = \begin{bmatrix} \frac{df_x}{ds}(s) \\ \frac{df_y}{ds}(s) \end{bmatrix} \tag{5.2}$$

then for any s, the ray that emanates from $\mathbf{f}(s)$ and has direction:

$$\mathbf{f}^+(s) = \begin{bmatrix} -f'_y(s) \\ f'_x(s) \end{bmatrix} \tag{5.3}$$

points toward the lane centerline of a boundary curve. We arrange that all boundary curves considered are oriented in this fashion by reversing the order of a curve control points when necessary.

5.1.1 Relative Orientation

Given two oriented curves $\mathbf{f}$ and $\mathbf{g}$, we can describe their relative orientation. Define the function $c^{\mathbf{g}}_{\mathbf{f}}(s)$ to be the scalar value yielding the closest point on $\mathbf{g}$ to $\mathbf{f}(s)$:

$$c^{\mathbf{g}}_{\mathbf{f}}(s) = \arg\min_t ||\mathbf{g}(t) - \mathbf{f}(s)|| \tag{5.4}$$

We say that $\mathbf{f}$ and $\mathbf{g}$ face the *same direction* if, for every point $\mathbf{f}(s)$ on $\mathbf{f}$ and every point $\mathbf{g}(t)$ on $\mathbf{g}$, $\mathbf{f}^+(s)^\top \mathbf{g}^+(c^{\mathbf{g}}_{\mathbf{f}}(s)) > 0$ and $\mathbf{g}^+(t)^\top \mathbf{f}^+(c^{\mathbf{f}}_{\mathbf{g}}(t)) > 0$. We also say that $\mathbf{f}$ and $\mathbf{g}$ face *opposite directions* if, for every point $\mathbf{f}(s)$ on $\mathbf{f}$ and every point $\mathbf{g}(t)$ on $\mathbf{g}$, $\mathbf{f}^+(s)^\top \mathbf{g}^+(c^{\mathbf{g}}_{\mathbf{f}}(s)) < 0$ and $\mathbf{g}^+(t)^\top \mathbf{f}^+(c^{\mathbf{f}}_{\mathbf{g}}(t)) < 0$. If neither is true, then the relative orientation of $\mathbf{f}$ and $\mathbf{g}$ is undefined.

Intuitively, relative orientation simply encodes whether or not the normal vectors of $\mathbf{f}$ and $\mathbf{g}$ point in the same direction at the closest points on the two curves. If this is the case, then the two curves face the same direction. If the normal vectors always point in opposite directions, then the two curves face opposite directions. It is possible for neither to be true, in which case the relative orientation is undefined. Note that relative orientation is a symmetric relationship.

The relative orientation of two curves will be useful during the data association steps for lane estimation. For example, we can enforce the requirement that an observation of a lane boundary and the boundary curve itself always always face the same direction. If we then arrange for a single strip of road paint to be modeled as two curves with identical geometry and opposite directions, then the road paint can serve as an observation of two adjacent lanes.

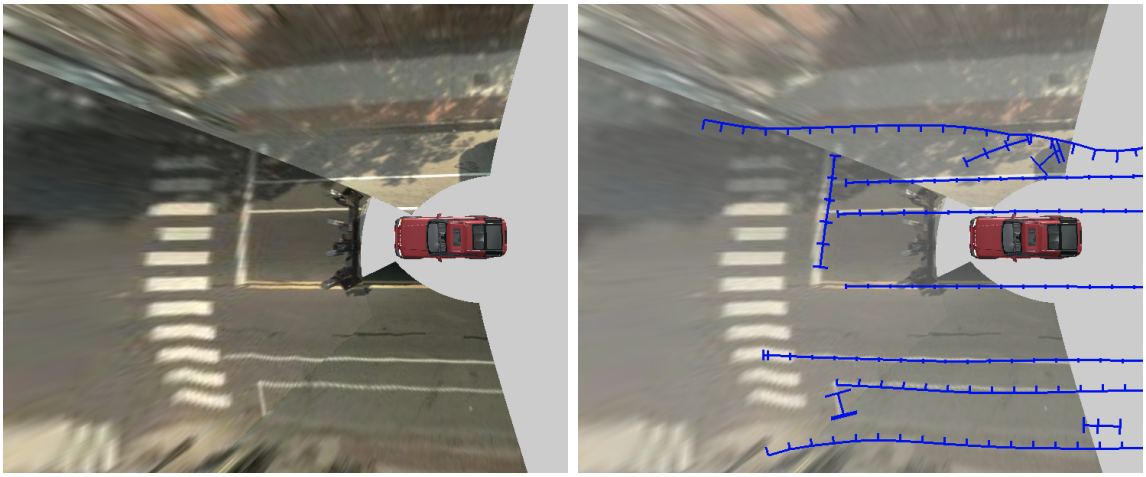


Figure 5-1: Our system attempts to estimate and track all spatial curves on the ground, many of which may not correspond to actual lane boundaries. (left) A synthesized overhead view of the vehicle and three camera images projected onto an assumed ground plane. (right) Spatial curves estimated from camera and LIDAR data. These curves are later used to infer the presence of lanes and estimate their geometry.

5.2 Boundary Estimation

Our algorithm initializes lane estimates from pairs of existing boundary estimates. Therefore, it is necessary to obtain boundary estimates before a lane can be initialized. We model painted lines and physical road edges as curves, and represent each with a basis curve normal distribution. Each estimated curve may or may not correspond to a lane boundary, but our algorithm attempts to estimate and track each nearby spatial curve that has the potential to be a lane boundary (Figure 5-1).

Throughout the boundary estimation process, we reason about each control point of a boundary independently from the other control points of the boundary. In other words, we enforce diagonal covariance matrices and ignore the off-diagonal terms that may result from basis curve re-sampling and incorporation of observations. This introduces significant computational benefits, as the data association and update steps of the Kalman filter can be accomplished in time $O(n)$ rather than $O(n^{2.4})$ [69]. However, it also introduces approximation errors and discards potentially useful information.

Allowing the covariance matrix to be fully dense can be expected to improve the results, as it could allow for tighter thresholds during the data association step, and it would also have a smoothing effect on the resulting curve estimates. Doing so would incur the additional computational burden mentioned above, and rank-deficient (i.e., not positive-definite) covariance matrices would need to be carefully handled. In practice, dropping the off-diagonal terms has yielded reasonable results in our experiments.

5.2.1 Observation Model

We now describe how features detected by the algorithms described in Chapter 2 are mapped to observations. Vision and LIDAR algorithms have different error characteristics, and the primary concern in mapping features to observations is correctly modeling the error.

Gaussian Error

The road paint and curb detectors described in Chapter 2 are noisy detectors in the sense that the estimates they produce typically correspond to the true features corrupted by various sources of error. The errors are not truly Gaussian, and observation errors are not truly independent, but we model them as such.

In the case of the road paint detectors, error sources include camera mis-calibrations, imperfect time synchronization, terrain modeling errors, and approximations made by the feature detection algorithms. Additionally, some of these errors may persist over time and across observations, resulting in strong correlations. For example, an error in the terrain model will introduce a projection error that affects any detections made on the affected area. Camera calibration and time synchronization errors also introduce persistent error correlations.

LIDAR-detected features are less susceptible to projection errors, due to providing more precise range measurements, but neither are the errors in these features truly Gaussian or independent. Calibration and time synchronization errors still introduce correlations, and road surfaces with different reflectance and absorption characteristics may introduce range measurement bias.

The result in both modalities is an error that is not truly Gaussian, and not truly independent across observations. This is a difficulty encountered in virtually all practical Kalman filtering applications. However, modeling the error as Gaussian and independent across observations yields reasonable results in our experiments. One practical impact of this approximation is overconfidence in the resulting estimates, and we apply a standard technique of routinely scaling the covariance matrix such that the diagonal terms have a minimum bound (0.1) to compensate for these inaccuracies [5].

Mapping Road Paint Features to Observations

Since a single strip of road paint can serve as an observation of two adjacent lanes, we map each single road paint feature to two curve distributions. The two distributions are identical except for basis curve orientation. While this may at first seem computationally wasteful, this duplication simplifies the process of tracking multiple lanes. Additionally, our implementation marks pairs of observations generated in this way and is able to reason about them jointly in a number of cases, thus saving some computational effort.

We begin by defining a basis curve normal distribution using the feature curve itself as a basis curve, and mean vector of zero. How to choose the covariance matrix

is less obvious. Currently, we use a simple model of uncertainty in image space, which is then projected into a world frame.

To determine the uncertainty for a control point $\mathbf{z}_i$ of a vision-detected road paint feature, we define two parameters ρ, and k. The parameter ρ models uncertainty of the control point in image space, and k represents the remaining unmodeled uncertainty terms. The variance of i is then described as:

$$\sigma_i^2 = \rho \frac{d_i^2}{f^2} + k; \tag{5.5}$$

where f is the focal length of the camera used to detect the feature, and d_i is the Euclidean distance from the camera to the local-frame projection of the feature control point. The intuition behind this formulation is that the uncertainty of a control point increases quadratically with its distance from the vehicle, reflecting the uncertainty in perspective projection. We currently ignore other sources of uncertainty mentioned earlier such as from terrain modeling and camera calibration. In our experiments, $k = 0.3$ and $\rho = 3.0$.

Mapping Curb Features to Observations

Unlike a strip of road paint, a curb feature can serve as the boundary only for a single lane. Thus, a curb feature is used to initialize a single curve distribution, oriented according to the curb orientation.

Defining an observation from a LIDAR-detected curb is slightly simpler than for visually detected road paint, as there is no perspective projection to consider. Once again, we define a basis curve identical to the feature curve and set the mean to zero. Each control point $\mathbf{z}_i$ of a new observation is assigned the same variance of $\sigma_i^2 = \sigma_c^2$, where $\sigma_c^2 = 0.25$ for our experiments.

The uncertainty parameters for the road paint and curb features were manually selected, based on a qualitative analysis of previously recorded data sets. It should be possible to improve performance by statistical regression on the parameters, but this also runs the risk of overfitting to the training data set. Thus, we currently view the choice of covariance terms as a tunable parameter.

5.2.2 Data Association and Update

Once a feature has been mapped to a distribution over curves, the next step is to assess whether or not the feature is an observation of an existing curve estimate. We use a gated nearest-neighbor greedy assignment procedure, based on the Mahalanobis distance in Section 4.3.3, and match features independently from each other. Each observation is assigned to the curve estimate curve with the smallest Mahalanobis distance, subject to a χ^2 threshold. We use a p-value of 0.06.

In addition to the Mahalanobis distance, we also apply two separate criteria. First, the feature curve must face the same direction as the tracked curve. Second, the projection of the feature curve onto the tracked basis curve must have a minimum length of 6.0 m for road paint features, and 2.0 m for LIDAR-detected curbs.

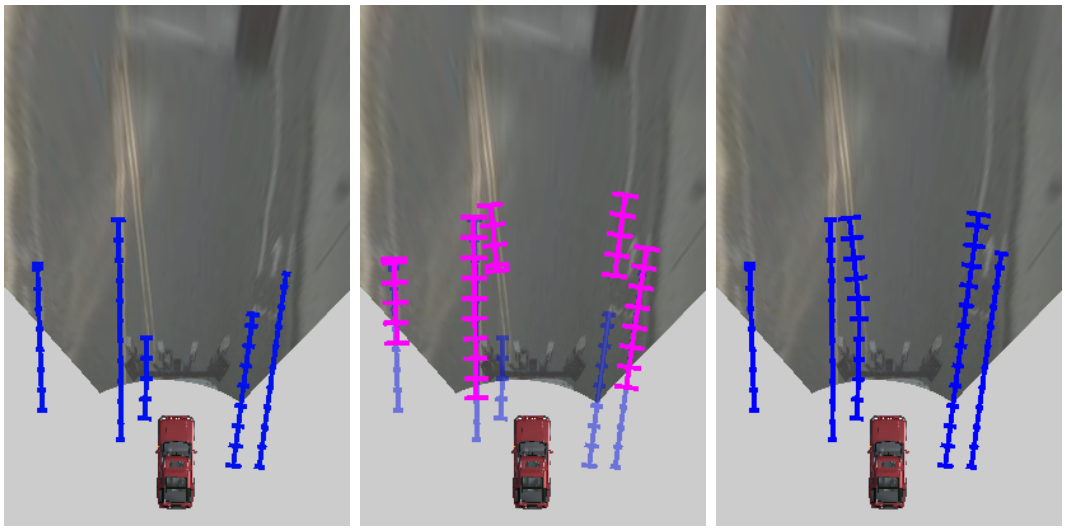


Figure 5-2: (left) Boundary curve estimates are maintained near the vehicle. (middle) Observations are made of each boundary curve. (right) Updated boundary curve estimates after incorporating observation data. A camera image is projected onto the ground plane for reference. Short horizontal line segments denote the 1-σ uncertainty bounds for each control point.

If the feature curve is successfully associated with a tracked curve, then the algorithm updates the tracked curve according to Section 4.3.5. If none of the existing tracked curves is likely to have generated the feature curve as an observation, then the feature curve itself is used to initialize a new tracked curve. In this manner, the system attempts to estimate and track *all* spatial curves that could possibly correspond to lane boundaries. Figure 5-2 shows a full update step, where several boundary curves are observed and updated.

5.2.3 Merging Curves

It is possible for the system to maintain two or more curve estimates that actually correspond to different parts of the same underlying curve. For example, if an object blocks the sensor's view of some portion of a boundary curve, the feature detectors may produce two disjoint feature curves. These feature curves may then be incorporated into separate curve filters. As the vehicle moves, its field of view may change so that it observes the previously unobserved portion of the curve. In this situation, it would be useful to merge the two separate curve estimates into a single curve estimate encompassing both curve fragments.

Since each individual curve estimate is simply a basis curve normal distribution, merging two curves simply consists of considering one curve to be an observation of the other. If the "observation" curve passes the data association test of Section 4.3.3, then the other curve is updated accordingly, and the observation curve is discarded.

5.3 Lane Estimation

As is the case for boundary curves, when reasoning about lanes we make assumptions that allow for simpler formulations and implementations, at the expense of model accuracy.

Each lane is represented as a distribution over bands. Our algorithm reasons about each lane independently of other lanes, and conceptually maintains N_l separate filters, where N_l is the number of lanes being tracked. In reality, spatially proximate lanes are not independent, especially adjacent lanes on the same roadway. Estimating and maintaining these correlations would require additional state and introduce complexity. Therefore, we model lanes independently and propose some simple heuristics to prevent some of the most common errors that would otherwise result from the independence assumption.

In addition to treating lanes independently from each other, we also reason about the control points of a lane as mutually independent. However, each control point has two degrees of freedom (one for the offset, one for the half-width) and it is important that the structure for each control point is jointly estimated. The covariance matrix Σ for a lane estimate therefore has a block-diagonal structure, and can therefore be represented as the sparse $2n \times 2n$ matrix:

$$\Sigma = \begin{bmatrix} \sigma_{l1}^2 & \sigma_{l1h1} & 0 & \dots & & & & \\ \sigma_{l1h1} & \sigma_{h1}^2 & 0 & \dots & & & & \\ 0 & 0 & \sigma_{l2}^2 & \sigma_{l2h2} & \dots & & & \\ \vdots & \vdots & \sigma_{l2h2} & \sigma_{h2}^2 & \ddots & & & \\ & & \vdots & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & 0 & 0 \\ & & & & & 0 & \sigma_{ln}^2 & \sigma_{lnhn} \\ & & & & & 0 & \sigma_{lnhn} & \sigma_{hn}^2 \end{bmatrix} \tag{5.6}$$

Jointly estimating the width and offset at each control point allows the system to maintain a lane estimate when a single boundary is observed. Since a boundary observation is fully correlated with the width and offset, information about one boundary provides valuable information about the geometry of the opposite boundary. This is reflected in the correlation between lane width and offset (Figure 5-3).

5.3.1 Lane Detection

Our feature detectors do not produce features that can be directly reasoned about as lanes, so a single feature curve cannot be used to initialize a lane estimate. Instead, each lane estimate is initialized by "detecting" lanes from pairs of boundary curves that are sufficiently lane-like.

Intuitively, if two curves are observed and tracked that appear to be reasonably long, locally parallel, and approximately 3 m apart, then they are likely to correspond to the two boundaries of a lane. Our algorithm considers each pair of boundary curves

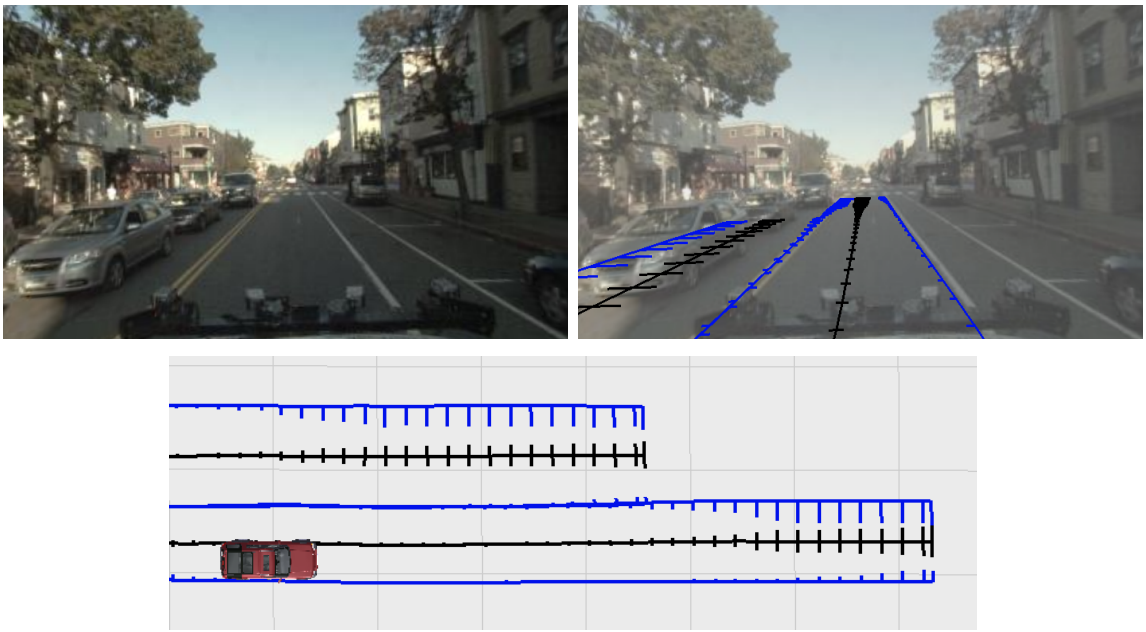


Figure 5-3: (top left) The farthest boundary of the opposite lane is occluded by other vehicles. However, by maintaining a joint distribution over centerline geometry and lane width, (top right) the system can still maintain an estimate of the opposite lane, using observations of the closer boundary. (bottom) A synthesized overhead view of the vehicle and its lane estimates. Black curves correspond to the estimated lane centerlines, and blue curves denote the marginal distributions of the lane boundaries. Short line segments extending laterally from the curves indicate control point uncertainty.

in this way, and greedily initializes a lane estimate whenever a pair of curves satisfy these criteria.

If two subsections of a pair of boundary curves are both at least 15 m long, are never less than 3.0 m or more than 7.0 m apart, and the curves face opposite directions, then they are used to initialize a new lane estimate. Boundary curves can be derived from either LIDAR or vision data. In some cases, a lane estimate is initialized entirely from vision-detected road paint or from LIDAR detections alone. In others, curves detected with both modalities are used for lane detection (Fig. 5-4). Detection is accomplished in two steps. First, we define an initial basis curve and project the two boundaries onto this initial basis curve. We can then estimate a mean vector and covariance, and finally reparameterize the resulting distribution to set the mean vector to zero, similar to the end of an update step.

For convenience, call one boundary curve the left curve, and the other the right curve. The initial basis curve $\mathbf{b}$ is chosen as the medial axis of the left and right curves, such that every control point of the initial basis curve is equidistant from the two boundaries. Projecting one curve onto the other and scaling the offset vector by 0.5 provides a sufficiently close approximation to the medial axis. We then seek to estimate the initial mean vector and covariance $\boldsymbol{\mu}$ and Σ.

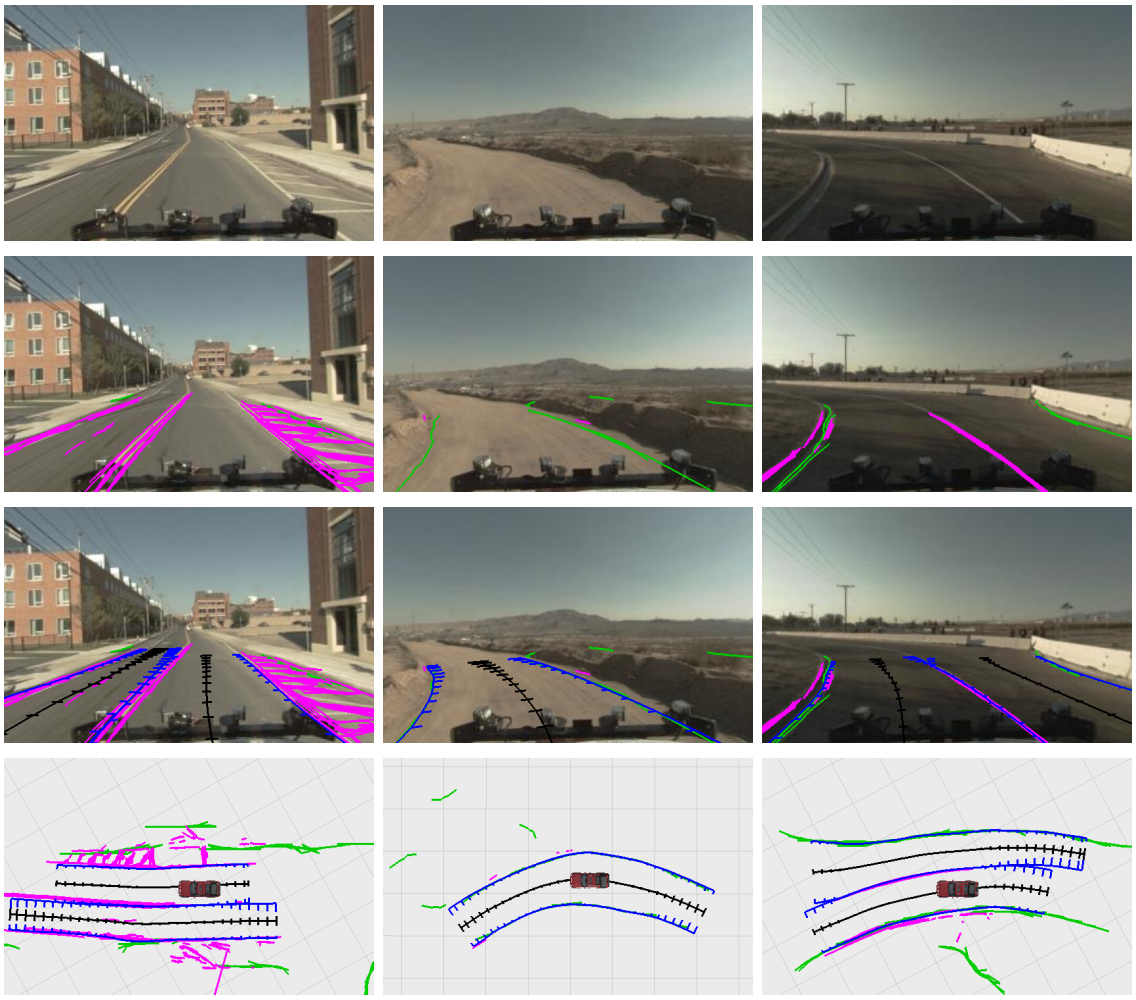


Figure 5-4: Lane initialization from individual curve estimates in three settings. (top row) Camera images. (second row) Road paint (magenta) and curb detections (green). (third row) Lane estimates projected into the image. Black line denotes the centerline estimate, and blue lines denote boundary curves. (bottom row) Synthesized overhead view of lane estimates.

To do this, the left and right boundary distributions are reparameterized such that they are defined on the basis curve $\mathbf{b}$, with the left and right mean vectors described as $\boldsymbol{\mu}_l$ and $\boldsymbol{\mu}_r$, and the left and right covariances given by Σ_l and Σ_r, respectively. For the purpose of initializing the lane estimate, we treat the left and right boundaries as independent observations of the lane, and express them jointly as:

$$\begin{aligned} \mathbf{z} = \begin{bmatrix} \boldsymbol{\mu}_l \\ \boldsymbol{\mu}_r \end{bmatrix} &= \begin{bmatrix} \mathrm{A}_l \\ \mathrm{A}_r \end{bmatrix} \mathbf{f}^{\mathrm{B}} + \mathbf{v} \\ &= \mathrm{A}_z \mathbf{f}^{\mathrm{B}} + \mathbf{v} \end{aligned} \tag{5.7}$$

where A_l and A_r are the transformation matrices relating a wide curve to its left and right boundary observations (Section 4.6.1), $\mathbf{f}^{\mathrm{B}}$ is the projection of the unobserved

true lane onto $\mathbf{b}$, and $\mathbf{v} \sim N(\mathbf{0}, \Sigma_z)$ is a noise term described by the covariance matrix Σ_z:

$$\Sigma_z = \begin{bmatrix} \Sigma_l & 0 \\ 0 & \Sigma_r \end{bmatrix} \tag{5.8}$$

Note that in our implementation, Σ_l and Σ_r, and Σ_z are all diagonal matrices, a result of the independence assumptions made earlier. The initial distribution parameters that best represent the information provided by the boundary curves can then be defined as:

$$\Sigma = (\mathrm{A}_z^\top \Sigma_z^{-1} \mathrm{A}_z)^{-1} \tag{5.9}$$

and:

$$\boldsymbol{\mu} = \Sigma \mathrm{A}_z^\top \Sigma_z^{-1} \mathbf{z} \tag{5.10}$$

This can be shown using the information filter, the dual of the Kalman filter [69]. Choosing initial distribution parameters is equivalent to starting with an initial distribution with infinite covariance, and then incorporating the observation $\mathbf{z}$. This can be easily represented in information form using a zero information matrix, and leads to Equations (5.9) and (5.10). Finally, the initial distribution is reparameterized such that the basis curve matches the initial mean vector, as described in Section 4.6.2.

5.3.2 Observation Model

The lane observation model incorporates observations of either the left or right boundary into a lane estimate. Observations include the feature curves produced by the road paint and curb detectors, and the boundary curves tracked in Section 5.2.

To see why the tracked boundary curves are useful as observations, consider the situation where the feature curve corresponds to the boundary of a lane being tracked, but the curve is far from the current lane estimates. This commonly occurs when the sensor's view is occluded by an obstacle such as another vehicle in the travel lane. In this case, the feature cannot be directly incorporated into the lane estimate. However, if it is tracked as an individual curve, then as the vehicle or occluding object moves, the lane estimate may be extended such that the individually tracked curve can then be merged into the lane estimate as a boundary observation.

Whenever a feature curve is produced by a feature detector, it is checked against existing lane estimates, and merged greedily if possible. If the feature is not immediately associated with a lane estimate, it is passed through the curve estimation system described earlier in the chapter. Periodically, tracked individual curves are checked against lane estimates and merged into a lane estimate if possible. Our implementation runs this latter procedure at 10 Hz.

5.3.3 Data Association

Many features are outliers that do not signify lane boundaries. These include false road paint detections due to tree shadows, curb detections on stationary vehicles, and true detections of road paint that simply does not mark a lane boundary (Fig. 5-5). Several criteria are used to determine if a curve is an observation of a lane boundary.

First, the observation curve is projected onto the lane basis curve. If the length of this projection is shorter than a fixed threshold, then the association is rejected. Similar to the data association process for individual curves, we enforce a minimum projection length of 6.0 m.

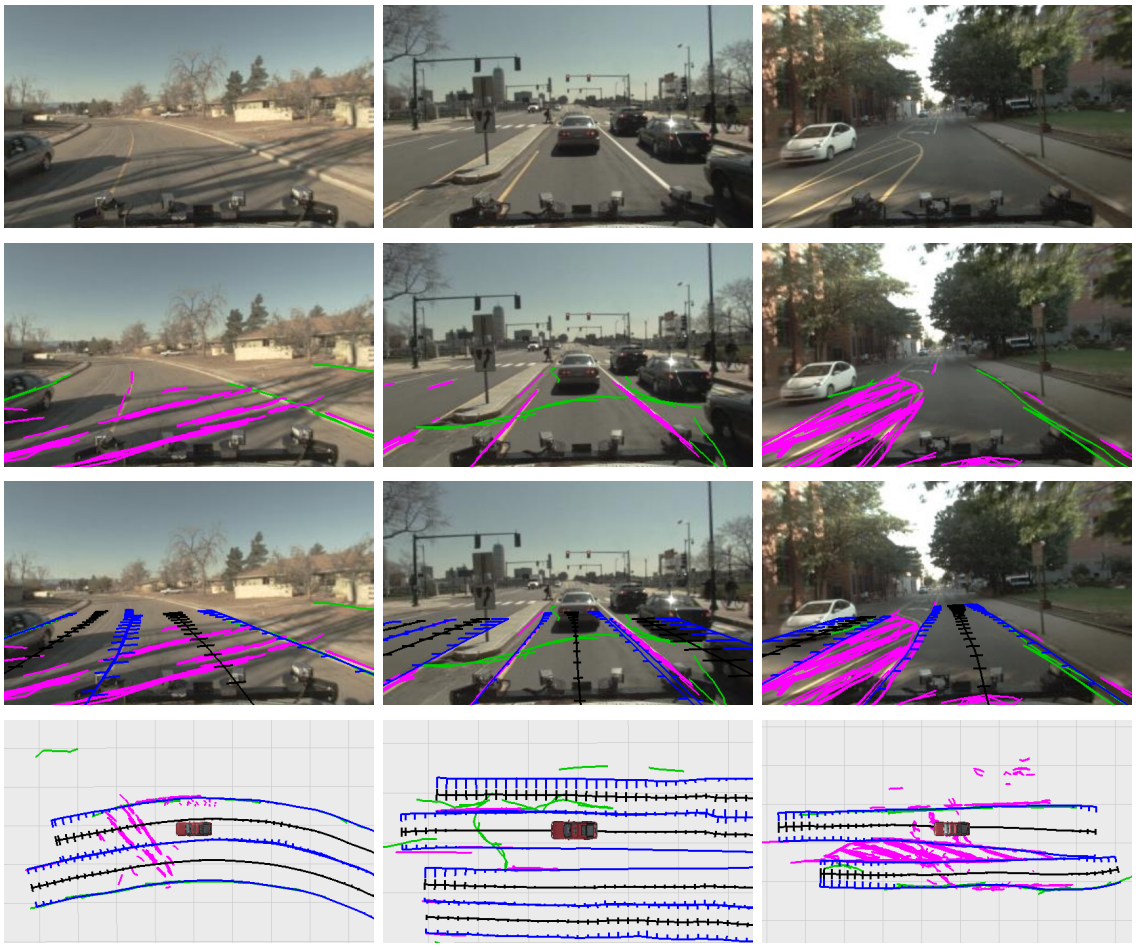


Figure 5-5: Data association steps map observed features to lane estimates, and detect features that do not mark lane boundaries such tree shadows (left), spurious LIDAR features (middle), or true road paint markings that do not mark lane boundaries (right). Images arranged as in Figure 5-4.

Next, the χ^2 test is used to check curve geometry against expected boundary geometry, given the current lane estimate. Since it is initially unknown if the curve is an observation of the left or right boundary, this test is computed against both boundaries and the best match of the two boundaries is used for further tests. Once the left or right boundary has been chosen, a p-value is computed to assess the match likelihood. If the resulting p-value is less than 0.94 (the same threshold used for estimating individual curves), the match is rejected.

Third, the observation curve must pass an orientation test. If it does not face the same direction as the boundary curve, the match is rejected. This criterion is perhaps not as intuitive as the others, and can be explained by its primary purpose,

which is to allow a single strip of road paint to serve as the boundary of two adjacent lanes. Recall that a road paint feature is used to generate two observation curves that are geometrically identical but have opposite orientations. This check enforces the restriction that each of the two observation curves can be matched to exactly one side of the road paint.

Two additional heuristics are used to account for errors that may commonly occur as a result of the Gaussian distribution assumption. Since a Gaussian distribution technically has infinite support, our model allows for the possibility, however small, of very narrow and very wide lanes. Even lanes of negative width are allowed, which is clearly impossible. As a gross correction, we introduce a minimum lane width of 2.74 m (9 ft) and a maximum lane width of 7.01 m (23 ft); data associations that would cause the width of a lane to exceed these limits are summarily rejected.

Finally, we encode the notion that a physical road boundary is not allowed to run through the middle of a lane, by forcing each observation curve to match a lane estimate when the physical road boundary is "inside" the lane boundaries. Together, we have found these criteria to provide good results when associating boundary-like curve features with lane estimates.

5.3.4 Update

Once an observation has been associated with an existing lane, the lane estimate is recomputed according to the update step described in Section 4.6.2. To compensate for the overconfidence that results from unmodeled correlations in observation error, the covariances are occasionally scaled to maintain minimum values on the diagonal. This is similar to the "fudge factor" commonly found in numerous practical Kalman filtering applications where modeling correlated error is difficult or impractical [5].

The system extends lane estimates to model newly observed sections of a lane as the vehicle travels. To prevent unbounded growth in dimensionality, lane estimates are also periodically truncated such that portions no longer needed for autonomous navigation are discarded. Sections of a lane estimate farther than 75 m behind the vehicle are trimmed by removing the relevant control points of the basis curve, and adjusting the mean and covariance matrices accordingly. Lastly, basis curves are periodically re-sampled to maintain 1 m control point spacing. Curve and band distributions are updated accordingly.

Overlapping Lanes

A direct consequence of reasoning about lanes independently is that the data association and update steps described thus far allow lanes to overlap. For example, an update may cause a boundary of one lane to intrude into the area covered by an adjacent lane. In some situations, this may actually be correct, as lanes can merge, fork and intersect, and the boundaries of adjacent lanes inside these transition regions are not as well defined. However, in other situations, faulty data association may cause two lanes to overlap when in truth they do not.

Estimating the geometry of a lane during a transition region is out of the scope of this work, so we restrict lane estimates simply by forbidding any overlap. When two lanes overlap by a significant amount after an update step, we assume that the lanes are actually adjacent in the overlapping regions. The two overlapping subsections of each lane are then treated as observations of the opposite lane boundary, and both lanes are updated with the resulting "artificial" observations.

Using this approach to enforce the non-overlapping lane constraint is a crude way to reason about lanes jointly, and touches on the fact that the geometry of nearby lanes are almost never mutually independent. A more robust approach would be to model and reason about the joint geometric distribution of multiple lanes. We leave this for future work and note that our simple heuristic has proved effective in a number of settings.

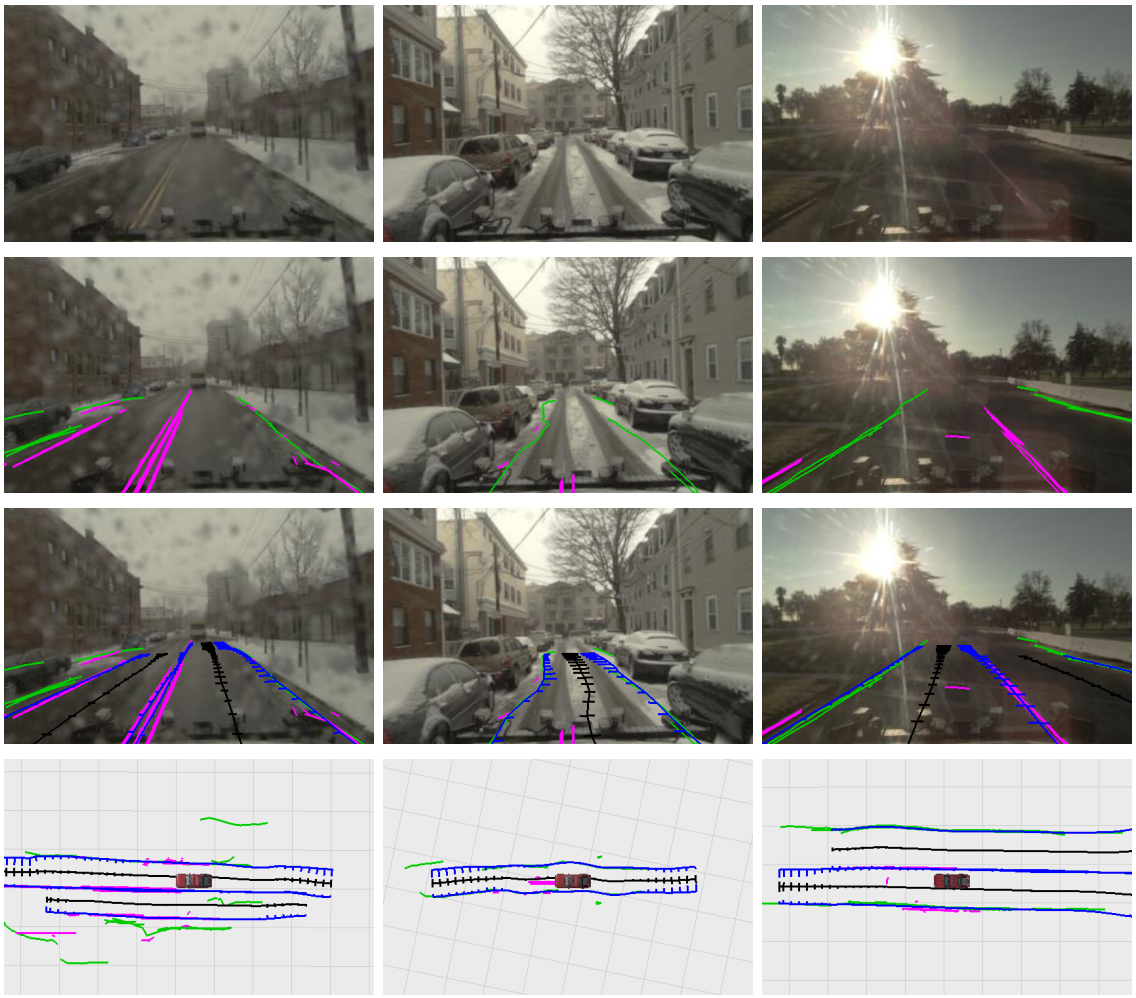


Figure 5-6: Lane estimation in adverse environmental conditions with falling snow (left), snow-covered roads (middle), and bright sunshine (right). Road paint (magenta) and curb fragments (green) are often complementary for maintaining lane centerline (black) and boundary (blue) estimates.

5.4 Evaluation

Qualitative results for lane estimation are given in Figures 5-4, 5-5, and 5-6. These images illustrate lane estimation in a variety of situations including wide suburban roads, dirt roads, and crowded urban streets. In addition to testing on days with clear skies and indirect sunlight, we also show results for operating in snow storms and with harsh direct light.

To objectively assess the performance of our system, we evaluated it on two datasets. The first dataset, which we refer to as UCE_MISSION_1, was collected during the first of three autonomous missions that Talos completed during the 2007 DARPA Urban Challenge. It consists of 30.2 km of travel in 182 minutes through the suburban environment described in Chapter 3. This dataset can be characterized by wide lanes, no pedestrians, generally well-marked roads, and few vehicles. It was collected on the morning of November 4, 2007, when the sun was initially low in the sky. In addition to suburban roads, the vehicle also traverses a short 0.4 km dirt road and a 1.7 km stretch of highway.

The second dataset, which we refer to as the CAMBRIDGE dataset, was collected on the afternoon of September 2, 2009 during a manual drive through the city of Cambridge, Massachusetts. It is 58 minutes long and spans 13.6 km of travel through a densely populated urban environment. This dataset can be characterized by roads of varying quality, large numbers of parked and moving vehicles, and many pedestrians.

Ground truth for both datasets consists of a densely labeled road network and a listing of the position and orientation of the vehicle at all times of travel through the road network. These were obtained by the method described in Appendix A.

The two vision-based road-paint detectors and the LIDAR-based curb-detector described in Chapter 2 were used to provide road-paint and curb features. We first consider results where the road-paint feature detectors operated on images from a single forward-facing wide-angle (approximately 90° FOV) camera, and curb-detector operates on the Velodyne HDL-64E data.

The road paint and curb features were used as input to the lane estimation algorithm described in this chapter. For comparison, we also ran the lane estimation algorithm described in Chapter 3 on the same inputs. We refer to these algorithms as the basis curve (BasCurv) and evidence image (EvImg) algorithms, respectively. Although the evidence image algorithm was described for use with a weak geometric prior, it can also be used as a standalone lane estimation system by taking the centerline candidates produced by the RANSAC curve fitting step as lane estimates of fixed width. These centerline candidates are used for comparison with the basis curve algorithm.

5.4.1 Lateral error

The lateral error of a lane centerline estimate at a given point on the estimate is defined as the shortest distance from the estimated centerline to the true centerline of the nearest lane. Since an autonomous vehicle will in general attempt to follow the lane centerline, the error of this estimate can be seen as a measure of how well the

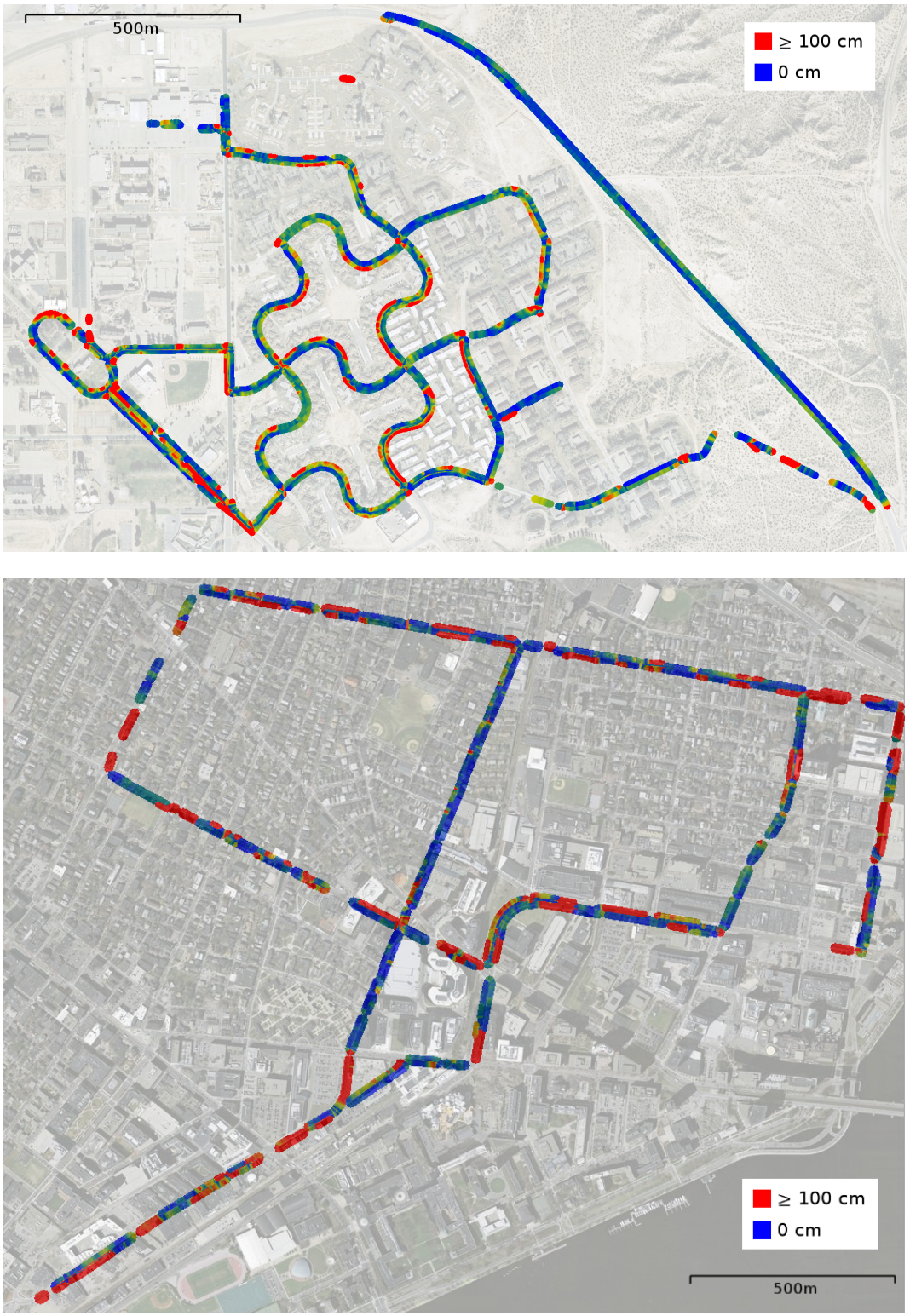


Figure 5-7: Mean lane centerline error for the (top) UCE_MISSION_1 and (bottom) CAMBRIDGE datasets for the basis curve algorithm.

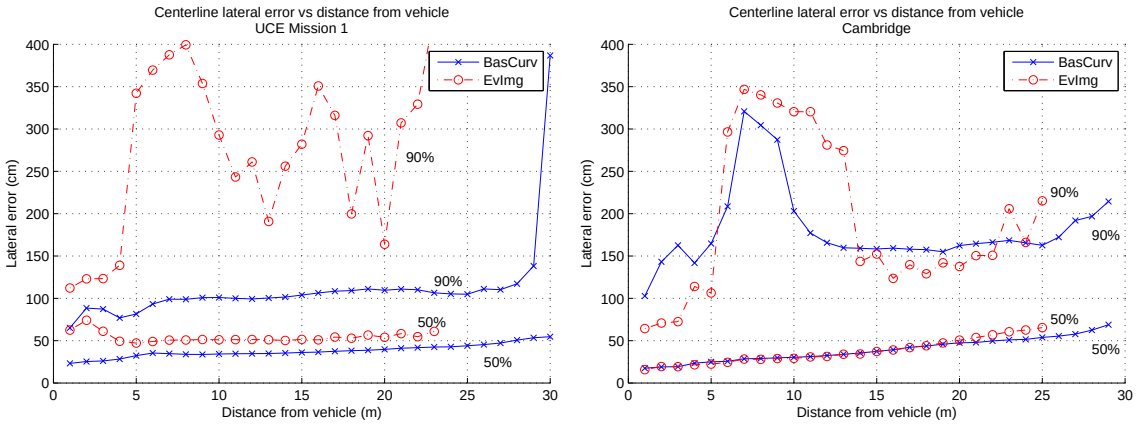


Figure 5-8: The 50 and 90 percentile values for centerline lateral error, as a function of increasing distance from the vehicle. UCE_MISSION_1 dataset shown on the left, CAMBRIDGE dataset shown on the right.

lane estimation system guides the vehicle when the system provides a lane estimate. Figure 5-7 shows the centerline lateral error of the basis curve algorithm on the UCE_MISSION_1 and CAMBRIDGE datasets.

Figure 5-8 shows the 50 and 90 percentile values for the centerline lateral error of the evidence image and basis curve approaches as a function of distance from the vehicle. The basis curve approach significantly outperforms the evidence image approach in the UCE_MISSION_1 dataset, but the performance is almost identical in the CAMBRIDGE dataset.

The discrepancy in centerline error across datasets can largely be explained by the different structure of the road networks, in particular the lane width. In the CAMBRIDGE dataset, 94% of the lanes in the road network were between 3.0 m (10 ft) and 3.7 m (12 ft) wide, compared to only 33% in the UCE_MISSION_1 dataset. The lanes in the UCE_MISSION_1 dataset were much wider, where 63% were between 4.2 m (13 ft) and 5.4 m (18 ft) wide (Figure 5-9). Although 5.4 m may seem very wide for a lane, wide lanes are more common in suburban areas where the side of the road is often used for parking. The strong prior on lane width assumed by the evidence image approach is largely satisfied by the CAMBRIDGE dataset, and much less so by the UCE_MISSION_1 dataset.

Although the median lateral error for both approaches in the CAMBRIDGE dataset is fairly small, looking at a higher error percentile reveals a significant number of lane estimates with a lateral error up to 3.5 m. These errors correspond to a higher rate of false lane detections in both approaches. These false detections were almost always made on shoulder or bicycle lanes, hence the lateral error of approximately one lane width. Additionally, the much weaker lane width prior assumed by the basis curve algorithm sometimes resulted in lane detections spanning two lanes. This was more common on four-lane roads where the dashed markings separating adjacent travel lanes were not detected by the road paint detectors.

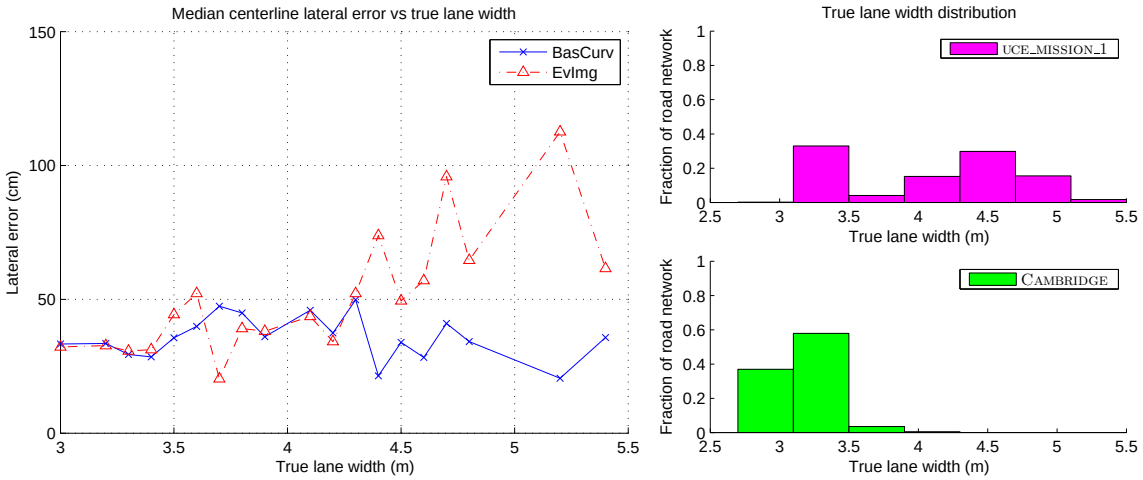


Figure 5-9: (left) Median centerline lateral error as a function of true lane width. (right) Lanes in the UCE_MISSION_1 dataset are considerably wider than lanes in the CAMBRIDGE dataset, accounting for much of the performance discrepancy in Figure 5-8.

5.4.2 Lookahead distance and time

In addition to the accuracy of the lane estimates produced, we are also interested in how frequently the lane estimation system produces lane estimates, and how far in front of the vehicle the estimates are projected. We refer to the *lookahead distance* of a lane estimate as the distance from the vehicle to the farthest point of the lane estimate that lies forward of the vehicle. Additionally, we refer to the *lookahead time* as the lookahead distance divided by the vehicle's instantaneous speed. Intuitively, the lookahead distance indicates how much farther the vehicle can travel before reaching the end of its current lane estimate, and the lookahead time indicates how long this would take, assuming constant velocity.

In general, the vehicle's situational awareness is directly related to the lookahead distance and lookahead time of the lane estimates. For a fully autonomous vehicle navigating using only its lane estimates, the lookahead distance can also be used to determine a safe travel speed. By traveling no faster than the maximum vehicle speed with a stopping distance less than the lookahead distance, the vehicle can guarantee that it always travels inside a lane estimate, and select a different navigation strategy (e.g. simple obstacle avoidance) when no lane estimates are available.

Figure 5-10 shows the lookahead distance and lookahead time cumulative distributions of the two algorithms for the CAMBRIDGE and UCE_MISSION_1 datasets. In all cases, the basis curve algorithm outperforms the evidence image algorithm, and is able to provide some lane estimate forward of the vehicle for 74% of distance traveled in the UCE_MISSION_1 dataset, and 63% of distance traveled in the CAMBRIDGE dataset. In comparison, the evidence image algorithm only produces centerline candidates forward of the vehicle for 31% and 46% of distance traveled, respectively. Note that we do not expect the lookahead distance of the evidence image approach to

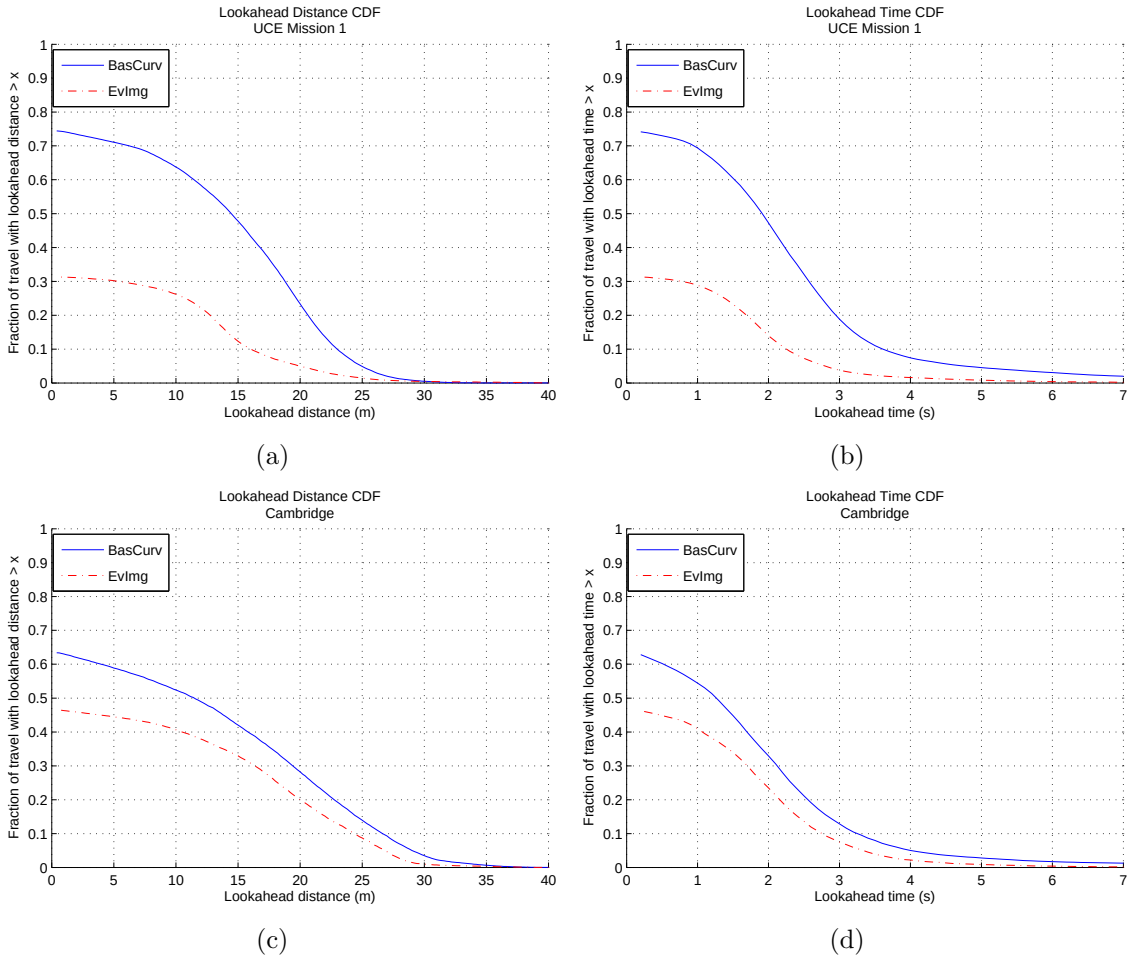


Figure 5-10: Lane estimation lookahead distance (left column) and time (right column).

match the performance given in Chapter 3, as the implementation described in this chapter was run with only one camera instead of five.

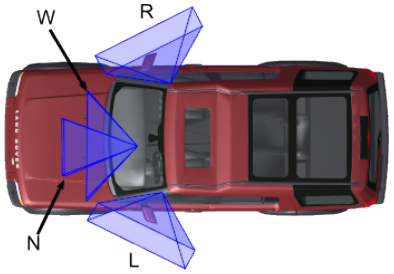
As shown in the previous section, the basis curve algorithm provides lane estimates of equal or better accuracy to the evidence image approach, but does so more often and with a greater lookahead. We attribute this to the better data association and outlier rejection properties of the basis curve algorithm. Instead of allowing every road paint and curb feature an equal "vote," as the evidence image approach does, the basis curve algorithm explicitly attempts to detect outliers and reject their influence on the final lane estimate.

5.4.3 Multiple cameras

The lookahead distance of a lane estimation algorithm is limited by the range of its input features. The feature detectors in turn are limited by the range and resolution of the sensors on which they operate. Here, we consider the use of additional cameras in a second experiment.

The road-paint feature detectors were each run on four on-board Point Grey Firefly MV cameras. Cameras varied by focal length, position, and orientation. Two forward-facing cameras mounted above the rear-view mirror differed by focal length. Two additional cameras were placed above the left and right side mirrors, respectively, and were oriented to point slightly down and away from the vehicle. Figure 5-11 shows the calibrated camera extrinsics and viewing volumes. All cameras had an image resolution of 752×480 pixels, and acquired images at 22.8 Hz.

The curb detector was also run on the Velodyne HDL-64E data, identical to the first experiment described.



Camera	Horizontal FOV	Pitch
W	87.4°	2.7°
N	41.3°	3.0°
R	87.1°	14.9°
L	87.4°	14.7°

Figure 5-11: (left) Camera viewing volumes. (right)

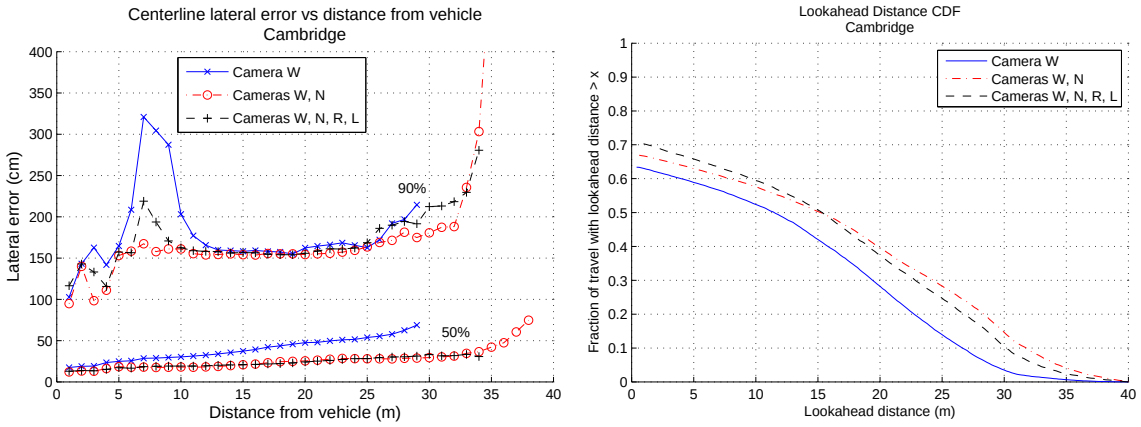


Figure 5-12: Median and 90% centerline error (left) and lookahead distance (right) when using one, two, and four cameras. Camera calibrations are given in Fig. 5-11.

Figure 5-12 shows the median and 90% centerline error and lookahead distance when the basis curve lane estimation algorithm was run with one, two, and four cameras as input to the road-paint detectors and the Velodyne as input to the curb detector.

The results show that using two forward-facing cameras with different focal lengths provides a modest improvement in centerline error over using a single wide-angle camera alone. For example, median centerline error at 25 m from the vehicle was reduced from 54 cm to 28 cm. As expected, the lookahead distance shows a significant improvement, with the vehicle maintaining some forward lane estimate for 68% of distance traveled using the two cameras, as opposed to 63% of distance traveled

using one camera. The median lane estimate lookahead increased from 11.3 m to 15.6 m.

Two additional side-facing cameras did not significantly improve the centerline error, and in some cases resulted in an increased error. This can be attributed to a slight increase in the number of falsely detected lanes on the side of the road such as in breakdown lanes and parking spots. While the side-facing cameras increased the fraction of travel with lane lookahead up to 15 m forward of the vehicle, the lookahead rate for beyond 15 m decreased slightly over using the two forward-facing cameras alone. This be attributed to processing time; as the number of feature observations near the vehicle increased as a result of the additional side-facing cameras, the estimation algorithms were slower to incorporate updates from the forward-facing cameras.

Overall, the additional forward-facing narrow-angle camera significantly improved the centerline error and lookahead distance. The side-facing cameras improved overall "near-field" awareness, but not centerline error.

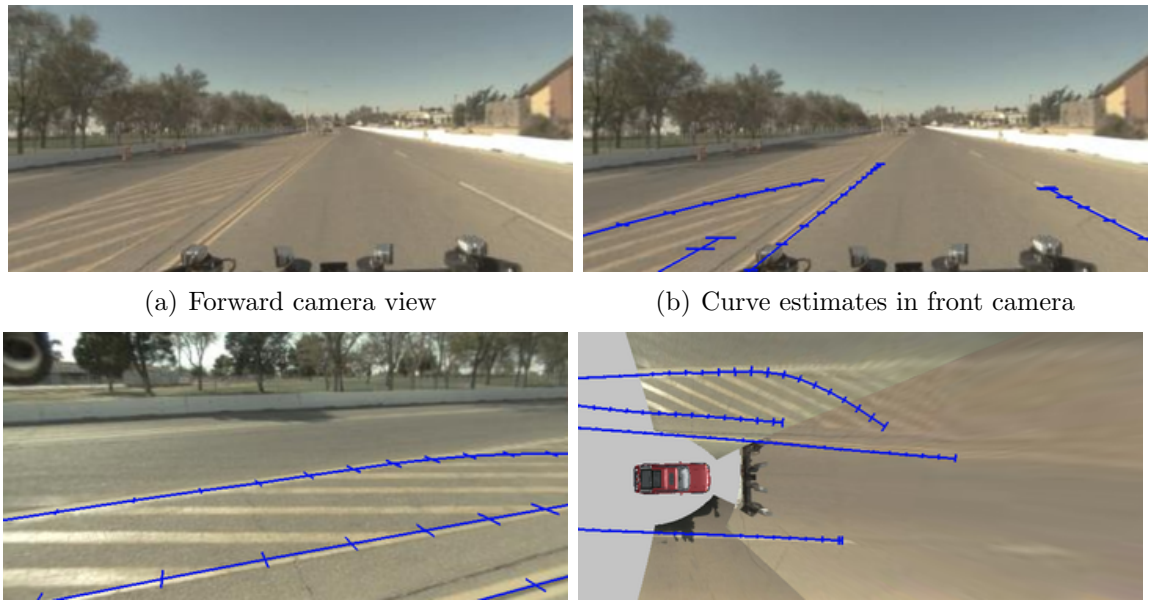
5.5 Discussion

In applying the basis curve estimation algorithm described in the previous chapter to the lane estimation problem, we have made numerous assumptions to both formulate the problem appropriately, and to simplify the implementation. In general, we introduce independence relations for convenience and to allow fast computation. Control points within a boundary curve or lane are assumed to be mutually independent, and lanes themselves are also assumed to be mutually independent. Observation error is also assumed to be mutually independent, and the Gaussian distribution is used to approximate the true error distribution to allow closed-form belief updates.

In reality, these assumptions result in models that are rough approximations of the true distributions. Observation error does not actually conform to a Gaussian distribution, and tends to be highly correlated over short time windows. The geometry of one lane provides significant information about the geometry of adjacent lanes, information that our system ignores. Despite these approximations and others, our implementation is able to robustly estimate lanes and boundary curves in the presence of noisy inputs and a variety of real-world conditions. False detections from shadows and other sources of error can themselves be estimated and ignored. Partial observations of lane boundaries that result from occlusions or poorly marked roads can still be used to provide useful lane estimates.

5.5.1 Data Association

We have formulated the lane estimation problem in such a way that standard estimation and tracking algorithms such as the Kalman filter can be used to estimate complex lane geometries. In doing so, we gain all of its advantages and also invite all of its shortcomings. Figure 5-13 shows a typical failure resulting from the unimodal nature of the Gaussian distribution. A strip of road paint is correctly detected, but



(a) Forward camera view

(b) Curve estimates in front camera

(c) Curve estimates in left camera

(d) Synthesized overhead view

Figure 5-13: A feature is erroneously associated with a boundary curve. The road paint is detected correctly, but the wrong detection is used to update a curve estimate.

erroneously associated with a lane boundary. The system is unable to recover from this error, as it assigns virtually zero probability to the correct curve.

A similar failure is shown in Figure 5-14, where a false detection occurs on a tree shadow that happens to lie almost parallel to the true boundary. The shadow is erroneously associated with the lane, and causes the lane estimate to narrow. The true boundary is observed with a curb detection shortly after, but is not correctly matched to the lane. Each row in the figure shows one frame of the camera, detected features, and lane estimates.

Many of these difficulties have also been widely studied in other domains, and it should also be possible to apply lessons learned in other topics to lane estimation with basis curves. One approach that might have better success here is a particle filter, which has been successfully applied in many estimation and tracking problems to model complex distributions and provide multi-hypothesis tracking capabilities.

As is the case for any practical Kalman filtering application, a key challenge is to manage the system confidence and correct for errors that arise from the modeling approximations. Accounting for correlated observation error to prevent overconfidence is a particularly difficult task, as the correlations are usually unobservable. Heuristics such as the fudge factor mentioned in Section 5.3.4 and discarding updates when the vehicle is stopped or moving very slowly can help, but devising more reliable and accurate methods to maintain more accurate and valid confidence measures is still an open question.

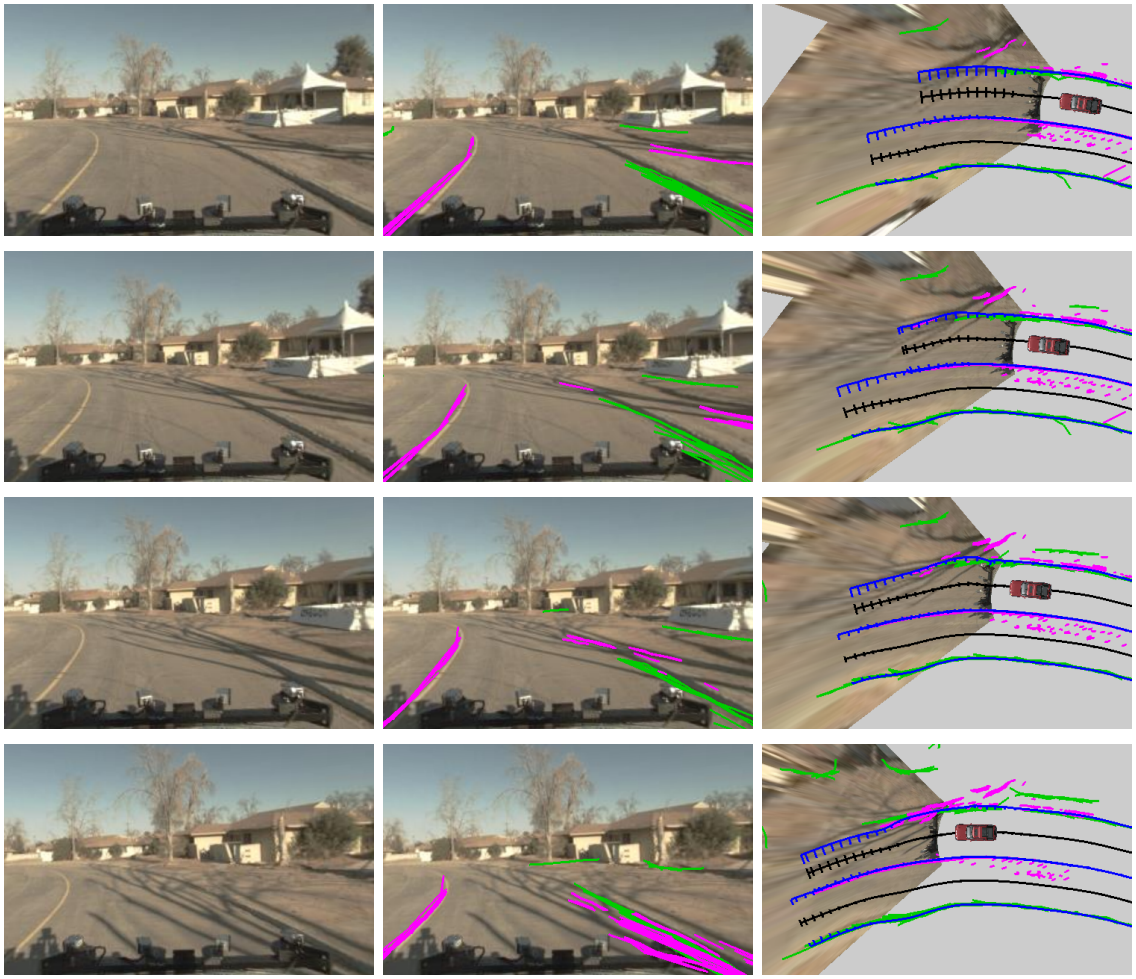


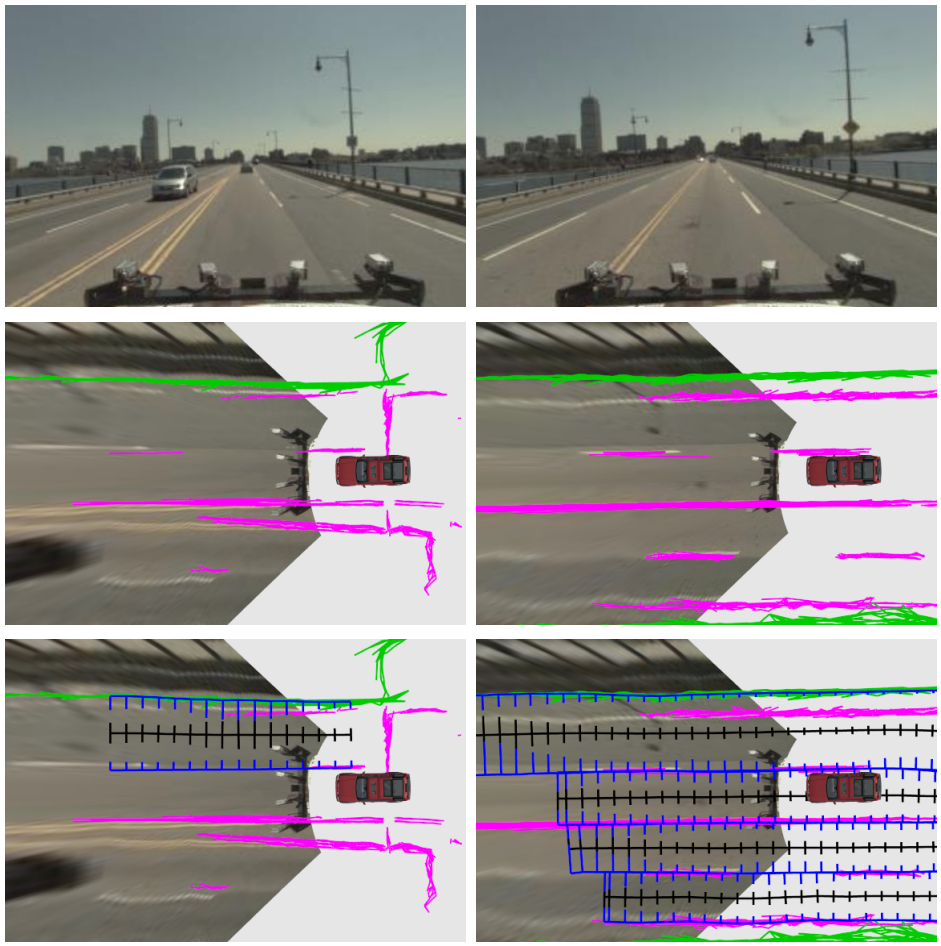
Figure 5-14: As is the case with many data association problems, the most difficult outliers to detect are those that happen to be very similar to the true inliers.

5.5.2 Lane Detection

The algorithms described in this chapter have focused primarily on how to maintain a lane estimate with new observations, given the initial estimate. We have described a simple method for detecting lanes from features that is successful in a variety of situations, but also fails in others. Figure 5-15 illustrates a situation where a lane estimate is initialized from a dashed lane boundary on the left of the vehicle, and a curb to the right. While this is plausible, the initialized lane estimate is too wide and spans both the correct lane and also the road shoulder. Several seconds later, the true lane boundary is well-observed by the road-paint detector, but the basis curve lane estimation algorithm continues to "track" the curb as a boundary. The curb observations continue to provide a close match for the lane estimate that result in a plausible lane estimate.

While a simple heuristic to re-initialize the lane estimate on the closest two boundary curves might address this scenario, other situations may be more difficult. For example, if the old lane markings on a recently repainted road are still visible, then

it may not be obvious even to human drivers which lane markings denote the true lanes. These situations illustrate the need for an explicit procedure to reason about the most likely set of nearby lanes, given the current observations. The lane estimates may start off as plausible, given the initial observations, but as more information is later received, a more likely configuration may emerge.



t = 0 t = 5

Figure 5-15: Simple model tracking is not always sufficient for lane estimation. (left) Strong curb detections on the side of the road result in a lane detection that spans both the correct lane and also the road shoulder. (right) Even though the correct boundary is observed several seconds later, continued and persistent curb detections result in a high-confidence lane estimate that has "locked" on to the wrong boundary curve.

5.6 Summary

This chapter described an application of the basis curve estimation algorithm to the lane estimation problem and a detailed evaluation of its performance on two real-world

datasets. A quantitative comparison with a previously used lane estimation approach shows distinct advantages of the basis curve algorithm, particularly for estimating lanes using partial observations, noisy data with high false positive rates, and for jointly estimating centerline geometry and lane width.

The algorithms described have limitations of their own, and in particular are not able to correctly reject false positives that have a similar geometry to true lane boundary markings. A common failure occurs when a lane estimate is initialized on an incorrect but well-observed boundary curve, resulting in a persistent error. Higher-level estimation algorithms can be expected to address situations such as this, by reasoning about the most likely lane configuration, given the current estimates and observed features.

Chapter 6

Conclusion

By organizing and orienting traffic into flows sharing a common travel direction, speed, or other intent, lanes serve as a critical component in safe and effective travel. Dividing a roadway into multiple clearly-marked lanes allows vehicles to reliably predict the motion of other vehicles, and provides valuable cues on safe regions of travel. Lane infrastructure has been so effective that vehicles routinely pass within meters of each other while traveling in opposite directions at closing speeds of more than 200 km/h on a curved roadway. In other situations, such as 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.

A system that can estimate semantic, topological, or geometric aspects of lane geometry and a road network has valuable applications for land-based travel. Such a system could be integrated into vehicle safety systems, road-surveying equipment, driver-assistance and navigation programs, and fully autonomous vehicles. We take an early step at extending lane estimation to multiple lanes and environments with complex lane geometries such as urban spaces, and do so in real-world conditions.

6.1 Contributions

We have described the detection and estimation of lane geometry as a composition of model selection, feature detection, lane detection, and lane tracking. This thesis presented novel curve representations, feature detectors, algorithms for lane geometry estimation, and a ground truth dataset for large scale quantitative analysis of a lane estimation system. The feature detectors can operate in real-time on high resolution images, and false positives are reduced by incorporation of LIDAR-detected obstacles and computation of the solar ephemeris. We have demonstrated an initial version of our algorithms and incorporated them into a closed-loop system that guided a fully autonomous vehicle through the 2007 DARPA Urban Challenge (DUC).

Our experience in the DUC provided valuable insight into some of the primary challenges in lane estimation, and guided the design of a novel method for representing curves as variations of a basis curve. In a basis curve space, curve observations are linear transforms of the true curve geometry, and complex curves can be robustly

estimated with standard Kalman filtering techniques. We described the application of this technique to lane estimation, and demonstrated a clear improvement over our previous in a direct comparison on manually labeled ground truth datasets.

6.2 Future Work

While immediately useful to driver assistance technologies and other applications such as road surveying, robust estimation of lane geometry is only one of many capabilities needed for a fully autonomous vehicle to navigate an arbitrary, unfamiliar road network. To date, most studies in lane estimation have focused on lane geometry, and largely ignored topological aspects of a road network, such as intersections, forks, and merges. Dolgov and Thrun have recently proposed a method for inferring topological structure of a road network in large parking lots [18], although there have not yet been any results for estimation of road network topology on typical public roads.

Automatic estimation of semantic aspects of a road network, such as direction of travel and lane usage have yet to be approached by all but the simplest methods (e.g. assume that vehicles travel on the right side of the roaad). While much of this information could be obtained by appropriately strong priors embedded in a road map, a great deal of semantic information can be reliably obtained only by sensing local street signs and other cues such as traffic lights.

Even within the topic of lane geometry, there is much work to be done. The feature detectors we have proposed have a high false positive rate, and still miss many valuable lane markings. Particularly, faded road paint, eroded curbs, and lane markings that do not fit the detection priors encoded in the feature detectors present a challenge. The feature detectors have limited range, and estimates could be improved to reliably detect farther features, even with their current sensor resolutions. Additionally, LIDAR intensity data could be used to detect highly reflective road paint in situations where the cameras are blinded by sunlight, or the algorithms confused by shadows [51].

Road paint and curbs are not the only sources of information about the roadway and nearby lanes. Other vehicles, both moving and stationary, provide valuable information about the road network. In cases where the lanes are not marked or are otherwise obscured (e.g., due to water or snow cover), vehicles may self-organize into lanes.

Currently, our feature detectors operate independently of the lane detection and estimation algorithms, and receive no feedback. The features accepted or rejected by basis curve lane estimation algorithm for incorporation into the lane estimates could be used as real-time feedback to train feature detectors in real-time. Intuitively, if a set of features fail the feature detector thresholds, but are the only features that could plausibly serve as lane boundaries, then this provides evidence that the feature detector thresholds should be adapted.

The lane detection algorithm described in Chapter 5 is based on a handful of manually adjusted heuristics. While effective for a variety of scenarios, it could be improved to detect lanes from the tracked boundary curves and other features much

more quickly and reliably than it currently does. In many cases, a two-lane road is not marked by a dividing lane marker; determining that there are two lanes instead of one, or generating a distribution over the possible number of lanes, would be useful in these scenarios.

The lane tracking algorithm, as mentioned in the previous chapter, enjoys all the benefits of a problem well-suited for Kalman filtering, but also all of the disadvantages. Outliers that could plausibly be lane markings (e.g., tree shadows that are almost parallel to the roadway) can cause the estimates to diverge. Multi-hypothesis trackers would be well-suited for handling many of these situations.

Appendix A

Ground Truth

In an effort to provide a more rigorous analysis of our lane estimation algorithms and allow for meaningful large-scale comparisons between algorithms, we have prepared ground truth data sets that contain the geometry of lanes within a road network and the pose of the vehicle relative to these lanes. Roughly speaking, the data sets provide a highly accurate road map, along with the position and orientation of the vehicle within this map as it travels and collects data. This chapter describes the creation of these data sets.

Each ground truth data set is constructed in two stages. First, a road network is defined by annotating geo-registered ortho-rectified aerial imagery (ortho-imagery). Second, the pose of vehicle within the road network is determined by superimposition of the vehicle's sensor data onto the ortho-imagery and manual adjustment of the pose until the two data sources align sufficiently well.

Ortho-imagery is obtained from Google Maps [46] and the Massachusetts Executive Office of Transportation [14]. A graphical tool displays the imagery and allows a user to pan and zoom (Figure A-1). Using mouse and keyboard commands, the user can draw lanes on top of the ortho-imagery, modify the width of a single portion of the lane or the entire lane at once, and modify the lane geometry. The design of this tool is inspired largely by that of popular image- and vector-based software packages such as Adobe Photoshop and Illustrator. Lane geometry is modified by placing and adjusting control points of cubic hermite splines.

Once a road network is created, the pose of a vehicle within the road network for a particular data collection is determined. Because the ortho-imagery is geo-registered, the GPS pose of the vehicle reported by the on-board Applanix navigation system usually provides good initial estimates. Further adjustment is almost always required, as additional errors arising from mis-projections in the ortho-imagery compounded with GPS error typically result in an initial error of up to several meters.

A second tool was developed that allows a user to view local sensor data superimposed onto the ortho-imagery, and manually correct the estimated vehicle GPS pose. Discrete adjustments can be made at arbitrary points throughout a data set, and the adjustments are linearly interpolated at points in between. Figure A-2 illustrates this process. In this example, the Applanix-derived pose of the vehicle has an initial error of approximately two meters. Once the camera data is projected onto an assumed

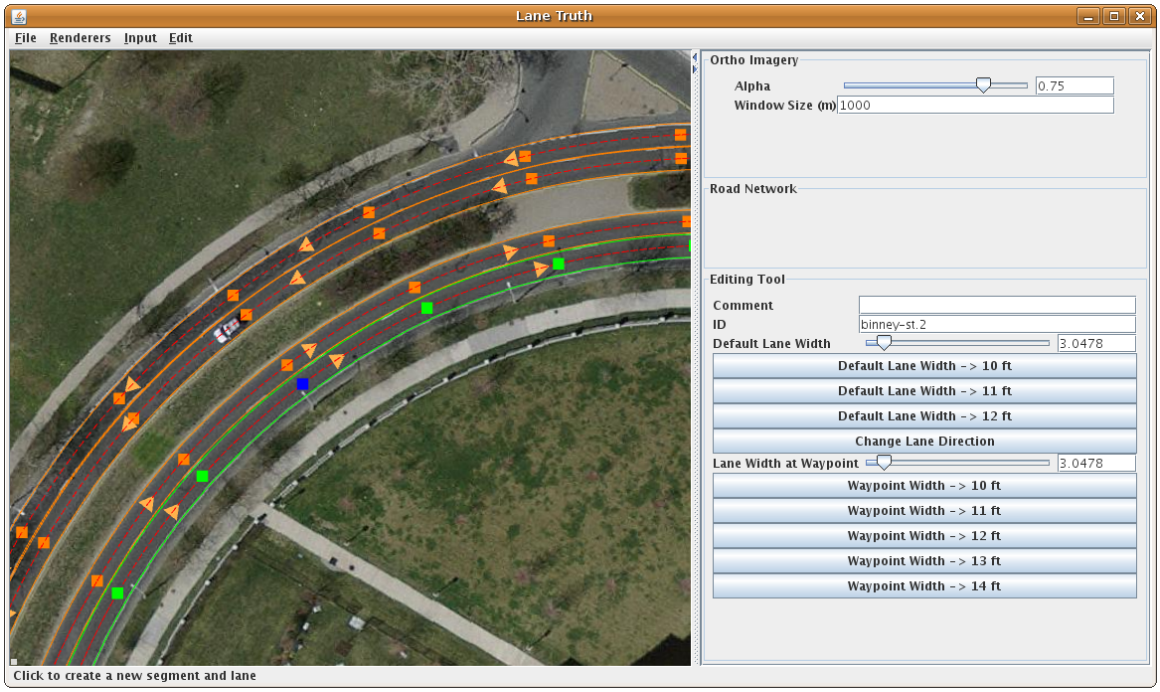
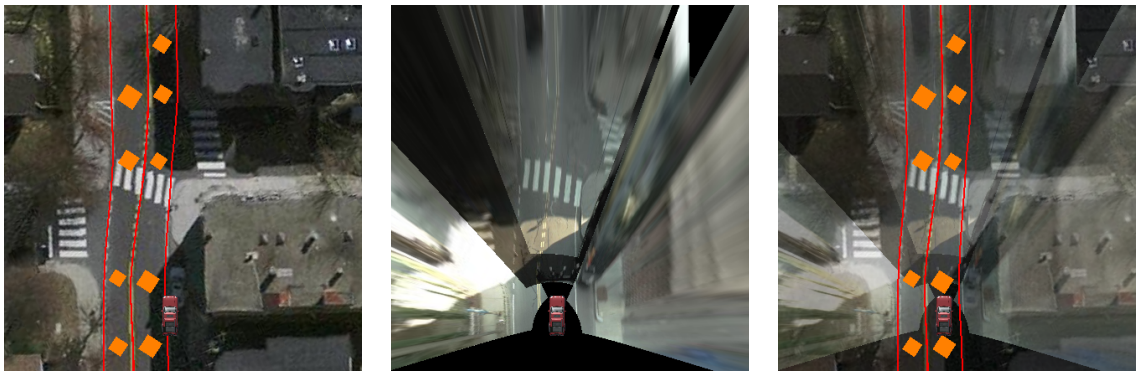


Figure A-1: A graphical tool displays geo-registered ortho-rectified aerial imagery. Users can create highly accurate labelings of lane geometry by manipulating the control points of cubic splines.



(a) (b) (c)

Figure A-2: (a) A high-end GPS/INS device provides a useful, but still incorrect, initial guess for the vehicle pose within our ground-truth road network. (b) Live imagery from high resolution onboard sensors projected onto a ground surface allow a user to manually correct the initial pose estimate. (c) The sensor data superimposed on the ortho-rectified aerial imagery, illustrating the pose of the vehicle within the ground-truth road network.

ground plane, it can be manually registered to the ortho-imagery and provide a much more accurate pose of the vehicle relative to the ortho-imagery and road network.

We note that what we are calling ground truth lane geometry is perhaps more

accurately described as how a human would indicate 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.

These tools were used to create two ground truth datasets spanning 43.8 km of vehicle travel over a total of 4 hours. The first dataset was collected on Nov. 4, 2007, during the 2007 DARPA Urban Challenge in Victorville, California. We refer to this dataset as the UCE_MISSION_1 dataset. Ortho-imagery for the UCE_MISSION_1 dataset was acquired from Google Maps. The second dataset was collected on Sep. 2, 2009, during a drive through Cambridge, Massachusetts. We refer to this dataset as the CAMBRIDGE dataset. Figures A-4 and A-3 show the two road networks overlaid on aerial imagery.

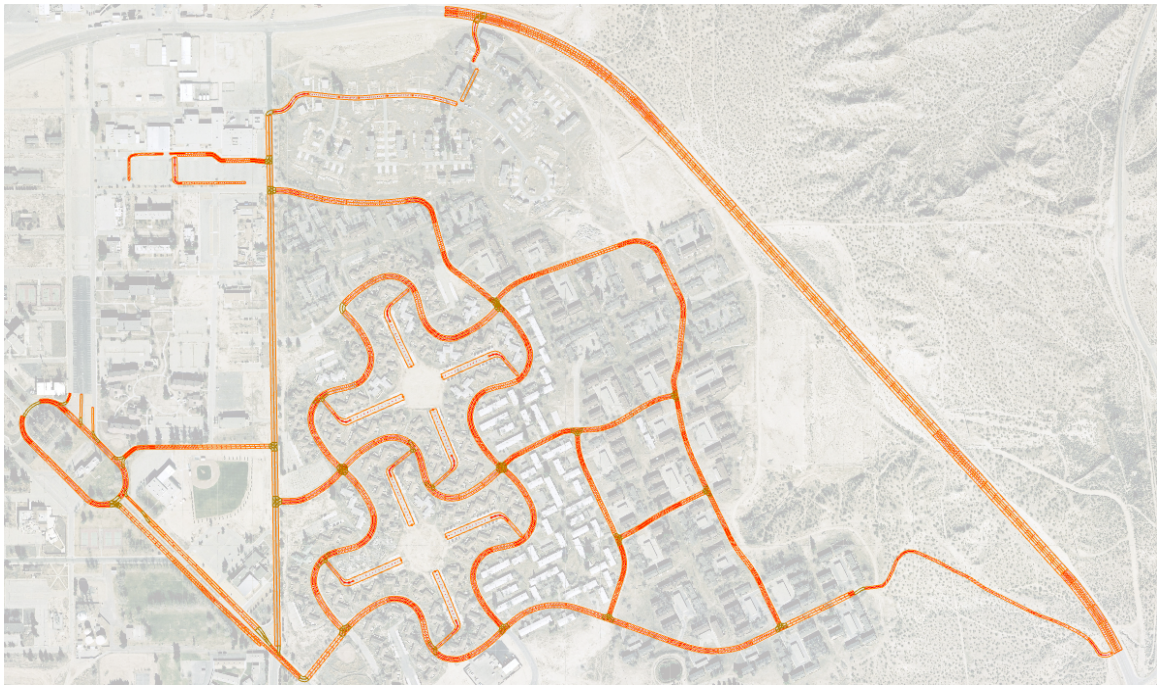


Figure A-3: Ground truth road network for the UCE_MISSION_1 dataset.

Figure A-4: Ground truth road network for the Cambridge dataset.

Appendix B

Terrain Estimation

This chapter describes a simple terrain estimation algorithm that can be used to project vision-detected features from image space into a local coordinate frame. While a variety of sophisticated techniques exist for estimating terrain from sparse measurements, the Velodyne HDL-64E LIDAR sensor provides such a densely sampled point cloud of the environment surrounding the vehicle that the simple approach described in this chapter is sufficient for our application.

The estimation algorithms presented in this thesis operate on vision-detected road paint features. To be useful as input to our algorithms, these features must be projected from image space into a 3D local frame in which the robot's pose is known. Unfortunately, this is a non-trivial task, as camera data alone does not provide sufficient information for the projection. In many cases, the system can make an assumption such as a perfectly level ground plane and achieve accurate results. As evidence, the lane estimation system described in Chapter 3 and implemented for the 2007 DARPA Urban Challenge made this assumption and successfully navigated a vehicle for 90 km of travel.

However, there are still numerous situations where these simple assumptions do not hold. As can be seen in Figure 3-6, the Urban Challenge environment was largely level ground, and the lane estimation system failed in areas with significant roll or pitch. On these roadways, projection onto an incorrect ground surface will fail and yield incorrect results that could in turn confuse the lane estimator (Figure B-1). In this case, a more accurate way to project features into the local frame is desired.

Inferring depth from image data is one of the most difficult and long-standing problems in machine vision, and remains a highly active area of research. In the road and lane estimation problem space, one approach is to assume that road and lane boundaries are parallel, and use this constraint to estimate depth [27]. Similar to the flat-world assumption, this approach is usually, but not always, valid. When boundaries are not parallel, such as at forks, merges, and intersections, this approach fails.

General techniques for computing the depth of an imaged feature such as stereo and structure from motion have been useful in many applications [25], and may prove useful in this domain. However, our data collection platform was not equipped with a stereo camera, thus preventing exploring stereo techniques with our existing data.

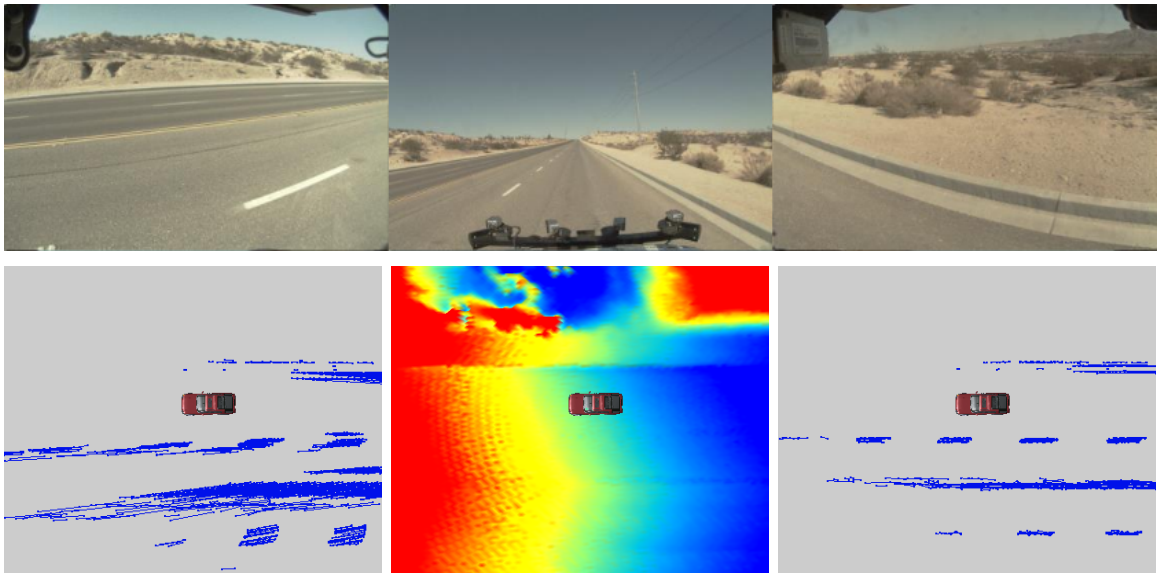


Figure B-1: (top) Images from the left, center, and right cameras. (left) Naively projecting vision-detected features onto an assumed ground plane results in significant projection error. (center) A terrain model constructed from Velodyne LIDAR data, colored by height. (right) Projecting vision-detected features onto the terrain model yields more accurate results.

One alternative is to use the relative motion estimates provided by the vehicle's navigation system to reduce the structure from motion problem to that of structure from known motion. Unfortunately, we are largely concerned with the structure of smooth curves that have few distinguishing features along the length of the curve. This introduces a matching ambiguity that complicates the estimation of curve motion in image space (the well known Aperture problem). Even if we use a naive matching approach such as matching the closest points on two curves across images, there is still ambiguity when the motion is along the optical axis of the camera, a common situation for forward-pointing cameras.

Terrain estimation has also been approached from the perspective of sensing with LIDAR data. Plagemann and Lang have proposed using Gaussian processes for non-parametric terrain modeling [40, 59], although these methods would be computationally expensive and not suitable for the one million range measurements generated by the Velodyne each second. More recently, Hadsell et al. demonstrated an efficient way to reason about LIDAR data as rays rather than just point clouds [24]. However, even their results are not clear about the suitability of their method for real-time implementation in a way that could utilize all of the available Velodyne data. Thus, there is currently a trade-off between producing accurate results and utilizing all of the data provided by a state of the art sensor.

Finally, we note that these LIDAR-based terrain approaches are intended to provide estimates of the load-bearing surface surrounding the robot, with the stated application of assisting terrain-adaptive planning algorithms. In contrast to these

applications, our requirements are much simpler, involving only the correct inference of the depth of a handful of curves in image space. Any error in the terrain model that does not involve these curves does not reduce its utility for this purpose. In light of this, we chose an approach that is sufficiently accurate for the depth estimation task, simple to implement, and runs in real-time on our existing platform.

B.1 Terrain model

Our approach to terrain estimation is a simple accumulator-based system that attempts to account for noisy measurements characteristic of the Velodyne LIDAR. Terrain is represented as a 2D height map, represented in memory as a grid of cells. The height of a point within the bounds of this map is defined to be the linear interpolation of the nearest four points. The coordinates of this map are maintained in the local frame, and can be easily maintained in memory by shifting rows and columns of the matrix, adding and deleting rows as the vehicle moves about.

Each point return generated by the LIDAR is mapped to the nearest grid cell and added to an array of point returns for the cell. Point returns within a cell are clustered by elevation, with a 10 cm maximum vertical range within each cluster (i.e. a cluster with points spanning more than 10 cm is broken into two clusters). The lowest cluster with more than 3 point returns is assumed to be the ground cluster for the cell.

If a cell does not have a ground cluster, then its elevation is determined by nearby cells with ground clusters. To compute this efficiently, the cells are arranged into a directed graph, where each node corresponds to a single cell. The parents of a cell consist of the cells that are both in its 8-neighborhood as defined by the grid, and that are geometrically closer to the vehicle. This results in an acyclic graph structure with a single root node that corresponds to the cell closest to the vehicle.

The elevation at each cell with a ground cluster is estimated as the elevation of the lowest point return in the ground cluster. If the cell does not have a ground cluster, then its elevation is estimated as a weighted average of the elevations of each of its parents. If the parent cell has a ground cluster, then it is assigned a higher weight. In our implementation, the weight of a neighboring ground cluster is defined as 10 times the weight of a neighboring non-ground cluster. The root node is a special case, and is always assigned an elevation corresponding to the vehicle's elevation. Assigning elevation estimates to nodes by breadth-first traversal guarantees that the elevation estimates for each of a cell's parents will have been computed prior to visiting the cell itself.

The graph traversal estimation procedure is run at 10 Hz on a 90 m x 90 m grid with cells spaced every 0.3 m. Each iteration has a runtime linear in the number of nodes in the graph, as each node is visited exactly once and has at most 3 parents. Fast update cycles allow a real-time implementation with high resolution grids, able to process the approximately one million range measurements generated by the Velodyne each second. Figure B-2 shows a sample Velodyne scan and the resulting terrain model constructed from this scan and several previous scans.

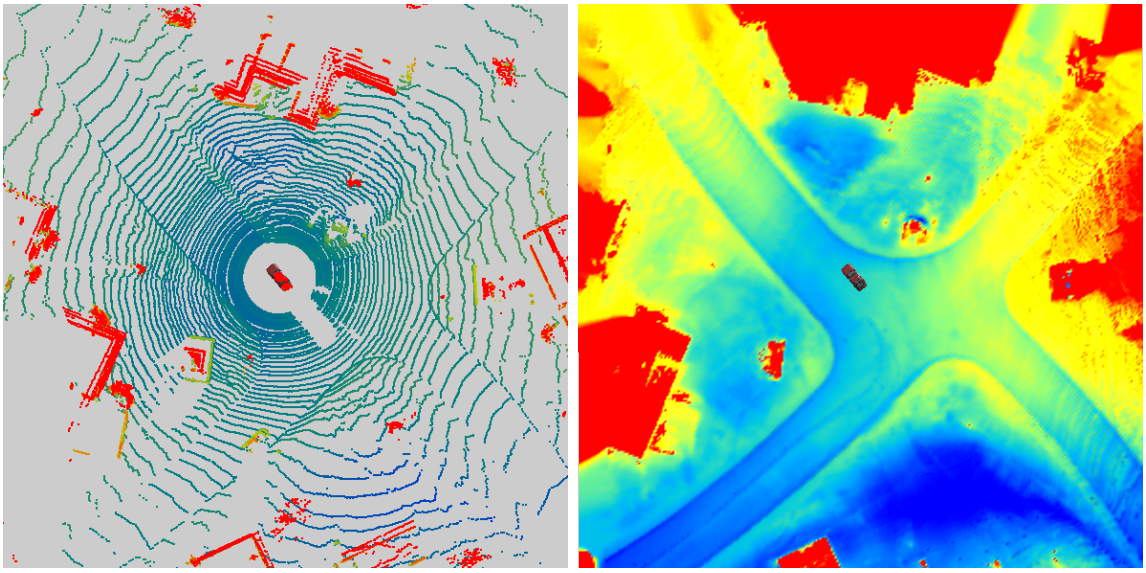


Figure B-2: (left) A single rotation of the Velodyne generates a dense point cloud. (right) Terrain model constructed from successive Velodyne scans.

B.2 Projecting features onto a terrain model

Once a terrain model has been estimated, it can be used as a surface mesh onto which features can be projected from image space. Projecting a point in image space onto the mesh corresponds to intersecting a ray with the mesh, a common operation in computer graphics. While ray intersection with an arbitrary mesh is a difficult problem, the grid structure of the terrain mesh permits a simple and efficient approach. The terrain mesh is first triangulated by dividing each cell into two triangles, separated by the diagonal. The local frame coordinates of the ray is computed, and projected onto the X-Y plane by discarding the Z coordinate. This flattened ray is used to construct a list of cells that potentially intersect the ray by computing a Bresenham line on the grid [12], starting from the ray's origin. This list is traversed in order of increasing distance from the ray origin, and each cell is checked for intersection with the original ray using a standard ray-triangle intersection algorithm [23]. The first intersections is assumed to be the source corresponding to the imaged point.

The Bresenham line can be computed quickly, and contains at most $2N$ cells, where N is the length and width of the square grid. Thus, each ray can be cast in time $O(N)$, allowing for fast projection onto a potentially complex surface model.

Acknowledgments

Seth Teller supervised this thesis, providing immense support, guidance, and mentorship throughout its development. In addition to Seth, the thesis committee includes Berthold Horn and Tomas Lozano-Perez, whose suggestions and insight have been invaluable in refining the concepts presented in this work.

All experiments were conducted on the MIT 2007 DARPA Urban Challenge race vehicle, and would not have been possible without the concerted efforts of the entire MIT DARPA Urban Challenge team. Besides the author, these team members include John Leonard, Matthew Antone, David Barrett, Jonathan How, Troy Jones, Mitch Berger, Bryt Bradley, Ryan Buckley, Stefan Campbell, Alexander Epstein, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, Keoni Maheloni, David Moore, Katy Moyer, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Robert Galejs, Siddhartha Krishnamurthy, Seth Teller, and Jonathan Williams.

In addition to those mentioned above, a great number of people assisted, either directly or indirectly, the development of this thesis. Matt Walter, Olivier Koch, Edwin Olson, David Moore, Luke Fletcher, and Matthew Antone were instrumental in exploring, clarifying, and identifying many of the key concepts and ideas. The various residents and members of 32-33x who have provided feedback on presentations and paper drafts include Abe Bachrach, Alex Bahr, Jonathan Brookshire, Emma Brunskill, Ruijie He, Been Kim, Thomas Kollar, Sam Prentice, Nick Roy, and Alec Shkolnik.

Finally, the importance of friends and family cannot be overstated. Special thanks go to Stacy Wong, my sister and parents, Jennifer Hsieh, Angelina Lee, and Larry Rudolph.

Portions of Chapters 2 and 3 were adapted from previous publications [28, 29]. Preliminary versions of the concepts presented in Chapter 4 have also appeared in a previous publication [30].

Bibliography

[1] Defense Advanced Research Projects Agency. Route network definition file (RNDF) and mission data file (MDF) formats. http://www.darpa.mil/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf, Mar. 2007.

[2] Nicholas Apostoloff and Alexander Zelinsky. Robust vision based lane tracking using multiple cues and particle filtering. In *IEEE Intelligent Vehicles Symposium*, pages 558–563, June 2003.

[3] Nicholas Apostoloff and Alexander Zelinsky. Vision in and out of vehicles: Integrated driver and road scene monitoring. *Int. Journal of Robotics Research*, 23(4-5):513–538, Apr. 2004.

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

[5] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[6] Richard H. Bartels and John C. Beatty. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

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

[8] Massimo Bertozzi, Alberto Broggi, and Alessandra Fascioli. Vision-based intelligent vehicles: State of the art and perspectives. *Robotics and Autonomous Systems*, 1:1–16, 2000.

[9] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[10] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, 1998.

[11] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *Int. Journal of Robotics Research*, 23(12):1113–1139, December 2004.

[12] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, Jan. 1965.

[13] Commonwealth of Massachusetts Office of Geographic and Environmental Information. MassGIS planning roads datalayer description. http://www.mass.gov/mgis/eotroads.htm, 2008.

[14] Commonwealth of Massachusetts Office of Geographic and Environmental Information. MassGIS USGS color ortho imagery. http://www.mass.gov/mgis/colororthos2008.htm, 2008.

[15] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary Bradski. Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems*, Philadelphia, PA, USA, Aug. 2006.

[16] Ernst Dickmanns and Birger Mysliwetz. Recursive 3-D road and ego-state recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):199–213, Feb. 1992.

[17] Ernst Dickmanns and Alfred Zapp. Autonomous high speed road vehicle guidance by computer vision. In *Proc. 10th Triennial World Congress of the International Federation of Automatic Control*, Munich, Germany, 1987.

[18] Dmitri Dolgov and Sebastian Thrun. Autonomous driving in semi-structured environments: Mapping and planning. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3407–3414, May 2009.

[19] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

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

[21] Luke Fletcher, Nicholas Apostoloff, Lars Petersson, and Alexander Zelinsky. Vision in and out of vehicles. *IEEE Intelligent Systems*, 18(3):12–17, 2003.

[22] Luke Fletcher and Alexander Zelinsky. Context sensitive driver assistance based on gaze - road scene correlation. In *Int. Symposium on Experimental Robotics*, pages 287–296, Rio De Janeiro, Brazil, Jul. 2006.

[23] James D. Foley, Andries van Dam, Stephen K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1995.

[24] Raia Hadsell, J. Andrew Bagnell, Daniel Huber, and Martial Hebert. Accurate rough terrain estimation with space-carving kernels. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[25] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2001.

[26] Berthold K. P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4):441–460, 1983.

[27] Berthold K. P. Horn. *Robot Vision*. MIT Press, 1986.

[28] Albert S. Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. Multi-sensor lane finding in urban road networks. In *Proceedings of Robotics: Science and Systems*, Zürich, Switzerland, June 2008.

[29] Albert S. Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. Finding multiple lanes in urban road networks with vision and lidar. *Autonomous Robots*, 26(2-3):103–122, Apr. 2009.

[30] Albert S. Huang and Seth Teller. Lane boundary and curb estimation with lateral uncertainties. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, St. Louis, Missouri, Oct. 2009.

[31] Iteris. AutoVue. http://www.iteris.com.

[32] Seung Gweon Jeong, Chang Sup Kim, Dong Youp Lee, Sung Ki Ha, Dong Hwal Lee, Man Hyung Lee, and Hideki Hasimoto. Real-time lane detection for autonomous vehicle. In *Proc. Int. Symp. on Industrial Electronics*, pages 1466–1471, Pusan, Korea, 2001.

[33] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):3545, 1960.

[34] Dong-Joong Kang and Mun-Ho Jung. Road lane segmentation using dynamic programming for active safety vehicles. *Pattern Recogn. Lett.*, 24(16):3177–3185, 2003.

[35] Axel Kaske, Didier Wolf, and René Husson. Lane boundary detection using statistical criteria. In *International Conference on Quality Control by Artificial Vision*, pages 28–30, 1997.

[36] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Trans. Intelligent Transportation Systems*, 9(1):16–26, Mar. 2008.

[37] Karl Kluge and Charles Thorpe. The YARF system for vision-based road following. *Mathematical and Computer Modelling*, 22(4-7):213–233, 1995.

[38] Chris Kreucher and Shridhar Lakshmanan. LANA: A lane extraction algorithm that uses frequency domain features. *IEEE Transactions on Robotics and Automation*, 15(2):343 – 350, Apr. 1999.

[39] Chris Kreucher, Sridhar Lakshmanan, and Karl Kluge. A driver warning system based on the LOIS lane detection algorithm. In *The Proceedings of The IEEE International Conference on Intelligent Vehicles*, volume 1, pages 17 – 22, Oct. 1998.

[40] Tobias Lang, Christian Plagemann, and Wolfram Burgard. Adaptive non-stationary kernel regression for terrain modeling. In *Proceedings of Robotics: Science and Systems*, Atlanta, Georgia, USA, June 2007.

[41] Joon Woong Lee. A machine vision system for lane-departure detection. *Computer Vision and Image Understanding*, 86(1):52–78, 2002.

[42] Joon Woong Lee, Chang-Doo Kee, and Un Kun Yi. A new approach for lane departure identification. In *IEEE Intelligent Vehicles Symposium*, pages 100–105, Columbus, OH, June 2003.

[43] Joon Woong Lee and Un Kun Yi. A lane-departure identification based on LBPE, Hough transform, and linear regression. *Computer Vision and Image Understanding*, 99(3):359–383, 2005.

[44] Sukhan Lee and Woong Kwon. Robust lane keeping from novel sensor fusion. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 4, pages 3704–3709, 2001.

[45] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous vehicle. *Journal of Field Robotics*, 25(10):727–774, Oct 2008.

[46] Google Maps. http://maps.google.com.

[47] Yoshiteru Matsushita and Jun Miura. On-line road boundary modeling with multiple sensory features, flexible road model, and particle filter. In *Proc. European Conference on Mobile Robots*, Sep. 2009.

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

[49] Ted R. Miller. Benefit-cost analysis of lane marking. *Public Roads*, 56(4):153–163, Mar. 1993.

[50] Mobileye. EyeQ. http://www.mobileye.com.

[51] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, Aug. 2008.

[52] David Moore, Albert S. Huang, Matthew Walter, Edwin Olson, Luke Fletcher, John Leonard, and Seth Teller. Simultaneous local and global state estimation for robotic navigation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3794–3799, Kobe, Japan, May 2009.

[53] Sergiu Nedevschi, Rolf Schmidt, Thorsten Graf, Radu Danescu, Dan Frentiu, Tiberiu Marita, Florin Oniga, and Ciprian Pocol. 3D lane detection system based on stereovision. In *IEEE Trans. Intelligent Transportation Systems*, pages 161–166, Washington, D.C., Oct. 2004.

[54] José Neira and Juan D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robotics and Automation*, 17(6):890–897, Dec 2001.

[55] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Int. Conf. Computer Vision and Pattern Recognition*, June 2004.

[56] U.S. Department of Transportation, Federal Highway Administration, Office of Information Management. Highway statistics 2005. http://www.fhwa.dot.gov/policy/ohim/hs05.

[57] Edwin Olson. Real-time correlative scan matching. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 4387–4393, Kobe, Japan, June 2009.

[58] Edwin Olson, Matthew Walter, John Leonard, and Seth Teller. Single-cluster spectral graph partitioning for robotics applications. In *Proceedings of Robotics: Science and Systems*, pages 265–272, Cambridge, MA, June 2005.

[59] Christian Plagemann, Sebastian Mischke, Sam Prentice, Kristian Kersting, Nicholas Roy, and Wolfram Burgard. Learning predictive terrain models for legged robot locomotion. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, pages 3545–3552, Nice, France, 2008.

[60] Dean Pomerleau. Neural network vision for robot driving. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. 1995.

[61] Dean Pomerleau and Todd Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert: Special Issue on Intelligent Systems and their Applications*, 11(2):19–27, Apr. 1996. see also IEEE Intelligent Systems.

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

[63] Paul Schlyter. How to compute planetary positions. http://stjarnhimlen.se/comp/ppcomp.html, Aug. 2007.

[64] Stephan Sehestedt, Sarath Kodagoda, Alen Alempijevic, and Gamini Dissanayake. Robust lane detection in urban environments. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, San Diego, CA, USA, Oct 2007.

[65] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous robot vehicles*, pages 167–193, 1990.

[66] Ben Southall and Camillo J. Taylor. Stochastic road shape estimation. In *Proc. Int. Conference on Computer Vision*, volume 1, pages 205–212, 2001.

[67] Camillo J. Taylor, Jana Kosecká, Robert Blasi, and Jitendra Malik. A comparative study of vision-based lateral control strategies for autonomous highway driving. *Int. Journal of Robotics Research*, 18(5):442–453, 1999.

[68] Charles Thorpe, Martial Hebert, Takeo Kanade, and Steven Shafer. Vision and navigation for the Carnegie-Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, May 1988.

[69] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[70] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using B-Snake. *Image and Vision Computing*, 22(4):269 – 280, 2004.