BELIEF SPACE-GUIDED NAVIGATION FOR ROBOTS AND AUTONOMOUS VEHICLES

A Dissertation

by

BINBIN LI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Dezhen Song
Committee Members,	Anxiao(Andrew) Jiang
	Dylan A. Shell
	Suman Chakravorty
Head of Department,	Scott Schaefer

May 2021

Major Subject: Computer Engineering

Copyright 2021 Binbin Li

ABSTRACT

Navigating through the environment is a fundamental capability for mobile robots, which is still very challenging today. Most robotic applications these days, such as mining, disaster response, and agriculture, require the robots to move and perform tasks in a variety of environments which are stochastic and sometimes even unpredictable. A robot often cannot directly observe its current state but instead estimates a distribution over the set of possible states based on sensor measurements that are both noisy and partial. The actual robot position differs from its prediction after applying a motion command, due to actuation noise. Classic algorithms for navigation should adapt themselves to where the behavior of the environment is stochastic, and the execution of the motions has great uncertainty.

To solve such challenging problems, we propose to guide the robot's navigation in the belief space. Belief space-guided navigation differs fundamentally from planning without uncertainty where the state of the robot is always assumed to be known precisely. The robot senses its environment, estimates its current state due to perception uncertainty, and decides whether a new (or priori) action is appropriate. Based on that determination, it actuates its sensors to move with motion uncertainty in the environment. This inspires us to connect robot perception and motion planning, and reason about the uncertainty to improve the quality of plan so that the robot can follow a collision-free, feasible kinodynamic, and task-optimal trajectory.

In this dissertation, we explore the belief space-guided robotic navigation problems, which include belief space-based scene understanding for autonomous vehicles, and introduce belief space guided robotic planning.

We first investigate how belief space can facilitate scene understanding under the con-

text of lane marking quality assessment in the application of autonomous driving. We propose a new problem by measuring the quality of roads and ensuring they are ready for autonomous driving. We focus on developing three quality metrics for lane markings (LMs), correctness metric, shape metric, and visibility metric, and algorithms to assess LM qualities to facilitate scene understanding.

As another example of using belief space for better scene understanding, we utilize crowdsourced images from multiple vehicles to help verify LMs for high-definition (HD) map maintenance. An LM is consistent if belief functions from the map and the image satisfy statistical hypothesis testing. We further extend the Bayesian belief model into a sequential belief update using crowdsourced images. LMs with a higher probability of existence are kept in the HD map whereas those with a lower probability of existence are removed from the HD map.

Belief space can also help us to tightly connect perception and motion planning. As an example, we develop a motion planning strategy for autonomous vehicles. Named as virtual lane boundary approach, this framework considers obstacle avoidance, trajectory smoothness (to satisfy vehicle kinodynamic constraints), trajectory continuity (to avoid sudden movements), global positioning system (GPS) following quality (to execute the global plan), and lane following or partial direction following (to meet human expectation). Consequently, vehicle motion is more human-compatible than existing approaches.

As another example of how belief space can help guide robots for different tasks, we propose to use it for the probabilistic boundary coverage of unknown target fields (UTFs). We employ Gaussian processes as a local belief function to approximate a field boundary distribution in an ellipse-shaped local region. The local belief function allows us to predict UTF boundary trends and establish an adjacent ellipse for further exploration. The process is governed by a depth-first search process until UTF is approximately enclosed by connected ellipses when the boundary coverage process ends. We formally prove that our

boundary coverage process guarantees the enclosure above a given coverage ratio with a preset probability threshold.

ACKNOWLEDGMENTS

I would like to thank my advisors, colleagues, friends, and family who have helped me throughout my Ph.D. study.

I would like to thank my committee chair, Dr. Dezhen Song, for his support and guidance with my research. Many thanks to Dr. Haifeng Li, Dr. Hongpeng Wang, and Dr. Baifan Chen for being my great research collaborators and friends. I also want to thank my lab mates/alumni Dr. Joshph Lee, Dr. Chieh Chou, Hsin-min Cheng, Shu-Hao Yeh, Aaron Kingery, Aaron Angert, and Di Wang for their support of my research projects at A&M.

Sincere thanks to my advisory committee members Dr. Anxiao Jiang, Dr. Dylan Shell, and Dr. Suman Chakravorty for their dedicated insights, valuable suggestions, and exciting discussions.

Last but not least, special thanks to my family and friends for accompanying me through my toughest time.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by my committee consisting of Dr. Dezhen Song, Dr. Anxiao Jiang and Dr. Dylan Shell of the Department of Computer Science and Engineering, and Dr. Suman Chakravorty of the Department of Aerospace Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by the graduate research assistantship and teaching assistantship from Texas A&M University.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Navigating through a environment is the most fundamental capability for mobile robots, which is still very challenging nowadays. A mobile robot is required to perform inference from sensor measurements, to build a model of the surrounding environment, and to estimate variables of interest. Moreover, it has to plan actions to accomplish given goals. Most robotic applications these days, such as mining, disaster response, and agriculture, require the robots to move and perform tasks in a variety of environments which are stochastic and sometimes even unpredictable. A robot often cannot directly observe its current state but instead estimates a distribution over the set of possible states based on sensor measurements that are both noisy and partial. Besides, the actual robot position differs from the prediction after applying a motion command due to actuation noise. Classic algorithms for navigation should adapt themselves to where the behavior of the environment is stochastic, and the execution of the motions has great uncertainty.

To solve such challenging problems, we propose to guide the robot's navigation in the belief space. Belief space-guided navigation differs fundamentally from planning without uncertainty where the state of the robot is always assumed to be known precisely. The robot cannot directly observe its state but can only infer it from past observations and actions. This leads to the necessity of maintaining the space of all possible distributions over the state space called belief space and computing a control policy to select the best plan. However, belief space-guided navigation is still very challenging for several reasons. The robot must produce plans that reliably achieve its tasks despite the difference between the model (both perception and motion) and the real world. Meanwhile, the robot has to compensate for model uncertainties, unknown external disturbances, and time-varying key parameters. Explicitly considering motion and sensing uncertainty when computing robot

motions can improve the quality of computed plans.

Robots employ different sensors to “see", “touch", and “sense" the surrounding environment. For navigation alone, cameras, light detection and ranging (LIDAR) sensors, radar sensors, ultrasonic sensors, and infrared sensors are widely used. Cameras are not only inexpensive, but also able to capture texture, color and contrast information, and high level of details about the world. LIDAR sensors can scan more than 100 meters in all directions, generating a precise 3D map of its surroundings. Radar uses radio waves to determine the velocity, range, and angle of objects, which can work in every condition and even use reflection to “see" objects behind obstacles. Ultrasonic sensors measure distance by sending out a sound wave at a specific frequency and wait for that sound wave to bounce back. Infrared sensors are capable of measuring the heat being emitted by an object and detecting motion by either emitting and/or detecting infrared radiation. Through sensory systems, mobile robots acquire information about their surrounding environment to obtain the belief space, the space of possible values that the robot state can take, and generate feasible robot motions to avoid obstacles and maneuver to targets. However, the sensor readings often contain great uncertainties due to sensing conditions, physical properties, environmental variations, and resolution limitations. Furthermore, sensors have limited sensing ranges. For instance, digital camera image noise is a random variation of brightness or color information and produced by the sensitivity setting in the camera, length of the exposure, and temperature [1, 2]. The depth noise of a Kinect is a quadratic function of the depth [3]. The depth uncertainty of LIDAR has a linear relationship with the measured depth [4]. All of these cause uncertainty of robot’s state and limit a robotic systems’ ability to provide efficient and precise understandings of its environment to ensure proper planning and action of the robot. Ignoring such uncertainty is unwise and may lead to incorrect decision-making for the robots.

To deal with the challenges, the robot is expected to utilize knowledge from the be-

lief space during the navigation by considering perception uncertainty, originally from the sensing system, and motion uncertainly carefully in order to make correct decisions. The planner should not merely compute a static path but rather a collision-free, feasible kinodynamic, and task-optimal trajectory that allows the robot motion to perform given the current state information. The optimal plan will maximize, for instance, the probability of reaching a specified goal location while avoiding collisions with obstacles. These inspire us to investigate how to guide the robot navigation in the belief space.

In this dissertation, we explore belief space-guided robotic navigation problems, which include belief space-based scene understanding for autonomous vehicles, and introduce belief space guided robotic planning.

We first investigate how belief space can facilitate scene understanding under the context of lane marking quality assessment in the application of autonomous driving. We focus on developing metrics and algorithms to assess LM qualities from an egocentric view of an inspection vehicle equipped with a global positioning system (GPS) receiver, a frontal-view camera, and a LIDAR system. We propose three quality metrics for LMs: correctness metric, shape metric, and visibility metric. The correctness metric measures the divergence between the expected LMs based on prior map inputs and the actual sensor inputs. The shape metric evaluates smoothness in road curvature and width range. The visibility metric evaluates the contrast between LMs and background road surfaces. We propose a dual-modal algorithm to compute these metrics. We have implemented the algorithms and tested them under open dataset. The results show that our metrics can successfully detect LM anomalies in all testing scenarios.

As another example of using belief space for better scene understanding, we utilize crowdsourced images from multiple vehicles to help verify LMs for high-definition (HD) map maintenance. We obtain the LM distribution in the image space by considering the camera pose uncertainty in perspective projection. Both LMs in the HD map and LMs in

the image are treated as observations of LM distributions which allow us to construct posterior conditional distribution (a.k.a Bayesian belief functions) of LMs from either source. An LM is consistent if belief functions from the map and the image satisfy statistical hypothesis testing. We further extend the Bayesian belief model into a sequential belief update using crowdsourced images. LMs with a higher probability of existence are kept in the HD map whereas those with a lower probability of existence are removed from the HD map. We verify our approach using real data. Experimental results show that our method is capable of verifying and updating LMs in the HD map.

Belief space can also help us to tightly connect perception and motion planning. As an example, we develop a motion planning strategy for autonomous vehicles. Existing autonomous vehicle (AV) navigation algorithms treat lane recognition, obstacle avoidance, local path planning, and lane following as separate functional modules which result in driving behavior that is incompatible with human drivers. It is imperative to design human-compatible navigation algorithms to ensure transportation safety. We develop a new perception-planning framework that combines all these functionalities to ensure human-compatibility. Using GPS-camera-LIDAR sensor fusion, we detect actual lane boundaries (ALBs) and propose availability-reasonability-feasibility (ARF) threefold tests to determine if we should generate virtual lane boundaries (VLBs) or follow ALBs. If needed, VLBs are generated using a dynamically adjustable multi-objective optimization framework that considers obstacle avoidance, trajectory smoothness (to satisfy vehicle kinodynamic constraints), trajectory continuity (to avoid sudden movements), GPS following quality (to execute the global plan), and lane following or partial direction following (to meeting human expectation). Consequently, vehicle motion is more human-compatible than existing approaches. We have implemented our algorithm and tested under open-source data with satisfying results.

As another example of how belief space can help guide robots for different tasks, we

propose to use it for probabilistic boundary coverage of unknown target fields (UTFs). The robot accumulates sufficient sensory readings at each step to instantiate Gaussian processes (GPs) as a local belief function to approximate field dispersion in an ellipse-shaped local region, which allows us to predict UTF boundary trends and establish adjacent ellipses for further exploration in the next step. The overall process is governed by a depth-first search process until the UTF is enclosed by fully connected ellipses. We prove that our boundary coverage process can guarantee that the enclosure of UTF is above a given coverage ratio with a preset probability threshold. We have implemented our method and tested with different types of UTFs in simulation. The results show that the proposed algorithm always guarantees that the coverage ratio is above the given threshold for all testing cases (1D vs. 2D, smooth vs. non-smooth boundary, and convex vs. non-convex).

The rest of this dissertation is organized as follows. Section 2 reviews literature related to this dissertation. Section 3 presents the belief space-based scene understanding for a single robot, which we want to quantify the LM quality for autonomous driving. In Section 4, we demonstrate the belief space-based scene understanding for multiple robots by reporting a LM verification approach through crowdsourced images. Section 5 presents our multi-module VLB generation methods that tightly connect the perception with motion planning. In Section 6, we present the boundary coverage for UTFs. Section 7 concludes the dissertation and discusses future work directions.

2. RELATED WORK

Our work is related to belief space representation, uncertainty model for robotic navigation, and robotic navigation in belief space.

Adoption of appropriate belief space for robot navigation not only makes robotic problems computationally feasible, but also provides robustness in the presence of uncertainty. Murray et al. [5] constructs robot-specific circuitry for motion planning, capable of generating motion plans approximately three orders of magnitude faster than existing methods. Pivtoraiko et al. [6] propose deterministic search in a specially discretized state space, and compute a set of elementary motions that connects each discrete state value to a set of its reachable neighbors via feasible motions. Pivtoraiko et al. [7] compute a control set which connects any state to its reachable neighbors in a limited neighborhood. Equivalence classes of paths are used to implement a path sampling policy which preserves expressiveness while eliminating redundancy. Howard et al. [8] presents an effective algorithm for belief space sampling using a model-based planning algorithm that enables high-speed navigation for robot. Klanvar et al. [9] use linearized tracking-error dynamics to predict future system behavior and derive a control law from a quadratic cost function penalizing the system tracking error and control effort. Bouton et al. [10] provide an efficient strategy to navigate through urban intersections is a difficult task, which help navigate the robot through unsignalized intersections as a partially observable Markov decision process and solves it using a Monte Carlo sampling method. Patil et al. [11] address the problem of incorporating sensing discontinuities due to factors such as limited field of view of sensors and occlusions, in an optimization-based framework for belief space planning. Sadigh et al. [12] embrace the fact that robot actions affect what humans do, leverage it to improve state estimation, and enable robots to do active information gath-

ering, by planning actions that probe the user in order to clarify their internal state. Note that the robot state typically only has six degrees of freedom (three for rotation and three for translation) in our problems. Instead of applying configuration space that performs well in high-dimensional space, we fully utilize the belief space to guide the navigation for the robot, and select the best route considering perception and motion uncertainty by minimizing a customized cost to navigate the robot.

Different methods have been proposed to estimate the uncertainty for navigation tasks. Markku et al. [13] devise a Poisson-Gaussian noise estimation method for images combining variance-stabilization and noise estimation for additive Gaussian noise. Liu et al. [14] estimate an upper bound noise level based on a piece-wise smooth image prior model and measured CCD camera response functions. Michal et al. [15] quantify camera uncertainty by computing the inversion instead of the pseudo inversion of the information matrix, which allows the scaling of the values of the information matrix and produces more precise results. Kovalev et al. [16] analyze the uncertainty of LIDAR data to provide accurate measurements. Matthies et al. [17] use occupancy grid to integrate noisy range data from multiple sensors and multiple robot positions into a common description of the environment. Hanheide et al. [18] handle uncertain and incomplete information from the sensors for robot task planning and explanation through three layers from the bottom level to the top. Fabian et al. [19] improve the visual odometry performance through the analysis of the sensor noise and the propagation of error through the entire visual odometry system. Van et al. [20] propose linear-quadratic Gaussian motion planning, which is based on the linear-quadratic controller with Gaussian models of uncertainty, and explicitly characterizes in advance (i.e. before execution) the a priori probability distributions of the state of the robot along its path, to design a feasible trajectory after assessing the quality of the path, for instance by computing the probability of avoiding collisions. Nardi et al. [21] propose the use of an uncertainty-augmented Markov Decision Process to approximate the

underlying Partially Observable Markov Decision Process, and employ a localization prior to estimate how the belief about the robot's position propagates through the environment. Gonzalez et al. [22] uses a path planner that calculates optimal paths while considering uncertainty in position and that uses landmarks to localize the vehicle as part of the planning process. However, existing approaches try to reduce the sensor uncertainty through different probability models instead of proposing efficient approaches to help guide the navigation in belief space for mobile robots.

We review three different categories for navigation in belief space to reduce uncertainty in robot: *reactive planning-based, sampling-based and optimization-based*. *Reactive planning-based method* considers the obstacle and target goal position into the planning pipeline. Koren et al. [23] propose iterative Gauss-Seidel method on discretized cell grid to generate a single minimum globally at the location of the goal configuration. Lengyel et al [24] describe a navigation function for each free configuration sample by guaranteeing the global minimum at the goal configuration. Khatib et al. [25], Krogh et al [26] and Hwang et al. [27] use potential field (PF) path planning to calculate force fields generated by the goal target and the surrounding obstacles. A further extension of the original PF is made by Huang et al. [28] for non-holonomic robots. They utilize robot's headings to the obstacles and the goal to track the path through the potential fields. Thought reactive planning methods are successfully to generate feasible path, it is hard to account for predicting the involvement of the environment in order to yield a deliberative plan that has certain optimality sense overall a long spatial or temporal planning horizon. *Sampling-based method* aims to discretize the configuration space and convert the motion planning problem into a generate-and-evaluate problem or a graph search problem, which has been widely used [29, 30]. Fox et al. [31] define the dynamic window method to generate sampled trajectories for the robots to select. Nagy et al. [32] generate cubic spirals by considering a kinematically feasible control trajectory that connects the start position to

the goal. Kavraki et al. [33] propose the probabilistic road-map for path planning for path planning problem, in which a learning phase explores the configuration space by growing the tree of randomly sampled points and a query phase traverse the graph to find a feasible route. LaValle et al. [34] propose the Rapidly-exploration Random Tree, which can be viewed as a single query version of PRM. Variation of PRM and RRT have been widely used for mobile robots [35, 36, 37, 38]. Pivtoraiko et al. [6] propose to use state lattice, which combine a discrete searching graph with kinodynamic constraints, which has been applied in structured [39] and unstructured environment [40]. In terms of graph construction and searching of the optimal path for motion planning problem, algorithm like Dijkstra's algorithm [41], A^* [42], and D^* [43] are developed to find the optimal but exploring the graph less. Overall, the sampling-based planning methods provide a systematic approach for converting the continuous workspace into a discrete workspace modeled by a graph in order to find a probabilistically complete or resolution-optimal planning solution. The lattice-based approaches adapted for both unstructured and structured environments can easily integrate a non-holonomic vehicle's kinematic constraints. On the other hand, the downsides are equally obvious: the curse of dimensionality, the sub-optimality introduced due to discretization, and the potentially intractable computational overhead. *Optimization-based method* iteratively refines a solution until the termination/convergence conditions are met while taking into account the robot's dynamic model. Depending on whether the optimization routine is applied to states or control, there are two classes of trajectory optimization methods – direct and indirect. The direct methods enforce model feasibility between states numerically by altering the state-control-state sequence along the trajectory. Thrun et al. [44] propose a gradient descent optimizer to nudge teach sampled point by minimizing the cumulative path jerk. Brandt et al. [45] graduate deforming a path by achieving an equilibrium according to the artificial forces. Roesmann et al. [46] consider the temporal aspects and dynamic constraints of the robot motion by augmenting the

time interval information. The indirect methods utilize the initial state to manipulate the controls and the states along the robot trajectory, which must be obtained through forward shooting (a.k.a, integration, evaluation, pass) using the dynamics model. Jaacobson et al. [47] propose a second-order method (Differential Dynamic Programming) to improves the robot path by the manipulating the control sequence locally. Berg et al. [48] exploit the balance between the optimal control and the estimation to smooth the path using a extended Linear-quadratic regulator algorithm. Though optimization-based approaches are efficient to deal with states or control, it is easier for the algorithm to be trapped into the local minimal without careful tuning. In this dissertation, we carefully design our approaches that consider planning for robots in belief space rather than treating the robotic motion as individual problems.

3. BELIEF SPACE-BASED SCENE UNDERSTANDING FOR LANE MARKING QUALITY ASSESSMENT[1]

We first investigate how belief space can facilitate scene understanding under the context of quantifying the lane marking quality in the application of autonomous vehicle. As AVs are getting closer and closer to our life, one critical question remains unanswered: are our roads ready for AVs? AV developers attempt to deal with all kinds of road conditions. However, safety can be challenged when poor road conditions (see Fig. 3.1) appear because there are limited training data for abnormalities. Due to the limited sensory capabilities and on-board computation resources of AVs, exhaustively predicting road scenarios is infeasible. A solution to reliable autonomous driving is to ensure our infrastructure is ready for the technology.

Here we present a lane marking quality assessment (LMQA) method to help road inspection crews examine the quality of lane markings. The method assumes an egocentric view with a GPS receiver, a frontal-view camera, and a LIDAR. Based on data from GPS, prior maps from geographic information systems (GIS) and on-board sensors, we propose three different lane quality metrics: *correctness*, *shape*, and *visibility*. The correctness metric measures the divergence between the expected lane markings based on prior map inputs and the actual sensor inputs. Building on the difference between posterior distributions, it takes uncertainties in inputs into consideration. The shape metric verifies if the lane has smooth curvature according to road grade and is within satisfying width range. The visibility metric evaluates the contrast between lane markings and background road surfaces. Fusing camera images and LIDAR data, we propose an algorithm to compute

[1]Reprinted with permission from "Lane Marking Quality Assessment for Autonomous Driving" by B.Li, D. Song, H. Li, A. Pike and P. Carson, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, October, 1-5, 2018.

these metrics. The algorithms extensively utilize both camera images and LIDAR data in road surface detection and lane marking recognition to generate segmented left and right lane marking points required by the three metrics computation. We have implemented the algorithms and tested it using the KITTI dataset. The results show that our metrics successfully recognize anomalies in lane markings.

It is worth noting that LMQA is NOT the same as the well-known lane marking detection problem. It is not in our best interest to develop/apply the most sensitive and accurate lane detection algorithm in LMQA because we want to ensure that our roads are safe for less capable vehicles. *Here we measure roads instead of vehicles*. LMQA focuses on evaluating lane marking qualities instead of abilities to detect them. It needs to be able to mark low quality road segments and output different types of quality issues instead of just reporting "no lane" detected. It also takes into consideration of common sensor configurations for all AVs instead of optimizing lane detection for a particular sensor configurations.

3.1 Related Work

The second part of our work is relevant to road quality assessment, road surface extraction, and lane detection in transportation and autonomous driving research. In this section, we develop a review of related literature as follows.

Lane markings play an important role in autonomous driving. To the best of our knowledge, little has been done to quantify lane markings to assist road maintenance. Harwood et al. [49] use operational analysis procedures to assess the capacity and level of service of two-lane highways. Flannery et al. [50] quantify the road service quality by comparing drivers' assessments of the performance of urban streets with objective measures of performance. Thomas et al. [51] propose a systematic approach to evaluate algorithms for extracting road marking features. Pohl et al. [52] estimate driver's visual distraction level to provide sufficient reliability of lane-keeping. However, none of them focus on

(a) New lane markings coexists with the old one.

(b) Lane border shape may not satisfy the standard.

(c) Faded or blurred lane markings.

Figure 3.1: Examples of urban road scenario.

measuring lane marking quality itself.

During road surface extraction, color and texture are the main perceptual cues for navigation systems of semi- or fully autonomous vehicles. Besides obstacle avoidance, road surface detection facilitates path planning and decision making. Common sensory methods include cameras [53] and LIDARs. Pradeep et al. [54] use stereo camera data to extract the road surface structure. Hernández et al. [55] filter and segment the road surface from 3D point clouds acquired through mobile LIDAR systems. Li et al. [56] utilize the structural information to find the road surface by combining multiple task deep convolutional neural networks with a recurrent neural network detector. Yu et al. [57] extract road surface points directly from three dimensional point clouds. Guan et al. [58] extract road surface through a curb-based method using geo-referenced intensity images.

More detailed surveys can be found in [59, 60]. Most existing efforts only utilize a single sensing modality, as we simultaneously employ both camera images and LIDAR data to extract the road surface in our approach for more robustness.

A lane border detection system detects lane markings from complex environments. Lane markings are important for reliable estimation of vehicle positions relative to lanes. Different sensors or perception modalities have been used for lane border detection, such as monocular vision [59, 61], LIDAR [62, 63, 64], stereo imaging [54, 65], GPS and inertial measurement unit (IMU) [30, 66], and Radar [67]. Gu et al. [68] classify the lane markings by fusing images and LIDAR scans using convolutional neural networks. Huang et al. [69] describe and detect multiple lane borders in an urban road network from calibrated video imagery and laser range data acquired by a moving vehicle. Mammeri et al. [70] combine the Maximally Stable Extremal Region technique with the Hough Transform to detect and recognize lane markings. Various lane border detection systems have been proposed in the automotive industry [71, 72]. Built on existing efforts, our dual-modal lane marking detection leverages inputs from both camera images and LIDAR scans to facilitate lane marking quality metric computation with an attempt to provide a baseline performance under common sensor configurations.

3.2 Problem Formulation

3.2.1 Inputs, Assumptions, and Notations

Our objective is to quantify the lane marking quality from an egocentric view. The inspection vehicle is equipped with a frontal view camera and a LIDAR, and we use their data as problem input. We also employ GPS data and prior maps consisting of a set of 3D lane boundaries, such as Google Maps, as part of inputs. We only evaluate immediate left and right lane markings with respect to the vehicle due to their importance in guiding the vehicle. We do not evaluate multiple parallel lanes simultaneously because the sensors

on-board the vehicle have perspective limitations. We have the following assumptions.

a.1 The camera is pre-calibrated and we know its intrinsic parameters. The nonlinear distortion of camera images has been removed.

a.2 The GPS, the camera, and the LIDAR readings are already synchronized.

a.3 The coordinate system transformations between any two sensors are known by calibration.

Common notations are defined as follows,

- $\mathbf{P}_{i,t} \in \mathbb{R}^3$, the i-th 3D LIDAR point in the LIDAR reference system at time t. Also, it is defined in a system with x-axis pointing to vehicle forward direction, y-axis pointing to the left of the vehicle lateral direction, and z-axis pointing upward.
- I_i, the intensity value of the LIDAR point $\mathbf{P}_{i,t}$.
- $\mathscr{P}_t := \{\mathbf{P}_{i,t}\}$, LIDAR point cloud data set at time t.
- $\mathbf{I}_t$, the gray-scale camera image at time t.
- $\mathbf{p}_{k,t} = [u\ v]^\intercal \in \mathbb{R}^2$, the k-th pixel point in image $\mathbf{I}_t$ where (u,v) is the image coordinate.
- $\tilde{\mathbf{X}}$, homogeneous vector $\tilde{\mathbf{X}} = [\mathbf{X}^\intercal, 1]^\intercal$ where $\mathbf{X}^\intercal$ denotes the inhomogeneous part of $\tilde{\mathbf{X}}$.

3.2.2 Quality Metrics and Problem Definition

We introduce three types of quality metrics for LMQA. Here we just define them. We will model them mathematically in Section 3.3.

- *Correctness Metric:* Defined as μ_c, this metric quantifies the divergence between sensed lane marking positions and that from the prior map in the GIS system. It may be caused by slow updates of GIS database when road construction or maintenance changes lane markings. Lane markings may disappear completely or be painted incorrectly due to poor maintenance. All of these cause discrepancies between prior maps and sensory inputs which introduces difficulty for AVs to make decisions.

- *Shape Metric:* Defined as μ_s, this metric measures whether the road segment is smooth in curvature which is defined by road grade and conformity to lane width standard. A smooth road with proper width makes it easier for AVs to perform trajectory following and leads to smooth and safe rides. Lane width may differ according to different road grades and countries but is always bounded between a lower bound and an upper bound. Sometimes, the desired width may be a single fixed value. For example, the US Interstate Highway standard dictates 3.7 meters lane width.

- *Visibility Metric:* Defined as μ_v, the metric measures how visible lane markings compare to background surface in both images and LIDAR data. High contrast makes lane markings easy to be detected and segmented by AVs.

With assumptions, notations and quality metrics made, our problem can be defined as follows,

Problem 1. *Given the GPS coordinate, a prior map, the LIDAR point cloud $\mathscr{P}_t$, and camera image $\mathbf{I}_t$, quantify the LMQA according to the aforementioned quality metrics.*

3.3 Metric Modeling

Now let us model the three metrics mathematically. Note that we sample data periodically. At discrete time t, we have a camera image $\mathbf{I}_t$ and LIDAR point cloud $\mathscr{P}_t$. To avoid

too much overlap with previous or following iterations, we only use a partial set of points

$$\bar{\mathscr{P}}_t := \{\mathbf{P}_{i,t} | \|\mathbf{P}_{i,t}\| \leq \zeta \cdot v\tau, \mathbf{P}_{i,t} \in \mathscr{P}_t\}, \tag{3.1}$$

where $\|\cdot\|$ is the vector l^2-norm, τ is the sampling interval, v is current vehicle velocity, and ζ is a positive constant controlling the overlap between $\bar{\mathscr{P}}_t$ and $\bar{\mathscr{P}}_{t+1}$. To ensure full coverage, we set $\zeta = 2$. Since we know the relationship between the image coordinate and LIDAR coordinate, we also use $\bar{\mathscr{P}}_t$ to obtain the corresponding $\bar{\mathbf{I}}_t$ in $\mathbf{I}_t$. Also, given the GPS coordinate, we know the prior map region overlaps with $\bar{\mathscr{P}}_t$. Define $\mathbf{X}_p \in \mathbb{R}^3$ as the corresponding lane marking from the prior map in this overlapping region and set $\mathscr{P}_p := \{\mathbf{X}_p\}$ for all $\mathbf{X}_p$'s. All metrics below are based on $\mathscr{P}_p$, $\bar{\mathscr{P}}_t$, and $\bar{\mathbf{I}}_t$.

3.3.1 Correctness Metric

We define the correctness metric in $\bar{\mathbf{I}}_t$. Let C_k represent an event that a pixel $\mathbf{p}_{k,t}$ in image $\bar{\mathbf{I}}_t$ is a lane marking pixel,

$$C_k = \begin{cases} 1, & \mathbf{p}_{k,t} \text{ is a lane marking point} \\ 0, & \text{otherwise.} \end{cases}$$

Define $P(C_k)$ as the probability for event C_k. Define prior map lane pixel $\mathbf{x}_p$ as the projection of $\mathbf{X}_p$ from the prior map into $\bar{\mathbf{I}}_t$. Define $\mathbf{K}$ as the intrinsic camera parameters and $\{\mathbf{R}, \mathbf{t}\}$ as the extrinsic parameters between the camera and LIDAR, where $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector that relate the laser coordinate system to the camera coordinate system. The projection is based on the perspective projection model,

$$\tilde{\mathbf{x}}_p = \mathbf{K}[\mathbf{R}\ \mathbf{t}]\tilde{\mathbf{X}}_p. \tag{3.2}$$

Define $\mathscr{S}_P = \{\mathbf{x}_p\}$ to be the set that covers all points in $\mathscr{P}_p$. Define set $\mathscr{S}_Q$ that con-

tains all the lane marking pixels in $\bar{\mathbf{I}}_t$. Define posterior probability $P(C_k|\mathscr{S}_P)$ to capture the lane marking distribution in the image space through a prior map. It is not deterministic because we have uncertainties in the map due to resolution limitations and errors in GPS coordinates. Similarly, we define posterior probability $P(C_k|\mathscr{S}_Q)$ to be the lane marking distribution given the sensory inputs. It is probabilistic due to sensory uncertainty. Then the correctness metric is modeled as the difference between these two conditional probabilistic distributions. We employ the smoothed Kullback-Leibler (KL) divergence to characterize this difference,

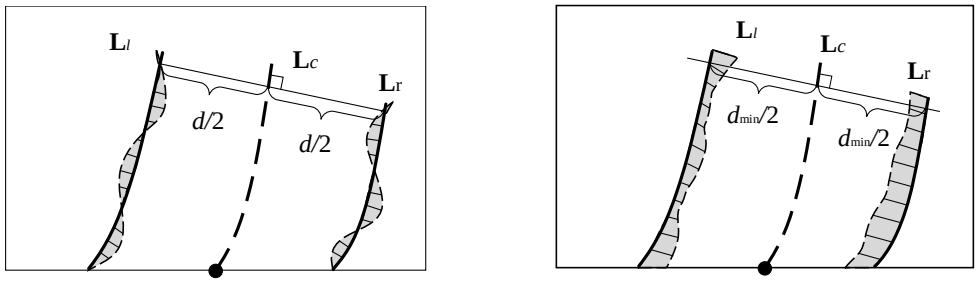
$$\mu_c = \sum_k P(C_k|\mathscr{S}_P) \log \frac{P(C_k|\mathscr{S}_P)}{P(C_k|\mathscr{S}_Q)}, \tag{3.3}$$

A KL divergence of 0 indicates that we have two identical distributions, while a KL divergence of 1 indicates that the two distributions are totally different. Therefore, we prefer small values in this metric.

Now let us explain how to compute $P(C_k|\mathscr{S}_P)$ and $P(C_k|\mathscr{S}_Q)$. Recall that $\mathbf{X}_p \in \mathscr{P}_p$ represents a lane marking point in the prior map and $\mathbf{x}_p \in \mathscr{S}_P$ is its projection using (4.2). We use the set $\{(\mathbf{x}_p, C_p)\}$ as the training set to instantiate a recursive Bayesian estimation process to obtain the lane marking distribution $P(C_k|\mathscr{S}_P)$ for the lane information on the prior map. It can be computed by using a two-phase approach. For an $\mathbf{x}_p^* \in \mathscr{S}_P$ and its corresponding event C_p^*, we update the probability distribution,

$$P(C_q^*|\mathscr{S}_P) = \overline{bel}(C_q^*|f(\mathbf{x}_q^*), \sigma^2, \mathbf{x}_q^*, \mathscr{S}_P), \tag{3.4}$$

where σ^2 is the variance of the noise, and the latent function f is represented by Gaussian Process (GPs) [73]. Here, σ^2 encodes the vehicle's current state, which is represented as high-dimensional Gaussian distribution. As the vehicle is moving along the lane, the value



(a) A cubic polynomial fitting smooths observation noises.

(b) An overly narrow lane results in large μ_s.

Figure 3.2: An illustration of how to compute lane center curve and the shape metric. Solid curves correspond to $\mathbf{L}_l$ and $\mathbf{L}_r$ which are determined by $\mathbf{L}_c$. Thin and dashed curves close to them represent points $\mathbf{X}_l$ and $\mathbf{X}_r$, respectively. The area of shaded region is μ_s.

of σ also changes accordingly. The GP provides posterior distribution of pixel $\mathbf{x}_k \in \bar{\mathbf{I}}_t$ for prediction,

$$P(C_k|\mathscr{S}_P) = \overline{bel}(C_k|\mu_I, \sigma_I^2, \mathbf{x}_k, \mathscr{S}_P). \tag{3.5}$$

Here, μ_I and σ_I^2 are the expectation and variance of the posterior distribution related to the kernel function, which characterizes the correlation between the function values at different pixels. Here we employ a Gaussian kernel K as

$$K(\mathbf{p}_{i,t}, \mathbf{p}_{j,t}) = \sigma_f^2 \exp(-\frac{1}{2\lambda_f^2}\|\mathbf{p}_{i,t} - \mathbf{p}_{j,t}\|^2), \tag{3.6}$$

for $\{\mathbf{p}_{i,t}, \mathbf{p}_{j,t}\} \subset \bar{\mathbf{I}}_t$ with $\mu_I = \mathbf{k}_*^\intercal(\mathbf{K}_o + \sigma^2\mathbf{I})^{-1}C_p^*$ and $\sigma_I^2 = \mathbf{k}_{**}\mathbf{k}_*^\intercal(\mathbf{K}_o + \sigma^2\mathbf{I})^{-1}\mathbf{k}_*$ where σ_f is the variance of the lane marking position, λ_f is the length scale variable, $\mathbf{k}_* = K(\mathscr{S}_P, \mathbf{x}_k)$, $\mathbf{K}_o$ is the kernel matrix of the training data $\mathscr{S}_P$, $\mathbf{k}_{**} = K(\mathbf{x_k}, \mathbf{x_k})$, and $\mathbf{I}$ is an identity matrix.

Similarly, we get $P(C_k|\mathscr{S}_Q)$ through (6.8)-(6.9). Therefore, we can obtain μ_c in (3.3) through the two posterior distributions $P(C_k|\mathscr{S}_Q)$ and $P(C_k|\mathscr{S}_P)$.

3.3.2 Shape Metric

The shape metric evaluates the lane shape defined by lane boundaries by examining its smoothness and width. To achieve this, we need to find the 'best shape' that fits the observation data. The best shape refers to a segment of lane that has smooth curvature defined by road grade and fits the width requirement. Then we evaluate how the existing lane marking point set compares to the best shape.

First, we need to model the best shape. We adopt a cubic polynomial lane center curve with a width to describe it because a cubic polynomial is sufficient to describe road curve and can be computed straightforwardly from cubic spline fitting. Define the lane center curve as a function of univariate parameter s,

$$\mathbf{L}_c(s) = \mathbf{a}_0 + \mathbf{a}_1 s + \mathbf{a}_2 s^2 + \mathbf{a}_3 s^3 \in \mathbb{R}^3, \tag{3.7}$$

where $\{\mathbf{a}_i | i = 0,1,2,3\}$ are 3-vectors for polynomial coefficients, and $\mathbf{p}_a = [\mathbf{a}_0^\intercal, \mathbf{a}_1^\intercal, \mathbf{a}_2^\intercal, \mathbf{a}_3^\intercal]^\intercal$. By forcing $\mathbf{L}_c(s) \subset \mathscr{P}_t \cup \mathscr{P}_{t-1}$, we obtain s range set $\mathbf{S}$:

$$\mathbf{S} := \{s | \mathbf{L}_c(s) \subset \mathscr{P}_t \cup \mathscr{P}_{t-1}\}.$$

The reason that we choose s's range, $\mathbf{S}$, to be much bigger than that of $\mathscr{P}_t$ is to ensure smoothness in future curve transition and full lane boundary coverage. For each point $\mathbf{X}_c \in \mathbf{L}_c(s)$ and a given width d, we find a point on the left boundary and a point on the right boundary by walking along the direction perpendicular to $\mathbf{L}_c(s)$ by $d/2$ to the left or right, respectively (see Fig. 3.2a). Therefore, lane boundaries $\mathbf{L}_l(s)$ and $\mathbf{L}_r(s)$ are determined by $\mathbf{L}_c(s)$ for the given width d.

Now let us explain how to obtain the lane center curve. Define $\mathbf{X}_l$ and $\mathbf{X}_r$ to be left and right lane marking points in $\bar{\mathscr{P}}_t$, respectively. For a given left boundary set $\mathbf{L}_l(s)$, we

evaluate each point in $\mathbf{X}_l$ by measuring the closest distance d_l:

$$d_l(\mathbf{X}_l, \mathbf{L}_l) = \min_{s \in \mathbf{S}} \|\mathbf{X}_l - \mathbf{L}_l(s)\| \approxeq \min_{s \in \mathbf{S}} \|\mathbf{X}_l - \mathbf{L}_c(s)\| - d/2. \tag{3.8}$$

The approximation works when the road is relatively flat. Similarly, we can evaluate each point in $\mathbf{X}_r$ by measuring the closest distance d_r:

$$d_r(\mathbf{X}_r, \mathbf{L}_l) = \min_{s \in \mathbf{S}} \|\mathbf{X}_r - \mathbf{L}_r(s)\| \approxeq \min_{s \in \mathbf{S}} \|\mathbf{X}_r - \mathbf{L}_c(s)\| - d/2.$$

It is clear that both left and right boundaries are based on the lane center curve which needs to be estimated with respect to inputs $\mathbf{X}_l$ and $\mathbf{X}_r$. We evaluate a given lane center curve using the observations $\mathbf{X}_l$ and $\mathbf{X}_r$ by the following objective function,

$$f_l(\mathbf{L}_c, d) = \frac{1}{n_l} \sum_{\mathbf{X}_l \in \bar{\mathscr{P}}_t} d_l(\mathbf{X}_l, \mathbf{L}_l) + \frac{1}{n_r} \sum_{\mathbf{X}_r \in \bar{\mathscr{P}}_t} d_r(\mathbf{X}_r, \mathbf{L}_r), \tag{3.9}$$

where n_l and n_r are numbers of lane marking points in the left and right lanes, respectively. Therefore, we can obtain the lane center curve and the optimal width by minimizing the

$$[\mathbf{p}_a^{*\mathsf{T}},\ d^*]^{\mathsf{T}} = \arg\min_{\mathbf{p}_a, d} f(\mathbf{L}_c, d), \tag{3.10}$$

subject to width constraint

$$d_{\min} \leq d \leq d_{\max}, \tag{3.11}$$

where $\mathbf{p}_a^*$ and d^* are optimal lane center curve parameters and the optimal lane width, and $d_{\min}$ and $d_{\max}$ are the minimum and maximum allowable width, respectively. The $\mathbf{p}_a^*$ defines the optimal center curve L_c^* according to (3.7), and can be further constrained to reflect desirable curvature range according to the road grade from GIS information so that

the optimization in (3.10) does not over fit the observations. With L_c^* and d^*, our shape metric is,

$$\mu_s = f(\mathbf{L}_c^*, d^*). \tag{3.12}$$

It is worth noting that μ_s characterizes both the smoothness and the width requirement. Preferably μ_s should be small. In fact, d_l and d_r are the distances from the estimated boundaries to their respective observations, which means the area of the shaded area in Fig. 3.2 is μ_s. It is clear that μ_s becomes large if the lane markings do not correspond to a smooth desirable curve. The same applies to the lane that is overly wide or narrow. In such cases, μ_s becomes excessively large as shown in Fig. 3.2b.

3.3.3 Visibility Metric

The visibility metric is defined based on both image pixels and LIDAR data. Define $\mu_{m,L}$ and $\mu_{b,L}$ as the mean intensity values for the lane marking points and the background points of the LIDAR scan, respectively. Recall that set $\mathscr{S}_Q$ contains lane marking pixels in image $\bar{\mathbf{I}}_t$, and define $\mathscr{S}_{b,I}$ to be the background pixel set. Define $\mu_{m,I}$ and $\mu_{b,I}$ to be the mean intensity values of the $\mathscr{S}_Q$ and $\mathscr{S}_{b,I}$ in $\bar{\mathbf{I}}_t$, respectively. The visibility metric is to verify intensity ratios in two modalities.

$$\mu_v = \max\left\{\frac{\mu_{m,L}}{\mu_{b,L}}, \frac{\mu_{m,I}}{\mu_{b,I}}\right\}. \tag{3.13}$$

It is clear that large values of μ_v are preferable. As long as the lane markings are visible in either modality, we treat them as satisfactory here. It is also possible to change max to min if we want to be more conservative.

3.4 Dual Modal Lane Detection Algorithm

To compute the aforementioned metrics, we need the segmented left and right lane markings in both camera image and LIDAR data. This means that we need a lane detection

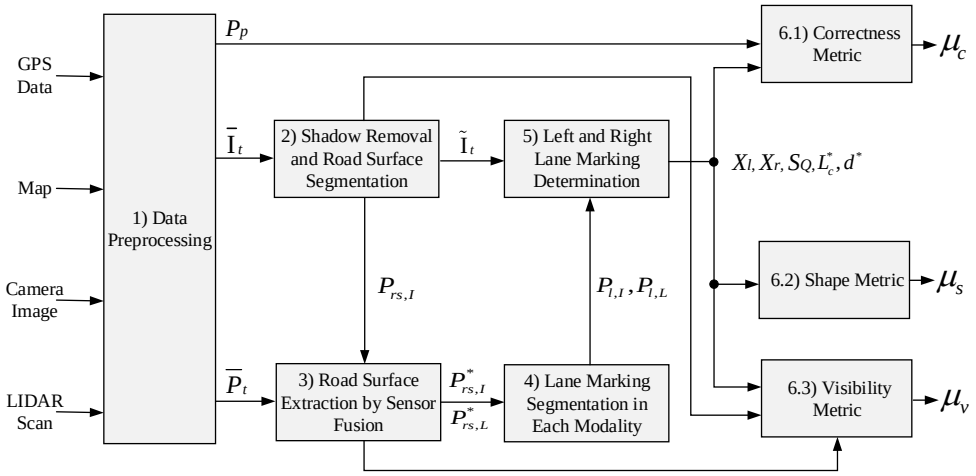


Figure 3.3: Algorithm diagram.

algorithm. However, it is important to build this algorithm using the most common sensor configurations without catering to a particular hardware choice. In fact, it is not in our best interest to use the best lane detection algorithm for LMQA purposes because we measure roads instead of vehicles. We need a baseline version of lane detection which can provide inputs required by our metrics. Unfortunately, existing commercial products only provide lane departure warnings instead of providing us with segmented pixels or coordinate. For completeness, we describe our lane detection algorithm here.

The overall sensor fusion pipeline is described in Fig. 5.2. In data preprocessing, we ensure all input data is synchronized. We then process camera images to remove shadows [74]. The shadow-free RGB image is recovered by relighting each pixel. For each image $\bar{\mathbf{I}}_t$, we extract the road surface by using fully convolutional networks [75, 76]. Note that we use full RGB colored images instead of gray-scale images in this step. Define $P_{rs,I} \subset \bar{\mathbf{I}}_t$ as road surface pixel set. Since noisy points are inevitable in $P_{rs,I}$, we need to fuse LIDAR data to reduce the influence of noise to refine the segmentation result.

3.4.1 Road Surface Extraction by Sensor Fusion

Define $\tilde{\mathbf{P}}_r = [x\ y\ z\ 1]^\intercal$ as the homogeneous 3D road surface point corresponding to the road surface image pixel $\mathbf{p}_r \in P_{rs,I}$. For each $\mathbf{p}_r$, we obtain the corresponding LIDAR point $\mathbf{P}_r$ using the inverse of (4.2),

$$\mathbf{P}_r = [\mathbf{KR}]^{-1}\tilde{\mathbf{p}}_r - \mathbf{R}^{-1}\mathbf{t}. \tag{3.14}$$

Assembling all $\mathbf{P}_r$, we obtain set $\mathscr{S}_r := \{\mathbf{P}_r\}$ containing the 3D LIDAR points belonging to road surface.

We model the road surface $S(x,y,z)$ using curved surface patches through bivariate polynomials [77],

$$z - \mathbf{H}_r \cdot \mathbf{z}_r = 0, \tag{3.15}$$

where $\mathbf{H}_r = [a_{00}\ a_{01}\ a_{11}\ a_{20}\ a_{21}\ a_{22}\ a_{30}\ a_{31}\ a_{32}\ a_{33}]^\intercal$ is the surface parameter vector that needs to be estimated, and $\mathbf{z}_r = [1\ x\ y\ x^2\ xy\ y^2\ x^3\ x^2y\ xy^2\ y^3]^\intercal$. We apply RANSAC [78] to filter out outliers and estimate the road surface model. A minimal solution can be established by randomly choosing 9 points from set $\mathscr{S}_r$ using a singular value decomposition (SVD) based algorithm. We set constraint $\|\mathbf{H}_r\|^2 = 1$ to avoid zero value solutions. In each iteration of RANSAC, we randomly select a minimal set of data from set $\mathscr{S}_r$ to estimate the $S(x,y,z)$. Denote $d_\perp(\mathbf{H}_r, \mathbf{P}_r)$ to be the shortest distance for a point $\mathbf{P}_r$ to the road surface, which is

$$d_\perp(\mathbf{H}_r, \mathbf{P}_r) = \min_{\mathbf{X}_r \in S} \|\mathbf{X}_r - \mathbf{P}_r\|, \tag{3.16}$$

subject to (3.15) for all $\mathbf{X}_r \in S(x,y,z)$, where $\mathbf{X}_r$ indicates a point on surface $S(x,y,z)$ that has the shortest distance to $\mathbf{P}_r$. By introducing the Lagrange multipliers λ, we find the

point $\mathbf{X}_r$ on the road surface $\mathscr{S}$ for each $\mathbf{P}_r$ through solving the Lagrange function,

$$f_r(\mathbf{X}_r, \mathbf{P}_r, \mathbf{H}_r) = \|\mathbf{X}_r - \mathbf{P}_r\|^2 + \lambda(z - \mathbf{H}_r \cdot \mathbf{z}_r). \tag{3.17}$$

We employ the distance measurement in (3.16) to determine inlier/outlier from the set $\mathscr{S}_r$. Define $\mathscr{A}$ as the inlier consensus set of road surface. We accept an inlier point set if the ratio between the set cardinality of $\mathscr{A}$ and the sample size is greater than threshold τ_t. $\tau_t = 0.6$ in all experiments. After obtaining the largest consensus set, we refine the $\mathbf{H}_r$ using all inliers by applying the maximum likelihood estimation (MLE) to minimize the sum of distance errors,

$$\hat{\mathbf{H}}_r = \arg\min_{\mathbf{H}_r} \sum_{\mathbf{P}_r \in \mathscr{A}} d_\perp(\mathbf{H}_r, \mathbf{P}_r)^2, \tag{3.18}$$

where $\hat{\mathbf{H}}_r$ denotes the estimation of $\mathbf{H}_r$ using the Levenberg-Marquardt (LM) algorithm.

After extracting road surface $S(x, y, z)$, we calculate the distance for all the LIDAR points in set $\{\overline{\mathscr{P}}_t \setminus \mathscr{A}\}$ using (3.16). We include the LIDAR points with distances to the surface less than threshold d_ε along with $\mathscr{A}$ itself,

$$P_{rs,L} = \left\{\mathbf{P}_{i,t} | d_\perp(\hat{\mathbf{H}}_r, \mathbf{P}_{i,t}) \leq d_\varepsilon, \mathbf{P}_{i,t} \in \overline{\mathscr{P}}_t, \mathbf{P}_{i,t} \notin \mathscr{A}\right\} \cup \mathscr{A}.$$

Define $\mathbf{p}_e$ to be the corresponding image pixel projection for the LIDAR point $\mathbf{P}_e \in P_{rs,L}$ on the image $\bar{\mathbf{I}}_t$ through (4.2). We adopt the DBSCAN clustering algorithm [79, 80] to eliminate outliers of pixel $\mathbf{p}_e$ located far away from the road surface. We then get the boundary and the interiors to obtain new road surface pixel set $P^*_{rs,I}$. We also remove the corresponding outliers from set $P_{rs,L}$ to get updated road surface LIDAR data set (see Fig. 3.4a). By employing the road surface model, we reduce the noise from image segmentation and include more LIDAR points that fit for the surface model, which reduces outliers for lane marking detection from the LIDAR scan in the later part.

3.4.2 Lane Marking Segmentation in Each Modality

With the road surface pixel $P^*_{rs,I}$ extracted, we detect lane markings from the segmented road surface in the image. Define $g(\mathbf{p}_{k,t})$ as the intensity value for k-th pixel $\mathbf{p}_{k,t} = [u\ v]^\intercal$.

We set the intensity value of non-road pixels in $\bar{\mathbf{I}}_t$ to be zero. Now $\bar{\mathbf{I}}_t$ only contains black pixels and road surface pixels including lane markings. Lane marking pixels usually have higher intensity values. To reduce the noise from the image, we apply Gaussian blurring before segmenting lane marking pixels through image histogram. We obtain the binned histogram according to 256 intensity levels. We apply Gaussian mixture model using EM algorithm [81] to the histogram data and find the peak with the largest intensity value μ_γ with variance σ_γ. By applying three-sigma thresholding [82], we obtain a lower bound of the intensity value as $g_\gamma = \mu_\gamma - 3\sigma_\gamma$. We obtain lane marking pixels $\mathbf{p}_l$ (see Fig. 3.4b) in set

$$P_{l,I} = \{\mathbf{p}_{k,t} | g(\mathbf{p}_{k,t}) \geq g_\gamma\}. \tag{3.19}$$

Fig. 3.4b illustrates the lane marking pixels.

Lane markings are also detected using LIDAR scans due to the their high laser reflectivity by design. Recall we have extracted road surface data from the LIDAR scan in set $P^*_{rs,L}$. Recall that variable $I_e \in [0, 255]$ to be the intensity value for LIDAR point $\mathbf{P}_e$. We threshold I_e to obtain lane markings in LIDAR data,

$$P_{l,L} = \{\mathbf{P}_e \mid I_e \geq T_s, \mathbf{P}_e \in P^*_{rs,L}\}, \tag{3.20}$$

where threshold T_s is obtained using Otsu thresholding [83] that determines the optimal intensity value T_s by maximizing the variance between background (asphalt or concrete) and foreground (lane marking) classes.

3.4.3 Left and Right Lane Marking Determination

The lane markings from individual modalities can be further filtered through cross modality validation. Eqs. (3.14) and (4.2) allow us to project points between LIDAR coordinates and image coordinates back and forth. Hence, we can intersect the lane marking points between $P_{l,I}$ and $P_{l,L}$ at LIDAR coordinates and generate a set $P^*_{l,L}$ which contain dual-modal lane markings that are more robust than those in individual modalities.

At this moment, the lane markings may belong to several lane boundaries in a multi-lane highway and include many outliers. We first filter out all candidate lane boundaries before identifying the exact left and right lane boundaries. Define $\mathbf{L}_j$ as the j-the lane boundary. We apply T-Linkage [84] to obtain $\mathbf{L}_j$'s. T-Linkage is capable of detecting multiple lane boundaries in the presence of outliers but it requires a model for $\mathbf{L}_j$. We employ the cubic uniform B-spline lane boundary curve which is defined for a collection of $n+1$ control points $\mathscr{M}_l = \{\mathbf{P}_q\}$ from the set $P^*_{l,L}$ as,

$$\mathbf{L}_j(s) = \sum_{q=0}^{n} \mathbf{P}_q N_{q,3}(s). \tag{3.21}$$

Here, $N_{q,3}(s)$ are the basis functions with

$$\begin{aligned} N_{q,0}(s) &= \begin{cases} 1, & \text{if } s_q \le s \le s_{q+1} \text{ and } s_q < s_{q+1} \\ 0, & \text{otherwise} \end{cases} \\ N_{q,h}(s) &= \frac{s - s_q}{s_{q+3} - s_q} N_{q,h-1}(s) + \frac{s_{q+4} - s}{s_{q+4} - s_{q+1}} N_{q+1,h-1}(s), \end{aligned} \tag{3.22}$$

where $h = 1,2,3$, $s_q = q-3, q = 3,4,...,n+1$ with $s_0 = s_1 = s_2 = s_3$, and $s_{n+1} = s_{n+2} = s_{n+3} = s_{n+4}$. Note that the shape of the cubic uniform B-spline curve is dominated by the control points. We can impose curvature constraints when choosing points to instantiate

models in T-Linkage.

After T-Linkage, we have a set of candidate lane boundaries $\mathbf{L}_j$'s. We need to identify left and right lane boundaries and their associated lane markings. A simple observation is that our left and right lane boundaries must intersect the low boundary of the image at positions closer to center of the low boundary because that is the current vehicle location. Recall that the horizontal dimension in the image is the u-axis. The intersection of $\mathbf{L}_j$ with low boundary generate u_j. If the center is at u_c, it is a natural divider for the left and right sides. Then we sort $|u_j - u_c|$ to generate two sorted sequences with increasing distances. We then pair them by considering the fact that the distance between left and right boundaries should be longer than $d_{\min}$ and shorter than $d_{\max}$. We might have multiple solutions but we use how close they are to the previous period to find the optimal. This simple search help us determine left and right boundaries, defined as $\mathbf{L}_l$ and $\mathbf{L}_r$, respectively. For each boundary, we find all closest points in $P^*_{l,L}$ and hence we determine $\mathscr{X}_l := \{\mathbf{X}_l\}$ and $\mathscr{X}_r := \{\mathbf{X}_r\}$ as the resulting left and right lane marking sets, respectively.

With the $\mathscr{X}_l$ and $\mathscr{X}_r$ obtained, we project them back to $\bar{\mathbf{I}}_t$ to search for more lane marking points. Denote $\mathbf{x}_{j,w}$ to be the corresponding projection pixel in $\bar{\mathbf{I}}_t$ for the LIDAR points in set $\mathscr{X}_l \bigcup \mathscr{X}_r$. We have lane marking pixel set

$$\mathscr{S}_Q = \{\mathbf{p}_{k,t} | \ \|\mathbf{p}_{k,t} - \mathbf{x}_{j,w}\| \leq d_j, \mathbf{p}_{k,t} \in \tilde{\mathbf{I}}_t, g(\mathbf{p}_{k,t}) \geq g_\gamma\}, \tag{3.23}$$

where nonnegative variable d_j is a constant threshold value. Thus we have all values needed for metrics in Section 3.3.

3.5 Experiments

We have implemented the proposed method on a Laptop PC with an Intel(R) Core$^{\text{TM}}$ i7-3517U CPU@1.90GHz and 8 GB memory. The Benchmark contains images showing a variety of street scenes captured from a vehicle driving around the city of Karlsruhe.

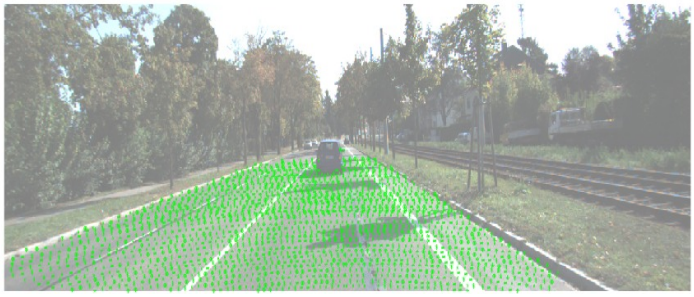
Besides the raw data, KITTI comes with a number of labels for different tasks relevant to autonomous driving to evaluate the performance of our road extraction and lane detection results. Parameters are set according to the experiments empirically. We set $d_{\max}$ and $d_{\min}$ in (3.11) to be 4.60 m and 2.70 m, respectively. We also set d_ε in (3.19) to be 0.1 m, and d_j to be 20 pixels for (3.23).

To verify our metrics, we use six different sequences of two categories from KITTI dataset including city trail data and road trail data. Fig. 3.5 illustrates testing results. Let us explain the abnormality of lane markings reflected by performance metrics as follows. Fig. 3.6a shows a case that lacks lane markings at the beginning of the video sequence. Fig. 3.6b shows a case that lanes start merging while the vehicle's current lane does not have left lane markings. Fig. 3.6c shows that the vehicle is entering a main road but the current lane does not have lane markings. Fig. 3.6d shows that an intersection does not have the lane markings to guide the vehicle. Fig. 3.6e shows a 3-way junction lacks part of the left lane markings and has irregular lane markings. Fig. 3.6f shows a case that one side of vehicle is just the shoulder with no right lane markings. To summarize, Fig. 3.5 shows that our metrics are able to capture the abnormality of the lane markings and can be used as a measurement tool for road inspection.

3.6 Conclusion

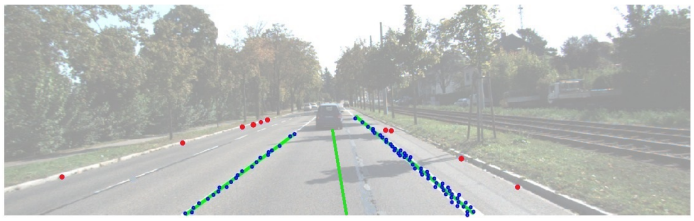
We focused on development of a LMQA method for improving infrastructure for autonomous driving. The method assumed an egocentric view from an inspection vehicle equipped with a GPS receiver, a frontal view camera, and a LIDAR for LMQA. We presented metrics and algorithms for lane marking assessment. Three lane marking quality metrics were proposed and modeled mathematically: correctness, shape, and visibility. We also proposed a dual-modal algorithm to facilitate the computation of the three metrics. We took both prior map uncertainty and sensory uncertainty into consideration in

formulating our metrics and the algorithm. We implemented the algorithm and tested it under an open dataset. The results were satisfying. Our method was able to identify low quality segments of lane markings.



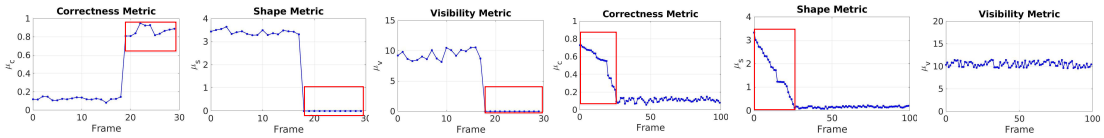
(a) he green points are the projected 3D LIDAR points from the road surface model by sensor fusion.

(b) Lane marking pixels in image.

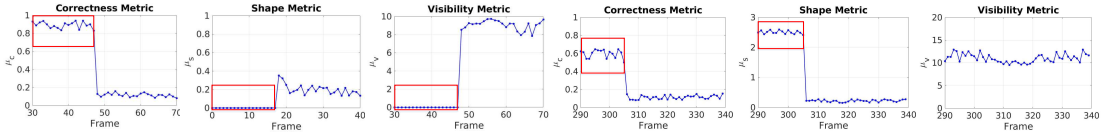


(c) Blue points are the pixel-wise projection of the lane marking points from the LIDAR scan, red points are outliers, and the green curve in the middle is the lane center curve and the other two are the left and right lane boundaries, respectively.

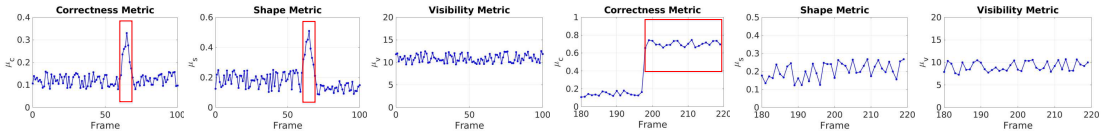
Figure 3.4: Sample intermediate algorithm outputs (best viewed in color).



(a) 2011_09_26_drive_0005. (b) 2011_09_26_drive_0056.

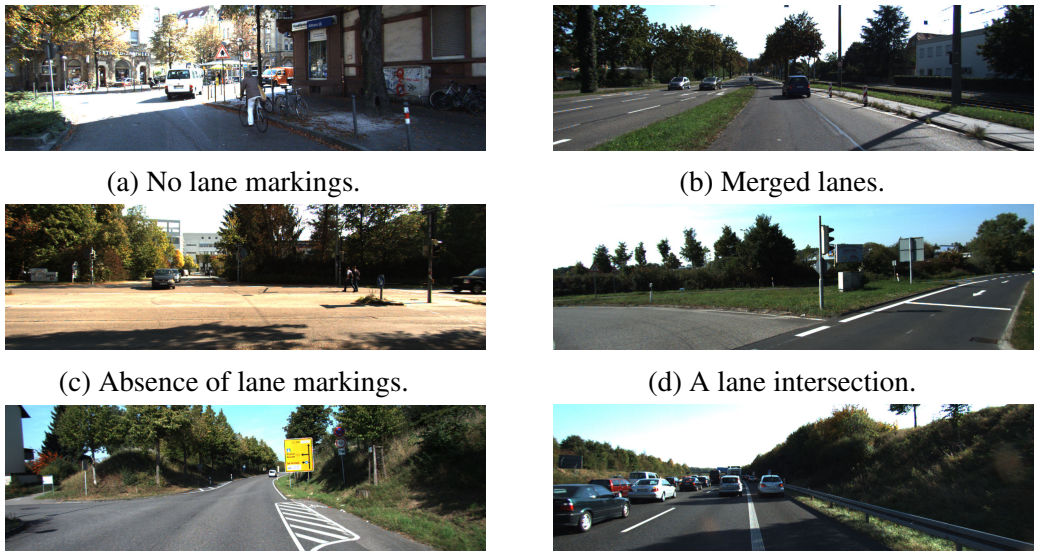


(c) 2011_09_26_drive_0001. (d) 2011_09_26_drive_0029.



(e) 2011_09_26_drive_0070. (f) 2011_09_26_drive_0047.

Figure 3.5: Performance metrics for six sequences from the KITTI dataset. Red boxes indicate lane marking anomalies identified by correctness, shape or visibility metrics.



(a) No lane markings. (b) Merged lanes.

(c) Absence of lane markings. (d) A lane intersection.

(e) A 3-way junction. (f) Right shoulder w/o markings.

Figure 3.6: Typical scenarios of abnormal lane markings. Figures labels correspond to those in Fig. 3.5.

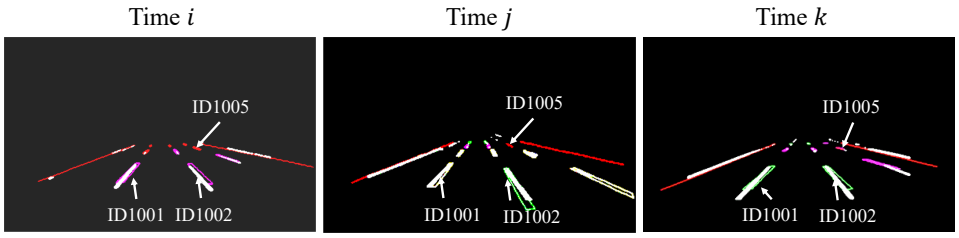
4. BELIEF SPACE CROSS VALIDATE FROM CROWDSOURCED DATA FOR LANE MARKING VERIFICATION [1]

4.1 Introduction

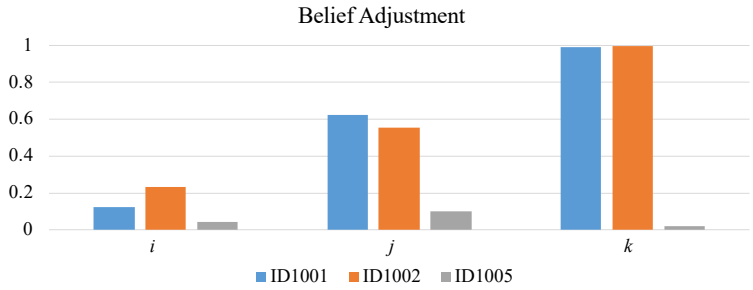
Fo better scene understanding in the belief space, we utilize crowdsourced images from multiple vehicles to verify the LMs for HD map maintenance. The fast-evolving autonomous vehicle (AV) technology has the potential to drastically change modern transportation. Many AVs rely on a HD map to navigate around. HD maps include a highly accurate and realistic representation of the road, including many types of objects such as LMs, traffic signs, street lamp posts, etc. In the absence of accurate GPS signals, the precision of LMs in HD maps is important for the vehicle to recognize lanes and plan for its motion. However, LMs are not necessarily constant because they wear out due to road usage and also vary due to road construction and maintenance. A set of outdated LMs may lead to erroneous localization results. Frequently recollecting LM data is not only cost-prohibitive but also unviable. It is common that the change frequency of the road environment, especially in urban environment, is always faster than the rate of sending out a mapping fleet.

We propose to utilize crowdsourced images to keep LMs up-to-date in the HD map. We view LMs in both HD map and the crowdsourced images as observations of LM distribution. We model the posterior LM distribution in either source using Gaussian kernels. We take the uncertainty from camera poses into consideration for image-based observations. We examine their consistence within the same image coordinate using statistical hypothesis testing. We then establish a sequential Bayesian model for updating the posterior

[1] Reprinted with permission from "Lane Marking Verification for High Definition Map Maintenance Using Crowdsourced Images" by B. Li, D. Song, A. Kingery, D. Zheng, Y. Xu, and H. Guo, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, October. 25-29, 2020.



(a) LMs are extracted from front-view camera images. We project LMs from the HD map, which have unique ID numbers in the map database, into the front-view camera images (in color). Different colors stand for different stages for LM belief. LM in purple is “inconsistent", LM in Yellow is “undetermined", and LM in green is “consistent".



(b) With accumulated observations such as camera images in (a) from left to right, the LMs with ID1001 and ID1002 are “consistent" and kept in the HD map, while the LM with ID1005 is labeled as “inconsistent" and removed from the HD map.

Figure 4.1: We extract belief for each LM in the map and accumuate historical observations from camera to verify the LMs.

LM distributions using a sequence of crowdsourced images. We threshold the conditional probability to determine if each LM is consistent, inconsistent, or undetermined. We have implemented our map verification algorithm and tested it using real data. The experimental results show that the algorithm has achieved its design goal and outperformed commonly used intersection over union (IoU) metric in precision, recall and F1-measure.

4.2 Related Work

AVs require up-to-date high definition maps to ensure safe navigation and to cope with environmental changes [85, 86, 87, 88]. To create updatable HD maps, it is necessary to

1) have the ability to detect LMs, 2) design a flexible data structure to represent maps, and 3) develop algorithms to validate and maintain HD maps.

LM detection and tracking play an important role in autonomous driving, which has been studied for years [60, 89, 90, 91, 92]. Andrade et al. [93] use Hough transform to track LMs through the shape-preserving spline interpolation. In [94], we fuse camera images and lidar point clouds to detect LMs and assess LM quality by proposing correctness, shape and visibility metrics. Our recent work [95] also generates virtual LMs in sensor space while considering vehicle size and kinodynamic constraints. Huang et al. [69] detect and estimate multiple LMs by fusing calibrated video images and laser range data captured by a moving vehicle. Kang et al. [96] propose a probabilistic decision-making algorithm to track curbs that uses interacting multiple model method for autonomous mobile robot navigation. Here we build on existing LM detection work to provide inputs for HD map maintenance.

In robotics, simultaneous localization and mapping (SLAM) has developed many map representations as a collection of landmarks which include occupancy grids [97, 98], sparse visual features [99, 100, 101], and point clouds [102]. However, most existing map representation are designed for stationary objects without consideration of frequent updates.

In recent developments, Ryde et al. [103] employ multi-resolution occupied voxel lists to represent 3D spatial maps, which detects changes by finding points that do not locate inside an occupied voxel after alignment. Aijazi et al. [104] extract temporarily static and mobile 3D point clouds by matching sensor's observations on at different times of the day, which yields the progressively modified 3D urban landscape. Wang et al. [105] detect and track dynamic objects in dynamic environments, and build a map that satisfies both navigation and safety requirements for autonomous driving in urban areas. Julie et al. [106] assign scores for features in the map, which depend on the geometric distribution

and characteristics when the features are re-detected at a different time. Sun et al. [107] present a novel semantic mapping approach for the successfully mapping of a dynamic environment using more than two weeks of data. Nurminen et al. [108] propose methods to support spatial updating and rapid alignment of physical and virtual spaces in the 3D mobile maps. Unlike existing approaches, we employ Bayes' theorem to track the belief changes of LMs by fusing observations from crowdsourced data. Our method can remove or add LMs as needed over long periods of time.

4.3 Problem Formulation

A vehicle is driving on a street with HD maps. It takes images from its camera and verifies if LMs on the road are the same as those in its HD map. The HD map usually consists of a variety of objects such as LMs, traffic signs, street lamp posts, etc. Since LMs are the most common landmarks to help achieve high precision global localization. Here we focus on verifying LMs in the HD map.

4.3.1 Assumptions and Coordinate Systems

The vehicle is equipped with a front facing camera to observe the LMs. We assume that the camera is pre-calibrated, and the nonlinear distortion of images has been removed.

All coordinate systems or frames are right-handed systems and defined as follows,

- $\{\mathscr{C}\}$ defines the camera coordinate system with its origin at the camera center, z-axis pointing forward coinciding with the camera's principal axis, and its x-axis and y-axis parallel to the horizontal and vertical directions of camera imaging sensor, respectively.

- $\{\mathscr{I}\}$ defines the image coordinate system. Let ${}^{\mathscr{I}}\mathbf{x} = [u\ v]^{\mathsf{T}} \in \mathbf{I}_t$ be a pixel point in camera image $\mathbf{I}_t$ in $\{\mathscr{I}\}$ at time t where (u, v) is the image coordinate.

- $\{\mathscr{G}\}$ defines the vehicle global frame with the x-axis pointing to the east, y-axis

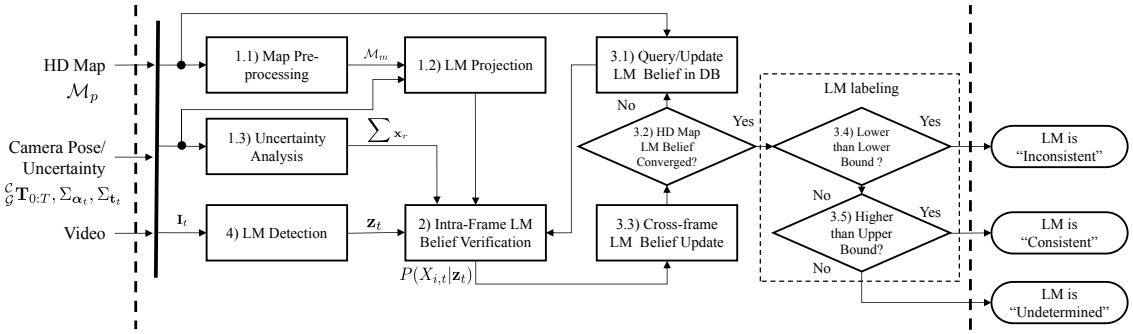


Figure 4.2: System diagram.

pointing north, and z-axis pointing upward.

Note that we will attach frames to a variable as the left super and sub scripts to indicate which frame the variable is associated with.

4.3.2 HD Map and Camera Inputs

We have inputs from both HD map database and the on-board camera/sensors. From the HD map, we have,

- ${}^{\mathcal{G}}\mathbf{M}_i$ is the i-th LM consisted of a set of points ${}^{\mathcal{G}}\mathbf{M}_i := \{{}^{\mathcal{G}}\mathbf{M}_{i,j} \in \mathcal{R}^3 | j = 1,2,...,n_b\}$, where $\mathbf{M}_{i,j}$ is the j-th point in the LM and n_b is the number of the points in the i-th LM. Correspondingly, we also have ${}^{\mathcal{I}}\mathbf{M}_i$ in camera frame.

- $\mathcal{M}_p$ is a HD map consisted of a set of LMs $\mathcal{M}_p := \{{}^{\mathcal{G}}\mathbf{M}_i \subset \mathcal{R}^3 | i = 1,2,...,n_a\}$ where ${}^{\mathcal{G}}\mathbf{M}_i$ is the i-th LM set and n_a is the number of LMs.

Through an on-board map-based localization algorithm, the vehicle obtains the camera pose and its uncertainty range. Denote the camera's pose at time t by ${}^{\mathcal{C}}_{\mathcal{G}}\mathbf{T}_t$. ${}^{\mathcal{C}}_{\mathcal{G}}\mathbf{T}_t$ is the rigid body transformation from frame $\mathcal{G}$ to frame $\mathcal{C}$,

$$ {}^{\mathcal{C}}_{\mathcal{G}}\mathbf{T}_t = \begin{bmatrix} {}^{\mathcal{C}}_{\mathcal{G}}\mathbf{R}_t & {}^{\mathcal{C}}_{\mathcal{G}}\mathbf{t}_t \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, $$

where ${}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t$ is the translation vector from the origin of $\{\mathscr{G}\}$ to the origin of $\{\mathscr{C}\}$, and ${}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t \in \mathscr{SO}^3$ is the rotation matrix from $\{\mathscr{G}\}$ to $\{\mathscr{C}\}$ and represented in Euler angle $\alpha_t = [\phi, \theta, \psi]^\mathsf{T}$ in Z-Y-X order.

For uncertainties of camera poses, we represent the vehicle state as Gaussian distribution, and have $\alpha_t \sim N(\bar{\alpha}_t, \Sigma_{\alpha_t})$ and ${}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t \sim N({}^{\mathscr{C}}_{\mathscr{G}}\bar{\mathbf{t}}_t, \Sigma_{\mathbf{t}_t})$, where Σ_{α_t} and $\Sigma_{\mathbf{t}_t}$ are the corresponding covariance matrices for rotation and translation, respectively. Here, the overhead symbol '–' represents the mean of the vector, and the vehicle state's distribution α_t and ${}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t$ is changing with incoming camera images $\mathbf{I}_t$.

For camera image $\mathbf{I}_t$, we can extract LM points using lane detection algorithms in the Chapter 3. It results in ${}^{\mathscr{I}}\mathbf{x}_s$ as LM points in the image (see Box 4 in Fig. 5.2). We do not need to group them into different LM sets. Assemble all LM points ${}^{\mathscr{I}}\mathbf{x}_s$, we obtain set $\mathbf{z}_t$ and its cardinality $n_h = |\mathbf{z}_t|$.

4.3.3 Problem Definition

We want to use crowdsourced images to confirm or disconfirm each LM set ${}^{\mathscr{G}}\mathbf{M}_i$ in HD map. This will generate three labeled categories including "consistent", "inconsistent" or "undetermined." The "consistent" LM ${}^{\mathscr{G}}\mathbf{M}_i$ will be kept in the HD map while "inconsistent" LM points will be removed, and those "undetermined" LM points will require more observations in the future to ascertain its consistency.

Given a sequence of crowdsourced data ordered by time $t = 0, ..., T$ where T the latest time index, our problem is defined as follows,

Problem 2. Given the HD map $\mathscr{M}_p$, camera images $\mathbf{I}_{0:T}$, and historical camera poses ${}^{\mathscr{C}}_{\mathscr{G}}\mathbf{T}_{0:T}$ with known covariance matrices, label consistence category for each LM set ${}^{\mathscr{G}}\mathbf{M}_i$ in the HD map $\mathscr{M}_p$.

4.4 Algorithm

Fig. 5.2 illustrates our system diagram. It mainly contains the following blocks: (1.1-1.3) we project LM points from HD map into the correct camera frame of the vehicle. We analyze and compute the uncertainty of the projected points; (2) We update LM point belief modeling given the current image observation; (3.1-3.5) LM point belief update by accumulating all the historical camera observations. We start with the first block.

4.4.1 Lane Marking Projection and Uncertainty Analysis

Note that the vehicle samples data periodically. At discrete time t, we have the camera's pose $\{{}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t, {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t\}$. We extract a subset of LMs from the HD map based on vehicle speed v_t and distance threshold d_m,

$$\mathscr{M}_m = \{{}^{\mathscr{G}}\mathbf{M}_{i,j} \,|\, \|{}^{\mathscr{G}}\mathbf{M}_{i,j} + {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t^{\mathsf{T}}\,{}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t\| \leq d_m, {}^{\mathscr{G}}\mathbf{M}_{i,j} \in \mathscr{M}_p\}. \tag{4.1}$$

Here, $\|\cdot\|$ is the vector l^2-norm. Distance threshold d_m is obtained as follows,

$$d_m = \begin{cases} \zeta \cdot v_t t & \text{if } v_t > 0 \\ v_v, & \text{otherwise} \end{cases}$$

where ζ controls the overlapping regions of the HD map between neighboring $\mathscr{M}_m$, t is the sampling interval, and v_v is a constant. Define $\mathbf{U} = \mathbf{K}\,{}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t$, and $\mathbf{U}^{3\mathsf{T}}$ to be the third row of $\mathbf{U}$. Here, $\mathbf{K}$ is the intrinsic camera matrix under the pin hole model. We remove point ${}^{\mathscr{G}}\mathbf{M}_{i,j}$ in $\mathscr{M}_m$ that is in the back of the camera if the condition,

$$\mathbf{U}^{3\mathsf{T}}({}^{\mathscr{G}}\mathbf{M}_{i,j} + {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t^{\mathsf{T}}\,{}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t) < 0,$$

is satisfied. Recall $\mathscr{M}_m$ is made of a set of LM points with known LM index, and we have grouped the points belonging to the i-th LM as ${}^{\mathscr{G}}\mathbf{M}_i$. We accumulate such LMs that can be projected into image $\mathbf{I}_t$ in set $\{{}^{\mathscr{G}}\mathbf{M}_i | i \in \mathscr{M}_t\}$, where $\mathscr{M}_t$ is the index set.

Given the camera pose $\{{}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t, {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t\}$, we can project LMs from $\{\mathscr{G}\}$ to $\{\mathscr{C}\}$ through perspective projection,

$$\tilde{\mathbf{x}}_r = c_p \mathbf{K}({}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t \mathbf{X}_r + {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t) \tag{4.2}$$

for each LM point $\mathbf{X}_r \in {}^{\mathscr{G}}\mathbf{M}_i$, where c_p is a scalar, and a vector with symbol '$\sim$' on top is in its homogeneous representation. This generates a projected HD map pixel set $\mathbf{z}_{m,t} := \{{}^{\mathscr{I}}\mathbf{x}_r | \mathbf{X}_r \in {}^{\mathscr{G}}\mathbf{M}_i\}$ at time t.

The point positions of $\mathbf{X}_r$ are not noise free. We need to understand how it propagates to the image frame. The noise distribution of $\mathbf{X}_r$ is modeled as a zero-mean Gaussian with covariance $\sigma_r^2 \mathbf{I}_3$, where $\mathbf{I}_3$ is a 3×3 identity matrix and σ_r is determined by the accuracy of the HD map. As a function of $\nu = [\alpha_t^\mathsf{T}, \mathbf{X}_r^\mathsf{T}, {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t^\mathsf{T}]^\mathsf{T}$ in (4.2), we have

$$\mathrm{cov}\left(\begin{bmatrix} \alpha_t \\ \mathbf{X}_r \\ {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t \end{bmatrix}\right) = \begin{bmatrix} \Sigma_{\alpha_t} & \mathbf{0}_{3\times 3} & \mathrm{cov}(\alpha_t, {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t) \\ \mathbf{0}_{3\times 3} & \sigma_r^2 \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ \mathrm{cov}(\alpha_t, {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t) & \mathbf{0}_{3\times 3} & \Sigma_{\mathbf{t}_t} \end{bmatrix}, \tag{4.3}$$

by assuming that α_t is independent of the other two vectors. Then we have

$$\Sigma_{\mathbf{x}_r} = J_\nu \mathrm{cov}\left(\begin{bmatrix} \alpha_t \\ \mathbf{X}_r \\ {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{t}_t \end{bmatrix}\right) J_\nu^\mathsf{T}, \tag{4.4}$$

under the first-order approximation (see Box 1.3 in Fig. 5.2) in error forward propagation,

where $J_{\mathcal{V}}$ is the Jacobian matrix of (4.2) by

$$J_{\mathcal{V}} = \frac{\partial \tilde{\mathbf{x}}_r}{\partial \mathcal{V}} = c_p \mathbf{K} \begin{bmatrix} \frac{\partial ({}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t \mathbf{X}_r)}{\partial \alpha_t} & {}^{\mathscr{C}}_{\mathscr{G}}\mathbf{R}_t & \mathbf{I}_{3\times 3} \end{bmatrix}. \tag{4.5}$$

The covariance matrix $\Sigma_{\mathbf{x}_r}$ characterizes the uncertainty of the projected LM points from HD map to the current camera frame. It allows us to establish a belief model for LM points.

4.4.2 Intra-Frame Lane Marking Verification

It is worth noting that verifying LMs between the HD map and those in the current camera frame is not a point-to-point verification. In fact, this is a set-to-set association and requires a new belief model to facilitate this. We first establish a pixel-wise intra-frame belief function to verify LMs using a single frame.

Due to the existence of noises in HD maps, the projected HD map $\mathbf{z}_{m,t}$ can be understood as an observation of actual LMs in the current camera coordinate system. We model the conditional probability distribution of a pixel being a true LM pixel as a weighted sum of Gaussian functions established given the pixel and its neighbors (see Fig. 5.3a) in the observation $\mathbf{z}_{m,t}$,

$$f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t}) = \sum_{{}^{\mathscr{I}}\mathbf{x}\in N_e({}^{\mathscr{I}}\mathbf{x}_r)} w_a \frac{\exp(-\frac{1}{2}d({}^{\mathscr{I}}\mathbf{x}, {}^{\mathscr{I}}\mathbf{x}_r))}{2\pi\sqrt{|\Sigma_{\mathbf{x}_r}|}} \tag{4.6}$$

where $d({}^{\mathscr{I}}\mathbf{x}, {}^{\mathscr{I}}\mathbf{x}_r) = ({}^{\mathscr{I}}\mathbf{x} - {}^{\mathscr{I}}\mathbf{x}_r)^{\mathsf{T}}\Sigma_{\mathbf{X}_r}^{-1}({}^{\mathscr{I}}\mathbf{x} - {}^{\mathscr{I}}\mathbf{x}_r)$, $N_e({}^{\mathscr{I}}\mathbf{x}_r)$ is the neighboring set with $d({}^{\mathscr{I}}\mathbf{x}, {}^{\mathscr{I}}\mathbf{x}_r) \leq \kappa_m^2$, κ_m is a threshold, $|\Sigma_{\mathbf{x}_r}|$ is the determinant of $\Sigma_{\mathbf{x}_r}$, w_a is a normalization factor, and $a = 1, 2, ..., n_g$. Here we set $\kappa_m^2 = F^{-1}(\alpha, 2)$, where $F^{-1}(\alpha, 2)$ is the inverse cumulative χ^2 distribution function with a desired confidence level of α and 2 degrees of freedom.

Similarly, the current camera image also provides an observation $\mathbf{z}_t$, we can model the

conditional probability distribution $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$ where $f_s(\cdot)$ shares the same format of $f_m(\cdot)$ (4.6) except that the noise covariance matrix is $\sigma_s^2\mathbf{I}_2$ instead of $\Sigma_{\mathbf{x}_r}$, and σ_s is determined by the accuracy of our LM segmentation model. An example of $f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})$ and $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$ is shown in Fig. 5.3b.

For a pixel ${}^{\mathscr{I}}\mathbf{x} \in \mathbf{z}_t$, we can obtain both $f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})$ and $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$. This allows us to test if ${}^{\mathscr{I}}\mathbf{x}$ is a consistent LM pixel across the HD map data and the camera image by verifying if $f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})$ and $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$ are the same distribution through goodness of fit test,

$\mathbf{H_0}$: ${}^{\mathscr{I}}\mathbf{x}$ is a consistent LM pixel.

$\mathbf{H_1}$: Otherwise.

Through chi-square goodness of fit test [109], we have

$$\chi^2 = \frac{\left(f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t) - f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})\right)^2}{f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})}.$$

We reject $\mathbf{H_0}$ if

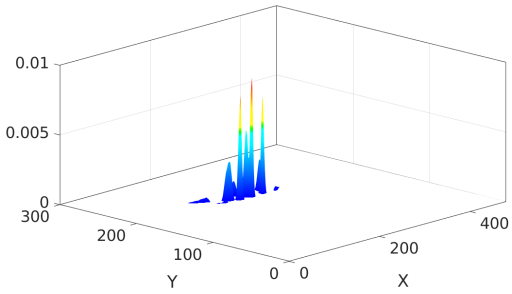
$$\chi^2 > \chi^2_{1-\beta,1},$$

where β is the significance level.

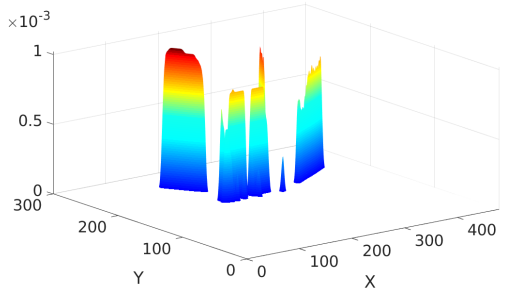
Thus for each LM $\mathbf{z}_{m,t}$ at time t, we obtain the consistent pixel set as,

$$\begin{aligned}\mathscr{X}_{i,t} := \big\{ {}^{\mathscr{I}}\mathbf{x} | f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t) \leq \\ f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t}) + \sqrt{f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})\chi^2_{1-\beta,1}}, {}^{\mathscr{I}}\mathbf{x} \in \mathbf{z}_{m,t} \big\}.\end{aligned}$$

Define $X_{i,t}|\mathbf{z}_t$ as the conditional spatial distribution of the i-th LM ${}^{\mathscr{G}}\mathbf{M}_i$ in camera frame



(a) Pixel-wise LM probability distribution $f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})$ from the HD map.



(b) $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$ from the image.

Figure 4.3: Pixel-wise LM probability distribution $f_m({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_{m,t})$ from the HD map in (a) and $f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t)$ from the image in (b).

given the observation $\mathbf{z}_t$. Then we have

$$P(X_{i,t}|\mathbf{z}_t) = \begin{cases} \frac{1}{\xi}\sum_{{}^{\mathscr{I}}\mathbf{x}\in\mathscr{X}_{i,t}} f_s({}^{\mathscr{I}}\mathbf{x}|\mathbf{z}_t), & \text{if } \mathscr{X}_{i,t} \neq \emptyset, \\ 0, & \text{otherwise}, \end{cases} \tag{4.7}$$

where ξ is a normalization factor.

Noted that as $P(X_{i,t}|\mathbf{z}_t)$ is easily influenced by the current observation $\mathbf{z}_t$, and current camera pose with respect to $\{\mathscr{G}\}$. We need to fuse observations from multiple vehicles at different times to ensure we can identify correct consistency category so that the i-th LM should be kept or removed from the HD map $\mathscr{M}_p$.

4.4.3 Cross-frame Lane Marking Belief Update

LMs in the HD map can be classified into three categories: 1) LMs with no matchings in the image; 2) LMs which have appeared in the image; and 3) false-positive LMs caused by noises, which will be filtered out with more observations. Here we combine all the historic observations for the LM ${}^{\mathscr{G}}\mathbf{M}_i$ from the crowdsourced images to establish a robust belief function (see Box 3.3 in Fig. 5.2) and verify the existence of LMs in the map.

Accumulating all images up-to-date into current observation $\mathbf{z}_t$ at time t, we have $\mathbf{Z}_{0:t} = \bigcup_t \{\mathbf{z}_t\}$. Note that $\mathbf{Z}_{0:t-1} = \mathbf{Z}_{0:t} \setminus \mathbf{z}_t$. Define $P(X_i|\mathbf{Z}_{0:t-1})$ as the conditional spatial probability of the LM ${}^{\mathscr{G}}\mathbf{M}_i$ given the observation set $\mathbf{Z}_{0:t-1}$. Similarly, we define the conditional probability $P(X_i|\mathbf{Z}_{0:t})$ here. To verify the existence of the LM ${}^{\mathscr{G}}\mathbf{M}_i$, our problem becomes how to compute $P(X_i|\mathbf{Z}_{0:t})$ given the current observation $\mathbf{z}_t$, previous observation set $\mathbf{Z}_{0:t-1}$ and the conditional probability $P(X_i|\mathbf{Z}_{0:t-1})$.

We decompose the conditional probability $P(X_i|\mathbf{Z}_{0:t})$ and have,

$$\begin{aligned} P(X_i|\mathbf{Z}_{0:t}) = P(X_i|\mathbf{Z}_{0:t-1},\mathbf{z}_t) &= \frac{P(\mathbf{z}_t,\mathbf{Z}_{0:t-1},X_i)}{P(\mathbf{z}_t,\mathbf{Z}_{0:t-1})} \\ &= \frac{P(\mathbf{z}_t,\mathbf{Z}_{0:t-1}|X_i)P(X_i)}{P(\mathbf{z}_t,\mathbf{Z}_{0:t-1})}. \end{aligned} \tag{4.8}$$

Since observations in $\mathbf{Z}_{0:t-1}$ are independent of each other, we have $P(\mathbf{z}_t,\mathbf{Z}_{0:t-1}|X_i) = \prod_{t=0}^{t} P(\mathbf{z}_t|X_{i,t})$ and $P(\mathbf{z}_t,\mathbf{Z}_{0:t-1}) = \prod_{t=0}^{t} P(\mathbf{z}_t)$. Plug them into (4.8), we obtain

$$P(X_i|\mathbf{Z}_{0:t}) = \frac{P(X_i)\prod_{t=0}^{t} P(\mathbf{z}_t|X_{i,t})}{\prod_{t=0}^{t} P(\mathbf{z}_t)}. \tag{4.9}$$

Similarly, we obtain $P(X_i|\mathbf{Z}_{0:t-1})$,

$$P(X_i|\mathbf{Z}_{0:t-1}) = \frac{P(X_i)\prod_{t=0}^{t-1} P(\mathbf{z}_t|X_{i,t})}{\prod_{t=0}^{t-1} P(\mathbf{z}_t)}. \tag{4.10}$$

Combine (4.9) and (4.10), we have

$$\frac{P(X_i|\mathbf{Z}_{0:t})}{P(X_i|\mathbf{Z}_{0:t-1})} = \frac{P(X_i)\prod_{t=0}^{t} P(\mathbf{z}_t|X_{i,t})}{P(X_i)\prod_{t=0}^{t-1} P(\mathbf{z}_t|X_{i,t})} \frac{\prod_{t=0}^{t-1} P(\mathbf{z}_t)}{\prod_{t=0}^{t} P(\mathbf{z}_t)} = \frac{P(\mathbf{z}_t|X_{i,t})}{P(\mathbf{z}_t)}. \tag{4.11}$$

Plug $P(\mathbf{z}_t|X_{i,t}) = P(X_{i,t}|\mathbf{z}_t)P(\mathbf{z}_t)/P(X_{i,t})$ into (4.11) and we have

$$P(X_i|\mathbf{Z}_{0:t}) = \zeta P(X_{i,t}|\mathbf{z}_t)P(X_i|\mathbf{Z}_{0:t-1}), \tag{4.12}$$

where ζ is a normalization factor.

With more observations, we update the conditional probability $P(X_i|\mathbf{Z}_{0:t})$ for the i-th LM until it converges. For initialization, we set $P(X_i|\mathbf{Z}_{0:t-1})$ to be 1, and utilize (4.7) to update $P(X_i|\mathbf{Z}_{0:t})$ in (4.12).

We threshold $P(X_i|\mathbf{Z}_{0:t})$ to determine if the i-th LM is consistent or not. Define ε_u and ε_v, $1 > \varepsilon_u > \varepsilon_v > 0$, as thresholds to determine if an LM is consistent or not. If $P(X_i|\mathbf{Z}_{0:t}) \geq \varepsilon_u$, then the i-th LM is consistent; if $P(X_i|\mathbf{Z}_{0:t}) \leq \varepsilon_v$, then the i-th LM is inconsistent; otherwise, the i-th LM ${}^{\mathscr{G}}\mathbf{M}_i$ is undetermined and we expect more observations to confirm its consistency.

4.4.4 Algorithm

We summarize our our LM verification algorithm in Algorithm 1. It is noted that we stop updating $P(X_i|\mathbf{Z}_{0:t})$ for the LM ${}^{\mathscr{G}}\mathbf{M}_i$ if $P(X_i|\mathbf{Z}_{0:t}) \geq \varepsilon_u$, thus the computation complexity can be greatly decreased with the increasing number of consistent LMs. Besides, we also utilize a local database to store the LM belief every time with new observations to decrease the memory usage (see Box 3.1 in Fig. 5.2). If an LM's belief by using crowdsourced images is still below the pre-selected threshold and required to be removed, we

Algorithm 1: Lane Marking Verification

Input: $\mathscr{M}_p, \mathbf{T}_{0:t}, \sum_{\alpha_{0:t}}, \sum_{\mathbf{t}_{0:t}}, \mathbf{Z}_{0:t}$
Output: The i-th LM ${}^{\mathscr{G}}\mathbf{M}_i$ is consistent or not
for $t \in \{0,1,...,t\}$ **do** // $O(t)$
  Obtain set $\mathscr{M}_m$ using (4.1); // $O(n\log n)$
  Compute ${}^{\mathscr{I}}\mathbf{x}_r$ through (4.2); // $O(1)$
  Generate $f_m({}^{\mathscr{I}}\mathbf{x})$ by (4.6) ; // $O(n_g)$
  Get $f_s({}^{\mathscr{I}}\mathbf{x})$ through $\mathbf{z}_t$; // $O(n_h)$
  Attain $P(X_i|\mathbf{z}_t)$ in (4.7); // $O(1)$
  Obtain $P(X_i|\mathbf{Z}_{0:t-1})$ and $P(X_i|\mathbf{Z}_{0:t})$; // $O(1)$
  if $P(X_i|\mathbf{Z}_{0:t}) \geq \varepsilon_u$ **then**
    Report ${}^{\mathscr{G}}\mathbf{M}_i$ as “consistent"; // $O(1)$
    Stop updating $P(X_i|\mathbf{Z}_{0:t})$; // $O(1)$
  else if $P(X_i|\mathbf{Z}_{0:t}) \leq \varepsilon_v$ **then**
    Mark ${}^{\mathscr{G}}\mathbf{M}_i$ as “inconsistent"; // $O(1)$
  else
    Mark ${}^{\mathscr{G}}\mathbf{M}_i$ as “undetermined"; // $O(1)$

query the HD map and remove the corresponding LM.

We summarize the computational complexity of our algorithm, and have

Lemma 1. *Our lane marking verification algorithm runs in* $O(tn\log(n))$.

4.5 Experiments

We have implemented our algorithm in C++ under Ubuntu 16.04. It is tested on a Laptop PC with an Intel® Core™ i5-8265U CPU@1.60GHz and 8 GB RAM. We collect images using forward-looking cameras mounted on data collection vehicles. The data have been collected on the north segment of the 4th ring road in the Beijing. The vehicles runs on the same part of road back and forth at different days and different times. We collected two datasets with different weather conditions (see Tab. 4.1). The image resolution is 300 $\times$ 480. We plan to release our data and algorithm output, a total of 14815 frames to the

Table 4.1: Datasets for Comparison

Dataset	Date	Length	#Images	Weather
A	2019_07_01	634s	4953	Partially sunny
B	2019_07_29	352s	3088	Light rain

pubilc[2].

We set $\varepsilon_u = 0.99$ and $\varepsilon_v = 0.01$ in experiments. To evaluate the performance of our approach quantitatively, three metrics, including precision, recall, and F1-measure [110], are employed. The F1-measure is the harmonic mean of precision and recall. Recall that set $\mathbf{z}_{m,t}$ contains all the project pixels for the i-th LM and $\mathbf{z}_t$ has the LM pixels extracted from the images. For comparison, we compare our algorithms to the following approaches.

- r-IoU: we replace (4.7) using a common similarity metric: intersection over union (IoU),
$$P(X_{i,t}|\mathbf{z}_t) = \frac{|\mathbf{z}_{m,t} \bigcap \mathbf{z}_t|}{|\mathbf{z}_{m,t} \bigcup \mathbf{z}_t|}.$$
- p-IoU: we project the all the historical LM pixels from the camera images for the ${}^{\mathcal{G}}\mathbf{M}_i$ to the latest image frame $\mathbf{I}_T$ and use IoU metric above to verify its existence.

The experimental results in Tab. 4.2 show that our approach outperforms the r-IoU or p-IoU based approach. Set similarity based approach generates a relatively lower $P(X_{i,t}|\mathbf{z}_t)$ by disregarding potential pixels that belong to the LM and require more observations than our algorithm even for consistent LMs. Besides, the p-IoU based method ignores the camera pose uncertainty, map accuracy and lane marking pixel noise, and yields poor verification performance.

[2]http://telerobot.cs.tamu.edu/lane/

Table 4.2: Evaluation using real data

Dataset	Methods	Precision	Recall	F1-measure
A	Ours	**91.13%**	**92.47%**	**91.80%**
	r-IoU	88.32%	90.13%	89.22%
	p-IoU	76.24%	79.38%	77.78%
B	Ours	**92.36%**	**93.75%**	**93.05%**
	r-IoU	87.22%	88.17%	87.69%
	p-IoU	72.13%	75.26%	73.66%

In Fig. 4.4, we plot the receiver operating characteristic (ROC) curves for each method by varying their respective thresholds. In the ROC plane, the upper left corner represents the ideal result. It is clear that our method outperforms the IoU by a large margin. In fact, this is not surprising because IoU produces too many false negatives.

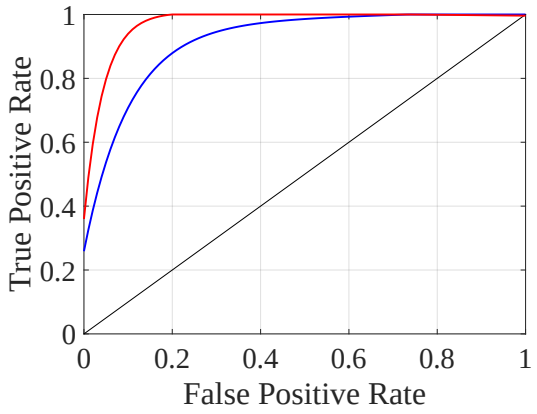


Figure 4.4: ROC curve for lane marking verification in comparison.

We present ten example LMs from the HD map to illustrate the belief update process: six consistent LMs have been identified and kept in the map by our algorithm and four LMs have been identified as inconsistent and hence removed. We plot their belief changes as the number of the observations increase in Fig. 4.5. Initially, most LMs are “undetermined"

due to the lack of enough observations. With more and more incoming observations, LM statuses converge to either "consistent" or "inconsistent." For an LM that is kept in the map, it is clear that $P(X_i|\mathbf{Z}_{0:t})$ grows monotonically toward ε_u. For LM that is inconsistent with the map, its belief is mostly at a lower level all the times and below the threshold ε_v, as expected.

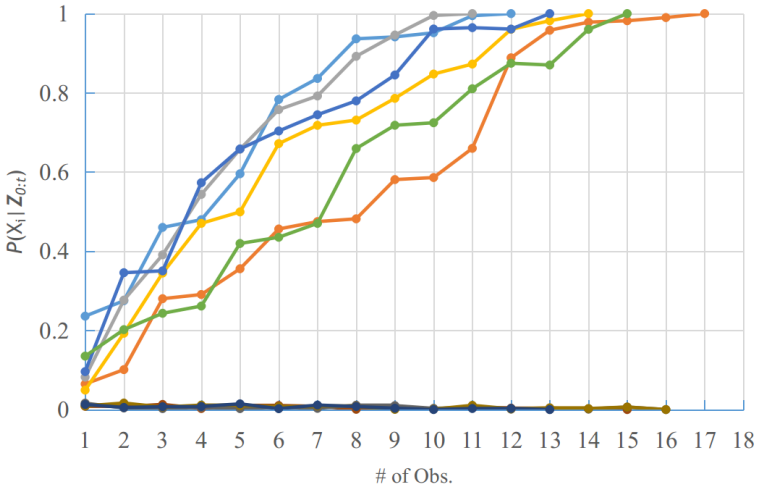


Figure 4.5: LM belief adjustment with more and more observations.

4.6 Conclusion

Here we presented an algorithm for updating LMs in HD maps using crowdsourced images. Realizing LMs in both the HD map and camera images contain noises, we model them respectively as observations of two LM spatial distributions in the camera frame and check if they agree with each other via a goodness of fit test. We model the Gaussian belief functions by considering noises from camera motion. We derive a sequential Bayes' model to allow the belief functions to be updated using crowdsourced images. We have implemented and tested our algorithms using data collected from testing vehicles and the

results showed that our approach is successful and outperformed the counterpart.

5. BELIEF SPACE FOR TIGHT CONNECTING BETWEEN PERCEPTION AND PLANNING[1]

5.1 Introduction

Belief space can also help us to tightly connect perception and motion planning. As an example, we develop a motion planning strategy for autonomous vehicles. As more and more companies are developing AVs, it is important to ensure that the driving behavior of AVs is human-compatible because AVs will have to share roads with human drivers in the years to come. When planning motion for an AV, we can adjust speed and trajectory in many possible ways but not all plans guarantee human compatibility, which requires the understanding of human decision process. A human driver is far better than an AV when handling complex situations. A human driver can avoid obstacles and still respect LMs and traffic cones to a large degree. A human driver can override lane boundaries (LBs) in appropriate scenarios: LMs may disappear or be blocked by construction or parked vehicles, LMs may not be consistent with the traveling direction, a vehicle may be traveling too fast, thus being temporarily unable to follow the sudden changes in LMs, etc. In fact, there is a tight connection between perception for scene understanding and motion planning, which involves finding an optimal trajectory under multiple objectives.

However, traditional navigation design in AVs treats functionalities such as lane recognition, obstacle avoidance, local path planning, and lane following as separate modules which results in unnatural driving behavior from a human perspective. For example, a low-level obstacle avoidance as reflex behavior often emphasizes speedy response instead of incorporating in-depth LB understanding. The resulting obstacle avoidance may not be

[1]Reprinted with permission from "Virtual Lane Boundary Generation for Human-Compatible Autonomous Driving: A Tight connecting between Perception and Planning" by B.Li, D. Song, A. Ramchandani, H. Cheng, D. Wang, Y. Xu, and B. Chen, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, Nov. 4-8, 2019.

human-compatible.

We propose a new tightly-connected perception-planning framework to improve human-compatibility. Using GPS-camera-lidar multi-modal sensor fusion, we detect ALBs and propose availability-resonability-feasibility tests to determine if we should generate VLBs or follow ALBs. When needed, VLBs are generated using a dynamically adjustable multi-objective optimization framework that considers obstacle avoidance, trajectory smoothness (to satisfy vehicle kinodynamic constraints), trajectory continuity (to avoid sudden movements), GPS following quality (to execute global plan), and lane following or partial direction following (to meet human expectation). The resulting trajectory is more human compatible than existing approaches, especially when coping with difficult conditions (see Fig. 5.1).

We have implemented our algorithm and tested it with the KITTI open source data set. The source codes have been released on Github™. The results have shown that our algorithm automatically and dyanmically switches between VLBs and ALBs. The ratio of time the VLB dominated segments range from 29% to 100% depending upon road scenarios. Our multiple-objective tightly-connected perception-planning framework produces high quality trajectories in city environments.

5.2 Related Work

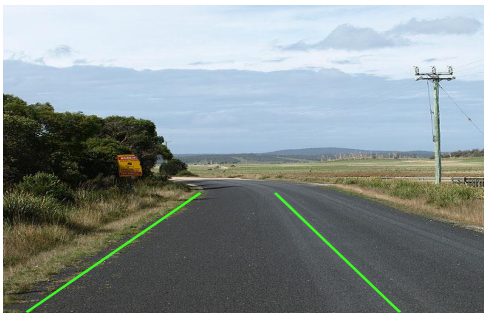
Our research is related to LB detection and tracking, local path planning, and obstacle avoidance.

LB detection and tracking plays an important role in autonomous driving, which has been studied for years [60, 89]. Andrade et al. [93] propose to detect and track LBs by using Hough transform and a shape-preserving spline interpolation. Li et al. [111] introduce predictive random sample consensus (RANSAC) to fit and track LBs in the presence of heavy noise and outliers. Petrovai et al. [91] apply stereovision to track 3D LBs. Huang

(a) Current lane lacks left lane boundary.

(b) Traffic cones alter roads

(c) Parked cars block streets



(d) There are no LMs at all.

Figure 5.1: We generate virtual lane boundaries for autonomous driving to ensure human compatible driving under complex road conditions. Green curves are the VLBs generated by our algorithm (best viewed in color).

et al. [69] detect and estimate multiple LBs by fusing calibrated video imagery and laser range data for a moving vehicle. Joshi et al. [92] use a 1D Laplacian filter to extract and track LBs from 3D lidar data. Kang et al. [96] propose a probabilistic decision-making algorithm to track curbs that uses interacting multiple model method for autonomous mobile robot navigation. Most existing methods detect and track LBs as an isolated perception problem. In this work, we tightly connect perception with planning by generating VLBs in sensor space while considering vehicle size and kinodynamic constraints.

Traditionally, obstacle avoidance is often designed as a low level reflex for a robot to stay away from obstacles. Obstacle avoidance for autonomous driving involves planning the AV's trajectory by satisfying control objectives subject to non-collision constraints.

Many methods for obstacle avoidance have been proposed [112, 113]. Khatib [25] designs artificial potential field to represent the obstacles so that a robot reaches the goal without colliding with obstacles. Song et al. [114] construct a vision vector space to facilitate motion planning to avoid obstacles by fitting the dynamic requirement of a motorcycle. Kahlouche et al. [115] employ optical flow to get the information about the robot environment for visual obstacle avoidance. Sgorbissa et al. [116] integrate a prior knowledge of the environment with local perceptions, and guarantee that the robot can never be trapped in deadlocks even when operating within a partially unknown dynamic environment. For simple mobile robots in slow speed, obstacle avoidance does not have to be built on sophisticated perception model. However, an AV has to follow traffic rules and handle conflicting goals to meet human expectations.

Local path planning produces a collision-free path for AVs based on a predefined global route and *in situ* information from on-board sensors [117]. Compared with the grid-based methods [118], the sampling-based methods [119] are more widely used to find a collision-free path due to the high-speed driving requirement. Likhachev et al. [120] present a graph-based planning and re-planning algorithm, which is able to produce bounded sub-optimal solutions to speed up decision time. Chu et al. [121] propose to generate an optimal path for off-road autonomous driving with static obstacles. Li et al. [122] employ a hierarchical planning strategy by extracting a reference path from the lidar-based localization map. Bai et al. [123] utilize an intention-aware online planning approach for AVs to drive near pedestrians safely, efficiently, and smoothly. Ma et al. [124] propose an efficient sampling-based planning method, which introduces a rule-template set based on the traffic scenes and an aggressive extension strategy of search tree. However, these dedicated planning approaches seek to find an optimal trajectory in the free space to avoid static or dynamic obstacles. The trajectory generated may not be compatible with human drivers.

5.3 Problem Definition

The vehicle is equipped with a frontal view camera, a lidar, and a GPS receiver, which is the common sensory configuration for AVs. Prior maps, such as Google™ Maps or OpenStreetMaps™[125], are used as a part of the inputs. We have the following assumptions,

a.1 The camera is pre-calibrated, and the nonlinear distortion of images has been removed.

a.2 All sensor readings are synchronized.

a.3 The coordinate system transformations between any two sensors are known by prior calibration.

All coordinate systems are right hand system and common notations are defined as follows,

- $\{\mathscr{L}\}$ defines the lidar coordinate system with x-axis pointing in the vehicle forward direction, y-axis pointing to the left, and z-axis pointing upward. $\mathbf{P}_{i,t} = [x_{i,t}, y_{i,t}, z_{i,t}]^\intercal \in \mathscr{R}^3$ is the i-th 3D lidar point with respect to $\{\mathscr{L}\}$ at time $t \in \{0, 1, ..., T\}$, and $\mathscr{P}_t := \{\mathbf{P}_{i,t}\}$ is the set of lidar points at time t.

- $\{\mathscr{C}\}$ defines the camera coordinate system with x-axis pointing to the right of the vehicle lateral direction and z-axis pointing forward coinciding with the front-view camera's principal axis.

- $\{\mathscr{I}\}$ defines image coordinate system. Let $\mathbf{p}_{k,t} = [u\ v]^\intercal \in \{\mathscr{I}\}$ be the k-th pixel point in image $\mathbf{I}_t$ at time t, where (u, v) is the image coordinate.

- $\{\mathscr{W}\}$ defines the world coordinate system which overlaps with $\{\mathscr{L}\}$ at the vehicle starting position.

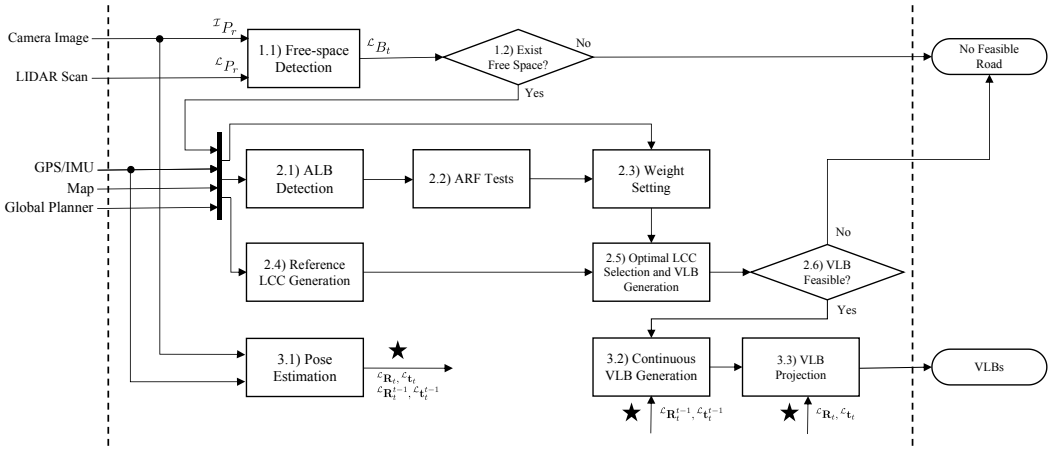


Figure 5.2: System diagram. The solid star represents the output of pose estimation, which is also the input to the continuous LB generation and LB projection.

Denote the left and right LBs in $\{\mathscr{W}\}$ by ${}^{\mathscr{W}}\mathbf{L}_l$ and ${}^{\mathscr{W}}\mathbf{L}_r$ at time t, respectively. Note that left superscript in this paper describes the coordinate system for the corresponding variable. With the assumptions and notations defined, our problem is defined as follows,

Problem 3. Given a prior map, current GPS position, and *in situ* camera and lidar inputs, and velocity profile and global route from a global planner, recognize, generate and track LBs ${}^{\mathscr{W}}\mathbf{L}_l$ and ${}^{\mathscr{W}}\mathbf{L}_r$ in $\{\mathscr{W}\}$, or report when the VLBs cannot be generated.

5.4 Algorithm

Fig. 5.2 shows the system diagram. It mainly contains the following blocks: A) Free-space detection, B) VLB generation where we perform ALB detection and also determine how we should generate VLBs, and C) VLB registration where we track the LBs through an extended Kalman filter (EKF) and re-project VLBs in $\{\mathscr{W}\}$. We start with the free-space detection.

5.4.1 Free-space Detection

The free-space is collision free surface in front of the vehicle which can be defined by road edges and obstacle boundaries. We detect free space in both camera and lidar

modalities and extract free-space surface boundary in $\{\mathscr{L}\}$ (see Box 1.1 in Fig. 5.2).

We start with recognizing road surface in both image and lidar data based on our prior work [94] where we have employed camera-lidar fusion to obtain road surface pixel set ${}^{\mathscr{I}}P_r$ in image coordinate $\{\mathscr{I}\}$ using the appearance classification. We also have the corresponding 3D point lidar point cloud set ${}^{\mathscr{L}}P_r \subset \mathscr{P}_t$ for ${}^{\mathscr{I}}P_r$. For each point $\mathbf{p}_r \in {}^{\mathscr{I}}P_r$ and its corresponding lidar point $\mathbf{P}_r \in {}^{\mathscr{L}}P_r$, we have the projection relationships between them, $\tilde{\mathbf{p}}_r = c_p\mathbf{K}[{}^{\mathscr{C}}_{\mathscr{L}}\mathbf{R}\ {}^{\mathscr{C}}_{\mathscr{L}}\mathbf{t}]\tilde{\mathbf{P}}_r$ and $\mathbf{P}_r = c_q[\mathbf{K}{}^{\mathscr{C}}_{\mathscr{L}}\mathbf{R}]^{-1}\tilde{\mathbf{p}}_r - {}^{\mathscr{C}}_{\mathscr{L}}\mathbf{R}^{-1}{}^{\mathscr{C}}_{\mathscr{L}}\mathbf{t}$, where c_p and c_q are scalars, a vector with symbol '~' on top is in its homogeneous representation, $\mathbf{K}$ is the intrinsic camera matrix under the pin hole model, and ${}^{\mathscr{C}}_{\mathscr{L}}\mathbf{R}$ and ${}^{\mathscr{C}}_{\mathscr{L}}\mathbf{t}$ are the rotation matrix and translation vector between $\{\mathscr{L}\}$ and $\{\mathscr{C}\}$, respectively. We also use two more inputs from [94]: the road surface model with coefficient vector $\mathbf{H}_r^*$ which is acquired by fitting points in ${}^{\mathscr{L}}P_r$ to a polynomial model, and $d_\perp(\mathbf{H}_r^*, \mathbf{P}_{i,t})$ which is the shortest distance for a point $\mathbf{P}_{i,t}$ to the road surface.

Building on these prior results, we design a two-step approach to obtain 3D free-space surface boundary points. 1) We only keep lidar points ${}^{\mathscr{L}}C_t$ with small elevation difference to the surface model, ${}^{\mathscr{L}}C_t = \{\mathbf{P}_{i,t} |\ c_l \le d_\perp(\mathbf{H}_r^*, \mathbf{P}_{i,t}) \le c_u, \mathbf{P}_{i,t} \in \mathscr{P}_t\}$, where c_l and c_u are thresholds. 2) We compute the average surface normal of each pixel's neighbor set and use it to determine if it is on the smooth surface. Let us detail the second step here.

For each point $\mathbf{P}_{i,t} \in {}^{\mathscr{L}}C_t$, we can find its neighbor set ${}^{\mathscr{L}}E_i$ by selecting the K-nearest neighbors (KNNs) [126] with an upper bound d_r. ${}^{\mathscr{L}}E_i = \{\mathbf{P}_{j,t} |\ \|\mathbf{P}_{i,t} - \mathbf{P}_{j,t}\| \le d_r, \mathbf{P}_{j,t} \in {}^{\mathscr{L}}C_t\}$, where index variable $j \in \mathscr{N}$ satisfies $j \ne i$ and $1 \le j \le K$. Next we apply methods in [127] to extract surface normal for the neighbor set to determine if $\mathbf{P}_{i,t}$ is a smooth road point. Define $\hat{\mathbf{C}}_{i,t} = \frac{1}{|{}^{\mathscr{L}}E_i|}\sum_{\mathbf{P}_{j,t} \in {}^{\mathscr{L}}E_i} \mathbf{P}_{j,t}$ to be the 3D centroid of ${}^{\mathscr{L}}E_i$, and

$$d_e(\mathbf{P}_{i,t}) = \frac{1}{|{}^{\mathscr{L}}E_i|} \sum_{\mathbf{P}_{j,t} \in {}^{\mathscr{L}}E_i} \frac{\|\mathbf{P}_{i,t} - \mathbf{P}_{j,t}\|}{\|\mathbf{P}_{i,t}\|}, \tag{5.1}$$

to be the normalized average distance for all points in ${}^{\mathscr{L}}E_i$ to $\mathbf{P}_{i,t}$. Define $d_s(\mathbf{P}_{i,t},\mathbf{P}_{j,t}) = \|\mathbf{P}_{j,t} - \hat{\mathbf{C}}_{i,t}\|/\|\mathbf{P}_{i,t}\|$ as the normalized distance for the point $\mathbf{P}_{j,t}$ to remove scale effect. Define a weight value $w_{j,t}$ for the $\mathbf{P}_{j,t}$ to be

$$w_{j,t} = \begin{cases} \exp^{-\frac{d_s(\mathbf{P}_{i,t},\mathbf{P}_{j,t})^2}{d_e(\mathbf{P}_{i,t})^2}}, & \text{if } d_s(\mathbf{P}_{i,t},\mathbf{P}_{j,t}) \geq d_e(\mathbf{P}_{i,t}), \\ 1, & \text{otherwise.} \end{cases}$$

Let λ_1, λ_2 and λ_3 be the eigenvalues of the correlation matrix $\sum_{j=1}^{|{}^{\mathscr{L}}E_i|} w_{j,t}(\mathbf{P}_{j,t} - \hat{\mathbf{C}}_{i,t})(\mathbf{P}_{j,t} - \hat{\mathbf{C}}_{i,t})^\intercal$, and suppose $\lambda_1 \leq \lambda_2 \leq \lambda_3$. According to [128], a point on smooth road surface has λ_1 significantly smaller than the other two; for a free-space surface boundary point, λ_1 and λ_2 are substantially smaller than λ_3. Therefore, we can use this property to obtain boundary point set ${}^{\mathscr{L}}F_t$ by thresholding,

$${}^{\mathscr{L}}F_t = \left\{\mathbf{P}_{i,t} | \lambda_3 / \sum_{i=1}^{3} \lambda_i \geq \lambda_d, \mathbf{P}_{i,t} \in {}^{\mathscr{L}}C_t\right\}, \tag{5.2}$$

where λ_d is the threshold. Inspired by [129], we can further remove noisy points in set ${}^{\mathscr{L}}F_t$ by examining surface normal vector directions. For a point $\mathbf{P}_{i,t} \in {}^{\mathscr{L}}F_t$, we compute the average surface normal $\theta_{i,t}$ as follows, $\theta_{i,t} = \frac{1}{|{}^{\mathscr{L}}E_i|}\sum_{j=1}^{|{}^{\mathscr{L}}E_i|} \arctan \frac{|z_{i,t} - z_{j,t}|}{\sqrt{|x_{i,t} - x_{j,t}|^2 + |y_{i,t} - y_{j,t}|^2}}$. Note that points on the road surface have small variations in z direction which means small $\theta_{i,t}$ values. Therefore, we can identify boundary/obstacle points by thresholding on $\theta_{i,t}$ and obtain free-space surface boundary point set ${}^{\mathscr{L}}B_t$ as follows,

$${}^{\mathscr{L}}B_t = \{\mathbf{P}_{i,t} | \theta_{i,t} \geq \theta_v, \mathbf{P}_{i,t} \in {}^{\mathscr{L}}F_t\}, \tag{5.3}$$

where θ_v is the threshold.

Next, we need to verify if there is available free space in front of the vehicle given

vehicle kinodynamic and size constraints. We apply a state lattice planner [8] to generate a set of seven candidate arc trajectories $\{^{\mathscr{L}}\mathbf{L}_{l_p}\}_{l_p=1}^{7}$ that evenly cover curvatures in the allowable range given the current speed. The length of arc is the braking distance. Let d_v be the haft width of the vehicle. We evaluate all points in the region swiped by the vehicle if following the arc $^{\mathscr{L}}\mathbf{L}_{l_p}$ which is set $^{\mathscr{L}}P_{l_p} = \{\mathbf{P}_{i,t} | \min \|\mathbf{P}_{i,t} - \mathbf{P}_w\| \leq d_v, \mathbf{P}_{i,t} \in \mathscr{P}_t, \mathbf{P}_w \in {}^{\mathscr{L}}\mathbf{L}_{l_p}\}$. Denote the logic OR operator by $\bigvee$. If we have,

$$\bigvee_{l_p} \{ {}^{\mathscr{L}}P_{l_p} \bigcap {}^{\mathscr{L}}B_t = \emptyset \} = 1, \tag{5.4}$$

then the free space exists (see decision box 1.2 in Fig. 5.2) and we move on to next step. Otherwise, there is no feasible road and global planner needs to be notified to re-plan route. The global planner concerns overall routing and is not the concern of this paper.

5.4.2 VLB Generation

VLBs and corresponding lane center curves (LCCs) regulate how the vehicle can move. Generating them is equivalent to local planning but with tight connecting to perception and vehicle kinodynamic constraints. By tight connecting we mean that LCCs and VLBs are evaluated directly and locally in the sensor space without an additional world model. We have to answer two important questions here: 1) when should we decide to deviate from ALBs? and 2) how to generate VLBs to balance multiple requirements to be human compatible?

5.4.2.1 LB representation

Before we dive into details, let us define LCC and the information obtained from ALB as shown in our prior work [94]. In fact, it is also possible to use lane detection methods from other existing works. From [94], we obtain ALB and the corresponding LCC $^{\mathscr{L}}\mathbf{L}_a$ is

represented as cubic B-spline curves that are made of l piecewise polynomial functions,

$$^{\mathscr{L}}\mathbf{L}_{a,l}(s) = \mathbf{a}_{l,0} + \mathbf{a}_{l,1}s + \mathbf{a}_{l,2}s^2 + \mathbf{a}_{l,3}s^3 \tag{5.5}$$

be the l-th curve segment where $\{\mathbf{a}_{l,j} | l = 1,2,...,n_c - 3, j = 0,1,2,3\}$ are 3-vectors for polynomial coefficients, $0 \leq s \leq s_e$, n_c is the number of the control points for the spline curve, and $s_e = n_c + 3$ is the maximum knots. Subscript a indicates this LCC is from ALB. As shown in [94], for a given LCC and a lane width, it is trivial to obtain the left and right LBs $^{\mathscr{L}}\mathbf{L}_l(s)$ and $^{\mathscr{L}}\mathbf{L}_r(s)$, respectively, and vice versa.

5.4.2.2 Examining ALB quality

For question 1), we determine if the vehicle should follow ALBs using *availability, reasonability, and feasibility* (ARF) tests (see Box 2.2 in Fig. 5.2). For *availability*, we examine if ALBs provide a sufficiently long trajectory to follow.

$$\int_0^{s_e} \|^{\mathscr{L}}\mathbf{L}_a^{'}(s)\| ds \geq l_{\min}, \tag{5.6}$$

where $\|\cdot\|$ is the vector l^2-norm and $l_{\min}$ is the trajectory length threshold.

For *reasonability*, we check if the LCC $^{\mathscr{L}}\mathbf{L}_a$ heading agrees with the vehicle's current heading. Let $\mathbf{n}_v \in \mathscr{R}^3$ point to the vehicle's driving direction at time t, and $\mathbf{n}_u \in \mathscr{R}^3$ be the first derivative of the LCC $^{\mathscr{L}}\mathbf{L}_a(s)$ when $s = 0$, respectively. Let $\langle \cdot, \cdot \rangle$ represent the inner product between two vectors. For a threshold $\beta_l = 10^\circ$, if

$$\arccos \frac{\langle \mathbf{n}_v, \mathbf{n}_u \rangle}{\|\mathbf{n}_v\| \|\mathbf{n}_u\|} \leq \beta_l, \tag{5.7}$$

then the current LCC $^{\mathscr{L}}\mathbf{L}_a$ is reasonable.

For *feasibility*, we want to make sure that the curvature of the LCC is compatible

with the current vehicle speed. We precompute a look-up table offline considering the vehicle speed and the curvature. Let $\langle \cdot \times \cdot \rangle$ represent vector cross product. Let $\kappa_{\max}$ be the maximum allowable LCC curvature for the vehicle given the current forward speed v_t. We have a feasible LCC if

$$\frac{\|\langle {}^{\mathscr{L}}\mathbf{L}_a'(s) \times {}^{\mathscr{L}}\mathbf{L}_a''(s)\rangle\|}{\|{}^{\mathscr{L}}\mathbf{L}_a'(s)\|^3} \leq \kappa_{\max}. \tag{5.8}$$

ARF test results are used to set weights in selecting LCCs for VLB and will be detailed later in Section 5.4.2.4.

5.4.2.3 VLB generation

For question 2), to generate human-compatible VLBs, we need to a) respect partial information from ALB, b) follow GPS waypoints, c) avoid dynamic and stationary obstacles, and d) consider vehicle kinodynamic constraints.

Therefore, we need the planned GPS trajectory as a seed. From the current GPS reading and the prior map, we can extract a set of GPS way points to represent the road ahead. The number of points depends on the velocity of the vehicle and the minimum number needed to construct a cubic B-spline representation. We can project these 2D map points onto the road surface model to obtain 3D points. Applying cubic B-spline fitting and coordinate transformation, we obtain its representation in current lidar coordinates to be ${}^{\mathscr{L}}\mathbf{L}_g(s)$ where subscript g means this is from GPS reference. Note that LCC of VLBs should start with the endpoint of previous LCC (denoted by ${}^{\mathscr{L}}\mathbf{L}^-(s)$) at time $t-1$ which happens when $s = s_e$. ${}^{\mathscr{L}}\mathbf{L}_g(s)$ and ${}^{\mathscr{L}}\mathbf{L}^-$ do not necessarily overlap. A minimum distance parallel shift of ${}^{\mathscr{L}}\mathbf{L}_g(s)$ allows point ${}^{\mathscr{L}}\mathbf{L}^-(s_e)$ be located on the shifted ${}^{\mathscr{L}}\mathbf{L}_g(s)$. The shifted ${}^{\mathscr{L}}\mathbf{L}_g(s)$ is cropped to start at the point and serve as the seed trajectory for candidate trajectory generation. In fact, the shifted ${}^{\mathscr{L}}\mathbf{L}_g(s)$ does not need to be collision free. The new trajectory along with velocity profile and vehicle size are then used to generate candidate trajectories by sampling on lattice using [8], which provide us a set of candidate

LCCs ${}^{\mathscr{L}}\mathbf{L} \subset \mathscr{L}_c$ considering the vehicle's kinodynamic constraints. Of course, any candidate LCC ${}^{\mathscr{L}}\mathbf{L}$ also have to pass our ARF tests. If none of the candidate LCC pass ARF tests, the system reports "no feasible road" to the global planner.

We then select the best candidate LCC by minimizing a cost function $C({}^{\mathscr{L}}\mathbf{L})$ (see Box 2.5 in Fig. 5.2)

$$ {}^{\mathscr{L}}\mathbf{L}^{\star} = \underset{{}^{\mathscr{L}}\mathbf{L} \subset \mathscr{L}_c}{\arg\min}\, C({}^{\mathscr{L}}\mathbf{L}), \tag{5.9} $$

that is designed to consider human compatibility by integrating smoothness f_s, obstacle avoidance f_o, GPS trajectory following f_g, trajectory continuity f_c, and ALBs f_a as follows

$$ \begin{aligned} C({}^{\mathscr{L}}\mathbf{L}) = f_s({}^{\mathscr{L}}\mathbf{L}) + w_2 f_o({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}B_t) + w_3 f_g({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_g) \\ + w_4 f_c({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}^{-}) + w_5 f_a({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_a), \end{aligned} \tag{5.10} $$

where $w_2, .., w_5$ are non-negative weighting variables.

Function $f_s({}^{\mathscr{L}}\mathbf{L})$ controls the smoothness of the LCC [130],

$$ f_s({}^{\mathscr{L}}\mathbf{L}) = \int_0^{s_e} \|{}^{\mathscr{L}}\mathbf{L}'(s)\|^2 ds + w_1 \int_0^{s_e} \|{}^{\mathscr{L}}\mathbf{L}''(s)\|^2 ds, \tag{5.11} $$

where w_1 is a non-negative weight variable, $[0, s_e]$ define spline parameter range for the LCC.

Function $f_o({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}B_t)$ is the cost related to the clearance to boundary/obstacle ${}^{\mathscr{L}}B_t$ set in (5.3). Let $d_o^* = \min_{\mathbf{P}_{i,t} \in {}^{\mathscr{L}}B_t, 0 \le s \le s_e} \|{}^{\mathscr{L}}\mathbf{L}(s) - \mathbf{P}_{i,t}\|$, be the shortest distance between a

candidate LCC and a road edge point $\mathbf{P}_{i,t} \in {}^{\mathscr{L}}B_t$, we have

$$f_o({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}B_t) = \begin{cases} 0 & \text{if } d_o^* \geq d_r, \\ \frac{c_b}{d_l - d_r}(d_o^* - d_r) & \text{if } d_l < d_o^* < d_r, \\ \infty & \text{otherwise.} \end{cases} \tag{5.12}$$

where c_b is linear cost coefficient for distance to obstacle, d_l and d_r define the distance interval where the linear cost function is applied.

Cost function $f_g({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_g)$ wants the output trajectory to be similar to that of the GPS trajectory,

$$f_g({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_g) = \int_0^{s_e} \|{}^{\mathscr{L}}\mathbf{L} - {}^{\mathscr{L}}\mathbf{L}_g\|^2 ds. \tag{5.13}$$

Cost function $f_c({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}^-)$ maintains continuity of LCC from prior period ${}^{\mathscr{L}}\mathbf{L}^-$ (noting it has been transformed to current $\{\mathscr{L}\}$ coordinate),

$$f_c({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}^-) = \int_0^{s_e} \|{}^{\mathscr{L}}\mathbf{L} - {}^{\mathscr{L}}\mathbf{L}^-\|^2 ds. \tag{5.14}$$

This cost function helps avoid sudden motion and makes the LCC more compatible with human drivers.

Cost function $f_a({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_a)$ regulates the LCC to be close to ALBs,

$$f_a({}^{\mathscr{L}}\mathbf{L}, {}^{\mathscr{L}}\mathbf{L}_a) = \int_0^{s_e} \|{}^{\mathscr{L}}\mathbf{L} - {}^{\mathscr{L}}\mathbf{L}_a\|^2 ds. \tag{5.15}$$

This function regulates LCC to follow ALBs as much as possible which makes LCC to meet human expectation better.

5.4.2.4 *Weight settings*

Non-negative weighting variables w_1, w_2, w_3, w_4, and w_5 play an important role in regulating the LCC. This is done before LCC generation (see Box 2.3 in Fig. 5.2).

w_1 and w_4 control the smoothness of the resulting LCC. They should be an increasing function of velocity due to vehicle kinodynamic constraints. They are also related to driving status. If the vehicle decides to make a turn or switch lanes as instructed by the global planner, then we set them to be zero since we do not need to follow the previous direction.

w_2 controls how conservative the vehicle should be in obstacle avoidance. It should be a function of the relative velocity to obstacles. For example, the existence of a cyclist demands higher w_2 settings.

w_3 controls GPS following quality. It should be determined by how good the prior map quality is. If the road is under construction and the prior map has not been updated, then we should reduce w_3 to allow more deviation from the original map.

w_5 is adjusted according to ARF test results. If ALBs do not exist or are not reasonable (i.e. fail the first two tests of ARF tests), we set w_5 to be 0 because there is no trustable ${}^{\mathscr{L}}\mathbf{L}_a$. However, w_5 remains positive if ALBs are infeasible due to vehicle kinodynamic constraints. If ALBs pass all ARF tests, then w_5 is set to the highest value to ensure good lane following.

5.4.3 VLB Registration

So far, we have obtained LCCs which are computed in local lidar coordinates and are piece-wise polynomials over time. To ensure a smooth and continuous trajectory in $\{\mathscr{W}\}$, we apply an EKF to track VLBs to generate and register continuous curves (see Boxes 3.2 and 3.3 in Fig. 5.2).

We need the coordinate transformation from time $t-1$ to t. This can be obtained using the optimization-based multi-sensor state estimator [131] (see Box 3.1 in Fig. 5.2)).

Denote the rotation matrix and translation vector with respect to $\{\mathscr{W}\}$ at time t by ${}^{\mathscr{L}}\mathbf{R}_t$ and ${}^{\mathscr{L}}\mathbf{t}_t$, respectively. Let ${}^{\mathscr{L}}\mathbf{R}_{t-1}^{t}$ and ${}^{\mathscr{L}}\mathbf{t}_{t-1}^{t}$ be the relative rotation matrix and translation vector from $t-1$ to t, respectively. We have ${}^{\mathscr{L}}\mathbf{R}_{t-1}^{t} = {}^{\mathscr{L}}\mathbf{R}_{t-1}{}^{\mathscr{L}}\mathbf{R}_{t}^{-1}$ and ${}^{\mathscr{L}}\mathbf{t}_{t-1}^{t} = {}^{\mathscr{L}}\mathbf{t}_{t-1} - {}^{\mathscr{L}}\mathbf{R}_{t-1}^{t}{}^{\mathscr{L}}\mathbf{t}_{t}$. Recall the cubic B-spline curves are made of piecewise polynomial functions, and each polynomial function needs four control points to satisfy its continuity properties. We sort the control points for the optimal LCC ${}^{\mathscr{L}}\mathbf{L}^{\star}$ according to the increasing order of their distance to the origin of $\{\mathscr{L}\}$. let ${}^{\mathscr{L}}P_{\star} = \{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_{n_c}\}$ be the control point set for the ${}^{\mathscr{L}}\mathbf{L}^{\star}$. Here, $\|\mathbf{P}_p\| < \|\mathbf{P}_q\|$ for $\mathbf{P}_p, \mathbf{P}_q \in {}^{\mathscr{L}}P_{\star}$ if $p < q$. Let

$$\mathbf{z}_{p,t} = [\mathbf{P}_p^{\intercal}, \mathbf{P}_{p+1}^{\intercal}, \mathbf{P}_{p+2}^{\intercal}, \mathbf{P}_{p+3}^{\intercal}]^{\intercal}, \tag{5.16}$$

be the observations for the LCC for $p = 1, 2, ..., n_c - 3$. Define a state vector

$$\mathbf{x}_{p,t-1} = [\mathbf{a}_{p,3}^{\intercal}, \mathbf{a}_{p,2}^{\intercal}, \mathbf{a}_{p,1}^{\intercal}, \mathbf{a}_{p,0}^{\intercal}]^{\intercal}, \tag{5.17}$$

through (5.5) for the LCC at time $t-1$. The state transition function is just the coordinate system transformation between adjacent time epochs,

$$\mathbf{x}_{p,t} = \mathscr{R}_{t-1}^{t}\mathbf{x}_{p,t-1} + \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ {}^{\mathscr{L}}\mathbf{t}_{t-1}^{t} \end{bmatrix} + w_t, \tag{5.18}$$

where w_t has zero mean and covariance $\mathbf{Q}_t$, and $\mathscr{R}_{t-1}^{t} = \text{diag}({}^{\mathscr{L}}\mathbf{R}_{t-1}^{t}, {}^{\mathscr{L}}\mathbf{R}_{t-1}^{t}, {}^{\mathscr{L}}\mathbf{R}_{t-1}^{t}, {}^{\mathscr{L}}\mathbf{R}_{t-1}^{t})$

is a diagonal block matrix. According to [132], the observation function is

$$\mathbf{z}_{p,t} = \begin{bmatrix} \mathbf{0}_{3\times3} & \frac{2}{3}\mathbf{I} & -\mathbf{I} & \mathbf{I} \\ \mathbf{0}_{3\times3} & -\frac{1}{3}\mathbf{I} & \mathbf{0}_{3\times3} & \mathbf{I} \\ \mathbf{0}_{3\times3} & \frac{2}{3}\mathbf{I} & \mathbf{I} & \mathbf{I} \\ 6\mathbf{I} & \frac{11}{3}\mathbf{I} & 2\mathbf{I} & \mathbf{I} \end{bmatrix} \mathbf{x}_{p,t} + m_t, \tag{5.19}$$

where m_t is zero mean and has the covariance $\blacksquare_t$, and $\mathbf{I}$ is a 3×3 identity matrix. We continuously predict and update the EKF as more data comes in, and register the optimal LCC by

$$^{\mathscr{W}}\mathbf{L}^{\star}(s) = {}^{\mathscr{L}}\mathbf{R}_t^{\intercal}\,{}^{\mathscr{L}}\mathbf{L}^{\star}(s) - {}^{\mathscr{L}}\mathbf{t}_t, \tag{5.20}$$

from $\{\mathscr{L}\}$ to $\{\mathscr{W}\}$. We also apply (5.20) to obtain the left VLB $^{\mathscr{W}}\mathbf{L}_l(s)$ and the right VLB $^{\mathscr{W}}\mathbf{L}_r(s)$, respectively.

The LCC $^{\mathscr{W}}\mathbf{L}^{\star}(s)$ is tracked by the vehicle controller, which generates corresponding throttle, brake and steering commands to determine the vehicle status from time t to $t+1$. Typically, the vehicle state can be represented as a high-dimensional Gaussian distribution. Considering the updated vehicle states, our methods continuously generate new LCC to lead the navigation of the vehicle with incoming sensor measurements.

5.5 Experiments

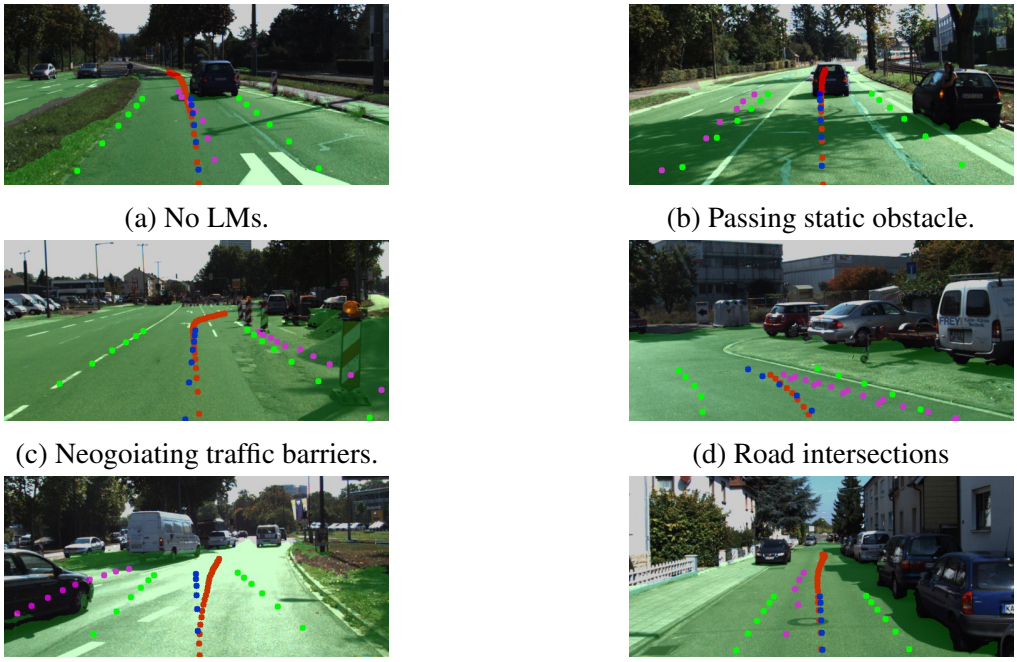
We have implemented the proposed method in C++ and shared it on Github™[2]. It is tested on a Laptop PC with an Intel® Core™i7-3517U CPU@1.90GHz and 8 GB RAM. We test our approach using the KITTI dataset [133], which contains images covering a variety of street scenes captured from a vehicle driving around the city of Karlsruhe.

We have tested our algorithm on four different sequences of two categories from KITTI dataset including city and residential area (see Tab. 5.1). In all cases, our algorithm can

[2]https://github.com/bli-tamu/LDRT

Table 5.1: %VLBs on KITTI Dataset

Sequence	Duration	length	% ($w_5 = 0$)
2011_09_26_drive_0035	13 s	60.41 m	100%
2011_09_26_drive_0039	40 s	297.09 m	100%
2011_09_26_drive_0051	44 s	255.42 m	92%
2011_09_26_drive_0056	30 s	419.95 m	29%



(a) No LMs.

(b) Passing static obstacle.

(c) Neogoiating traffic barriers.

(d) Road intersections

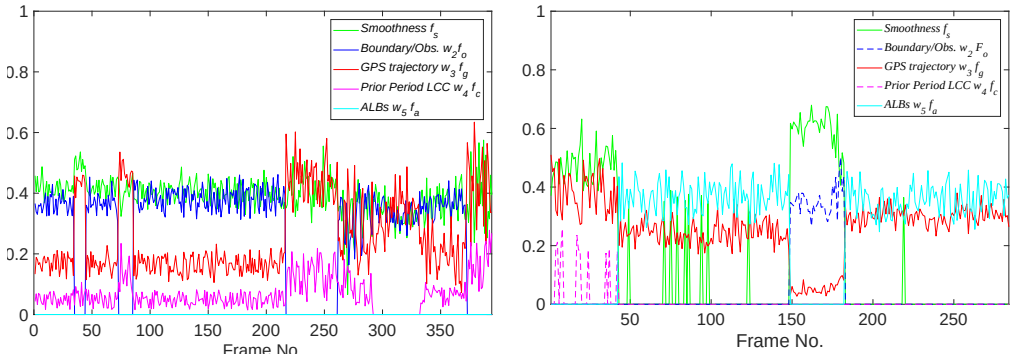
(e) Merging w. dynamic obstacles.

(f) Parked cars and no LMs.

Figure 5.3: Sample algorithm outputs for six different scenarios.

generate feasiable LCC and VLBs to guide the vehicle. In last column, we track the ratio when $w_5 = 0$ because it indicates that the vehicle decides to deviate from ALBs. The ratio varies from from 29% to 100% due to different road scenarios. Some road segments have great ALBs and do not need VLB generation as much (e.g. the fourth row) while some roads do not have ALBs at all (e.g. the first two rows).

Sample outputs are shown in Fig. 5.3. The green masked area is the free space detected by the algorithm. Four different dotted lines are drawn on the six figures: purple



(a) 2011_09_26_drive_0039 (b) 2011_09_26_drive_0056

Figure 5.4: Contribution to LCC cost by different components.

lines represent GPS way points from Google maps, blue lines are the algorithm output LCCs ${}^{\mathscr{W}}\mathbf{L}^{\star}(s)$, red lines are the high precision GPS recording of actual human driving the vehicle which can be viewed as the human decision counterpart, and two green lines are the left LB ${}^{\mathscr{W}}\mathbf{L}_l(s)$ and the right LB ${}^{\mathscr{W}}\mathbf{L}_r(s)$, respectively. It is clear that way points from Google maps are too lousy to be used as direct navigation guides, as indicated by the poor quality of purple lines. When comparing our algorithm outputs to the GPS recording of the human driving, blue lines are quite in agreement with red lines with the only exception in Fig. 5.3e. Note that red lines extend beyond blue lines due to different trajectory length which does not mean that they do not agree. Even in Fig. 5.3e, both the blue line and the red line are viable choices. In all cases, our algorithm can generate LCCs that are compatible with human expectations.

Fig. 5.4 further illustrates how different components contribute to the VLB LCC selection in (5.10) using the second and fourth sequence in Tab. 5.1. The plots are the normalized ratios in the overall objective function value. During the computation, the weight settings for the optimization problem are set as $w_1 = 1$, $w_2 = 1$, $w_3 = 2$, $w_4 = 0.2$ and $w_5 = 5$ for non-zero cases to balance the multiple objectives in the LCC selection. It is

clear that every component in (5.10) plays a role in determining LCC.

The more interesting part is the dynamic change of ratios, as shown by Fig. 5.4a which really exposes the inner-works of VLB generation. First, there are no ALBs in the entire sequence and w_5 has to be zero during 100% of the time. Second, both f_s (green solid line) and $w_2 f_o$ (blue dashed line) are relatively high throughout the entire sequence because it is important to avoid obstacles and maintain smooth motion during the driving. A close look reveals that there are four sudden drops for $w_2 f_o$. Two short segments are located at frames 34–85, one long segment appears at frames 216–254 and the last one is at frames 374–394. These are due to the fact that there are no obstacles at the time and the road is empty. Consequently, the vehicle relies more on GPS trajectory following and we can see that ratio of $w_3 f_g$ increases. It means that the algorithm automatically falls back to rely on other available information when there are no LMs and no obstacles, which is desirable. The $w_4 f_c$ usually has a segment of being zero at frames 293–334 because the vehicle is make a 90° turn and actively set w_4 to be zero. Similar scenario happens at the beginning of Fig. 5.4b. The sequence in Fig. 5.4b has high quality ALBs mostly and only needs to rely on VLBs 29% of the time. It is clear that $w_5 f_a$ remains high at frames 44–149 and 184–293 where the AV relies a lot on the ALB following. In addition, the reason that we have a segment of VLBs at frames 150-183 is due to a parked vehicle occupying part of the road which is shown in Fig. 5.3b.

5.6 Conclusion

We reported our development of a new tightly connected perception and planning framework to enable AVs to consider multiple conflicting goals simultaneously and generate human-compatible navigation trajectories. We built on our prior work to detect free space using camera-lidar sensor fusion and proposed ARF tests to determine whether the AV should simply follow ALBs or generate VLBs by taking into account vehicle kino-

dynamic constraints, obstacle avoidance, smooth motion, GPS trajectory following, respecting direction of LMs in a multi-objective optimization framework with dynamically adjustable weights for different road scenarios. We implemented our algorithm and the test results confirmed our design.

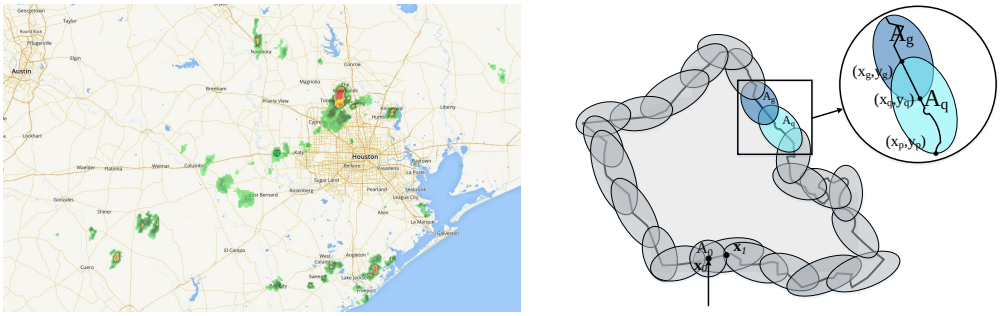
6. BELIEF SPACE-BASED APPROACH OF PROBABILISTIC BOUNDARY COVERAGE FOR UNKNOWN TARGET FIELDS[1]

6.1 Introduction

We propose to use belief space for probabilistic boundary coverage of UTFs. Imagine that an unmanned aerial vehicle (UAV) is dispatched to map boundaries of excessive wind shear or low pressure regions in storm cells (see Fig. 6.1). The UAV has to plan its motion based on its on-board sensor readings to quickly enclose the UTF, which is a form of boundary coverage problem. However, UTFs often do not have a clear boundary or a known dispersion function and the UAV has to get closer to the field to take multiple readings to predict field dispersion for boundary coverage. Moreover, the sensor readings often contain large uncertainties due to variations of the field itself or difficult sensing conditions. It is clear that regular boundary traversing techniques are not applicable. Such problems are not unusual. Another example is that an inspection robot is tasked to find thin hairline cracks on the airport runway. These applications propose a new problem: how can we design a principled approach to ensure the robot can effectively cage UTFs under large perception uncertainty and limited sensing range.

We present this new boundary coverage problem and propose an algorithm to solve the problem. At each step, the robot accumulates sufficient sensory readings to instantiate Gaussian processes (GPs) as a local belief function to approximate field dispersion in an ellipse-shaped local region. The local belief function allows us to predict UTF boundary trends and establish adjacent ellipses for further exploration in next step. The process is governed by a depth-first search process until UTF is enclosed by connected ellipses (see

[1]Reprinted with permission from "Probabilistic Boundary Coverage for Unknown Target Fields with Large Perception Uncertainty and Limited Sensing Range" by B.Li and D. Song, Robotics Research. Springer, Cham, 2020. 711-726.



(a) Wind shear region boundary coverage application.

(b) Output of our boundary coverage algorithm.

Figure 6.1: Problem illustration. a) Wind shear region boundary coverage application: The green & orange clouds represent potential regions of interest that may contain UTFs. To map each UTF, we need to send an UAV to cage the UTF. b) Output of our boundary coverage algorithm is to cover the boundary using a sequence of connected ellipses.

Fig. 6.1b) with probability guarantees, as we formally prove that our boundary coverage process guarantees that the enclosed UTF is above a given coverage ratio with a preset probability threshold. We have implemented our method and tested with different types of UTFs (1D vs. 2D, smooth vs. non-smooth boundary, and convex vs. non-convex) in simulation. The results show that the algorithm always guarantees that the coverage ratio is above the given threshold for all testing cases, which is conformable to our analysis.

6.2 Related Work

The first part of our work relates to coverage in continuous fields, discrete space search, and robotic caging in manipulation and grasping.

Boundary coverage of UTFs is related to well-studied coverage problems because the latter also need to identify target field boundary before planning for coverage. However, identifying the boundary might not be an issue if target field information is known whereas our problems focus on identifying boundaries. In coverage problems, the focus is to calculate optimal trajectories with respect to a given objective function for known field func-

tions. Most existing methods depend on gradients or other information from the known field functions. Yun et al. [134] present decentralized algorithms for the coverage with mobile robots on a graph. Miller et al. [135] use an ergodic control algorithm for the coverage with respect to the expected information density. Shnaps et al. [136] perform online tethered coverage in planar unknown environments using position and local obstacle detection sensors. Bekris et al. [137] apply cloud computing to efficiently plan the motion of new robot manipulators designed for flexible manufacturing floors. In our boundary coverage problem, we have to approximate field functions based on noisy and local/partial observations before planning for robot motion which further complicates the problem.

Searching for point/small objects without field functions can be viewed as a discrete search problem. The original search space could be either continuous or discrete but it is often discretized into grids or graphs in the searching process [138]. Acar et al. [139] introduce a hierarchical decomposition that combines the Morse decompositions and the generalized Voronoi diagram to ensure that the robot covers the searching domain. Paull et al. [140] present a sensor driven on-line approach for seabed coverage for mine countermeasure using grid-based coverage. Xu et al. [141] address the problem of effective graph coverage with environmental constraints and incomplete prior map information. Mannadiar et al. [142] guarantee the complete coverage of the free space based on the Boustrophedon cellular decomposition. Although these methods can be applied to UTF searching problem, their approaches are low efficient because the existing methods do not exploit the continuity of UTF structure.

Boundary coverage of UTFs is related to the caging problem in grasping which focuses on using geometric information of the manipulated object to generate stable grasps [143, 144, 145, 146, 147]. Vongmasa et al. [148] compute coverage parameters for 2D polygons to form a cage to transport an object. Pereira et al. [149] enable a team of robots with limited sensing range to achieve a condition of object closure, and move toward a

goal position while maintaining the object closure condition. Ivan et al. [150] compute coverage without the reliance to geometrical detail but capture the topology of punctured euclidean spaces. Zarubin et al. [151] use geodesic balls to estimate object's surface in the presence of noise. These methods compute the waypoints for the object to generate a set of caging grasps. One issue that separates UTF caging from object caging is the issue of limited sensing range because objects are often small and fully covered by the sensor in the grasping process.

6.3 Problem Definition

6.3.1 Scenario and Assumptions

A mobile robot or UAV is dispatched to find an efficient path to enclose UTFs in an obstacle-free 2D space. This robot is equipped of sensors with a limited sensing range. The robot observation noise follows a Gaussian distribution with zero mean and variance σ^2. To formulate the UTF boundary coverage problem, we have the following assumptions:

1. We assume that the UTF is much larger than the sensing capabilities of the robot, and the moving speed of the robot is much faster than that of the UTF. This is common for large scale coverage occurring on the surface of the Earth.

2. The robot knows its current position using global positioning system and has a memory of where the robot has visited before.

3. We have no knowledge about UTF shapes.

4. GPs are capable of approximating the UTF boundary distribution.

6.3.2 UTF Properties and Modeling Perception

To further clarify our problem, let us define UTF and its key properties. Denote $\mathbf{z}_t$ to be the sensor readings at time t when the robot is at position $\mathbf{x}_t = [x(t)\ y(t)]^T \in \mathbb{R}^2$, and set

$\mathbf{Z}_T = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_T\}$ as all observations sensed from the beginning of localization process up to time T. Denote $\mathbf{T}$ to be the UTF region and $\mathbf{x} = [x, y]^T \in \mathbb{R}^2$ to be a point in the 2D space. $\mathbf{T}$ is usually obtained by thresholding the field boundary distribution function or geometric constraints which are described as follows,

$$\mathbf{T} := \left\{ \mathbf{x} \middle| \left\{ I_{\text{IN}} = \bigwedge_{\mathbf{z}_t \in \mathbf{Z}_T} \left(f(\mathbf{z}_t, \mathbf{x}_t, \boldsymbol{\sigma}) \geq f_t \right) \right\} = 1 \text{ (is TRUE)} \right\}, \tag{6.1}$$

where indicator variable $I_{\text{IN}} \in \{0, 1\}$ is binary (boolean) variable depending on if the thresholding criteria are satisfied, $\bigwedge$ is logic AND operator, f is a nonnegative field boundary distribution function, σ is the standard deviation of observation noise, and thresholding value f_t for field value is predetermined by application.

Unfortunately, there is often no prior knowledge about shape and position of UTFs. Instead of computing from geometric constraints, indicator variable I_{IN} values are often obtained by thresholding sensory readings. We assume the robot is equipped with an omnidirectional sensor with the maximum sensing distance d_s. If $d_s = 0$, the sensor becomes a point sensor such as a barometer measuring air pressure changes at its current position; if $d_s > 0$, the sensor may have measurable coverage like a camera covering the UTF. Let $\partial\mathbf{T}$ be the $\mathbf{T}$'s boundary. $\partial\mathbf{T}$ would be unmeasurable if there were no uncertainty in the sensing process. Due to the fact that $\mathbf{z}_t$ is normally distributed, we have,

$$\partial\mathbf{T} := \left\{ \mathbf{x} \middle| I_{\text{UTF}} = 1, \mathbf{x} \in \mathbf{T} \right\}, \tag{6.2}$$

where $I_{\text{UTF}} \in \{0, 1\}$ is a binary indicator variable describing if the point is a boundary point and

$$I_{\text{UTF}} = \left(\bigvee_{\mathbf{z_t} \in \mathbf{Z_T}} \left(f(\mathbf{z}_t, \mathbf{x}_t, \boldsymbol{\sigma}) = f_t \right) \right) \bigwedge \left(I_{\text{IN}=1} \right). \tag{6.3}$$

It is worth noting that (6.3) usually cannot be directly computed since we do not know

f. With observations from multiple sensor readings, it can be predicted by a GP based on observations $(\mathbf{z}_t, \mathbf{x}_t)$. To ensure the coverage of a UTF, we just need to cover its boundary $\partial\mathbf{T}$. The coverage of UTF interior is trivial if $\partial\mathbf{T}$ is covered.

6.3.3 Problem Definition

To quantify the boundary coverage performance, we need to define a performance metric to determine the trade-off between quality and effort: let $\beta \in (0, 1]$ be the coverage ratio threshold for UTF boundary. Denote $S_{\mathbf{T}}$ to be the area of the UTF's boundary $\partial\mathbf{T}$. The UTF's boundary is a 2D region due to robot sensing uncertainties. We have the following boundary coverage success metric.

Condition 1 (Quality Metric)**.** *The boundary coverage task for the UTF is considered to be accomplished if the covered boundary is no less than $\beta S_{\mathbf{T}}$ where $S_{\mathbf{T}}$ is the area of the UTF's boundary $\partial\mathbf{T}$.*

With inputs and quality metric defined, our problem is described as follows,

Problem 4. *Given the observation set $\mathbf{Z}_T$, plan robot trajectory $\mathbf{x}_{T+1}$ based on $\mathbf{x}_T$ to generate ellipses to cover $\partial\mathbf{T}$ with Condition 1 satisfied.*

6.4 System Modeling

The robot starts the boundary coverage process when it encounters an UTF region. The boundary coverage process generates a sequence of ellipses to enclose the UTF, which are treated as nodes for a depth-first search of a tree. Employing a sequence of ellipses allows us to break down a long boundary traversing problem into a sequence of local problems to reduce problem scale. In each local problem, we can handle challenges associated with limited sensing range and observation uncertainty. We arrange each ellipse to have its long axis aligned with the boundary of the UTF to speed up the boundary coverage process. At each ellipse, the robot accumulates observations to instantiate a Gaussian Process (GP)

to predict the position of next ellipse along the boundary. The depth-first search ensures that the robot traverses the entire UTF boundary and coordinates ellipse generation as tree expansion which will be detailed later.

Let us clarify the use of index variables in the depth-first search progress. Ellipse $\mathbf{A}_q$ is where the robot is currently located. Index g refers to the total number of ellipses. Since we start the index with its root at $\mathbf{A}_0$, $\mathbf{A}_g$ is also the new ellipse during new node generation. $\mathbf{A}_p$ refers to which neighboring ellipse the robot uses to enter $\mathbf{A}_q$. The "neighboring ellipse" refers to either parent or child nodes of q on the search tree.

The whole process consists of two main steps: initialization and boundary traversing. We start with system initialization.

6.4.1 Ellipses, Robot Trajectory, Observation Set, and Initialization of the Depth-First Search

The boundary coverage process relies on a sequence of ellipses to track UTF boundaries. Define $\mathbf{A}_q$ as the q-th ellipse,

$$\mathbf{A}_q = \left\{ \mathbf{x} \middle| \left(\mathbf{x} - \mathbf{x}_q\right)^T \mathbf{C}_q \left(\mathbf{x} - \mathbf{x}_q\right) \leq 1 \right\}, \tag{6.4}$$

where $\mathbf{x}_q = [x_q, y_q]^T$ is its center point and $\mathbf{C}_q$ is a 2×2 positive definite matrix.

(a) Accumulate obs. in $\mathbf{A}_q$ (b) Compute $L_q(\lambda)$ and obtain $\mathbf{A}_g$

Figure 6.2: The robot accumulates observations in $\mathbf{A}_q$ in the blue shaded area in a), establishes belief functions in $\mathbf{A}_q$ using a GP based on observation set $\mathbf{O}_q$, which assists in determining $\mathbf{A}_g$ in b).

When the robot enters $\mathbf{A}_q$ from a neighboring ellipse center $\mathbf{x}_p$ to current ellipse center $\mathbf{x}_q$ along the shortest path, the robot trajectory set $\mathbf{u}_q$ in $\mathbf{A}_q$ can be defined as,

$$\mathbf{u}_q = \{\mathbf{x} | \mathbf{x} = \rho \mathbf{x}_p + (1-\rho)\mathbf{x}_q, \rho \in [0,1]\} \tag{6.5}$$

if ignoring obstacles. Denote $t_j \in \mathbb{R}$ as the exact continuous time at the moment of the discrete time j when the robot with constant velocity traverses from $\mathbf{x}_p$ to $\mathbf{x}_q$. Let $t_j - t_{j-1} = c_0$ for $j > 0$ where $c_0 \in \mathbb{R}_+$ is a constant variable. The index j is reset to zero every time when the robot reaches $\mathbf{x}_q$. During the travel, the robot accumulates observations from its on-board sensor to establish its observation set $\mathbf{O}_q$,

$$\mathbf{O}_q = \left\{ (\mathbf{x}, \mathbf{z}_{t_j}) \middle| \, |\mathbf{x} - \mathbf{x}_{t_j}| \leq d_s, \mathbf{x}_{t_j} \in \mathbf{u}_q \right\}, \tag{6.6}$$

and denote $\mathbf{x} \in \mathbf{X}_q$. Fig. 6.2a illustrates the observation set $\mathbf{O}_q$ coverage in $\mathbf{A}_q$.

The ellipse generation process initializes at the moment when the robot first encounters an UTF at point $\mathbf{x}_0$. It immediately generates $\mathbf{A}_0$ which is chosen to be a circle because it is likely that we do not have enough information to determine boundary direction yet. For $\mathbf{A}_0$, we have

$$\mathbf{C}_0 = \begin{bmatrix} 1/4d_s^2 & 0 \\ 0 & 1/4d_s^2 \end{bmatrix}. \tag{6.7}$$

On the other hand, $\mathbf{u}_0$ is slightly different from (6.5) because we do not have a neighboring ellipse. Alternatively, we substitute $\mathbf{x}_p$ with $\mathbf{x}_{\text{enter}}$ in (6.5) to obtain $\mathbf{u}_0$ where $\mathbf{x}_{\text{enter}}$ is the point of entry to $\mathbf{A}_0$ during the global search process. Consequently, we have a non-empty observation set $\mathbf{O}_0$.

6.4.2 Depth-First Search-based Boundary Traversing

We employ a depth-first search over a tree, which contains all ellipses as tree nodes. Ellipse $\mathbf{A}_0$ is the root node of the ellipse tree. Each node q stores its $\mathbf{u}_q$ and $\mathbf{O}_q$. $\mathbf{O}_q$ is updated as the robot travels inside the ellipse. As mentioned before, the robot only moves between ellipse centers $\mathbf{x}_q$ of neighboring tree nodes along a linear path because we ignore the obstacle in the process. This also yields a piecewise linear trajectory for the robot.

6.4.2.0.1 Branching Method At each ellipse $\mathbf{A}_q$, the robot uses $\mathbf{O}_q$ to instantiate a GP [152] approximating the belief function for $\partial\mathbf{T}$ in $\mathbf{A}_q$ which provides information to determine $\mathbf{A}_g$. More specifically, for an $\mathbf{x} \in \mathbf{A}_q$, the GP provides posterior distribution $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$ for the UTF region to stand for the field function. It is worth noting that $\mathbf{O}_q$ theoretically contains infinite number of observations. To facilitate computation, we sample $\mathbf{O}_q$ using a local lattice according to sensor spatial resolution or task needs to accelerate GP training time. Recall $\mathbf{O}_q$ is continuously updated according to the robot trajectory set. This leads to a recursive Bayesian estimation process, which can be computed using a two-phase approach [153]:

1. Update Phase: For an $\mathbf{x}_* \in \mathbf{O}_q$,

$$P(I^*_{\text{UTF}}|\mathbf{x}_*, \mathbf{O}_q) = bel(I^*_{\text{UTF}}|f^*(\mathbf{x}_*), \sigma^2), \tag{6.8}$$

   where the latent function f^* is represented by GP, an approximation of (6.3).

2. Prediction Phase: The GP provides posterior distribution for the UTF boundary for a given $\mathbf{x} \notin \mathbf{O}_q$,

$$P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q) = \overline{bel}(I_{\text{UTF}}|\mu_I, \sigma_I^2). \tag{6.9}$$

   Here, μ_I and σ_I^2 are the expectation and variance of the posterior distribution re-

lated to the kernel function, which characterizes the correlation between the function values at different locations, namely, $f^*(\mathbf{x}_i)$ and $f^*(\mathbf{x}_j)$. We employ a histogram intersection kernel K as

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{2} \min(\mathbf{x}_i(d), \mathbf{x}_j(d)). \tag{6.10}$$

for $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^2$ with $\mu_I = \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} I^*_{\text{UTF}}$ and $\sigma_I^2 = \mathbf{k}_{**} \mathbf{k}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_*$ where $\mathbf{k}_* = K(\mathbf{X}_q, \mathbf{x})$, $\mathbf{K}$ is the kernel matrix of the training data $\mathbf{X}_q$, $\mathbf{k}_{**} = K(\mathbf{x}, \mathbf{x})$, and $\mathbf{I}$ is an identity matrix.

To approximate the UTF boundary $\partial \mathbf{T}$, we calculate the level set $L_q(\lambda)$ where threshold $\lambda > 0$, to cover regions with high probability of containing the boundary. This is done by thresholding on $P(I_{\text{UTF}} | \mathbf{x}, \mathbf{O}_q)$.

$$L_q(\lambda) = \left\{ \mathbf{x} | P(I_{\text{UTF}} | \mathbf{x}, \mathbf{O}_q) \geq \lambda, \mathbf{x} \in \mathbf{A}_q \right\}. \tag{6.11}$$

The value of threshold λ is determined by coverage ratio threshold β in Condition 1 and will be discussed in Section 6.5.2.

Now let us show how to determine $\mathbf{x}_g$, center of the new node on the ellipse tree. The boundary of $L_q(\lambda)$ intercepts the boundary of $\mathbf{A}_q$ and generates a set of points $\mathbf{X}_q^L$. As illustrated Fig. 6.2b, we evenly divide $\mathbf{A}_q$ into $k_d = 12$ sectors with each sector spanning $2\pi/k_d = \pi/6$. For each sector, we identify a middle angle boundary point $\mathbf{x}_q^s$ by intercepting $\mathbf{A}_q$ boundary with the ray shooting from the ellipse center along the middle angle (π/k_d from the sector side). We add $\mathbf{x}_q^s$ to the candidate solution set $\mathbf{X}_q^*$ for $\mathbf{x}_g$ if we can find a solution in set $\mathbf{X}_q^L$ located on the corresponding sector boundary. This means that we use $\mathbf{X}_q^L$ to filter out less likely candidate center locations. To avoid repeated search, we remove $\mathbf{x}_q^s$ of the sector where the robot enters $\mathbf{A}_q$ from the candidate solution set $\mathbf{X}_q^*$.

Therefore, set $\mathbf{X}_q^*$ only contains branches that robot has not visited.

6.4.2.0.2 Termination Scenarios With $\mathbf{X}_q^*$ introduced, let us explain the termination condition of the search, which has two scenarios. When $\mathbf{X}_q^*$ is empty, it means that the robot reaches the extreme end of the UTF which is the leaf of the depth-first search tree. Now the only choice is to let the robot traverse back on the tree to the parent ellipse and check if the candidate solution set of the parent ellipse is non-empty. If a non-empty node is found, we enter the tree expansion case as described in the sub-section. Otherwise, we keep back-traversing the robot and repeat this process along the tree to upper level parent node. We update p and q in the process.

If we return to the root and find $\mathbf{X}_0^*$ is empty, it means the robot has covered the entire search tree in the boundary traversing. This concludes the first termination scenario and the depth-first search ends. This scenario occurs a lot with curves, lines or other thin UTFs.

For a compact and sizable UTF, it is likely that the robot loops around the UTF (see Fig.6.1b), which is the second termination scenario. Because it leads the robot to travel back to $\mathbf{A}_0$, we can identify this by verifying if $\mathbf{x}_q \in \mathbf{A}_0$ is true. We also need to remove the candidate solution from the sector that contains $\mathbf{x}_q$ in $\mathbf{X}_0^*$. Again, if $\mathbf{X}_0^*$ is empty, the search ends.

6.4.2.0.3 Node/Ellipse Generation Here comes the tree expansion or node generation process. It happens if the candidate solution set $\mathbf{X}_q^*$ is non-empty for the original q or the updated q from the back-traversing process, then we choose the $\mathbf{x}_g$ as follows,

$$\mathbf{x}_g = \arg\max_{\mathbf{x}\in\mathbf{X}_q^*} \left(\arccos \frac{\langle \mathbf{x}_{q^-}, \mathbf{x}_{*,q} \rangle}{\|\mathbf{x}_{q^-}\| \cdot \|\mathbf{x}_{*,q}\|} \right). \tag{6.12}$$

where $\mathbf{x}_{q^-} = \mathbf{x}_q - \mathbf{x}_p$ represents the vector describing how the robot enters $\mathbf{A}_q$, $\mathbf{x}_{*,q} = \mathbf{x} - \mathbf{x}_q$, $\|\cdot\|$ is vector l-2 norm, and $\langle \cdot, \cdot \rangle$ is vector inner product. This means that the candidate is the closest to the direction that the robot enters $\mathbf{A}_q$. Once $\mathbf{x}_g$ is chosen, we

remove it from $\mathbf{X}_q^* = \mathbf{X}_q^* \setminus \{\mathbf{x}_g\}$. This is to avoid repeated search when we return to $\mathbf{A}_q$ in the depth-first search process. Again, set $\mathbf{X}_q^*$ keeps track of the visited branch of the search tree. New node g is added to the tree with its parent to be q and $g = g + 1$.

After obtaining new center $\mathbf{x}_g$, we place $\mathbf{A}_g$ into its position by determining $\mathbf{C}_g$. Set the long and short axes of $\mathbf{A}_g$ to be $4d_s$ and $2d_s$, respectively. To approximate the UTF boundary, we want the long axis of $\mathbf{A}_g$ to be aligned with $\mathbf{x}_g - \mathbf{x}_q$ as follows,

$$\mathbf{C}_g = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \frac{1}{16d_s^2} & 0 \\ 0 & \frac{1}{4d_s^2} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{6.13}$$

where θ is determined by

$$\theta = \arctan\left(\frac{y_g - y_q}{x_g - x_q}\right). \tag{6.14}$$

After $\mathbf{A}_g$ is determined, the robot motion is also obtained as $\mathbf{u}_g$. As the robot moves toward $\mathbf{x}_g$, we update $\mathbf{X}_q^*$, p, and q accordingly.

6.5 Boundary Coverage Performance Analysis

The remaining question is how good the boundary coverage quality is and how to guarantee Condition 1. We first analyze the coverage quality for a point $\mathbf{x}$ in $\mathbf{A}_q$ and then aggregate it into the entire boundary.

6.5.1 Probability Bounds for a Point x in $\mathbf{A}_q$

The UTF boundary $\partial\mathbf{T}$ is covered by the level set $L_q(\lambda)$ which is generated by the thresholding on $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$ in (6.11). It is important to understand $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$. $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$ is a function of its condition $(\mathbf{x}, \mathbf{O}_q)$ which means it is still a random variable because $(\mathbf{x}, \mathbf{O}_q)$ are random variables. Since it is a random variable, it has an expectation $E\left(P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)\right)$. $E\left(P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)\right)$ can be estimated by averaging $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$ across all points using the output from GPs. $E\left(P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)\right)$ can be viewed as an observation of the unconditional

distribution $P(I_{\text{UTF}})$. $P(I_{\text{UTF}})$ is the mean value of $P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)$. $P(I_{\text{UTF}})$ is important because it can help us to determine if we miss any UTF boundary points when performing thresholding in (6.11). We do not know $P(I_{\text{UTF}})$ but we know its low bound in probability as follows.

Lemma 2. *With $1-\tau$ probability, $P(I_{UTF})$ which is the unconditional probability that a point in $\mathbf{A}_q$ belongs to UTF boundary has the following lower bound B_q^-,*

$$P(I_{UTF}) \geq B_q^- = \inf_{\eta>0}\left\{E(P(I_{UTF}|\mathbf{x},\mathbf{O}_q)) - \frac{\eta}{2} - \frac{1}{l_{\max}\eta}\log\frac{1}{\tau}\right\}, \tag{6.15}$$

where $\tau \in (0,1)$ is a chosen small number, $l_{\max}$ is the set cardinality of $L_q(\lambda)$, and non-negative variable η is determined by the inf *computation.*

Proof. The lower bound of $P(I_{\text{UTF}})$ can be proved by relating it to its estimator $E(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q))$. We construct the following probability event E_0 for $t > -1$,

$$P(E_0) = P\left(P(I_{\text{UTF}}) - \frac{\eta}{2} - E\left(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)\right) \leq -(t+1)\right). \tag{6.16}$$

This is equivalent to,

$$P(E_0) = P(e^{-\eta l_{\max}\left(P(I_{\text{UTF}})-\frac{\eta}{2}-E\left(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)\right)\right)} \geq e^{\eta l_{\max}(t+1)}), \tag{6.17}$$

and apply Markov's inequality, we have,

$$P(E_0) \leq e^{-\eta l_{\max}(t+1)}E\left(e^{-\eta l_{\max}\left(P(I_{\text{UTF}})-\frac{\eta}{2}-E\left(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)\right)\right)}\right). \tag{6.18}$$

Since $e^{\eta l_{\max}\frac{\eta}{2}} \geq 1$, we multiply it to the right hand side of (6.18) and move the constant

terms outside the expectation $E(\cdot)$, we have

$$\begin{aligned} P(E_0) \leq & e^{-\eta l_{\max}\left(t-\frac{\eta}{2}\right)-\eta l_{\max}\left(P(I_{\text{UTF}})-\frac{\eta}{2}\right)} \\ & \times E\left(e^{\eta l_{\max}\left(E\left(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)\right)-1\right)}\right) \end{aligned} \tag{6.19}$$

Using the fact that $E(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)) \leq 1$, we know

$$P(E_0) \leq e^{-\eta l_{\max}\left(t-\frac{\eta}{2}\right)-\eta l_{\max} P(I_{\text{UTF}})+\frac{\eta^2 l_{\max}}{2}} E\left(e^0\right), \tag{6.20}$$

which leads to

$$P(E_0) \leq e^{-\eta l_{\max}\left(t-\frac{\eta}{2}\right)+\frac{\eta^2 l_{\max}}{2}} = \tau. \tag{6.21}$$

We can solve t from (6.21) and plug it back to (6.16) and the lemma is proved.

□

6.5.2 Probability of Covering an UTF Boundary Point in Level Set Construction

Lower bound B_q^- can help us determine the probability that the UTF boundary is captured by the GP in level set construction (6.11). It also helps determine how to choose λ. It is clear that a reasonable λ should be smaller than B_q^-. Defining $F_{(\mu,\sigma^2)}(x)$ as the cumulative probability function of the Gaussian distribution $N(\mu,\sigma^2)$, we have the following lemma,

Lemma 3. *For a given lower bound B_q^- and $\lambda \leq B_q^-$ at point $\mathbf{x} \in \mathbf{A}_q$ when computing the level set $L_q(\lambda)$, the probability that an UTF boundary point satisfies* (6.11) *is no less than $(1-F_{(B_q^-,\sigma_I^2)}(\lambda))(1-\tau)$ where σ_I^2 is the variance of $P(I_{UTF}|\mathbf{x},\mathbf{O}_q)$.*

Proof. From GP model, we know $P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q)$ is a Gaussian distribution with mean $\mu_I = P(I_{\text{UTF}})$ and variance σ_I^2. For a given λ and B_q^-, the fact that an UTF boundary point is captured means the following conditional event $P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\, \mu_I \geq B_q^-$ occurs. We

now compute its probability by further conditioning on E_0 and applying the fact that B_q^- is not a deterministic bound,

$$\begin{aligned}
&P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I \geq B_q^-) \\
&\quad = P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I \geq B_q^-, E_0)(1-\tau) \\
&\qquad + P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I \geq B_q^-, \overline{E_0})\tau \\
&\quad \geq P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I \geq B_q^-, E_0)(1-\tau) \\
&\quad \geq P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I = B_q^-, E_0)(1-\tau). \qquad (6.22)
\end{aligned}$$

Therefore, we have

$$P(P(I_{\text{UTF}}|\mathbf{x},\mathbf{O}_q) \geq \lambda \,|\mu_I \geq B_q^-)) \geq (1 - F_{(B_q^-,\sigma_I^2)}(\lambda))(1-\tau). \qquad (6.23)$$

□

Note that σ_I^2 can be obtained using GP outputs.

6.5.3 Ensure Boundary Coverage Quality

Recall that we have coverage ratio threshold β in boundary coverage quality metric defined in Condition 1 in our problem definition in Section 6.3.3. To satisfy the condition, we choose λ and τ accordingly. This means that we can set

$$(1 - F_{(B_q^-,\sigma_I^2)}(\lambda))(1-\tau) = \beta, \qquad (6.24)$$

to help obtain a correct threshold λ. We do have some freedom in choosing τ to assist the selection of λ. It is not difficult to perform binary search to obtain it.

Algorithm 2: Boundary Coverage of an UTF

Input : Robot observations $\mathbf{O}_q$
Output: Robot trajectory in set $\mathbf{G}$

Generate $\mathbf{A}_0$ at $\mathbf{x}_0$; // $O(1)$
Stack $S = \{\}$; // $O(1)$
Push$(S, \mathbf{x}_0)$; // $O(1)$
while *S is not empty* **do** // $O(\nu)$
    $\mathbf{x}_q = \text{Pop}(S)$; // $O(1)$
    if *Visited*$(\mathbf{x}_q) :=$ *FALSE* **then**
        Visited$(\mathbf{x}_q) :=$ TRUE; // $O(1)$
        Update p and q according to trajectory; // $O(1)$
        $P(I_{\text{UTF}}|\mathbf{x}, \mathbf{O}_q)$ using GPs; // $O(n_q \log n_q)$
        Obtain B_q^- according to (6.15); // $O(\log n_s)$
        Calculate λ through (6.24); // $O(1)$
        $\mathbf{X}_q^* = \bigcup \mathbf{x}_q^s \backslash \{\mathbf{x}_p\}$; // $O(1)$
        if $\mathbf{X}_q^* = \emptyset$ *or* $\mathbf{x}_q \in \mathbf{A}_0$ *for* $q \neq 0$ **then**
            if *Robot at* $\mathbf{A}_0$ **then**
                Break; // $O(1)$
            else
                Travese back to parent node; // $O(1)$
        else
            Obtain $\mathbf{A}_g$ and move to $\mathbf{x}_g$; // $O(1)$
        Push$(S, \mathbf{x}_g)$; // $O(1)$
        $\mathbf{G} = \mathbf{G} \bigcup \{\mathbf{x}_g\}$; // $O(1)$

6.5.4 Algorithm and Complexity Analysis

Algorithm 2 summarizes the proposed method. To overcome the computational limitations of naive GPs with time complexity $O(n_q^3)$, we employ a Gaussian process with generalized histogram intersection kernels to speed up the naive GPs to $O(n_q \log n_q)$ where n_q is the set cardinality of $\mathbf{O}_q$ [153]. Recall that ν refers the number of ellipse centers, and we initially set $\mathbf{O}_q = \emptyset$, $\mathbf{X}_q^* = \emptyset$, and $\mathbf{G} = \emptyset$. For the coverage process, Algorithm 2 details the pseudocode, which leads to the following complexity result.

Lemma 4. *Boundary coverage algorithm for an UTF runs in* $O(\nu n_q \log n_q)$ *time, where*

n_q is the set cardinality of $\mathbf{O}_q$, and ν is the number of ellipses to enclose the UTF.

6.6 Experimental Result

We have implemented the proposed method in Matlab on a Laptop PC with an Intel(R) Core^{TM} i7-3517U CPU@1.90GHz and 8 GB memory. To verify the proposed local coverage method, we simulate different field shapes to test our approach (see Table 6.1). It includes both simple geometric shapes such as lines, circles, squares, and two complex shapes including storm cells and an island. Fig. 6.3 shows the images of the two complex shapes. Each image has a resolution of 720×480 pixels.

To measure our algorithm's boundary coverage capability, we define $\mathbf{A}_L$ as the boundary area covered using our method, and $\mathbf{A}$ as the actual area the UTF boundary occupies. We evaluate the coverage ratio β_r by using:

$$\beta_r = \frac{|\mathbf{A}_L \cap \mathbf{A}|}{|\mathbf{A}|}. \tag{6.25}$$

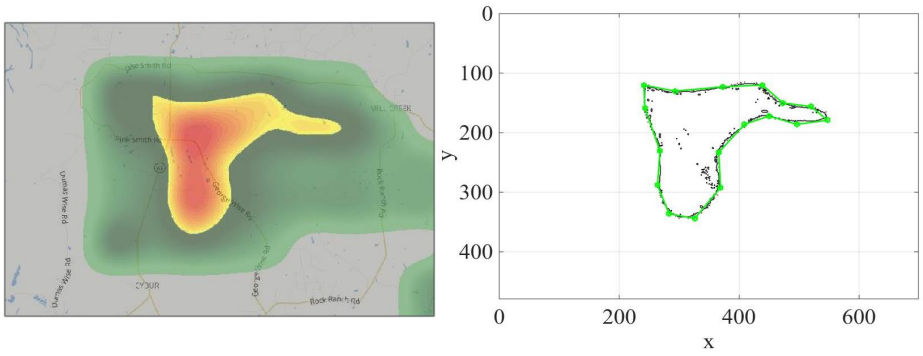
We set the value $d_s = 1$ for the simple geometric shapes and d_s equal to 20 pixels for the two image-based case.

Table 6.1: Local coverage experiment settings and results.

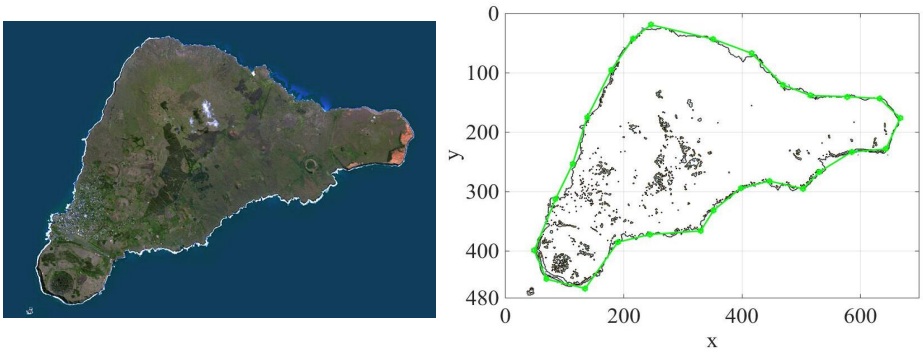
UTF Type	$\mathbf{f}_j(\mathbf{x})$	Dimension	Boundary	Shape	β_r	β
Line	$2x-y+5=0$ with $3 \le x \le 12$	1D	smooth	convex	98.12%	95.00%
Circle	$25-(x-6)^2-(y-6)^2 \ge 0$	2D	smooth	convex	97.56%	95.00%
Square	$\lvert x-5\rvert \le 6, \lvert y-5\rvert \le 6$	2D	non-smooth	convex	93.34%	90.00%
Storm cell	see Fig. 6.3a	2D	non-smooth	non-convex	87.32%	85.00%
Island	see Fig. 6.3b	2D	non-smooth	non-convex	88.59%	85.00%

The experimental settings and results are shown in Table 6.1. The last two rows show that for a given different threshold β, our algorithm has guaranteed that the actual coverage is no less than β for all testing cases, which is conformable to our analysis. Also, the

sample robot trajectories for boundary traversing are illustrated as green piecewise linear curves in the right side of Fig. 6.3. It is clear that the robot successfully covers the both testing cases.



(a) A weather radar map showing a storm cell.



(b) An island map.

Figure 6.3: Local coverage testing with real image data and robot coverage path.

6.7 Conclusion

We introduced a new UTF boundary coverage problem with many applications. We reported a probabilistic boundary coverage method for addressing UTF problems. We generated a sequence of ellipses to cover UTF boundary. The introduction of ellipse se-

quence also allowed us to decompose the long trajectory traversing problem to multiple local problems with each ellipse represented a local problem. In each local problem, we employed Gaussian processes (GPs) as a local belief function to approximate field distribution. The local belief function allows us to predict UTF boundary trends and establish adjacent ellipses for further exploration. The process was governed by a depth-first search process until UTF is approximately enclosed by connected ellipses. We formally proved that our boundary coverage process guarantees the enclosure above a given coverage ratio with a preset probability threshold. We implemented our algorithm and successfully tested it in experiments with different field types.

7. CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

Navigation is a fundamental problem for robots as a combination of motion planning, obstacle avoidance, object detection, and tracking, mapping and localization. A typical problem is how to navigate the robot in an uncertain belief space due to noisy and partial observations of the state of both the surrounding environment and the robot. As autonomous systems leave the factory floor and become more pervasive in the form of drones and self-driving cars, it is becoming increasingly important to understand how to design systems that will not fail under these real-world conditions. Although it is important that these systems be safe, it is also important that they do not operate so conservatively as to be ineffective. They must have a strong understanding of the risks induced by their actions so they can avoid unnecessary risks and operate efficiently. In this dissertation, we explore the belief space-guided robotic navigation problems, which include belief space-based scene understanding for autonomous vehicles, and introduce belief space guided robotic planning.

When autonomous vehicle (AV) navigates in the city or urban environment in Chapter 4, the quality of lane markings (LMs) has to be assessed to help improve infrastructure for autonomous driving. Based on our multi-module fused based methods, we presented metrics and algorithms for lane marking assessment. Three lane marking quality metrics were proposed and modeled mathematically: correctness, shape, and visibility. We also proposed a dual-modal algorithm to facilitate the computation of the three metrics. We took both prior map uncertainty and sensory uncertainty into consideration in formulating our metrics and the algorithm.

We also use belief space for better scene understanding. As an example, we utilize

crowdsourced images from multiple vehicles to help verify LMs for high-definition (HD) map maintenance in Chapter 5. We model them respectively as observations of two LM spatial distributions in the camera frame and check if they agree with each other via goodness of fit test by realizing LMs in both the HD map and camera images contain noises. We derive a sequential Bayes' model to allow the belief functions to be updated using crowdsourced images.

In chapter 6, we utilize the belief space to tightly connect perception and planning for autonomous vehicles. We detect free space using camera-lidar sensor fusion and proposed a availability-reasonability-feasibility test to determine whether the AV should simply follow actual lane boundary or generate virtual lane boundary by taking into account vehicle kinodynamic constraints, obstacle avoidance, smooth motion, ground-positioning-system trajectory following, respecting the direction of lane markings in a multi-objective optimization framework with dynamically adjustable weights for different road scenarios.

We propose to use belief space for probabilistic boundary coverage of unknown target fields (UTFs) in Chapter 7. We generated a sequence of ellipses to cover the UTF boundary, which allowed us to decompose the long trajectory traversing problem to multiple local problems with each ellipse represented a local problem. In each local problem, we employed Gaussian processes as a local belief function to approximate field distribution. The local belief function allows us to predict UTF boundary trends and establish adjacent ellipses for further exploration. The process was governed by a depth-first search process until UTF is approximately enclosed by connected ellipses. We formally proved that our boundary coverage process guarantees the enclosure above a given coverage ratio with a preset probability threshold.

7.2 Future Work

In this dissertation, we propose different robotic applications for belief space-guided navigation. The following directions can be explored in the future.

- *Belief space-based scene understanding for lane marking quality assessment*: We will perfect the algorithm and perform more tests. We will also develop assessment algorithms for traffic signals, signs, and surface quality.

- *Belief space cross validation from crowdsourced data for lane marking verification*: We will extend approach to other objects in HD maps to ensure HD maps can be kept up-to-date at low cost. We will consider to reconstruct LMs from the perception inputs and update the HD maps. We will theoretically analyze the number of samples required to verify the LMs.

- *Belief space for tightly connection between perception and motion planning*: We will conduct more physical experiments and we will incorporate more functionalities such as velocity planning to make navigation decisions more human-like and human-compatible.

- *Belief space-based approach for probabilistic boundary coverage of UTFs*: We will provide overall trajectory length prediction for the algorithm. We will consider a multiple robot team and moving targets. We will also consider robots/UAVs with kinodynamic constraints in the trajectory generation.

REFERENCES

[1] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 146, no. 2, pp. 80–84, 1999.

[2] M. C. Motwani, M. C. Gadiya, R. C. Motwani, and F. C. Harris, "Survey of image denoising techniques," in *Proceedings of GSPX*, pp. 27–30, 2004.

[3] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[4] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *Applied Robotics for the Power Industry (CARPI), 2012 2nd International Conference on*, pp. 16–21, IEEE, 2012.

[5] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. D. Konidaris, "Robot motion planning on a chip.," in *Robotics: Science and Systems*, 2016.

[6] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[7] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Optimal, smooth, nonholonomic mobile robot motion planning in state lattices," *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-15*, 2007.

[8] T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 325–345, 2008.

[9] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and autonomous systems*, vol. 55, no. 6, pp. 460–469, 2007.

[10] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for autonomously navigating urban intersections," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 825–830, IEEE, 2017.

[11] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6483–6490, IEEE, 2014.

[12] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 66–73, IEEE, 2016.

[13] M. Mäkitalo and A. Foi, "Noise parameter mismatch in variance stabilization, with an application to poisson–gaussian noise estimation," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5348–5359, 2014.

[14] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise estimation from a single image," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 901–908, IEEE, 2006.

[15] M. Polic and T. Pajdla, "Uncertainty computation in large 3d reconstruction," in *Image Analysis* (P. Sharma and F. M. Bianchi, eds.), (Cham), pp. 110–121, Springer International Publishing, 2017.

[16] V. A. Kovalev and W. E. Eichinger, *Elastic lidar: theory, practice, and analysis methods*. John Wiley & Sons, 2004.

[17] L. Matthies and A. Elfes, "Integration of sonar and stereo range data using a grid-based representation," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pp. 727–733, IEEE, 1988.

[18] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, *et al.*, "Robot task planning and explanation in open and uncertain worlds," *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[19] J. Fabian and G. M. Clayton, "Error analysis for visual odometry on indoor, wheeled mobile robots with 3-d sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1896–1906, 2014.

[20] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[21] L. Nardi and C. Stachniss, "Uncertainty-aware path planning for navigation on road networks using augmented mdps," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5780–5786, IEEE, 2019.

[22] J. P. Gonzalez and A. Stentz, "Planning with uncertainty in position using high-resolution maps," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 1015–1022, IEEE, 2007.

[23] Y. Koren, J. Borenstein, *et al.*, "Potential field methods and their inherent limitations for mobile robot navigation.," in *ICRA*, vol. 2, pp. 1398–1404, 1991.

[24] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, "Real-time robot motion planning using rasterizing computer graphics hardware," *ACM SIGGRAPH Computer Graphics*, vol. 24, no. 4, pp. 327–335, 1990.

[25] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*, pp. 396–404, Springer, 1986.

[26] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1664–1669, IEEE, 1986.

[27] Y. K. Hwang and N. Ahuja, "Gross motion planning—a survey," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 219–291, 1992.

[28] W. H. Huang, B. R. Fajen, J. R. Fink, and W. H. Warren, "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 288–299, 2006.

[29] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[30] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[31] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[32] B. Nagy and A. Kelly, "Trajectory generation for car-like robots using cubic curvature polynomials," *Field and Service Robots*, vol. 11, 2001.

[33] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[34] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[35] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[36] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *2013 American control conference*, pp. 188–193, IEEE, 2013.

[37] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 3067–3074, IEEE, 2015.

[38] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113–5120, IEEE, 2018.

[39] M. McNaughton, *Parallel algorithms for real-time motion planning*. PhD thesis, Citeseer, 2011.

[40] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[41] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[42] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[43] S. Koenig and M. Likhachev, "Dˆ* lite," *Aaai/iaai*, vol. 15, 2002.

[44] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," in *The 2005 DARPA grand challenge*, pp. 1–43, Springer, 2007.

[45] T. Brandt, T. Sattel, and J. Wallaschek, "Towards vehicle trajectory planning for collision avoidance based on elastic bands," *International Journal of Vehicle Autonomous Systems*, vol. 5, no. 1-2, pp. 28–46, 2007.

[46] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6, VDE, 2012.

[47] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications*, vol. 2, no. 6, pp. 411–440, 1968.

[48] J. van den Berg, "Iterated lqr smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost," in *2014 American Control Conference*, pp. 1912–1918, IEEE, 2014.

[49] D. W. Harwood, A. D. May, I. B. Anderson, L. Leiman, and A. R. Archilla, "Capacity and quality of service of two-lane highways," *Final Report, NCHRP Project*, pp. 3–55, 1999.

[50] A. Flannery, K. Wochinger, and A. Martin, "Driver assessment of service quality on urban streets," *Transportation Research Record: Journal of the Transportation*

Research Board, no. 1920, pp. 25–31, 2005.

[51] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier, "Evaluation of road marking feature extraction," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pp. 174–181, IEEE, 2008.

[52] J. Pohl, W. Birk, and L. Westervall, "A driver-distraction-based lane-keeping assistance system," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 221, no. 4, pp. 541–552, 2007.

[53] K. Yamaguchi, A. Watanabe, T. Naito, and Y. Ninomiya, "Road region estimation using a sequence of monocular images," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[54] V. Pradeep, G. Medioni, and J. Weiland, "Piecewise planar modeling for step detection using stereo vision," in *Workshop on computer vision applications for the visually impaired*, 2008.

[55] J. Hernández and B. Marcotegui, "Filtering of artifacts and pavement segmentation from mobile lidar data," in *ISPRS Workshop Laserscanning 2009*, 2009.

[56] J. Li, X. Mei, D. Prokhorov, and D. Tao, "Deep neural network for structural prediction and lane detection in traffic scene," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 690–703, 2017.

[57] Y. Yu, J. Li, H. Guan, F. Jia, and C. Wang, "Learning hierarchical features for automated extraction of road markings from 3-d mobile lidar point clouds," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 2, pp. 709–726, 2015.

[58] H. Guan, J. Li, Y. Yu, C. Wang, M. Chapman, and B. Yang, "Using mobile laser scanning data for automated extraction of road markings," *ISPRS Journal of Pho-*

togrammetry and Remote Sensing, vol. 87, pp. 93–107, 2014.

[59] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 20–37, 2006.

[60] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[61] F. Samadzadegan, A. Sarafraz, and M. Tabibi, "Automatic lane detection in image sequences for vision-based navigation purposes," *ISPRS Image Engineering and Vision Metrology*, 2006.

[62] S. Kammel and B. Pitzer, "Lidar-based lane marker detection and mapping," in *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1137–1142, IEEE, 2008.

[63] A. von Reyher, A. Joos, and H. Winner, "A lidar-based approach for near range lane detection," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pp. 147–152, IEEE, 2005.

[64] K. Takagi, K. Morikawa, T. Ogawa, and M. Saburi, "Road environment recognition using on-vehicle lidar," in *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 120–125, IEEE, 2006.

[65] L. T. Sach, K. Atsuta, K. Hamamoto, and S. Kondo, "A robust road profile estimation method for low texture stereo images," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 4273–4276, IEEE, 2009.

[66] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[67] U. Hofmann, A. Rieder, and E. D. Dickmanns, "Radar and vision data fusion for hybrid adaptive cruise control on highways," *Machine Vision and Applications*, vol. 14, no. 1, pp. 42–49, 2003.

[68] X. Gu, A. Zang, X. Huang, A. Tokuta, and X. Chen, "Fusion of color images and lidar data for lane classification," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, p. 69, ACM, 2015.

[69] A. S. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Finding multiple lanes in urban road networks with vision and lidar," *Autonomous Robots*, vol. 26, no. 2-3, pp. 103–122, 2009.

[70] A. Mammeri, A. Boukerche, and Z. Tang, "A real-time lane marking localization, tracking and communication system," *Computer Communications*, vol. 73, pp. 132–143, 2016.

[71] B.-S. Shin, Z. Xu, and R. Klette, "Visual lane analysis and higher-order tasks: a concise review," *Machine vision and applications*, vol. 25, no. 6, pp. 1519–1547, 2014.

[72] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216–234, 2018.

[73] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, pp. 63–71, Springer, 2004.

[74] R. Guo, Q. Dai, and D. Hoiem, "Single-image shadow detection and removal using paired regions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2033–2040, 2011.

[75] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[76] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[77] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision*, vol. 5. McGraw-Hill New York, 1995.

[78] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," *European Conference on Computer Vision*, pp. 500–513, 2008.

[79] D. Arlia and M. Coppola, "Experiments in parallel clustering with dbscan," in *European Conference on Parallel Processing*, pp. 326–331, Springer, 2001.

[80] H. B. Barua and S. Sarmah, "An extended density based clustering algorithm for large spatial 3d data using polyhedron approach," *International Journal of Computer Applications*, vol. 58, no. 2, 2012.

[81] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[82] J. M. Duncan, "Factors of safety and reliability in geotechnical engineering," *Journal of geotechnical and geoenvironmental engineering*, vol. 126, no. 4, pp. 307–316, 2000.

[83] M. H. J. Vala and A. Baxi, "A review on otsu image segmentation algorithm," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 2, pp. pp–387, 2013.

[84] L. Magri and A. Fusiello, "T-linkage: A continuous relaxation of j-linkage for multi-model fitting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3954–3961, 2014.

[85] D. Borrmann, R. Heß, H. Houshiar, D. Eck, K. Schilling, and A. Nüchter, "Robotic mapping of cultural heritage sites.," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2015.

[86] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, 2015.

[87] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 5729–5736, IEEE, 2016.

[88] J. R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez, "Robot@ home, a robotic dataset for semantic mapping of home environments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 131–141, 2017.

[89] K. Abdulrahim and R. A. Salam, "Traffic surveillance: A review of vision based vehicle detection, recognition and tracking," *International journal of applied engineering research*, vol. 11, no. 1, pp. 713–726, 2016.

[90] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[91] A. Petrovai, R. Danescu, and S. Nedevschi, "A stereovision based approach for detecting and tracking lane and forward obstacles on mobile devices," in *Intelligent*

Vehicles Symposium (IV), 2015 IEEE, pp. 634–641, IEEE, 2015.

[92] A. Joshi and M. R. James, "Generation of accurate lane-level maps from coarse prior maps and lidar," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 19–29, 2015.

[93] D. C. Andrade, F. Bueno, F. R. Franco, R. A. Silva, J. H. Z. Neme, E. Margraf, W. T. Omoto, F. A. Farinelli, A. M. Tusset, S. Okida, *et al.*, "A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–11, 2018.

[94] B. Li, D. Song, H. Li, A. Pike, and P. Carlson, "Lane marking quality assessment for autonomous driving," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, IEEE, 2018.

[95] B. Li, D. Song, A. Ramchandani, H.-M. Cheng, D. Wang, Y. Xu, and B. Chen, "Virtual lane boundary generation for human-compatible autonomous driving: A tight coupling between perception and planning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019.

[96] Y. Kang, C. Roh, S.-B. Suh, and B. Song, "A lidar-based decision-making method for road boundary detection using multiple kalman filters," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4360–4368, 2012.

[97] A. Elfes, *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[98] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, no. 9, pp. 61–74, 1988.

[99] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[100] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[101] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," in *IEEE Transactions on Robotics (T-RO)*, vol. 31, pp. 736 —- 749, June 2015.

[102] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 206–211, IEEE, 2003.

[103] J. Ryde and N. Hillier, "Alignment and 3d scene change detection for segmentation in autonomous earth moving," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1484–1490, IEEE, 2011.

[104] A. Aijazi, P. Checchin, and L. Trassoudaine, "Automatic removal of imperfections and change detection for accurate 3d urban cartography by classification and incremental updating," *Remote Sensing*, vol. 5, no. 8, pp. 3701–3728, 2013.

[105] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[106] J. S. Berrio, J. Ward, S. Worrall, and E. Nebot, "Identifying robust landmarks in feature-based maps," *arXiv preprint arXiv:1809.09774*, 2018.

[107] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.

[108] A. Nurminen and A. Oulasvirta, "Designing interactions for navigation in 3d mobile maps," in *Map-based mobile services*, pp. 198–227, Springer, 2008.

[109] A. Satorra and P. M. Bentler, "A scaled difference chi-square test statistic for moment structure analysis," *Psychometrika*, vol. 66, no. 4, pp. 507–514, 2001.

[110] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[111] Y. Li and N. R. Gans, "Predictive ransac: Effective model fitting and tracking approach under heavy noise and outliers," *Computer Vision and Image Understanding*, vol. 161, pp. 99 – 113, 2017.

[112] A. Mukhtar, L. Xia, and T. B. Tang, "Vehicle detection techniques for collision avoidance systems: A review.," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.

[113] A. Pandey, S. Pandey, and D. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review," *Int Rob Auto J*, vol. 2, no. 3, p. 00022, 2017.

[114] D. Song, H. N. Lee, J. Yi, and A. Levandowski, "Vision-based motion planning for an autonomous motorcycle on ill-structured roads," *Autonomous Robots*, vol. 23, no. 3, pp. 197–212, 2007.

[115] K. Souhila and A. Karim, "Optical flow based robot obstacle avoidance," *International Journal of Advanced Robotic Systems*, vol. 4, no. 1, p. 2, 2007.

[116] A. Sgorbissa and R. Zaccaria, "Planning and obstacle avoidance in mobile robotics," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 628–638, 2012.

[117] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles.," *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[118] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive-an autonomous journey on a historic route.," *IEEE Intell. Transport. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, 2014.

[119] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 250–256, IEEE, 2015.

[120] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm.," in *ICAPS*, pp. 262–271, 2005.

[121] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, 2012.

[122] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2016.

[123] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 454–460, IEEE, 2015.

[124] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Transactions on*

Intelligent Transportation Systems, vol. 16, no. 4, pp. 1961–1976, 2015.

[125] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Ieee Pervas Comput*, vol. 7, no. 4, pp. 12–18, 2008.

[126] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985.

[127] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[128] S. Gumhold, X. Wang, and R. S. MacLeod, "Feature extraction from point clouds.," in *IMR*, Citeseer, 2001.

[129] A. Y. Hata, F. S. Osorio, and D. F. Wolf, "Robust curb detection and vehicle localization in urban environments," in *Intelligent vehicles symposium proceedings, 2014 IEEE*, pp. 1257–1262, IEEE, 2014.

[130] T. Mörwald, J. Balzer, and M. Vincze, "Modeling connected regions in arbitrary planar point clouds by robust b-spline approximation," *Robotics and Autonomous Systems*, vol. 76, pp. 141–151, 2016.

[131] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[132] G. E. Farin and G. Farin, *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[133] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[134] S.-k. Yun and D. Rus, "Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning," *Robotica*, vol. 32, no. 02, pp. 257–277, 2014.

[135] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2016.

[136] I. Shnaps and E. Rimon, "Online coverage by a tethered autonomous mobile robot in planar unknown environments," *IEEE Transactions on Robotics*, vol. 30, pp. 966–974, Aug 2014.

[137] K. Bekris, R. Shome, A. Krontiris, and A. Dobson, "Reducing roadmap size for network transmission in support of cloud automation," *IEEE Robotics and Automation Magazine*, 2016.

[138] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous robots*, vol. 31, no. 4, p. 299, 2011.

[139] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

[140] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2013.

[141] L. Xu and A. Stentz, "An efficient algorithm for environmental coverage with multiple robots," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4950–4955, IEEE, 2011.

[142] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *IEEE International Conference on Robotics and Automation*, pp. 5525–

5530, 2010.

[143] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 1471–1476, 2008.

[144] Y. Maeda, N. Kodera, and T. Egawa, "Caging-based grasping by a robot hand with rigid and soft parts," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5150–5155, 2012.

[145] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 886–900, 2012.

[146] P. Pipattanasomporn, T. Makapunyo, and A. Sudsang, "Multifinger caging using dispersion constraints," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 1033–1041, 2016.

[147] W. Wan and R. Fukui, "Efficient planar caging test using space mapping," *IEEE Transactions on Automation Science and Engineering*, vol. 15, pp. 278–289, Jan 2018.

[148] P. Vongmasa and A. Sudsang, "Coverage diameters of polygons," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4036–4041, IEEE, 2006.

[149] G. A. Pereira, M. F. Campos, and V. Kumar, "Decentralized algorithms for multi-robot manipulation via caging," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 783–795, 2004.

[150] V. Ivan and S. Vijayakumar, "Space-time area coverage control for robot motion synthesis," in *International Conference on Advanced Robotics (ICAR)*, pp. 207–212, IEEE, 2015.

[151] D. Zarubin, F. T. Pokorny, M. Toussaint, and D. Kragic, "Caging complex objects with geodesic balls," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2999–3006, IEEE, 2013.

[152] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[153] E. Rodner, A. Freytag, P. Bodesheim, and J. Denzler, "Large-scale gaussian process classification with flexible adaptive histogram kernels," in *European Conference on Computer Vision*, pp. 85–98, Springer, 2012.