Tackling Limited Sensing Capabilities for Autonomous Driving

by

Ming-Yuan Yu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2022

Ming-Yuan Yu
myyu@umich.edu
ORCID iD: 0000-0003-1926-1553

Table of Contents

List of Figures

List of Tables

Abstract

Limited sensing capabilities pose challenges to autonomous vehicles. Common limitations can result from a limited sensory range, occlusions, or adverse weather conditions, and hinders the development of both general and safe autonomy in real-world environments. In the pursuit of safe autonomy under imperfect measurements, an autonomous vehicle must assess the risk to take safe actions, and actively gather informative data to reduce the uncertainty. This thesis develops efficient algorithms for risk assessment and noise removal, specifically with Light Detection And Ranging (LiDAR), to aid the safe navigation of autonomous vehicles.

First, a probabilistic risk assessment algorithm is proposed, which utilizes scene geometry to quantify the risk imposed by occluded oncoming traffic. Simulations show that the quantified risk can then be used as an objective function of a subsequent planning algorithm to improve both safety and ride comfort at real-world urban intersections.

Second, the efficiency of the aforementioned algorithm is further enhanced by leveraging reachability analysis to constrain its sensory resources to only regions with potential collisions. This leads to an algorithm that not only quantifies the risk more efficiently, but also localizes the risk-inducing traffic participants in an environment. As a result, unlike the previous approach, the vehicle no longer needs to put equal attention to the entire scene and could focus only on essential information in collision avoidance.

Lastly, a real-time de-noising algorithm is proposed to remove noise from LiDAR point clouds corrupted by snow. This reduces uncertainty from adverse weather conditions. The proposed network outperforms state-of-the-art methods while also running orders of magnitude faster. The de-noised point clouds also show positive impacts on off-the-shelf Simultaneous Localization And Mapping (SLAM) algorithms, resulting in maps nearly free of noise while preserving salient information of the scene.

These algorithms are efficient and make safe autonomy feasible in the near future, even under imperfect LiDAR measurements.

Chapter 1

Introduction

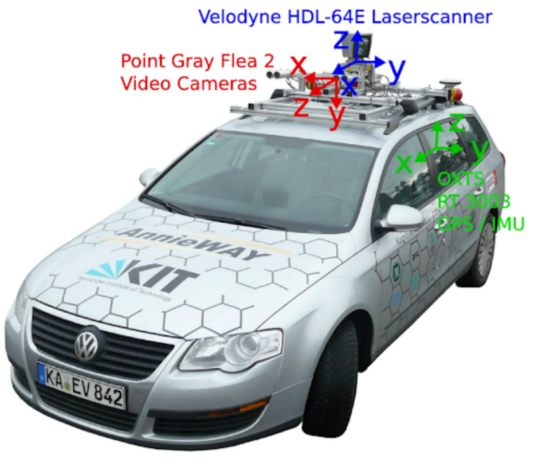
1.1 Limited Sensing Capabilities

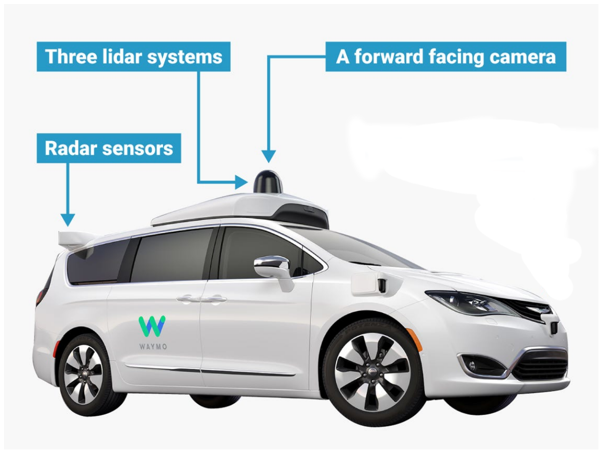
Visible cameras, radars, and LiDARs are widely adopted by modern autonomous vehicles, as shown in Figure 1.1. These sensors gather information from the environment, and enables an autonomous vehicle to perform tasks, such as object detection, tracking, and motion planning [1]–[3]. However, *all* sensors have limitations, and can impact the safety in autonomous driving.

Visible cameras gather photons in the visible spectrum within their field-of-view, and produce sequences of images. The resulting images closely represent what a human driver would see in various driving scenarios, and have been widely used for tasks including 2D object detection and semantic segmentation. One inherent limitation of visible cameras is that they are not able to see through opaque objects, like buildings, trees, and other vehicles. Moreover, they have very limited dynamic range compared to the human visual system, thus tend to be over/under-exposed in non-ideal lighting conditions.

Radars are standard equipment in many modern vehicles, and have been reliable in primitive collision avoidance systems. Although radars perform consistently in various lighting and weather conditions, their resolutions are usually too low to be suitable for more complicated tasks [4], [5].

LiDARs, by contrast, provide high-resolution (and usually panoramic) 3D information, and are more robust to various lighting conditions. Unlike visible cameras, LiDARs can work at night since they actively illuminate the environment with laser beams in the infrared spectrum. However, similar to visible cameras, LiDARs still can not see through opaque


Velodyne HDL-64E Laserscanner
Point Gray Flea 2 Video Cameras
OXTS RT 3003 GPS / IMU
(a) KITTI [1]



(b) Waymo [2]

Figure 1.1: The sensor suites on autonomous vehicles.

objects. In addition, they perform poorly in adverse weather conditions such as snow and heavy rain.

These limitations leave blind spots to an autonomous vehicle, and thus introduce risk and uncertainty to the system. The purpose of this thesis is to address such limitations, particularly with LiDARs, and mitigate their impact in order to develop fully autonomous systems in real-world environments.

1.2 Risk Assessment

A typical pipeline in decision making for an autonomous vehicle is as follows:

1. Gathering raw sensor readings.

2. Estimating the system's state and evaluating safety, comfort, or efficiency.

3. Taking an action, such as throttling or steering, to achieve the ultimate goal.

Risk assessment comes into play in all three stages. It is a process to quantify the risk for an event, such as a collision in the future, introduced by the state of the environment and the action of the ego vehicle.

Previous works tackle such problems by predicting observed agents, and estimating quantities like time-to-collision or the probability of colliding with another agent. However, this scheme alone does not work well in crowded or occluded environments. When dealing with unobserved agents and shadow regions, traditional algorithms usually assume the worst case scenario and result in very conservative actions. An autonomous vehicle may slow down unnecessarily, or even stop completely.

This thesis aims to develop efficient algorithms handling non-ideal sensory information, for example, occlusions or adverse weather conditions. In particular, the proposed algorithms quantify the risk induced by limited sensing capabilities, and utilize it to navigate safely and comfortably in a timely manner.

1.3 Reachability Analysis

Reachability analysis has been widely used for developing provably-safe control algorithms [6]–[12]. It involves predicting and analyzing the states of the environment, including the locations of other agents and the ego vehicle. Such approach first makes prediction of where others may be in the future, and generates target trajectories to avoid collision. However, generating the optimal safe trajectory in real-time in a crowded or occluded environment is often infeasible due to the number of agents and the amount of uncertainty.

Since collisions happen only in regions that the ego vehicle can physically reach in the future, one can reduce the amount of prediction to achieve significantly better efficiency. This involves incorporating the *backward* reachability.

By definition, a collision is an event of multiple agents occupying the same space at the same time. This implies the ego vehicle needs to focus primarily on its forward reachable set (FRS), and can backtrack risk-inducing agents that can actually reach the FRS by propagating their motions backwards in time. Such formulation alleviates the need to predict every agent in the scene, and results in a real-time risk-assessment algorithm that scales well in complex real-world environments.

1.4 Adverse Weather Conditions

Many state-of-the-art objection detection algorithms are data-driven, and are trained on large scale datasets. These datasets enable a deep neural network to capture detailed features of the input data, and often to perform better than human drivers. However, generalization

(a)

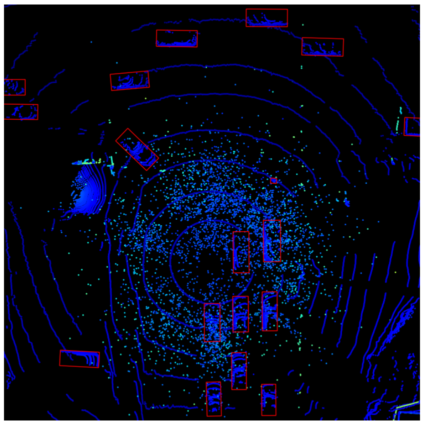

(b) Birds-eye-view of the point cloud data [13].

Figure 1.2: **Autonomous vehicles in snow**.

to various weather conditions and domain adaptation remain challenging.

In particular, snow poses challenges to LiDARs. A cloud of *ghost points* centered at the ego vehicle appears, as shown in Figure 1.2b, due to the reflection from snowflakes in the air. Moreover, deflection reduces the number of usable points and thus degrades the performance of tasks in autonomous driving.

We aim to mitigate such problem by proposing a deep convolutional neural network (CNN) to remove the ghost measurements in point clouds. The proposed network does not rely on any label thus can be trained with any noisy point clouds, taking advantage of large-scale unlabeled datasets. It outperforms current state-of-the-art methods while running $47\times$ faster.

1.5 Problem Statement

The motivation of the thesis is to tackle limited sensing capabilities in autonomous driving. In particular, this thesis aims to address the following questions:

1. How to quantify the risk induced by limited sensing capabilities for motion planning?
2. How to improve efficiency of risk assessment algorithms on large-scale and crowded environments?

3. How to mitigate noisy LiDAR data in adverse weather conditions?

1.6 Contributions

The remainder of this thesis is organized into individual chapters addressing each of the questions above. The specific contributions of this thesis are summarized below:

1. A risk assessment algorithm tailored to occluded urban environments is proposed. It is probabilistic and can easily be integrated into motion planning algorithms for lower collision rates and better ride comfort. (Chapter 2)

2. A risk assessment algorithm utilizing bidirectional reachability is proposed. It is efficient and can scale to large and complex environments. (Chapter 3)

3. A real-time de-noising algorithm is proposed to remove noise from point clouds corrupted by snow. (Chapter 4)

Chapter 2

Occlusion-Aware Risk Assessment for Autonomous Driving in Urban Environments

2.1 Introduction

Advancements in sensing technology and algorithmic improvements bring the reality of everyday autonomous driving closer to fruition. LiDAR sensors enable the construction of 3D maps [14] and can see tens or hundreds of meters away, even at night [15]. High definition cameras capture images that can be used for tasks such as semantic segmentation [16] and object detection [17]. Many tasks can be performed at levels surpassing that of humans thanks to recent developments in deep neural networks [18].

However, all sensors still have limited sensing capabilities. LiDARs and cameras, for instance, have difficulty identifying objects beyond a certain distance due to finite range, sensitivity, and angular resolution. In addition, both of these sensors can not see through opaque objects which could results in large unobserved regions. An illustration of such a scenario is shown in Fig. 2.1.

One of the reasons why human drivers can safely navigate even under occlusions is that they augment their sensing capabilities by leveraging semantic and geometrical information of the environment, and anticipate the need to slow down due to the potential risk of collision that arises due to occlusions [7], [19]. In addition, earlier braking would reduce the maximum deceleration which consequently leads to greater ride comfort.

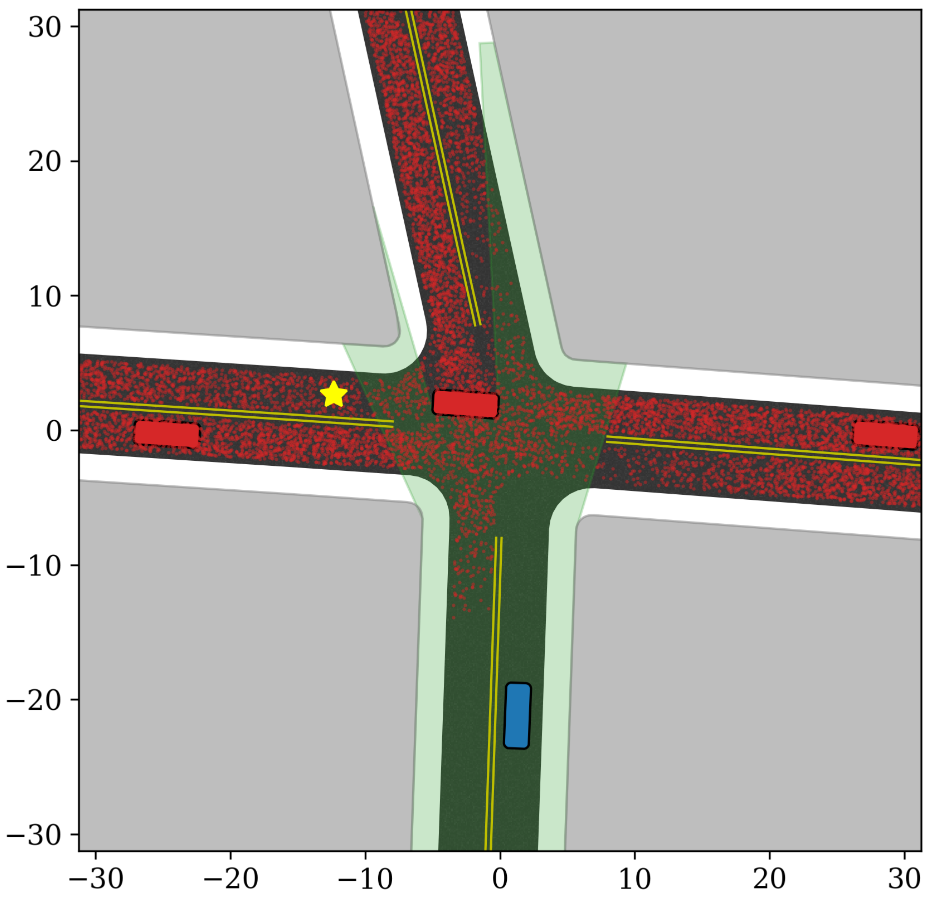


Figure 2.1: Ego vehicle (blue box) intends to perform an unprotected left turn to the goal (yellow star) at an intersection. The irregular shape of the observable polygon (green shaded region) is caused by 1) limited sensor range and 2) occlusions from other vehicles (red boxes) and buildings (gray regions.) Our algorithm quantifies the distribution of risk (red particles) posed by other vehicles including the ones which are outside of the observable polygon. This is possible under the assumption that we know the geometry of road layout and the nominal (or worst case) speed of other vehicles at this particular intersection. Both axes are in meters.

To provide an example from the real-world imagine the following: pulling up to a left turn next to a tall tree or building, similar to the scenario shown in Fig. 2.1. Typically a driver leans forward and pulls the car slightly ahead to see into oncoming traffic before completing the turn. In the driver's mind, they have a map of the unseen spaces and know that a car could emerge from beyond the current line of sight. As a result, they proceed cautiously to try to improve their visibility and do not turn until they can confirm a sufficient gap in the

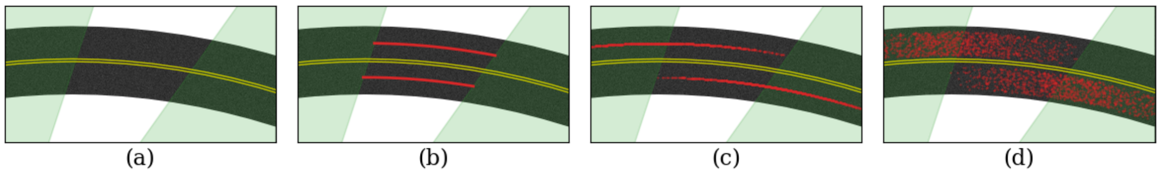


Figure 2.2: Illustration of our algorithm on a (a) partially observed road, where the green shaded regions are within the sensor's field-of-view. (b) Firstly, assuming that the map and the ego vehicle's location are known, the centerlines of the unobserved road segments (red) are extracted. (c) Secondly, we sample particles $\left\{(s^{[i]}, v^{[i]})\right\}_{i=1}^{N_k}$ along the extracted splines $c_k, k \in \{1, 2\}$, where the location s and speed v are drawn from uniform distributions, and propagate each particle forward in time assuming constant speed. (d) Finally, a random offset perpendicular to each centerline is added to each particle to incorporate the non-zero size of potentially unobserved vehicles.

traffic.

This chapter presents an algorithm that encodes this form of human driving by quantifying the risk caused by limited sensing capabilities and geometric occlusion. The proposed algorithm can be used to make autonomous vehicles navigate safely with improved ride comfort in urban environments, and it is agnostic to how the vehicle makes decisions.

The remainder of this chapter is organized as follows: Section 2.2 reviews related work in the field of risk assessment and planning under occlusion. Section 2.3 describes how our algorithm leverages the known road layout to quantify risk in the environment, and demonstrates how it can be easily integrated with a simple planning algorithm. Section 2.4 introduces a baseline (occlusion-unaware) method and our evaluation methodologies. Section 2.5 evaluates the proposed occlusion-aware method, and shows that statistically our algorithm performs significantly better in terms of collision rate and ride comfort on both synthetic and real-world intersections. Section 2.6 concludes and discusses future directions of this work.

2.2 Related Work

Most of the previous work on motion prediction and risk assessment can be categorized into one of two following categories. The first category quantifies the risk as the probability of having a collision with another vehicle or pedestrian. The second category assesses risk as the degree of deviation from a nominal set of behaviors (e.g. veering from the lane rapidly). A well-organized survey is given in [20]. This chapter addresses the first category of problems with a specific focus on collisions caused by occluded objects.

Prior work has addressed the issue of occlusion from a tracking perspective. Wyffels and Campbell [21] keep track of obstacles in occluded areas by utilizing negative information under the assumption that undetected objects are not likely to appear in visible space. Yu and LaValle [22] maintain the tracks of targets that move outside the field-of-view by formulating the problem as a pursuit evasion game. Galceran, Olson, and Eustice [23] augment states of a standard tracker to estimate occluded states for other agents and provide more robust data association when the occluded agents reappear in the scene. Although these models keep tracks of missing targets that enter occluded regions, they all need at least one detection to start tracking. They do not explicitly handle risks caused by potential incoming traffic which is occluded or outside the sensor horizon and thus never detected in the first place.

Partially Observable Markov Decision Process (POMDP) is a common approach to tackle decision making problems under uncertainty and consequently can implicitly handle probabilistic occlusion. Brechtel, Gindele, and Dillmann [24] use Monte Carlo Value Iteration and Brechtel, Gindele, and Dillmann [25] show it is possible to optimize a continuous POMDP model. Bouton, Nakhaei, Fujimura, *et al.* [26] approximate the global solution by solving a POMDP for each agent independently through utility fusion [27]. The reduction in state space required to make these approaches tractable limits their applicability, particularly for real-time high speed driving. The algorithm we present in this chapter differs in both goal and implementation. We focus on quantifying risk in the environment instead of the risk associated with the actions of the ego vehicle. Our algorithm is agnostic to planning and so could be coupled with a POMDP or any other planning or decision making algorithms.

Most closely related to the approach presented here are two risk quantification approaches [7], [19]. Orzechowski, Meyer, and Lauer [7] over-approximate all possible states of the incoming traffic by considering the leading edges of the visible polygon. Although safety is guaranteed, the resulting over-approximated polygons are not probabilistic, whereas our approach captures the full distribution of risk. Lee, Jo, and Sunwoo [19] perform probabilistic risk assessment by utilizing prebuilt high definition maps. While the results in [19] look promising, they do not show how their risk assessment could be used for planning to achieve safer driving. Furthermore, both [7] and [19] show very limited results with only a single additional vehicle in the scene and it is unclear how these approaches perform in crowded scenes such as urban intersections. We focus on realistic intersections derived from real map data and occupied with many vehicles.

Our approach presents several novel contributions: 1) we present an algorithm which performs probabilistic risk assessment of both observed and unobserved regions at urban intersections; 2) the approach is control algorithm agnostic and can be integrated with any deterministic or probabilistic planning approach; 3) we derive risk assessment from large-scale map data and extensively evaluate our approach with up to five other vehicles in the scene, and show significant reduction in collision rate and increase in ride comfort.

2.3 Method

We first describe our method in probabilistic risk assessment in Section 2.3.1. We generate a distribution over the Cartesian space. In Section 2.3.2 we show how to integrate the risk to a simple optimization-based planning algorithm and describe the primary cost function.

2.3.1 Risk Assessment Over Cartesian Space

High Definition (HD) maps are used commonly in autonomous driving [19]. These maps have rich data about intersections and can encode information such as nominal trajectories and maximum speed of all traffic through a region. Assuming that the map and the location of the ego vehicle are known, an *observable polygon* can be generated for a vehicle's sensor configuration (maximum range, angular resolution, and field of view) without the actual sensor returns. Here we focus on LiDARs, but the principles remain the same for other sensor modalities. The shape of the observable polygon is constrained by occlusions caused by objects such as other vehicles, trees, and buildings. With the observable polygon, one can identify free space at the current time. However, the current observable polygon alone can provide little information about long-term risk.

Current free space estimates are insufficient for planning for the future as vehicles can suddenly appear from regions outside of the observable polygon. In order to quantify the risk due to limited sensing in the context of long-term planning, we need to consider vehicles that are potentially hidden in unobserved regions. We leverage the paradigm of the particle filter to perform this prediction. Particles are used to represent the distribution of potential vehicle locations originated from unobserved regions. This approach was selected because of its simplicity and parallelizablility.

Consider a scenario with two lanes shown in Fig. 2.2. We represent the lanes of travel by cubic splines. Each cubic spline c_k is parameterized by its position s along the spline:

$$c_k(s) = \begin{bmatrix} x_k(s) \\ y_k(s) \end{bmatrix}, s \in [0, \overline{s}_k]$$

$$k \in \{1, 2, \dots, M\}$$

where $[x_k(s)\ y_k(s)]^\top$ is the position of a point on c_k at s, M is the number of lanes in the scene, and $\overline{s}_k$ is the total length of the spline c_k.

We first extract all possible centroids for all valid vehicle positions in all lanes of travel in the unobserved regions. On each spline c_k, we consider L_k disjoint unobserved segments. A set of N_k particles $\left\{(s^{[i]}, v^{[i]})\right\}_{i=1}^{N_k}$ are sampled independently from uniform distributions in these unobserved segments, where $s^{[i]}$ and $v^{[i]}$ are the position and speed of the i-th particle. The position s and speed v is distributed as follows:

$$s \sim U\left(\bigcup_{j=1}^{L_k} [\underline{s}_j, \overline{s}_j]\right)$$

$$v \sim U\left([\underline{v}_k, \overline{v}_k]\right)$$

where $[\cdot, \cdot]$ is a closed set between two real numbers, $U(\cdot)$ is an uniform distribution on a set, $\underline{s}_j$ and $\overline{s}_j$ is the starting and ending position of an unobserved segment j on spline c_k, $\underline{v}_k$ and $\overline{v}_k$ are the minimum and maximum speed of other vehicles, respectively. Assuming that each particle is traveling with a constant speed, we can then propagate all the particles forward in time for T_f seconds:

$$\hat{s}^{[i]} = s^{[i]} + v^{[i]} \cdot T_f$$

where $\hat{s}^{[i]}$ is the position of the i-th particle after T_f seconds. This results in a distribution of particles along the centerline of each lane, as shown in Fig. 2.2c.

To account for the size of vehicles and lateral displacements within the lane, an offset $b^{[i]}$ is sampled from an uniform distribution and added to each particle in Cartesian space perpendicular to the spline.

$$b \sim U\left([-\overline{b}, \overline{b}]\right)$$

$$u_k(s) := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \frac{\partial c_k}{\partial s}(s)$$

$$\hat{p}_k^{[i]} = c_k\left(\hat{s}^{[i]}\right) + \frac{b^{[i]}}{\left\|u_k\left(\hat{s}^{[i]}\right)\right\|_2} \cdot u_k\left(\hat{s}^{[i]}\right)$$

where $\|\cdot\|_2$ is the 2-norm of a vector, u_k is the unnormalized vector perpendicular to c_k, and $\overline{b}$ is the maximum deviation among all the particles from their corresponding centerline. We define the set $\left\{\hat{p}_k^{[i]}\right\}_{i=1}^{N_k}$ to be the distribution of risk over the Cartesian space on lane k after T_f seconds, as shown in Fig. 2.2d.

For observed vehicles, we model them as rectangles along valid lanes. To incorporate them into our proposed formulation, we treat them similarly. For each vehicle, we extract the spline segments within the corresponding rectangle, and apply the aforementioned method as if the segments are in the unobserved regions.

The overall distribution of risk $\left\{\hat{p}^{[i]}\right\}_{i=1}^{N}$ is simply the union of all sets.

$$\left\{\hat{p}^{[i]}\right\}_{i=1}^{N} = \bigcup_{k=1}^{M} \left\{\hat{p}_k^{[i]}\right\}_{i=1}^{N_k}$$

where N is the total number of particles in the scene on M lanes.

This risk over Cartesian space can be easily integrated with any control or planning algorithm as either the primary cost or in conjunction with other costs as an auxiliary cost function. In addition, it can also be used along with any existing risk assessment method designed for only observed vehicles. In Section 2.3.2 we show how we can utilize the risk $\left\{\hat{p}^{[i]}\right\}_{i=1}^{N}$ as the major cost function of a simple optimization-based planning algorithm.

2.3.2 Planning

In this subsection we demonstrate how the risk described in Section 2.3.1 can be used in practice. We integrate it into a optimization-based planning algorithm and show improvements in both safety and ride comfort. The rudimentary planner used here can be replaced by any other cost-based planners as the technique is agnostic to planning approach.

Assuming that at time t the ego vehicle travels with speed v_{ego} on the intended route c_{ego}, which is also a cubic spline parameterized by its position s_{ego} along the spline. The planner

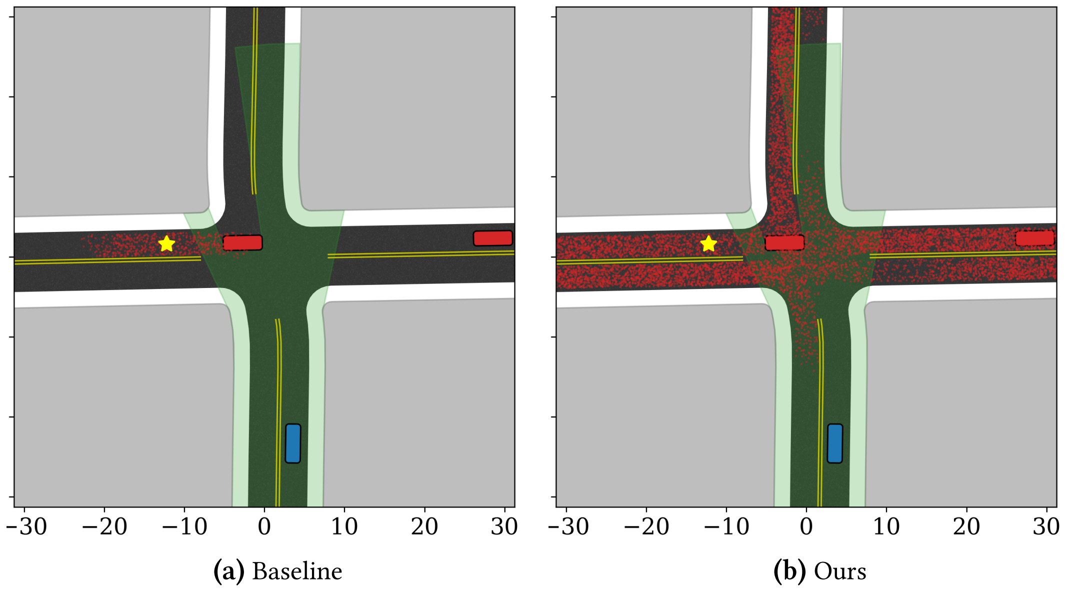


Figure 2.3: Comparison between the (a) baseline and (b) proposed method. The baseline method only predicts distribution of risk (red particles) caused by observed vehicles, whereas the proposed method also predicts the risk caused by unobserved regions.

first considers the *safety cost* $J_1(a_{ego})$ associated with an acceleration (or deceleration) a_{ego}:

$$J_1(a_{ego}) = \sum_{i=1}^{N} f^{[i]}(a_{ego})$$

where $f^{[i]}$ is the potential function of the i-th particle which is positive when the particle $\hat{p}^{[i]}$ is within the ego lane and zero otherwise. The function $f^{[i]}$ is defined as follows:

$$f^{[i]}(a_{ego}) := \begin{cases} \exp\left(-\frac{r^{[i]}(a_{ego})^2}{\sigma^2}\right) & \text{, if } d^{[i]} \leq \bar{b} \\ 0 & \text{, otherwise} \end{cases}$$

$$\hat{s}_{ego} := s_{ego} + v_{ego} \cdot T_f + \frac{1}{2} a_{ego} \cdot T_f^2$$

$$r^{[i]}(a_{ego}) := \left\| c_{ego}\left(\hat{s}_{ego}\right) - \hat{p}^{[i]} \right\|_2$$

$$d^{[i]} := \inf_s \left\| c_{ego}(s) - \hat{p}^{[i]} \right\|_2$$

where $\hat{s}_{ego}$ is the future position of the ego vehicle along c_{ego}, $r^{[i]}(a_{ego})$ is the distance between particle $\hat{p}^{[i]}$ and $c_{ego}\left(\hat{s}_{ego}\right)$, $d^{[i]}$ is the minimum distance between particle $\hat{p}^{[i]}$ and the ego vehicle's intended route c_{ego}, and σ is the bandwidth of the repulsive potential field. In practice, particles with $r^{[i]}(a_{ego}) \geq 2\sigma$ are discarded to speed up the calculation.

In addition to the safety cost $J_1(a_{ego})$, a *speed cost* $J_2(a_{ego})$ is also considered to drive the ego vehicle to meet the desired speed v_{des}.

$$J_2(a_{ego}) = \left|v_{ego} + a_{ego} \cdot T_f - v_{des}\right|$$

where $|\cdot|$ is the absolute value of a scalar. The optimal acceleration between time t and $t + T_p$ can be found by solving the following optimization problem:

$$\begin{aligned} \min_{a_{ego}} \quad & J_1(a_{ego}) + \lambda \cdot J_2(a_{ego}) \\ \text{s.t.} \quad & \underline{v}_{ego} \leq v_{ego} + a_{ego} \cdot T_f \leq \overline{v}_{ego} \\ & \underline{a}_{ego} \leq a_{ego} \leq \overline{a}_{ego} \end{aligned}$$

where T_p is the replan time, λ is the weight affecting how aggressive the ego vehicle behaves, $\underline{v}_{ego}$ and $\overline{v}_{ego}$ are the minimum and maximum speed of the ego vehicle, and $\underline{a}_{ego}$ and $\overline{a}_{ego}$ are the maximum deceleration and maximum acceleration, respectively. Note that a smaller λ favors more conservative behaviors. As shown here, the proposed risk assessment method can be incorporated with any optimization-based planner.

2.4 Evaluation

To demonstrate how safety and comfort can be improved by our algorithm, we compare it to a baseline approach which only models observed vehicles at intersections. In particular, we focus on scenarios where the ego vehicle tries to make difficult maneuvers such as an unprotected left turn.

2.4.1 Simulation

We simulate various random scenarios with five other vehicles in the scene. Each vehicle travels on a random route at a constant speed ranging from 4 to 12 m/s. A valid combination of trajectories is generated by rejection sampling so that there is no collision or overlap

Table 2.1: Parameters for Simulations

Parameter	Value
Forecast horizon, T_f	$1.5\ s$
Replan period, T_p	$0.1\ s$
Vehicle length, l_v	$4.88\ m$
Vehicle width, w_v	$1.86\ m$
Number of particles, $N_k\ \forall k$	$\leq 2^{15}$
Weight, λ	$2^{14} \cdot 10^{-6}$
Bandwidth, σ	$0.5l_v$
Max. offset, $\overline{b}$	$0.75w_v$
Desired speed, v_{des}	$10\ m/s$
Min. speed, $\underline{v}_k = \underline{v}_{ego}\ \forall k$	$0\ m/s$
Max. speed, $\overline{v}_k = \overline{v}_{ego}\ \forall k$	$12\ m/s$
Min. acceleration, $\underline{a}_{ego}$	$-8\ m/s^2$
Max. acceleration, $\overline{a}_{ego}$	$2.5\ m/s^2$
Threshold acceleration, a_{thresh}	$4.0\ m/s$

among the simulated vehicles.

Here we focus on four-way, un-signaled intersections for compactness and not on T- or Y-junctions, but the proposed approach conceptually generalizes. The layout of intersections can be either synthetic or from real-world map data. For the synthetic layout, the roads are constructed using straight and perpendicular segments. For real-world layouts, we obtain the geometry information from 73 real-world intersections extracted from OpenStreetMaps (OSMs) around Ann Arbor, Michigan.

To simulate scenarios with heavy occlusion, buildings are added to the map with a 2 m buffer from the boundary of the driving surface. The ego vehicle starts 15 m before the stopline with initial speed 10 m/s, and tries to perform an unprotected left turn, as shown in Fig. 2.3. More details of the parameters used in the simulator are listed in Table 2.1. Note that as length of unobserved segments vary, N_k is calculated dynamically such that the density of particles stays constant at 2^{15} particles per 100 m.

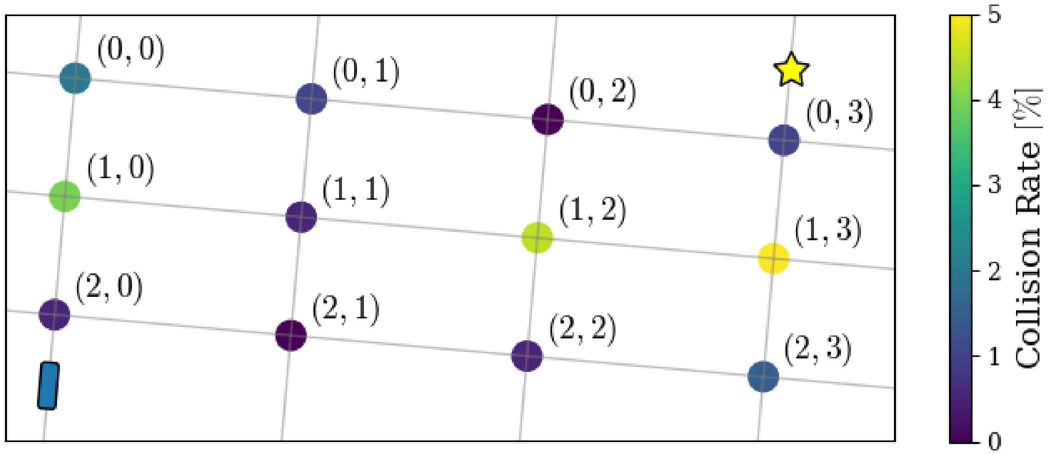


Figure 2.4: Illustration of collision rates overlaid on a map with 12 intersection. A high-level planner can plan a route based on the collision rates, taking the route with the lower collision rates: $(2,0) \rightarrow (2,1) \rightarrow (1,1) \rightarrow (0,1) \rightarrow (0,2) \rightarrow (0,3)$.

2.4.2 Baseline

An occlusion-unaware risk assessment method is used as a baseline for comparison. The baseline method only predicts distribution of risk caused by observed vehicles, as shown in Fig. 2.3a. The same planning algorithm described in 2.3.2 is used with both the baseline and proposed method throughout all simulations.

2.4.3 Metrics

We first simulate 2000 random scenarios with the ego vehicle performing an unprotected left turn at each intersection with the baseline method, then simulate the exact same set of experiments with identical trajectories with the proposed method. For each intersection, we calculate its collision rate for both methods as follows:

$$\text{Collision Rate} = \frac{\text{\# of simulations with collision}}{\text{Total \# of simulations}} \times 100\%$$

This meta-collision rate for a given intersection can be used by a high-level planner, which needs to plans a route between two points across a city. Overlaying the collision rates with associated intersections, a high-level planner can avoid dangerous intersections, as show in Fig. 2.4.

Speed and acceleration profiles are also calculated to quantify ride discomfort. However, to the best of our knowledge, there is no common computational metric in the literature for ride comfort. Typically, the literature reports thresholds on acceleration and jerk as the metric for ride comfort[28]. We define the following *discomfort score* to represent a continuous range of discomfort.

$$\text{Discomfort Score} = \frac{1}{T}\int_0^T g(t)dt$$

where

$$g(t) = \begin{cases} -a_{thresh} - a_{ego}(t) & \text{, if } a_{ego}(t) \leq -a_{thresh} \\ -a_{thresh} + a_{ego}(t) & \text{, if } a_{ego}(t) \geq a_{thresh} \\ 0 & \text{, otherwise} \end{cases}$$

and T is the duration to reach the goal. Note that the function g outputs positive values when more than half of the braking force is applied and zero otherwise.

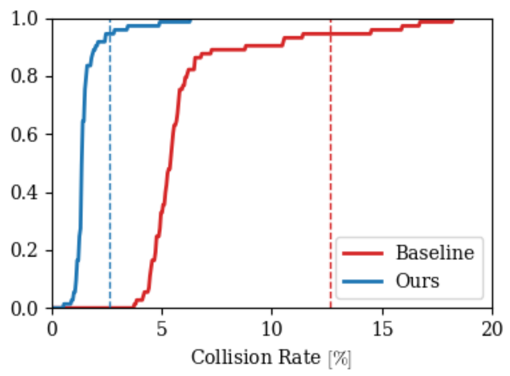
2.5 Results

The distributions of both collision rate and discomfort score are discussed in this section, as shown in Fig. 2.5a and 2.5b as CDFs. In particular, the median and the 95th percentile are reported. The former describes the nominal behavior, whereas the latter represents the near-worst case.

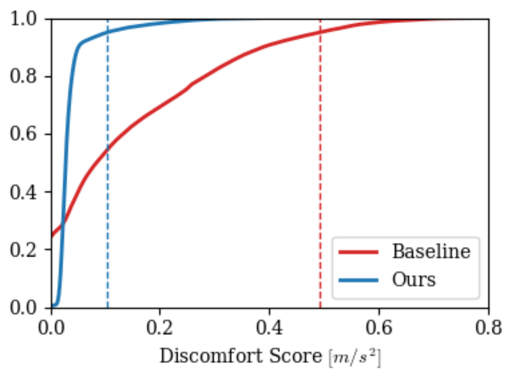
2.5.1 Collision Rate

We first evaluate the collision rate of both the baseline and proposed method. By utilizing our algorithm, collision rates drop significantly. At the synthetic intersection, the rate decreases by $4.1\times$, from 5.75% to 1.40%. Results at real-world intersections also show similar improvements. Among all the 73 real-world intersections, the median collision rate decreases by $3.7\times$, from 5.40% to 1.45%, and the 95th percentiles decreases by $4.8\times$, from 12.64% to 2.61%, as shown in Fig. 2.5a.

The distribution of collision rates of the 73 real-world intersections is overlaid in Fig. 2.6. This can be added as extra information to a high-level planner to plan a route with lower collision rates. This enables a vehicle to reason about safety prior to embarking on a journey.



(a) CDFs of collision rates



(b) CDFs of discomfort scores

Figure 2.5: CDFs of (a) collision rates and (b) discomfort scores among all 73 real-world intersections. Our method outperforms the baseline where the 95th percentile of the collision rate (vertical dashed lines) decreases from 12.64% to 2.61%, and the 95th percentile of the discomfort decreases from 0.4925 to 0.1043.

It also enables urban planners and civil engineers to reason about safety of interactions for autonomous vehicles in a systematic and quantitative way.

2.5.2 Ride Comfort

In addition to safety, which is evaluated as collision rate in Section 2.5.1, another benefit from our algorithm is more ride comfort. At the synthetic intersection, the median discomfort score is reduced by $2.9\times$, from 0.0795 to 0.0271. The 95th percentile of the discomfort score decreases by $10\times$, from 0.4687 to 0.0466.

Similarly, the median score among all real-world intersections is reduced by $3\times$, from 0.0849 to 0.0284, and the 95th percentile of the score decreases by $4.7\times$, from 0.4925 to 0.1043.

An illustration of the synthetic and real-world intersections are shown in Fig. 2.7. In both cases, the baseline method has larger variations in both speed and acceleration, which means that the ego vehicle can abruptly brake only after other vehicles enter the observable polygon. On the other hand, our method naturally introduces a *virtual* stop sign, slowing the ego vehicle down even when there is no other vehicle in the observable polygon, which generates a consistent behavior across all simulations.

2.6 Conclusion

We propose a probabilistic risk assessment algorithm for autonomous driving under occlusion. We show how it can be integrated with a simple planning algorithm, and compare the proposed algorithm with a baseline method which only performs risk assessment for observed vehicles. We evaluate our algorithm in terms of collision rate and ride comfort with a large number of simulations at one synthetic and 73 real-world intersections. The results show that the proposed algorithm reduces the collision rate by up to $4.8\times$ and increase comfort by up to $10\times$. Our method shows great potential for improving both safety and comfort for autonomous driving in urban environments.

Future directions include tackling scenarios with aggressive vehicles hidden in the unobserved regions, and incorporating information from other vehicles for cooperative planning.

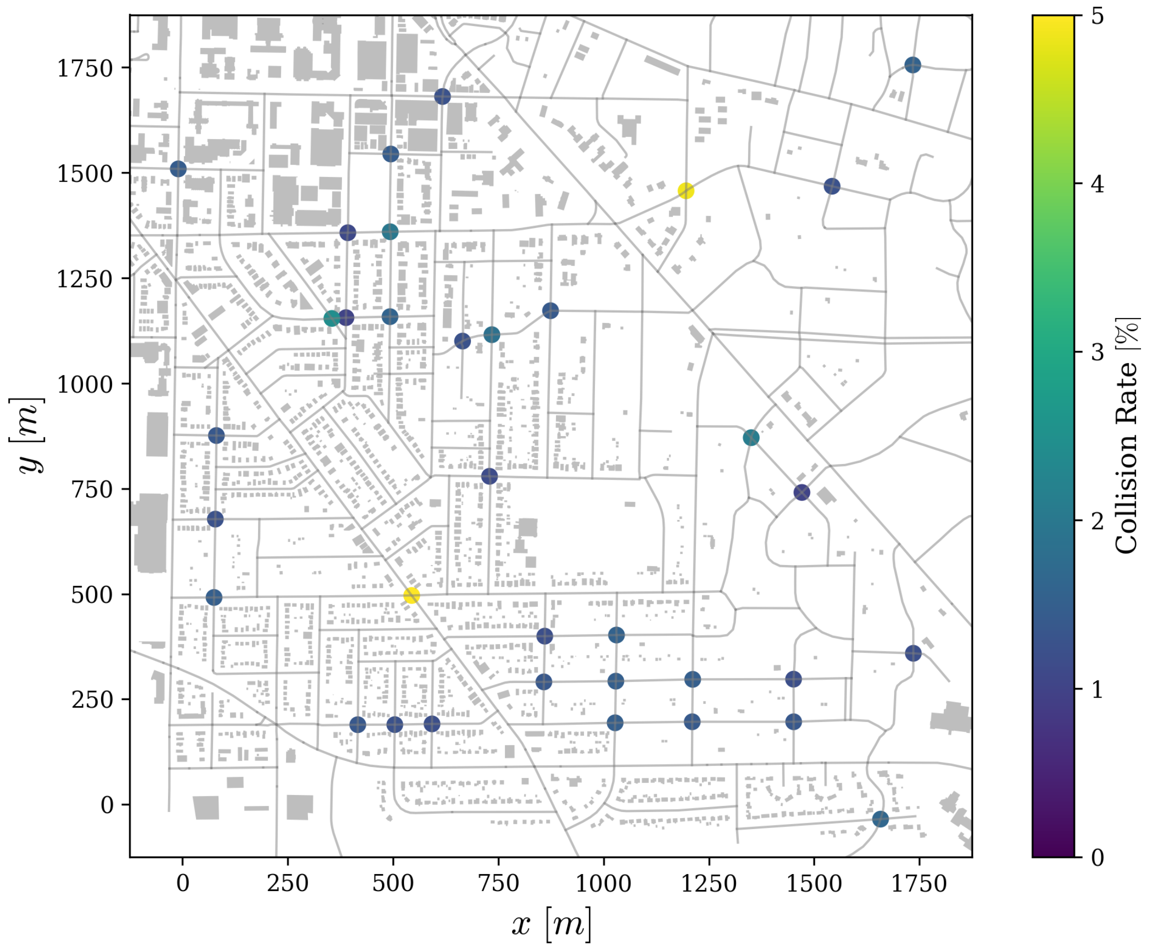
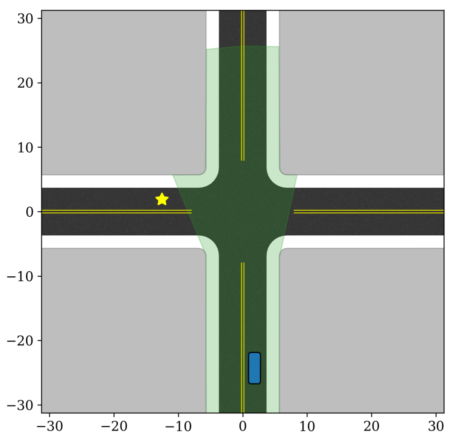
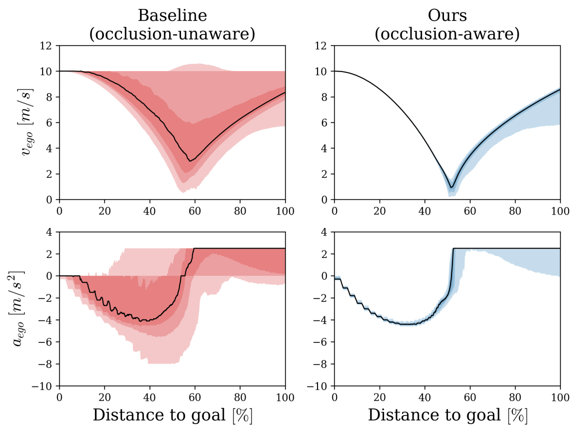
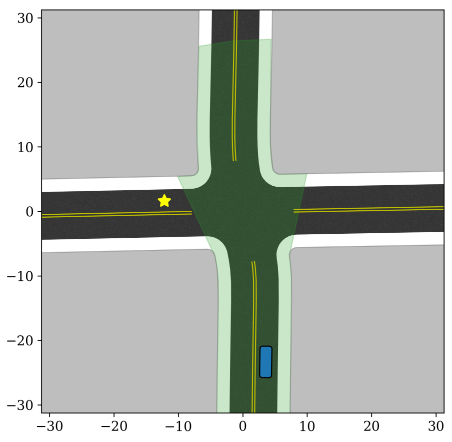


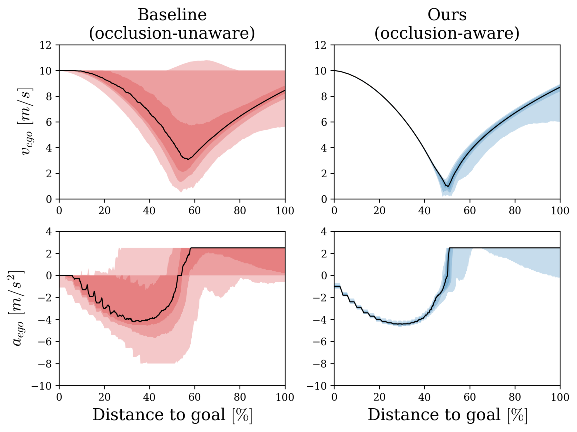
Figure 2.6: Collision rates of a subset of intersections overlaid on a real-world map. A high-level planner can utilize this information to avoid dangerous intersection such as the two yellow ones.


(a) Synthetic Intersection Layout *[m]*



(b) Synthetic Intersection Results


(c) Real-world Intersection Layout *[m]*



(d) Real-world Intersection Results

Figure 2.7: Speed and acceleration profiles for the baseline and proposed method at both (a) the synthetic intersection and (c) one of the real-world intersection. The initial location of the ego vehicle (blue box) and the goal (yellow star) are also shown. The left column (red) shows the profiles of the baseline (occlusion-unaware) method, and the right column (blue) shows profiles for the proposed (occlusion-aware) method. The medians of profiles are shown in solid black lines, and the percentiles are shown in different shades of colors (from dark to light: $50 \pm 15\%$, $50 \pm 30\%$, $50 \pm 45\%$.) In both synthetic and real-world intersections, the baseline method shows large variations due to abrupt braking. In addition, the deceleration can reach down to $-8\ m/s^2$, which can be very uncomfortable. On the other hand, our method predominantly stays above $-4\ m/s^2$ and shows smaller variations, which indicates that it performs consistently well across all simulations.

Chapter 3

Risk Assessment and Planning with Bidirectional Reachability for Autonomous Driving

3.1 Introduction

Most recent autonomous vehicles are equipped with a full sensor suit, including radars, cameras, and LiDARs [29]. These sensors give the vehicle the ability to perceive the world and to assess the risk of taking a certain action. The quantified risk can then be passed to a planning algorithm to eventually take a action. However, it is still challenging for an autonomous vehicle to navigate through complex environments such as occluded urban intersections such as the one shown in Figure 3.1.

The primary challenge of safely navigation through these occluded scenarios is predicting the behavior of all possible incoming traffic. The challenge arises due to the size of the state space of all possible incoming traffic from every occluded region which can make computation intractable.

Traditionally only forward reachability is used for risk assessment. In other words, algorithms make predictions of all other agents' future trajectories, and select an optimal action based on collision probability [19] and time-to-collision [30].

In contrast, human drivers can usually handle scenarios like the one in Figure 3.1 even with relatively limited sensing capabilities, such as a narrow field-of-view. We argue that this is because human drivers sense the world in a more efficient way: using semantic and prior information to focus sensing resources on portions of the scene that are most pertinent. For instance, when driving normally, drivers focus mostly on the current and future location of the ego vehicle. We only glimpse to check the rear view mirror when driving forward, but never focus our attention on it for an extended period of time. This is due to the fact that we have 1) a semantic understanding of the environment (e.g. road layout, speed limit, etc) and 2) prior knowledge such as the ability to estimate the reachability of not only for the ego vehicle but also other agents. As a result, we know to focus our limited sensing resources on only the regions where the ego vehicle can reach, and properly assess the risk to take an action.

This chapter proposes an algorithm that leverages this intuition from human driving for probabilistic risk assessment and planning by utilizing not only forward but also backward reachability. Since collisions can only happen within the FRS of a vehicle, it is unnecessary to predict everything in the occluded region. Instead, staring from a location in the FRS of the ego vehicle, one can backward propagate the motion of other vehicles and eventually get all possible initial conditions of risky agents. These initial conditions quantifies and spatially identifies risk-inducing regions and agents, as shown in Figure 3.1. With this information, we show how this quantified risk can be used for motion planning for safe navigation in urban environments.

This chapter is organized as follows. Section 3.2 reviews related work in reachability-based planning and risk assessment under occlusion. Section 3.3 shows our problem formulation and proposed risk assessment and motion planning algorithm based on forward and backward reachability. Section 3.4 describes two baselines and our evaluation metrics. Section 3.5 describes the quantitative evaluation of the proposed algorithm and illustrates that the proposed method is a significant improvement over existing methods. Section 3.6 concludes and discusses future directions of this work.

3.2 Related Work

Risk assessment is the process of predicting and quantifying the risk of taking a certain action. The proposed method belongs to the category of methods that quantify the risk of having a collision with another agent in the scene as defined in the well-organized survey by Lefèvre, Vasquez, and Laugier [20].

Reachability-based motion planning is widely used for autonomous driving [6]–[12]. Manzinger and Althoff [6] assume cooperative agents and proposed an algorithm to negotiate conflicting motions. They try to solve a winner determination problem, which is NP-hard, by restricting the set of combinations to a tree structure. This makes the original problem computationally tractable. Orzechowski, Meyer, and Lauer [7] over-approximate the FRS of incoming traffic, including those who are occluded, by only considering the leading edge of the observable polygon and make sure that the ego vehicle does not collide with the FRS. Koschi and Althoff [8] use forward reachability to predict future both the FRS of the ego vehicle and the ones of other traffic participants. It performs collision checking of the FRSs and is evaluated on both urban intersection and highway scenarios. Ahn, Berntorp, and Di Cairano [9] use both forward and backward reachability, just like the proposed method, to formulate a discrete decision making problem for autonomous driving. Vaskov, Sharma, Kousik, *et al.* [11] propose Reachability-based Trajectory Design (RTD) algorithm for real-time trajectory planning in a static scene by showing how to utilize a FRS computed offline during safe online motion planning. Vaskov, Kousik, Larson, *et al.* [12] later extended the idea [11] to scenes with dynamics agents.

Our approach differs in the following ways. First of all, we formulate the problem probabilistically, whereas the approaches developed by Orzechowski, Meyer, and Lauer [7], Koschi and Althoff [8], Ahn, Berntorp, and Di Cairano [9], Vaskov, Sharma, Kousik, *et al.* [11], and Vaskov, Kousik, Larson, *et al.* [12] do not. The proposed algorithm captures the distribution of risk in both the action space and Cartesian space thus can be more flexible and gives one to potentially adjust the balance between aggressiveness and effectiveness. Secondly, the work by Ahn, Berntorp, and Di Cairano [9] is intended to be used as a high-level planner for switching between behaviors and thus relies on a low-level planner to generate throttle and steering commands. By contrast, our approach is more like a controller and plans low-level commands directly. Finally, methods by Manzinger and Althoff [6], Ahn, Berntorp, and Di Cairano [9], Vaskov, Sharma, Kousik, *et al.* [11], and Vaskov, Kousik, Larson, *et al.* [12] do not consider occlusions or other sensory limitations, whereas our approach is designed to operate even under occlusions.

The algorithm proposed in this chapter builds upon our previous work [31], which is also a risk assessment and motion planning algorithm. One weakness of our prior work is its poor scalability with respect to the size of the environment. The computational cost is proportional to the area of the occluded regions within the size of the map. The cost of the new approach proposed in this chapter does not scale with the area of occluded regions and has a lower collision rate. Also, our previous approach does not identify where to focus the sensory resource.

The main contributions of this chapter are as follows:

- We propose a probabilistic risk assessment method which identifies the location of the risk-inducing agents and regions.
- The proposed method utilizes both forward and backward reachability for efficient risk assessment.
- We show how the quantified risk can be used for motion planning for navigating safely in urban environments.
- We evaluate our method on 73 real-world intersections and show quantitative improvement in terms of reduced collision rate when compared to the state of the art.

3.3 Method

We first define both forward and backward reachability in Section 3.3.1. In Section 3.3.2, we show how to use aforementioned reachability for efficient risk assessment. Finally, in Section 3.3.3, we demonstrate using the quantified risk for motion planning.

3.3.1 Bidirectional Reachability

Let $t \in \mathbb{R}$ be time, $\mathcal{X}$ be the state space and $\mathcal{U}$ be the action space. Let $x(t) \in \mathcal{X}$ be the state and $u(t) \in \mathcal{U}$ be the control input of a dynamic system at time t. Assume the vehicle's dynamics are described as an ordinary differential equation (ODE):

$$\dot{x}(t) = f(t, x(t), u(t)), \tag{3.1}$$

where f is a Lipschitz continuous function.

The FRS describes a set of all possible states of a system in the future. Given the dynamics in (3.1), an initial set $\mathcal{X}_0$ and a fixed time horizon T, the FRS is defined as

$$\begin{aligned} \text{FRS}(\mathcal{X}_0, T) := \{x(T) \in \mathcal{X} \mid x_0 = x(0) \in \mathcal{X}_0, u(t) \in \mathcal{U}, \\ \dot{x}(t) = f(t, x(t), u(t)), \forall t \in [0, T]\} \end{aligned} \tag{3.2}$$

On the other hand, the backward reachable set (BRS) describes all possible initial states that are able to reach a target set. Given the dynamics in (3.1), a target set $\mathcal{X}_f$ and a fixed time horizon T, the BRS is defined as

$$\begin{aligned} \text{BRS}(\mathcal{X}_f, T) := \{x(0) \in \mathcal{X} \mid x_f = x(T) \in \mathcal{X}_f, u(t) \in \mathcal{U} \\ \dot{x}(t) = f(t, x(t), u(t)), \forall t \in [0, T]\} \end{aligned} \tag{3.3}$$

Normally, reachable sets are calculated for each observed agent [11], [12]. These methods usually perform well when only a limited number of agents are presented in a non-occluded scene. Scenes like urban intersections can still be challenging for these method since predicting *everything*, especially in a occluded environment, can computationally intensive in many cases. Efforts have been made to ease the calculation under occlusion [7], [31], but the complexity still grows along with either the number of occluded regions [7] or the total area of the occluded regions [31]. For this reason, in Section 3.3.2 we introduce multiple stages of our algorithm that utilize both forward and backward reachability for efficient risk assessment under occlusion. The complexity of proposed method stays constant no matter how the number or size of the occluded regions grow.

3.3.2 Risk Assessment

For autonomous driving, collisions happen only at the intersections of the FRSs of different agents. In other words, if another agent (occluded or not) cannot reach where the ego vehicle can possibly be in the future, 1.5 seconds for example, it does not induce risk to the ego vehicle. This observation leads to an important insight – it is unnecessary to predict the behavior of everything in a scene. Instead, we can focus resources on identifying only those agents who pose a risk to the ego vehicle.

The proposed algorithm, whose behavior is depicted in Figure 3.2, is performed iteratively every T_r seconds, and has a forecast horizon of T_f seconds. It can be broken down into three stages.

Stage 1 – FRS of the Ego Vehicle

In the first stage, the FRS of the ego vehicle is calculated. In particular, we borrow the idea of using particles from our previous work [31], [32] to represent the FRS since it makes the

FRS probabilistic and the calculation parallelizable.

Without loss of generality, let $t = 0$ at the beginning of each iteration. Let the control inputs be parameterized by some parameters θ_e and θ_o such that

$$\begin{aligned} u_e(t) &= u_e\left(t; \theta_e\right) \in \mathcal{U} \quad \forall t \\ u_o(t) &= u_o\left(t; \theta_o\right) \in \mathcal{U} \quad \forall t, \end{aligned} \tag{3.4}$$

where the subscripts e and o indicate quantities tailored to the ego vehicle and other vehicles, respectively. Let the $i-$th particle, $p^{[i]}$, be defined as

$$p^{[i]} := \left\{x_e^{[i]}(0), \theta_e^{[i]}(t), \theta_o^{[i]}(t), T^{[i]}\right\} \quad \forall i = 1, \dots, N, \tag{3.5}$$

where the superscript $[i]$ indicates $i-$th sample of a variable, $x_e(0)$ is ego vehicle's initial state, T is a fixed time horizon. and N is the total number of particles.

In each iteration, we first sample N particles from the following distributions:

$$\begin{aligned} x_e(0) &\sim U\left(\mathcal{X}_{e,0}\right) \\ \theta_e &\sim P\left(\theta_e^+ \mid \theta_e^-\right) \\ \theta_o &\sim P\left(\theta_o^+ \mid \theta_o^-\right) \\ T &\sim U\left(\left[0, T_f\right]\right), \end{aligned} \tag{3.6}$$

where $\mathcal{X}_{e,0}$ is the initial set of the ego vehicle, $[\cdot, \cdot]$ is a closed interval, $U(\cdot)$ is an uniform distribution over a set, and $P(\cdot^+ \mid \cdot^-)$ is the transition probability of a quantity between subsequent iterations. The transition probabilities enforces smoothness in the action space $\mathcal{U}$ which reduces jittery motion. Since the smoothness is enforced on the distribution level but not directly on the optimal value, actions such as emergency breaking are still allowed. More details about determining the optimal control input from the distribution are described in Section 3.3.3.

Now the probabilistic FRS of the ego vehicle can be re-written as

$$\begin{aligned} \text{FRS}_e := \Big\{ x_e\left(T^{[i]}\right) \in \mathcal{X} \mid x_e(0) &= x_{e,0}^{[i]} \in \mathcal{X}_{e,0} \\ \dot{x}_e(t) &= f_e\left(t, x_e(t), u_e^{[i]}(t)\right) \\ u_e^{[i]}(t) &\in \mathcal{U}, \forall t \in \left[0, T^{[i]}\right] \Big\}_{i=1}^{N} \end{aligned} \tag{3.7}$$

The set FRS_e can be easily calculated by solving N initial value problems (IVPs) in parallel with a regular ODE solver. Note that FRS_e contains N hypotheses of the future states of the ego vehicle, as shown in Figure 3.2b.

Stage 2 – BRS of Other Agents

In the second stage of the proposed algorithm, we calculate the BRS of other agents in a similar fashion by setting the target set $\mathcal{X}_f$ to FRS_e such that

$$\begin{aligned} \text{BRS}_o := \Big\{ x_o(0) \in \mathcal{X} \mid \; & x_e\left(T^{[i]}\right) \in \text{FRS}_e \\ & \mathcal{P}_o\left(x_o\left(T^{[i]}\right)\right) = \mathcal{P}_e\left(x_e\left(T^{[i]}\right)\right) \\ & \dot{x}_o(t) = f_o\left(t, x_o(t), u_o^{[i]}(t)\right) \\ & u_o^{[i]}(t) \in \mathcal{U}, \forall t \in \left[0, T^{[i]}\right] \Big\}_{i=1}^{N}, \end{aligned} \tag{3.8}$$

where $\mathcal{P}_o : \mathcal{X}_o \to \mathbb{R}^2$ and $\mathcal{P}_e : \mathcal{X}_e \to \mathbb{R}^2$ are invertible functions projecting the state spaces into Cartesian space. We use projections since collisions occur when a subset of the states (e.g. positions) overlap. In this case, collision happens at one point in the Cartesian space, while other states such as heading and velocity do not necessary have to be the same and should be assigned separately.

Unlike the FRS_e, the BRS_o set cannot be calculated directly with an ODE solver. Instead, we need to first reformulate the final value problem into an equivalent IVP. For Lipschitz continuous dynamic systems such as Constant Turn Rate and Velocity (CTRV) model, Constant Turn Rate and Acceleration (CTRA) model, and the bicycle model, the initial value $x_o(0)$ corresponding to particle i can be obtained by solving the following IVP:

$$\begin{aligned} \dot{z}(t) &= -f_o\left(t, z(t), u_o^{[i]}(t)\right) \\ z(0) &= x_e\left(T^{[i]}\right) \end{aligned} \tag{3.9}$$

The set BRS_o can now be obtained by solving (3.9) in parallel with a regular ODE solver, and the solution $z\left(t\right)$ is $x_o(t)$ in (3.8), but reversed in time.

BRS_o represents a collection of risky initial conditions of other agents. Each particle corresponds to a possible collision between the ego vehicle and another agent by design, since the target set of BRS_o is chosen to be FRS_e. By doing so, this approach can handle occlusion in an efficient way. We no longer need to predict all possible agents in the scene. Instead, the set BRS_o originates only from $\mathcal{X}_{e,0}$ and tells us directly where the potentially dangerous agents are, no matter whether they are occluded or not. This step is depicted in Figure 3.2c.

Stage 3 – Consistency with the Observation

BRS_o alone does not quantify the risk associated with a particle. We need to incorporate sensory systems, such cameras, radars and LiDARs, to identify risky control inputs. A combination of the aforementioned sensors generates a free space around the ego vehicle know as the *observable polygon*, $\mathcal{X}_s$, as shown in Figure 3.2d.

If the entire trajectory of $i-$th particle, $x_o^{[i]}(t)$, ends in $\mathcal{X}_s$, it is considered to be collision-free; otherwise, it is potentially risky. Quantitatively speaking, a weight $w_s^{[i]}$ is assigned to each particle i based on its entire trajectory.

$$w_s^{[i]} = \frac{1}{T^{[i]}} \int_0^{T^{[i]}} \mathbb{1}\left(x_o^{[i]}(t) \in \mathcal{X}_s\right) dt \tag{3.10}$$

where $\mathbb{1}(\cdot)$ is an indicator function. As a results, the weight $w_s^{[i]}$ renders the likelihood of the corresponding action $u_e^{[i]}(t)$ being collision-free or not.

3.3.3 Motion Planning

Based on the weight w_s from Section 3.3.2 it is possible to generate a set of safe actions by resampling. Resampling the particles with w_s defined in (3.10) filters out actions with potential collision. However, it is insufficient to use just w_s during resampling. For example, in many cases the safest action may be stopping completely before reaching the goal or getting up to the target speed. The ego vehicle can freeze and never reaches the goal. Hence we need another factor to promote forward motion of the ego vehicle.

Let u_d be the desired control input when no risk is presented in the scene. The desired control input can be from a predefined behavior, such as slowing down gently when approaching an intersection, or maintaining the target speed, etc. Assume that u_d is given, we assign another weight w_d to each particle.

$$w_d^{[i]} := \exp\left(-\frac{\delta_u^{[i]2}}{2\sigma_u^2}\right) \tag{3.11}$$

where

$$\delta_u^{[i]2} := \frac{1}{T^{[i]}} \int_0^{T^{[i]}} \left(u_d(t) - u_e^{[i]}(t)\right)^\top \left(u_d(t) - u_e^{[i]}(t)\right) dt$$

and σ_u is a hyperparameter for scaling. The weight w_d promotes those control inputs which are closer to the desired one, and thus is able to drive the ego vehicle forward.

A naive way to utilize both w_s for safety and w_d for driving is to interpret them as likelihoods and take their product as the final weight, and resample the particles. However, safe actions might be suppressed when they are no where near the desired action. So taking the direct product may end up with an action that is still too risky.

Instead, we calculate the final weight $w^{[i]}$ in the following way:

$$w^{[i]} = w_s^{[i]} \cdot \left(\varepsilon \cdot w_d^{[i]} + (1-\varepsilon) \cdot \left(1 - \min_j w_s^{[j]}\right)\right), \tag{3.12}$$

where ε is a small number. We set it to 10^{-4} in this work.

To understand (3.12) intuitively, let us first consider an extreme case. Assume $w_s^{[i]} = 1 \; \forall i$, indicating that all the trajectories $x_o^{[i]}(t)$ are in the observable polygon $\mathcal{X}_s$ ans thus being collision-free. The weight $w^{[i]}$ becomes proportional to $w_d^{[i]}$, so the actions closer to the desired action $u_d(t)$ will have higher final weights. In this case, resampling with $w^{[i]}$ makes the ego vehicle move according to the desired action.

On the other hand, if *any* of the trajectories shows slight risk (e.g. $1 - \min_j w_s^{[j]} > 10\varepsilon$), the final weight $w^{[i]}$ will be approximately proportional to $w_s^{[i]}$. As a result, safer actions get promoted and the value $w_d^{[i]}$ has negligible effect in this case. In short, using (3.12) with a reasonably small ε forces the ego vehicle to prioritize safety and to move forward only when it is risk-free. Resampling with the weight from (3.12) gives us a distribution represented by a set of ”good” particles. The final step is to select an optimal control input from this distribution for the ego vehicle to execute.

In general, the distribution can be multi-modal, meaning that there may be multiple control inputs that are equally good. For instance, if there are two vehicles approaching an intersection at the same time, two possibly equally good actions for one of the vehicles can be either 1) being aggressive – accelerating and pass first or 2) being conservative – waiting until the other one passes. For this reason, we make use of a well-developed clustering algorithm from the machine learning and computer vision community – Density-based spatial clustering of applications with noise (DBSCAN) [33] – to identify multi-modal distributions and select the optimal action.

DBSCAN clusters particles in the action space based on the local density without specifying the number of clusters a priori. It identifies the number of cluster automatically, and the centroids of clusters are candidates for the optimal control input. We choose to prioritize safety and make the ego vehicle to be more conservative. So if there are multiple clusters, we pick the one with the most deceleration. The optimal control input is then executed until the next iteration.

3.4 Evaluation

This section introduces the baselines for the proposed algorithm to be compared against. We also define the metrics for evaluation.

3.4.1 Model

Section 3.3.2 only gives a general description of the pipeline for clarity, but to test and implement the proposed algorithm, we need to pick appropriate models for both the ego vehicle and other vehicles. Due to its simplicity and tractable dimension, we choose to use the train-like model, meaning that all the agents travels in their own curvilinear coordinates. A concrete example of the model is to have a vehicle traveling on a lane without lane changing. Although we assume no lane changing in this work, it is possible to relax this constraint by using physics-based model like the bicycle model or other behaviour-based models.

By working in curvilinear coordinates, the location of the ego vehicle can be parameterized by only two numbers: longitudinal position s_e along the center line and the lateral offset b_e with respect to the center line. In addition, the heading is completely determined by s_e thus is not necessary to be included as a state. The dynamics of the ego vehicle can be written as

$$\begin{aligned}\dot{x}_e(t) &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_e(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_e(t) \\ x_e(t) &:= \begin{bmatrix} s_e(t) & \dot{s}_e(t) \end{bmatrix}^\top \\ u_e(t) &:= a_e \quad \forall t \geq 0,\end{aligned} \tag{3.13}$$

where a_e is a constant acceleration along the center line. The dynamics of other vehicles is written in a similar way, but assumed to have a constant speed v_o such that

$$\begin{aligned}\dot{x}_o(t) &= u_o(t) \\ x_o(t) &:= s_o(t) \\ u_o(t) &:= v_o \quad \forall t \geq 0\end{aligned} \tag{3.14}$$

If there are multiple valid lanes for non-ego vehicles, each lane should have its own dynamics.

Here we want to emphasize that more sophisticate models can be substituted in directly, but at the cost of higher computational complexity. We choose constant speed/acceleration models because their simplicity and reasonable performance in similar scenarios, as illustrated in our previous work [31]. Moreover, since we work in curvilinear coordinates, both models, (3.13) and (3.14), can be approximated by CTRV and CTRA models, respectively, when the replanning time T_r is small. Both CTRV and CTRA are shown to have comparable performance in prediction in urban environments [34].

The desired action $u_d(t)$ is designed to have a constant deceleration when approaching the stopline, and to switch to a saturating proportional controller to track a target speed.

3.4.2 Baseline 1

The first baseline is an occlusion-aware algorithm by Orzechowski, Meyer, and Lauer [7]. We recreate and simulate two of the scenarios demonstrated in their paper. The map is a four-way unsignalized intersection partially occluded by a non-transparent static object.

The first scenario consists of no other agent, and the second one has one other agent coming from the left hand side of the ego vehicle. In both scenarios, the goal is to navigate safely through the intersection and to reach a target speed of $9\ m/s$. The geometries of the map are measured using AutoCAD, and the velocity of the other agent is estimated afterwards.

3.4.3 Baseline 2

The second baseline is another occlusion-aware method from our previous work [31]. We simulate the same set of 1000 random scenarios with up to five other agents for each scene on both the proposed method and the second baseline. Each scene is an four-way unsignalized intersection and the geometries are extracted from OSM. There are 73 real-world intersections in total. In addition, there is also a synthetic layout where the roads are straight and perpendicular to each other.

These maps have significantly more severe occlusion comparing to those in the first baseline [7]. Buildings are added to the map with a two-meter buffer around the driving surface, as shown in Figure 3.1 and 2.2. Speed of a non-ego agent is randomly set to a constant between 4 to 12 m/s. For simplicity, we assume no collision between non-ego agents and they do not interact with other agents.

3.4.4 Metrics

For the first baseline, we investigate the speed and acceleration profiles for both ride comfort and efficiency. Ride comfort is evaluated by the number of abrupt breaking and the maximum deceleration, whereas efficiency is indicated by the terminal speed of the ego vehicle.

Collision rate is evaluated for the second baseline on one synthetic and all the 73 real-world intersections. This results in 74000 simulations in total for each method.

3.5 Results

3.5.1 Baseline 1

Figure 3.3 shows the behaviors under two scenarios recreated from the first baseline [31]. Both the first baseline and the proposed method enable the ego vehicle to reach to the goal without collision in either scenario. The ego vehicle slows down when approaching the intersection, no matter if there is any incoming traffic.

However, they have three key differences. To begin with, in the first scenario the baseline [7] has a sudden break at $t = 1.0s$, while the proposed method stays roughly at a constant deceleration. In addition, the maximum breaking (i.e. deceleration) in the second scenario is 27.5% lower in the proposed method. The reduced breaking and jerk indicate higher ride comfort. Finally, in the second scenario the ego vehicle creeps forward between $t = 2.7s$ and $t = 4.3s$ in order to get a clearer view, as shown in Figure 3.3h and 3.3j. Once it gets enough information and thinks it is risk-free, it then proceeds. This behavior leads to a higher efficiency, meaning that it obtains a 57.9% higher speed at the end of the simulation all without sacrificing safety.

	median collision rate	# of zero-collision intersections
Baseline 2 [31]	1.70%	0 out of 74
Proposed	0.20%	10 out of 74

Table 3.1: Simulation results at 74 **intersections.**

3.5.2 Baseline 2

Among the 74 intersections, the proposed method obtains a median collision rate of 0.20%, which is a significant reduction compared to the 1.70% of the second baseline [31], as shown in Table 3.1. Moreover, the proposed method achieves zero collision at not only the synthetic layout, but also 9 real-world layouts. In contrast, the baseline [31] has a minimum collision rate of 0.70% at one of the 74 intersections.

3.6 Conclusion

We propose a probabilistic risk assessment method for motion planning under occlusion in urban intersections. The proposed algorithm is able to quantify the distribution of risk in the action space, and can be used to generate low-level control inputs for an autonomous vehicle to navigate safely. It also identifies spatially where the risk-inducing regions are via both forward and backward reachability. The proposed method is compared quantitatively to previous work and shows significant improvement in terms of both efficiency and safety.

Future work include using the proposed method to redirect and focus sensory resources to further reduce impact on limited sensing, formulating and solving the problem analytically without sampling to obtain safety guarantees, and deploying it to hardware platforms for testing in the real-world.

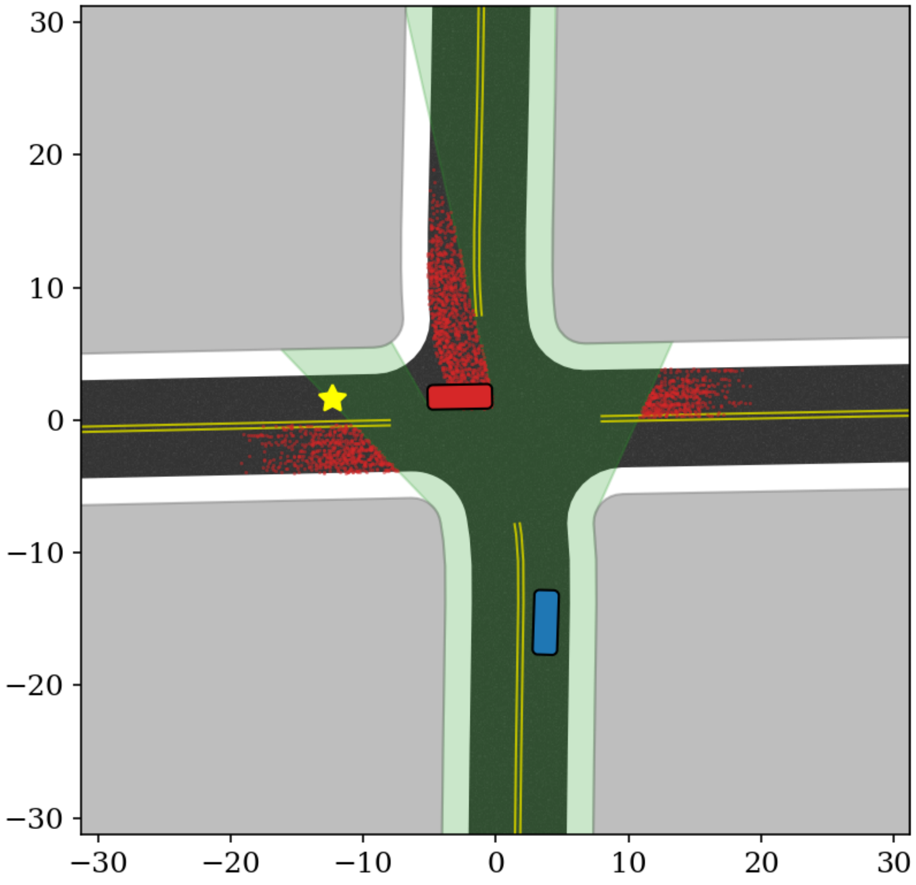


Figure 3.1: The ego vehicle (blue box) intends to perform an unprotected left turn to the goal (yellow star) at an intersection. The green shaded region is the observable polygon and arises due to the 1) limited sensor range and 2) occlusions from another vehicle (red box) and buildings (gray regions.) The proposed algorithm quantifies and identifies the location of risk posed by other vehicles (red particles). Our approach identifies and focuses its attention on risk-inducing regions in the near future (e.g. 1.5 seconds) to ensure that it does not waste computational resources predicting everything in the occluded region. Both axes are in meters.

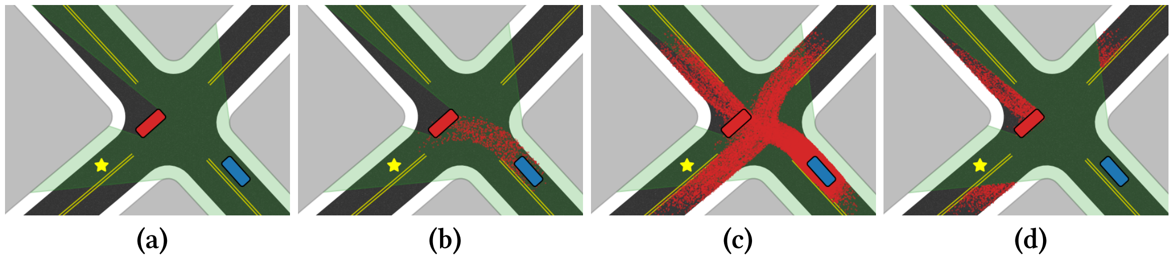
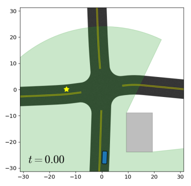
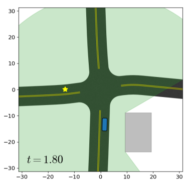


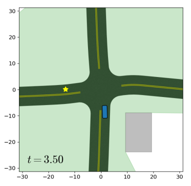
Figure 3.2: Step-by-step illustration of different stages of the proposed algorithm. (a) An intersection with occlusion and with one other agent (red box) in the scene. The observable polygon is indicated by the green shaded region, and the goal location is indicated by the yellow star. (b) State 1: calculating FRS$_e$. The dots are projections of the particles $\left\{x_e\left(T^{[i]}\right)\right\}_{i=1}^{N}$ in 2D. (c) Stage 2: calculating BRS$_o$. The dots are projections of the particles $\left\{x_o^{[i]}(0)\right\}_{i=1}^{N}$ in 2D. (d) Stage 3: filtering with the observable polygon. The remaining red dots in the figure indicates spatially the source of risk-inducting regions.



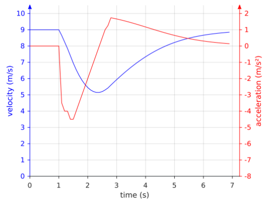
(a)



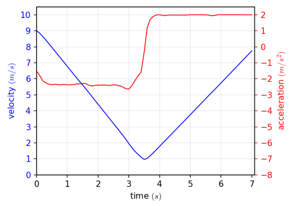
(b)



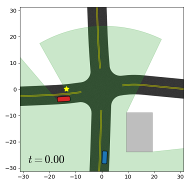
(c)



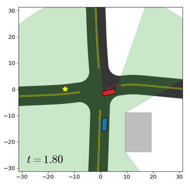
(d) Baseline 1 [7]



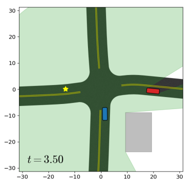
(e) Proposed



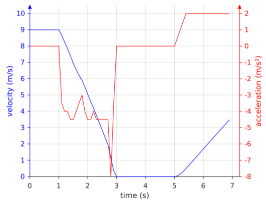
(f)



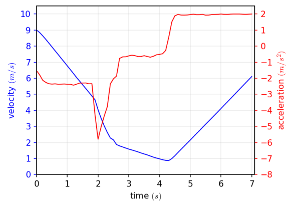
(g)



(h)



(i) Baseline 1 [7]



(j) Proposed

Figure 3.3: Two scenarios in the first baseline [7]. The first scenario (top row) is in a map with one static obstacle (gray) and no incoming traffic, and the second scenario (bottom row) is in the same map with one other vehicle (red) coming from the left. The first three columns (a-c, f-h) show key frames of the proposed method, and the other two columns (d,e,i,j) show the speed and acceleration profiles. In the first scenario, the baseline abruptly breaks at $t = 1.0s$ whereas the proposed method maintains a near constant deceleration, as shown in (d) and (e). In the second scenario, the baseline breaks at both $t = 1.0s$ and $t = 2.8s$, whereas the proposed method only breaks once at $t = 1.9s$, as shown in (i) and (j). In addition our method does not stop completely in the second scenario. Instead, it creeps forward between $t = 2.7s$ and $t = 4.3s$ to gather more information. This behavior results in a higher final speed and thus makes the ego vehicle faster to reach the goal. The fourth column is adopted directly from Figure 4 in [7].

Chapter 4

DeSnowNet: Real-time Snow Removal for LiDAR Point Clouds

4.1 Introduction

LiDARs are classified as active sensors, emitting pulsed light waves and utilizing the returning pulse to calculate the distances between it and the surrounding objects. This makes LiDARs suitable during both daytime and nighttime, constantly measuring at centimeter-level accuracy of the environment. Such measurements can be easily found in modern datasets [35]–[38] as frame-by-frame point clouds, commonly sampled at 10Hz, and has been used for 3D object detection, semantic segmentation, and SLAM [39].

Though LiDARs provide more actuate 3D measurements compared to radar, they are prone to adverse weather conditions. Unlike radars, which see through smog, fog, snowflakes and rain droplets, LiDARs are greatly affected by different sizes of particles in the air. Specifically during snowfall, the pulsed light waves hit the snowflakes and return to the sensor as *ghost* measurements, as shown in Fig. 4.1a and 4.1b. It is crucial to remove such measurements and unveil the underlying geometry of the scene for applications like SLAM.

Recently, there is an increasing number of large-scale datasets containing adverse weather conditions. The most noticeable ones are the CADC Dataset [13] and the WADS [40]. De-noising algorithms [41], [42] have also been developed alongside with them. To the best of our knowledge, all of the unsupervised de-noising algorithms are based on nearest neighbor search and have difficulty operating in real-time even with a moderate $64-$beam LiDAR (about $100k$ points/frame). Therefore, it is critical to introduce an alternative formulation which do not rely on nearest neighbor search.

An alternative way to represent point clouds is using *range images* [43], as shown in Fig. 4.2a and 4.2b. Each range image has two channels, and the pixel values are 1) the Euclidean distances in the body coordinate and 2) the intensities. With this representation, we can leverage common algorithms in image processing, including the Fast Fourier Transform (FFT), the Discrete Wavelet Transform (DWT), and the Difference of Gaussians (DoG) [44]. CNNs with pooling, batch normalization [45] and dropout [46] can also be applied without any modification. More importantly, these operations can be performed very efficiently on modern hardware such as GPUs.

However, developing a CNN for de-noising point clouds poses its own challenges. The most obvious one is that obtaining point-wise labels to isolate snowflakes from the scene is usually prohibitively expensive on large-scale datasets. Thus the formulation needs to be either unsupervised or self-supervised to exploit unlabeled data. With this in mind, we propose a novel network, *DeSnowNet*, which can be trained with unlabeled point clouds.

Assuming the noise-free range images are sparse in some basis, we design loss functions that relies purely on quantifying the sparsity of the data. The L_1 norm is used as a proxy to sparsity, and encourages sparse coefficients during training. In addition, the DWT and the Inverse Discrete Wavelet Transform (IDWT) are used for down sampling and up scaling within the network instead of pooling and convolution transpose, respectively. The proposed network is evaluated on a subset of WADS [40], showing similar or superior performance in de-noising compared to other state-of-the-art methods while running 47 times faster.

This chapter is organized as follows: Section 4.2 lists related work in datasets with adverse weather conditions, existing de-noising methods, and backgrounds in compressed sensing with sparse representations. Section 4.3 goes through the formulation of the proposed network and loss functions in detail, including how we train the network in a unsupervised fashion. Section 4.4 introduces the metrics for comparing the performance of the proposed network and other methods. Section 4.5 quantitatively shows the results of the aforementioned metrics, as well as qualitatively presents the resulting map from an off-the-shelf SLAM algorithm. Lastly, Section 4.6 summarizes the contributions and advantages of the proposed method.

4.2 Related Work

4.2.1 Datasets

Though large-scale datasets for autonomous driving [35]–[38] become more ubiquitous nowadays, most of them contain very little or no LiDAR data collected during adverse weather conditions. However, there are some exceptions.

Heinzler, Piewak, Schindler, *et al.* [43] collected a dataset with points clouds in a weather-controlled climate chamber, simulating rain and fog in front of a vehicle. The dataset provides point-wise labels with three classes: clear, rain, and fog, where the "clear" points are the ones which are not caused by the adverse weather.

Pitropov, Garcia, Rebello, *et al.* [13] published the CADC dataset which contains 32754 point clouds under various snow precipitation levels. The point clouds were collected from a top-mounted HDL-32e and were grouped into sequences where each sequence has about 100 consecutive point clouds. Though a subset of the CADC dataset has 3D bounding boxes for vehicles and pedestrians, we do not use them in our work as the proposed method is designed to be unsupervised.

Recently Bos, Chopp, Kurup, *et al.* [40] released the WADS with only 1828 point clouds which is significantly smaller than CADC. But unlike CADC, each point cloud in WADS has point-wise labels. There are 22 classes, including "active falling snow" and "accumulated snow".

Since the target application of the proposed network is snow removal, only CADC and WADS are used for training and evaluation.

4.2.2 De-noising Point Clouds Corrupted by Snow

Similar to the proposed method, the WeatherNet [43] projects point clouds onto a spherical coordinate as range images, which we go into detail in Section 4.3.1. With this representation and the dataset released alongside with the WeatherNet, they formulated the de-noising problem as supervised multi-class pixel-wise classification. The network is constructed by a series of LiLaBlock [47] and is trained with point-wise class labels. The WeatherNet is able to remove rain and fog, but it is unclear how it performs on snow. Additionally, training the WeatherNet requires point-wise label. This prevents the WeatherNet from taking advantage of large-scale unlabeled point cloud dataset such as CADC.

The Dynamic Radius Outlier Removal (DROR) [41] filter removes the noise by first constructing a $k-d$ tree, and count the number of neighbors of each point using dynamic search radii. The idea is that the density of the point clouds from the scene is inversely proportional to the distance. The search radius of each point is dynamically adjusted by accordingly. If a point does not have enough number of neighbors within it search radius, it is classified as an outlier (i.e. snow).

The Dynamic Statistical Outlier Removal (DSOR)[42] filter also rely on nearest neighbor search, but it calculates the mean and variance of relative distances given a fixed number of neighbors of each point. In other words, it estimates the local density around each point, and outliers can then be filtered out. DSOR is shown to be slightly faster than DROR while having a similar performance in de-noising.

Both DROR and DSOR require nearest neighbor search which prevents them from being real-time. A moderate LiDAR like the HDL-64E generates around $10k$ points at 10Hz. Querying this much points with a $k-d$ tree of this size takes hundreds of milliseconds on modern hardware. On the other hand, WeatherNet [43] and our method work with image-like data, which will be discussed later in Section 4.3.1. Common tools in image processing, such as FFT, DWT, and CNNs, and be applied directly and runs very efficiently on GPUs. This allows us to develop a real-time de-noising algorithm that runs orders of magnitude faster than DROR and DSOR.

4.2.3 Sparse Representations

Both DWT and FFT are know to generate sparse representations on real-world multi-dimensional grid data like audio and images, meaning that the Fourier and Wavelet coefficients of natural signals are mostly zeros. Typically, a signal corrupted by additive noise becomes less sparse the the underlying clean one, as shown in Fig. 4.2c and 4.2d. In other words, it is possible to perform de-noising if the underlying clean signal is sufficiently sparse in some basis. Theoretically this can be done by minimizing the L_1 norm of the coefficients.

Assuming that a noisy signal v has a similar sparsity under m different transformations $\mathcal{T}_1, \dots, \mathcal{T}_m$, we aim to solve

$$\min_u \lambda \|u - v\|_q + \frac{1}{m} \sum_{\mathcal{T}_j} \|\mathcal{T}_j(u)\|_1, \tag{4.1}$$

where u is the underlying clean signal, $\|\cdot\|_q$ is the L_q norm, and $\lambda \in \mathbb{R}$ is the weight keeping u close to v. The value q is commonly set to be either 1 or 2, depending on the structure of the noise.

However, solving Eq. 4.1 directly is computationally intensive for any reasonably sized signal v. Thus we propose to use a CNN to approximate the solution and use Eq. 4.1 as the groundwork of our loss functions. More details can be found in Section 4.3.2 and 4.3.3.

4.3 Method

4.3.1 Preprocessing

The first step is to convert point clouds into range images. Given a point $p = \begin{bmatrix} x & y & z \end{bmatrix}^\top \in \mathbb{R}^3$ in the $k-$th point cloud and its corresponding intensity value $i \in [0, 1]$, we calculate the following values:

$$\begin{aligned} d &:= \|p\|_2 \in \mathbb{R}_+, \\ \phi &:= \sin^{-1}\left(\frac{z}{d}\right) \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right], \\ \psi &:= \tan^{-1}\left(\frac{y}{x}\right) \in [-\pi, \pi), \end{aligned} \tag{4.2}$$

where d is the distance from the LiDAR center, ϕ is the inclination angle, and ψ is the azimuth angle. By discretizing the inclination and azimuth angles within the field-of-view (FOV) of the LiDAR, we project each point within a point cloud onto a spherical coordinate, resulting a range image $I_k \in \mathbb{R}_+^{h \times w \times 2}$, where $k \in \mathbb{Z}$ is the frame index, h is the vertical resolution, and w is the horizontal resolution. Throughout this chapter h is set to be the number of beams of the LiDAR and w is set to be a constant 2048. The first channel of the range image I_k is the distance d of each point, and the second channel is the corresponding intensity value i.

The second step is to squash the range images to a proper scale. A key observation is that the ghost measurements mainly concentrate within a distance of 25 meters, as shown in Fig. 4.1, whereas the maximum distance of a LiDAR easily exceeds 150 meters. We scale the range images I_k with an element-wise function $g_1 : \mathbb{R}_+^{h \times w \times 2} \mapsto \mathbb{R}_+^{h \times w \times 2}$, defined as

$$g_1(I_k) := \sqrt[3]{I_k}, \tag{4.3}$$

to amplify the relative importance of points near the ego vehicle while preserving its ordering in distances. The function g_1 also enhances the contrast between the noise and the scene in the intensity channel, since the intensity values of the snowflakes are almost always 0 and the scene usually has positive intensity values.

Finally, not every pixel has a value due to the lack of points at certain directions such as the sky and some transparent surfaces. The void pixels is processed with a series of operations, collectively denoted as $g_2 : \mathbb{R}_+^{h\times w\times 2} \mapsto \mathbb{R}_+^{h\times w\times 2}$, before entering the network. In sequence, the operations are:

1. 3×3 dilation to fill isolated void pixels.
2. Filling with the value $(\mu_k + \sigma_k)$, where $\mu_k \in \mathbb{R}_+^{h\times 2}$ and $\sigma_k \in \mathbb{R}_+^{h\times 2}$ are the row-wise mean and standard deviation, respectively.
3. Subtraction with a 3×3 DoG to reduce the amplitude of isolated noisy pixels.
4. 7×7 average pooling to smooth out the image.

Note that g_2 does not modify any pixel with a valid measurement. Only the void pixels are altered. An example of the a resulting image

$$\tilde{I}_k := (g_2 \circ g_1)(I_k) \tag{4.4}$$

is shown in Fig. 4.2a and 4.2b.

4.3.2 Network Architecture

The proposed network, DeSnowNet $f_\theta : \mathbb{R}_+^{h\times w\times 2} \mapsto \mathbb{R}^{h\times w\times 2}$, is based on the MWCNN [48] with several key modifications. First, all convolution layers are replaced by residual blocks [49] with two circular convolution layers. In addition, a dropout layer is placed after the first ReLU activation in each residual block to further regularize the network. Lastly, the number of channels is dramatically reduced – the first level only has 8 channels instead of 160 channels in MWCNN. The number of channels of the higher levels are also reduced accordingly. These modifications are critical for processing sparse representations (i.e. FFT and DWT coefficients) of panoramic range images.

The proposed network is designed to produce the *residual image* $\Delta_k \in \mathbb{R}^{h\times w\times 2}$, satisfying

$$\begin{aligned} \Delta_k &:= f_\theta\left(\tilde{I}_k\right), \\ \hat{I}_k &:= \tilde{I}_k - \Delta_k, \end{aligned} \tag{4.5}$$

where $\hat{I}_k$ is the de-noised range image representing the geometry of the underlying scene.

4.3.3 Loss Functions

Let $\mathcal{F} : \mathbb{R}^{h\times w\times 2} \mapsto \mathbb{R}^{h\times w\times 2}$ be the real-valued FFT and $\mathcal{W} : \mathbb{R}^{h\times w\times 2} \mapsto \mathbb{R}^{h\times w\times 2}$ be the DWT with the Haar basis. With the formulation in Eq. 4.1 and 4.5, we designed three novel loss functions, two of which quantify the sparsity of the range image, while the third one quantifies the sparsity of the noise.

$$\begin{aligned} \mathcal{L}_{\mathcal{F}} &:= \frac{1}{N}\sum_{k=1}^{N}\left\|\log\left(\left|\mathcal{F}\left(\hat{I}_k\right)\right| + 1\right)\right\|_1, \\ \mathcal{L}_{\mathcal{W}} &:= \frac{1}{N}\sum_{k=1}^{N}\left\|\mathcal{W}\left(\hat{I}_k\right)\right\|_1, \\ \mathcal{L}_{\Delta} &:= \frac{1}{N}\sum_{k=1}^{N}\left\|\Delta_k\right\|_1, \end{aligned} \tag{4.6}$$

where N is the number of point clouds in the training set, and $|\cdot|$ is the element-wise absolute value.

Note that $\mathcal{L}_{\mathcal{F}}$ uses the log-magnitude of the Fourier coefficients. Since $\log(1+\epsilon) \approx \epsilon$ for some small ϵ, it prevents the network from focusing too much on the low-frequency part of the spectrum while pertaining the properties of the L_1 norm. The total loss function $\mathcal{L}$ can thus be constructed as

$$\mathcal{L} := \alpha \cdot \frac{\mathcal{L}_{\mathcal{F}} + \mathcal{L}_{\mathcal{W}}}{2} + (1-\alpha)\cdot\mathcal{L}_{\Delta}, \tag{4.7}$$

where $\alpha \in (0,1)$ is a hyperparameter balancing the sparsity of the range image and the sparsity of the residual.

4.3.4 Prediction

Let $\Delta_k^d \in \mathbb{R}^{h\times w}$ be the distance channel and $\Delta_k^i \in \mathbb{R}^{h\times w}$ be the intensity channel of the residual image Δ_k. We define the decision boundary primarily in the residual space. A point p with residuals $\delta_k^d \in \Delta_k^d$ and $\delta_k^i \in \Delta_k^i$ is classified as snow (i.e. positive) if it satisfies all the following conditions:

- Being the foreground (i.e. p is closer than nearby pixels).

$$\delta_k^d > 0$$

- Absorbing most of the energy of LiDAR beams (i.e. p is darker than nearby pixels).

$$\delta_k^i > 0$$

- The residuals of p are not sparse.

$$\left(\delta_k^d\right)^{n_d} \cdot \left(\delta_k^i\right)^{n_i} > \bar{\delta}$$

where $n_d \in \mathbb{R}_+$ and $n_i \in \mathbb{R}_+$ are the shape parameters of the primary decision boundary, and $\bar{\delta} \in \mathbb{R}_+$ is a threshold value.

4.4 Evaluation

Quantitatively, the proposed network and other two state-of-the-art methods are evaluated with three common metrics for binary classification on a subset of WADS [40], since it is the only publicly available dataset containing point-wise labels with snow at the moment of writing this dissertation.

- Precision: $\frac{TP}{TP+FP}$
- Recall: $\frac{TP}{TP+FN}$
- Intersection-over-Union (IoU): $\frac{TP}{TP+FP+FN}$

	Precision	Recall	IoU	Runtime [ms]
DROR [41]	0.8565	0.9197	0.7957	1229.5
DSOR [42]	0.6370	**0.9382**	0.6081	319.6
DeSnowNet (ours)	**0.9295**	0.9008	**0.8421**	**6.8**

Table 4.1: De-noising results.

where TP is the number of true positive points, FP is the number of false positive points, and FN is the number of false negative points. We also record the average runtime on a modern desktop computer with a Ryzen 2700X and an RTX 3060.

In addition, qualitative results in SLAM with a sequence of de-noised point clouds are presented.

4.5 Results

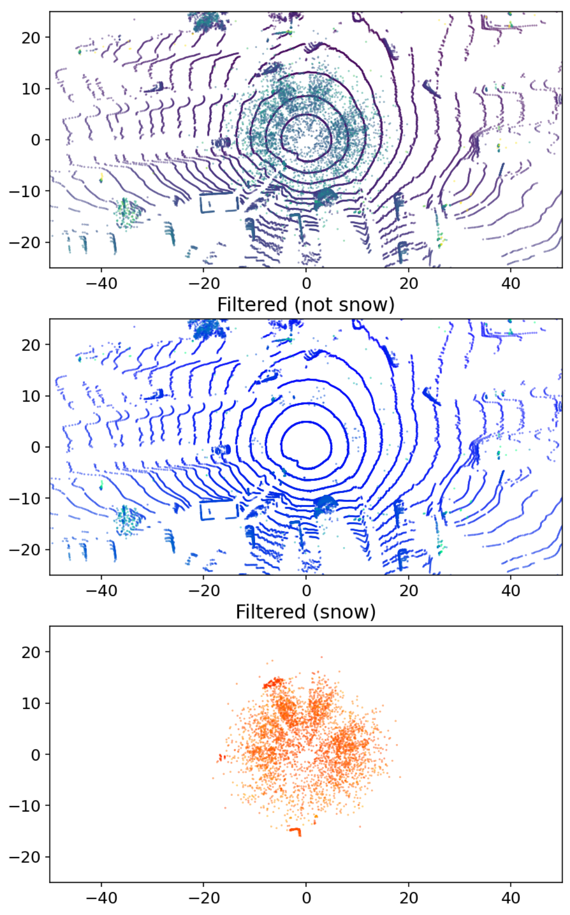
Compared to the two baselines, the proposed network yields similar or superior performance in de-noising, as shown in TABLE 4.1. The recall is 3.74% lower than DSOR, while the precision and IoU are noticeably higher than both DROR and DSOR.

More importantly, the proposed network is $47\times$ faster than DSOR and $181\times$ faster than DROR with about $10k$ points per frame. Considering that the sampling rate of LiDARs are usually at 10Hz, the proposed method is the only one that is truly capable of running at real-time.

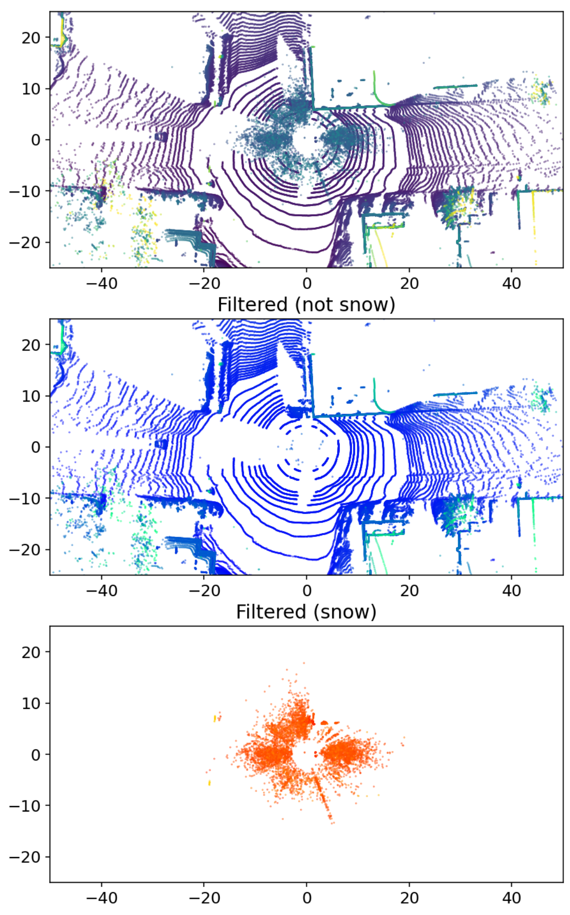
The de-noised point clouds also enable building clean maps using an off-the-shelf SLAM algorithm like the LeGO-LOAM, as shown in Fig. 4.3. The map with raw point clouds shows a large number of ghost points above where the ego vehicle traveled. By contrast, resulting map with de-noise point clouds is much cleaner, containing a minimal amount of noise.

4.6 Conclusions

We propose a deep CNN, DeSnowNet, for de-noising point clouds corrupted by snowfall. The proposed network can be trained without any labeled data, and generalizes well to multiple datasets. The network processes $100k$ points under 10ms while yielding superior performance compared to the state-of-the-art method.

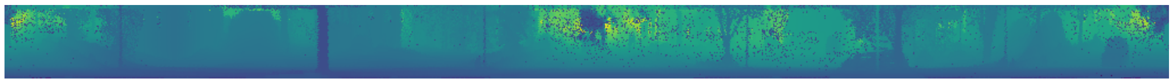


(a) CADC [13]

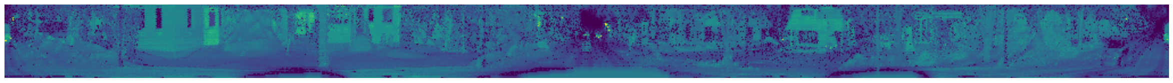


(b) WADS [40]

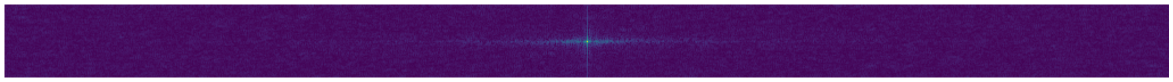
Figure 4.1: **Point clouds corrupted by snowflakes from two datasets, and the de-noised results using the proposed method. Top to bottom: Raw point clouds, de-noised point clouds, and the ghost measurements. Both axes are in meters.**



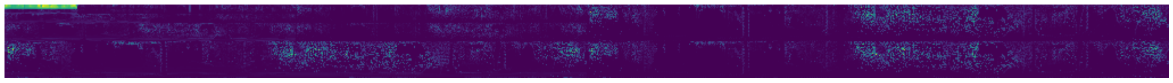
(a) The distance channel.



(b) The intensity channel.

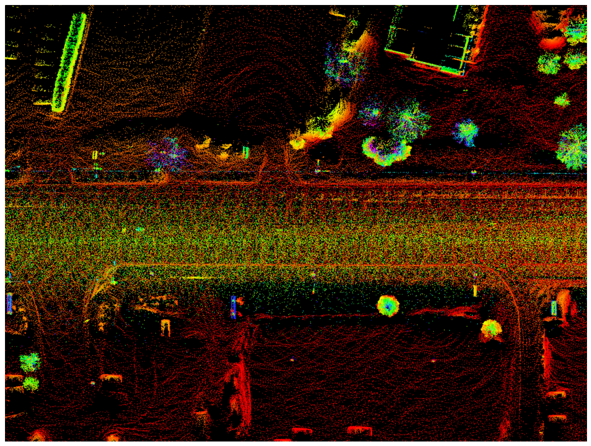


(c) The log-magnitude of the FFT coefficients of the distance channel.

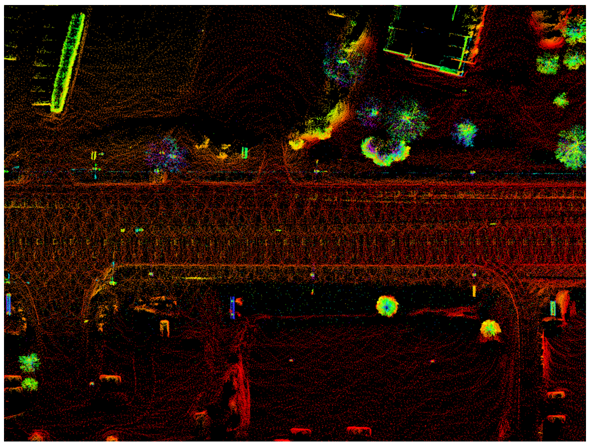


(d) The magnitude of the DWT coefficients of the distance channel.

Figure 4.2: (a), (b): a range image $\tilde{I}_k$ from WADS [40]. (c), (d): the corresponding sparse coefficients of the distance channel.



(a) Raw point clouds



(b) De-noised point clouds

Figure 4.3: Maps built from (a) raw point clouds and (b) de-noised point clouds from a sequence in CADC [13]. Both maps use LeGO-LOAM [39]. The colors indicate the height (z) of the points.

Bibliography

[1] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 3354–3361.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “Nuscenes: A multimodal dataset for autonomous driving,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[4] H. Lee, H. Chae, and K. Yi, “A geometric model based 2d lidar/radar sensor fusion for tracking surrounding vehicles,” *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 130–135, 2019.

[5] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, “Towards a viable autonomous driving research platform,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2013, pp. 763–770.

[6] S. Manzinger and M. Althoff, “Tactical decision making for cooperative vehicles using reachable sets,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 444–451.

[7] P. F. Orzechowski, A. Meyer, and M. Lauer, “Tackling occlusions & limited sensor range with set-based safety verification,” *arXiv preprint arXiv:1807.01262*, 2018.

[8] M. Koschi and M. Althoff, “Spot: A tool for set-based prediction of traffic participants,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1686–1693.

[9] H. Ahn, K. Berntorp, and S. Di Cairano, “Reachability-based decision making for city driving,” in *2018 Annual American Control Conference (ACC)*, 2018, pp. 3203–3208.

[10] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, “An efficient reachability-based framework for provably safe autonomous navigation in unknown environments,” *arXiv preprint arXiv:1905.00532*, 2019.

[11] S. Vaskov, U. Sharma, S. Kousik, M. Johnson-Roberson, and R. Vasudevan, “Guaranteed safe reachability-based trajectory design for a high-fidelity model of an autonomous passenger vehicle,” in *2019 American Control Conference (ACC)*, 2019, pp. 705–710.

[12] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. R. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, “Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments,” *CoRR*, vol. abs/1902.02851, 2019.

[13] M. Pitropov, D. E. Garcia, J. Rebello, M. Smart, C. Wang, K. Czarnecki, and S. Waslander, “Canadian adverse driving conditions dataset,” *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 681–690, 2021.

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

[15] P. Sudhakar, K. A. Sheela, and M. Satyanarayana, “Imaging lidar system for night vision and surveillance applications,” in *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan. 2017, pp. 1–6.

[16] 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, Apr. 2017.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, “SSD: single shot multibox detector,” *CoRR*, vol. abs/1512.02325, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034.

[19] M. Lee, K. Jo, and M. Sunwoo, “Collision risk assessment for possible collision vehicle in occluded area based on precise map,” in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 1–6.

[20] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *ROBOMECH Journal*, vol. 1, no. 1, p. 1, Jul. 2014.

[21] K. Wyffels and M. Campbell, “Negative information for occlusion reasoning in dynamic extended multiobject tracking,” *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 425–442, Apr. 2015.

[22] J. Yu and S. M. LaValle, “Shadow information spaces: Combinatorial filters for tracking targets,” *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 440–456, Apr. 2012.

[23] E. Galceran, E. Olson, and R. M. Eustice, “Augmented vehicle tracking under occlusions for decision-making in autonomous driving,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3559–3565.

[24] S. Brechtel, T. Gindele, and R. Dillmann, “Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 392–399.

[25] S. Brechtel, T. Gindele, and R. Dillmann, “Solving continuous pomdps: Value iteration with incremental learning of an efficient space representation,” in *International Conference on Machine Learning*, 2013.

[26] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, “Scalable decision making with sensor occlusions for autonomous driving,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[27] S. Russell and A. L. Zimdars, “Q-decomposition for reinforcement learning agents,” in *Proceedings of the 20th International Conference on International Conference on Machine Learning*, ser. ICML’03, Washington, DC, USA: AAAI Press, 2003, pp. 656–663.

[28] L. L. Hoberock, “A survey of longitudinal acceleration comfort studies in ground transportation vehicles,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, pp. 76–84, 1977.

[29] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, “An open approach to autonomous vehicles,” *IEEE Micro*, vol. 35, no. 6, pp. 60–68, 2015.

[30] S. Ammoun and F. Nashashibi, “Real time trajectory prediction for collision risk estimation between vehicles,” in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, 2009, pp. 417–422.

[31] M. Yu, R. Vasudevan, and M. Johnson-Roberson, “Occlusion-aware risk assessment for autonomous driving in urban environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2235–2241, 2019.

[32] H. O. Jacobs, O. K. Hughes, M. Johnson-Roberson, and R. Vasudevan, “Real-time certified probabilistic pedestrian forecasting,” *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2064–2071, 2017.

[33] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” AAAI Press, 1996, pp. 226–231.

[34] R. Schubert, E. Richter, and G. Wanielik, “Comparison and evaluation of advanced motion models for vehicle tracking,” in *2008 11th International Conference on Information Fusion*, 2008, pp. 1–6.

[35] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[36] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," *CoRR*, vol. abs/2006.14480, 2020.

[38] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom, and A. Valada, "Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking," *arXiv preprint arXiv:2109.03805*, 2021.

[39] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4758–4765.

[40] J. P. Bos, D. Chopp, A. Kurup, and N. Spike, "Autonomy at the end of the Earth: an inclement weather autonomous driving data set," in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2020*, International Society for Optics and Photonics, vol. 11415, SPIE, 2020, pp. 36–48.

[41] N. Charron, S. Phillips, and S. L. Waslander, "De-noising of lidar point clouds corrupted by snowfall," in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 254–261.

[42] A. Kurup and J. Bos, *Dsor: A scalable statistical filter for removing falling snow from lidar point clouds in severe winter weather*, 2021.

[43] R. Heinzler, F. Piewak, P. Schindler, and W. Stork, "Cnn-based lidar point cloud de-noising in adverse weather," *IEEE Robotics and Automation Letters*, 2020.

[44] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[45] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[46] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 1050–1059.

[47] F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, D. Pfeiffer, M. Enzweiler, and M. Zöllner, *Boosting lidar-based semantic labeling by cross-modal training data generation*, 2018.

[48] P. Liu, H. Zhang, W. Lian, and W. Zuo, "Multi-level wavelet convolutional neural networks," *CoRR*, vol. abs/1907.03128, 2019.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.