2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Safe Path Planning with Gaussian Process Regulated Risk Map

Hongliang Guo[1†], Zehui Meng[1†], Zefan Huang[3], Leong Wei Kang[1], Ziyue Chen[3], Malika Meghjani[1], Marcelo Ang Jr[3] and Daniela Rus[2]

***Abstract*— Government data identifies driver behaviour errors as a factor in 94% of car crashes, and autonomous vehicles (AVs), which avoids risky driver behaviours completely, are expected to reduce the number of road crashes significantly. Thus, one of the central focuses of developing AVs is to ensure safety during navigation. However, in reality, AV safety has been far below its expectation, and so far, no government has allowed for complete autonomous driving without human supervision. This paper proposes a dynamic safe path planning algorithm for AVs with Gaussian process regulated risk map. By reasonably assuming that the output of the object detection and tracking module follows a multi-variate Gaussian distribution, we put forward a safe path planning paradigm with Gaussian process regulated risk map, ensuring safety with high confidence. Both simulation results and in-vehicle tests demonstrate the effectiveness of the proposed algorithm.**

I. INTRODUCTION

Path planning has been one of the prevailing research topics in autonomous vehicles along with environmental perception, localization and mapping [1]. To implement path planning for autonomous vehicles and/or robots, one typically requires (a). a route planner which generates a sequence of configurations (position and orientation) taking the vehicle from the start to the goal, and (b). a motion planner, which transforms the configuration sequence into actuator commands conforming to the constraints of vehicle dynamics [2]. This paper focuses on the former with an emphasize on the safety requirement in the configuration space under perception uncertainties.

Statitistics show that driver behaviour errors lead to as many as 94% of the total car accidents [3], while autonomous vehicles, which excels in manoeuvre accuracy, are expected to increase driving safety. However, in order to achieve absolute safe manoeuvres in AVs, one requires perfect environmental models, which are too challenging, if not impossible, given current sensing and communication technologies and noisy real world scenarios. Hence, one needs a planning algorithm which works safely with perception uncertainties. This paper assumes that the perception uncertainty follows a Gaussian process, i.e. the perceived obstacle location and velocity from the perception module is a sample drawn from a multi-variate Gaussian distribution around its true values, and proposes a **S**afe **P**ath **P**lanning **A**nd **M**apping (SPPAM) algorithm, which ensures the path safety with high confidence while iteratively reducing perception uncertainties.

This paper formulates AV's real time path planning and decision making process into a canonical optimization framework with the objective of reaching the goal with minimal 'risk-projected' cost, while conforming to the safety constraint with high confidence. The contribution of the paper can be summarized as follows: (1) an on-line Gaussian risk map update process is proposed taking the real time obstacle detection and tracking data stream as input; (2) SPPAM outputs the safest path based on current risk map and decreases the environmental uncertainty within the risk map with subsequent object detection and tracking; (3) SPPAM is able to perform proactive path planning in dynamic environment with forecasted risk map; and (4) SPPAM is integrated together with canonical 3D object detection and tracking framework and verified in a real AV testbed.

The remainder of the paper is organized as follows. Section II surveys the related works on AV path planning, safe path planning, and emerging safety planning/exploration techniques in GP-regulated environments, followed by the formal introduction and analysis of SPPAM in Section III. Simulation results and analysis are provided in Section IV, followed by the in-field test on an AV platform in Section V. The paper ends with concluding remarks and future directions in Section VI.

II. RELATED WORKS

In this section, we first give a brief survey on path planning for autonomous vehicles, and then focus on safe path planning algorithms under uncertainties. As safety issue has been gaining increasing popularity in a broader domain than path planning, we also review some of the recent technologies on safe decision making in GP-regulated environment as the underlying rationale is closely related to our SPPAM algorithm.

A. Path Planning for Autonomous Vehicles

Path planning for AVs is a difficult decision making problem in that a vehicle is a non-linear, non-holonomic system and such system is required to maintain the desired performance (e.g., running at a desired speed) while avoiding collision with surrounding vehicles and infrastructure [4].

Canonical path planning methods can be roughly divided into three categories, namely, graph-based planners, sampling-based planners and trajectory optimization methods [5]. (1) ***Graph-based planners*** represent the environment

[1]Hongliang Guo, Zehui Meng, Leong Wei Kang and Malika Meghjani are with Singapore MIT Alliance for Research and Technology, Singapore hongliang@smart.mit.edu

[2]Daniela Rus is with Computer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA rus@mit.edu

[3]Marcelo Ang Jr, Zefan Huang and Ziyue Chen are with National University of Singapore, Singapore mpeangh@nus.edu.sg

[†]These authors have equal contribution

in a grid-based manner, and many (well known) algorithms exist for efficient path planning such as Dijkstra's algorithm, A* algorithm and its variants e.g., Anytime A* [6] and D* lite [7]. (2) ***Sampling-based planners*** such as Rapidly-exploring Random Trees (RRT) [8] and Probabilistic Road Maps (PRM) [9] generate feasible trajectories conforming to vehicle dynamics, and an evaluation function is performed over those trajectories and may even guide the sampling strategy. The best trajectory out of the sampled ones is selected for execution. (3) ***Trajectory optimization methods*** formulate path planning as an optimization problem taking account of the desired performance and relevant constraints [10]. For example, Liu et al. use model predictive control to solve a sequence of finite time trajectory optimization problem in a recursive manner and take account of the update of environmental states during its planning process [4].

B. Safe Path Planning Under Uncertainties

Almost all the path planning methods, which directly concentrate on safety under environmental uncertainties, follow the sampling-based planner framework, see [11]. The methods typically propose sampled trajectories to a certain safety-critical performance evaluator, and the returned metric will guide the next sampling strategy until the proposed trajectory attains optimum or reaches a certain termination condition, say time limits. Bouraine et al. proposes a p-safe RRT for safe path planning in unknown dynamic environment [12], and the algorithm iteratively proposes sampled RRT trajectories until the proposed one is p-safe per definition.

Another research direction in safe path planning is to formulate the problem into a risk-aversive way, in that risk is deemed as the opposite of safety. The environmental uncertainties are transformed into a risk value either deterministically [13] or stochastically [14]. This paper focuses on the stochastic nature of the environment, hence, we only survey the literature which treats risk as a random variable (RV). The corresponding research field can be referred as stochastic risk-aversive path planning (SRAPP). In SRAPP, the problem is usually formulated into an optimization framework, in which the term risk[1] can be defined as value at risk (VaR), conditional value at risk (CVaR) or mean-variance minimization [15]. Corresponding algorithms are developed to solve the optimization problem either globally or locally.

The risk map in risk-aversive path planning is usually established through some heuristic predictors, such as in [16] or function approximators which maps locations to corresponding risk values or distributions. For example, Pierson et al. proposes a path planning method for car overtake, and defines the risk as a combined H-function over the space [16]. Yang et al. builds a neural network approximation to generalize the representation of risk values across the environmental field [17]. To the best of our knowledge, no prior work in risk map building assumes probabilistic dependencies in the risk map. This paper pioneers in a way of building the risk map through explicitly establishing the spatial correlation among risk values in the environment. In this way, when we get measurements of a certain subset of the risk values, we are able to update and hence predict the risk value distributions over other areas in the environment. It helps degrades the uncertainty in risk map, and hence makes the safe path planning phase easier.

[1]It is worth noting that the definition of RV risk is usually required to be first order stochastically dominant (FSD) [14]

C. Safe Decision Making in GP-Regulated Environment

Safe decision making and/or planning under uncertainty has been gaining increasing popularity over the past several years, and one of the research directions is to assume that the environmental uncertainties are regulated by Gaussian processes [18]. Berkenkamp et al. propose a model-based safe exploration strategy to guarantee the stability of a control system while exploring the unknown but safe regions of interest [19]; Turchetta et al. applies the safe exploration strategy in GP-regulated environment to the model-free settings, and demonstrate its applicability to UAV's safe exploration [20]. More and more researchers are investigating safe decision making and/or planning algorithms in GP-regulated environment, see [21]. Wachi et al. further investigate safe reinforcement learning techniques in GP-regulated environment, and showcase the Mars' Rover safe navigation in simulation [22].

The SPPAM approach assumes GP-regularities in environmental uncertainties, but different from most of the researches in safe decision making, SPPAM is essentially a path planning algorithm which plans an executable path for AVs while ensuring safety. State-of-the-art safe decision making techniques in GP-regulated environment are targeting at different objectives (either pure exploration or learning), thus are not directly applicable for path planning tasks.

III. Safe Path Planing and Mapping (SPPAM)

This section introduces the SPPAM methodology: we first describe the target AV application scenario, and then we introduce the risk map representation and its update process for stationary obstacles, followed by the dynamic obstacle risk map update and prediction process. After that, the SPPAM method with its improvements is presented, followed by the algorithm flow diagram.

A. Application Scenario, Assumptions and Risk Map Representation

The scenario that we are looking at is to navigate an autonomous vehicle from an origin (O) to a destination point (D) in a two dimensional (2D) environment with (possibly dynamic) obstacles, which can be pedestrians, bicycles or other vehicles. We presume that the ego vehicle has real time object detection and tracking algorithms which output the real time estimated location ($\hat{\boldsymbol{\mu}}_i$) and speed ($\hat{\boldsymbol{v}}_i$) of obstacle i. Both $\hat{\boldsymbol{\mu}}_i$ and $\hat{\boldsymbol{v}}_i$ are assumed to be Gaussian Processes (GPs) with known variance and the mean of the GP is the true value of corresponding estimators, i.e., $\hat{\boldsymbol{\mu}}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_\mu)$ and $\hat{\boldsymbol{v}}_i \sim \mathcal{N}(\boldsymbol{v}, \boldsymbol{\Sigma}_v)$. For localization and path planning,

the 2D environment is discretized as grids, and we assume that the ego vehicle knows its current location s_t, and can deterministically transit to $s_{t+1} \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ refers to the set of grids who are reachable from s_t, i.e. s_t's neighbours.

The risk map of the 2D environment is a spatially varying function, which describes the probability that any point in the environment is possessed by an obstacle, i.e., $p(x, y)$, where x and y are the coordinates of the 2D environment[2]. We use $p_i(x, y)$ to denote the probability that obstacle i is in position (x, y), and the following equation holds: $p(x, y) = 1 - \prod_{j=1}^{m}(1 - p_j(x, y))$, where m is the total number of obstacles in the environment. The equation is derived from [23].

B. Risk Map Update Process for Static Obstacles

Now, we have modelled the output of the object detection module as a Gaussian process and the risk map as a function delivering the probability that any point is possessed by an obstacle. Another assumption is that the prior of the obstacle location distribution ($p(u)$) is also a GP, whose mean and variance are $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$, i.e. $\mu \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. We will relax this assumption a bit during the implementation phase, in which we approximate the prior mean with the object detection module's first output. Focusing on one static obstacle in the environment, with a streamline of the outputs from the object detection module ($\hat{\boldsymbol{\mu}}_1$,$\hat{\boldsymbol{\mu}}_2$,...,$\hat{\boldsymbol{\mu}}_k$), and we then derive the risk map update process.

We denote $\hat{\boldsymbol{\mu}}_{t:t+k}$ as the streamline output of the object detection module from time t to time $t+k$, and $p(\mu|\hat{\boldsymbol{\mu}}_{t:t+k})$ as the posterior distribution after k outputs from the object detection module. The risk map update process is to derive $p(\mu|\hat{\boldsymbol{\mu}}_{t:t+k})$ given $p(\mu|\hat{\boldsymbol{\mu}}_{t:t+k-1})$ and $\hat{\boldsymbol{\mu}}_k$. The derivation procedure is as follows:

$$
\begin{aligned}
p(\boldsymbol{\mu}|\hat{\boldsymbol{\mu}}_{1:k}) &\propto p(\hat{\boldsymbol{\mu}}_{1:k}|\boldsymbol{\mu})p(\boldsymbol{\mu}) \\
&= p(\boldsymbol{\mu}) \prod_{i=1}^{k} p(\hat{\boldsymbol{\mu}}_i|\boldsymbol{\mu}) \\
&\propto \exp\big(-\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\big) \\
&\prod_{i=1}^{k} \exp\big(-\frac{1}{2}(\hat{\boldsymbol{\mu}}_i-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}_\mu^{-1}(\hat{\boldsymbol{\mu}}_i-\boldsymbol{\mu})\big) \\
&= \exp\Big(-\frac{1}{2}\big((\boldsymbol{\mu}-\boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_0) \\
&+ \sum_{i=1}^{k}(\hat{\boldsymbol{\mu}}_i-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}_\mu^{-1}(\hat{\boldsymbol{\mu}}_i-\boldsymbol{\mu})\big)\Big) \\
&\propto \exp\Big(-\frac{1}{2}\big(\boldsymbol{\mu}^\top(\boldsymbol{\Sigma}_0^{-1}+k\boldsymbol{\Sigma}_\mu^{-1})\boldsymbol{\mu} \\
&- 2(\boldsymbol{\mu}_0^\top \boldsymbol{\Sigma}_0^{-1} + \sum_{i=1}^{k}\hat{\boldsymbol{\mu}}_i^\top \boldsymbol{\Sigma}_\mu^{-1})\boldsymbol{\mu}\big)\Big) \\
&\propto \exp\Big(-\frac{1}{2}\big((\boldsymbol{\mu}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\mu}-\boldsymbol{\mu}_k)\big)\Big),
\end{aligned}
$$

where $\boldsymbol{\Sigma}_k = (\boldsymbol{\Sigma}_0^{-1} + k\boldsymbol{\Sigma}_\mu^{-1})^{-1}$ and $\boldsymbol{\mu}_k = (\boldsymbol{\Sigma}_0^{-1} + k\boldsymbol{\Sigma}_\mu^{-1})^{-1}(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + \sum_{i=1}^{k}\boldsymbol{\Sigma}_\mu^{-1}\hat{\boldsymbol{\mu}}_i)$. The on-line risk map update process (which computes $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ given real time input $\hat{\boldsymbol{\mu}}_k$ and $\boldsymbol{\mu}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$) is straightforward, which can be expressed as: $\boldsymbol{\Sigma}_k = (\boldsymbol{\Sigma}_{k-1}^{-1} + \boldsymbol{\Sigma}_\mu^{-1})^{-1}$, and $(\boldsymbol{\Sigma}_{k-1}^{-1} + \boldsymbol{\Sigma}_\mu^{-1})^{-1}(\boldsymbol{\Sigma}_{k-1}^{-1}\boldsymbol{\mu}_{k-1} + \boldsymbol{\Sigma}_\mu^{-1}\hat{\boldsymbol{\mu}}_k)$.

From the risk map derivation process, it can be seen that the posterior distribution of the obstacle given a streamline of outputs from the object detection module follows a Gaussian process with mean $\boldsymbol{\mu}_k$ and covariance function $\boldsymbol{\Sigma}_k$, which is calculated in the preceding paragraph. For environment with only one obstacle, the risk map is represented as $p(x, y) = 1/\sqrt{|2\pi\boldsymbol{\Sigma}_k|}\exp\big(-\frac{1}{2}((x, y)^\top - \boldsymbol{\mu_k})^\top \boldsymbol{\Sigma}_k^{-1}((x, y)^\top - \boldsymbol{\mu_k})\big)$. For environment with m static obstacles, with risk map for each obstacle represented as $p_i(x, y)$, the overall risk map is calculated as: $p(x, y) = 1 - \prod_{i=1}^{m}(1 - p_i(x, y))$.

C. Risk Map Update Process for Dynamic Obstacles

For dynamic obstacles, the risk map representation is more complex than the static one, as the risk value for a point in the environment is defined as the probability that any object is in that point *within a period of time*. The risk map for dynamic obstacles is thus a spatially varying function ($p(x, y)$), which describes the probability that there is an obstacle in that point within a specific time horizon (t to $t+k$), which is referred as k-step lookahead risk map. We prescribe that the ego vehicle is equipped with both object detection module which outputs the (possibly noisy) position ($\hat{\boldsymbol{\mu}}_t$) of the object and object tracking module which outputs the noisy velocity ($\hat{\boldsymbol{v}}_t$) of the object.

It has been assumed that both $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{v}}_t$ are GPs with mean value at the corresponding true values ($\boldsymbol{\mu}_t$, and $\boldsymbol{v}_t$ respectively), and known variance denoted as $\boldsymbol{\Sigma}_u$ and $\boldsymbol{\Sigma}_v$, respectively. We further make a reasonably assumption that the acceleration $\boldsymbol{a}_k$ of the object is also a GP with mean $\boldsymbol{a}_{k-1}$ and known covariance function $\boldsymbol{\Sigma}_a$. We first focus on the risk map calculation process for one dynamic obstacle case, and then derive risk map for multiple dynamic obstacles.

This risk map update problem can be decomposed into two phases. The first phase (called filter phase) is to calculate the distribution of $\boldsymbol{\mu}_k$ and $\boldsymbol{v}_k$ given the series of detection and tracking outputs from time 1 to time k, and the second phase (called the prediction phase) is to predict the distribution of $\boldsymbol{\mu}_{k+1}$ and $\boldsymbol{v}_{k+1}$ given $\hat{\boldsymbol{\mu}}_{1:k}$ and $\hat{\boldsymbol{v}}_{1:k}$.

Before laying down the calculation process of the filter phase and the prediction phase, we first display the following three theorems related to GP omitting the proving process due to page limitations.

Theorem 1. *Given $\boldsymbol{X} \sim GP(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{X} \in \mathcal{R}^n$, and $\boldsymbol{A} \in \mathcal{R}^{m\times n}$. We have $\boldsymbol{Y} = \boldsymbol{AX}$ which is also a GP, i.e. $\boldsymbol{Y} \sim GP(\boldsymbol{A\mu}, \boldsymbol{A\Sigma A}^\top)$.*

[2]Note that the environment, when used for obstacle and risk map description, we treat it as a continuous map, and when used for path planning and localization, we treat it as a discretized grid map. The transformation between continuous map and discrete one is straightforward. We can assume that each grid is 1×1, and whatever quantity computed for the continuous environment representation is equal to the discrete case.

Theorem 2. *Given $\boldsymbol{X} \sim GP(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\boldsymbol{Y} \sim GP(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, a RV $\boldsymbol{Z}$ whose pdf is defined as $p(z) = p_x(z)p_y(z)$ is also a GP, with mean and covariance matrix as $\boldsymbol{\mu}_z = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2)$ and $\boldsymbol{\Sigma}_z = (\boldsymbol{\Sigma_1}^{-1} + \boldsymbol{\Sigma_2}^{-1})^{-1}$ respectively.*

Theorem 3. *Given $\boldsymbol{X} \sim GP(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\boldsymbol{Y}|\boldsymbol{X} \sim GP(\boldsymbol{x}, \boldsymbol{\Sigma}_2)$, then $\boldsymbol{Y}$ is also a GP, with covariance matrix $\boldsymbol{\Sigma}_y = \left(\boldsymbol{\Sigma}_2^{-1} - \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\Sigma}_2^{-1} + \boldsymbol{\Sigma}_1^{-1})^{-1}\boldsymbol{\Sigma}_2^{-1}\right)^{-1}$ and mean $\boldsymbol{\mu}_y = \boldsymbol{\Sigma}_y\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\Sigma}_2^{-1} + \boldsymbol{\Sigma}_1^{-1})^{-1}\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1$, i.e. $\boldsymbol{Y} \sim GP(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$.*

With the three theorems above, we are ready to proceed with the risk map derivation process for dynamic obstacles. The filter phase is to calculate $p\big((\boldsymbol{\mu}_k, \boldsymbol{v}_k)|\hat{\boldsymbol{\mu}}_{1:k}, \hat{\boldsymbol{v}}_{1:k}\big)$.

Theorem 4. *Given $\boldsymbol{\mu}_0 \sim GP(\bar{\boldsymbol{\mu}}_0, \bar{\boldsymbol{\Sigma}}_0^{\boldsymbol{\mu}})$, $\boldsymbol{v}_0 \sim GP(\bar{\boldsymbol{v}}_0, \bar{\boldsymbol{\Sigma}}_0^{\boldsymbol{v}})$ and $\forall\ k \geq 0$, $\boldsymbol{a}_k \sim GP(\bar{\boldsymbol{a}}_{k-1}, \bar{\boldsymbol{\Sigma}}_0^{\boldsymbol{a}})$, we have $\big((\boldsymbol{\mu}_k, \boldsymbol{v}_k)|(\hat{\boldsymbol{\mu}}_{1:k}, \hat{\boldsymbol{v}}_{1:k})\big) \sim GP(\bar{\boldsymbol{\mu}}_k\bar{\boldsymbol{\Sigma}}_k)$ with $\bar{\boldsymbol{\mu}}_k$ and $\bar{\boldsymbol{\Sigma}}_k$ recursively represented as a function of $\bar{\boldsymbol{\mu}}_{k-1}$ and $\bar{\boldsymbol{\Sigma}}_{k-1}$.*

The proving process is omitted here due to page limitations, the key idea is to recursively derive the conjugate relationship between posterior ($\boldsymbol{\mu}_k$ and $\boldsymbol{v}_k$ given new detection results) and prior ($\boldsymbol{\mu}_{k-1}$ and $\boldsymbol{v}_{k-1}$)). We present the recursive computation procedure of $\bar{\boldsymbol{\mu}}_k$ and $\bar{\boldsymbol{\Sigma}}_k$ here. $\bar{\boldsymbol{\Sigma}}_k^s = \Big(\boldsymbol{\Sigma}_s^{-1} - \boldsymbol{\Sigma}_s^{-1}\big(\boldsymbol{\Sigma}_s^{-1} + (\boldsymbol{A}\bar{\boldsymbol{\Sigma}}_{k-1}^s\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{\Sigma}_0^a\boldsymbol{b}^\top)^{-1}\big)^{-1}\boldsymbol{\Sigma}_s^{-1}\Big)^{-1}$, and $\bar{\boldsymbol{\mu}}_k^s = \bar{\boldsymbol{\Sigma}}_k^s(\boldsymbol{A}\bar{\boldsymbol{\Sigma}}_{k-1}^s\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{\Sigma}_0^a\boldsymbol{b}^\top)^{-1}\big(\boldsymbol{\Sigma}_s^{-1} + (\boldsymbol{A}\bar{\boldsymbol{\Sigma}}_{k-1}^s\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{\Sigma}_0^a\boldsymbol{b}^\top)^{-1}\big)^{-1}(\boldsymbol{A}\bar{\boldsymbol{\Sigma}}_{k-1}^s\boldsymbol{A}^\top + \boldsymbol{b}\boldsymbol{\Sigma}_0^a\boldsymbol{b}^\top)^{-1}\hat{\boldsymbol{s}}_k$, where the definitions of $\boldsymbol{A}$ and $\boldsymbol{b}$ are presented in Theorem 5.

Theorem 5. *Given that $\boldsymbol{s}_k|\boldsymbol{s}_0, \hat{\boldsymbol{s}}_{1:k} \sim GP(\bar{\boldsymbol{s}}_k, \bar{\boldsymbol{\Sigma}}_k)$, and $\boldsymbol{a}_k \sim GP(\boldsymbol{0}, \boldsymbol{\Sigma}_a)$, the RV $\boldsymbol{s}_{k+1}$ is also a GP.*

Proof. We can represent $\boldsymbol{s}_{k+1}$ from $\boldsymbol{s}_k$ and $\boldsymbol{a}_k$ as follows[3]: $\boldsymbol{s}_{k+1} = \boldsymbol{A}\boldsymbol{s}_k + \boldsymbol{b}\boldsymbol{a}_k$, where $\boldsymbol{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$ and $\boldsymbol{b} = \begin{bmatrix} 0 \\ \Delta_t \end{bmatrix}$. We can see that $\boldsymbol{s}_{k+1}$ is an affine transformation of $\boldsymbol{s}_k$ and $\boldsymbol{a}_k$. According to Theorem 1, we conclude that $\boldsymbol{s}_{k+1}$ is also a GP. We have $\boldsymbol{s}_{k+1} \sim \text{GP}(\bar{\boldsymbol{s}}_{k+1}, \bar{\boldsymbol{\Sigma}}_{k+1})$, where $\bar{\boldsymbol{s}}_{k+1} = \boldsymbol{A}\bar{\boldsymbol{s}}_k$ and $\bar{\boldsymbol{\Sigma}}_{k+1} = \boldsymbol{A}\bar{\boldsymbol{\Sigma}}_k\boldsymbol{A}^\top + b\boldsymbol{\Sigma}_a b^\top$.

□

D. Risk Map Representation for both Static and Dynamic Obstacles

As we have described in the first subsection, the risk map ($p(x, y)$) is a spatially varying function describing the probability of having an obstacle in (x, y), however, when dealing with dynamic obstacles, we have to include the probability of having an obstacle in the future time step. For dynamic obstacle i, we denote $p_k^i(x, y)$ as the probability that obstacle i is in position (x, y) at time step k. For static obstacle j, we denote $p^j(x, y)$ as the probability that obstacle i is in position (x, y). We define $p(x, y)$ as the probability that there is at least one obstacle in position (x, y) from time step t to $t + k$; suppose there are m static obstacles and n dynamic obstacles, then, $p(x, y)$ can be calculated as $p(x, y) = 1 - \prod_{i=1}^m(1 - p^i(x, y))\prod_{j=1}^n(1 - p_k^j(x, y))(1 - p_{k+1}^j(x, y))$.

As $p^i(x, y)$ for static obstacles, $p_t^j(x, y)$ to $p_{t+k}^j(x, y)$ for dynamic obstacles can be analytically derived out of the object detection and tracking module's data stream output from the last two subsections, then the risk map is obtained.

E. Safe Path Planning with Gaussian Process Regulated Risk Map

Suppose that the risk map which is a spatially varying function has been calculated based on the streamline input from the object detection and tracking module, SPPAM will calculate the shortest safe path connecting the ego vehicle from an origin (O) to a destination point (D) in the environment with possibly dynamic obstacles. To formulate SPPAM into a mathematical programming framework, we first need a clear definition of safety, and hence the definition of a safe path.

Definition 1. *An ϵ-safe grid: A grid in a 2D grid map is said to be 'ϵ-safe' if the probability that the grid has any obstacle between the current time step k and the next time step $k + 1$ is less than ϵ.*

Definition 2. *An ϵ-safe path: A path is said to be 'ϵ-safe' if all the nodes along the path are ϵ-safe.*

Therefore, the SPPAM algorithm is to find the shortest ϵ-safe path with the risk map. We wish to note here that we treat each grid in the grid map as a 1×1 grid, and approximate the probability that any grid has an obstacle as $p(x, y) \times 1 \times 1 = p(x, y)$, in which x, y refer to the center coordinate of the grid. With the two definitions above, we can formulate the SPPAM problem as:

$$\begin{aligned} \underset{\boldsymbol{g_1}, \cdots, \boldsymbol{g_T}}{\text{minimize}} \quad & T \\ \text{subject to} \quad & g_t \in \mathcal{A}(g_{t-1}) \qquad \forall 1 \leq t \leq T \\ & p(g_t) \leq \epsilon \qquad \forall 1 \leq t \leq T \\ & g_0 = s_0 \\ & g_T = s_d, \end{aligned} \tag{1}$$

where s_0 and s_d are the origin grid and destination grid of the ego vehicle, respectively; $\mathcal{A}(g_{t-1})$ refers to the set of grids that are achievable from g_{t-1}, i.e, the neighbour set of g_{t-1}, and $p(g_t)$ is the risk value of the grid at g_t.

The problem as defined in Eq. 1 is a constrained shortest path problem. We can use Dijkstra's algorithm to get the solution by removing the 'un-safe' grids in the map. However, in reality, the ego vehicle is navigating in dynamic environment, the path needs to be adapted with new inputs, and on the other hand, the safety criterion might be too strict to reach a feasible path, let alone the shortest path. Therefore, we improve the formulation of the SPPAM problem to consider the infeasible path planning problem, and let the ego vehicle find the 'overall safest' path and take the immediate

[3]Note that we are being a little sloppy here in that every element of $\boldsymbol{A}$ and $\boldsymbol{b}$ should be multiplied by an identity matrix $\boldsymbol{I} \in \mathcal{R}^{2\times 2}$, because $\boldsymbol{s}_t$ and $\boldsymbol{v}_t$ are two dimensional vectors rather than scalar variables. Nevertheless, the current representation forms of $\boldsymbol{A}$ and $\boldsymbol{b}$ convey the same meaning.

action to one of the ϵ-safe nodes during path planning. The improved SPPAM problem formulation is as follows:

$$\begin{aligned} \underset{g_1,\cdots,g_T}{\text{maximize}} \quad & \sum_{i=1}^{T} \log\left(1-p(g_i)\right) \\ \text{subject to} \quad & g_t \in \mathcal{A}(g_{t-1}) \qquad \forall 1 \le t \le T \\ & p(g_1) \le \epsilon \qquad \forall 1 \le t \le T \\ & g_0 = s_0 \\ & g_T = s_d. \end{aligned} \tag{2}$$

The problem as defined in Eq. 2 releases the constraint that the path has to be a safe path, instead, it requires that the immediate action is a safe action (which leads to an ϵ-safe node), and the overall safety score for the path as quantized by $\sum_{i=1}^{T} \log\left(1-p(g_i)\right)$ is maximized[4]. In this way, even when the safe path per definition does not exist, the improved SPPAM is still able to yield an action which leads to an ϵ-safe node. With new information from the object detection and tracking module, the risk map is updated, then SPPAM-improved is executed again to calculate the next safe action for the ego vehicle. The solution to Eq. 2 can be achieved through Dijkstra's algorithm as well.

F. Algorithm Flow Process

The SPPAM algorithm is depicted in **Algorithm 1**. After initialization, SPPAM will loop between choosing the best and safe action, and updating the risk map according to output from object detection and track module until reach the destination. The inputs to the improved SPPAM algorithm are the risk map, an OD pair, and the safety threshold (ϵ).

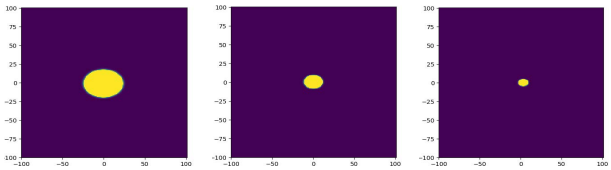
IV. Simulation Results and Analysis

This section evaluates the on-line risk map update process in a simulated environment, and Section V shows system implementation with safe path planner and the experimental results on an autonomous vehicle testbed. Simulation is performed on a PC with Ubuntu14.04 LTS, Intel Xeon(R) CPU $E5-2620$ processor and $125.8GiB$ RAM. In this section, we compare the risk map update process with [16], which proposes a heuristic risk function, applicable to both static and dynamic obstacle use cases. The safety threshold (ϵ) is set to be 0.001 throughout the simulation, which means that any point with $p(x,y) \ge \epsilon$ within the environment is deemed as a risky point.

A. Risk Map Update for Static Obstacles

We construct a very simple scenario to test whether the GP-regulated risk map update process for static obstacles works. The environment is modelled as a 100×100 square, with $-50 \le x, y \le 50$. We insert one static obstacle at the center point $(0,0)$, which means that $\boldsymbol{\mu} = [0,0]^\top$, we set $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_{\boldsymbol{\mu}} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$, where $\sigma_x^2 = 2$ and $\sigma_y^2 = 1$. We perform the risk update process for 100 time steps according to the iterative update algorithm as described in Section III-B, and at each time step k, a new input $\hat{\boldsymbol{\mu}}_k$ from the object detection module is provided.

Fig. 1 shows the snapshots of the evolving process of the ϵ-risky region[5] at $k=0$ (initial belief), $k=33$, $k=66$, respectively. We can see that the ϵ-risk region shrinks and gradually concentrates to the true obstacle location as k increases. In [16], the ϵ-risky region is an analytical function, which is not updated for static obstacles as new data comes, therefore, the risk region is always equal to Fig. 1(a).



(a) k=0 (initial belief) (b) k=33 (c) k=66

Fig. 1. Risk map update process for a static obstacle, the size of ϵ-risky region shrinks as more observations become available; yellow color refers to the ϵ-risk region.

We want to further verify the hypothesize that with more inputs from the object detection module, our proposed risk map update algorithm will have more understanding about the 'true' location of the obstacle, thus the posterior distribution of the obstacle will have lower entropy (smaller $|\boldsymbol{\Sigma}|$), and the posterior mean will asymptotically approach the 'true' location of the obstacle.

Algorithm 1: The Improved SPPAM Algorithm Flow Process

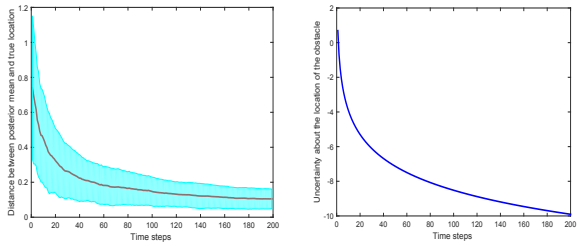
Input: safety threshold: ϵ, starting node s_0, destination node s_d,risk map $p(g_t)$, prior for m stationary obstacles and n dynamic obstacles

Output: Safe path: contains $g_0, g_1, \cdots, g_d$

1 $k=0$ and $g_0 = s_0$;
2 $\forall\ g_{k+1} \in \mathcal{A}(g_k)$, evaluate $p_{g_{k+1}}$;
3 **if** $r_{g_{k+1}} < \epsilon$ **then**
4 Apply Dijkstra's algorithm to solve Eq. 2;
5 Execute g_{k+1};
6 **if** $s_{k+1} \neq s_d$ **then**
7 $\forall 1 \le i \le m$ get $\hat{\boldsymbol{\mu}}^i_{k+1}$ from object detection module;
8 $\forall 1 \le j \le n$ get $\hat{\boldsymbol{\mu}}^j_{k+1}$ and $\hat{\boldsymbol{v}}^j_{k+1}$ from the object detection and tracking module;
9 update the risk map $(p(x,y))$ as described in Section III-B and Section III-C;
10 Set $k = k+1$;
11 return to Line 2;
12 Final.

[4]The overall safety score of a path is the probability that all the nodes contained in the path does not have any obstacle. Therefore, it can expressed as $\prod_{i=1}^{T}\left(1-p(g_i)\right)$. Taking the log transformation, it becomes the objective in Eq. 2.

[5]ϵ-risky region contains all the points with the risk value greater than ϵ, and is painted yellow.



(a) (b)

Fig. 2. (a) The mean and standard deviation of the distance between the estimated posterior mean and the true location of the obstacle. Blue color shows standard deviation for 200 independent simulation runs; (b) The uncertainty of the obstacle's location VS. time steps.

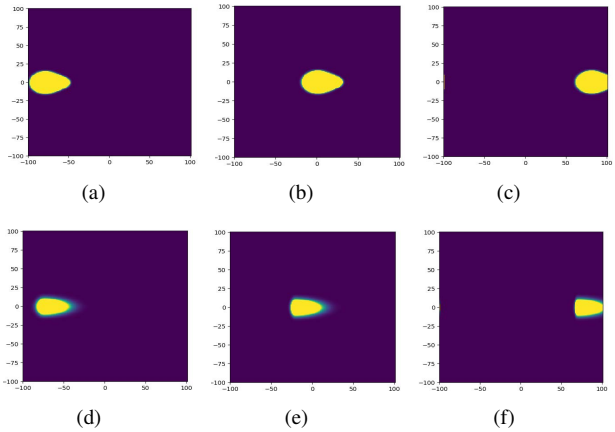
Fig. 2(a) shows the Euclidean distance between the posterior mean and the true location of the obstacle. Note that we run the simulation for 200 independent runs, and therefore show the mean results per step with standard deviation values shown in blue colors. In the figure, we can see that the posterior mean of the location estimation approaches the true location asymptotically, and the uncertainty (quantified as the standard deviation of the distance between the posterior mean and the true location) also decreases which is visualized as the width of the blue color. Fig 2(b) shows the evolving curve of the uncertainty of the environment. We can see that the posterior uncertainty about the location of the obstacle (quantized as $\log|\mathbf{\Sigma}|$) decreases steadily as the time step increases (because at each time step, there is a new input from the object detection module). This result verifies the hypothesize that with more data, the environment uncertainty decreases, as also reported in [23].

B. Risk Map Update for Dynamic Obstacles

This section focuses on evaluating the risk map update procedure for dynamic obstacles, and we construct two simple yet representative scenarios[6]. In the first scenario referred as the linear case, an obstacle is moving from $(-100, 0)$ to $(100, 0)$ with a starting speed $v_0 = 5$, and constant acceleration $a = 1$. In the second scenario referred as the circular case, an obstacle is moving with a constant linear velocity ($v_0 = 5$) in a circle centered at $(0, 0)$ with a radius of 75. These two simple scenarios are representative in that any other curved movement can be mathematically decomposed into a circular move and a linear move. We use the on-line risk map update procedure as described in Section III-C.

Fig. 3 shows comparison between k-step look-ahead GP-regulated risk map out of SPPAM ($k = 3$) and that of [16] for the linear case. The comparison of the two algorithms for the circular case are shown in Fig. 4. Note that it is difficult or unfair to compare the risk maps of the two algorithms quantitively because they are looking at different metrics and there is no absolute ground truth for the testing scenario. What we can see is that the GP-regulated risk map is able to generate curved risk map (as shown in the circular case) which is more realistic in that the obstacle may move along curves, while the work in [16] is predicting risk values in a straight-line fashion.

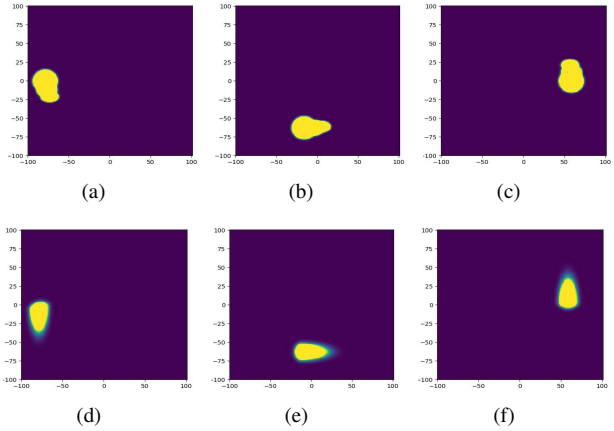
[6]The environment is a 200×60 rectangle, and in the two scenarios, we assume that there is only one obstacle which is the ego moving obstacle.



(a) (b) (c)

(d) (e) (f)

Fig. 3. Comparison of risk map update process for the linear case; (a)-(c) refer to SPPAM risk map, and (d)-(f) refer to risk map in [16]. Yellow color refers to ϵ-risky region for SPPAM risk map, and risky region for [16]



(a) (b) (c)

(d) (e) (f)

Fig. 4. Comparison of risk map update process for the circular case; (a)-(c) refer to SPPAM risk map, and (d)-(f) refer to risk map in [16]. Yellow color refers to ϵ-risky region for SPPAM risk map, and risky region for [16]

From the figures, we can see that our algorithm performs similarly with the work in [16] for the linear velocity case. However, for the circular case, our algorithm works better in that it can predict the future 'curved' path of the moving obstacle, with the estimated posterior distribution of $\boldsymbol{a}_k$ and $\boldsymbol{v}_k$, while the work in [16] assumes a linear constant velocity, thus cannot make 'accurate' risk map predictions for the circular case. In essence, the SPPAM risk map is, in fact, a spatially correlated pdf (probability distribution function) describing the probability of having an obstacle in a point within a time period. It has much more physical meaning than other heuristic risk functions which predefines a certain risk region around the observation.

V. System Integration and In-Vehicle Test

We have implemented the on-line risk map update process for both static and dynamic obstacles, and the safe path planning algorithm in two AV testbeds (a buggy and a car). 2D map is constructed through laser-based SLAM (Simultaneous Localization and Mapping), and the adaptive Monte-Carlo localization (AMCL) method [24] is used for localization. For car detection, we employ a CNN-based 3D object detection algorithm - Sparsely Embedded Convolutional Detection (SECOND [25]) as our backbone detection pipeline and modified the Voxel-Feature-Extraction (VFE) layers to include Variational Encoding (VE) layers, so as to provide generative voxel features for dealing with sparse/less-dense point cloud data. For pedestrian detection, we use the SVM trained moving object detection algorithm [26] for 2D lidar data stream, and 3D point cloud mask based model fitting for the 3D Lidar data stream, respectively. The detection modules output the bounding boxes of detected targets (i.e., car, pedestrian) containing the class, score, pose, and box dimensions. For object tracking, we employ Kalman filter and Hungarian Algorithm [27] as the backbone tracking pipeline and develop a novel data association manoeuvre with hand crafted velocity prior in certain regions where symbolic road context information is available [28]. Inside the Kalman filter, we assume a constant velocity motion transition as the common practice, but for places with curvature information or crosswalks, we embed the updated velocity with prior belief that pedestrians most probably will cross the road on crosswalks or follow the road curvatures. This is due to the assumption that velocity changes usually occur when the target enters or leaves road sections such as roundabouts, turnings, crossings, where the constant-velocity model does not fit in. The software framework of the AV testbed is shown in Fig. 5, which is adapted from [29].

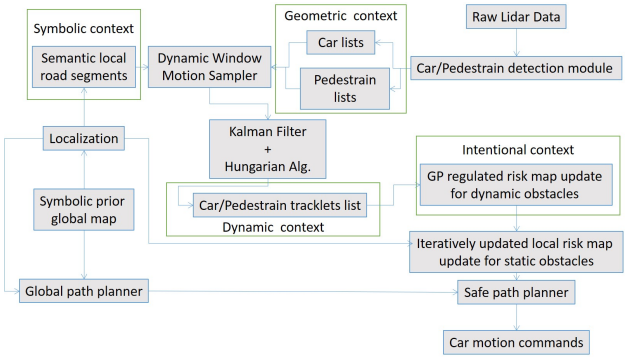


Fig. 5. The software framework implemented in the AV testbed.

Fig. 6 shows the object detection and tracking results, and Fig. 7 shows risk map update and corresponding safe path planner. The obstacle detection and tracking module is able to operate at 10Hz, risk map update together with safe path planner operate at 5Hz, and the micro-controller is operating at 40Hz. During in-field test, the ego vehicle reacts to the safe path planner within 0.2s. Hence, the safe path planner is able to react in real time (reaction time within one second), and plan a path avoiding potential collisions with other road participants. It is also able to re-plan another path whenever it believes that the previous planned one is unsafe[7].

VI. Conclusion and Future Works

This paper proposes a SPPAM approach for safe path planning with GP-regulated risk map. Taking advantage of the spatial dependencies of GPs, SPPAM is able to update the risk map and re-plan the safe path accordingly. Simulation results show that the on-line risk map update process gradually gains more understanding about the obstacles, as more object detection and tracking outputs become available, which also confirms our initial hypothesis in [23]. Experimental results on real AV testbeds show the feasibility of SPPAM reacting in real time to pedestrians and moving cars.

Currently, we have assumed accurate localization of the ego vehicle, and we didn't consider vehicle movement constraints, such as holonomic constraints. In the future, we plan to additionally model the ego vehicle's location within the map as a GP, and improve SPPAM to consider vehicle's holonomic constraints as well. Another direction is to include speed control explicitly into the safe path planner so that the ego vehicle is able to follow other vehicles to pass a narrow passenger instead of trying to overtake them aggressively, as currently shown in Fig. 7(c) and Fig. 7(d). We are also keen on including more road context information such as road markings, lane information, into the SPPAM algorithm as well, similar to the work in [29].

Acknowledgement

This research was partially supported by the National Research Foundation, Prime Ministers Office, Singapore, under its CREATE programme, Singapore-MIT Alliance for Research and Technology (SMART) Future Urban Mobility (FM) IRG. We gratefully acknowledge the technical support of Nvidia Corporation through the Memorandum of Understanding with the Advanced Robotics Centre of the National University of Singapore on autonomous systems technologies.

References

[1] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *CoRR*, vol. abs/1604.07446, 2016.

[2] H. Mouhagir, V. Cherfaoui, R. Talj, F. Aioun, and F. Guillemard, "Trajectory planning for autonomous vehicle in uncertain environment using evidential grid," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 545 – 12 550, 2017, 20th IFAC World Congress.

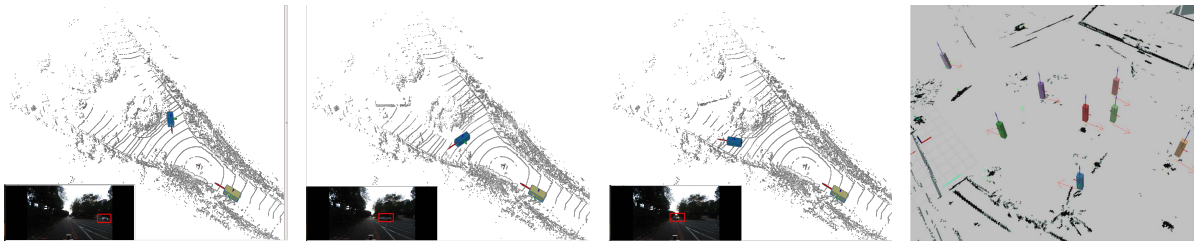
[3] S. Singh. (2015) Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Traffic Safety Facts Crash Stats.

[4] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 174–179, 2017.

[5] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

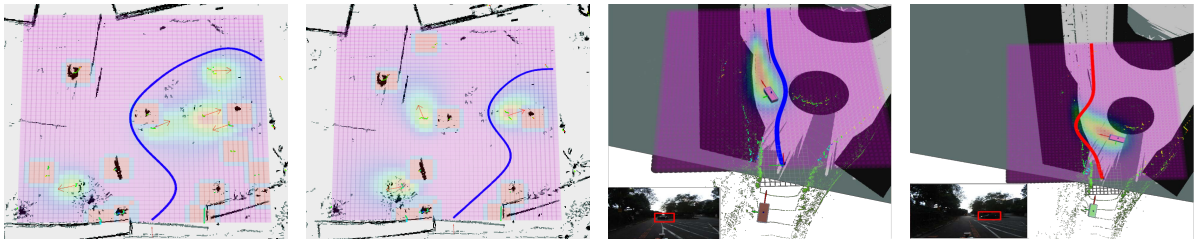
[6] D. Sharma and S. K. Dubey, "Anytime A * Algorithm-An Extension to A* Algorithm," *International Journal of Scientific & Engineering Research*, vol. 4, no. 1, 2013.

[7]More demonstrative videos are provided `https://drive.google.com/open?id=1wLHuEQV2IkC3PUQ4ObNPEk7bcTSfE6cK`



(a) (b) (c) (d)

Fig. 6. Real time object detection and tracking on AV testbeds; (a), (b) and (c) are for car detection and tracking, (d) is for pedestrian detection and tracking; in-vehicle camera data with red bounding box are shown inset (a), (b) and (c). All the figures are best viewed in color.



(a) (b) (c) (d)

Fig. 7. Risk map and local safe path planner showcase on AV testbeds. (a) and (b) are tested in free space with pedestrians, and (c) and (d) are tested against cars in a roundabout. When there is no feasible safe path, improved SPPAM is triggered, and the path is visualized in red. Red path triggers slow moving speed and high replanning frequency in system integration. More video demonstrations are available in `https://drive.google.com/open?id=1wLHuEQV2IkC3PUQ4ObNPEk7bcTSfE6cK`.

[7] S. Koenig and M. Likhachev, "D*lite," in *Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 476–483.

[8] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

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

[10] H. Andersen, W. Schwarting, F. Naser, Y. Eng, M. H. Ang, D. Rus, and J. Alonso-Mora, "Trajectory optimization for autonomous overtaking with visibility maximization," 10 2017, pp. 1–8.

[11] M. Naderan-Tahan and M. T. Manzuri-Shalmani, "Efficient and safe path planning for a mobile robot using genetic algorithm," in *IEEE Congress on Evolutionary Computation*, May 2009, pp. 2091–2097.

[12] S. Bouraine, T. Fraichard, and O. Azouaoui, "Real-time Safe Path Planning for Robot Navigation in Unknown Dynamic Environments," in *CSA 2nd Conference on Computing Systems and Applications*, 2016.

[13] N. MacMillan, R. Allen, D. Marinakis, and S. Whitesides, "Risk averse motion planning for a mobile robot," 2011.

[14] D. Li, P. Weng, and O. Karabasoglu, "Finding risk-averse shortest path with time-dependent stochastic costs," *CoRR*, vol. abs/1701.00642, 2017.

[15] W. Dangelmaier, B. Klopper, J. Wienstroer, and A. Doring, "Risk averse shortest path planning in uncertain domains," in *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*, Nov 2006, pp. 115–115.

[16] A. Pierson, W. Schwarting, S. Karaman, and D. Rus, "Navigating Congested Environments with Risk Level Sets," in *ICRA 2018 - International Conference on Robotics and Automation*, 2018.

[17] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Multi-cost optimal route planning under time-varying uncertainty," 2017.

[18] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *CoRR*, vol. abs/1606.04753, 2016.

[19] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. of Neural Information Processing Systems (NIPS)*, 2017.

[20] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite Markov Decision Processes with Gaussian processes," in *Proc. of Neural Information Processing Systems (NIPS)*, 2016.

[21] S. Manjanna, N. Kakodkar, M. Meghjani, and G. Dudek, "Efficient terrain driven coral coverage using gaussian processes for mosaic synthesis," in *2016 13th Conference on Computer and Robot Vision (CRV)*. IEEE, 2016, pp. 448–455.

[22] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained mdps using gaussian processes," in *AAAI 2018*, 2018.

[23] M. Meghjani and G. Dudek, "Multi-robot exploration and rendezvous on graphs," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5270–5276.

[24] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1, pp. 99 – 141, 2001.

[25] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018.

[26] B. Qin, Z. Chong, S. Soh, T. Bandyopadhyay, M. Jr, E. Frazzoli, and D. Rus, *A Spatial-Temporal Approach for Moving Object Recognition with 2D LIDAR*, 11 2016, pp. 807–820.

[27] S. Verma, Y. H. Eng, H. X. Kong, H. Andersen, M. Meghjani, W. K. Leong, X. Shen, C. Zhang, M. H. Ang, and D. Rus, "Vehicle detection, tracking and behavior analysis in urban driving environments using road context," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1413–1420.

[28] M. Meghjani, S. Verma, Y. H. Eng, Q. H. Ho, D. Rus, and M. H. Ang Jr., "Context-aware intention and trajectory prediction for urban driving environments," in *International Symposium on Experimental Robotics*. Springer, 2018.

[29] M. Meghjani, Y. Luo, Q. H. Ho, P. Cai, S. Verma, D. Rus, and D. Hsu, "Context and intention aware planning for urban driving," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.