Contingency planning over probabilistic obstacle predictions for autonomous road vehicles

Te SUN

Jan. 2020

1 Literature review

A main challenge for autonomous vehicles to bridge the gap between the controlled environments to the real world is due to the **inability to adequately identify predict and utilize obstacle intent**. Previous research tried to solve with two primary parts of the dynamic obstacle avoidance: 1) predicting future obstacle motion and 2) path planning in dynamic environments using predicted obstacle motions.

We have several approaches: For the prediction of obstacle motion:

1. model based: predictions of dynamic obstacles using constant velocity of fields and contexts. Use a tangent graph of the environment and a 1-D model of velocity variance along each potential path. Those methods could induce overconfident predictions and do not fully use the environment's information.
2. Nonparametric learning methods: usage of Gaussian processes to model multiple possible future trajectories or use hidden Markov model to predict mode transitions. These approaches require retrainining for each new environments

For the path-planning strategies, a typical way is to attempt to solve the full dimensional dynamic planning problem avoid directly optimizing over the continuous search space through the use of probabilistic sampling or through discretization of search space. Like a rapidly exploring random tree (RRT) planner.

To provide better dynamic planning capabilities, we incorporates probabilistic obstacle predictions to enable efficient generation of a safe set of contingency paths in dynamic environments.

In the paper, a better numerical optimization formulation is presented by: 1) adopting an efficient spline based trajectory representation, 2) solving the planning problem over a limited planning horizon and 3) using fast but accurate collision probability bounds.

Key contributions of this paper

1. simultaneous optimization of partially shared contingency trajectories over the robot's continuous planning space
2. efficient and accurate upper bound on point wise collision probability
3. a spline based trajectory representation and associated cost function and constraint definitions to express the contingency path planning problem
4. an obstacle trajectory clustering algorithm to reduce planning complexity

1.1 Obstacle prediction

Current Gaussian obstacle state estimation: (μ_i, Σ_i) for the ith of n_0 dynamic obstacles are assumed to be known from the obstacle tracker at each planning cycle. For each obstacle, a set of discrete goal modes $\mathcal{G}_i$ (e.x. straight, turn left, turn right, stop) is inferred for each obstacle based on a known topological representation. All possible goal modes for a newly observed obstacle are initialized with **equal probability**.

We can denote T as a sequence of obstacle state distributions over N time steps into the future.

$$T = \{(\mu_i, \Sigma_i), (\hat{\mu}_i, \hat{\Sigma}_i)_1, \dots (\hat{\mu}_i, \hat{\Sigma}_i)_N\}$$

And for each predicted trajectory for obstacle i, $T \in \mathcal{T}_i$. For multiple dynamic obstacles, the set of all possible permutations of predicted obstacle trajectories is defined as:

$$\boldsymbol{T} = \mathcal{T}_2 \times \mathcal{T}_2 \dots \mathcal{T}_{n_0}$$

We denote $\mathcal{T}^j$ as the jth set of trajectory permutations. We defined n_s as the cardinality of T.

$$n_s = \prod_1^{n_0} |\mathcal{T}_i|$$

The authors update the probability of each obstacle trajectory permutation $p(\mathcal{T}^j)$ as:

$$p(\mathcal{T}^j)_1 = \frac{p(\mathcal{T}^j) \prod_{i=1}^{n_0} p((\mu_i, \Sigma_i)_1 | (\hat{\mu}_i, \hat{\Sigma}_i)_1^j)}{\sum_{\mathcal{T}^m \in \boldsymbol{T}} p(\mathcal{T}^m) \prod_{i=1}^{n_0} p((\mu_i, \Sigma_i)_1 | (\hat{\mu}_i, \hat{\Sigma}_i)_1^m)}$$

That is, for the prediction of a certain trajectory j, the probability equals to the joint probability of the current trajectory $p(\mathcal{T}^j)$ with its updated states based on its own obstacle state predictions.

1.2 Path Optimization

The problem is reformulated as a general nonlinear constrained optimization problem:

$$\boldsymbol{h}_{opt} = arg \min_{\boldsymbol{h}} J(\boldsymbol{h}) C(\boldsymbol{h}) < 0$$

We can use convex optimization.

1.2.1 Contingency Planning Approach

We use a single shared path segment at the start of the path for all permutations of predicted obstacle trajectories, $\mathcal{T}^j$.

In our case, the cost function

$$J(\boldsymbol{h}) = J_{shared}(\boldsymbol{h}, \boldsymbol{T}) + \sum_{\mathcal{T}^j \in \boldsymbol{T}} p(\mathcal{T}^j) J_{conting(\boldsymbol{h}, \mathcal{T}^j)}$$

. where

$$J_{shared} = J_{shared}^{dynamics} + J_{static}^{shared} + J_{shared}^{collision}$$
$$J_{conting} = J_{conting}^{dynamics} + J_{static}^{conting} + J_{conting}^{collision} + J_{conting}^{goal}$$

The *dynamic* cost contains terms that penalize large accelerations and large path curvatures. *static* penalizes proximity to static obstacles. *Collision* penalizes high dynamic obstacle collision probabilities. And *goal* penalizes the distance between the end of a robot's planned path and its goal.

1.2.2 Spline-based path parameterization

To make the trajectory representation expressive in its ability to cover the search space and compact in its dimensionality, cubic splines are chosen for this purpose because two splines $y(t)$ and $x(t)$ define a continuous representation of the robot position (x, y).

$$\boldsymbol{h} = \{h_1^x, h_1^y | h_{2:n}^{1,x}, h_{2:n}^{1,y}, \dots h_{2:n}^{n_s,x}, h_{2:n}^{n_s,x}\}$$

$\{h_1^x, h_1^y\}$ is the end of the initial shared segment, and a separate cubic spline is defined for each contingency path starting from this point. By using a cubic spline definition, the robot position and its derivatives at a given time t depend linearly on the control point parameters $\boldsymbol{h}$.

1.2.3 Static obstacle cost map

The cost is designed to provide an easily differentiable static obstacle cost term that is convex $w.r.t$ the lateral offset of the robot from the center of the driving corridor.

$$f_{prx}(x, y) = \min_i \left(\frac{d_i}{w_i^*}\right)^2, \quad \forall i \in [1, n_c]$$

where d_i is the minimum distance between the robot position and t he ith centerline segment, w_i^* is the interpolated width of the driving corridor for the ith centerline segment at the robot position, and n_c is the total number of centerline segments.

So the robot is trying to travel align the centerline.

1.2.4 Collision Probability

Notation:

$$\begin{aligned} z_{robot} &= [x_{robot}, y_{robot}] \\ \mu^z_{obst} &= [\mu^x_{obst}, \mu^y_{obst}] \\ \Sigma^z_{obst} & \end{aligned}$$

where z_{robot} represents interpolated position of the robot and μ, Σ is the mean position and error covariance of the obstacle. In the article, the author use a normal distribution to represent the obstacle and we can use a mixture of Gaussians as a more general obstacle position distribution.

Instead of a circular overapproximation of the obstacle shape to account for uncertainties in the obstacle's orientation. We model the obstacle by using a rectangular to reduce the feasible area.

If we assume orientation of the obstacle to be an independent (independent to the position of the obstacle) Gaussian distribution $\gamma_{obst} \sim \mathcal{N}(\mu^\gamma_{obst}, (\sigma^\gamma_{obst})^2)$

By restricting the orientation ranges of the obstacle into some discrete intervals $[\gamma^{l-1}_{obst}, \gamma^l_{obst}]$ for $l \in [1, n_\gamma]$, where $\sum p([\gamma^{l-1}_{obst}, \gamma^l_{obst}]) = 1$ which means that the orientation of obstacle can not be arbitrary. If we can reduce the confidence interval by $\sum p([\gamma^{l-1}_{obst}, \gamma^l_{obst}]) = \delta_\gamma$. Then the collision probability can bounded with:

$$\begin{aligned} &p_{coll}(z_{robot}, \mu^z_{obst}, \Sigma^z_{obst}, \mu^\gamma_{obst}, \sigma^\gamma_{obst}) \\ &\qquad \leq \sum_{l=1}^{n_\gamma} p([\gamma^{l-1}_{obst}, \gamma^l_{obst}]) p^{poly}_{coll}(z_{robot}, \mu^z_{obst}, \Sigma^z_{obst}, [\gamma^{l-1}_{obst}, \gamma^l_{obst}]) \\ &\qquad + (1 - \delta_\gamma) p^{circ}_{coll}(z_{robot}, \mu^z_{obst}, \Sigma^z_{obst}) \end{aligned}$$

By doing so, we actually simplify an integral of Gaussian distribution into a discrete summation, which simplifies the numerical complexity. Nevertheless, we have to compute the probability of collision with the possibly oriented and positioned obstacle. By assuming again, the distribution follows a Gaussian distribution over the combined body of the robot shape and all possible obstacle shapes. Thus,

$$p^{poly}_{coll}(z_{robot}, \mu^z_{obst}, \Sigma^z_{obst}, [\gamma^{l-1}_{obst}, \gamma^l_{obst}]) = \int_{CB} \mathcal{N}(\mu^z_{obst}, \Sigma^z_{obst})$$

By using a decomposition of the covariance, we can decouple the distribution in order to compute more efficiently. Therefore, we can compute it independently through x and y direction.

$$\begin{aligned} &p^{poly}_{coll}(z_{robot}, \mu^z_{obst}, \Sigma^z_{obst}, [\gamma^{l-1}_{obst}, \gamma^l_{obst}]) \\ &\qquad \leq (\Psi(x^{CB}_{\max} - [\mu^x_{obst}]^{RW}; 0, 1) - \Psi(x^{CB}_{\min} - [\mu^x_{obst}]^{RW}; 0, 1)) \\ &\qquad\quad (\Psi(y^{CB}_{\max} - [\mu^y_{obst}]^{RW}; 0, 1) - \Psi(y^{CB}_{\min} - [\mu^y_{obst}]^{RW}; 0, 1)) \end{aligned}$$

where $\Psi(z; \mu, \sigma) = \frac{1}{2}[1 + erf(\frac{z-\mu}{\sqrt{2\sigma^2}})]$. CB denotes combined body, $y^{CB}_{\max}$ is the limit of the combined body.

1.2.5 Optimization

The planning procedure is reformulated into an nonlinear constrained numerical optimization problem. 1.2.1. By using some weighted coefficients before each separated term, we use some trade off between different objective. On the other hand, we formulated those cost terms to be convex to ensure the feasibility of the numerical problem.

1.3 Trajectory clustering

1.3.1 Trajectory clustering

Clustering spatial trajectories can be done by both spectral and hierarchical clustering

1.4 Limitation of this paper

The main contribution of the paper is to propose a upper bound to evaluate the dynamic collision probability. Nevertheless, we can see that this bound is found based on some strong priors: the obstacle may only rotate and move mildly.

2 Implementation

The most plausible part of the article, from my own perspective, is the proposition of a probability upper bound for the collision detection. Consequently, my efforts were targeted to reproduce the result of this part of the article.

2.1 Implementation details

The main efforts within my implementation can be concluded but not limited as:

1. 2D Gaussian distribution transformation in order to get a circle-liked decoupled Gaussian, which involves techniques like SVD, rotation, sampling.
2. Detection of overlap between two polygon. This part needs techniques like vector manipulation like cross-multiplication, segments cross detection, coordination transformation
3. probability manipulation: sampling with different configuration, Monte Carlo method (to compute the real collision probability), change of variable.
4. Visualization and the build of simulation environment

2.2 Implementation results

Firstly, we try to **find the transformation matrix** to change the base vectors in order to get a decoupled (in this case, we want to render the component X and Y independent in terms of probability integration) 2D Gaussian.

So ideally, after the transformation, the probability should be a circle-like distribution.

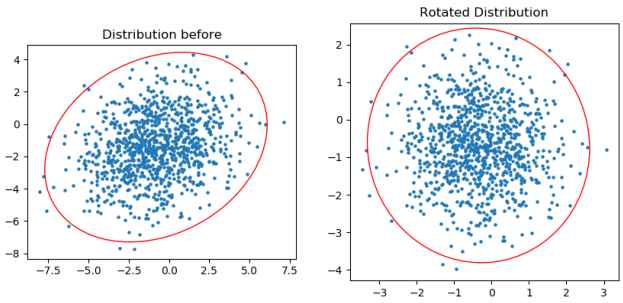

Figure 1: Probability transformation

We then formulate a combined body for robot-obstacle system showed at the left of the Figure 2. In general, find a bounding box for the obstacle that has some uncertain orientation. Consequently, this uncertainty will increase the size of obstacle's bounding box. And we sweep this bounding box along the edge of the robot to get the original combined body (Left, Figure 2).

Yet, the obstacle can have some positional uncertainty, which is generally not decoupled along x-y axis. By using the technique described above, we can find the matrix that renders the distribution decouple. On the same time, we should transform the whole system by multiplying this matrix (Middle, Figure 2)

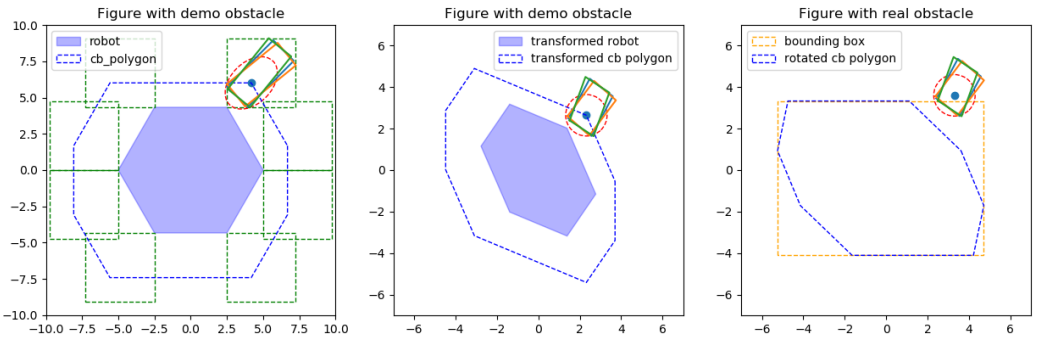

Figure 2: Combined body, original, transformed and rotated

Afterwards, we re-rotate the system to the robot-centric coordination in order to compute the the probability bound. And the probability is obtained by:

$$p_{coll}^{poly}(z_{robot}, \mu_{obst}^{z}, \Sigma_{obst}^{z}, [\gamma_{obst}^{l-1}, \gamma_{obst}^{l}]) \leq (\Psi(x_{\max}^{CB} - [\mu_{obst}^{x}]^{RW}; 0, 1) - \Psi(x_{\min}^{CB} - [\mu_{obst}^{x}]^{RW}; 0, 1)) \\ (\Psi(y_{\max}^{CB} - [\mu_{obst}^{y}]^{RW}; 0, 1) - \Psi(y_{\min}^{CB} - [\mu_{obst}^{y}]^{RW}; 0, 1))$$

At last, we implement all techniques above into a simulation. We simulate a scenario

where a the obstacle is moving in the robot coordination, with 12 timesteps. (Figure 3). The robot is the blue rectangular centered at $(0, 0)$. And the obstacle is visualised with its positional covariance (ellipse) and its angular uncertainty that is represented by two outlines having the same center as the obstacle. To simulate the scenario more logically, we increase

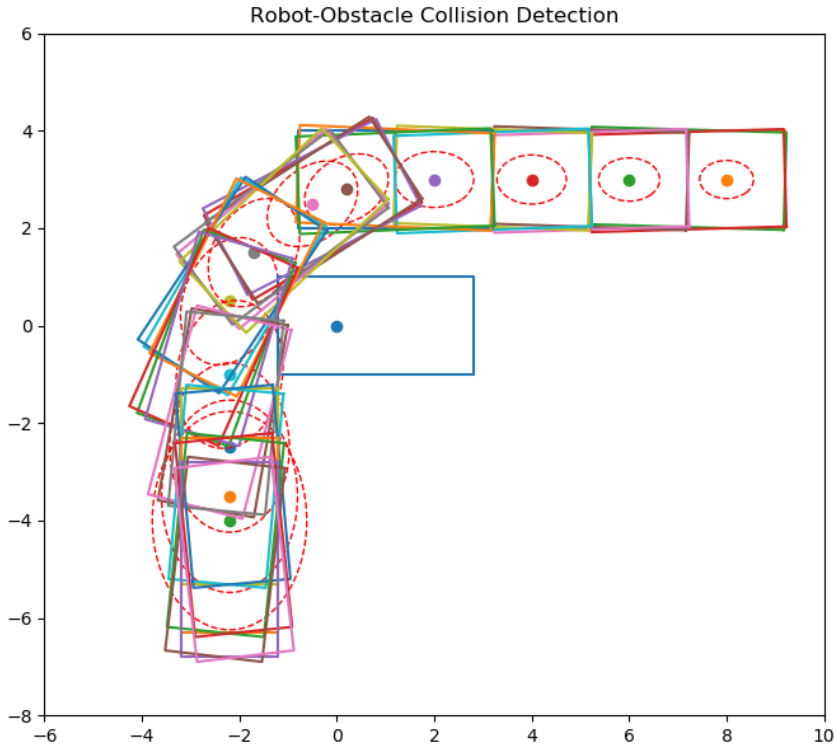

Figure 3: Simulation

the uncertainty during the timesteps. And we can see the equal-probability line is growing as the magnitude of uncertainty is growing. The exact way to realize this growing uncertainty is to be referred within source code `col_prob.py -> line 482`.

Finally, the collision probability is evaluated with three approaches: circular, Monte Carlo and a elaborated probability upper bound described in the article (Figure 4).

The highest bound is obtained without any prior on the obstacle's orientation. So it overestimate the collision probability. And red line is obtained with method described in the article, and it poses some priors on the orientation of the obstacle, which can be interpreted as the driver's behavior. The blue dash line is obtained by Monte Carlo method: we firstly randomize the center of the obstacle, then the orientation. Therefore, we sample an obstacle $w.r.t$ the normal distribution of the obstacle center and a normal distribution of obstacle's orientation angle. We detect the collision by detection any intersection among edges of two body.

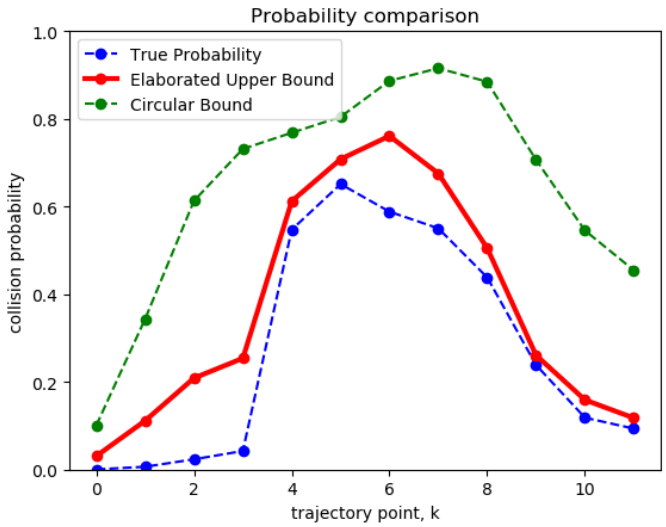

Figure 4: Comparison of three probability

By cutting down the probability bound, we enable the robot to travel in a larger exploration zone, facilitating the path optimization.

2.3 README

To run the code, it's simple. We only need `scipy`, `matplotlib` and `numpy` since the implementation is generally from scratch. To run the code, make sure two scripts are in the same file. Simple enough, go to the directory where two scripts are and please type:

```
python col_prob.py
```

And go to the next step by just closing the pop-up windows.