T*: Time-Optimal Risk-Aware Motion Planning for Curvature-Constrained Vehicles

Junnan Song[1] Shalabh Gupta[1*] Thomas A. Wettergren[2]

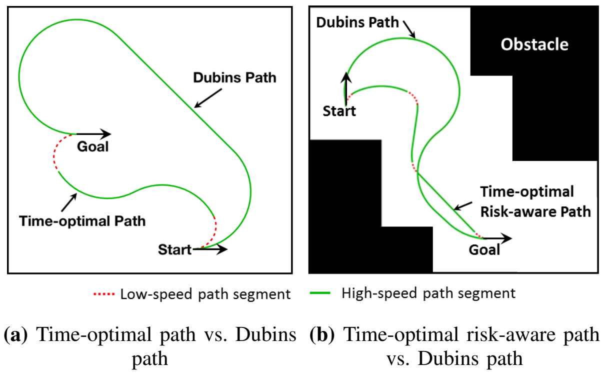
***Abstract*—This paper addresses the time-optimal risk-aware motion planning problem for curvature-constrained variable-speed vehicles in the presence of obstacles. To the best of our knowledge, this problem has not been solved. To obtain a feasible solution, we present a grid-based method, called T*, which computes the time-optimal risk-aware path by finding the optimal sequence of vehicle states from the start to the goal with the minimum total cost of time and risk. We propose a novel risk function based on the concept of collision time, which utilizes the complete information of the vehicle state, including its location with respect to obstacles, heading angle, and speed. We also propose a state pruning technique that can significantly reduce the computational complexity. The algorithm was validated in simulations of complex obstacle-rich scenarios, and the resulting paths are shown to be superior than the Dubins paths.**

***Index Terms*—Motion and Path Planning, Nonholonomic Motion Planning, Robot Safety, Optimization and Optimal Control.**

1. Introduction

A fundamental problem for autonomous vehicles is to plan a collision-free path to reach a goal state in minimal time. Typically, autonomous vehicles are subject to kinematic constraints, such as bounded curvature and bounded turn rate, which limit their manoeuvrability. Along this line, Dubins [1] addressed the problem of finding the shortest path for a vehicle with kinematic constraints, that moves at a constant speed in an obstacle-free space. He used a geometric approach and showed that the shortest path between a pair of vehicle poses must be one of the following 6 types: *RSR*, *RSL*, *LSR*, *LSL*, *RLR* and *LRL*, where $L(R)$ refer to a left (right) turn with maximum curvature, and S indicates a straight line segment. Dubins-like path planning is popular because the resulting path consists of optimized parametric curves that can be expressed analytically and computed quickly, based on which many path-following methods could be designed. Further research on this problem considered bounded acceleration [2], field-of-view constraint [3], the orienteering problem [4][5], polygonal obstacles [6][7], and external disturbances [8][9].

The above methods following the Dubins' approach focused on shortest path planning; however, autonomous vehicles can travel at variable speeds, thus the time-optimal path can be different from the shortest path. In a recent study, Wolek et al. [10] derived the solution to find the time-optimal path for curvature-constrained variable-speed vehicles in an obstacle-free space. They identified a sufficient set of 34 candidate paths, where each candidate path contains circular arcs or straight line segments on which the vehicle travels at extremal (i.e., maximum or minimum) speeds. Then, for a given pair of start and goal poses, the time-optimal path is the least-cost candidate path. Fig. 1a shows an example of time-optimal path vs. Dubins path, where the slow speed segments in the time-optimal path enable smaller turning radii which help in reducing the total path length.



(a) Time-optimal path vs. Dubins path **(b)** Time-optimal risk-aware path vs. Dubins path

Figure 1: Time-optimal and time-optimal risk-aware paths as compared to the Dubins paths in different environment

However, to the best of our knowledge, the time-optimal motion planning problem for curvature-constrained variable-speed vehicles has not been solved in the presence of obstacles. As shown in [6], the problem of deciding whether a curvature-constrained collision-free path exists between two given poses amid polygonal obstacles is NP-hard [7]. This implies that no efficient exact algorithms exist for curvature-constrained time-optimal motion planning in arbitrary environment. In this paper, we present the T* algorithm that provides an approximate solution to solve the above-mentioned problem in the discrete domain [11].

The second issue addressed in this paper is that of vehicle safety. Due to complexities of the environment, it is also critical that the time-optimal path is safe for the vehicle. Pereira et al. [12] proposed a minimum-risk planning strategy using the risk map for autonomous underwater vehicles (AUVs). Hernández et al. [13] presented a safe path planning algorithm for AUVs, where safety is ensured by creating a risk zone around the vehicle for obstacle avoidance. Liu et al. [14] proposed a episodic memory-based planning method, where the risk in the local behavioral planning strategy is measured by the minimum distance to danger areas. Pfeiffer

Manuscript received: June 9, 2018; Revised September 5, 2018; Accepted October 5, 2018. This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by US Office of Naval Research under Award Number N000141613032. Any opinions or findings herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

[1]J. Song and S. Gupta are with Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, 06269, USA. [2]T.A. Wettergren is with Naval Undersea Warfare Center, Newport, RI 02841, USA.
* Corresponding Author (email id: shalabh.gupta@uconn.edu)

Digital Object Identifier (DOI): see top of this page.

et al. [15] assumed *a priori* known threat zones to UAVs and presented an approach to find paths with minimized probability of being exposed to threats. Davoodi [16] proposed a bi-objective optimization approach to find the Pareto-optimal paths with minimized path length and maximized distance to the obstacles. Wang et al. [17] presented a multi-objective Particle Swarm Optimization (PSO) approach for car-like robots, where the objectives are to minimize the path length and the total risk defined by terrain roughness. Huang and Savkin [18] presented the Shortest Viable Path Planning (SVPP) algorithm to address a variant of Dubins Travelling Salesman Problem in presence of obstacles to visit a set of sensor nodes in a sensing field. The above papers addressed safe-path planning; however, the risk measures therein were based on only partial information of the vehicle state, i.e., its location with respect to the obstacles or threats, while ignoring its speed and heading angle. Fraichard and Asama [19] introduced the concept of *inevitable collision states* into safe-path planning, where the states that cannot avoid future collision must be prohibited.

Along this line, we propose a continuous risk function based on the concept of *collision time*, which is the time in which the vehicle can hit the obstacle along its heading direction, if it loses control. Thus, a vehicle is deemed safe if its collision time is greater than the time it takes to stop, maneuver around, or re-gain its control. The concept of collision time considers the complete information about the vehicle state, including its location with respect to the obstacles, heading angle, and speed. Furthermore, this risk function is seamlessly integrated with the time cost for motion planning. Fig. 1b shows an example where the time-optimal risk-aware path stays away from the obstacles as compared to the Dubins path.

After defining the joint optimization function, we construct a discrete configuration space and use a grid-based A*-like search for motion planning. As compared to other grid-based methods under motion constraints such as Hybrid-A*[20], the motion primitives in T* are optimized over both risk and time.

The paper makes the following novel contributions:

- presented a solution to the time-optimal risk-aware motion planning problem in the presence of obstacles,
- integrated a risk function into the time-optimal cost which is based on the concept of collision time, and
- presented an adaptive pruning technique that can significantly reduce the computation time while maintaining the solution quality and ensuring completeness.

The algorithm is validated on simulations of complex obstacle-rich scenarios, where the results show superiority in time savings over the Dubins paths. The algorithm can also produce multiple choices of paths with decreasing risk at the expense of increasing time cost for the planner.

The remainder of this paper is organized as follows. Section 2 formulates the time-optimal risk-aware motion planning problem. Section 3 presents the details of the T* algorithm. Section 4 presents the results on a simulated scenario and the paper is concluded in Section 5.

2. Problem Formulation

Let $\mathcal{A} \subset \mathbb{R}^2$ be the search area which is populated with a number of obstacles. The vehicle motion is described as:

$$\begin{cases} \dot{x}(t) &= v(t)\cos\theta(t) \\ \dot{y}(t) &= v(t)\sin\theta(t) \\ \dot{\theta}(t) &= u(t), \end{cases} \tag{1}$$

where $(x, y, \theta) \in SE(2)$, u is the turning rate, and v is the speed.

It is assumed that the autonomous vehicle is capable of traveling at a variable speed, s.t., $v \in [v_{\min}, 1]$, where $v_{\min} \in \mathbb{R}^+$ is the minimum speed, and with a modified distance unit the maximum speed $v_{\max}$ is normalized to 1 [10]. Note that the Dubins path considered the special case where v is constant.

Also, the turn rate u is symmetric and bounded, s.t., $u \in [-u_{\max}, u_{\max}]$, where $u_{\max} \in \mathbb{R}^+$ is the maximum turn rate and the $+/-$ sign refers to a left/right turn. The turn rate and speed are connected by the curvature $\kappa = u/v$, which is the inverse of the turning radius. The curvature $|\kappa| = 0$ means that the vehicle is moving in a straight line, this happens when $u = 0$. On the other hand, when the vehicle turns at the maximum turn rate (i.e., $\pm u_{\max}$), it can do so at the maximum or minimum speed, which result in the curvature $|\kappa| = u_{\max}$ or $|\kappa| = u_{\max}/v_{\min}$, respectively. These correspond to the maximum and minimum turning radii of the vehicle as $R = 1/u_{\max}$ and $r = v_{\min}/u_{\max}$, respectively. Thus, the curvature is bounded as $|\kappa| \in [0, u_{\max}/v_{\min}]$.

Now, let us denote the state of the vehicle as $\mathbf{p} = (x, y, \theta, v)$. Let Γ denote the set of all collision-free paths between the start state $\mathbf{p}_{start}$ and the goal state $\mathbf{p}_{goal}$. Then, for each path $\gamma \in \Gamma$, the control $\mathbf{c}(s) = (\kappa, v)$ at any point s on γ belongs to the following constraint set [10]:

$$\Omega = \left\{ (\kappa, v) \mid v_{min} \leq v \leq 1 \text{ and } |\kappa| \leq \frac{u_{max}}{v} \right\}. \tag{2}$$

The admissible control must be piece-wise continuous and should satisfy the boundary conditions, i.e., $\mathbf{c}$ must drive the vehicle from $\mathbf{p}_{start}$ to $\mathbf{p}_{goal}$ along the path γ while avoiding obstacles. Further, let $\mathcal{R}(s)$ denote the risk cost at a point s on γ. Then, the total cost of time and risk is defined as

$$J(\gamma) = \int_\gamma \mathcal{R}(s) \cdot \frac{1}{v(s)}\, ds, \tag{3}$$

where the term $\frac{1}{v(s)}$ evaluates the time cost when the vehicle moves along a small path segment ds.

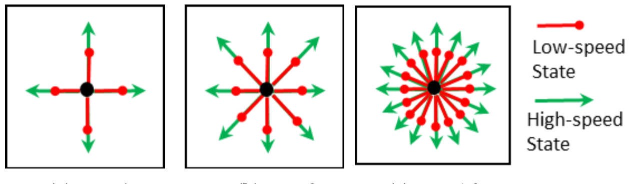
Therefore, the objective is to find the control $\mathbf{c}^\star \in \Omega$, which generates a collision-free path $\gamma^\star$ with the minimal cost $J(\gamma^\star)$, s.t. $J(\gamma^\star) \leq J(\gamma), \forall \gamma \in \Gamma$. Note that the outcome of optimization is a trajectory; however, we use the term path for trajectory in this paper with slight abuse of terminology.

3. T* Algorithm

Since there is no efficient exact algorithm to minimize (3) in the presence of obstacles, even when the risk is ignored [6], this section develops a novel grid-based algorithm, called T*, to obtain an approximate solution in the discrete domain.

A. Configuration Space

Since any feasible path γ between $\mathbf{p}_{start}$ and $\mathbf{p}_{goal}$ consists of multiple intermediate states, the time-optimal risk-aware motion planning problem can be solved by identifying the



(a) $L=4$ **(b)** $L=8$ **(c)** $L=16$

Figure 2: State expansion in each cell when $L=4$, 8 and 16, respectively

optimal sequence of states. This in turn motivates to partition the search area into grid cells and then assign each cell with a sufficient number of possible discrete states, which forms a configuration space described as below.

First, the search area $\mathcal{A}$ is discretized into a set of grid cells, $\mathcal{C}=\{c_\alpha \subset \mathbb{R}^2, \alpha=1,\dots,|\mathcal{C}|\}$, where $\bigcup_{\alpha=1}^{|\mathcal{C}|} c_\alpha = \mathcal{A}$, $c_\alpha^\circ \bigcap c_\beta^\circ = \emptyset$, $\forall \alpha,\beta \in \{1,\dots,|\mathcal{C}|\}$, $\alpha \neq \beta$, and $^\circ$ denotes the interior.

Then, each $c_\alpha \in \mathcal{C}$ is encoded with a symbolic state $s_\alpha \in \{O,F\}$, where $O \equiv obstacle$ and $F \equiv free$. Specifically, $s_\alpha = O$ if c_α is (partially) occupied by an obstacle; otherwise, $s_\alpha = F$. This generates the free space $\mathcal{C}_{free} = \{c_\alpha \in \mathcal{C} : s_\alpha = F\}$.

Then the configuration space $\mathcal{Q}$ is constructed as follows.

Definition 3.1 (**Configuration Space**). *Let $\mathcal{O}=\{(x_\alpha,y_\alpha)\in c_\alpha : c_\alpha \in \mathcal{C}_{free}\}$ be the set of center positions of all obstacle-free cells. Let $\Theta=\{\frac{2\pi l}{L}: l=0,\dots,L-1\}$ be the set of $L\in\mathbb{N}^+$ heading angles. Let $\mathcal{V}=\{v_{\min},1\}$ be the set of speeds. Then, the configuration space is defined as:*

$$\mathcal{Q}=\mathcal{O}\times\Theta\times\mathcal{V}.$$

Fig. 2 shows the state expansion for different L. Clearly, the size of $\mathcal{Q}$ relies on the size of $\mathcal{C}_{free}$ and the value of L. Although a larger L could potentially produce better results, it will also increase the size of $\mathcal{Q}$ that leads to higher computational complexity for motion planning. This paper adopts the *8-orientation* state expansion with $L=8$.

B. Approximate Optimization Function

Now, we present an approximation to the optimization problem in Section 2. Let $\mathcal{P}=\{\mathbf{P}^m, m=1,\dots,|\mathcal{P}|\}$ be the set of all valid state sequences, where $\mathbf{P}^m=[\mathbf{p}_1^m,\mathbf{p}_2^m,\dots,\mathbf{p}_n^m]$ is a sequence of states that connects $\mathbf{p}_{start}$ and $\mathbf{p}_{goal}$. Here $\mathbf{p}_i^m=(x_i,y_i,\theta_i,v_i)\in\mathcal{Q}$, $\forall i\in\{1,\dots,n\}$, and $n\in\mathbb{N}^+$ denotes the length of the sequence. To satisfy the boundary conditions, it should have $\mathbf{p}_1^m=\mathbf{p}_{start}$, and $\mathbf{p}_n^m=\mathbf{p}_{goal}$. Note that any two consecutive states $\mathbf{p}_i^m$ and $\mathbf{p}_{i+1}^m$, $\forall i\in\{1,\dots,n-1\}$, must belong to two neighboring cells. Also, a valid state sequence cannot contain duplicate states, i.e., $\mathbf{p}_i^m\neq\mathbf{p}_j^m, \forall i,j\in\{1,\dots,n\}, i\neq j$; otherwise a loop will be present, thus it will not be optimal since its cost can be further reduced by removing such loop.

Then, we compute the total cost of a state sequence $\mathbf{P}^m$ in a piece-wise manner as follows:

$$J(\mathbf{P}^m)\triangleq\sum_{i=1}^{n-1}\widetilde{J}(\mathbf{p}_i^m,\mathbf{p}_{i+1}^m), \tag{4}$$

where $\widetilde{J}$ is the step-wise cost to move from $\mathbf{p}_i^m$ to $\mathbf{p}_{i+1}^m$, subject to the vehicle motion constraints and the obstacle layout. Let Γ^c be the set of all feasible paths between $\mathbf{p}_i^m$ and $\mathbf{p}_{i+1}^m$, then $\widetilde{J}$ is computed as the minimal cost among all paths $\gamma_{i,i+1}\in\Gamma^c$:

$$\widetilde{J}(\mathbf{p}_i^m,\mathbf{p}_{i+1}^m)=\min_{\gamma_{i,i+1}\in\Gamma^c} J(\gamma_{i,i+1}), \tag{5}$$

where according to (3),

$$J(\gamma_{i,i+1})=\int_{\gamma_{i,i+1}}\mathcal{R}(s)\cdot\frac{1}{v(s)}\,ds. \tag{6}$$

For simplicity, the risk is assumed constant along $\gamma_{i,i+1}$ and is measured at the most dangerous state on $\gamma_{i,i+1}$ which has the least collision time (Details are in Section 3-D). Thus, the risk cost and the time cost can be separated, as follows:

$$J(\gamma_{i,i+1})=\underbrace{\mathcal{R}(\gamma_{i,i+1})}_{\text{risk cost}}\cdot\underbrace{\int_{\gamma_{i,i+1}}\frac{1}{v(s)}\,ds}_{\text{time cost }\mathcal{T}(\gamma_{i,i+1})}. \tag{7}$$

We aim to find the optimal state sequence $\mathbf{P}^\star\in\mathcal{P}$, s.t. $J(\mathbf{P}^\star)\le J(\mathbf{P}^m), \forall\mathbf{P}^m\in\mathcal{P}$. Then, the optimal path $\gamma^\star$ can be reconstructed in a piece-wise manner using the states of $\mathbf{P}^\star$ and the optimal paths between adjacent states. While the solution to the optimization problem is presented in Section 3-F, we first present the details on computation of the time cost $\mathcal{T}(\gamma_{i,i+1})$ and the risk cost $\mathcal{R}(\gamma_{i,i+1})$.

C. Time Cost $\mathcal{T}(\gamma_{i,i+1})$

Now, let us define the time cost $\mathcal{T}(\gamma_{i,i+1})$ in (7). As compared to the Dubins curves, since the vehicle can travel at variable speeds, the time-optimal path that minimizes $\mathcal{T}(\gamma_{i,i+1})$ can contain arcs of different turning radii (see Fig. 1), which implies more choices of candidate paths, or in other words, a larger candidate set than the Dubins curves.

*1) **The Sufficient Set of Candidate Paths**:* For any pair of states $\mathbf{p}_i^m$ and $\mathbf{p}_{i+1}^m$, the authors of [10] showed that the sufficient set, which guarantees to contain the time-optimal path in the absence of obstacles, consists of 34 candidate paths. These candidate paths form the set Γ^c, as shown in Table I. Each path $\gamma_{i,i+1}\in\Gamma^c$ consists of up to five segments, where each segment could be one of the following [10]:

1) Bang arcs (B), where the vehicle turns at maximum speed with maximum turn rate (i.e. with radius R);
2) Cornering arcs (C), where the vehicle turns at minimum speed with maximum turn rate (i.e., with radius r);
3) Straight line segments (S), where the vehicle moves straight with the maximum speed.

Any circular arc (i.e., B or C) can be either left (L) or right (R), as shown under the direction column. Each candidate path is read from left to right, and the consecutive turns within parentheses are of the same direction. For example, the No.18 path of $(B)S(BC)$ with direction LSR is read as: a left bang arc, followed by a straight line segment, a right bang arc, and then a right cornering arc.

Each candidate path has to be optimized for its parameters (i.e., the angles for arc segments and the lengths of straight line segments) to achieve time optimality, thus one must solve a nonlinear constrained optimization problem [10]. Specifically, the total time cost for each candidate path between a pair of states is the summation of cost for each segment. An arc segment (B or C) with angle $\Delta\theta$ contributes to the time cost

Table I: Γ^c: The set of candidate paths between any pair of states

No.	Path Type[1]	Direction[2]	No.	Path Type	Direction
1	(B)S(B)	LSL	18	(B)S(BC)	LSR
2	(B)S(B)	LSR	19	(B)S(BC)	RSL
3	(B)S(B)	RSL	20	(B)S(BC)	RSR
4	(B)S(B)	RSR	21	(CB)(BCB)	LL
5	(BCB)(B)	LL	22	(CB)(BCB)	LR
6	(BCB)(B)	LR	23	(CB)(BCB)	RL
7	(BCB)(B)	RL	24	(CB)(BCB)	RR
8	(BCB)(B)	RR	25	(CB)S(B)	LSL
9	(B)(BCB)	LL	26	(CB)S(B)	LSR
10	(B)(BCB)	LR	27	(CB)S(B)	RSL
11	(B)(BCB)	RL	28	(CB)S(B)	RSR
12	(B)(BCB)	RR	29	(C)(C)(C)	LRL
13	(BCB)(BC)	LL	30	(C)(C)(C)	RLR
14	(BCB)(BC)	LR	31	(CB)S(BC)	LSL
15	(BCB)(BC)	RL	32	(CB)S(BC)	LSR
16	(BCB)(BC)	RR	33	(CB)S(BC)	RSL
17	(B)S(BC)	LSL	34	(CB)S(BC)	RSR

[1] B is a bang arc, C is a cornering arc and S is a straight line segment. The parentheses are used to indicate consecutive turns of the same direction.

[2] L is a left turn, R is a right turn and S means move straight.

by $\frac{|\Delta\theta|}{u_{max}}$; while a displacement of $d \in \mathbb{R}^+$ for the straight line segment contributes to the cost by d while moving at $v_{\max} = 1$.

Since each candidate path is required to exactly reach $\mathbf{p}^m_{i+1}$ from $\mathbf{p}^m_i$, there are five constraints including the total displacement along each axis, the total change in the heading angle, and the speeds specified by the first and last arc types. These boundary conditions ensure the continuity in position, heading and speed when the vehicle reaches every state $\mathbf{p}^m_i \in \mathbf{P}^m$, including $\mathbf{p}_{start}$ and $\mathbf{p}_{goal}$.

In this paper, the path parameters for each candidate path are optimized using the nonlinear solver *IPOPT* (*I*nterior *P*oint *OPT*imizer) [10]. The time-optimal path between a given pair of states is the candidate path with the least time cost.

Next, we construct the Optimized Candidate Paths for State-pairs (OCPS) table, that contains the optimized candidate paths for all possible pairs of states $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$.

*2) **The OCPS Table:*** To avoid computational burden during motion planning, the parameters of the candidate paths are optimized offline to construct the OCPS Table.

Specifically, consider a state $\mathbf{p}^m_i$ located in a center cell and all the possible states $\mathbf{p}^m_{i+1}$ located in its 3×3 neighborhood. Since $\mathbf{p}^m_i$ has $8 \times 2 = 16$ choices corresponding to 8 directions and 2 speeds, and $\mathbf{p}^m_{i+1}$ has $8 \times 8 \times 2 = 128$ choices corresponding to 8 positions in the neighborhood, 8 directions, and 2 speeds, the total number of 2048 pairs are considered. However, by exploring symmetry, one can easily figure out that only 272 pairs must be optimized for, while the rest can be derived accordingly.

For each pair of $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$, depending on the speed information of $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$, the associated candidate paths belong to one of the following four subsets of Γ^c:

- Γ^c_{BB}, which contains No.1-12 paths that start and end with $v_{\max}$, i.e., they start and end with bang arcs B;
- Γ^c_{BC}, which contains No.13-20 paths that start with $v_{\max}$ and end with $v_{\min}$, i.e., they start with a bang arc B and end with a cornering arc C;
- Γ^c_{CB}, which contains No.21-28 paths that start with $v_{\min}$ and end with $v_{\max}$, i.e., they start with a cornering arc C and end with a bang arc B; and
- Γ^c_{CC}, which contains No.29-34 paths that start and end with $v_{\min}$, i.e., they start and end with cornering arcs C.

Thus, for each pair of $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$, the path parameters of candidate paths within the corresponding subset are optimized offline using *IPOPT*. The resulting optimized candidate paths and their time costs are stored in the OCPS table.

*3) **Computation of** $\mathcal{T}(\gamma_{i,i+1})$:* During the planning process, when computing for the time cost $\mathcal{T}$ between a certain state pair $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$, the planner can initiate a query to the OCPS table to obtain a set of optimized candidate paths within the corresponding subset determined by the speed information of $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$. Then for each obtained optimized candidate path $\gamma_{i,i+1}$, $\mathcal{T}(\gamma_{i,i+1})$ is assigned with the associated time cost if it is collision-free; otherwise, $\mathcal{T}(\gamma_{i,i+1}) = +\infty$.

D. Risk Cost $\mathcal{R}(\gamma_{i,i+1})$

This section presents a state-based risk function to evaluate the risk cost of each candidate path $\gamma_{i,i+1} \in \Gamma^c$.

As mentioned in Section 3-B, the risk cost $\mathcal{R}(\gamma_{i,i+1})$ of a candidate path $\gamma_{i,i+1}$ between a pair of states $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$ is determined by the most dangerous state along $\gamma_{i,i+1}$ that results in the least collision time to an obstacle or the space boundary in its heading direction, if the vehicle loses control. For any state, the vehicle is considered safe if the corresponding collision time is greater than a threshold $t^\star \in \mathbb{R}^+$, which indicates the time for the vehicle to fully stop, maneuver around, or re-gain its control. Thus, the risk of the vehicle relies on its state, including its location, heading and speed. Now, we present the computation of $\mathcal{R}(\gamma_{i,i+1})$.

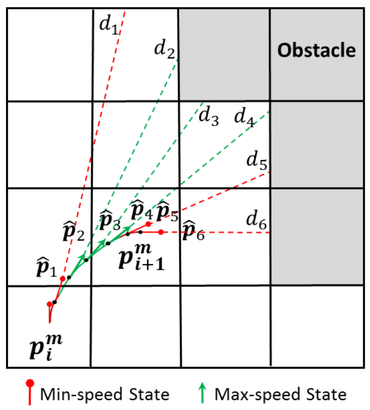
Consider a candidate path $\gamma_{i,i+1} \in \Gamma^c$. First, a set of $M \in \mathbb{N}^+$ states, $\{\hat{\mathbf{p}}_\ell : \ell = 1, \ldots, M\}$, are evenly sampled along $\gamma_{i,i+1}$ with a sampling interval $\Delta d \in \mathbb{R}^+$, s.t. $\hat{\mathbf{p}}_M = \mathbf{p}^m_{i+1}$. Specifically, $\hat{\mathbf{p}}_\ell = (x_\ell, y_\ell, \theta_\ell, v_\ell)$, where $(x_\ell, y_\ell, \theta_\ell) \in SE(2)$ and $v_\ell \in \{v_{\min}, 1\}$ indicates the speed of the sampled state. Fig. 3a shows an example of sampled states between $\mathbf{p}^m_i$ and $\mathbf{p}^m_{i+1}$ when $M = 6$.

Then, for each $\hat{\mathbf{p}}_\ell$, $\ell \in \{1, \ldots, M\}$, one can geometrically compute the collision distance d_ℓ, as shown in Fig. 3b. Denote by *line* ℓ the extended line of state $\hat{\mathbf{p}}_\ell$ along angle θ_ℓ, which hits the obstacle at the hit point $H = (x_h, y_h) \in \mathbb{R}^2$. Then, $d_\ell = \|(x_\ell - x_h, y_\ell - y_h)\|$. Once d_ℓ is determined, the corresponding collision time t_ℓ is computed as:

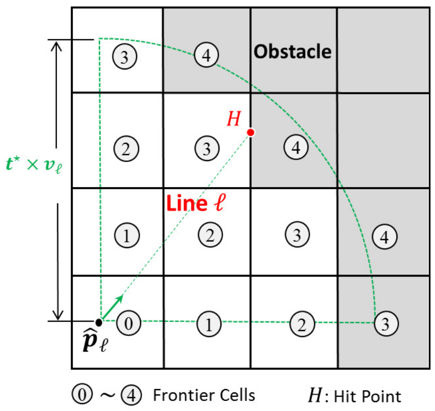
$$t_\ell = \frac{d_\ell}{v_\ell}. \tag{8}$$

Algorithm 1 computes t_ℓ, by sequentially exploring the frontier cells in the quadrant determined by θ_ℓ, until *line* ℓ intersects a frontier cell that either contains an obstacle or lies on the space boundary. The frontier cells are labeled with incrementing numbers during exploration, i.e., the numbers ⓪ $\sim$ ④ in Fig. 3b. Note that *line* ℓ cannot intersect a higher numbered frontier cell before intersecting a lower numbered frontier cell. Also, for all frontier cells that are labeled with the same number, *line* ℓ can only intersect one of them.

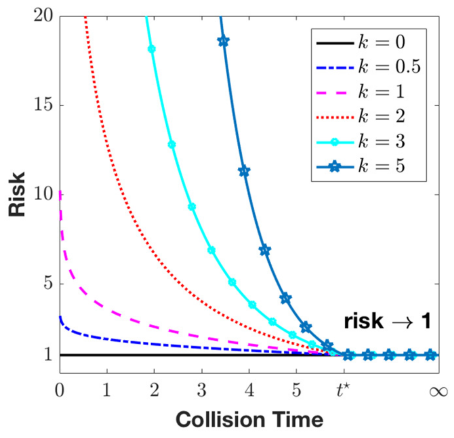
Algorithm 1 uses a queue q to record the frontier cells discovered during exploration. The queue is initialized with


Obstacle
d_1 d_2 d_3 d_4 d_5 d_6
$\widehat{\mathbf{p}}_1$ $\widehat{\mathbf{p}}_2$ $\widehat{\mathbf{p}}_3$ $\widehat{\mathbf{p}}_4$ $\widehat{\mathbf{p}}_5$ $\widehat{\mathbf{p}}_6$
$\mathbf{p}_{i+1}^m$
$\mathbf{p}_i^m$
Min-speed State
Max-speed State

(a) Sampling of $\gamma_{i,i+1}$



(b) Computation of Collision Distance d_ℓ



(c) The Risk Function $(risk(\hat{\mathbf{p}}_\ell))^k$

Figure 3: The collision distance and the risk function

```
Algorithm 1: Computation of Collision Time $t_\ell$
   input : $\hat{\mathbf{p}}_\ell = (x_\ell, y_\ell, \theta_\ell, v_\ell)$, $t^\star$
   output: $t_\ell$
 1 Initialize: queue $q \leftarrow$ cell that contains $\hat{\mathbf{p}}_\ell$
 2 while $q$ is not empty do
 3     Update $len \leftarrow$ current size of $q$;
 4     for $i \leftarrow 1$ to $len$ do
 5         Remove the front cell from $q$ and make it $c_\lambda$;
 6         if $s_\lambda = O$ or $c_\lambda$ contains the space boundary then
 7             if line $\ell$ intersects $c_\lambda$ at $(x_h, y_h) \in \mathbb{R}^2$ then
 8                 Compute $d_\ell \leftarrow \|(x_\ell - x_h, y_\ell - y_h)\|$;
 9                 Return $t_\ell \leftarrow \frac{d_\ell}{v_\ell}$;
10             end
11         end
           // Enqueue the new frontier cells
12         Push into $q$ the neighbor cells of $c_\lambda$ that are
             located within a distance of $d_\ell^\star = t^\star \times v_\ell$ from $\hat{\mathbf{p}}_\ell$;
13     end
14 end
15 Return $t_\ell \leftarrow t^\star$   // If $t_\ell$ cannot be found from the above steps
```

the cell that contains the state $\hat{\mathbf{p}}_\ell$ and it is labeled with ⓪ (**Line** 1). As long as the queue is non-empty (**Line** 2), the variable *len* is updated with the size of the queue (**Line** 3). Then, within the for loop, these frontier cells are sequentially extracted from the front of the queue (**Line** 4), and recorded in a variable c_λ at each iteration (**Line** 5). Thereafter, it checks whether: (i) c_λ is occupied by any obstacle (i.e., $s_\lambda = O$) and *line* ℓ intersects with any edge of c_λ; or (ii) c_λ lies on the space boundary and *line* ℓ intersects with its boundary edge (**Line** $6-7$). If either is true, then the collision distance d_ℓ and the collision time t_ℓ are computed based on the hit point $(x_h, y_h) \in \mathbb{R}^2$(**Line** $8-9$).

If t_ℓ is not found from the above steps, then new frontier cells (i.e., the neighbor cells of c_λ not in q) are enqueued for further computations and they are labeled with a number incremented by 1 (**Line** 12). Note that the new frontier cells are determined by θ_ℓ, e.g., as shown in Fig. 3b, θ_ℓ resides in the first quadrant, hence the new frontier cells are the neighbor cells located in the north and east directions of c_λ.

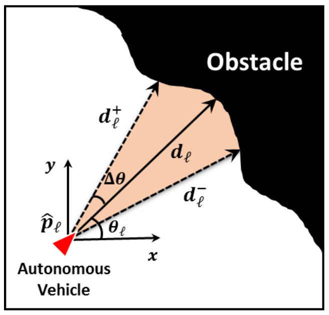


Figure 4: Uncertainties in heading angle θ_ℓ

Since the vehicle is safe at $\hat{\mathbf{p}}_\ell$ when $t_\ell \geq t^\star$, one only needs to search the frontier cells located within a distance of $d_\ell^\star = t^\star \times v_\ell$ from $\hat{\mathbf{p}}_\ell$. A frontier cell is said to be within a distance of $d_\ell^\star$ from $\hat{\mathbf{p}}_\ell$, if the distance between any of its four vertices and the point (x_ℓ, y_ℓ) is smaller than $d_\ell^\star$.

If no t_ℓ is returned and the queue becomes empty, then it implies that *line* ℓ does not intersect any obstacle cell or the space boundary up to distance $d_\ell^\star$. Then, $t_\ell = t^\star$, i.e., state $\hat{\mathbf{p}}_\ell$ is risk-free. (**Line** 15).

Based on the collision time t_ℓ, the risk at $\hat{\mathbf{p}}_\ell$ is defined as

$$risk(\hat{\mathbf{p}}_\ell) = \begin{cases} 1 + \log\left(\frac{t^\star}{t_\ell}\right) & \text{if } t_\ell < t^\star \\ 1 & \text{if } t_\ell \geq t^\star. \end{cases} \tag{9}$$

As $t_\ell \to 0$, the risk approaches ∞; while for $t_\ell \geq t^\star$, the risk reduces to 1, i.e., no risk penalty is added to the time cost.

The risk cost for a candidate path $\gamma_{i,i+1}$ is computed as

$$\mathcal{R}(\gamma_{i,i+1}) = \max_{\ell \in \{1,\ldots,M\}} \Big(risk(\hat{\mathbf{p}}_\ell)\Big)^k, \tag{10}$$

where $k \geq 0$ is the weight parameter. Fig. 3c shows the curves of $(risk(\hat{\mathbf{p}}_\ell))^k$ for different k when $t^\star = 6$. It is seen that for a fixed $t_\ell \in (0, t^\star)$, a larger k produces a higher risk. In particular, when $k = 0$, $\mathcal{R}(\gamma_{i,i+1}) = 1$, thus the total cost $\widetilde{J}(\mathbf{p}_i^m, \mathbf{p}_{i+1}^m) = \mathcal{T}(\gamma_{i,i+1})$, hence the resulting path is time-optimal. Note that $\mathcal{R}(\gamma_{i,i+1}) \in [1, \infty)$, because $risk(\hat{\mathbf{p}}_\ell) \in [1, \infty)$, $\forall \ell$.

Planning under Uncertainties: During plan execution, the vehicle may have uncertainties in its heading and position. Thus, an uncertainty of $\Delta\theta$ is added to θ_ℓ during planning, and

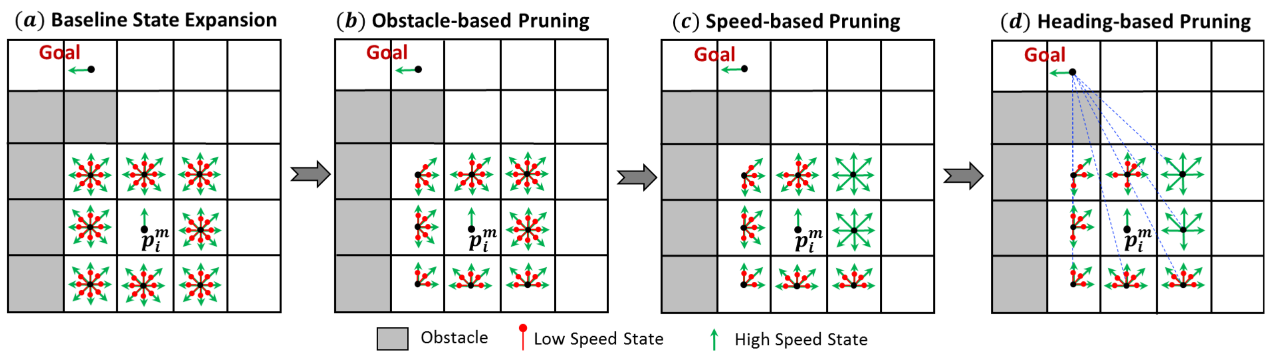

(a) Baseline State Expansion
(b) Obstacle-based Pruning
(c) Speed-based Pruning
(d) Heading-based Pruning
Goal
p_i^m
Obstacle
Low Speed State
High Speed State

Figure 5: Illustration of the adaptive state pruning within each cell

the collision distances d_ℓ^+ and d_ℓ^- corresponding to headings of $\theta+\Delta\theta$ and $\theta-\Delta\theta$ are computed, as shown in Fig. 4. Then t_ℓ is computed based on $\min\{d_\ell, d_\ell^+, d_\ell^-\}$. Since a modestly priced compass has an accuracy of 1^o [21], $\Delta\theta$ is chosen as 1.5^o. Also, a buffer of size 0.1m is added around all obstacles to compensate position uncertainties, such that any path $\gamma_{i,i+1}$ which intersects with the buffer has $\mathcal{R}(\gamma_{i,i+1}) = +\infty$.

E. Adaptive State Pruning for Complexity Reduction

This section presents a three-step adaptive state pruning technique for complexity reduction of $\mathcal{Q}$, as shown in Fig. 5. Consider a state $\mathbf{p}_i^m$ and the baseline 8-*orientation* 2-*speed* state expansion in its neighbor cells, as shown in Fig. 5a.

1) *Obstacle-based Pruning*: The states close to and facing obstacles or boundaries are considered as inevitable collision states, thus they are pruned, as shown in Fig. 5b.
2) *Speed-based Pruning*: In open regions away from obstacles, the vehicle is expected to travel at the highest speed to minimize the time cost; while the low-speed states are typically useful near obstacles to allow turning with a smaller turning radius for better controllability. Therefore, the low-speed states in the cells located far from obstacles can be pruned, as shown in Fig. 5c.
3) *Heading-based Pruning*: Since the states with a diagonal heading and in an opposite direction to the goal will very likely produce higher costs, they are less likely to be part of the optimal state sequence. Thus, they can be dynamically identified and removed from $\mathcal{Q}$. Note that the states with non-diagonal heading angles of 0, $\pi/2$, π and $3\pi/2$ must be retained to ensure the completeness of the algorithm. In this regard, first connect the centers of each cell and $\mathbf{p}_{goal}$ using a straight line as shown in Fig. 5d. Then, for each state with a diagonal heading, compute the angle $\xi \in (-\pi, \pi]$ formed with the corresponding line. If $|\xi| > \eta$, where $\eta \in (0, \pi]$ is a pre-defined threshold, then such state is pruned. Fig. 5d shows an example when $\eta = \pi/2$.

F. Searching for the Time-optimal Risk-aware Path

The step-wise cost $\widetilde{J}(\mathbf{p}_i^m, \mathbf{p}_{i+1}^m)$ between any two consecutive states $\mathbf{p}_i^m$ and $\mathbf{p}_{i+1}^m$ is determined by the least product cost of $\mathcal{T}(\gamma_{i,i+1})$ and $\mathcal{R}(\gamma_{i,i+1})$ among all corresponding optimized candidate paths in the OCPS table. Now, for an intermediate state $\mathbf{p}_i^m$, we define

$$f(\mathbf{p}_i^m) = g(\mathbf{p}_{start}, \mathbf{p}_i^m) + h(\mathbf{p}_i^m, \mathbf{p}_{goal}), \tag{11}$$

where the cumulative cost function g is

$$g(\mathbf{p}_{start}, \mathbf{p}_i^m) = \sum_{j=1}^{i-1} \widetilde{J}(\mathbf{p}_j^m, \mathbf{p}_{j+1}^m), \tag{12}$$

and the heuristic cost $h(\mathbf{p}_i^m, \mathbf{p}_{goal})$ is determined by the length of the shortest Dubins path using turning radius r divided by the maximum speed $v_{\max}$. Such heuristic is admissible thus it guarantees the optimality of $\mathbf{P}^\star$. Thereafter, we adopt the framework of $A^\star$ algorithm to search for $\mathbf{P}^\star$, where the states are gradually explored and assigned with the cost using (11). The search process repeats until the goal state is found. Then, the time-optimal risk-aware path $\gamma^\star$ can be reconstructed in a piece-wise manner using the states of $\mathbf{P}^\star$ and the optimal paths between the adjacent states of $\mathbf{P}^\star$.

4. Results and Discussion

The T* algorithm has been validated in complex obstacle-rich scenarios. The results were compared with Dubins paths, the effect of risk weight k in (10) was investigated, and the efficiency of the adaptive state pruning technique in reducing the computational complexity was quantitatively examined.

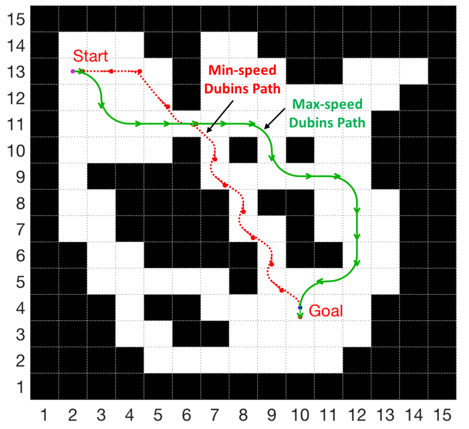
The autonomous vehicle is subject to the following kinematic constraints: maximum turn rate $u_{\max} = 0.5$rad/s, and speeds $v_{\min} = 0.5$m/s and $v_{\max} = 1$m/s. Hence, its turning radii are $R = 2$m and $r = 1$m. Also, $t^\star$ is chosen as 6s.

The search area $\mathcal{A}$ of size 30m $\times$ 30m is partitioned into a set $\mathcal{C}$ consisting of 15×15 cells, where each cell is of size 2m $\times$ 2m. The adaptive state pruning method was applied with the threshold $\eta = \pi/2$.

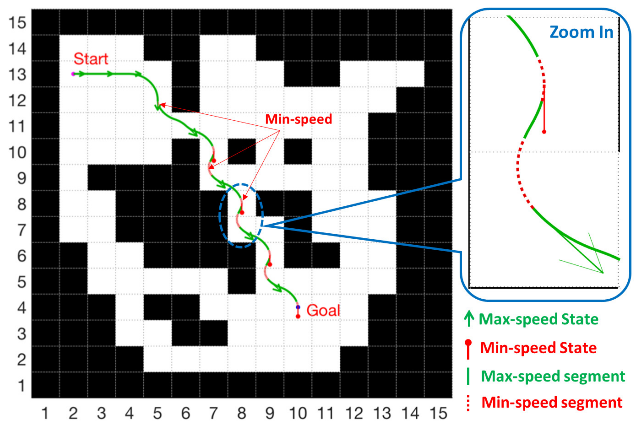
A. Comparison of Time-optimal and Dubins Approaches

First, we compare the time-optimal path with the Dubins paths. The time-optimal path was obtained by using $k = 0$ in (10), i.e., the effect of risk was not considered. On the other hand, since a Dubins vehicle must move at a constant speed, we generated two such paths one at the maximum and the other at the minimum speed. The start and goal states were chosen as $\mathbf{p}_{start} = (4\text{m}, 26\text{m}, 0, v_{\max})$ and $\mathbf{p}_{goal} = (20\text{m}, 8\text{m}, 3\pi/2, v_{\min})$.

Fig. 6a shows the two Dubins paths, while Fig. 6b shows the time-optimal path. Since the Dubins path with $v_{\max}$ results in large turning radius R, its movement through the shortcut taken by the time-optimal path is restricted. Thus, it takes a



(a) Dubins paths. Total time cost using $v_{\max}$: **35.99s**. Total time cost using $v_{\min}$: **55.95s**



(b) Time-optimal path (for $k=0$). Total time cost: **34.51s**

Figure 6: The Dubins paths vs. time-optimal path in an obstacle-rich environment

Table II: Total costs and computation times using adaptive state pruning

Risk Weight	*State Pruning Threshold*	*Total Cost* $J(\mathbf{P}^\star)$	*Computation Time*	*Saving in Computation Time*
$k=0$	None	33.31	259.92s	–
	$\eta=\pi$	34.51	73.31s	71.80%
	$\eta=\pi/2$	34.51	54.29s	79.11%
	$\eta=\pi/4$	34.51	38.06s	85.36%
$k=0.3$	None	38.46	709.56s	–
	$\eta=\pi$	38.87	227.90s	67.88%
	$\eta=\pi/2$	38.87	169.97s	76.05%
	$\eta=\pi/4$	39.52	124.78s	82.41%
$k=3$	None	62.98	569.43s	–
	$\eta=\pi$	62.98	157.95s	72.26%
	$\eta=\pi/2$	62.98	116.08s	79.61%
	$\eta=\pi/4$	71.94	96.56s	83.04%

longer path with a total time cost of 35.99s. On the other hand, the Dubins path with $v_{\min}$ has better controllability with the turning radius r; thus, it produces the minimum-length path through the shortcut. However, it takes 55.95s which is much higher due to the minimum speed.

In comparison, the time-optimal path shown in Fig. 6b is composed of segments with different speeds. This enables the vehicle to travel at $v_{\max}$ in relatively open regions to reduce the total time cost, while subject to a larger turning radius R. In congested regions, it tends to decrease its speed to $v_{\min}$ to gain better maneuverability with a smaller turning radius r. The total time cost of the time-optimal path is 34.51s, which is lower than both the Dubins paths described above.

B. Time-optimal Risk-aware Paths for Different k

This section examines the effect of k in (10) on the motion planning. A higher k is expected to produce a safer path, however, the time cost would be potentially higher.

Fig. 7a to Fig. 7c show the time-optimal risk-aware paths for $k=0$, 0.3 and 3, respectively. These paths are color-coded based on the speed information. Fig. 7d to Fig. 7f show the same paths but they are color-coded based on the risk information. The risk of a state along the path was evaluated using (9), where the sample states were generated using the sampling interval of $\Delta d = 0.4$m.

It is seen that, for $k=0$, the time-optimal path reaches the goal in 34.51s through the shortcut in congested regions. However, it has multiple dangerous segments as shown in Fig. 7d by the risk color-coding. The max risk of the whole path was 2.45. For $k=0.3$, the time-optimal risk-aware path selects a less-risky route to avoid the congested region, but requires a higher time cost of 36.30s. Accordingly, the number of risky segments was significantly reduced; however, the segments in cells $(9,9)$ and $(9,11)$ still present high risk. Further, for $k=3$, the resulting path picked the safest route, where the max risk reduced to 1.48, but the corresponding time cost reaches 41.89s.

In addition, a smoothing operation can help remove zigzag-shaped segments, e.g., near cell $(8,7)$ in Fig. 7a; and further reduce $J(\mathbf{P}^\star)$. Here, we re-optimize over randomly sampled state pairs along the optimal path and replace with lower-cost collision-free segments if they exist. This operation was repeated for four iterations, and the smoothed paths and their risk color-codings are shown in Fig. 7a$\sim$ 7f, respectively.

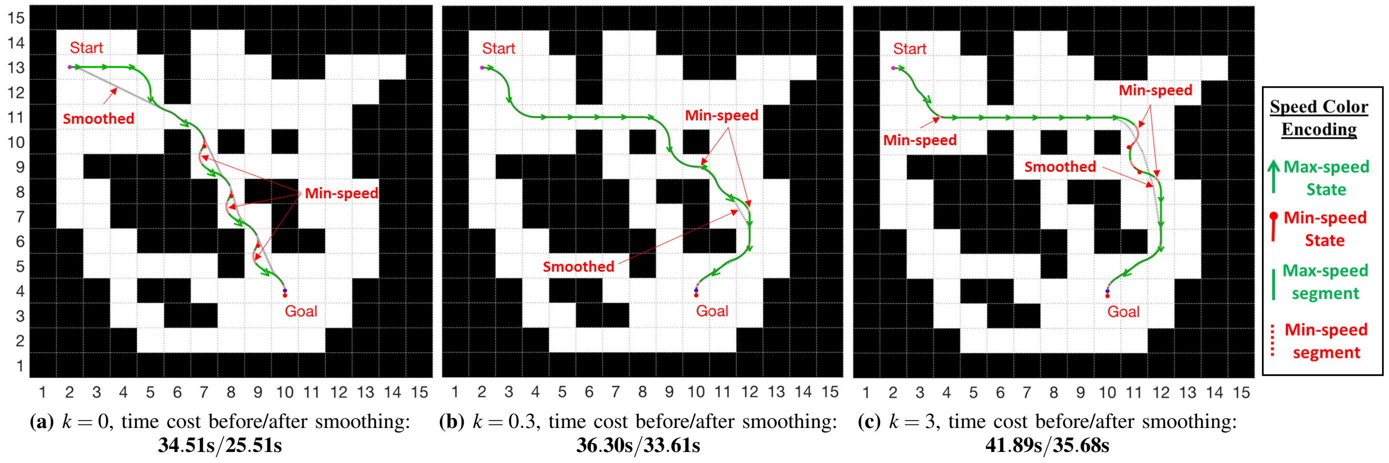
C. Complexity Reduction by Adaptive State Pruning

This section examines the efficiency of the adaptive state pruning technique in reducing the computational complexity. Table II summarizes the total cost $J(\mathbf{P}^\star)$, the average computation time over 5 runs, and the savings in computation time as compared to no pruning, for varying η. The results are presented for three values of $k=0$, 0.3 and 3. The results are generated on a computer with 3.4GHz CPU and 16GB RAM.

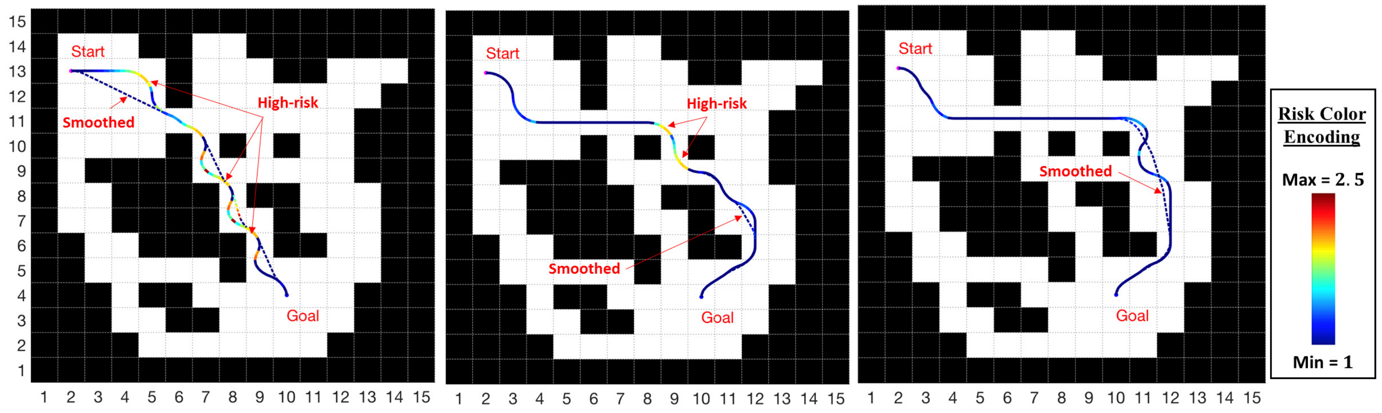
Note that a higher threshold of η would retain more diagonally facing states during the heading-based pruning. The results in Table II show that the adaptive state pruning technique can significantly reduce the computation time. When η is reduced from π to $\pi/4$, the computational cost reduces; however, the total optimization cost remains more or less the same, thus revealing the effectiveness of the pruning approach.

5. Conclusions

This paper presents an algorithm for time-optimal risk-aware motion planning for curvature-constrained variable-speed vehicles. The results show superiority in time savings over the Dubins paths. The algorithm allows users to generate



(a) $k = 0$, time cost before/after smoothing: **34.51s/25.51s** **(b)** $k = 0.3$, time cost before/after smoothing: **36.30s/33.61s** **(c)** $k = 3$, time cost before/after smoothing: **41.89s/35.68s**



(d) $k = 0$, max risk before/after smoothing: **2.45/2.40** **(e)** $k = 0.3$, max risk before/after smoothing: **2.01/2.01** **(f)** $k = 3$, max risk before/after smoothing: **1.48/1.33**

Figure 7: Time-optimal risk-aware paths for different k: top row shows the speed encodings, while bottom row shows the corresponding risk encodings.

various path choices of decreasing risk at the expense of increasing time cost. The effectiveness of the proposed adaptive state pruning technique was thoroughly tested to reduce the computation time while maintaining the path quality.

REFERENCES

[1] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[2] J. Faigl and P. Váňa, "Surveillance planning with bézier curves," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 750–757, 2018.

[3] P. Salaris, D. Fontanelli, L. Pallottino, and A. Bicchi, "Shortest paths for a robot with nonholonomic and field-of-view constraints," *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 269–281, 2010.

[4] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[5] N. Tsiogkas and D. Lane, "Dcop: Dubins correlated orienteering problem optimizing sensing missions of a nonholonomic vehicle under budget constraints," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2926–2933, 2018.

[6] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvature-constrained shortest-path problem," in *The Third International Workshop on the Algorithmic Foundations of Robotics*, Houston, TX, 1998, pp. 49–57.

[7] P. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides, "Curvature-constrained shortest paths in a convex polygon," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1814–1851, 2002.

[8] T. McGee and J. Hedrick, "Optimal path planning with a kinematic airplane model," *Journal of guidance, control, and dynamics*, vol. 30, no. 2, pp. 629–633, 2007.

[9] A. Wolek and C. Woolsey, "Feasible dubins paths in presence of unknown, unsteady velocity disturbances," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 4, pp. 782–787, 2014.

[10] A. Wolek, E. Cliff, and C. Woolsey, "Time-optimal path planning for a kinematic car with variable speed," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 10, pp. 2374–2390, 2016.

[11] J. Song, S. Gupta, and T. Wettergren, "Time-optimal path planning for underwater vehicles in obstacle constrained environments," in *MTS/IEEE OCEANS'17*, Anchorage, AK, 2017, pp. 1–6.

[12] A. Pereira, J. Binney, G. Hollinger, and G. Sukhatme, "Risk-aware path planning for autonomous underwater vehicles using predictive ocean models," *Journal of Field Robotics*, vol. 30, no. 5, pp. 741–762, 2013.

[13] J. Hernández, M. Moll, E. Vidal, M. Carreras, and L. Kavraki, "Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, South Korea, 2016, pp. 1313–1320.

[14] D. Liu, M. Cong, and Y. Du, "Episodic memory-based robotic planning under uncertainty," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 2, pp. 1762–1772, 2017.

[15] B. Pfeiffer, R. Batta, K. Klamroth, and R. Nagi, "Path planning for uavs in the presence of threat zones using probabilistic modeling," *IEEE Trans. Autom. Control*, vol. 43, pp. 278–283, 2005.

[16] M. Davoodi, "Bi-objective path planning using deterministic algorithms," *Robotics and Autonomous Systems*, vol. 93, pp. 105–115, 2017.

[17] B. Wang, S. Li, J. Guo, and Q. Chen, "Car-like mobile robot path planning in rough terrain using multi-objective particle swarm optimization algorithm," *Neurocomputing*, vol. 282, pp. 42–51, 2018.

[18] H. Huang and A. V. Savkin, "Viable path planning for data collection robots in a sensing field with obstacles," *Computer Communications*, vol. 111, pp. 84–96, 2017.

[19] T. Fraichard and H. Asama, "Inevitable collision states: A step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

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

[21] L. Paull, S. Saeedi, M. Seto, and H. Li, "Auv navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014.