# T*: Time-optimal Risk-aware Motion Planning for Curvature-constrained Vehicles

**This work has been published in:**
J. Song, S. Gupta and T.A. Wettergren, "T*: Time-optimal Risk-aware Motion Planning for Curvature-constrained Vehicles", ***IEEE Robotics and Automation Letters***, Vol. 4, Issue 1, pp 33-40, 2019.

The copyright of this presentation is held by the authors and the LINKS lab.

# $T^\star$: Time-optimal Risk-aware Motion Planning for Curvature-constrained Vehicles

❖ **Objective:** Develop a time-optimal risk-aware motion planning algorithm for curvature-constrained variable-speed vehicle.

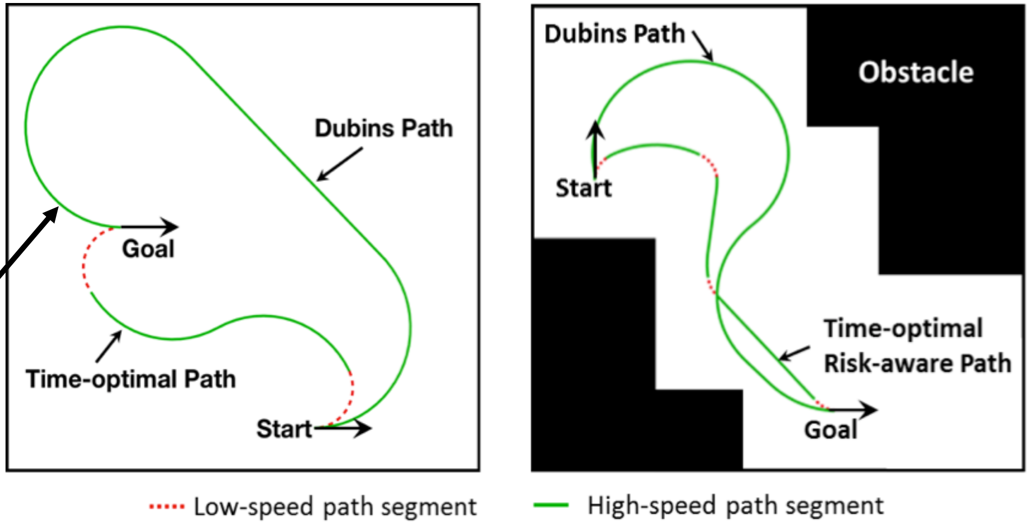
❖ **Challenges:**

- *NP-hardness*[1]: the time-optimal motion planning problem for curvature-constrained vehicles in the presence of obstacles is NP-hard; thus no exact and efficient solution exists.
- Variable-speed vehicle but its motion has bounded curvature.
  Time-optimal motion planning is ***unsolved*** with obstacles
- Vehicle safety: existing risk measures are typically based on vehicle location, without considering **heading and/or speed**.

❖ **Features and Novel Contributions of the $T^\star$ Algorithm:**

- Provides a solution to this ***unsolved problem***.
- Proposed a novel risk function based on *collision time,* using *full information* of vehicle state, including its location to nearby obstacles, heading and speed
- Integrated risk into time-optimal motion planning
- Proposed an adaptive state pruning technique to significantly reduce computation complexity

***Dubins vehicle***: a constant-speed curvature-constrained vehicle




Time-optimal (risk-aware) paths for a variable-speed vehicle vs. the Dubins paths

[1] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvature- constrained shortest-path problem," in Proceedings of the Third International Work- shop on the Algorithmic Foundations of Robotics,(Houston, Texas, USA), 1998, pp. 49–57.

# Problem Formulation

## The Search Area and The Autonomous Vehicle

❖ ***The Autonomous Vehicle***

Let $(x, y, \theta) \in SE(2)$, the vehicle motion is described as:

$$\begin{cases} \dot{x}(t) = v(t) \cdot \cos\theta(t) \\ \dot{y}(t) = v(t) \cdot \sin\theta(t) \\ \dot{\theta}(t) = u(t) \end{cases}$$

***Variable Speed***
$v(t) \in [v_{min}, 1]$

***Bounded Turn Rate*** $u(t) \in [-u_{max}, u_{max}]$, where $u_{max} \in \mathbb{R}^+$ is the max turn rate, and "+/-" indicates a *left/right* turn.

***Curvature*** $\kappa = \frac{u}{v}$ is bounded by $0 \le |\kappa| \le \left|\frac{u_{\max}}{v_{\min}}\right|$.

- Note: curvature is the inverse of its turning radius.

***Turning Radius***: when turn with $\pm u_{max}$, the turning radius is proportional to its speed $v$:

$$\mathbf{R} = \frac{1}{u_{max}} \quad \text{and} \quad \mathbf{r} = \frac{v_{min}}{u_{max}}$$

❖ ***Vehicle State***: $p = (x, y, \theta, v)$

❖ ***Tiling of the Search Area*** $\mathcal{R} \subset \mathbb{R}^2$ : construct a tiling $T = \{\tau_\alpha, \alpha = 1, \ldots |T|\}$ over $\mathcal{R}$. Then, identify the obstacle cells if it is (partially) occupied by any a-priori known obstacle.

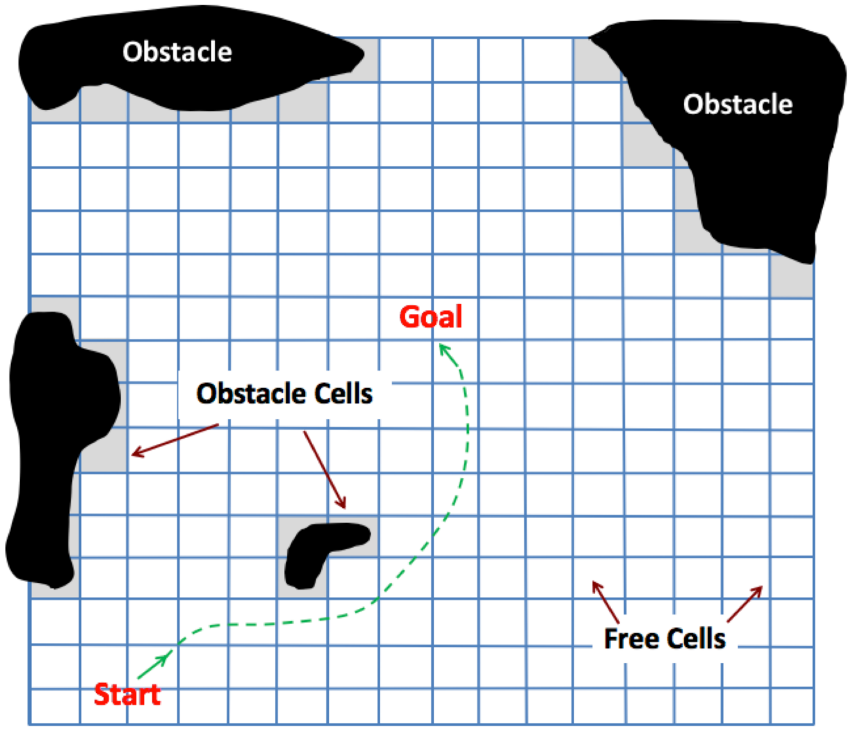



**Figure**. Tiling of the search area

# Problem Formulation

## The Cost Function

- ***Admissible Control***: Let $\Gamma$ denote the set of collision-free paths between the start state $p_{start}$ and goal state $p_{goal}$. For each path $\gamma \in \Gamma$, the control $c(s) = (\kappa, v)$ at any point $s$ on path $\gamma$, belongs to[1]:

$$\boxed{\Omega = \left\{(\kappa, v): v_{\min} \leq v \leq 1, |\kappa| \leq \frac{u_{\max}}{v}\right\}}$$

- ***Cost of a Path***: Let $R(s)$ denote the risk cost at point $s$ on path $\gamma$. Then the total cost is written as:

$$\boxed{J(\gamma) = \int_{\gamma} \underbrace{\boxed{R(s)}}_{\text{risk cost}} \cdot \underbrace{\boxed{\frac{1}{v(s)} ds}}_{\text{time cost}}}$$

- ***Objective***: Now, the goal is to find the optimal control $c^{\star} \in \Omega$, which generates the collision-free path $\gamma^{\star}$, such that: $J(\gamma^{\star}) \leq J(\gamma), \forall \gamma \in \Gamma$.

- ***NP-hardness***: The problem of deciding whether a curvature-constrained collision-free path exists between two given poses amid polygonal obstacles is *NP-hard*[2][3].
  - No efficient exact algorithms exist for the time-optimal motion planning problem in the presence of obstacles.


[1] 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.

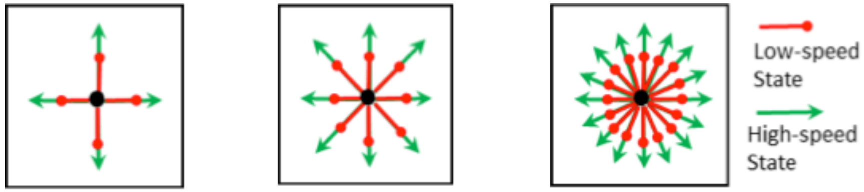
[2] P. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri, and S. Whitesides, "Curvature- constrained shortest paths in a convex polygon," SIAM Journal on Computing, vol. 31, no. 6, pp. 1814–1851, 2002.

[3] S. Lazard, J. Reif, and H. Wang, "The complexity of the two dimensional curvature- constrained shortest-path problem," in Proceedings of the Third International Work- shop on the Algorithmic Foundations of Robotics,(Houston, Texas, USA), 1998, pp. 49–57.

# $\mathbf{T}^\star$ Algorithm

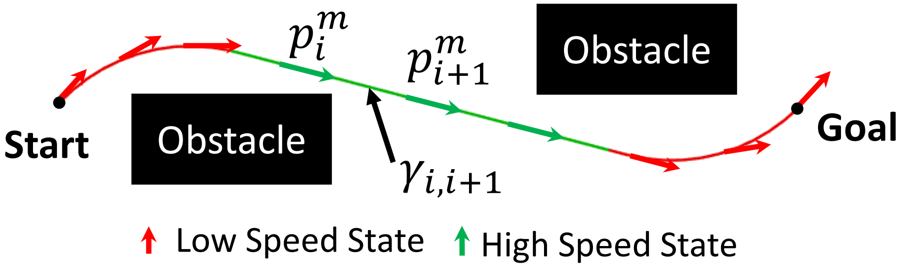
## Configuration Space and Approximate Optimization Function

❖ **The Configuration Space:** Assign obstacle-free cells with a set of possible vehicle states as follows.




(a) 4-orientation (b) 8-orientation (c) 16-orientation

An example of the state sequence $P^m$




❖ **Approximate Optimization Function**

- Let $\mathrm{P} = \{P^m, m = 1, \dots |\mathcal{P}|\}$ be the set of state sequences, where $P^m = [p_1^m, p_2^m, \dots p_n^m]$ is a state sequence from $p_{start} = p_1^m$ to $p_{goal} = p_n^m$. Its total cost is:

$$J(P^m) = \sum_{i=1}^{n-1} \tilde{J}(p_i^m, p_{i+1}^m)$$

where the step-wise cost:

$$\tilde{J}(p_i^m, p_{i+1}^m) = \min J(\gamma_{i,i+1})$$

and $\gamma_{i,i+1}$ refers to any collision-free path from $p_i^m$ to $p_{i+1}^m$.

- Assume the risk to be *constant* along $\gamma_{i,i+1}$, then:

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

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

- Now, the goal is to find the optimal state sequence $P^\star \in \mathrm{P}$, s.t., $J(P^\star) \leq J(P^m), \forall P^m \in \mathrm{P}$

# T* Algorithm

## The Time Cost $\mathcal{T}(\gamma_{i,i+1})$

- ***Sufficient Set $\Gamma^c$***: For any state pair $p_i^m$ and $p_{i+1}^m$, authors of [1] showed the sufficient set that contains the time-optimal path in *obstacle-free* space has 34 candidate paths.

**Table**: The set of candidate paths ($\Gamma^c$) between two 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 |

*Dubins Paths (Constant Speed)*

- ***Each Path $\gamma_{i,i+1} \in \Gamma^c$***
  - *Path Segments*
    1. Bang arc $(B)$: turn with $u_{\max}$ and $v_{\max}$ (i.e., radius $\mathbf{R}$);
    2. Cornering arc $(C)$: turn with $u_{\max}$ and $v_{\min}$, (i.e., radius $\mathbf{r}$);
    3. Straight line $(S)$: move straight with $v_{max}$.
  - *Direction of Each Segment*
    1. $L$: turn left with $u_{max}$;
    2. $R$: turn right with $-u_{max}$;
    3. $S$: move straight.
- ***Cost of $\gamma_{i,i+1}$***
  - $B$ or $C$ segment: turn with angle $\Delta\theta$, time is $\frac{|\Delta\theta|}{u_{\max}}$
  - $S$ segment: move straight with distance $d$, time is $d/v_{\max}$
- ***Solving for Path Parameters:***
  - **Constraints**: $\gamma_{i,i+1}$ must start with $p_{start}$ and reach $p_{goal}$, thus requiring $\Delta x$, $\Delta y$, $\Delta\theta$ and speeds to be matched
  - But the number of parameters can be more…
  - Optimize path parameters using IPOPT[1]

[1] 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.

# $\mathbf{T}^\star$ Algorithm

## The OCPS Table

- ***The OCPS Table:*** To avoid computational burden during planning, we construct offline the *Optimized Candidate Paths for State-pairs* (OCPS) table to store the optimized candidate paths for all possible state pairs in the neighborhood.
- ***Entries in the OCPS Table***: The table has at most 2048 ($8\times16\times16$) optimized candidate paths. Also, it is partitioned into the following subsets based on the starting/ending arc types:
  - $\Gamma_{BB}^{c}$: paths that start and end with $B$ arcs
  - $\Gamma_{BC}^{c}$: paths that start with $B$ arc and end with $C$ arc
  - $\Gamma_{CB}^{c}$: paths that start with $C$ arc and end with $B$ arc
  - $\Gamma_{CC}^{c}$: paths that start and end with $C$ arcs
- ***Quick Query for Time Cost $\mathcal{T}(\gamma_{i,i+1})$***: For a given pair of states $p_i^m$ and $p_{i+1}^m$, the planner initiates a query to the OCPS table to obtain a set of optimized candidate paths. Then, $\mathcal{T}(\gamma_{i,i+1})$ is determined by the collision-free one with the least time; otherwise, $\mathcal{T}(\gamma_{i,i+1})=\infty$.

**Table**: The set of candidate paths ($\Gamma^c$) between two 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 |

$\Gamma_{BB}^{c}$ (Nos. 1–12), $\Gamma_{BC}^{c}$ (Nos. 13–20), $\Gamma_{CB}^{c}$ (Nos. 21–28), $\Gamma_{CC}^{c}$ (Nos. 29–34)

# $T^{\star}$ Algorithm

## Construction of the OCPS Table based on Symmetry

- ***Reduction in Construction Time:*** Within the 2048 state pairs in the OCPS tables, one can use symmetry to significantly reduce the number of state pairs for optimization, which leads to much less optimization time during table construction.
- ***Symmetric State Pairs***: There are **272** (out of 2048) unique state pairs in the OCPS table, while the rest can be derived from them.

**Case 1:** the heading of the start state faces north, east, west or south. Suppose it faces north, then there are:

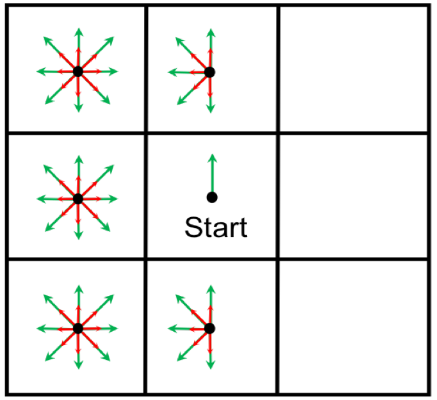
- 3 neighbor cells where each cell contains 8 heading choices and 2 speeds;
- 2 neighbor cells where each cell contains 5 heading choices and 2 speeds.

Thus, it has a total number of $3{\times}8{\times}2 + 5{\times}2{\times}2 = 68$ state pairs. Moreover, since the start state can take 2 speeds, there will be $68{\times}2 = 136$ states pairs for this case.

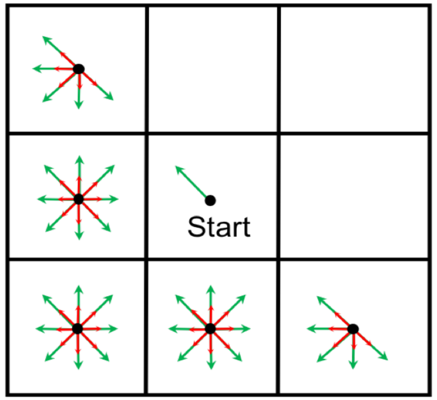
**Case 2:** the heading of the start state faces diagonally, i.e., northeast, northwest, southeast and southwest. Suppose it faces northwest, then there are

- 3 neighbor cells where each cell consists of 8 heading choices and 2 speeds;
- 2 neighbor cells where each cell contains 5 heading choices and 2 speeds.

Thus, it has $3{\times}8{\times}2 + 5{\times}2{\times}2 = 68$ state pairs. Moreover, due to two choices of speeds at the start state, there are $68{\times}2 = 136$ state pairs.




(a) Case 1: Start state facing up, down, left, right




(b) Case 2: Start state facing diagonally

# T$^\star$ Algorithm

## The Risk Cost $R(\gamma_{i,i+1})$



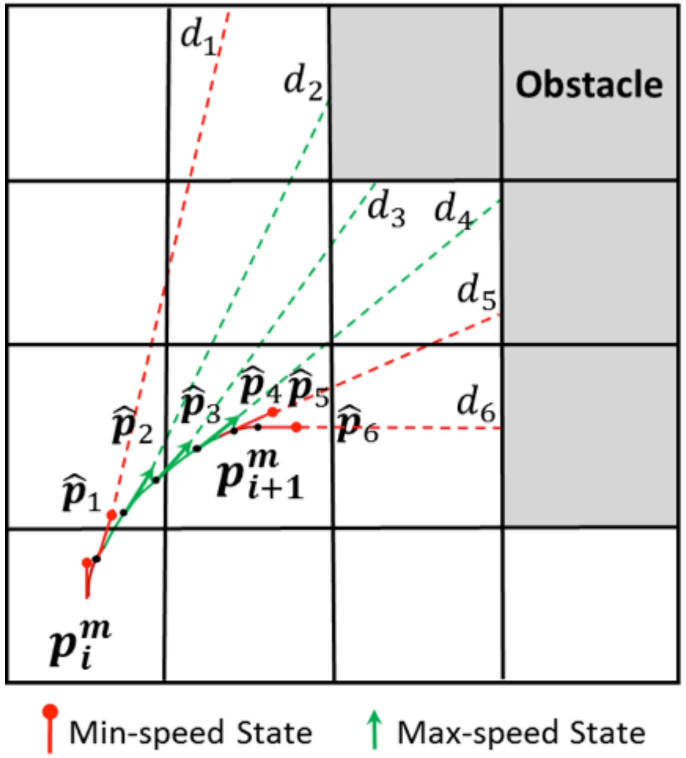
***Safety Threshold $t^\star$***: For any state, the vehicle is assumed safe if the collision time $t_\ell$ is over a threshold $t^\star \in \mathbb{R}^+$. This is the time for the vehicle to to fully stop, maneuver around, or re-gain its control.

❖ ***Collision Time $t_\ell$:***

- First, evenly sample along $\gamma_{i,i+1} \in \Gamma^c$ with sampling interval $\Delta d \in \mathbb{R}^+$, and obtain a set of states, $\{\hat{p}_\ell, \ell = 1, \dots, M\}$, where $\hat{p}_M = p_{i+1}^m$.
- Then, for each $\hat{p}_\ell = (x_\ell, y_\ell, \theta_\ell, v_\ell)$, one can geometrically compute the *collision distance* $d_\ell$ (see figure).
- The collision time is defined as follows, where $v_\ell \in \{v_{\min}, 1\}$.

$$\boxed{t_\ell = \frac{d_\ell}{v_\ell}}$$

Sample states between $p_i^m$ and $p_{i+1}^m$ for $M = 6$



# $\mathbf{T}^{\star}$ Algorithm

## The Risk Cost $R(\gamma_{i,i+1})$

❖ ***Risk of a Sample State*** $\widehat{\boldsymbol{p}}_{\ell}$:

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

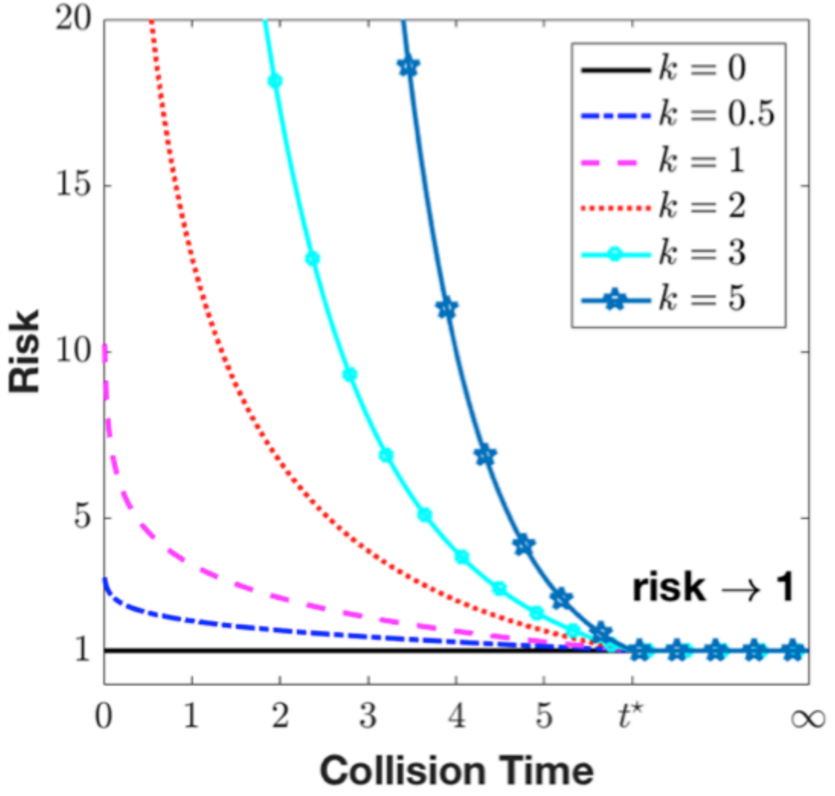
❖ ***Risk Cost*** $\boldsymbol{R(\gamma_{i,i+1})}$:

Let $k \in \mathbb{R}^{+}$ be the user-controllable risk factor

$$\boxed{R(\gamma_{i,i+1}) = \max_{\ell \in \{1,\ldots,M\}} \left(risk(\hat{p}_{\ell})\right)^{k}}$$

***Risk Cost*** $\boldsymbol{R(\gamma_{i,i+1})}$ is evaluated at the *most dangerous state* on $\gamma_{i,i+1}$ that results in the ***least*** collision time.

The risk function $\left(risk(\widehat{\boldsymbol{p}}_{\ell})\right)^{k}$



# $\mathbf{T}^\star$ Algorithm

## Searching for the Optimal State Sequence $P^\star$

❖ **$A^\star$ Search:**

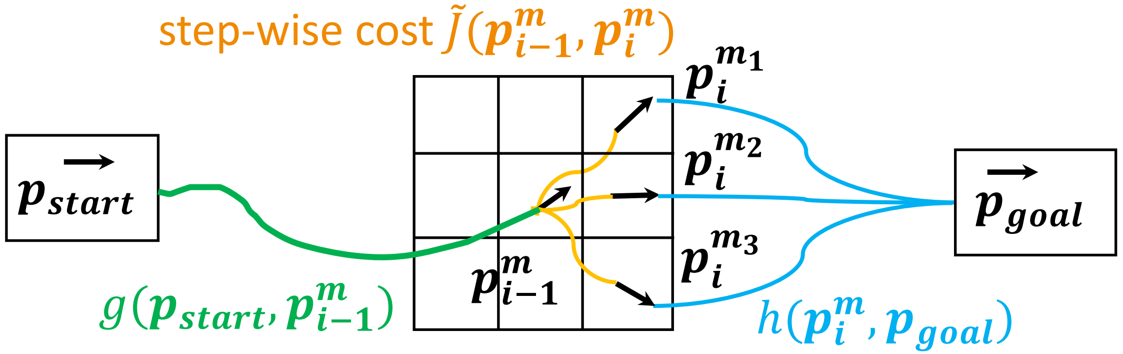
- We adopt the framework of $A^\star$ algorithm[1] to search for the optimal state sequence $P^\star$.
- The cost associated with each intermediate state $p_i^m$ is defined as

$$\boxed{f(p_i^m) = g(p_{start}, p_i^m) + h(p_i^m, p_{goal})}$$

❖ **The Cumulative Cost $g(p_{start}, p_i^m)$**

- The cumulative cost from $p_{start}$ to state $p_i^m$ along the state sequence $[p_1^m, p_2^m, \dots p_{i-1}^m, p_i^m]$, is defined as

$$\boxed{g(p_{start}, p_i^m) = \sum_{j=1}^{i-1} \tilde{J}(p_j^m, p_{j+1}^m)}$$




❖ **The Heuristic Cost $h(p_i^m, p_{goal})$**

- **Requirements**:
  - ➢ Must be admissible to guarantee optimality of $P^\star$
  - ➢ Should consider kinematic constraints of vehicle
- **Admissible Design**: Define $h(p_i^m, p_{goal})$ as the length of the shortest Dubins path using turning radius $r$, divided by the maximum speed $v_{\max}$.


[1] P.E. Hart, N.J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4. 4 (2): 100–107, 1968.

# T* Algorithm

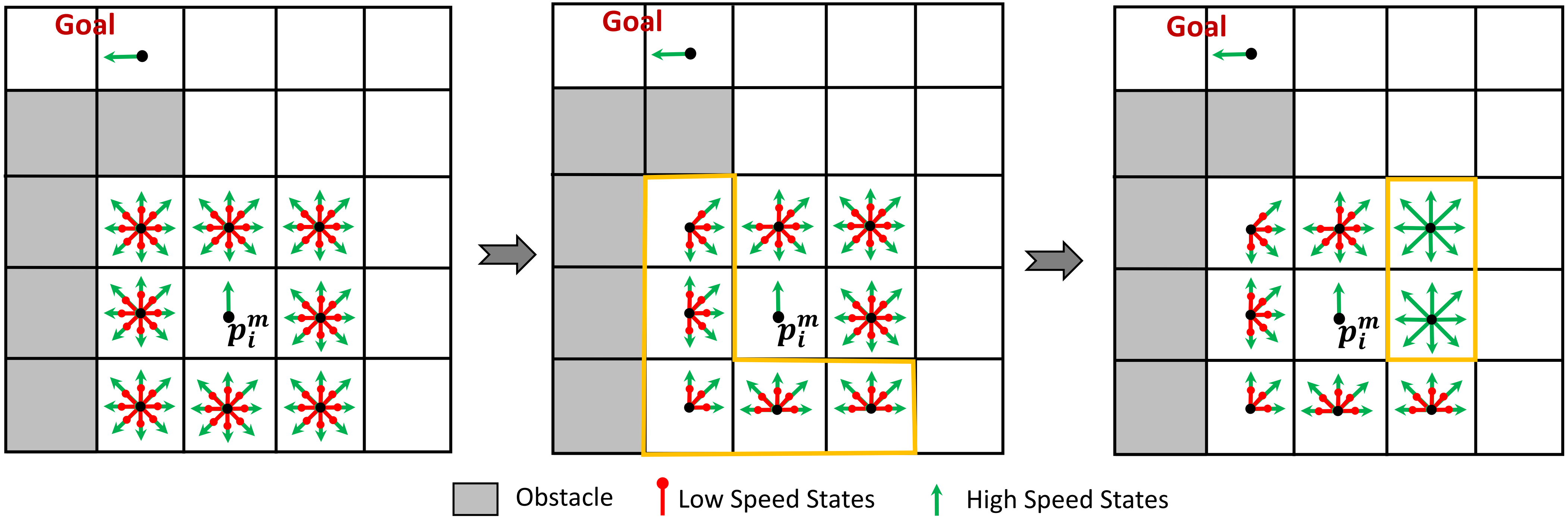
## The Adaptive State Pruning Technique

***Idea:*** dynamically identify and remove the states from the configuration space that are less likely to be part of $P^\star$

***Step 0*: *Basic State Expansion***
(associate *8-orientation* states in each neighbor cell)

***Step 1*: *Obstacle-based Pruning***
(The states close to and facing obstacles/boundaries are pruned)

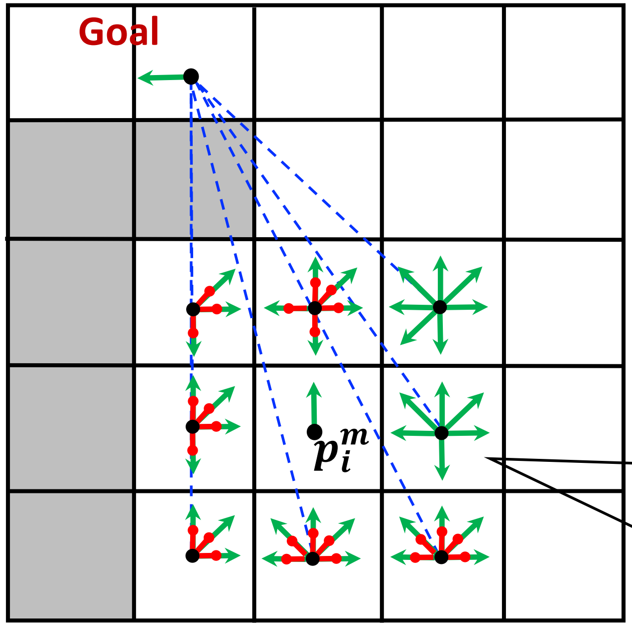
***Step 2*: *Speed-based Pruning***
(Low-speed states located far from obstacles are pruned)



# $\mathbf{T}^{\star}$ Algorithm

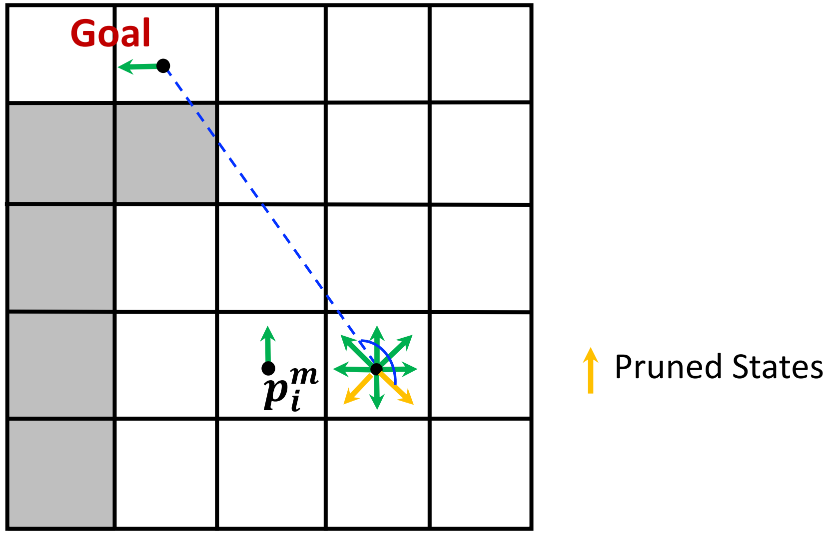
## The Adaptive State Pruning Technique

***Step 3***: ***Heading-based Pruning***
(Diagonally facing states in an opposite direction to the goal are pruned based on certain threshold)




- ***Pruning Threshold*** $\boldsymbol{\eta_0} \in [\mathbf{0}, \boldsymbol{\pi}]$: connect the state and the goal with a straight line, if the formed angle is over a threshold $\eta_0$ then this diagonal state is pruned

**Note**: Base states (i.e., up, down, left and right oriented states) are retained to ensure algorithm completeness.



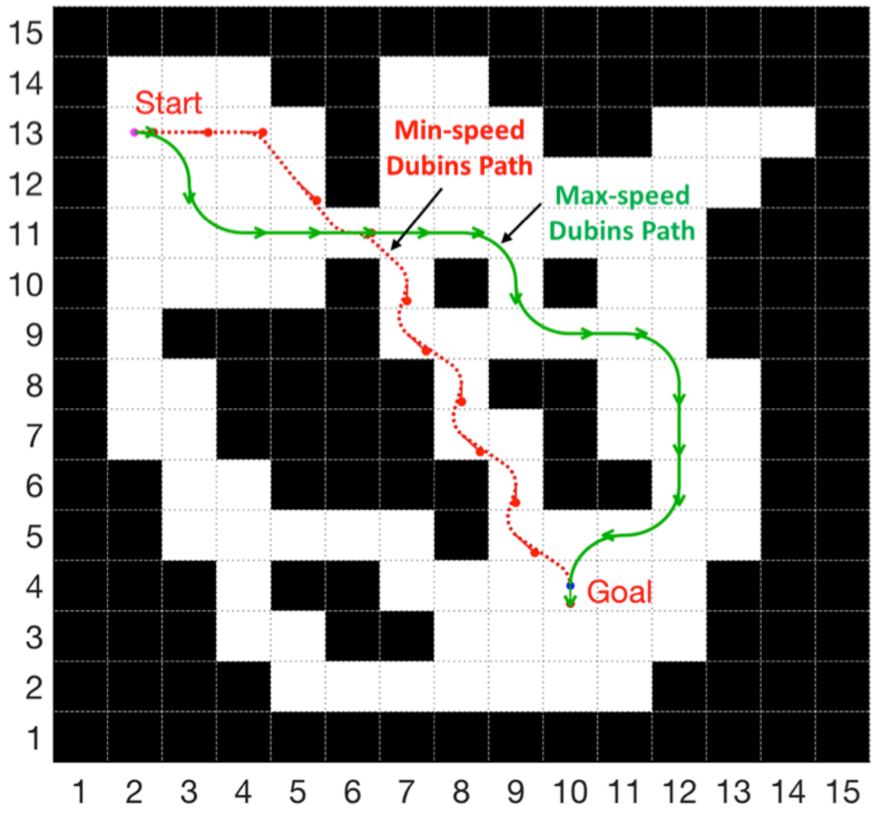

Heading-based pruning with $\eta_0 = \pi/2$

# Simulation Validations

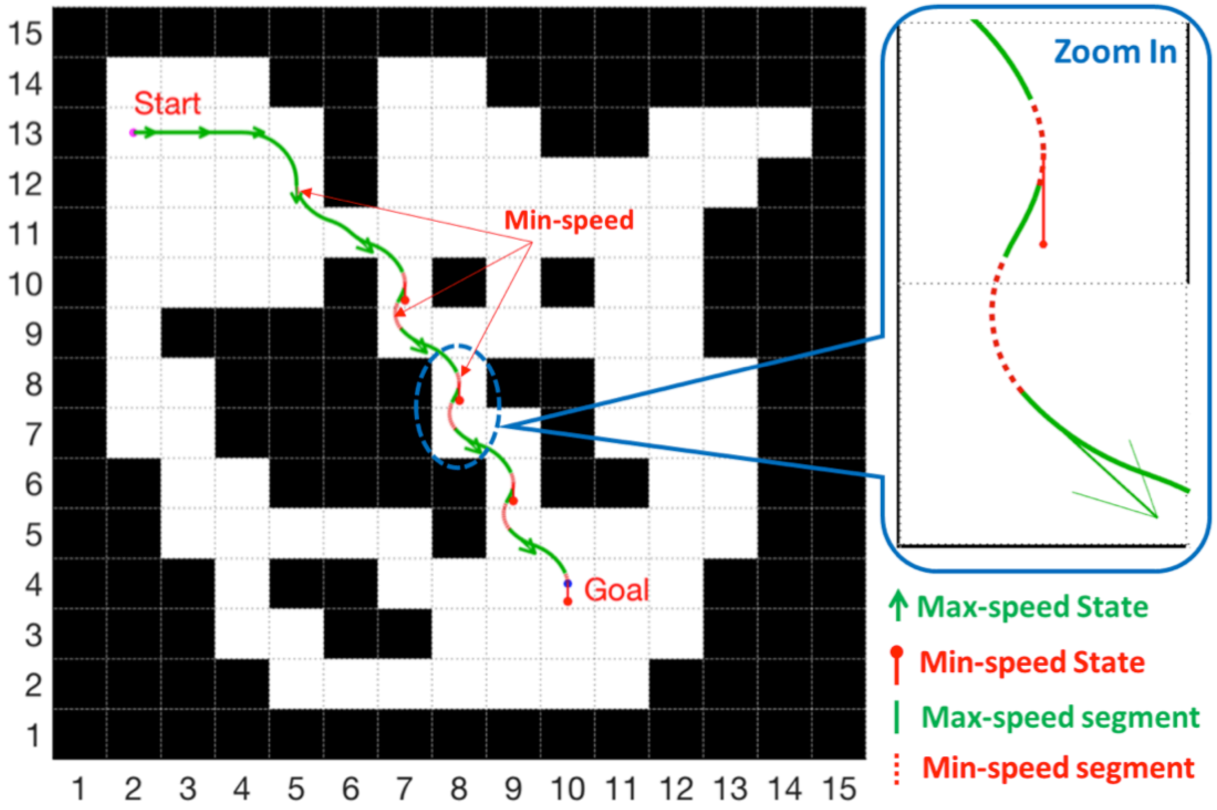
## Scenario 1: Time-optimal Paths vs. Dubins Paths

- **Parameters**: $p_{start} = (4\text{m}, 26\text{m}, 0, v_{\max})$, $p_{goal} = \left(20\text{m}, 8\text{m}, \frac{3\pi}{2}, v_{\min}\right)$, $v_{\min} = 0.5\text{m/s}$, $v_{\max} = 1\text{m/s}$, $t^\star = 6\text{s}$, $\mathbf{r} = 1\text{m}$, $\mathbf{R} = 2\text{m}$, grid size = 2m, pruning threshold $\eta_0 = \pi/2$, sampling interval $\Delta d = 0.4\text{m}$; buffer around obstacles with size 0.1m.
- **Time Costs**:
  - Dubins Paths: using $v_{\max}$: $35.99\text{s}$; using $v_{\min}$: $55.95\text{s}$
  - Time-optimal Path ($k = 0$): $34.51\text{s}$

**Dubins paths with constant speeds**




**Time-optimal paths generated by $\mathbf{T}^\star$**



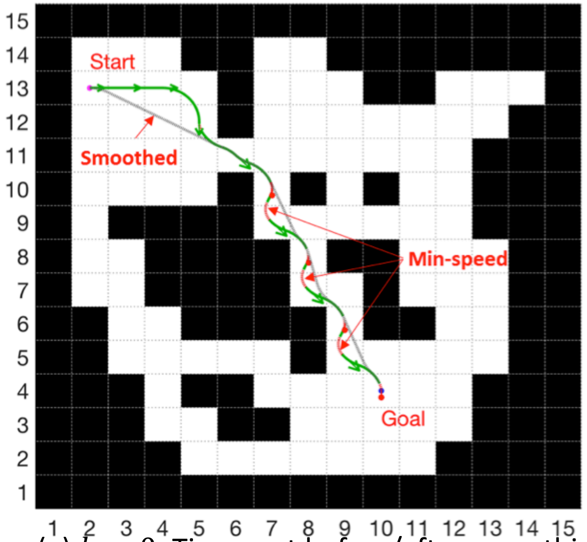
UCONN

LINKS
Laboratory of Intelligent Networks and Knowledge-Perception Systems

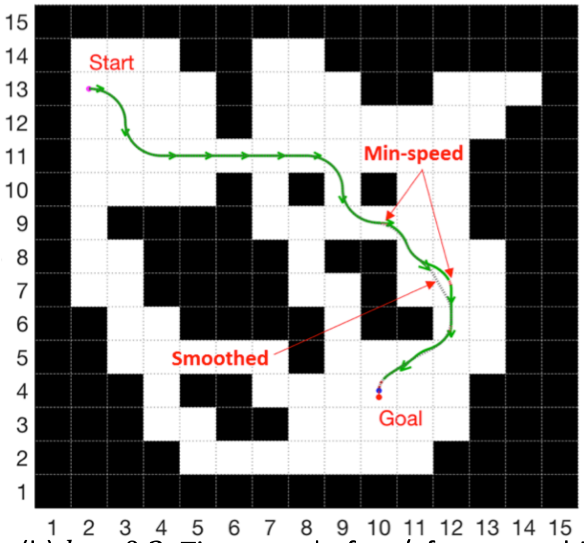
# Simulation Validations

## Scenario 1: Time-optimal Risk-aware Paths with Different Risk Parameters $k$

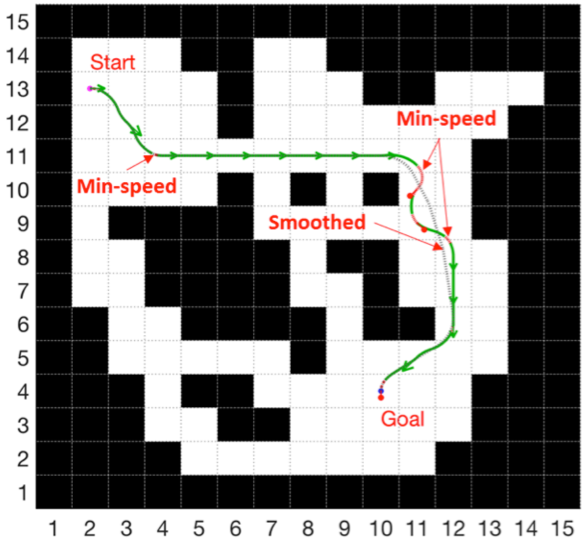
***Speed Color-coded Paths***



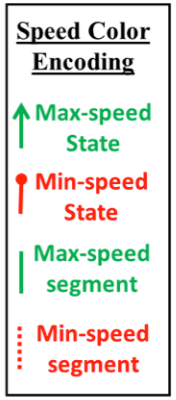

(a) $k = 0$. Time cost before/after smoothing:
**34.51s/25.51s**



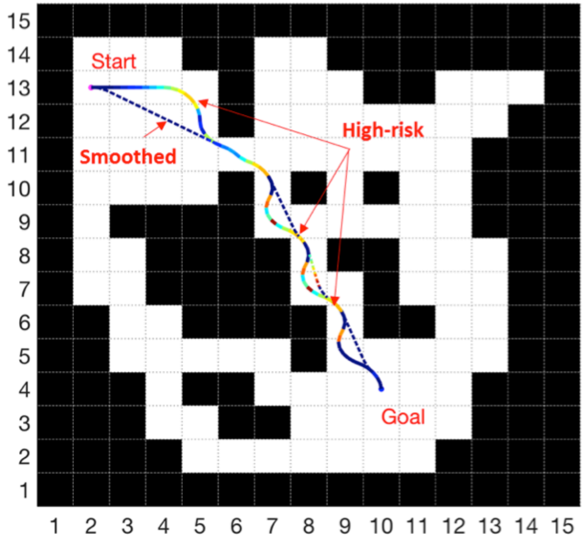

(b) $k = 0.3$. Time cost before/after smoothing:
**36.30s/33.61s**



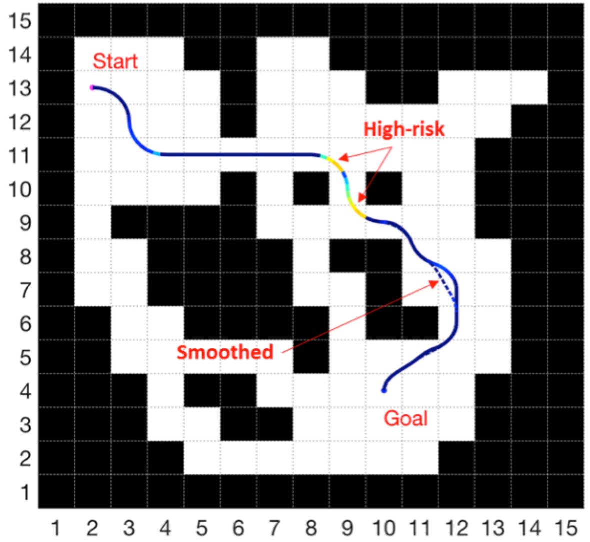

(c) $k = 3$. Time cost before/after smoothing:
**41.89s/35.68s**



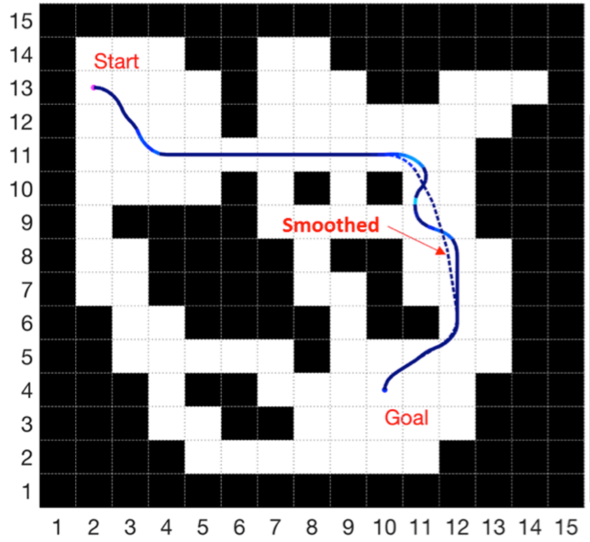

***Risk Color-coded Paths***



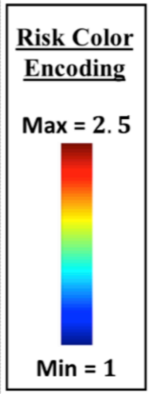

(d) $k = 0$. Max risk before/after smoothing:
**2.45/2.40**




(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**

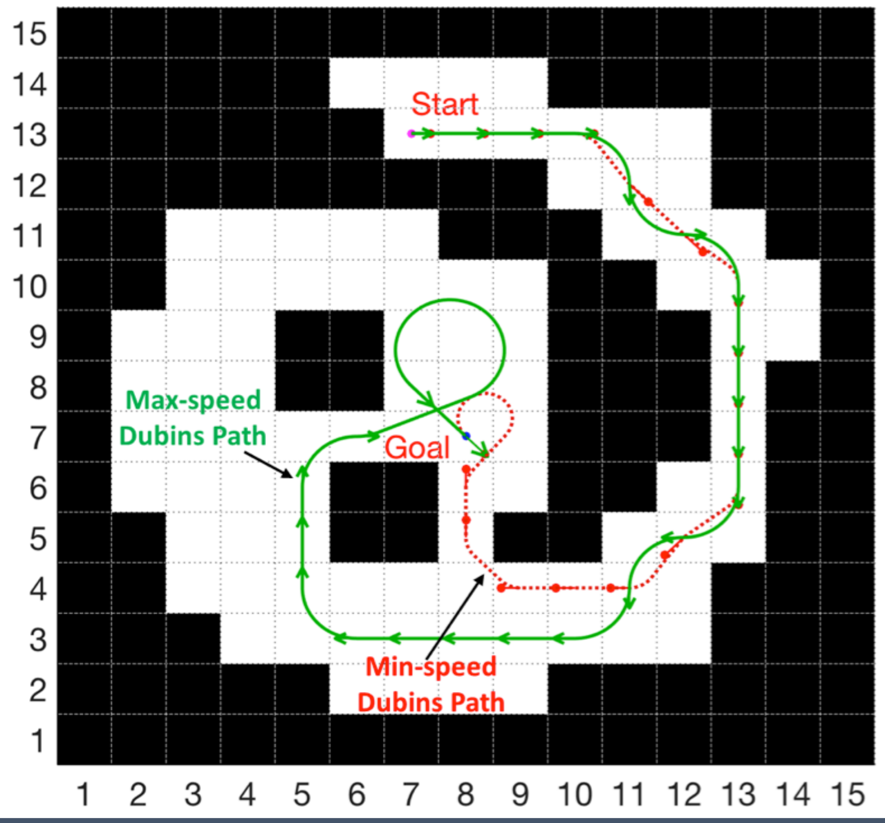


# Simulation Validations

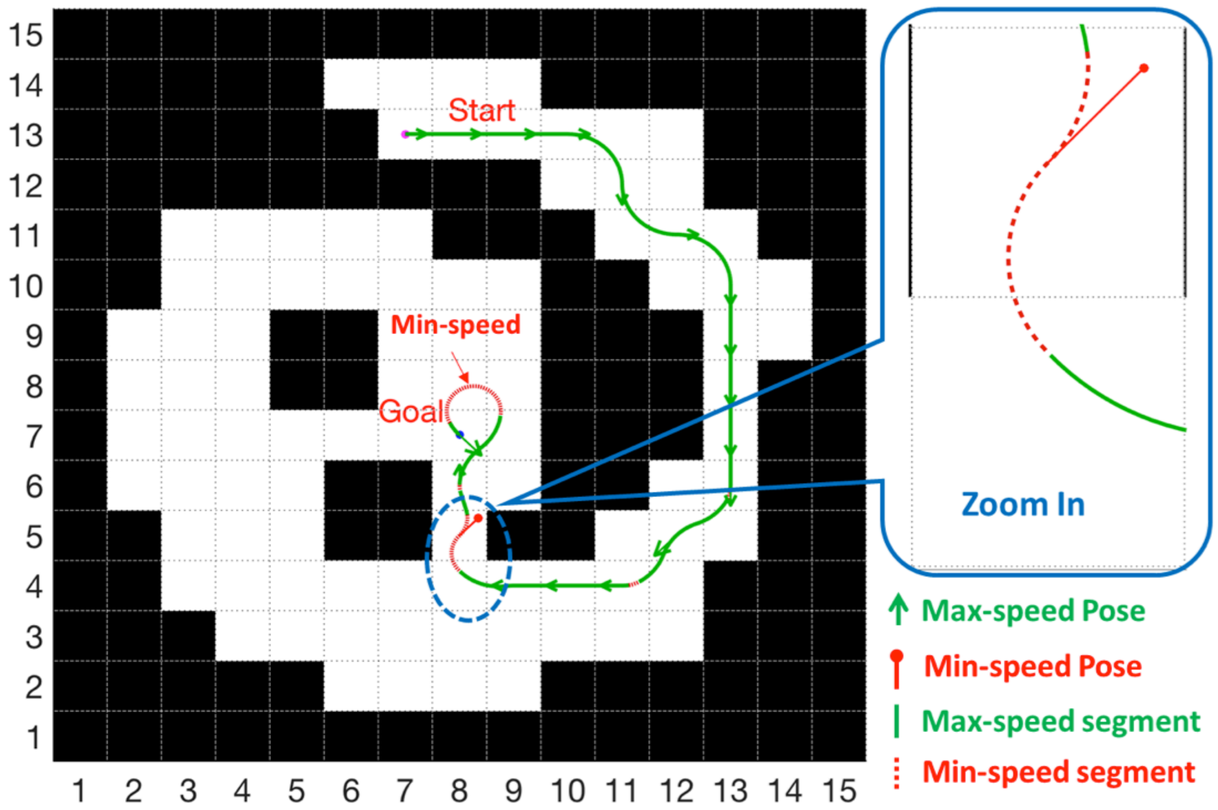
## Scenario 2: Time-optimal Paths vs. Dubins Paths

- **Parameters**: $p_{start} = (4\text{m}, 26\text{m}, 0, v_{\max})$, $p_{goal} = \left(20\text{m}, 8\text{m}, \frac{3\pi}{2}, v_{\min}\right)$, $v_{\min} = 0.5\text{m/s}$, $v_{\max} = 1\text{m/s}$, $t^\star = 6\text{s}$, $\mathbf{r} = 1\text{m}$, $\mathbf{R} = 2\text{m}$, grid size = 2m, pruning threshold $\eta_0 = \pi/2$, sampling interval $\Delta d = 0.4\text{m}$; buffer around obstacles with size 0.1m.
- **Time Costs**:
  - Dubins Paths: using $v_{\max}$: 68.65s; using $v_{\min}$: 89.47s
  - Time-optimal Path ($k = 0$): 54.24s

**Dubins paths with constant speeds**




**Time-optimal paths generated by $\mathbf{T}^\star$**



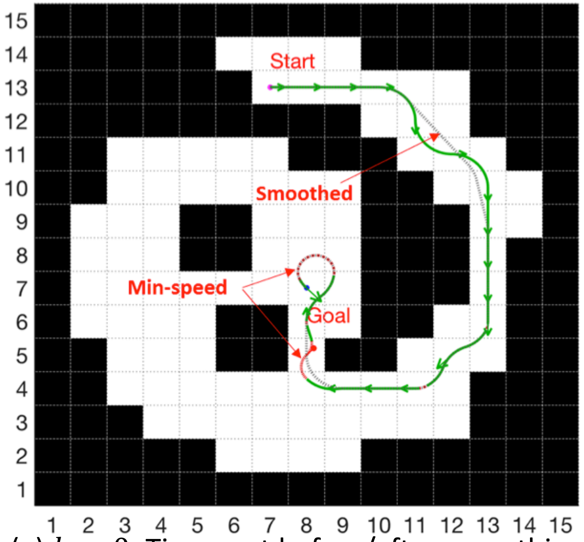
UCONN

LINKS
Laboratory of Intelligent Networks and Knowledge-Perception Systems

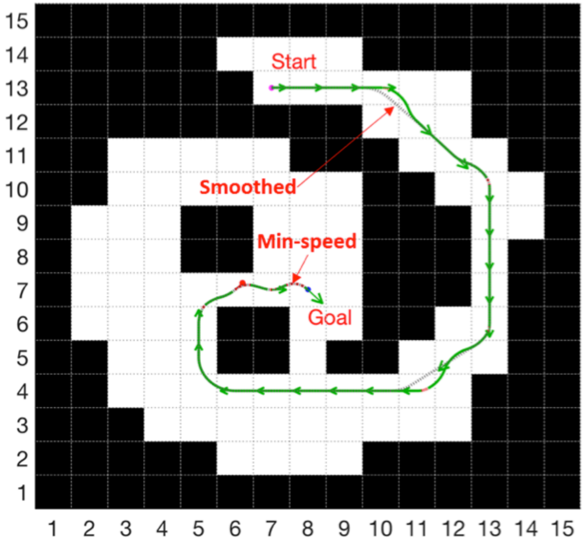
# Simulation Validations

## Scenario 2: Time-optimal Risk-aware Paths with Different Risk Parameters $k$

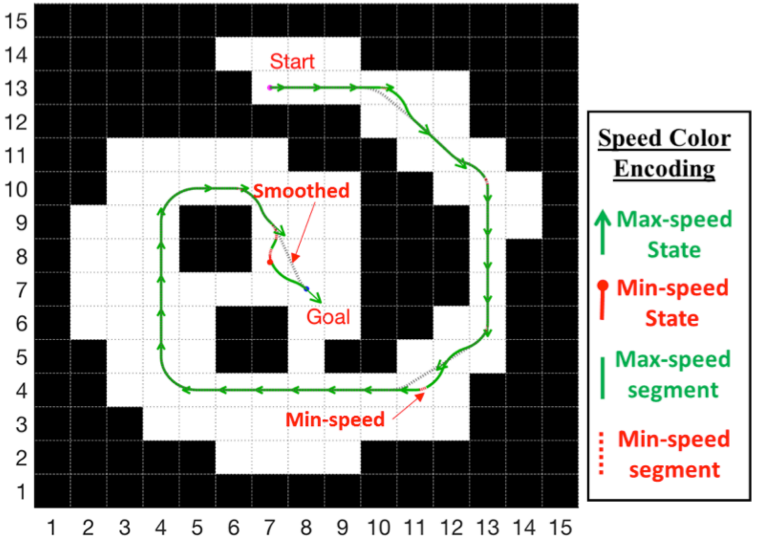
***Speed Color-coded Paths***




(a) $k = 0$. Time cost before/after smoothing: **54.24s/48.99s**

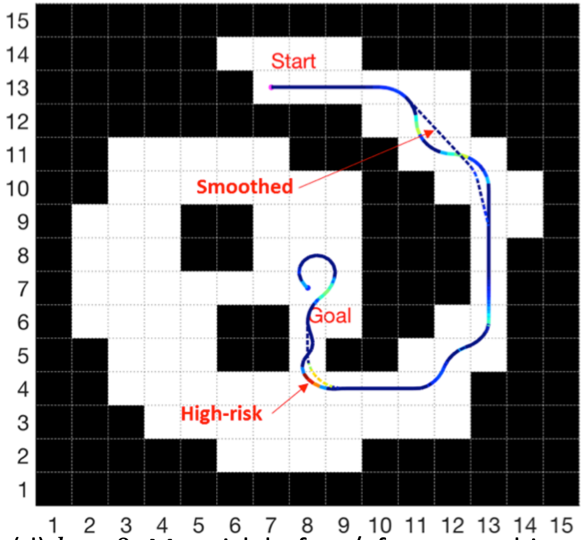



(b) $k = 0.3$. Time cost before/after smoothing: **56.00s/53.75s**

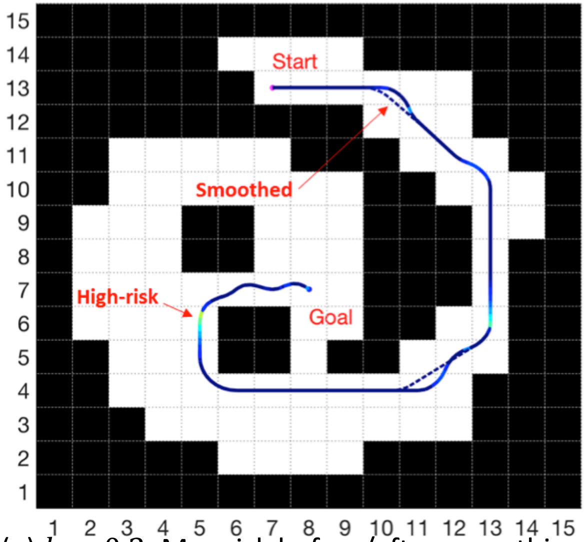



(c) $k = 3.5$. Time cost before/after smoothing: **68.92s/63.10s**

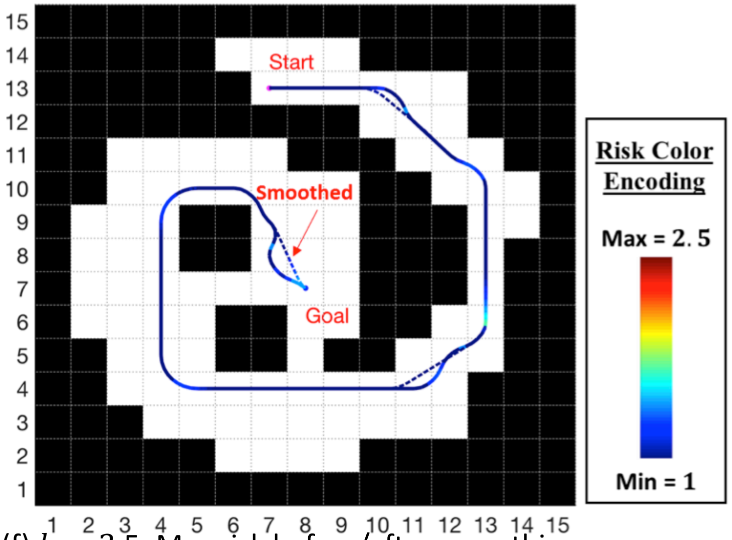
***Risk Color-coded Paths***




(d) $k = 0$. Max risk before/after smoothing: **2.45/2.01**




(e) $k = 0.3$. Max risk before/after smoothing: **1.83/1.83**




(f) $k = 3.5$. Max risk before/after smoothing: **1.69/1.66**

# Effectiveness of the Adaptive State Pruning Technique

- The uncertainty in heading angle has been incorporated into risk cost $risk(\hat{p}_\ell)$, i.e., $\theta_\ell \pm \Delta\theta$, where $\Delta\theta = 1.5^o$.
- **Note**: a higher $\eta_0$ retains more diagonally facing states during the heading-based pruning, thus may produce better results.
- When $\eta_0$ reduces from $\pi$ to $\pi/4$, the computation time reduces, while the total cost $J(P^\star)$ remains more or less the same.
- The results are generated on a computer with 3.4GHz CPU and 16GB RAM. The computation times are the average over 5 runs.

**Scenario 1**

| Risk Weight | State Pruning Threshold | Total Cost $J(P^\star)$ | Computation Time | Savings in Computation Time |
|---|---|---|---|---|
| $k = 0$ | None | 33.31 | 259.92s | - |
| | $\eta_0 = \pi$ | 34.51 | 73.31s | 71.80% |
| | $\eta_0 = \pi/2$ | 34.51 | 54.29s | 79.11% |
| | $\eta_0 = \pi/4$ | 34.51 | 38.60s | 85.36% |
| $k = 0.3$ | None | 38.46 | 709.56s | - |
| | $\eta_0 = \pi$ | 38.87 | 227.90s | 67.88% |
| | $\eta_0 = \pi/2$ | 38.87 | 169.97s | 76.05% |
| | $\eta_0 = \pi/4$ | 39.52 | 124.78s | 82.41% |
| $k = 3$ | None | 62.98 | 569.43s | - |
| | $\eta_0 = \pi$ | 62.98 | 157.95s | 72.26% |
| | $\eta_0 = \pi/2$ | 62.98 | 116.08s | 79.61% |
| | $\eta_0 = \pi/4$ | 71.94 | 96.56s | 83.04% |

**Scenario 2**

| Risk Weight | State Pruning Threshold | Total Cost $J(P^\star)$ | Computation Time | Savings in Computation Time |
|---|---|---|---|---|
| $k = 0$ | None | 54.24 | 97.20s | - |
| | $\eta_0 = \pi$ | 54.24 | 69.53s | 28.47% |
| | $\eta_0 = \pi/2$ | 54.24 | 58.47s | 39.85% |
| | $\eta_0 = \pi/4$ | 54.76 | 34.82s | 64.18% |
| $k = 0.3$ | None | 59.50 | 264.51s | - |
| | $\eta_0 = \pi$ | 59.50 | 204.72s | 22.60% |
| | $\eta_0 = \pi/2$ | 59.50 | 151.03s | 42.90% |
| | $\eta_0 = \pi/4$ | 62.87 | 94.59s | 64.24% |
| $k = 3.5$ | None | 138.56 | 306.10s | - |
| | $\eta_0 = \pi$ | 138.56 | 218.03s | 28.77% |
| | $\eta_0 = \pi/2$ | 144.18 | 128.52s | 58.01% |
| | $\eta_0 = \pi/4$ | 255.52 | 94.74s | 69.05% |