SMART: Self-Morphing Adaptive Replanning Tree

Zongyuan Shen, James P. Wilson, Shalabh Gupta, Ryan Harvey

Laboratory of Intelligent Networks and Knowledge-Perception Systems

Department of Electrical & Computer Engineering, University of Connecticut, Storrs, CT, USA

Z. Shen, J. P. Wilson, S. Gupta and R. Harvey, "SMART: Self-Morphing Adaptive Replanning Tree," in *IEEE Robotics and Automation Letters*, doi: 10.1109/LRA.2023.3315210.

Videos Available: These slides contain videos that are accessible online at: https://www.youtube.com/@linkslabrobotics

SMART: Self-Morphing Adaptive Replanning Tree

❖ **Objective:** Develop a real-time reactive replanning algorithm for robot navigation in dynamic environments.

❖ **Challenges:**

- Replanning timing and path efficiency.
  - Frequent replanning.
  - Oscillating movements and large detours.
- Fast path replanning to avoid dynamic obstacles.
- Must guarantee completeness, i.e., a path can be found if it exists.
- Low computational complexity and easy applicable to real-word environments.

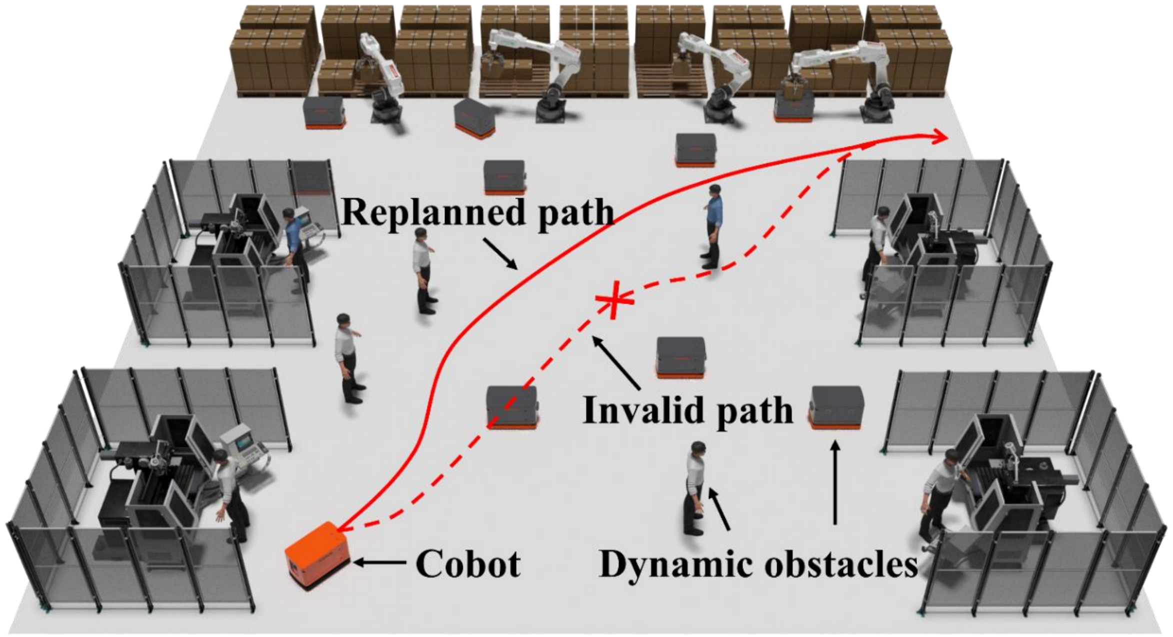


Fig. Real-time adaptive replanning in a dynamic factory scenario.

State-of-the-art and Contributions

D* Lite (2002)	Extended RRT (ERRT 2002)	Dynamic RRT (DRRT 2006)	Multipartite RRT (MPRRT 2007)
RRT^X (2016)	Horizon-based Lazy RRT* (HLRRT* 2019)	Efficient Bias-goal Factor RRT (EBGRRT 2020)	Multi-objective Dynamic RRT* (MODRRT* 2021)

❖ **Analysis of Existing Approaches:**

- D* Lite and MODRRT* repairs the solution on a graph/tree without adding new sample.
- Most algorithms except HLRRT* perform node feasibility checking around all dynamic obstacles.
- Some of the algorithms grow a single tree or a double tree-structure after pruning, resulting in repeated exploration of the already-explored area.
- Most algorithms except D* Lite and MODRRT* apply standard sampler for replanning, which could lead to wasteful and slow tree growth.

❖ **Features of SMART Algorithm:**

- Finds hot-spots for fast tree-repair and allows for random sampling if necessary
- Restricts path validation and tree-pruning to the robot's neighborhood.
- Leverages on the maximal tree structure with multiple disjoint subtrees which are incrementally merged during replanning.
- Performs informed tree-repair at hot-spots.

Problem Formulation

- Let $\mathcal{X} \in \mathbb{R}^2$ be the configuration space. Let $\mathcal{X}_N \subset \mathcal{X}$ be the configuration space free of static obstacles. Let $\mathcal{O} = \{O_i : i = 1,2,\cdots,m\}$ be the set of m dynamic circular obstacles, where $r_i \in \mathbb{R}^+$ is the radius of obstacle $O_i \in \mathcal{O}$, and $x_i(t) \in \mathcal{X}_N$ and $v_i(t) \in \mathbb{R}^+$ denote its position and speed at time $t \in \mathbb{R}^+$, respectively.

- Let $\mathcal{R}$ be a circular robot of radius $r_\mathcal{R} \in \mathbb{R}^+$, where $x_\mathcal{R}(t) \in \mathcal{X}_N$ and $v_\mathcal{R}(t) \in \mathbb{R}^+$ denote its position and speed at time t. Let x_s and x_g denote the start and goal positions.

- **Path:** Given the start position x_s and the goal position x_g, a continuous function $\sigma: [t_0, t_f] \to \mathcal{X}_N \backslash \mathcal{O}$ of bounded variation is called a path, s.t. $\sigma(t_0) = x_s$ and $\sigma(t_f) = x_g$.

- **Feasible replanning problem:** Given $\mathcal{X}_N$, x_g, $x_s = x_\mathcal{R}(t_0)$, and dynamic obstacle $\mathcal{O}$ find the feasible path $\sigma: [t_u, t_f] \to \mathcal{X}_N \backslash \mathcal{O}$ with robot's current position $x_\mathcal{R}(t_u)$ as its start, until $x_\mathcal{R}(t_u) = x_g$, constantly update $x_\mathcal{R}(t_u)$ along σ while recalculating σ whenever current path becomes infeasible.

SMART Algorithm

Initialization

❖ **Tiling**: The set $\mathcal{C} = \{c_j \subset \mathbb{R}^2 : j = 1 \ldots |\mathcal{C}|\}$ is a tiling of $\mathcal{X}$, if its elements, called tiles (or cells),

- have exclusive interiors, i.e., $c_j^o \cap c_{j'}^o = \emptyset, \forall j, j' \in \{1 \ldots |\mathcal{C}|\}, j \neq j'$. c_a^o denotes the interior of cell $c_j \in \mathcal{C}$.
- cover $\mathcal{X}$, i.e., $\mathcal{X} \subseteq \cup_{j=1}^{|\mathcal{C}|} c_j$.
- **Note:** the tiling is used only for searching hot-spots. The planning and navigation happens in $\mathcal{X}_N$.

❖ **Initial Tree:** a tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ rooted at the goal x_g is created using RRT* algorithm, where $(\mathcal{N}, \mathcal{E})$ denote the set of nodes and edges.

- Each node maintains a data structure including its position, parent, children, tree index, cell index, and node status as active or pruned.
- All nodes are marked active. An initial path is found using $\mathcal{T}$ considering only static obstacle.

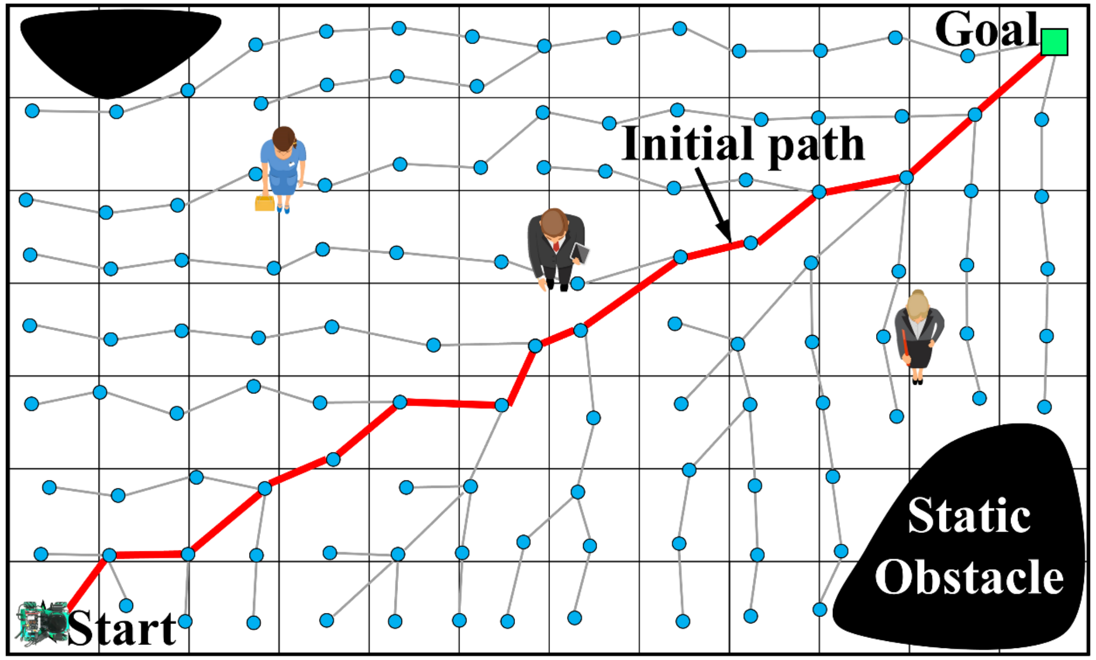


Fig. Tiling of the search area, initial tree, and path.

SMART Algorithm

Local Tree Pruning

- Dynamic obstacles far away from the robot do not pose an immediate risk, and it is computationally inefficient to do feasibility checking around all detected obstacles, thus we present local tree-pruning strategy.

- **Local Reaction Zone (LRZ):** A local reaction zone is defined for the robot $\mathcal{R}$, such that $LRZ_{\mathcal{R}}(t) = \{x \in \mathcal{X}_N : \|x - x_{\mathcal{R}}(t)\| \le v_{\mathcal{R}}(t) \times T_{RH}\}$, where $T_{RH} \in \mathbb{R}^+$ is the reaction time-horizon.

- **Obstacle Hazard Zone (OHZ):** An obstacle hazard zone is defined for each dynamic obstacle $O_i \in \mathcal{O}$, such that $OHZ_i(t) = \{x \in \mathcal{X}_N : \|x - x_i(t)\| \le v_i(t) \times T_{OH} + r_i + r_{\mathcal{R}}\}$, where $T_{OH} \in \mathbb{R}^+$ is the obstacle risk time-horizon.

- **Note:** the robot radius is added to the dynamic obstacles for feasibility checking.

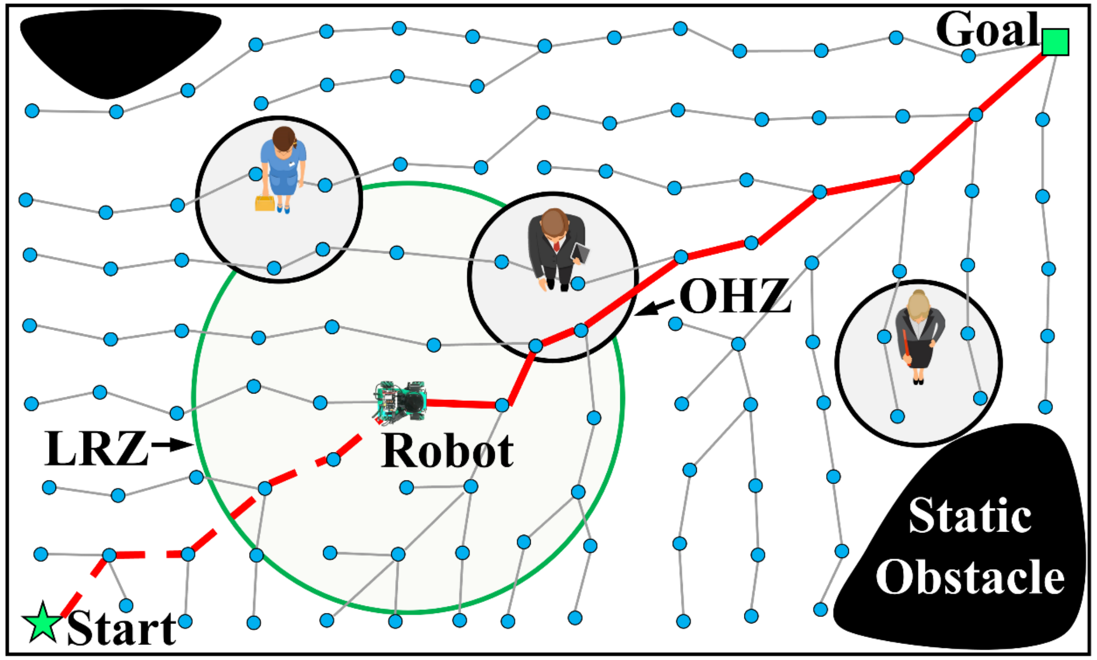


Fig. Local reaction zone and obstacle hazard zone.

SMART Algorithm
Local Tree Pruning

❖ **Critical Pruning Region (CPR):** Let $\mathcal{D}$ denote all dynamic obstacles that are intersecting with the LRZ and pose danger to the robot such that

$$\mathcal{D} = \{O_i : LRZ_{\mathcal{R}}(t) \cap OHZ_i(t) \neq \emptyset\}$$

Then, the critical pruning region is defined as

$$CPR(t) = \bigcup_{\mathcal{D}} OHZ_i(t)$$

❖ **Validity Evaluation of Current Path:**

- Checks the path nodes and edges in LRZ starting from robot's current position.
- If any of path portions is obstructed by CPR, the path is considered as invalid.

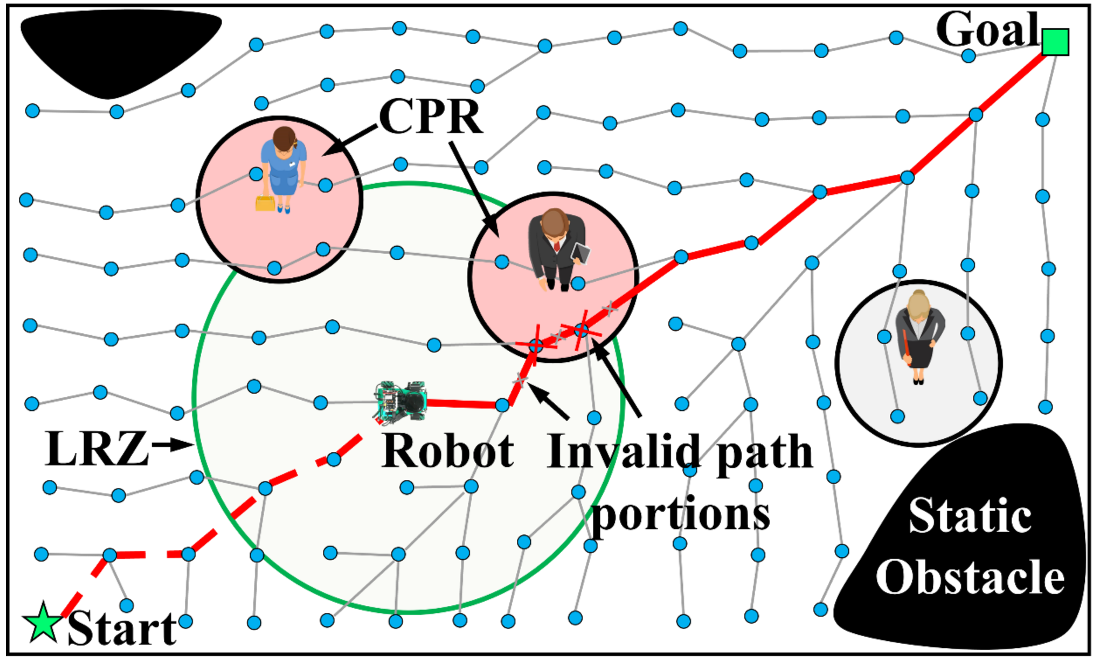


Fig. Critical pruning region and path validity evaluation.

SMART Algorithm
Local Tree Pruning

❖ **Local Tree Pruning:** If the current path is invalid, then all tree portions that fall within the CPR are considered to be risky for replanning and are thus pruned.

- If a node is invalid, it is pruned along with its edges.
- If an edge is invalid but its two end nodes are safe, then only this edge is pruned.
- **Note:** pruned nodes are stored for future additions after replanning is done. Let $\mathcal{N}^a$ and $\mathcal{N}^p$ be the sets of alive and pruned nodes, such that $\mathcal{N} = \mathcal{N}^a \cup \mathcal{N}^p$.

❖ Tree pruning could break the current tree $\mathcal{T}$ into disjoint subtrees defined as follows.

❖ **Disjoint Subtree:** A disjoint subtree is a portion of the current tree whose root is either the goal or a child of a pruned node or edge.

❖ SMART exploits the previous exploration efforts by retaining all the disjoint subtrees and perform tree-repair in the hot-spots.

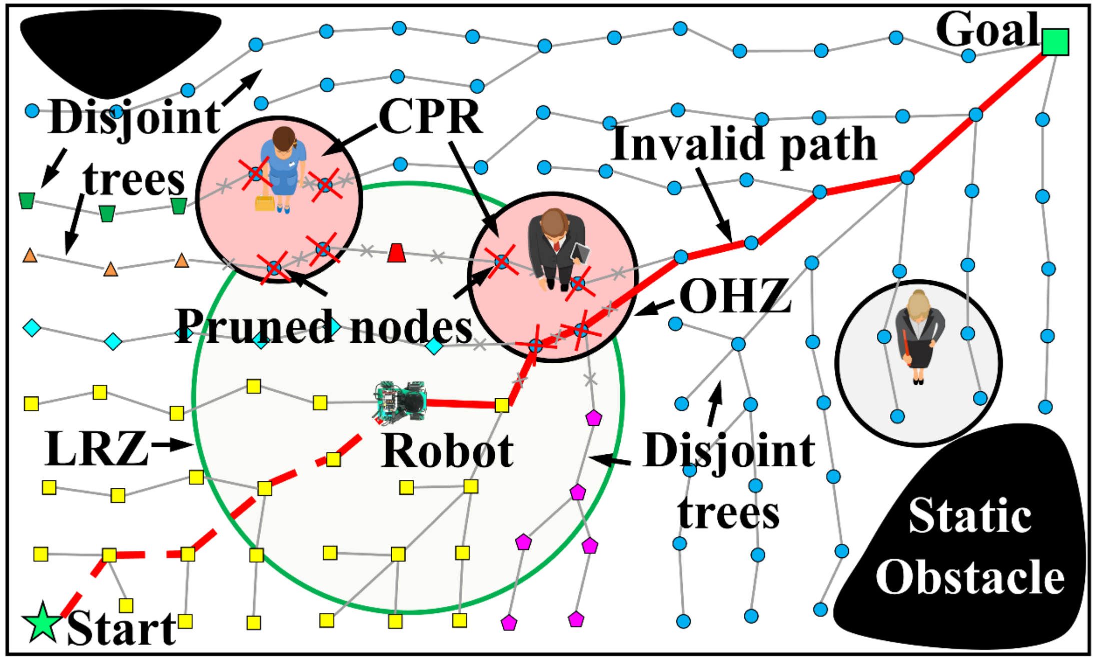


Fig. Local tree pruning.

SMART Algorithm
Hot-Spot Searching and Ranking

- ❖ Hot-Spot is the intersection region of disjoint subtrees, where reconnection can be done immediately. It is formally defined as follows.

❖ **Hot-Spot:** A cell $c \in \mathcal{C}$ is a hot-spot if:

1. It contains alive nodes of
   a) at least two disjoint subtrees, or
   b) a single subtree and at least one of its neighboring cell contains nodes of another disjoint subtree, and
2. The edge connecting at least one node pair corresponding to the two disjoint subtrees in 1.a) or 1.b) above is not obstructed by the CPR or static obstacles.

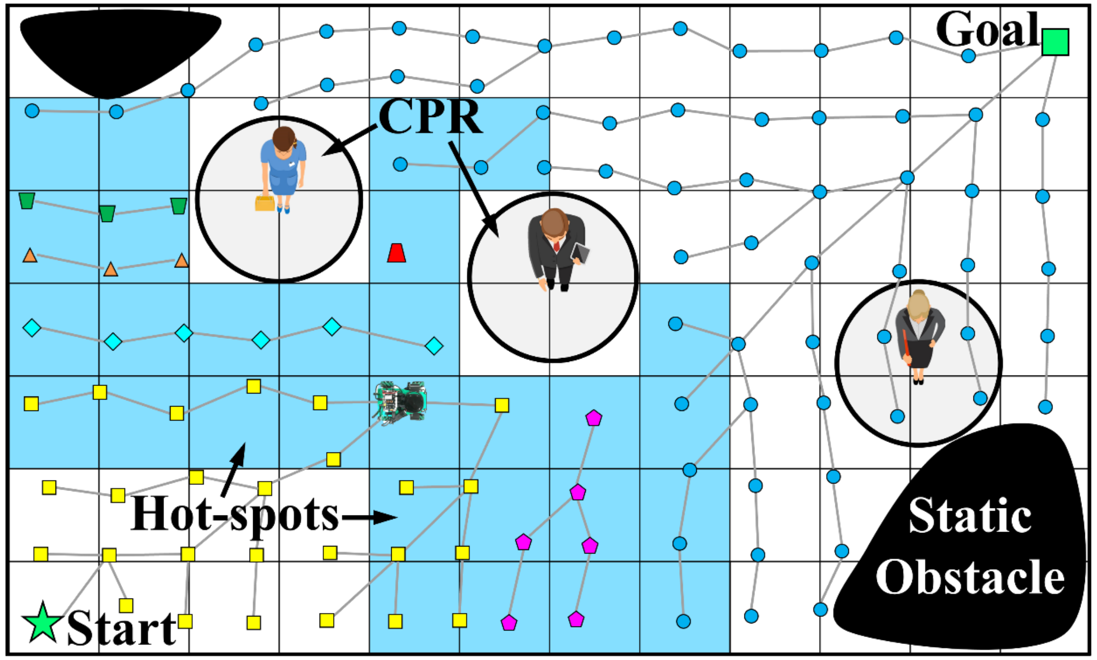


Fig. Hot-Spot.

SMART Algorithm

Hot-Spot Searching and Ranking

- Searching the entire space to find all hot-spots is computationally inefficient, thus it begins in the local search region defined as follows.

- **Local Search Region:** Let $\hat{n} \in \mathcal{N}^p$ be the pruned path node closest to the robot. Let $\hat{n}$ belong to a cell $c_{\hat{n}} \in \mathcal{C}$. Then, the local search region (LSR) is defined as $\ell \times \ell$ neighborhood ($\mathcal{S}^\ell \subseteq \mathcal{C}$) of $c_{\hat{n}}$ s.t. $\ell > 1$ is an odd number. The search starts in $\mathcal{S}^\ell$, ℓ=3.

- **Identifying Disjoint Subtrees in $\mathcal{S}^\ell$:**

1. An unlabeled node $n \in \mathcal{N}^a$ is picked within $\mathcal{S}^\ell$ and assigned a tree index $k \in \{0, \ldots, K-1\}$, where index 0 corresponds to the goal-rooted subtree.
2. It is backtracked while labeling all ancestor nodes with the same tree index.
3. The above process is repeated until all nodes inside $\mathcal{S}^\ell$ are labeled with their tree-indices.

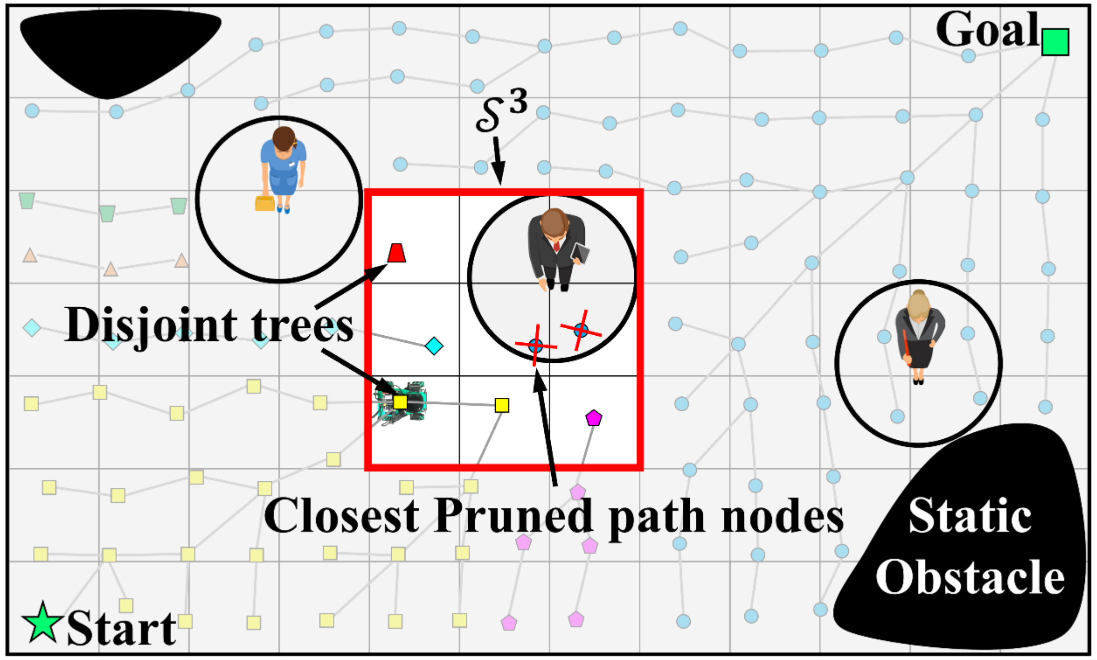


Fig. Local search region and disjoint subtrees identification.

SMART Algorithm

Hot-Spot Searching and Ranking

- The hot-spots are identified by searching all cells and their neighbors within $\mathcal{S}^{\ell}$. if no hot-spot is found within $\mathcal{S}^{\ell}$, then the search area is expanded to size $\ell = \ell+2$ and previous steps are repeated until at least one hot-spot is found.
- **Hot-Spot Map:** A hot-spot map is defined on $\mathcal{S}^{\ell}$ such that $h^{\ell}: \mathcal{S}^{\ell} \rightarrow \{-1,1\}$, where $-1,1$ denote hot-spot and not a hot-spot, respectively.

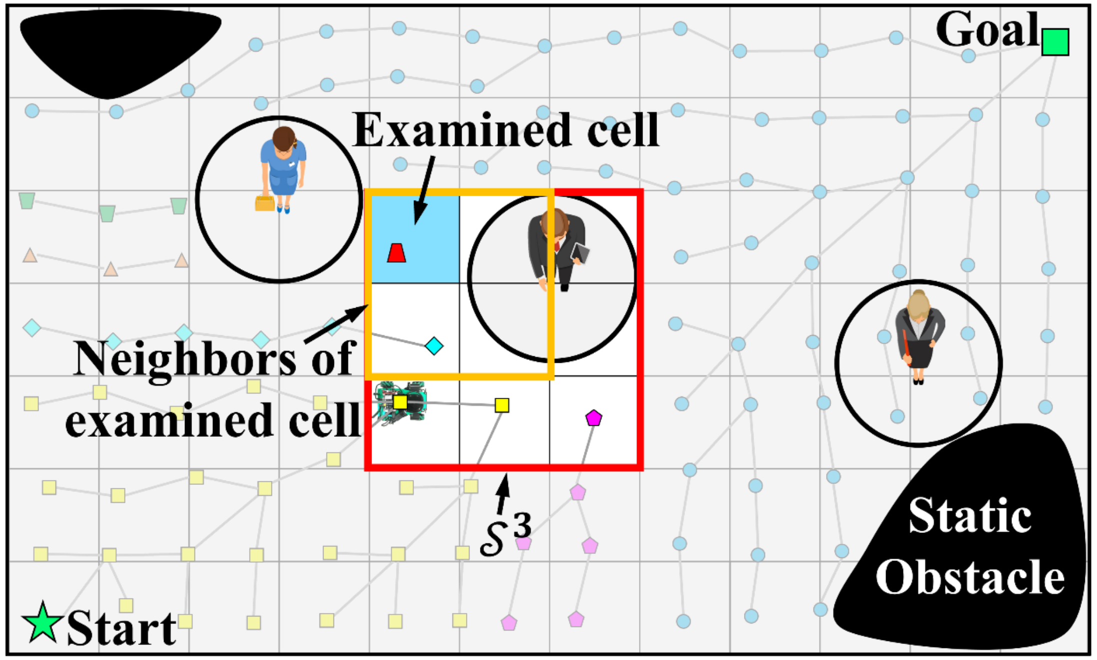


Fig. Hot-spot identification for the examined cell.

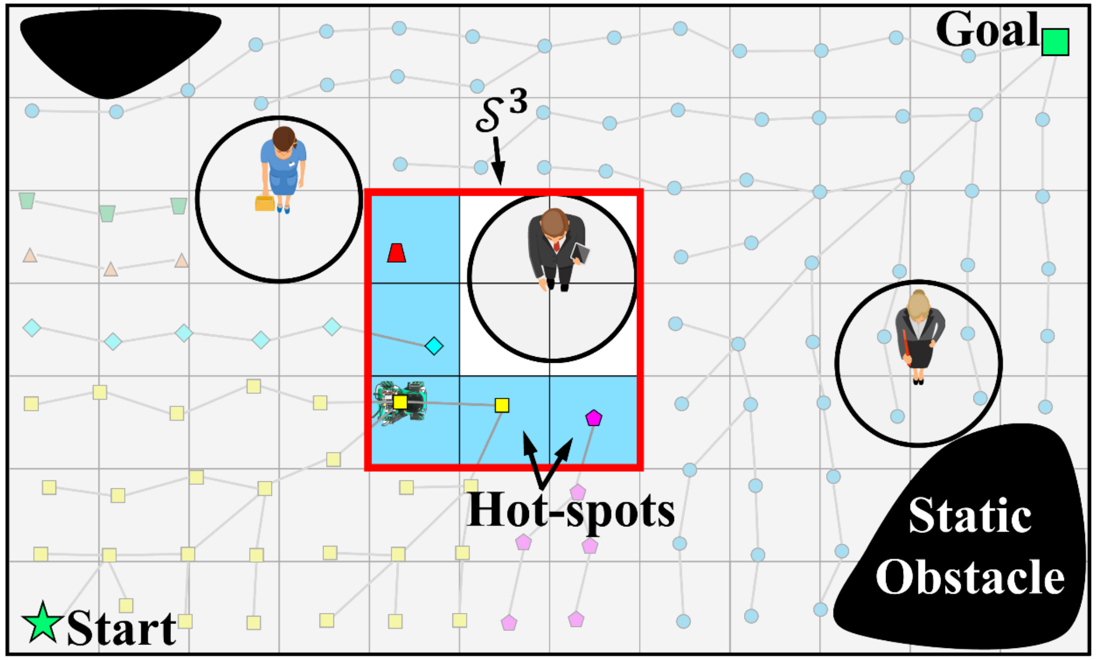


Fig. Hot-Spot map.

SMART Algorithm
Hot-Spot Searching and Ranking

❖ **Utility Map:** Let $\mathcal{H}^\ell \subseteq \mathcal{S}^\ell$ denote the set of all hot-spots in $\mathcal{S}^\ell$. A utility map is defined on $\mathcal{H}^\ell$ such that $\mathcal{U}^\ell : \mathcal{H}^\ell \rightarrow \mathbb{R}^+$, where the utility of a cell $c \in \mathcal{H}^\ell$ is computed as follows.

$$\mathcal{U}^\ell(c) = \begin{cases} \dfrac{1}{\|x_\mathcal{R}(t) - p_c\|_2 + \min\limits_{n_i \in \mathcal{N}_c \cap \mathcal{N}_0} g(n_i)} & if\, \mathcal{N}_c \cap \mathcal{N}_0 \neq \emptyset \\ \dfrac{1}{\|x_\mathcal{R}(t) - p_c\|_2 + \left\|p_c - x_g\right\|_2} & else \end{cases}$$

- $x_\mathcal{R}(t)$ is the robot position
- $p_c \in \mathbb{R}^2$ is the centroid of a hot-spot $c \in \mathcal{H}^\ell$
- $\mathcal{N}_c \subset \mathcal{N}^a$ is the set of alive nodes inside c
- $\mathcal{N}_0 \subset \mathcal{N}^a$ is the set of nodes of the goal-rooted subtree $\mathcal{T}_0$
- $g(n_i)$ returns the travel cost from node n_i to the goal via the path on $\mathcal{T}_0$
- x_g is the goal position

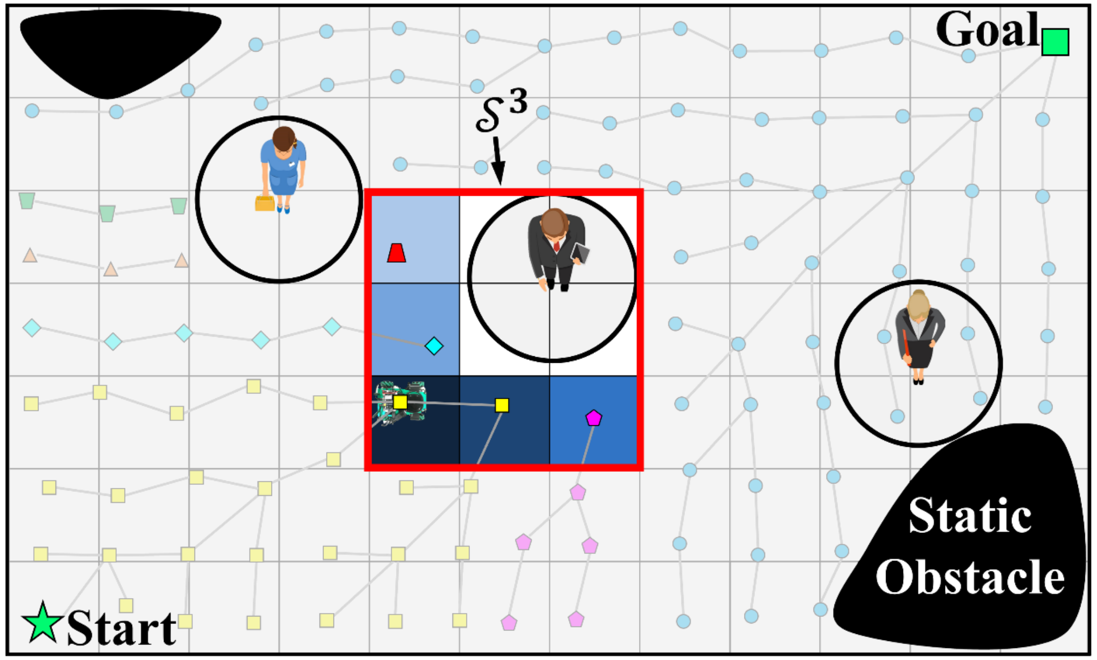


Fig. Utility map.

SMART Algorithm

Tree Reconnection

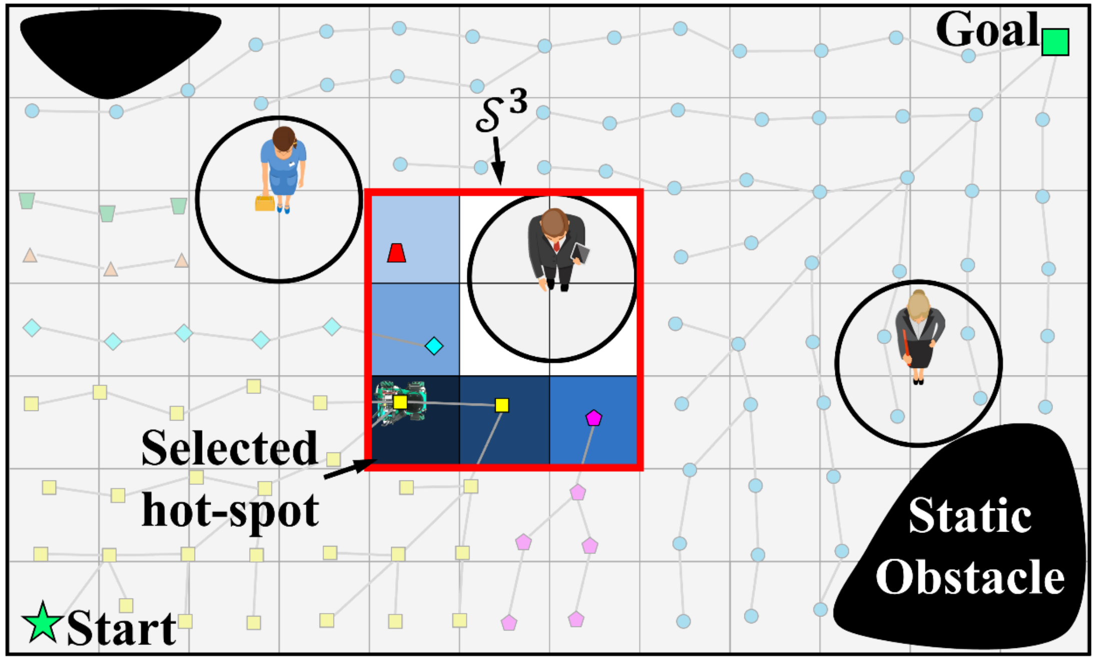


Fig. Hot-spots are color-coded with utilities and the highest one is selected.

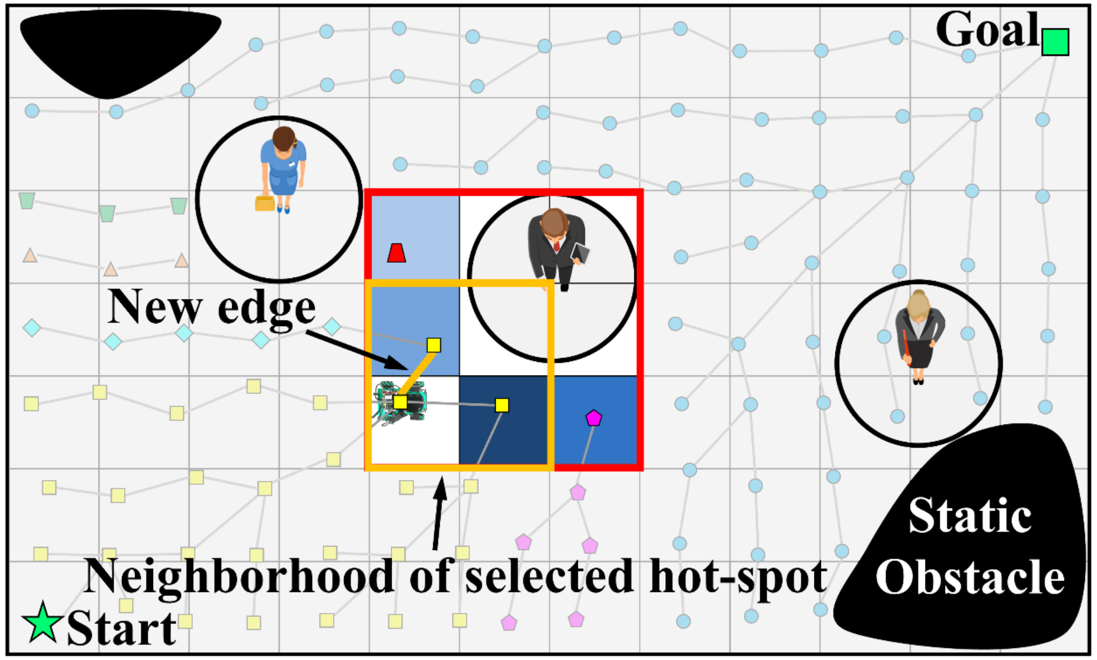


Fig. The tree is repaired by connecting the nodes of the disjoint subtrees in the neighborhood of the selected hot-spot. Then, the utility map is updated.

SMART Algorithm

Tree Reconnection

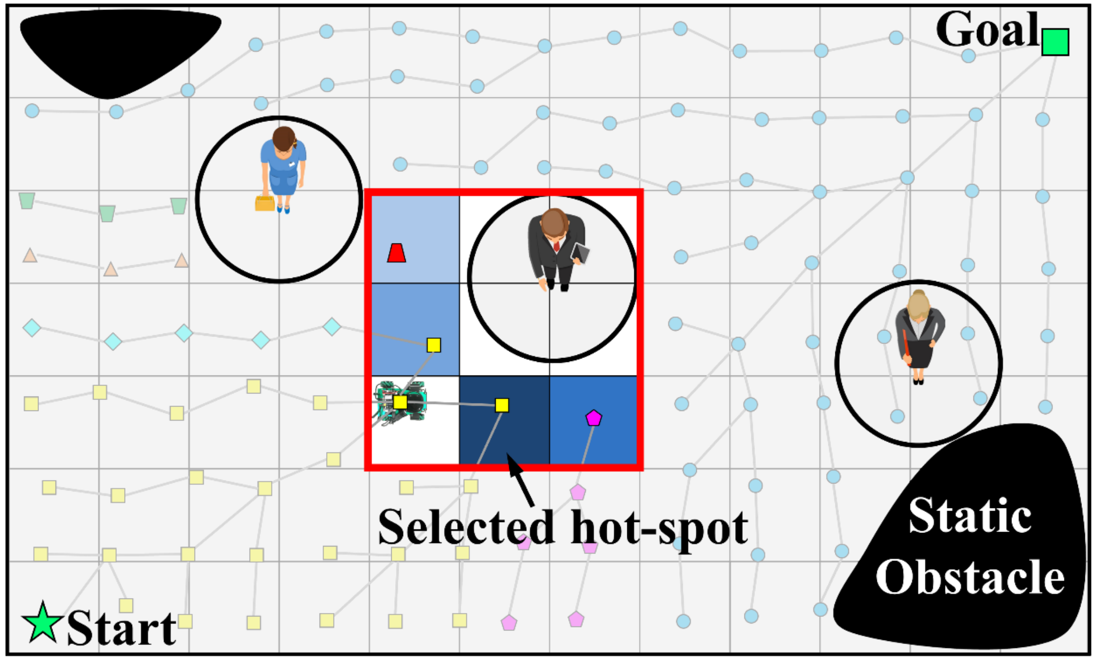


Fig. Hot-spots are color-coded with utilities and the highest one is selected.

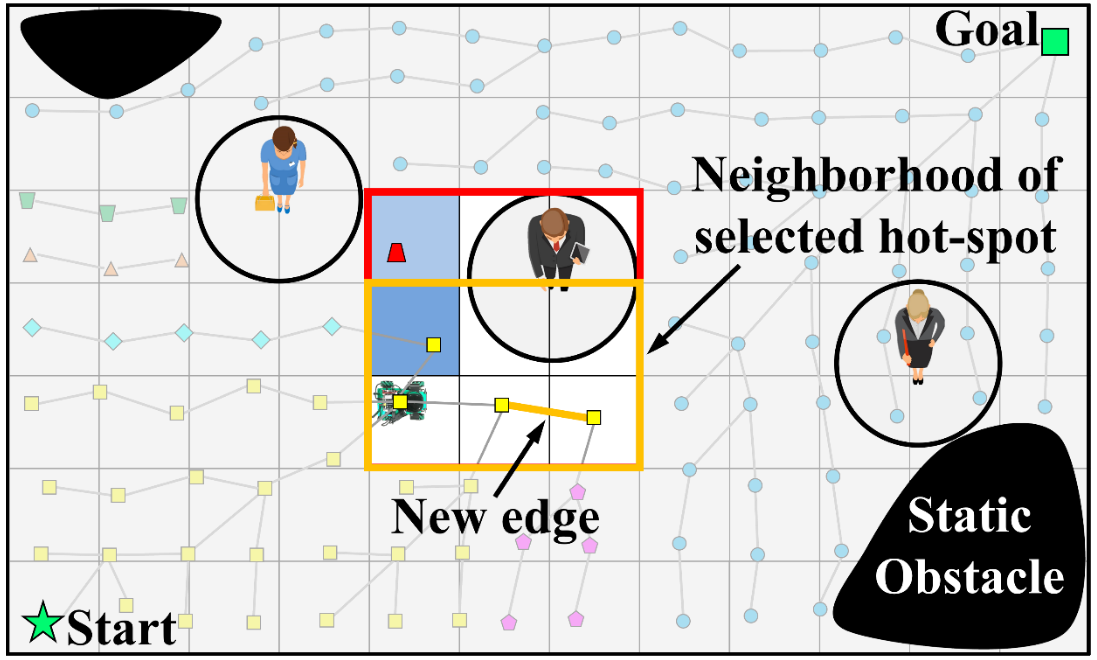


Fig. The tree is further repaired and the map is updated.

SMART Algorithm

Tree Reconnection

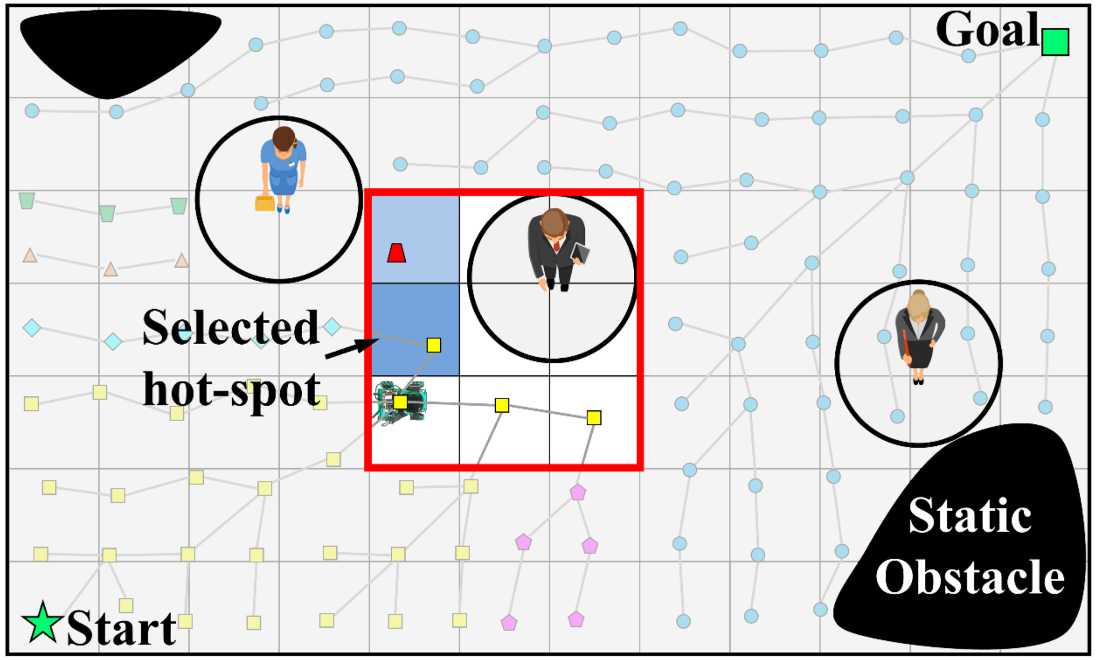


Fig. Hot-spots are color-coded with utilities and the highest one is selected.

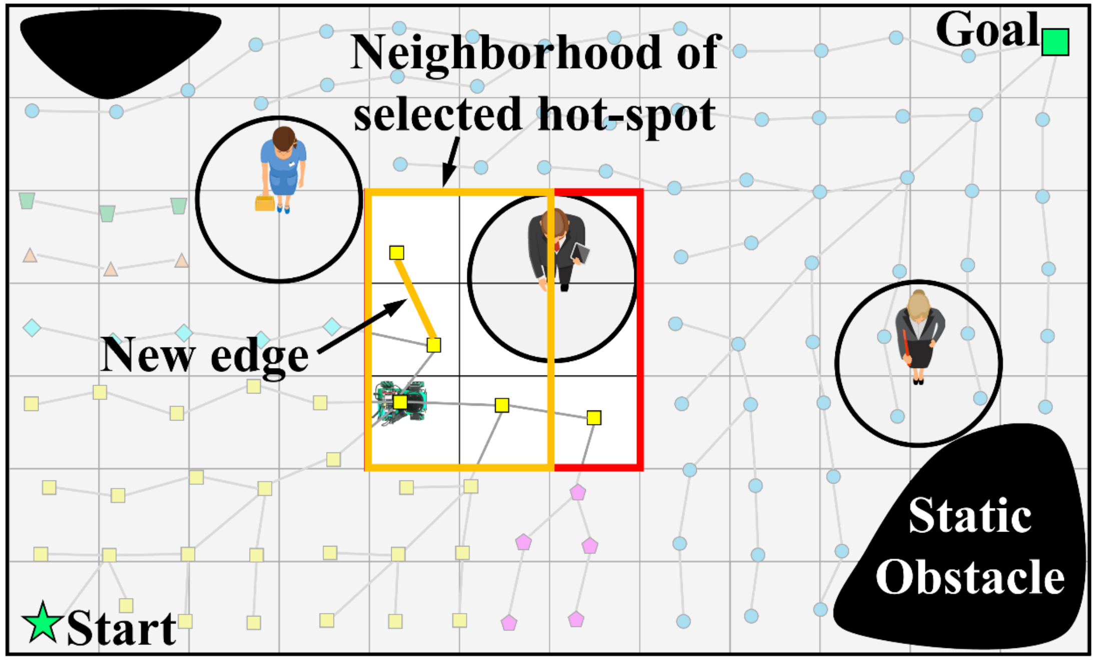


Fig. The tree is further repaired and the map is updated.

SMART Algorithm

Tree Reconnection

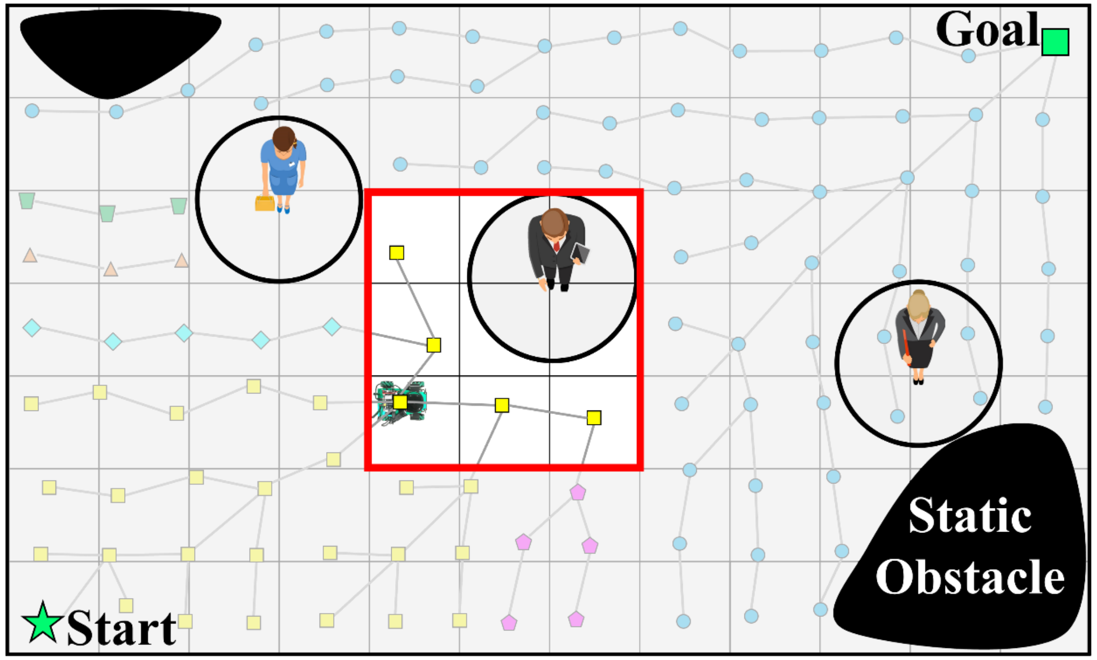


Fig. There is no more hot-spot in $\mathcal{S}^3$ and the goal-rooted subtree $\mathcal{T}_0$.

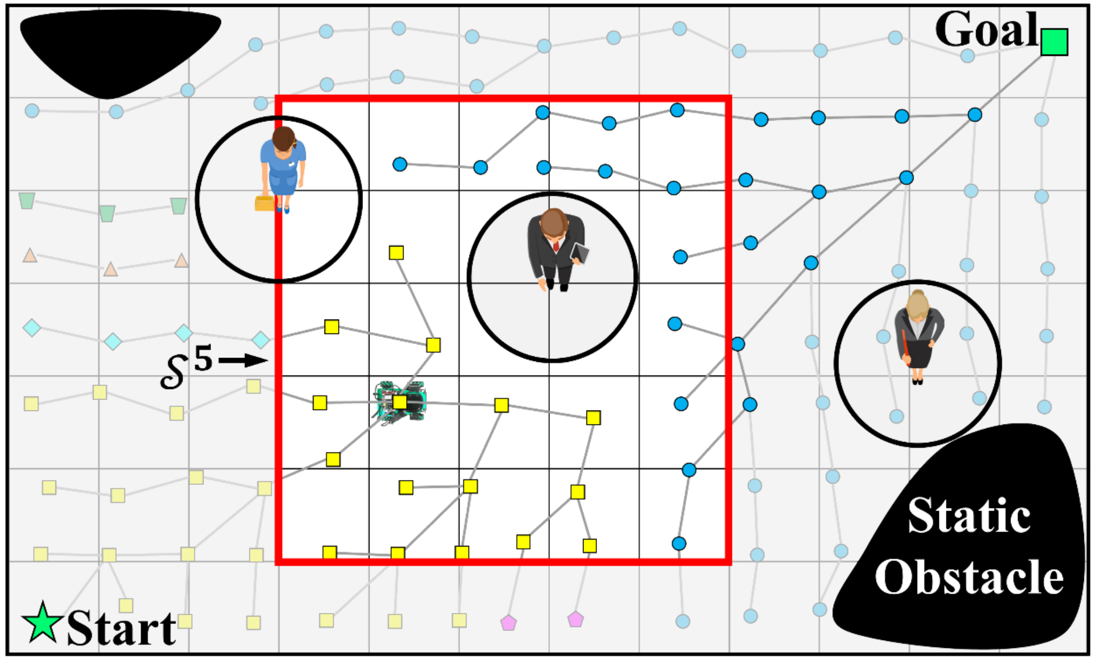


Fig. $\mathcal{S}^3$ is expanded to $\mathcal{S}^5$. The subtree nodes and are identified.

SMART Algorithm

Tree Reconnection

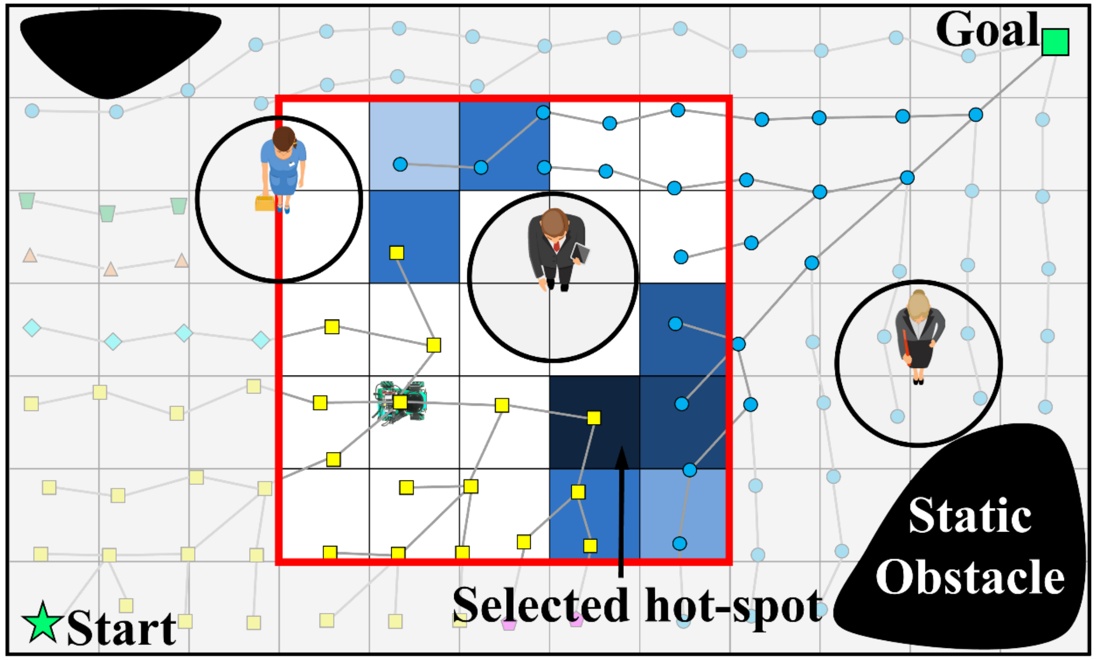


Fig. Hot-spots are color-coded with utilities and the highest one is selected.

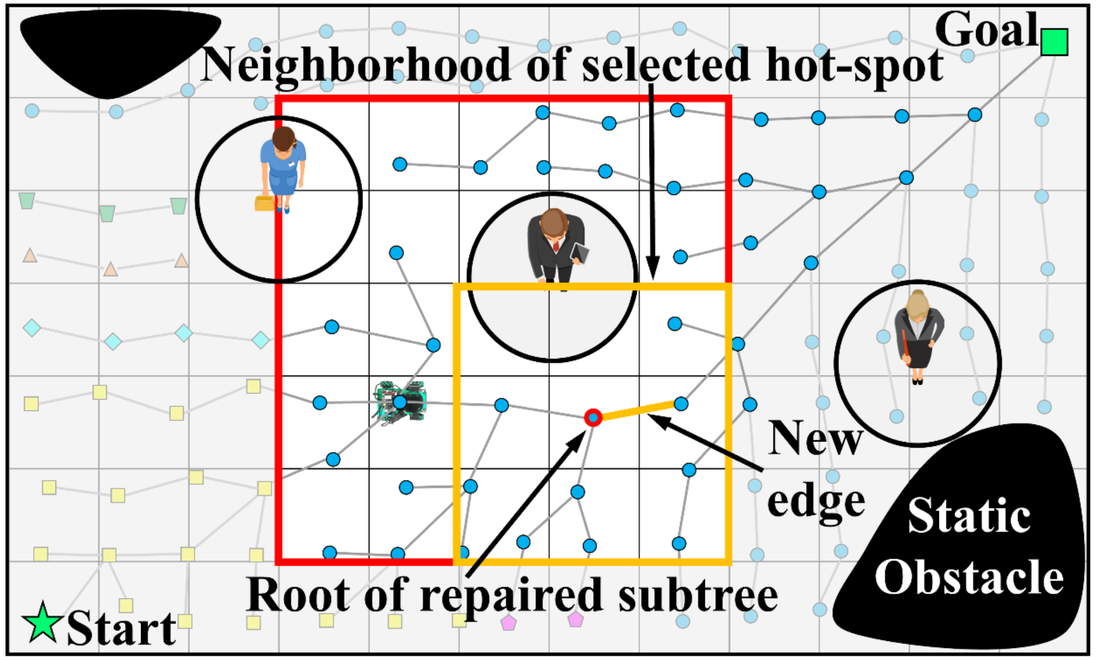


Fig. The tree is further repaired and the map is updated; the goal-rooted subtree $\mathcal{T}_0$ is connected.

SMART Algorithm

Tree Optimization and Path Search

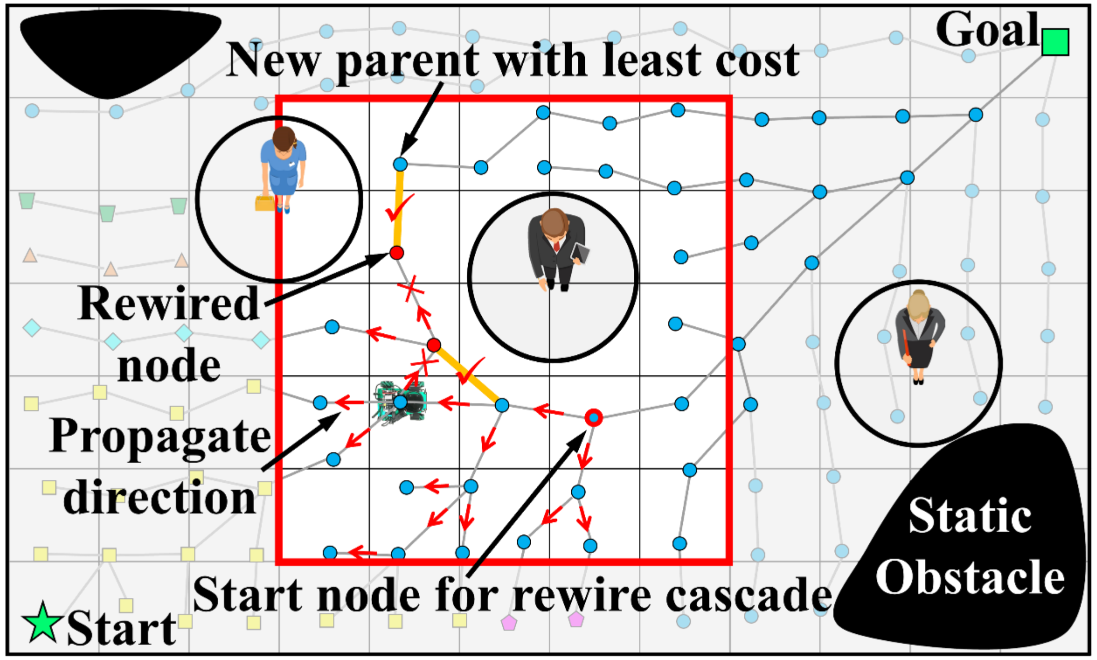


Fig. The repaired tree is optimized via a rewiring cascade, updating the cost-to-go of its nodes.

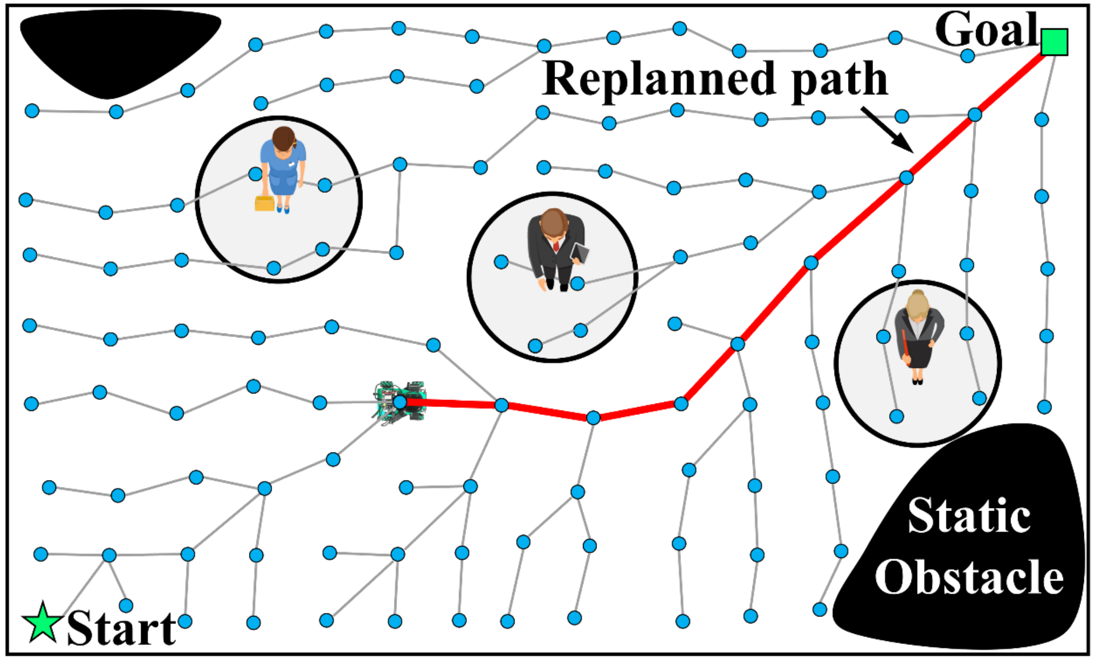


Fig. Finally, the path is replanned. Then, all pruned nodes are added back and any disjoint trees left are merged to form a single morphed tree $\mathcal{T}$.

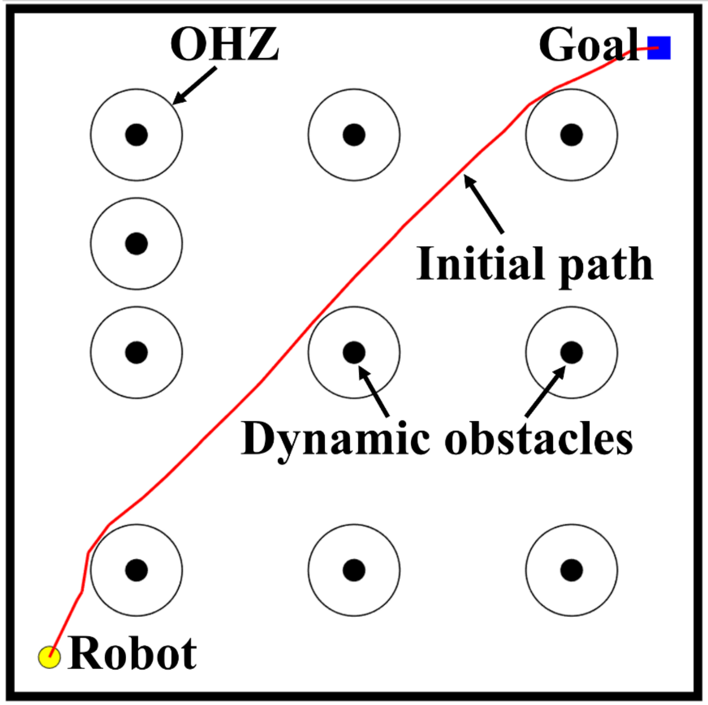
Simulation Validations

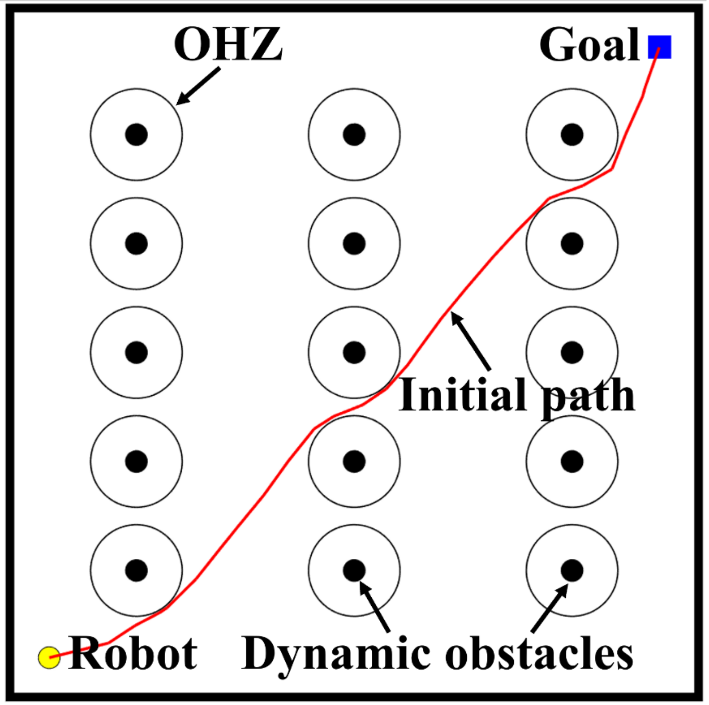
- **Holonomic robot:** it is with radius 0.5m that moves at a constant speed of 4m/s
- **Dynamic obstacles:** they are with radius 0.5m move along a random heading from $[0, 2\pi]$ for a random distance from $[0, 10m]$, after which a new heading and distance are selected.
- **Baseline algorithm:** D* Lite, ERRT, DRRT, MPRRT, RRT^X, HLRRT*, EBGRRT, MODRRT*
- **Parameters:**
- For SMART, the tiling is generated with cell size of $1m \times 1m$. The reaction time-horizon is set as 0.8s
- For all algorithms, an obstacle risk time-horizon of 0.4s is added to all dynamic obstacles. If the robot moves into an OHZ, then the OHZ is ignored but the actual obstacle is considered for collision checking.
- For the baseline algorithms, the goal bias and random sample rates shared by most algorithms are set as 0.1 and 0.2, respectively. All other algorithm-specific parameters are set based on the corresponding papers.
- **Performance metrics:**
- Replanning time: time to replan a new path
- Success rate: fraction of successful runs out of the total
- Travel time: time from start to goal without collision

Simulation Validations

❖ **Scenario 1:**

- it consists of a $32m \times 32m$ space populated with only dynamic obstacles
- Two cases are conducted including a) 10 and b) 15 obstacles. Each obstacle moves at the same speed selected from the set $\{1,2,3,4\}m/s$, resulting in 8 different combinations of the number of obstacles and speeds. For each combination, 30 different obstacle trajectories are generated, resulting in a total of 240 case studies. 100 trials are performed for each of cases for each algorithm.

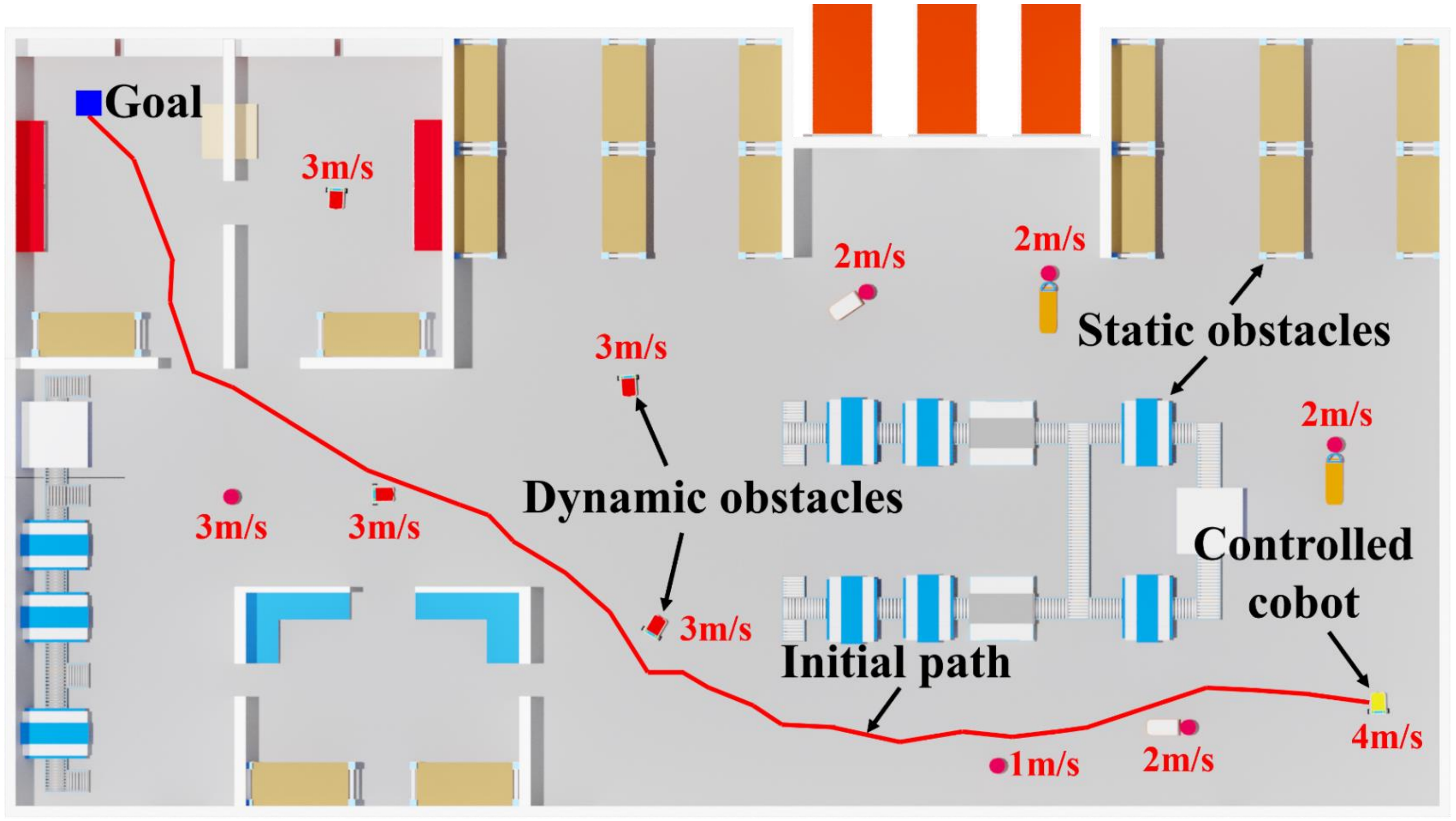




Simulation Validations

❖ **Scenario 2:**

- it consists of a $66m \times 38m$ space populated with a static obstacle layout and 10 dynamic obstacles
- Each obstacle moves at a different speed selected from the set $\{1,2,3,4\}m/s$. Then, 30 different obstacle trajectories are generated, resulting in a total of 30 case studies. 100 trials are performed for each of cases for each algorithm.



Simulation Validations

Scenario 1: Comparison with Baseline Algorithms

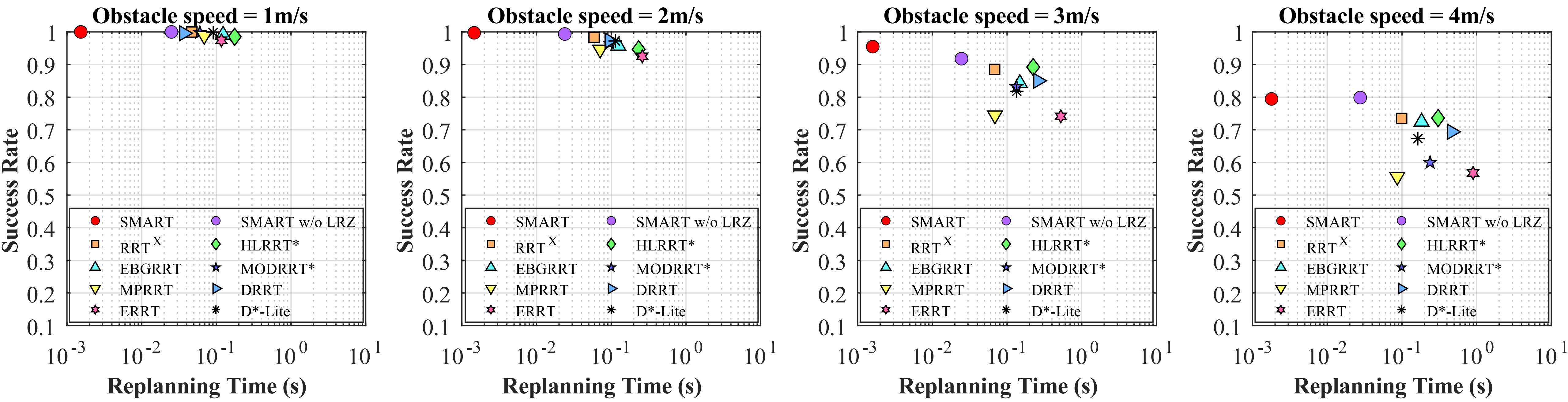


Fig. Success rate and average replanning time for Scenario 1 with 10 moving obstacles.

Simulation Validations

Scenario 1: Comparison with Baseline Algorithms

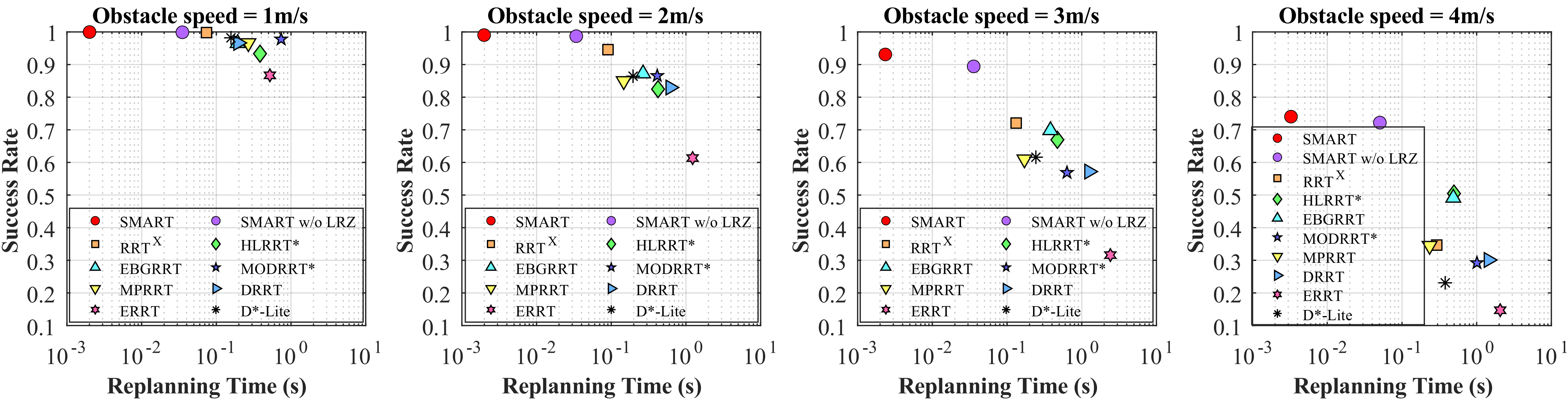
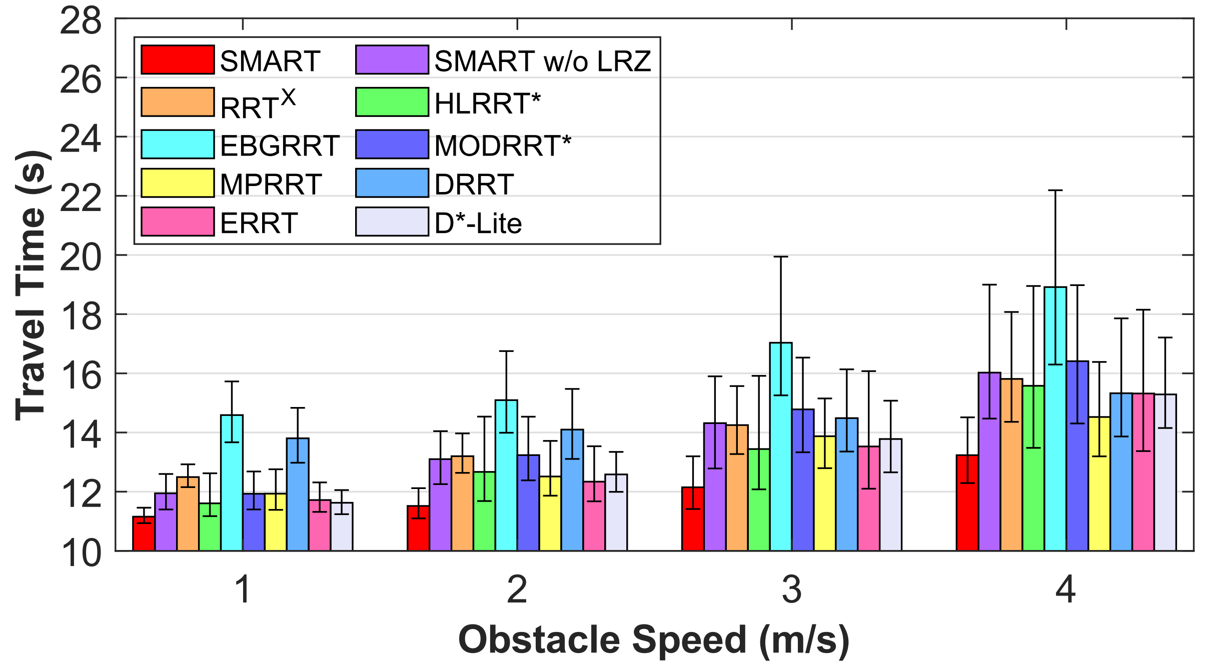


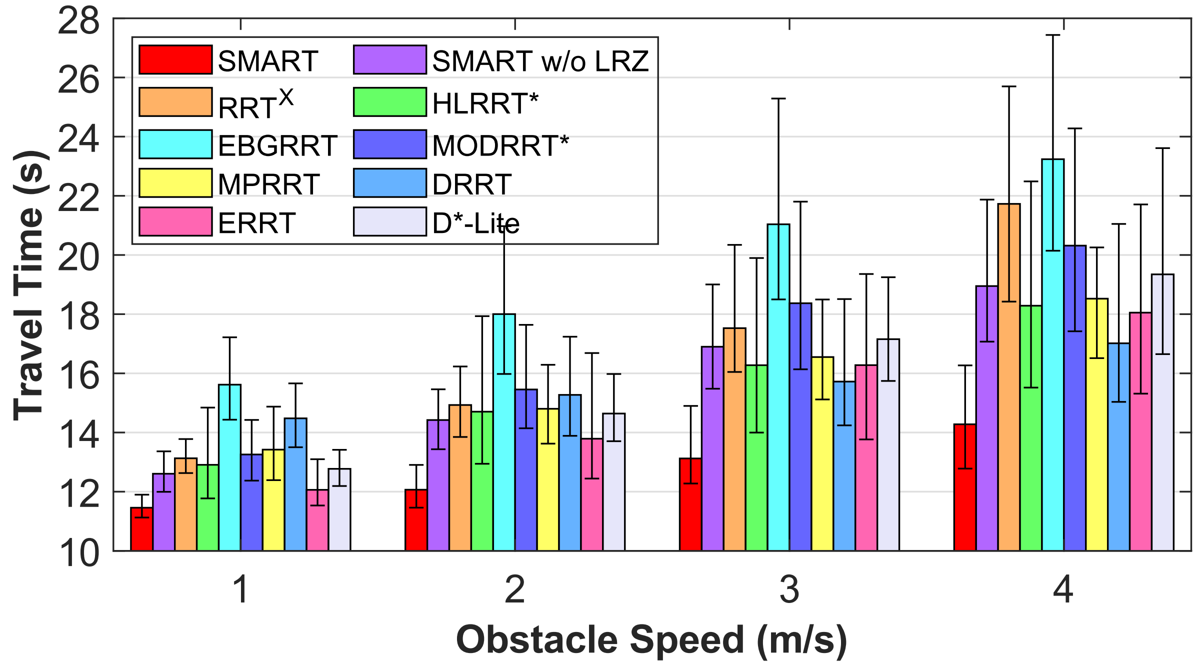
Fig. Success rate and average replanning time for Scenario 1 with 15 moving obstacles.

Simulation Validations

Scenario 1: Comparison with Baseline Algorithms



(a) Travel time for Scenario 1 with 10 moving obstacles.

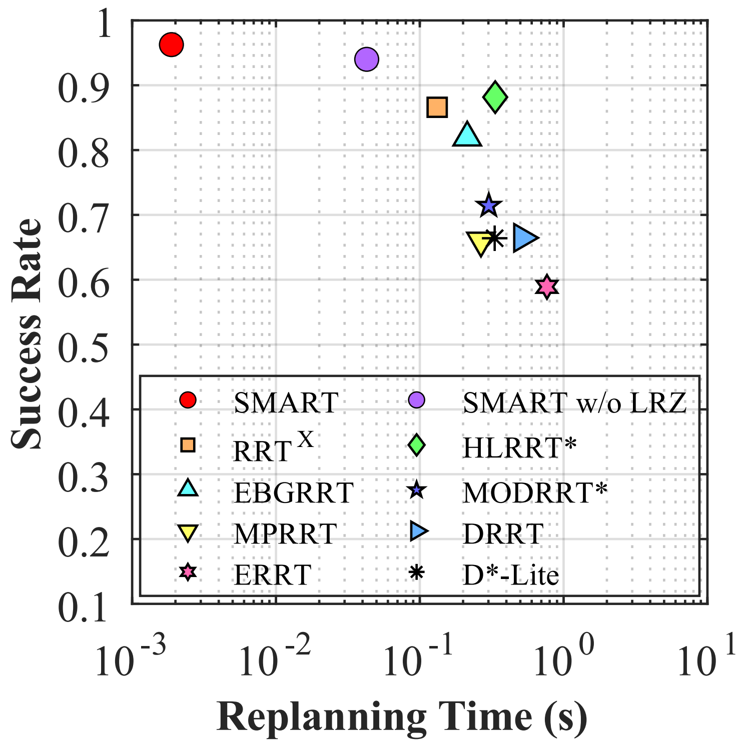


(b) Travel time for Scenario 1 with 15 moving obstacles.

Fig. Comparative evaluation results of travel time of successful trials for Scenario 1 with a) 10 and b) 15 moving obstacles. The plots show the median and the 25th and 75th percentile values.

Simulation Validations

Scenario 2: Comparison with Baseline Algorithms



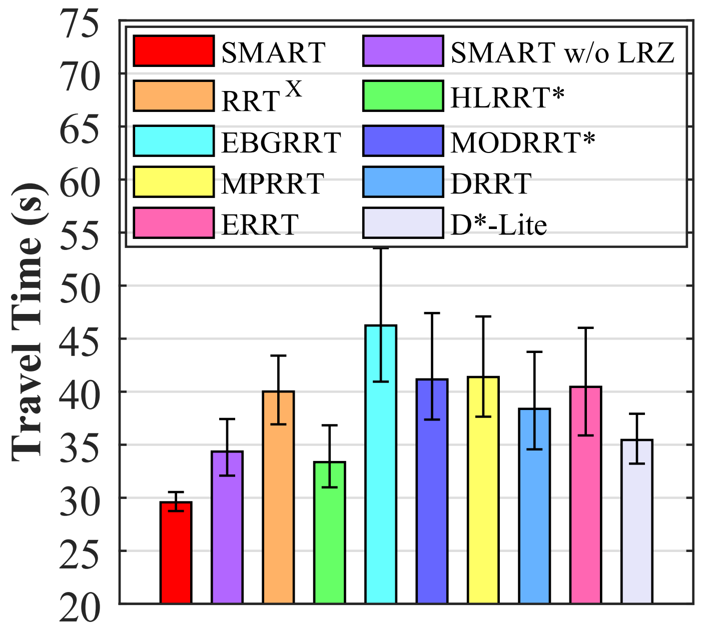
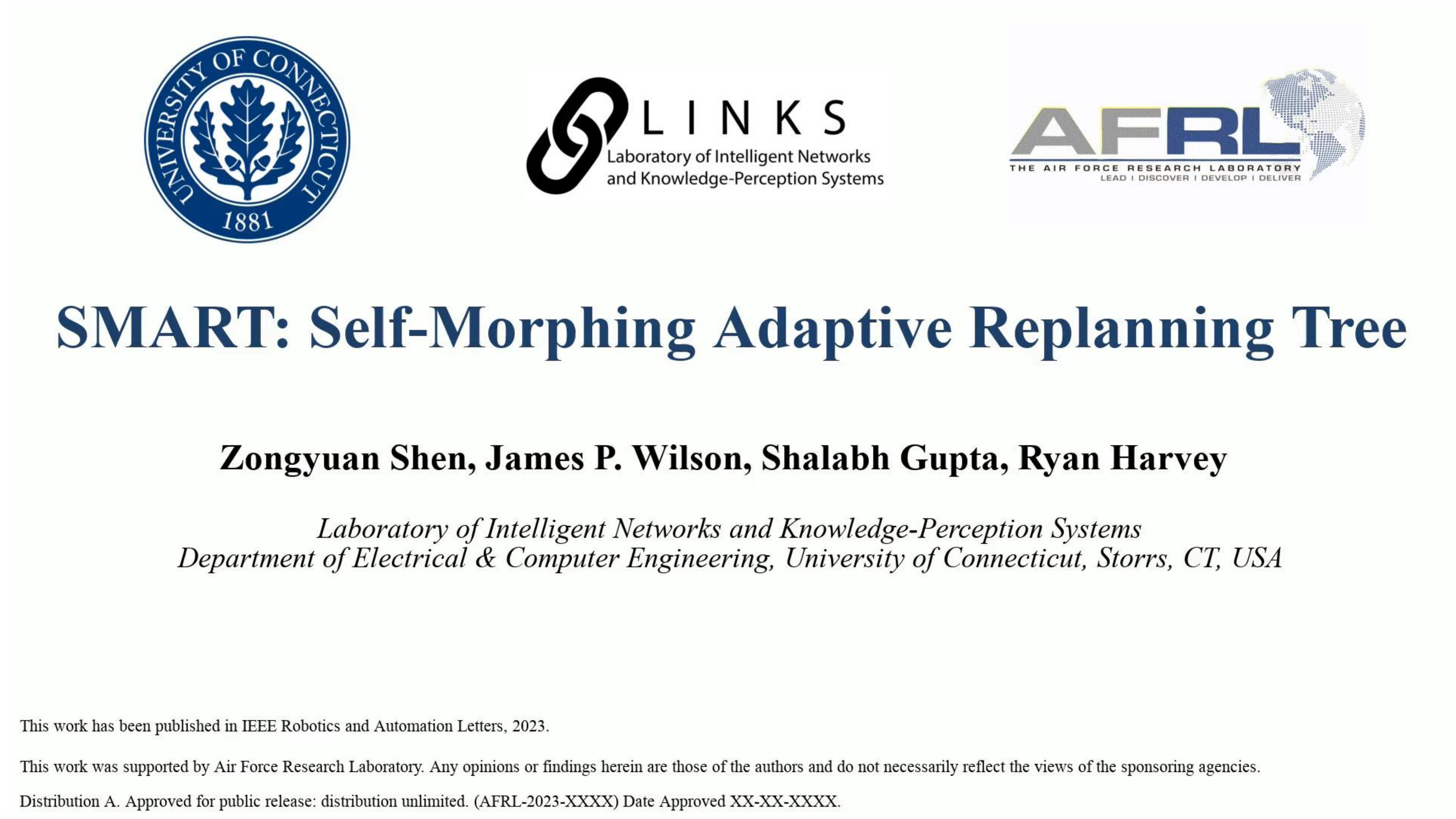


Fig. Comparative evaluation results for Scenario 2.

Simulation Validations

Demos of SMART Algorithm in Scenario 1 and 2



SMART: Self-Morphing Adaptive Replanning Tree

Zongyuan Shen, James P. Wilson, Shalabh Gupta, Ryan Harvey

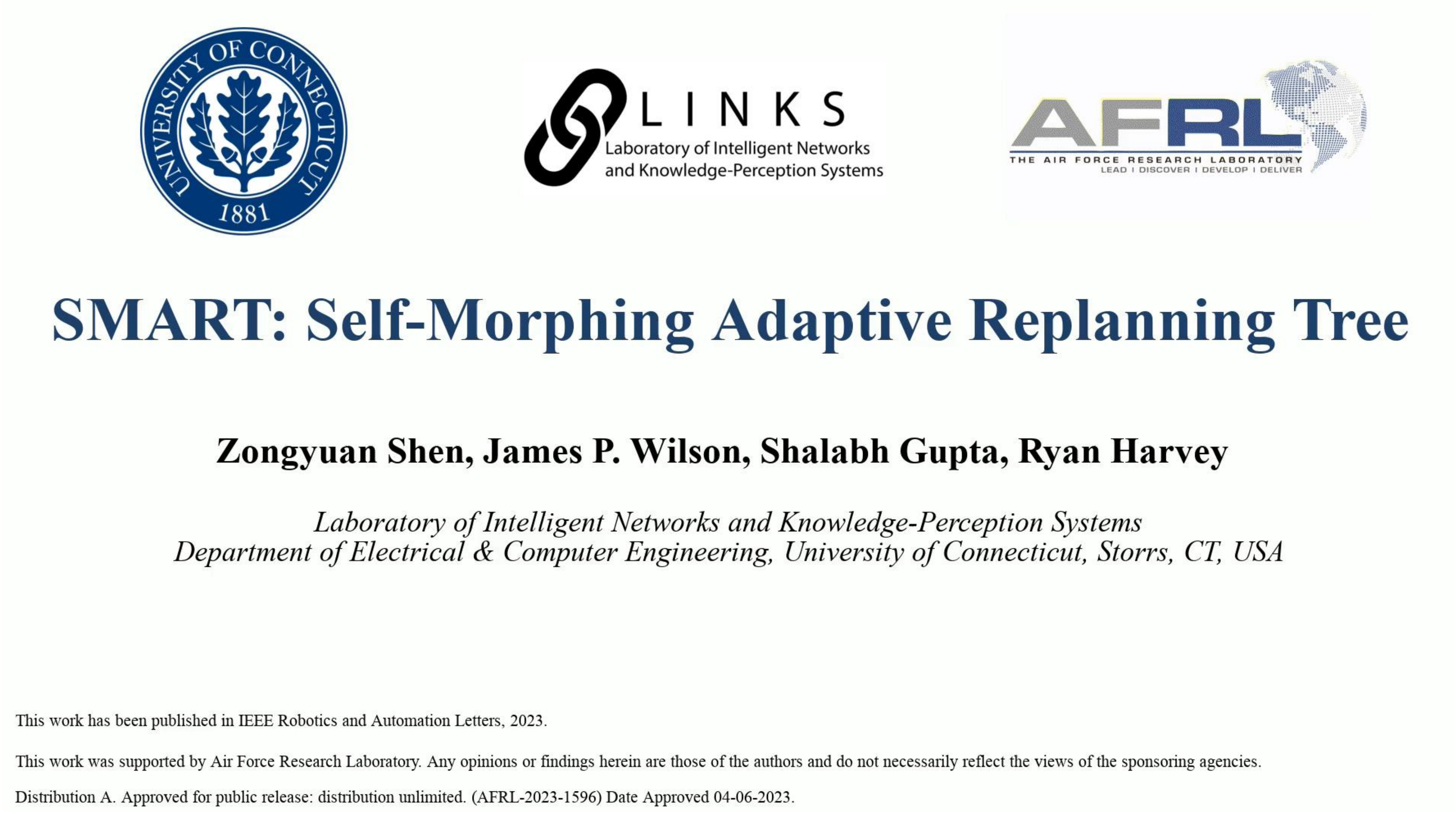
Laboratory of Intelligent Networks and Knowledge-Perception Systems
Department of Electrical & Computer Engineering, University of Connecticut, Storrs, CT, USA

This work has been published in IEEE Robotics and Automation Letters, 2023.

This work was supported by Air Force Research Laboratory. Any opinions or findings herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

Real Experiments

- The SMART algorithm is validated by real experiments in a $7m \times 7m$ lab space with both static and dynamic obstacles using a robot called ROSMASTER X3 which is programmable and controllable.



SMART: Self-Morphing Adaptive Replanning Tree

Zongyuan Shen, James P. Wilson, Shalabh Gupta, Ryan Harvey

Laboratory of Intelligent Networks and Knowledge-Perception Systems
Department of Electrical & Computer Engineering, University of Connecticut, Storrs, CT, USA

This work has been published in IEEE Robotics and Automation Letters, 2023.

This work was supported by Air Force Research Laboratory. Any opinions or findings herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

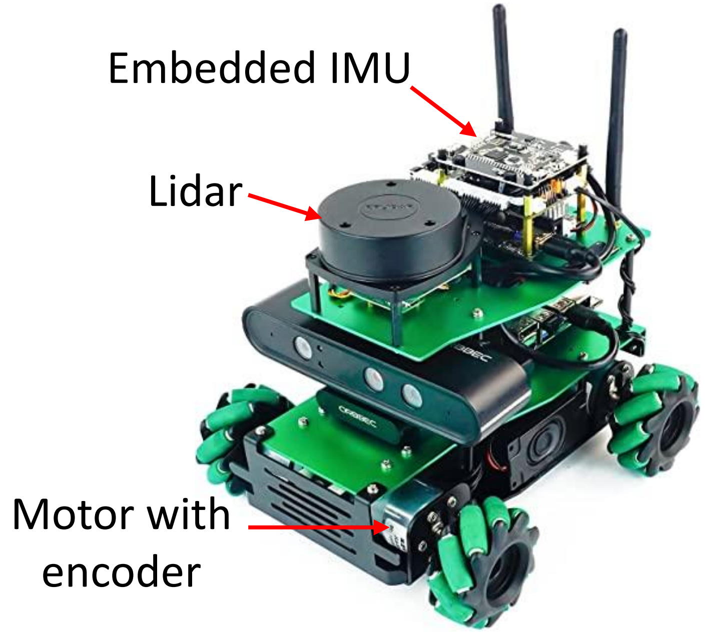


Fig. ROSMASTER X3 integrated with multiple sensing devices