$\epsilon^{\star}$: An Online Coverage Path Planning Algorithm

This work has been published in:
J. Song and S. Gupta, "$\epsilon^{\star}$: An Online Coverage Path Planning Algorithm", ***IEEE Transactions on Robotics***, Vol. 34, Issue 2, pp 526-533, 2018.

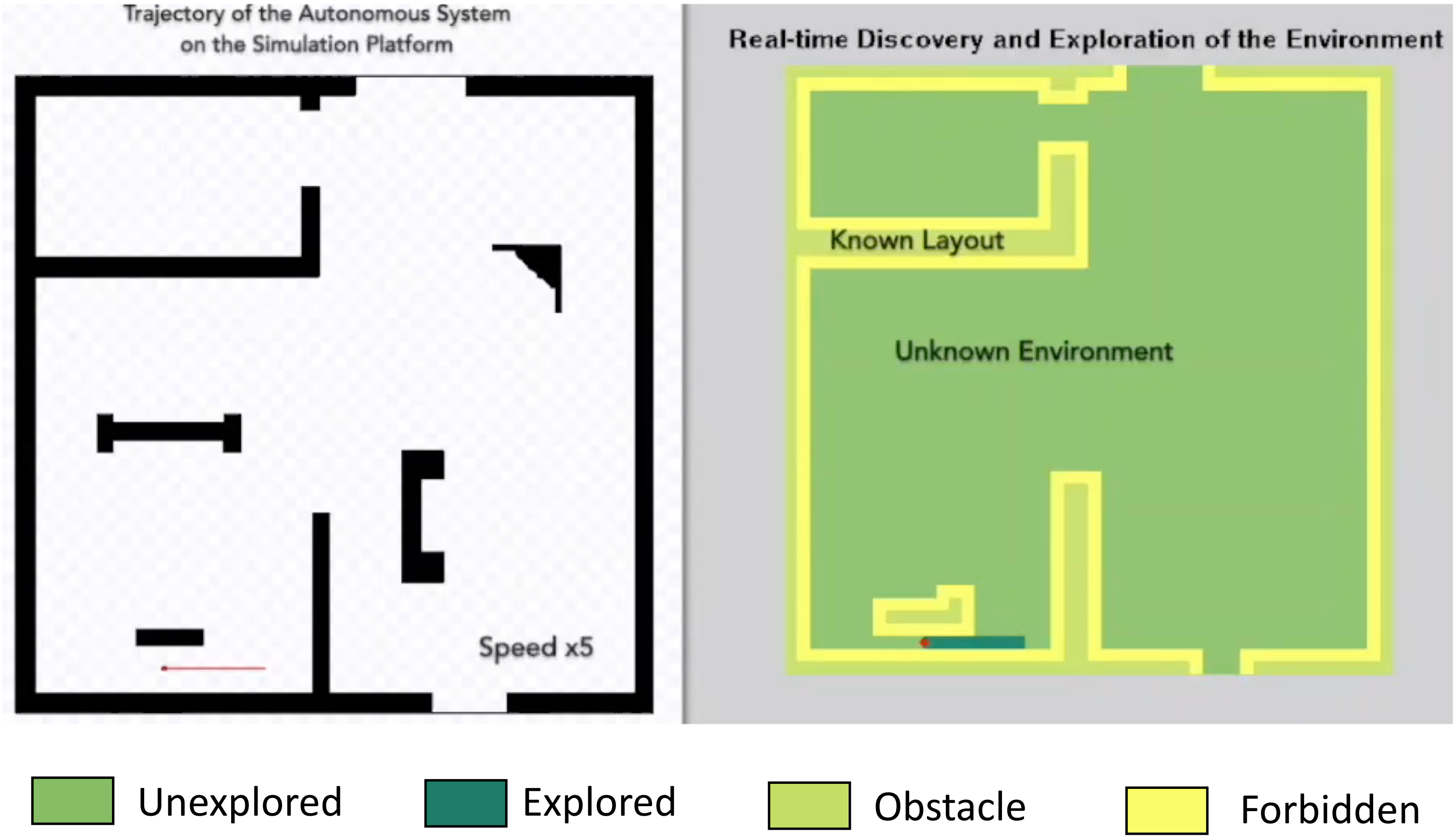
Videos Available: These slides contain videos that are accessible online at: https://linkslab.uconn.edu/videos/

$\epsilon^{\star}$: Online Coverage Path Planning in Unknown Environment

❖ **Objective:** Develop an online coverage path planning algorithm for an autonomous vehicle in unknown environment

❖ **Challenges:**

- Online detection and avoidance of unknown obstacles
- Generate back-and-forth path with minimized turns and overlappings
- Must guarantee complete coverage and prevent any local extremum
- Low computational complexity for real-world applications



Coverage Path Planning Algorithms

State-of-the-art and Novel Contributions

❖ Existing Approaches:

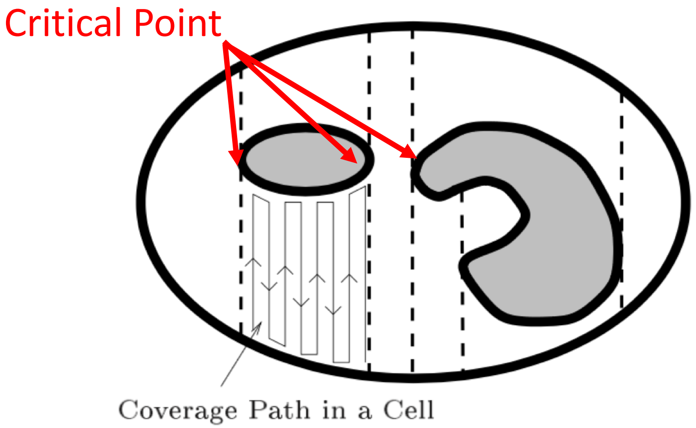
- Learning Real-time A* (LRTA*) ← *Strong path overlappings*
- Spanning-tree Coverage
- Backtracking Spiral Algorithm
- Brick-and-Mortar Algorithm

} *Generate spiral path with too many turns*

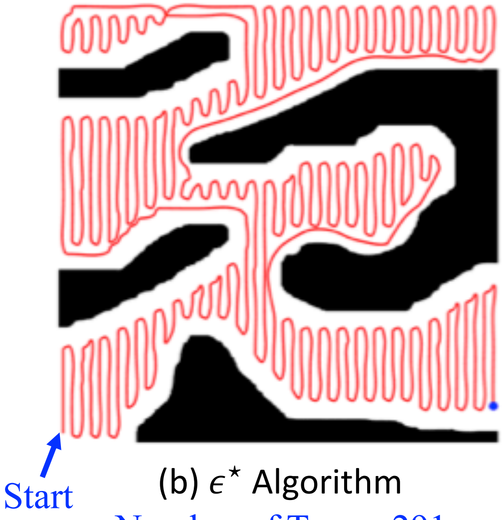
- Cellular Decomposition (*back-and-forth* path)
  - ➢ Rely on detection of critical points (detection and pairing of IN & OUT critical points are difficult in complex environment)
  - ➢ Require cycle algorithm which leads to overlappings
  - ➢ Cannot work in rectilinear environment

❖ Features and Novel Contributions of the $\epsilon^{\star}$ Algorithm:

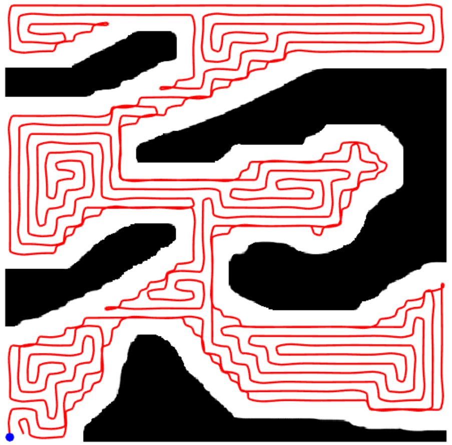
- Produces the desired *back-and-forth* path
- **Does not need critical point detection on obstacles**
- Guarantees complete coverage and prevents the local extrema problem using hierarchical potential surfaces (called MAPS)
- Capable of adapting sweep direction in known sub-regions to further reduce the number of turns
- Computationally efficient for real-time applications



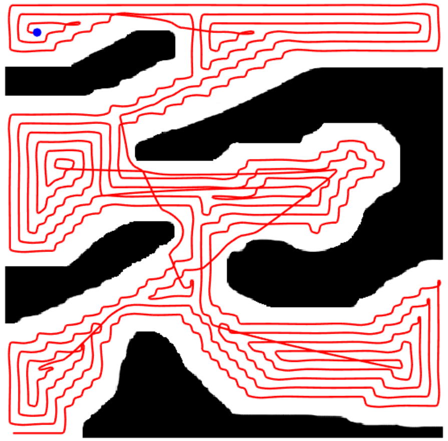
(a) Cellular Decomposition based Method[1]



(b) $\epsilon^{\star}$ Algorithm
Number of Turns: 291



(c) Spanning Tree Coverage
Number of Turns: 348



(d) Backtracking Spiral Algorithm
Number of Turns: 407

[1] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar and D. Hull, 2002. Morse decompositions for coverage tasks. The international journal of robotics research, 21(4), pp.331-344.

$\epsilon^\star$ Algorithm

Tiling of the Search Area

- Let $\mathcal{R} \subset \mathbb{R}^2$ be the estimated area that includes the desired area to cover.

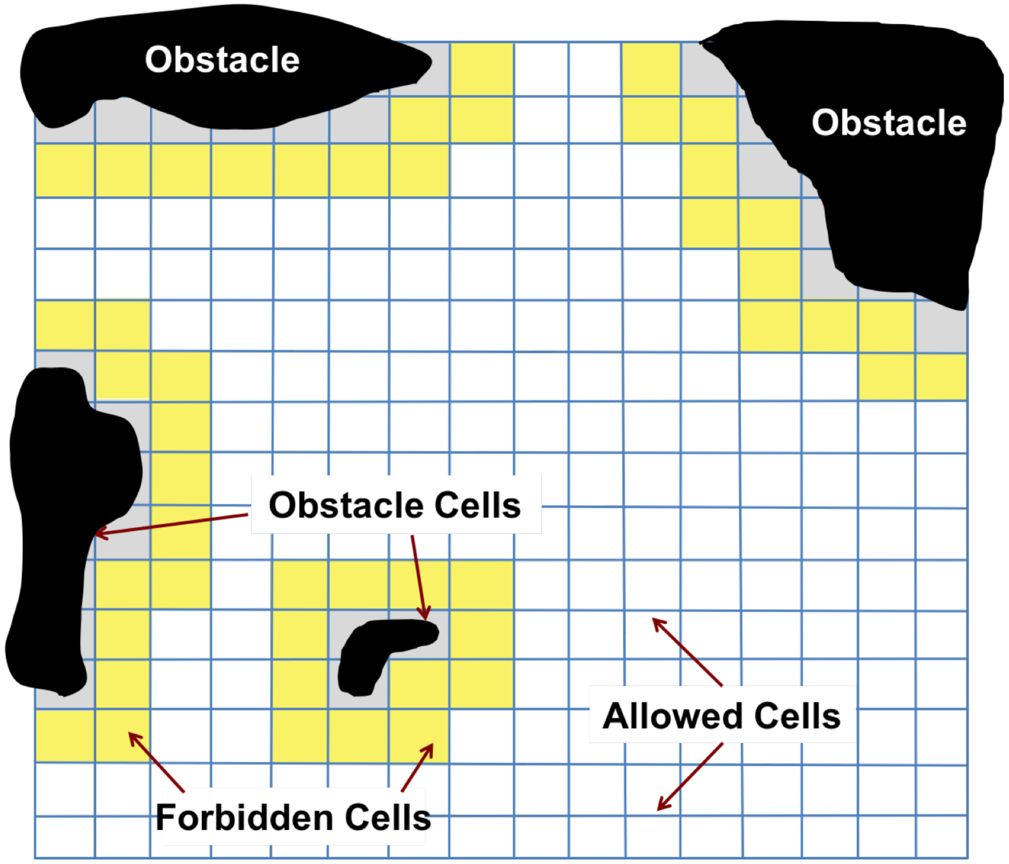
Tiling: The set $T = \{\tau_\alpha \subset \mathbb{R}^2 : \alpha = 1 \ldots |T|\}$ is called a tiling of $\mathcal{R}$ if its elements:

i) have mutually exclusive interiors, i.e., $\tau_\alpha^o \cap \tau_\beta^o = \emptyset, \forall \alpha \neq \beta$, where $\alpha, \beta \in \{1 \ldots |T|\}$.

ii) form a minimal cover, i.e., $\mathcal{R} \subseteq \cup_{\alpha=1}^{|T|} \tau_\alpha$, while removal of any tile destroys the covering property.

ϵ Cell: Each element $\tau_\alpha, \forall \alpha \in \{1, \ldots |T|\}$, is called an ϵ-cell.

- The tiling T is partitioned into three subsets:
  - ***Obstacle cells*** (T^o): they are detected online.
  - ***Forbidden cells*** (T^f): create buffer around obstacles
  - ***Allowed cells*** (T^a): these are the target cells to cover



Tiling of the Search Area

$\epsilon^\star$ Algorithm

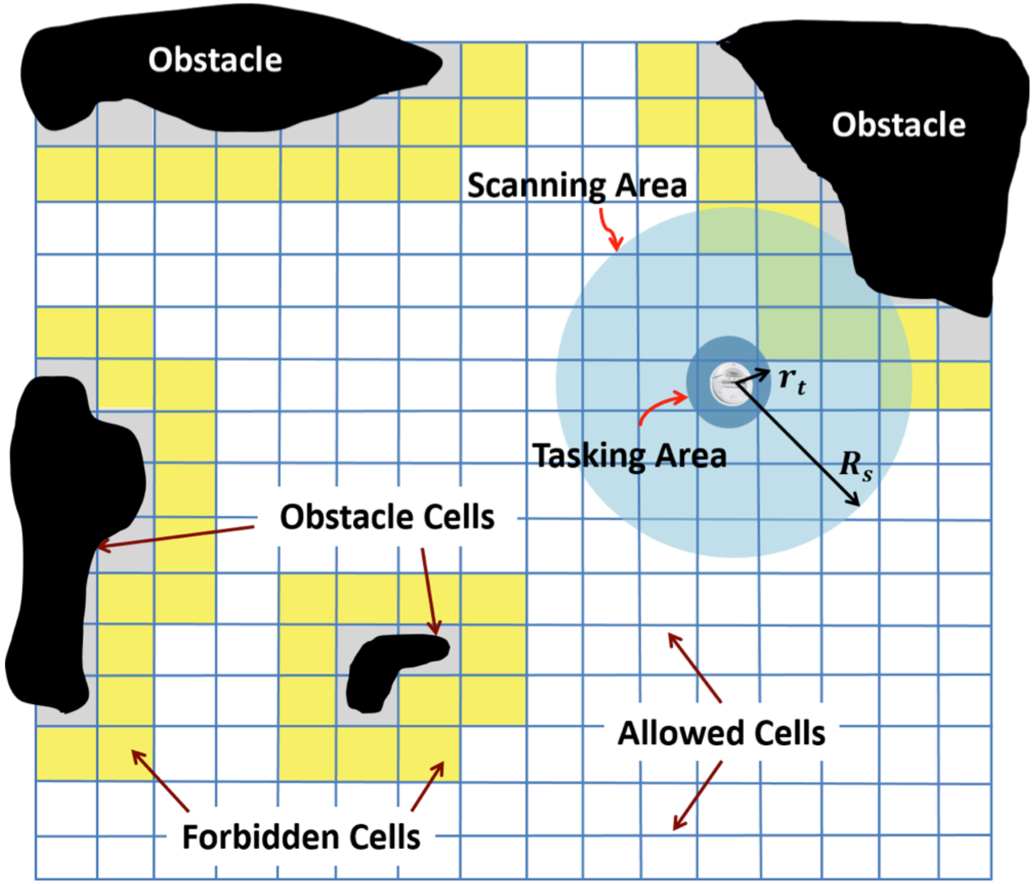
The Autonomous Vehicle and ϵ-Coverage

❖ **The autonomous vehicle is equipped with:**

1. Localization System
   - Provides vehicle location (e.g., GPS), and heading (e.g., Compass)
2. Range Detector with Sensing Radius R_s
   - Allows the vehicle to detect obstacles in the local neighborhood (e.g., laser)
3. Tasking Sensor with Radius r_t
   - Allows the vehicle to carry out certain tasks (e.g., cleaning, target detection, crops cutting) while it operates in the field

ϵ-Coverage Let $\mathcal{R}(T^a)$ denote the total area of the allowed cells. Let $\tau(k) \in T^a$ be the ϵ-cell visited by the autonomous vehicle at time k and explored by its tasking sensor. Then, $\mathcal{R}$ is said to achieve ϵ-coverage, if $\exists K \in \mathbb{Z}^+$, such that the sequence $\{\tau(k), k = 1, \dots, K\}$ covers $\mathcal{R}(T^a)$, i.e.,

$$\boxed{\mathcal{R}(T^a) \subseteq \cup_{k=1}^{K} \tau(k)}$$



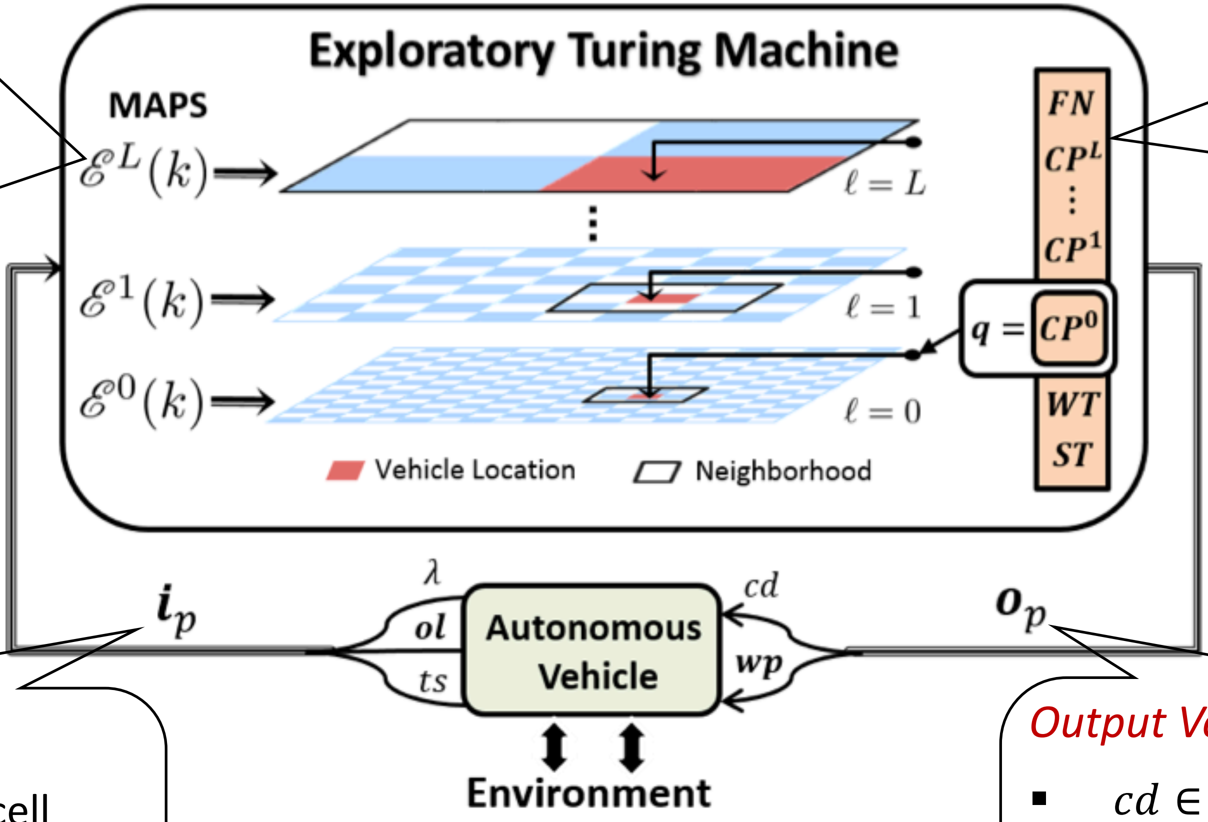
The autonomous vehicle and the tiling

$\epsilon^\star$ Algorithm

The Supervisory Controller: Exploratory Turing Machine (ETM)

Multi-scale Adaptive Potential Surfaces(MAPS)

- $\mathcal{E}^0$: time-varying potential surface at the finest level
- $\mathcal{E}^\ell, 1 \leq \ell \leq L$: time-varying potential surfaces at higher levels



Machine States:

- ST ≡ Start
- CP ≡ Compute
- WT ≡ Waiting
- FN ≡ Finished

Input Vector $\boldsymbol{i_p}$

- $\lambda \in \{1, \ldots |T|\}$: index of current cell
- $\boldsymbol{ol} \subset \{1, \ldots |T|\}$, indices of obstacle cells
- $ts \in \{cm, ic\}$, tasking status, where cm ≡ *Complete*, or ic ≡ *Incomplete*.

Output Vector $\boldsymbol{o_p}$

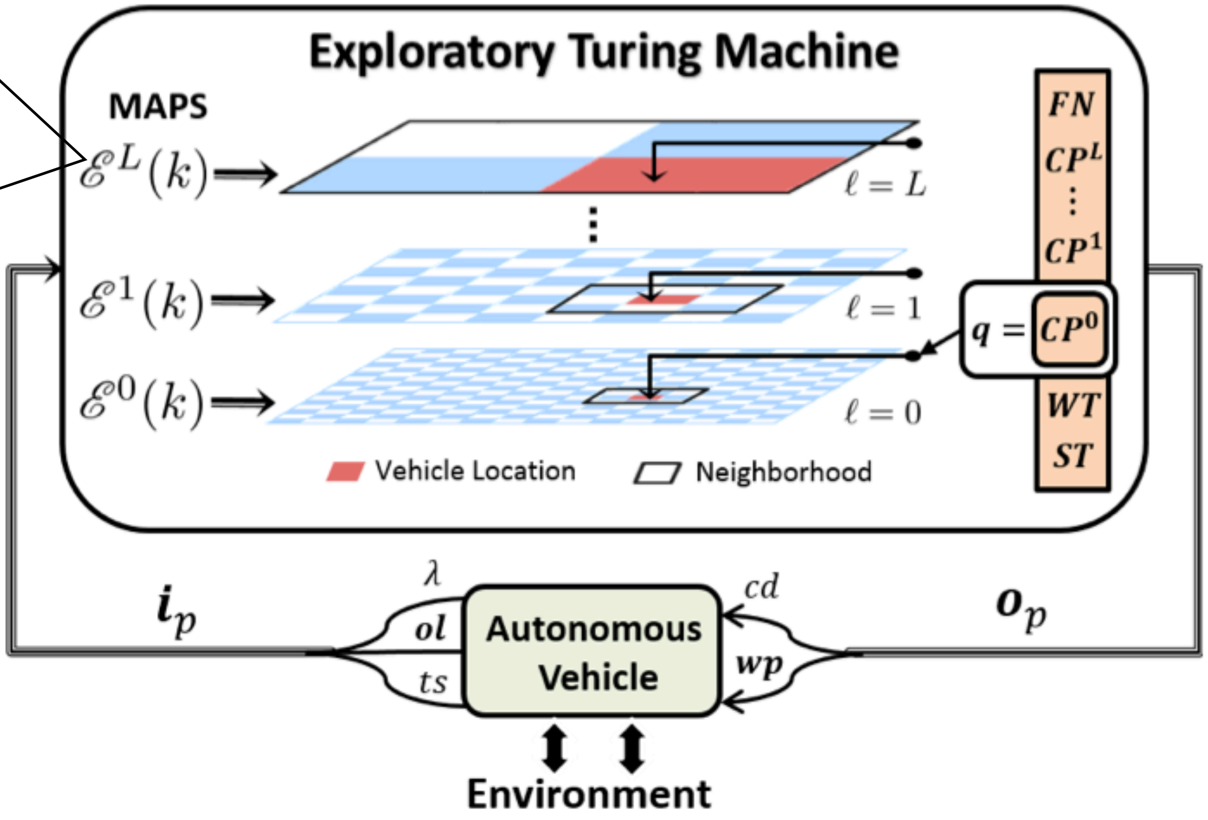
- $cd \in \{mv, tk, id, sp\}$: command to the vehicle, where: mv ≡ Move, tk ≡ Task, id ≡ Idle, and sp ≡ Stop.
- $\boldsymbol{wp} \subset \{1, \ldots |T|\}$, candidate set of new waypoints for the vehicle

$\epsilon^{\star}$ Algorithm

The Supervisory Controller: Exploratory Turing Machine (ETM)

Multi-scale Adaptive Potential Surfaces(MAPS)

- $\mathcal{E}^0$: time-varying potential surface at the finest level
- $\mathcal{E}^\ell, 1 \leq \ell \leq L$: time-varying potential surfaces at higher levels



$\epsilon^\star$ Algorithm

Dynamically Constructed Multi-scale Potential Surfaces (MAPS)

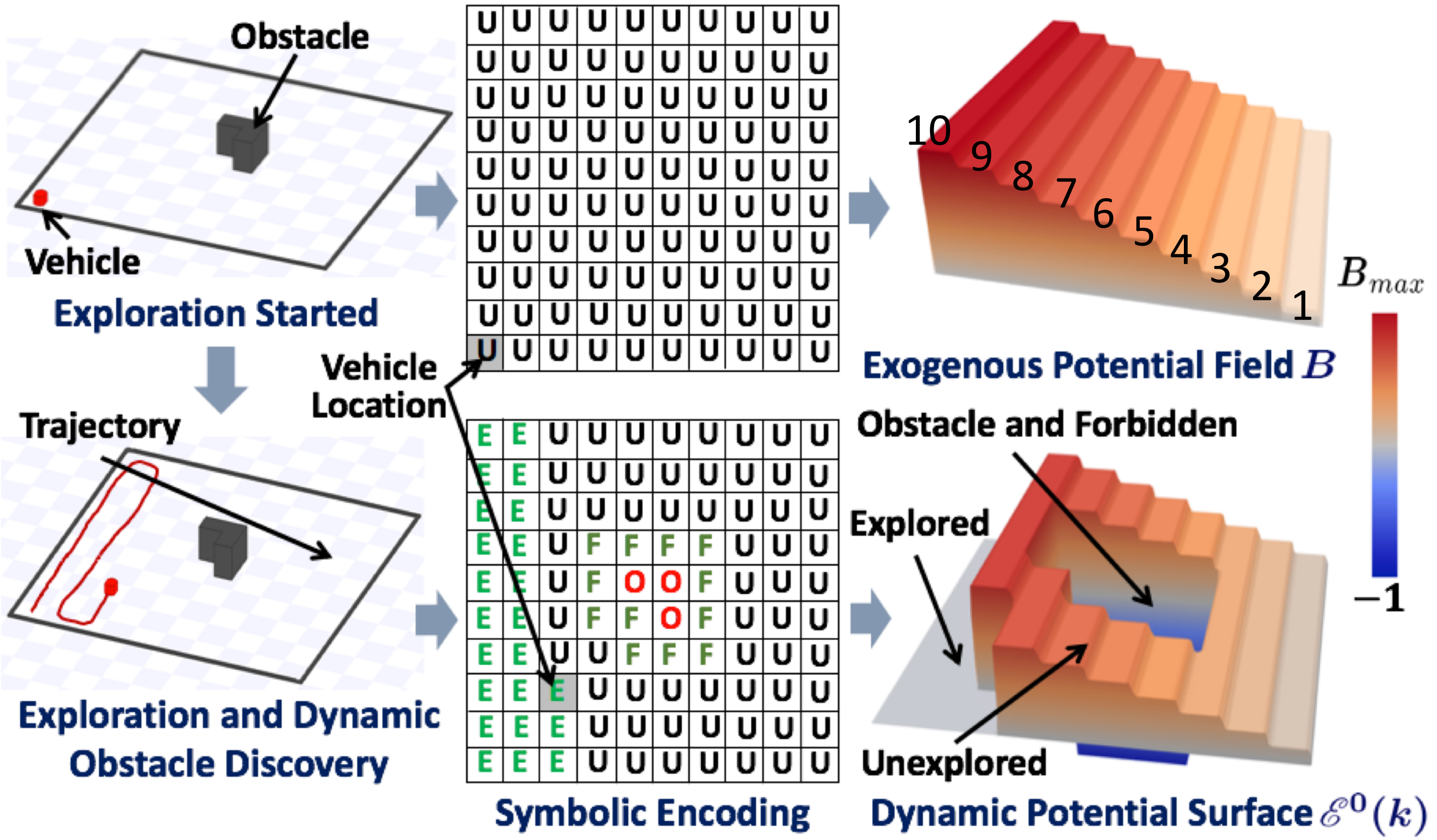
❖ Level 0 of MAPS

- ***Symbolic Encoding***: each ϵ-cell at level 0, τ_{α^0}, is assigned with a symbolic state s_{α^0}, from below:
  - O: *Obstacle*
  - F: *Forbidden*
  - E: *Explored*
  - U: *Unexplored*

  (E and U:) Allowed cells
- ***Potential Surface*** $\mathcal{E}^{\mathbf{0}}$:

$$\boxed{\mathcal{E}_{\alpha^0}(k) = \begin{cases} -1 & \text{if } s_{\alpha^0} = O \text{ or } F \\ 0 & \text{if } s_{\alpha^0} = E \\ B_{\alpha^0} & \text{if } s_{\alpha^0} = U \end{cases}}$$

where $B = \{B_{\alpha^0} \in \{1, \ldots, B_{\max}\}, \alpha^0 = 1, \ldots |T^0|\}$ is a time-invariant exogenous potential field. It is designed *offline* to have plateaus of equipotential surfaces along each column of the tiling.



$\epsilon^{\star}$ Algorithm

Dynamically Constructed Multi-scale Potential Surfaces (MAPS)

Note: Higher levels of MAPS are used to prevent the local extrema problem.

Local Extrema: no unexplored cells are available in the local neighborhood on Level 0.

❖ Levels $\ell = 1, 2, \dots L$ of MAPS

- ***Potential Surfaces*** $\mathcal{E}^{\ell}$, $\ell = 1, \dots L$, are constructed by assigning τ_{α^ℓ} the *average* potential generated by all the *unexplored* ϵ-cells within τ_{α^ℓ}, such that

$$\boxed{\mathcal{E}_{\alpha^\ell}(k) = p^{U}_{\alpha^\ell}(k) \cdot \bar{B}_{\alpha^\ell}}$$

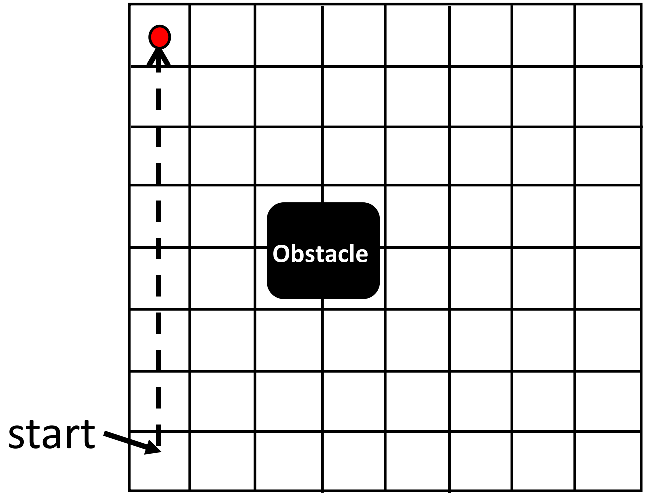
where $\bar{B}_{\alpha^\ell}$ is the mean exogenous potential of τ_{α^ℓ}, and $p^{U}_{\alpha^\ell}(k)$ is the probability of *unexplored* ϵ-cells in τ_{α^ℓ}.

$\epsilon^\star$ Algorithm

An Illustrative Example: Updates of MAPS at Level 0

❖ *An Illustrative Example*

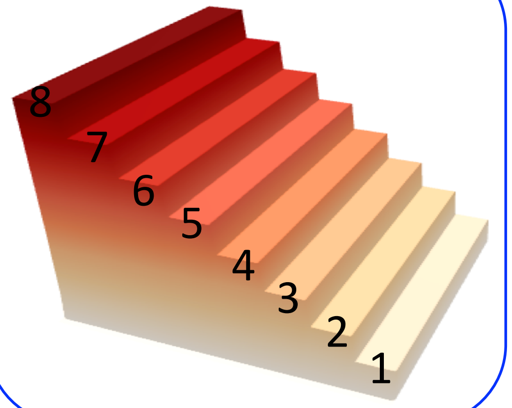
Vehicle trajectory and obstacle layout



Potentials B at Level 0

8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1

Visualization of B



Symbolic encodings at Level 0

E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U
E	F	F	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	F	F	U	U	U	U
E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U

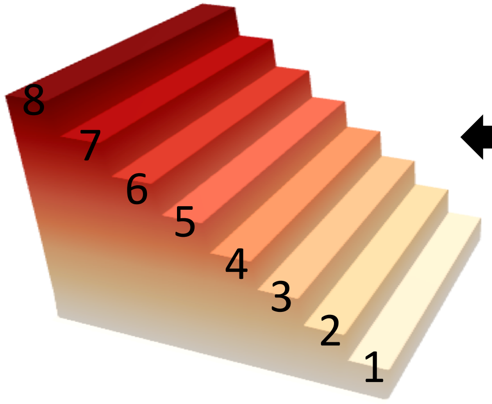
MAPS update on Level 0

0	7	6	5	4	3	2	1
0	7	6	5	4	3	2	1
0	-1	-1	-1	4	3	2	1
0	-1	-1	-1	4	3	2	1
0	-1	-1	-1	4	3	2	1
0	-1	-1	-1	4	3	2	1
0	7	6	5	4	3	2	1
0	7	6	5	4	3	2	1

$\epsilon^\star$ Algorithm

An Illustrative Example: Updates of MAPS at Level 1

Visualization of B



Potentials B at Level 0

8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1

Symbolic encodings at Level 0

E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U
E	F	F	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	F	F	U	U	U	U
E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U

MAPS update on Level 1

Average potential $\bar{B}$

7.5	5.5	3.5	1.5
7.5	5.5	3.5	1.5
7.5	5.5	3.5	1.5
7.5	5.5	3.5	1.5

×

Update probabilities $p^U_{\alpha^\ell}$

0.5	1	1	1
0	0	1	1
0	0	1	1
0.5	1	1	1

⇨

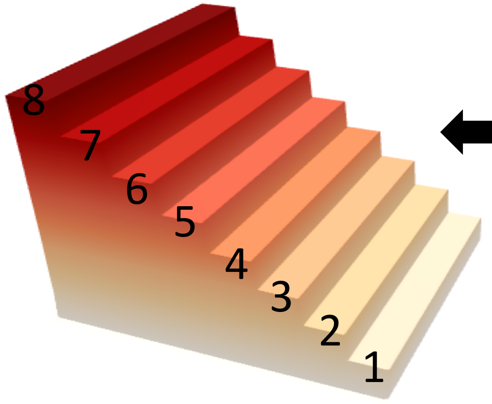
MAPS update

3.75	5.5	3.5	1.5
0	0	3.5	1.5
0	0	3.5	1.5
3.75	5.5	3.5	1.5

$\epsilon^{\star}$ Algorithm

An Illustrative Example: Updates of MAPS at Level 2

Visualization of B



Potentials B at Level 0

8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1
8	7	6	5	4	3	2	1

Symbolic encodings at Level 0

E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U
E	F	F	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	O	F	U	U	U	U
E	F	F	F	U	U	U	U
E	U	U	U	U	U	U	U
E	U	U	U	U	U	U	U

MAPS update on Level 2

Average potential $\bar{B}$

6.5	2.5
6.5	2.5

×

Update probabilities $p^{U}_{\alpha^{\ell}}$

0.375	1
0.375	1

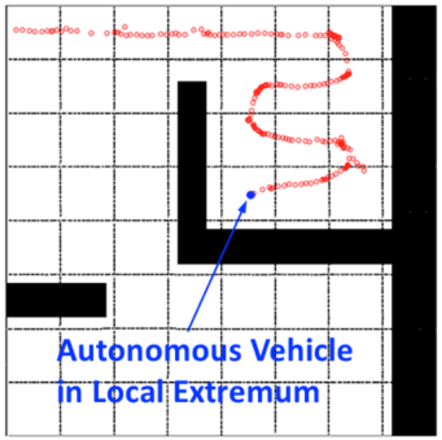
⇨

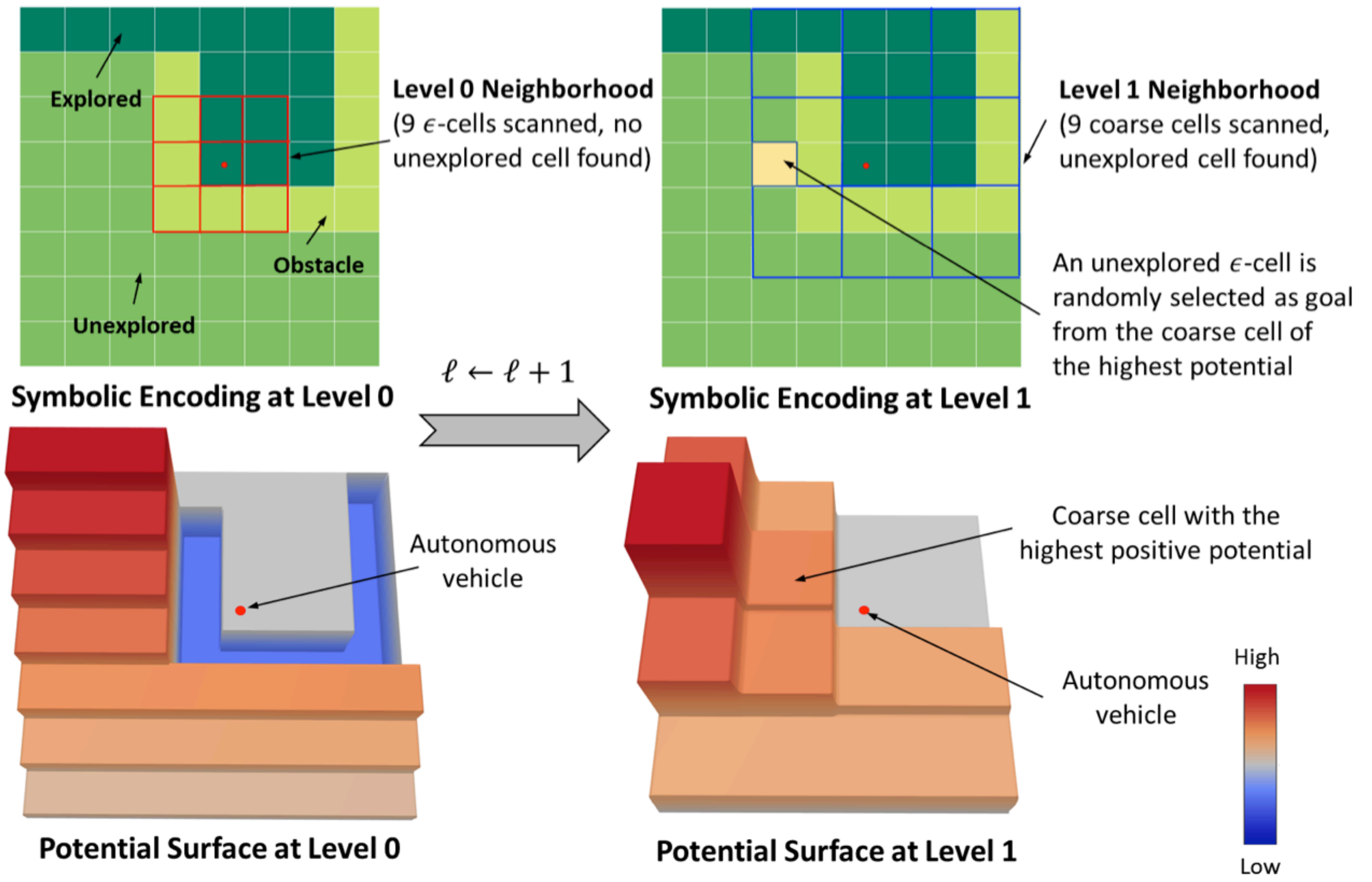
MAPS update

2.4375	2.5
2.4375	2.5

$\epsilon^\star$ Algorithm

An Example of using MAPS to Prevent the Local Extrema Situation



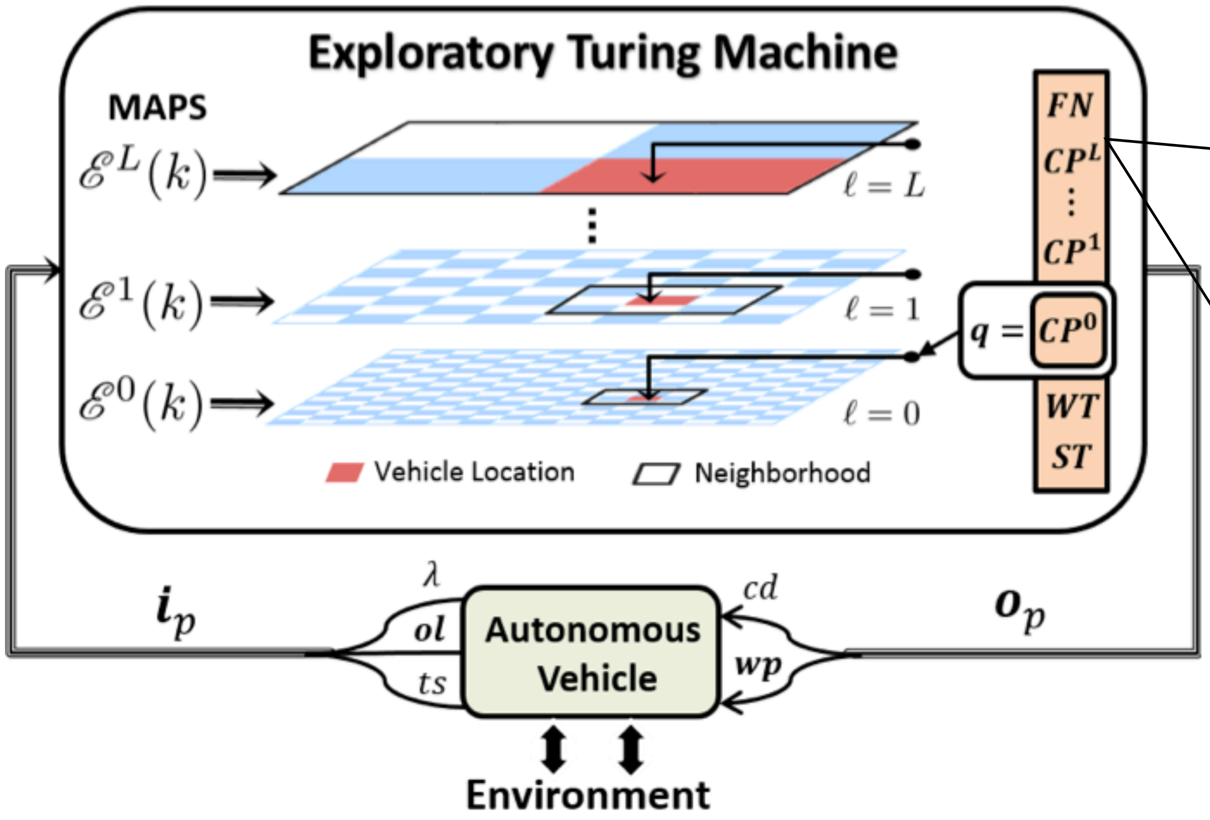


Local Extrema: no unexplored cells are available in the local neighborhood on Level 0.

Low Complexity: even in the worst-case scenario, it only takes $O(|N^0| + L \cdot |N^\ell|)$ to find waypoints, where N^ℓ is the local neighborhood on Level ℓ of MAPS, $\ell = 0,1,\ldots,L$.

$\epsilon^\star$: The Supervisory Control Structure

The Exploratory Turing Machine (ETM)



Machine States

- **The Start State (ST)**: start the machine and initialize the MAPS with all ϵ-cells as unexplored.
- **The Computing States (CP)**:
  - CP^0: compute waypoint wp using Level 0 of MAPS, and send navigation command cd.
  - $CP^1, CP^2 \ldots, CP^L$: sequentially used to compute wp in case of a *local extremum.*
- **The Waiting State (WT)**: wait for the vehicle to complete specific task (e.g., cleaning) in the current cell, until the status ts turns to *complete*
- **The Finished State (FN)**: terminate the operation upon complete coverage.

$\epsilon^\star$ Algorithm

The State Transition Graph

UCONN

LINKS Laboratory of Intelligent Networks and Knowledge-Perception Systems

Conditions:

$\mathcal{A}$: $\boldsymbol{wp} = \emptyset$; $\neg\mathcal{A}$: $\boldsymbol{wp} \neq \emptyset$

$\mathcal{B}$: $\boldsymbol{wp} = \lambda$; $\neg\mathcal{B}$: $\boldsymbol{wp} \neq \lambda$

$\mathcal{C}$: $ts = cm$; $\neg\mathcal{C}$: $ts = ic$

Input Vectors:

$\boldsymbol{i}_{p_1} = (\lambda, \boldsymbol{ol}, -)$;$\boldsymbol{i}_{p_2} = (-, -, ts)$

Output Vectors:

$\boldsymbol{o}_{p_1} = (id, -)$; $\boldsymbol{o}_{p_2} = (tk, -)$

$\boldsymbol{o}_{p_3} = (mv, \boldsymbol{wp})$; $\boldsymbol{o}_{p_4} = (sp, -)$

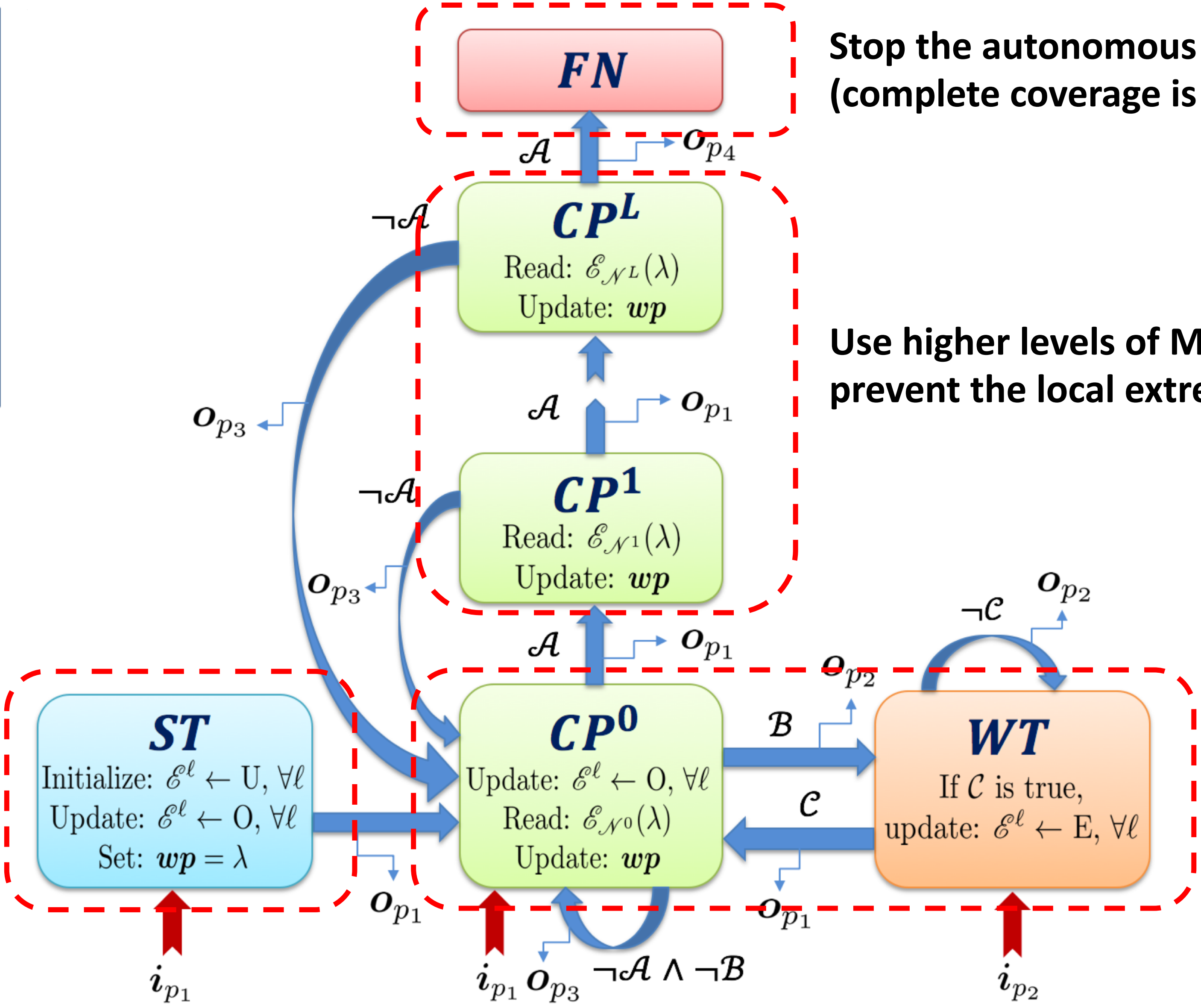
Legend:

Input Vector

Output Vector

State Transition

System initialization



Stop the autonomous vehicle (complete coverage is achieved)

Use higher levels of MAPS to prevent the local extrema problem

Default states for vehicle navigation and control

$\epsilon^\star$ Algorithm

Operation of the ETM: The *ST* State

When is it reached?

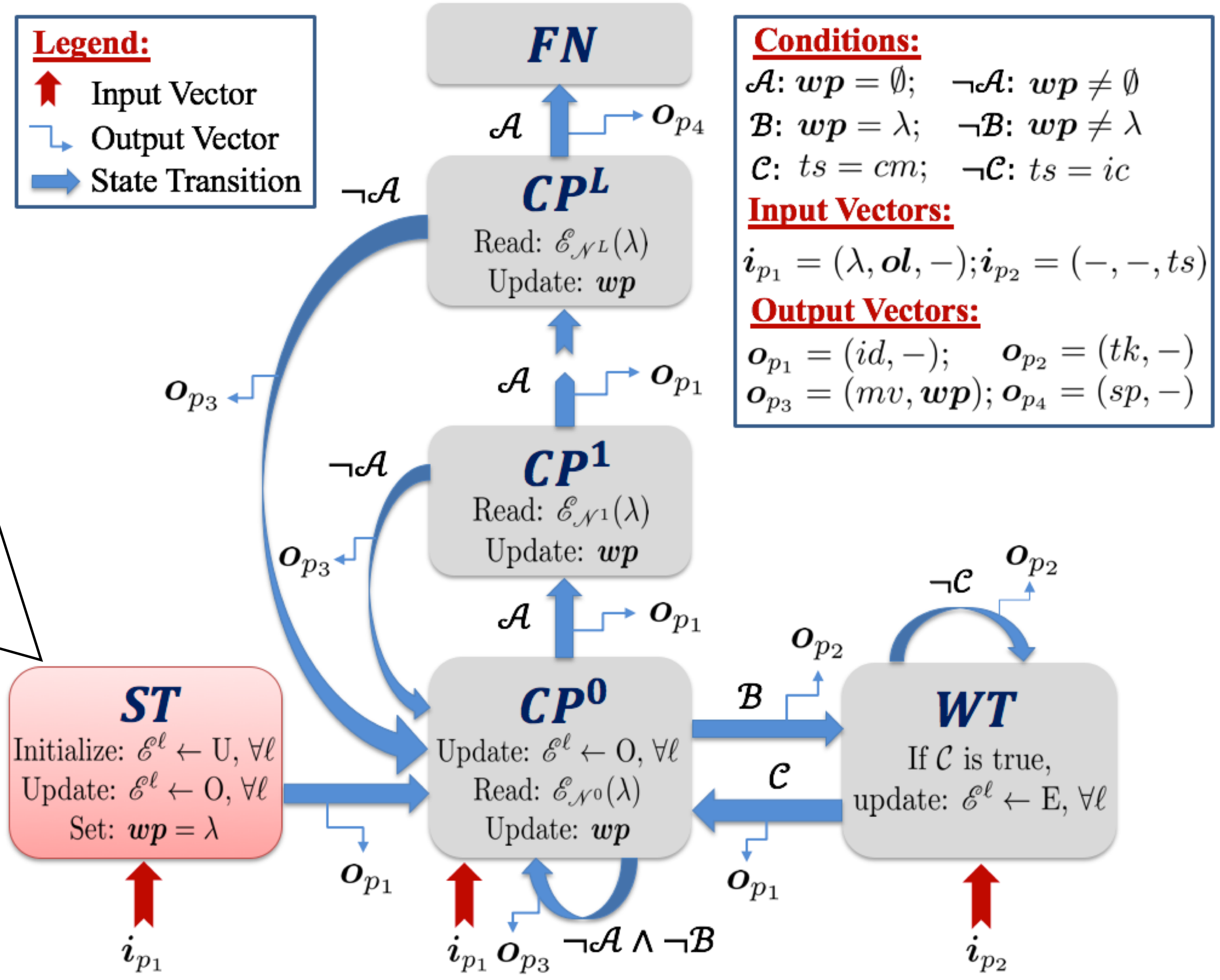
As soon as the autonomous vehicle is turned on.

What does it do?

Initialization of the system.

Operation in the *ST* State:

- **Initialization**:
  - MAPS $\mathcal{E}^\ell, \forall \ell = 0,1,\ldots L$:
    - Level 0: initialized with $\boldsymbol{U}$, i.e. *Unexplored*.
    - Level $1 \le \ell \le L$: all coarse cells $\tau_{\alpha^\ell} \in T^\ell$ are assigned potentials by substituting $p^U_{\alpha^\ell}(0) = 1$.
  - Initialize $\boldsymbol{wp}$ as the current cell λ.
- **Input:** the input vector i_{p_1} contains the current vehicle location λ and detected obstacle locations $\boldsymbol{ol}$
- **Output:** Set the vehicle to idle via output vector o_{p_1}.



$\epsilon^{\star}$ Algorithm

Operation of the ETM: The $\boldsymbol{CP^0}$ State

When is it reached?

Either after system initialization, or when the current cell has just been tasked and needs a new $\boldsymbol{wp}$.

What does it do?

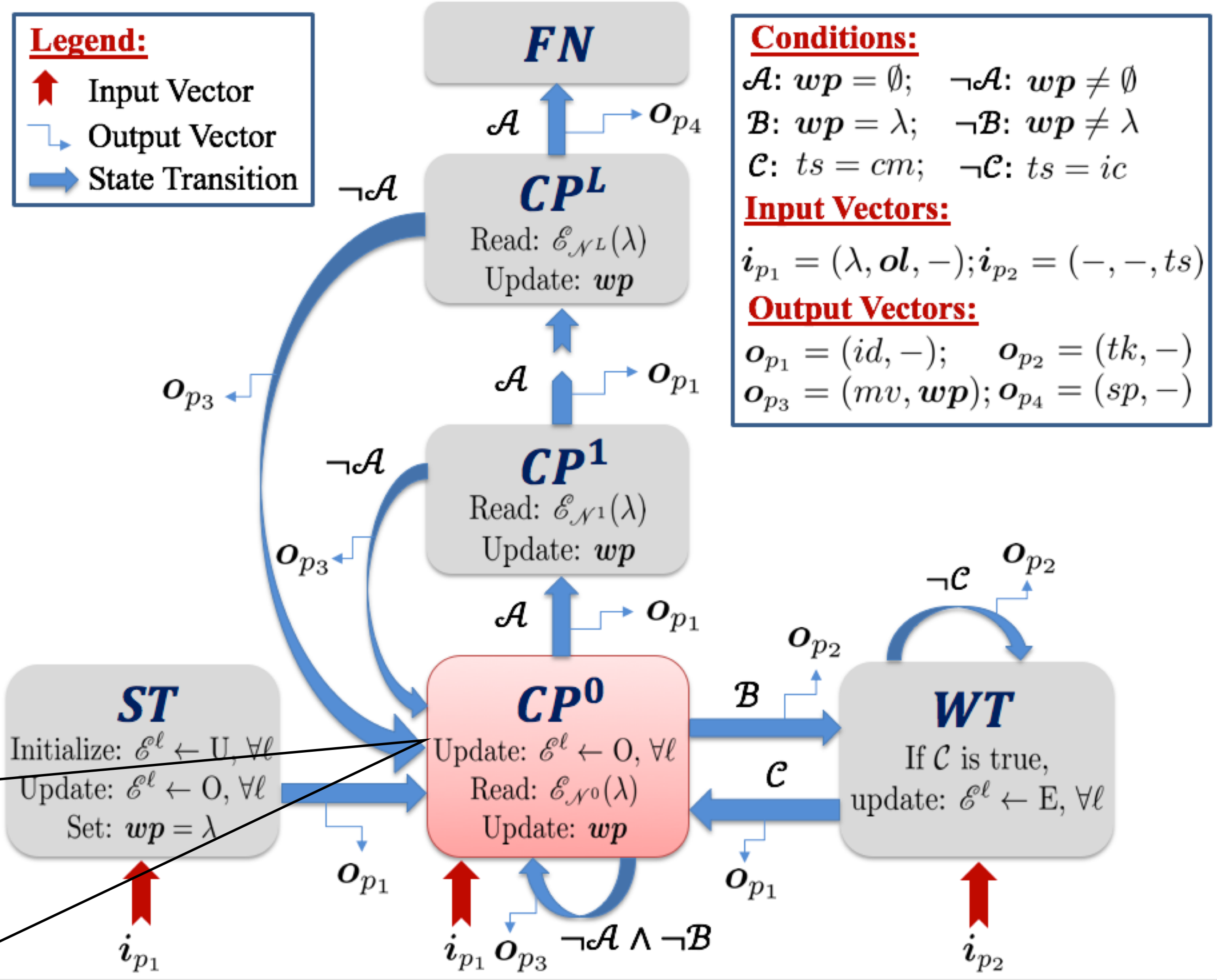
Default state to compute for $\boldsymbol{wp}$ on Level 0 of MAPS.

Operation in the CP^0 State:

- ***Input***: the input vector i_{p_1} contains vehicle location λ, obstacle locations $\boldsymbol{ol}$; they are used to update potential surfaces $\mathcal{E}^{\ell}, \forall \ell$.
- ***Compute wp***: the directly reachable neighbor cell with the highest positive potential in the neighborhood N^0:

$$\boxed{\boldsymbol{wp}(k) = \text{arcmax}_{\alpha^0 \in N^0} \mathcal{E}_{\alpha^0}}$$

- ***Output***: If $\boldsymbol{wp}$ is found, send vector o_{p_3} to move vehicle to $\boldsymbol{wp}$; and upon reaching, send vector o_{p_2} to start tasking.
  - If $\boldsymbol{wp}$ not found, switch to state CP^1



$\epsilon^\star$ Algorithm

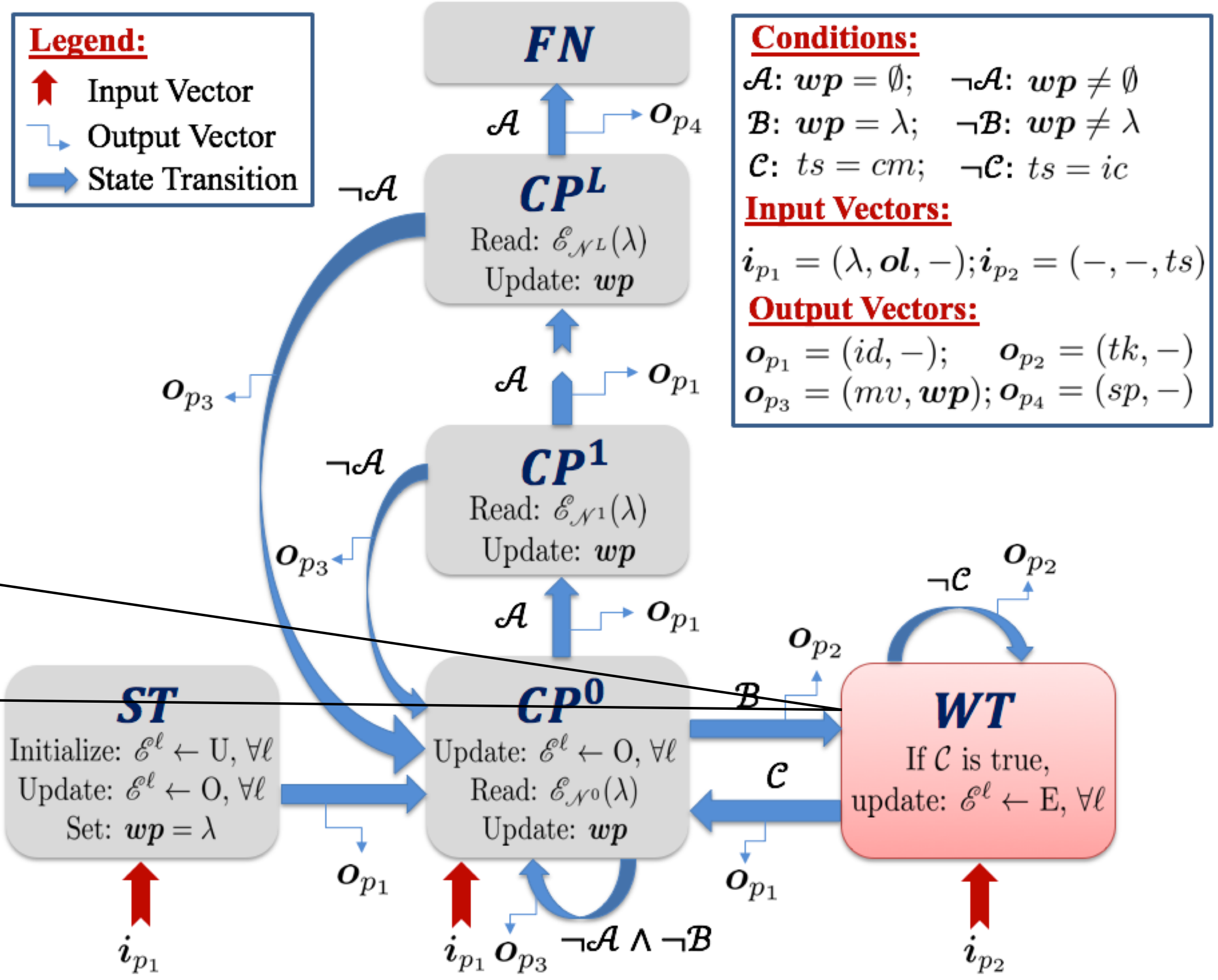
Operation of the ETM: The *WT* State

When is it reached?

When the autonomous vehicle reaches the computed $\boldsymbol{wp}$.

What does it do?

Command the vehicle to perform tasking (e.g., cleaning) in the current cell.



Operation in the *WT* State

- ***Input***: the input vector i_{p_2} contains task status ts; if it is *complete*, update current cell λ as $\boldsymbol{E}$, i.e., *Explored*; then update potential surfaces $\mathcal{E}^\ell, \forall \ell$.
- ***Output***:
  - If task status ts is *complete*, go to state CP^0 to compute for the next $\boldsymbol{wp}$
  - Otherwise, keep on waiting.

$\epsilon^\star$ Algorithm

Operation of the ETM: The $\boldsymbol{CP^1}, \boldsymbol{CP^2}, \dots \boldsymbol{CP^L}$ States

When are they reached?

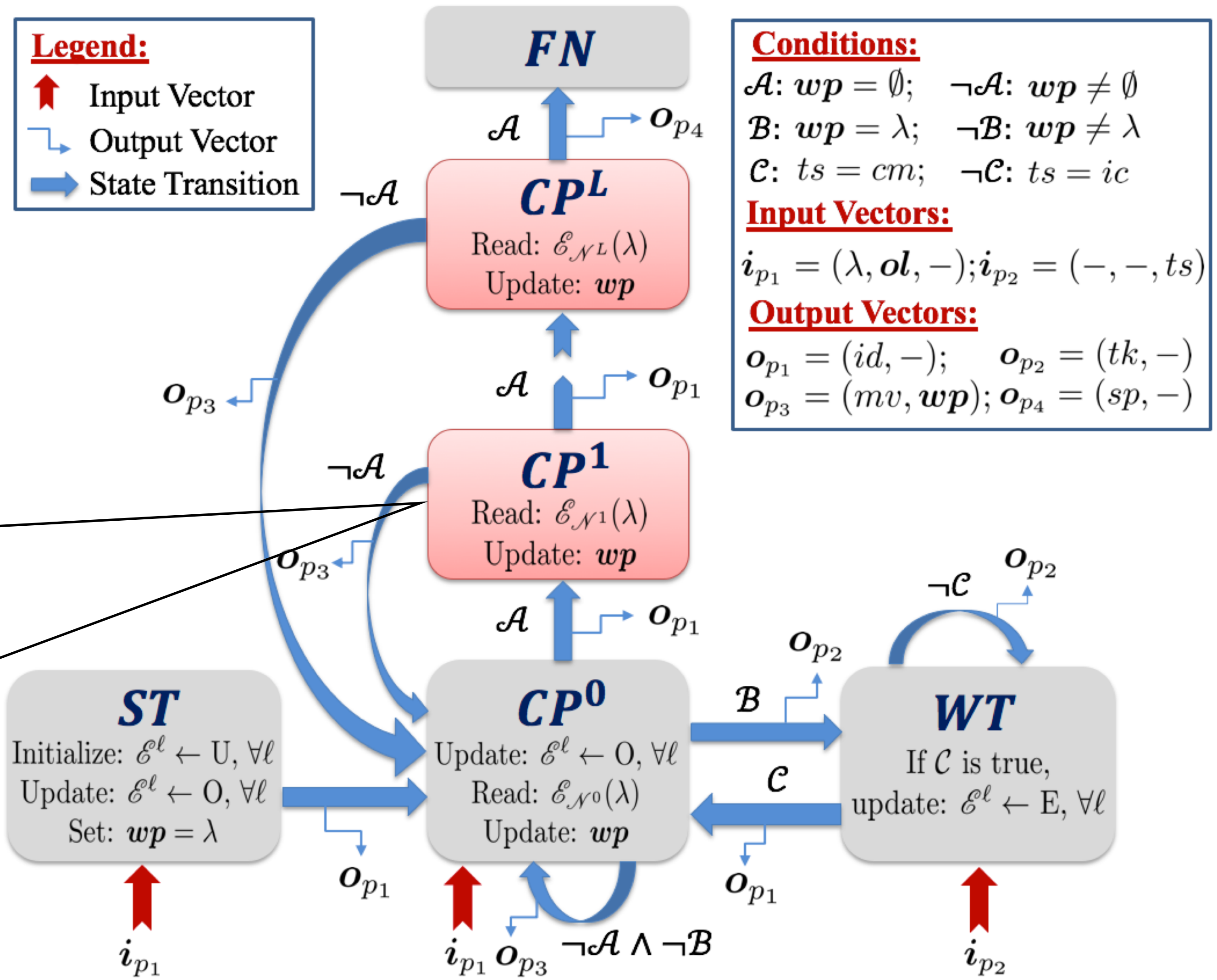
When waypoint $\boldsymbol{wp}$ cannot be found in CP^0 state.

What does it do?

Sequentially switches to higher levels of MAPS, until $\boldsymbol{wp}$ can be found at some Level $\ell \leq L$.

Operation in the $CP^\ell, \ell = 1, 2, \dots L$ States

- ***Compute wp***
  - First, read the potentials in the local neighborhood on Level 1 (i.e., $\mathcal{E}_{N^1}(\lambda)$).
  - If $\exists \tau_{\alpha^1} \in N^1(\lambda)$ with positive potential, then $\boldsymbol{wp}$ is set as an unexplored ϵ-cell in τ_{α^1}.
  - Otherwise, go to CP^2 state and repeat above.
- ***Output***: If $\boldsymbol{wp}$ is found, sends output o_{p_3} to move vehicle to $\boldsymbol{wp}$; otherwise, sends o_{p_1} to set vehicle idle.



$\epsilon^\star$ Algorithm

Operation of the ETM: The FN State

When is it reached?

When $\boldsymbol{wp}$ cannot be found in CP^L state.

What does it do?

Terminate operation since no unexplored cells are left.

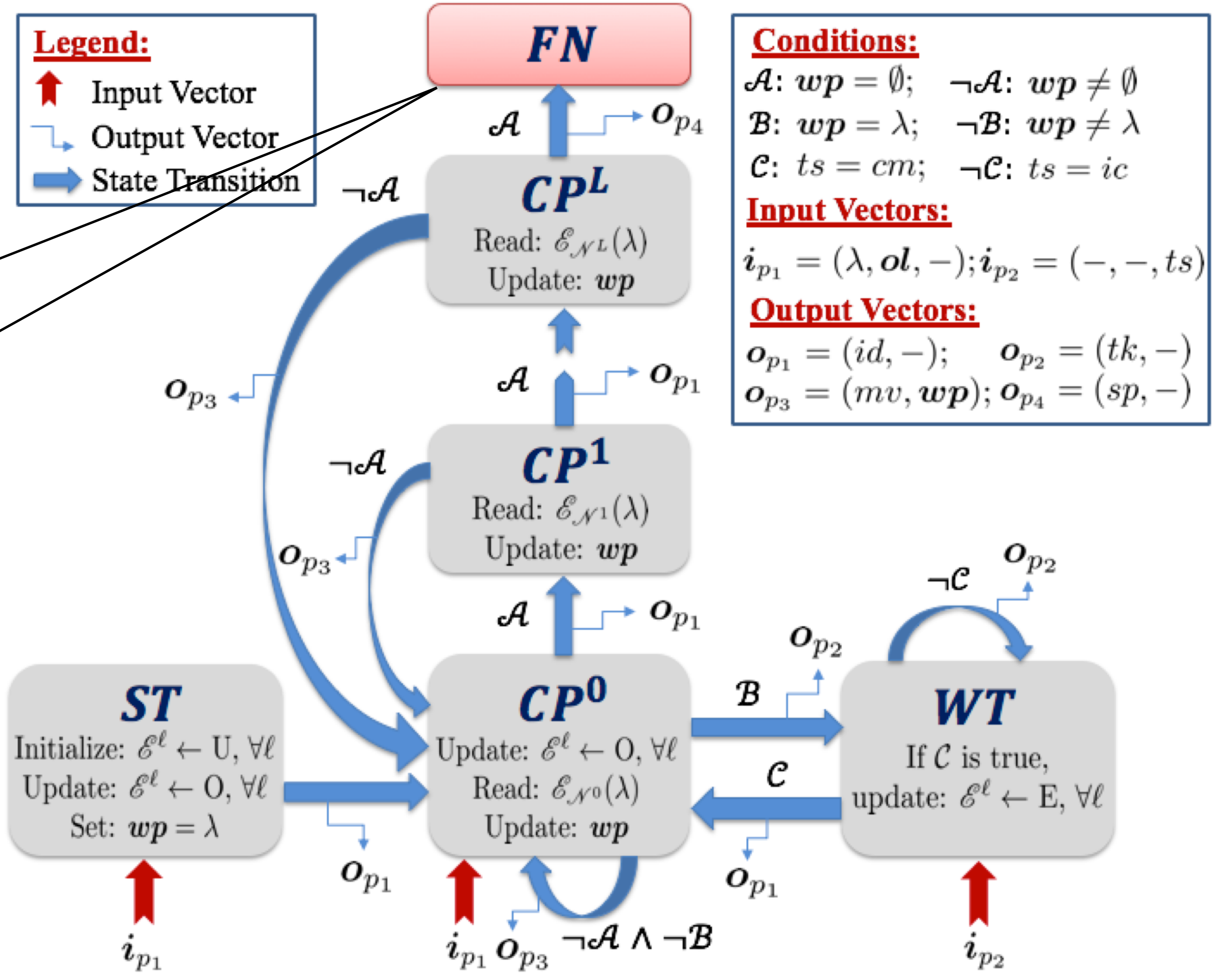
Operation in the FN State

- If $\boldsymbol{wp} = \emptyset$ at Level L, it implies the ETM cannot find unexplored ϵ-cells at the highest Level L, then it reaches the FN state and the machinery is terminated.

Theorem: *The ETM halts in finite time*[1].

Corollary 1: *Each allowed* ϵ*-cell is tasked only once*[1].

Corollary 2: ϵ*-coverage is achieved* upon halting[1].



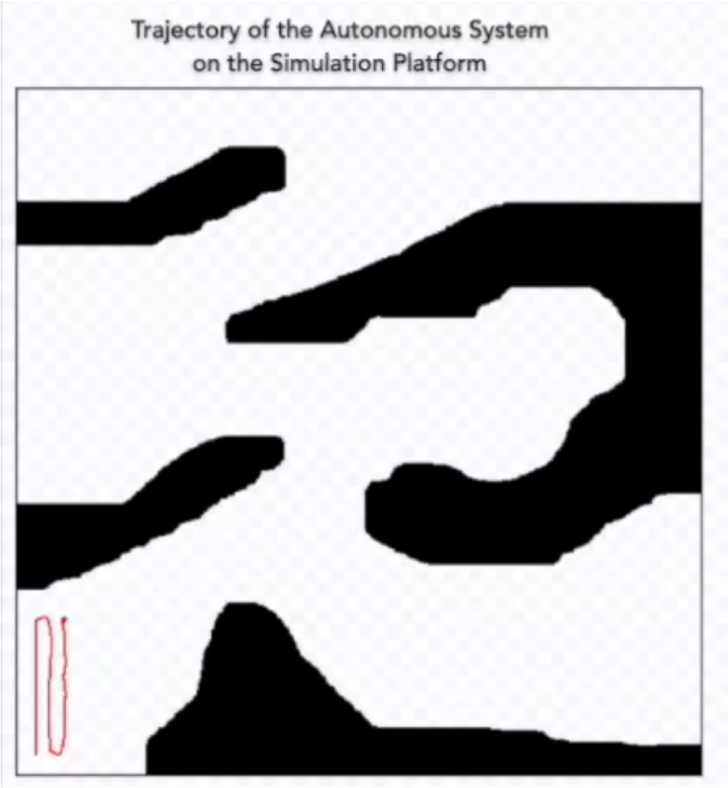
[1] J. Song and S. Gupta, "$\epsilon^\star$: An Online Coverage Path Planning Algorithm", IEEE Transactions on Robotics, Vol. 34, Issue 2, pp 526-533, 2018.

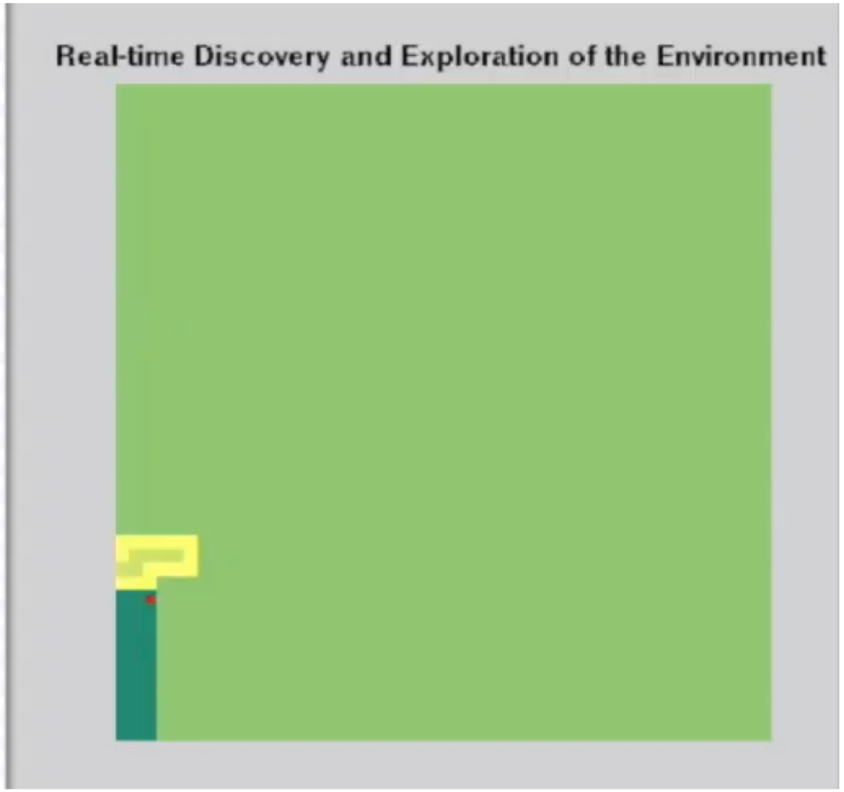
Simulation Validations

Validations on the Player/Stage Robotic Simulator

❖ Simulation Setup

- **Autonomous Vehicle**: a Pioneer AT2 of dimensions $0.44\text{m}\times0.38\text{m}\times0.22\text{m}$ was used with kinematic constraints of:
  - Top speed: 0.5m/s
  - Maximum acceleration: 0.5m/s^2
  - Minimum turning radius: 0.04m
- **Sensing systems**
  - ➢ Laser: detection range of 4m
- **Search Area**: the search area is of size $50\text{m}\times 50\text{m}$, which is partitioned into a 50×50 tiling consisting of ϵ-cells of size $1\text{m}\times1\text{m}$. This results in MAPS with $L = 5$ levels.



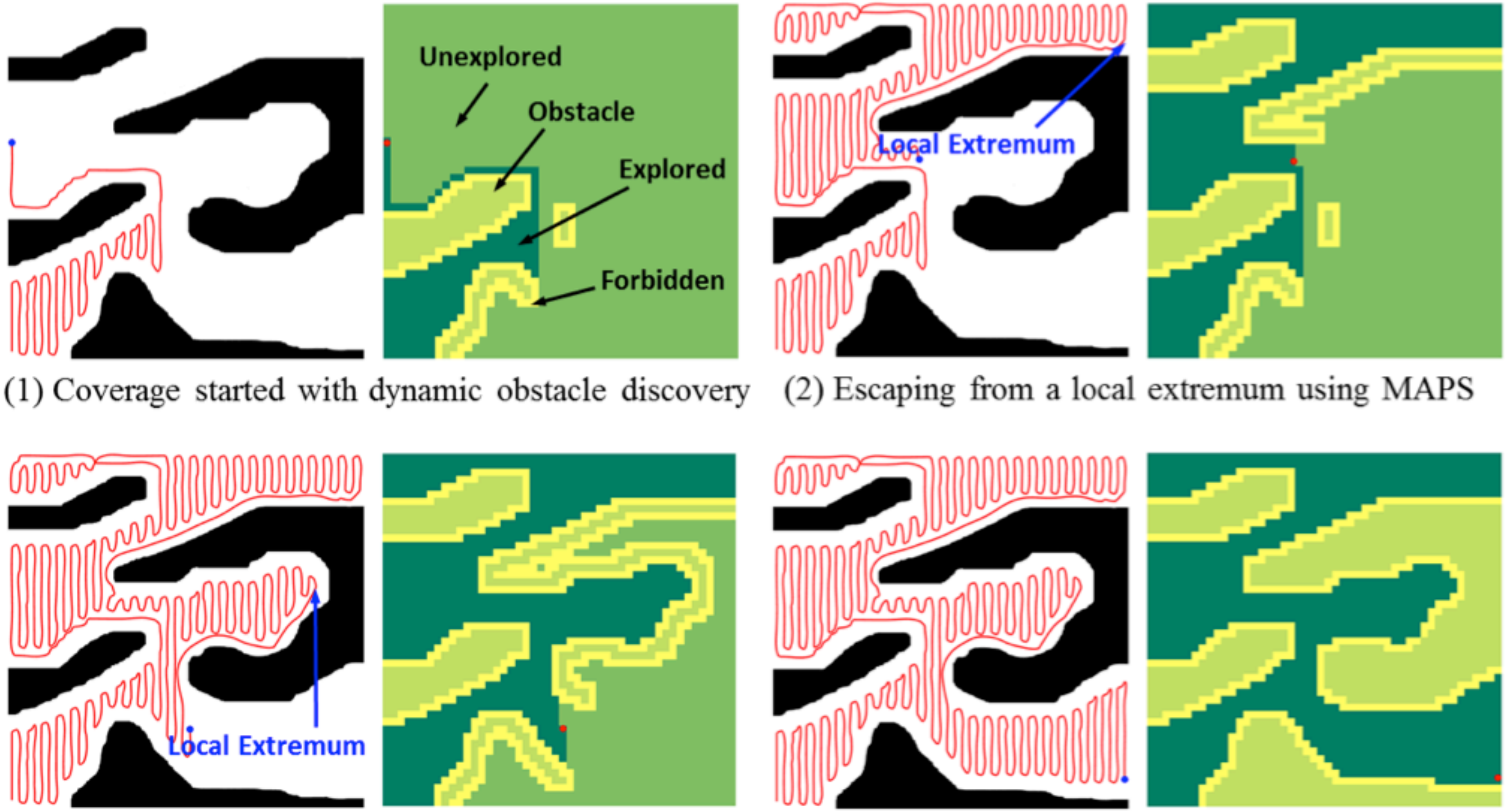


Unexplored Explored Obstacle Forbidden

Simulation Validations

Scenario 1: Coverage Trajectories and Symbolic Encodings of the Environment

- $\epsilon^\star$ incrementally builds the environment map, and complete coverage is achieved.

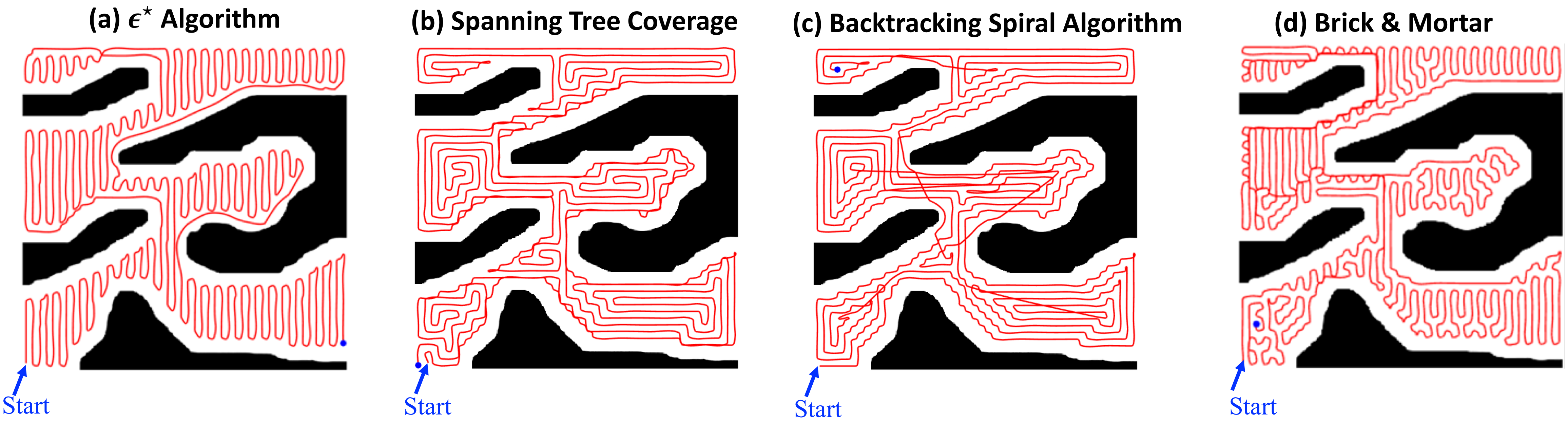


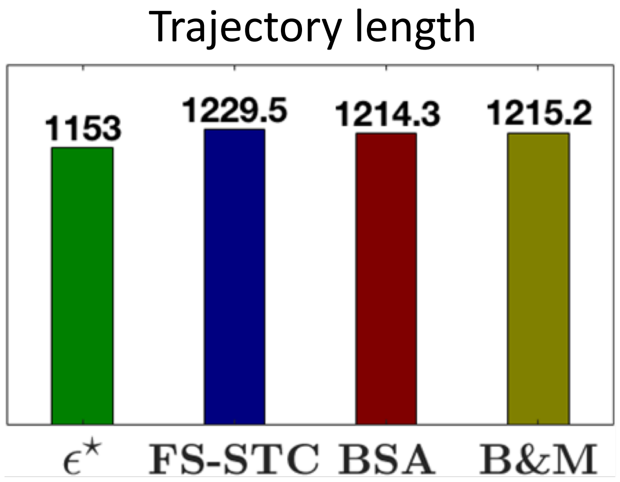
(1) Coverage started with dynamic obstacle discovery (2) Escaping from a local extremum using MAPS

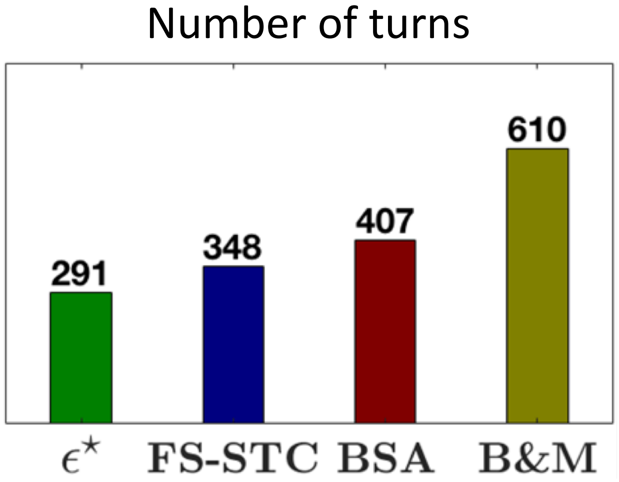
(3) Escaping from another local extremum (4) Complete coverage achieved

Simulation Validations

Scenario 1: Comparison with Alternative Methods







Simulation Validations

Scenario 2: Adaptive Sweep Direction in Known Sub-regions

❖ **User-controllable Sweep Direction**

- If provided (partial) environment knowledge in sub-regions, $\epsilon^\star$ can adapt the sweep direction to further reduce the number of turns.
- In Scenario 2 below, the layouts of all rooms are assumed ***known***, but the inside obstacles are ***unknown***.
- Then, the field B was designed in a manner such that the AV sweeps the top left room horizontally while the other two rooms vertically

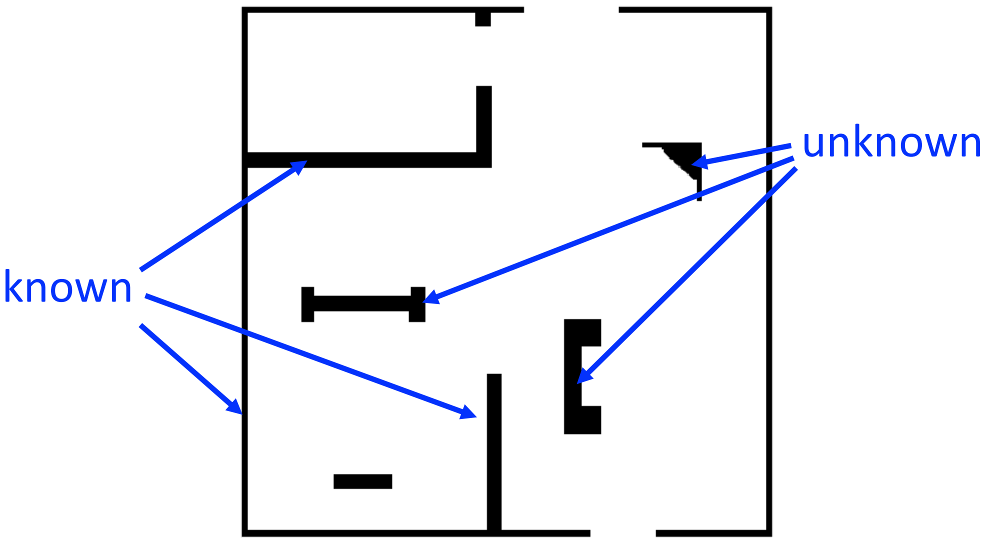


Fig. Scenario 2

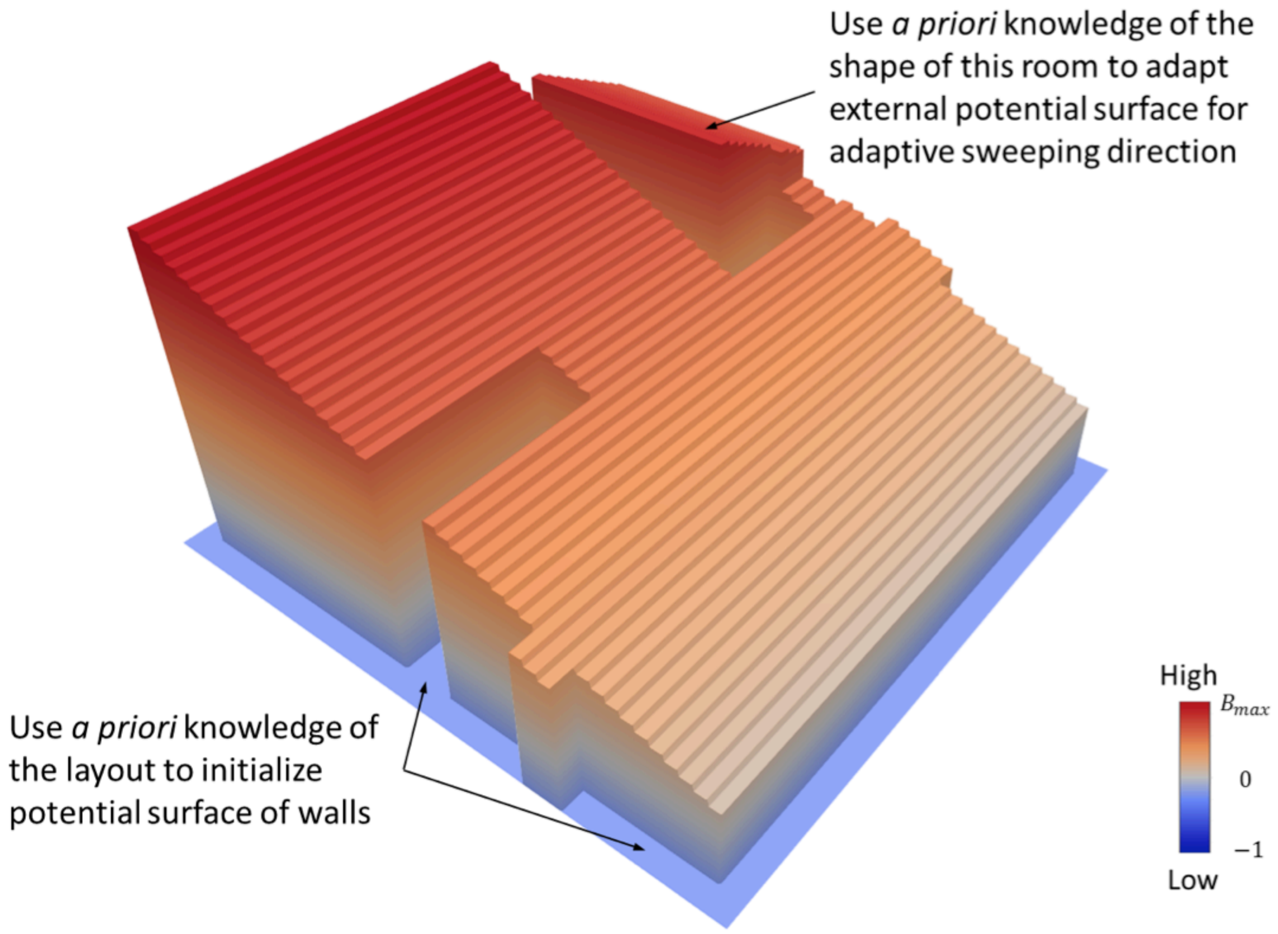


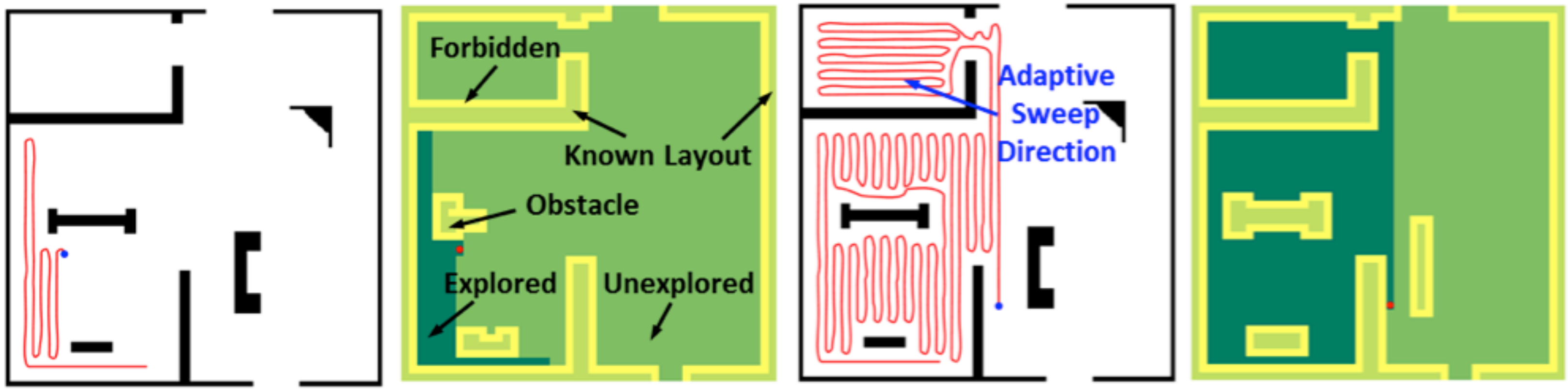
Fig. Exogeneous potential field B in Scenario 2

Simulation Validations

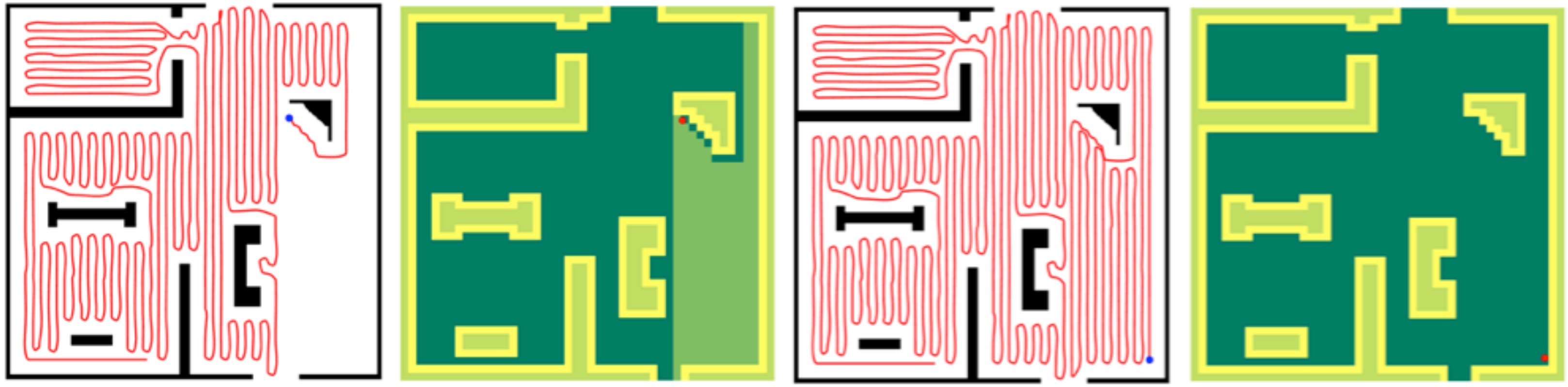
Scenario 2: Adaptive Sweep Direction in Known Sub-regions

- ***User-controllable Sweep Direction***: If provided (partial) environment knowledge in sub-regions, the sweep direction can be adapted to further reduce the number of turns. This is done by altering the exogeneous potential field B.

Scenario 2: Coverage trajectory of $\epsilon^\star$ in an apartment scenario

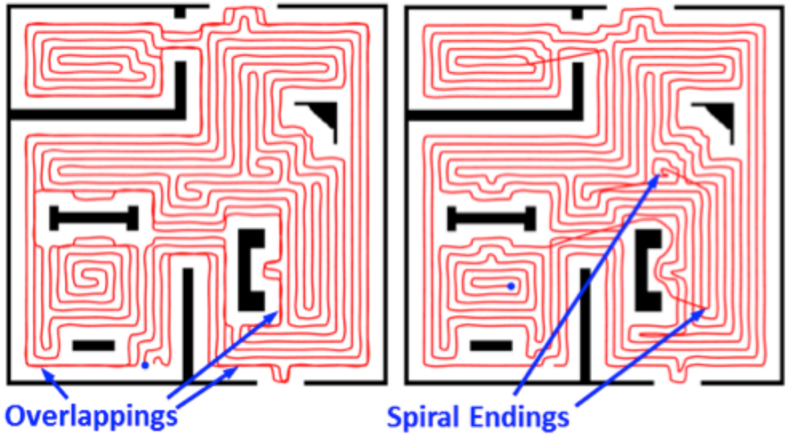


(1) Coverage started with dynamic obstacle discovery (2) Adaptive sweeping if layout is *a priori* known

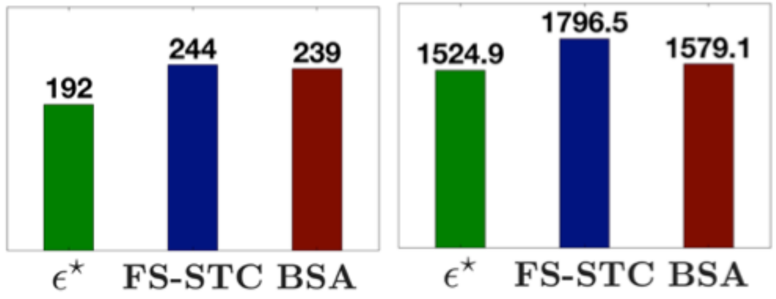


(3) Adapt to the shape of obstacle (4) Complete coverage achieved

Trajectories of alternative methods



FS-STC BSA



Number of turns Total trajectory length

Performance Evaluation

Coverage Performance under Uncertainties

❖ **Coverage Ratio r_c:**

$$r_c = \frac{\cup_k \tau(k)}{\mathcal{R}(T^a)}$$

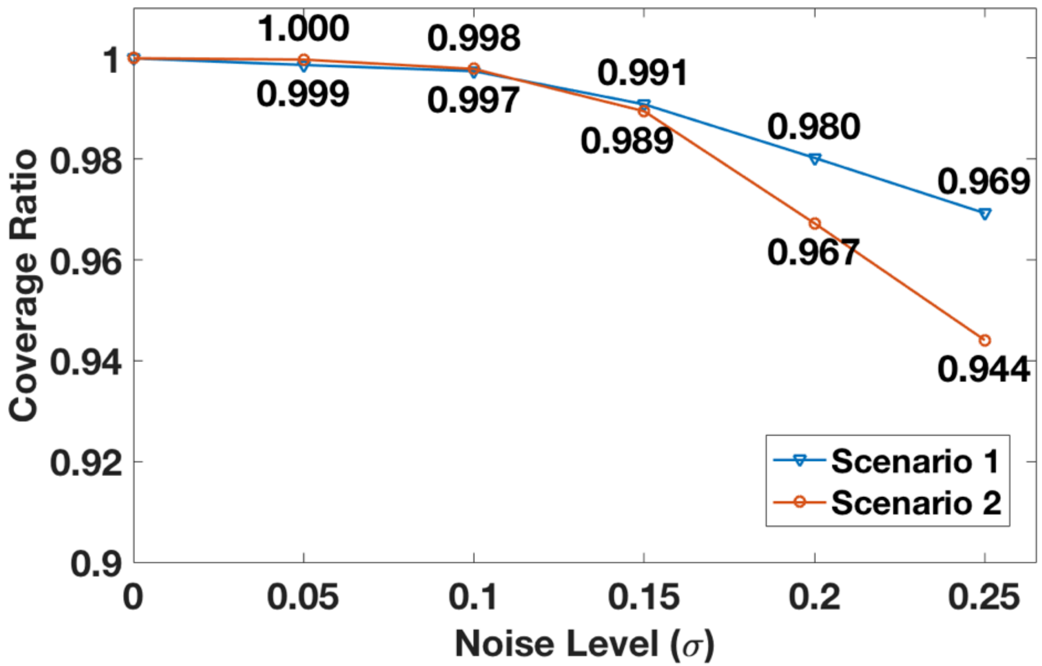
❖ **Sources of Uncertainties:**

- **Localization System**:
  - Outdoor: Real-time Kinematic (RTK) GPS can achieve an accuracy of $0.05\text{m}\sim0.5\text{m}$[1].
  - Indoor: Hagisonic StarGazer indoor localization system provides precision of 2cm.
- **Compass**: a modestly priced compass provides an accuracy of 1^{o}[1].
- **Laser Measurements**: a laser sensor typically admits an error of 1% of its operation range.

❖ **Monte Carlo Simulations**: The sensor noise are simulated as Additive White Gaussian Noise (AWGN), with:

- **Localization System**: $\sigma = 0.05\text{m}, 0.10\text{m}, \ldots\ 0.25\text{m}$
- **Compass**: $\sigma_{compass} = 0.5^{o}$
- **Laser Measurements**: $\sigma_{laser} = 1.5\text{cm}$

Coverage ratio vs. noise for ten Monte Carlo runs



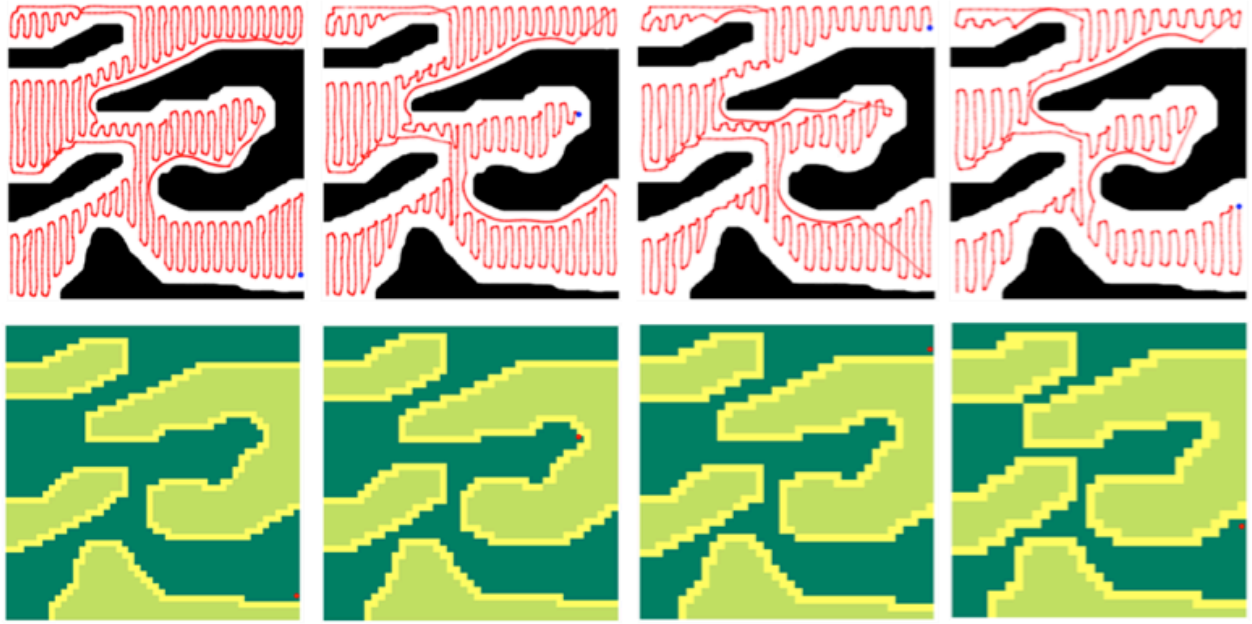
[1] L. Paull, S. Saeedi, M. Seto, and H. Li, “Auv navigation and localization: A review,” IEEE Journal of Oceanic Engineering, vol. 39, no. 1, pp. 131–149, 2014.
[2] J. Palacin, J. A. Salse, I. Valganon, and X. Clua, “Building a mobile robot for a floor- cleaning operation in domestic environments,” IEEE Transactions on Instrumentation and Measurement, vol. 53, no. 5, pp. 1418–1424, 2004.

Performance Evaluation

Choosing a Proper Sized ϵ

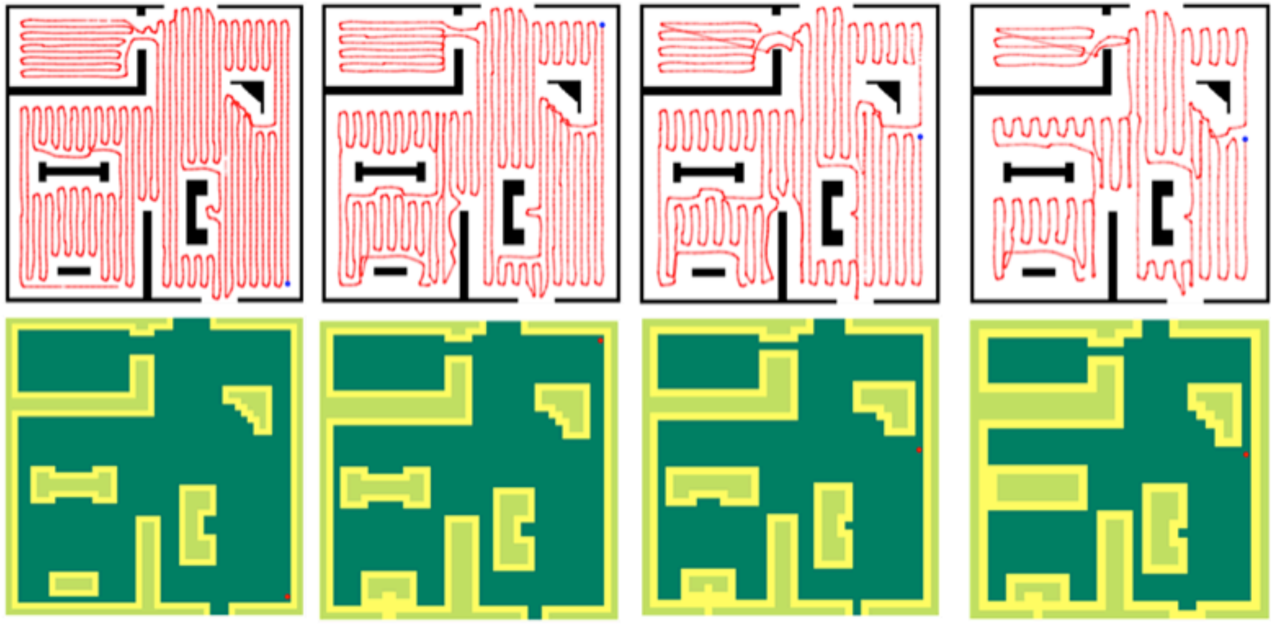
❖ Selection of the Size of ϵ:

- Should be big enough to contain the autonomous vehicle, and small enough for the tasking sensor to be able to cover it.
- Within these two bounds, the choice of ϵ depends on the following factors:
  - Smaller ϵ: provides a better approximation of the search area and its obstacles.
  - Larger ϵ: reduces the computational complexity by requiring less number of ϵ -cells to cover the area and it also provides improved robustness to uncertainties for localization within a cell.



Tiling: 50 cells by 50 cells $\epsilon = 1m$ | Tiling: 45 cells by 45 cells $\epsilon = 1.111m$ | Tiling: 40 cells by 40 cells $\epsilon = 1.25m$ | Tiling: 35 cells by 35 cells $\epsilon = 1.429m$

Figure 1. Scenario 1: coverage trajectories for varying size of ϵ



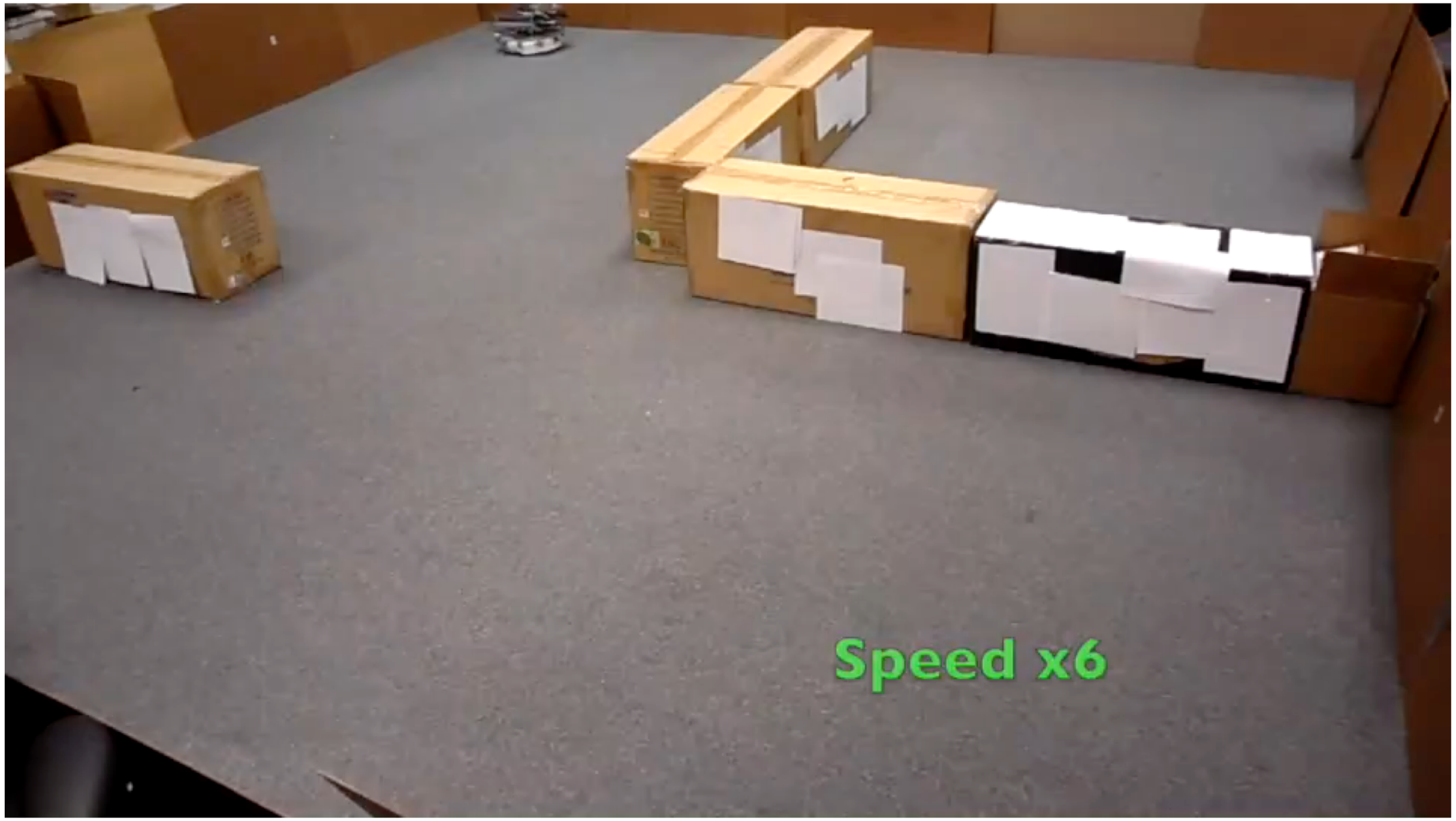
Tiling: 50 cells by 50 cells $\epsilon = 1m$ | Tiling: 45 cells by 45 cells $\epsilon = 1.11m$ | Tiling: 40 cells by 40 cells $\epsilon = 1.25m$ | Tiling: 35 cells by 35 cells $\epsilon = 1.429m$

Figure 2. Scenario 2: coverage trajectories for varying size of ϵ

Real Experiments

The Autonomous Ground Vehicle (AGV)

- $\epsilon^\star$ algorithm was validated in real laboratory-scale experiments to address real-life uncertainties in sensing and vehicle control
- iRobot Create was used as the AGV, which is *programmable* and *controllable* using feedbacks from popular sensing devices



An AGV integrated with multiple sensing devices

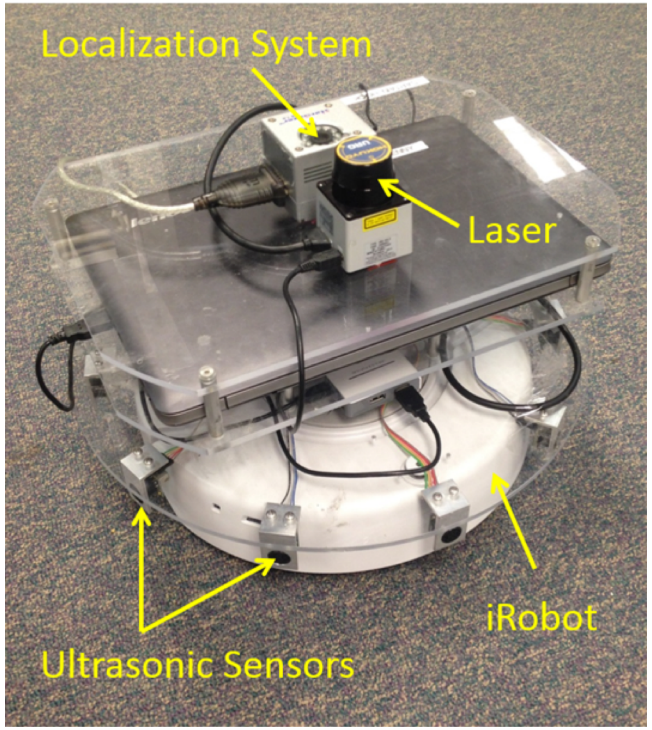


Table. Specifics of the on-board sensing systems

	Localization	Laser	Ultrasonic
Model	StarGazer	URG-04LX	XL-MaxSonar-EZ
Range	–	$0.02m \sim 5.6m, 240^o$	$0.2m \sim 7.65m$
Resolution	$1cm, 1^o$	$1mm, 0.36^o$	$1cm$
Accuracy	$2cm, 1^o$	$\pm 1\%$ of Measurement	–