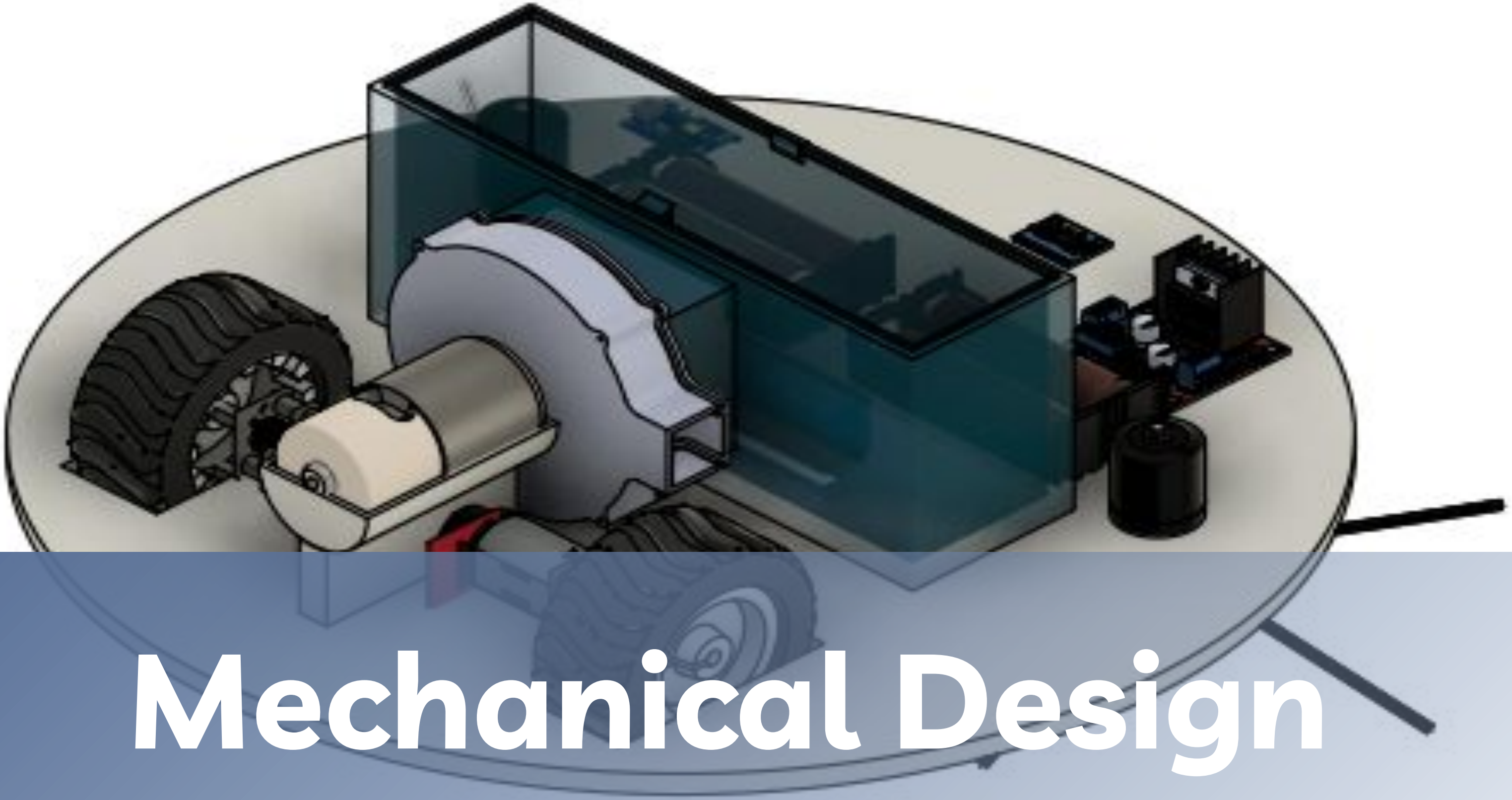
Multi-agent Coverage Planning System

TEAM TECHKNIGHT’S ALLIANCE

IIT MADRAS

KIRTAN PATEL
KAHAN LAKHANI
NISHARG MANVAR
HARISH R
DHRUV MAROO

TABLE OF CONTENTS



Mechanical Design

Ideation

The key factors taken into consideration while design the robot were:

- Dimension Constraints
- Cleaning Efficiency
- Equipment Onboard
  - Electrical equipment
  - Cleaning equipment
- Mobility

Taking into consideration the above factors, the robot was designed to be of a disk shape for better maneuvering with two rotating brushes on the sides for increasing cleaning area and efficiency.The wheel placement was decided by triangulation method to provide more room at the front for a higher sweeping area for the brushes.

Manufacturability

- Body and Bin
  - The material being used for the body and the bin of the robot is ABS (Acrylonitrile butadiene styrene). ABS is widely used for the following reasons:
    - Its cheap
    - Easy to manufacture
    - Can injection molded
- Brushes
  - The brushes are an integral part of the robot and play an important role in cleaning. They need to be very strong as they are needed to rotate at very high RPM but at the same time flexible enough to scrape the floor efficiently and also not hinder in the mobility of the robot. The brushes used are of nylon to provide necessary strength and flexibility.

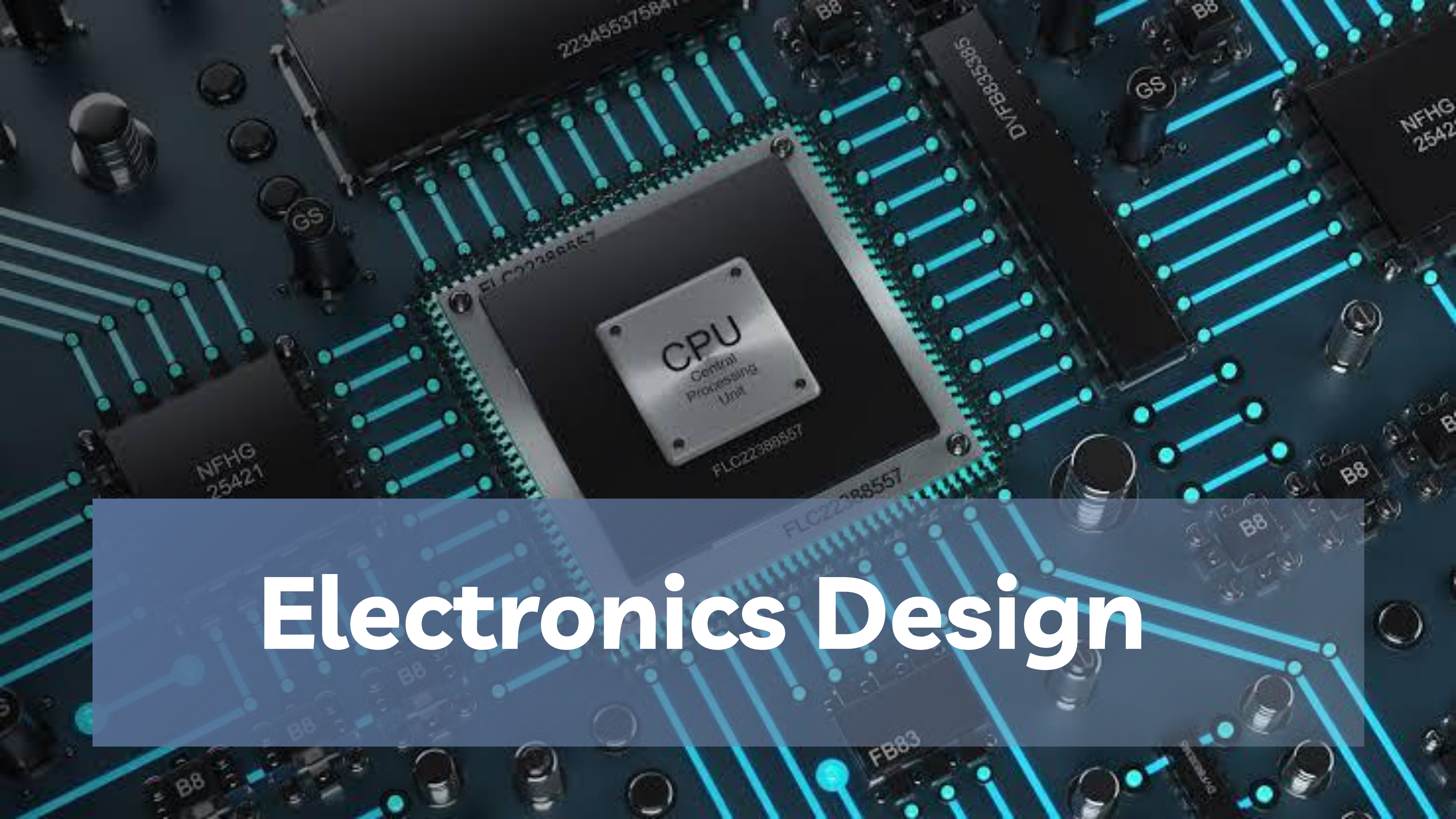
Serviceability

Since we expect the robot to function in a dirty environment, the mechanical parts used need regular maintenance for optimum performance and need to be replaced after a certain period of time.

PART	CLEANING FREQUENCY	REPLACEMENT FREQUENCY
BIN	AFTER EVERY USE	–
BRUSHES	AFTER EVERY 5 USES	AFTER 180 USES
FILTER	AFTER EVERY 5 USES	AFTER 50 USES
FRONT CASTER WHEEL	AFTER EVERY 10 USES	–

Failure Cases

- Material Failure
    - The body has been designed with an appropriate quantity of ABS plastic to handle the weight of the components and the high torques developed.
- Contamination
    - Vacuum fan
    - Motors
    - Electrical components
- Re-dispositon



Electronics Design

Sensors

IMU

- Component chosen : MPU6050
- Purpose : To get the accurate position as well as orientation of the bot at all times

LIDAR

- Component chosen : TF-LUNA Micro LiDAR
- Purpose : to map and understand the environment around the bot

Microcontroller

- Component chosen : ESP32 WROOM board
- Purpose : To obtain all the sensor data and publish to the ros core running on the local machine and give instructions to the motors and controllers.

Electronics and Power

BATTERY PACK

- A battery pack consisting of 4 LG HG-2 18650 3000 mAh Li-ion batteries
- Aspects considered :
  - Energy density
  - Should last long enough for one clean without having to recharge in middle
  - Minimum calendar ageing

MOTORS

- DC Gear Motor
  - Component Chosen : RS-540SH-5045
  - Purpose : To drive the wheels
- DC Motor - Fan
  - Component Chosen : 12V - 3000 RPM DC
  - Purpose : to run the vacuum fan

Manufacturability

Component	Manufacturer	Procurement in India	Component Testing
MPU 6050	InvenSense	Mouser Electronics	Basic Calibration (using Adafruit Library)
TF-LUNA Micro LiDAR	Benewake	Robu.in	Elementary software testing (using the Serial library)
ESP32 WROOM board	Espressif	Robu.in	Unit tests (using the ESP-IDF)
LG HG2 18650 Cells	LG	Battery Bro	Capacity tests and CCD tests
DC Motors	Kysan Electronics	Kysan Electronics	Direct testing to find temperature and efficiency

Failure Modes and Effects Analysis (FEMA)

Damaged Sensors

- **CAUSE :** The IMU and Lidar might get damaged due to constant exposure to water and other fluids.
- **EFFECT :** Decrement in accuracy and total non-working in case of severe damage
- **DETECTION :** Unusual performance
- **ACTION :** If one sensor is damaged, the sensor fusion algorithm detects a significant amount of discrepancy in the values obtained from two sensors.

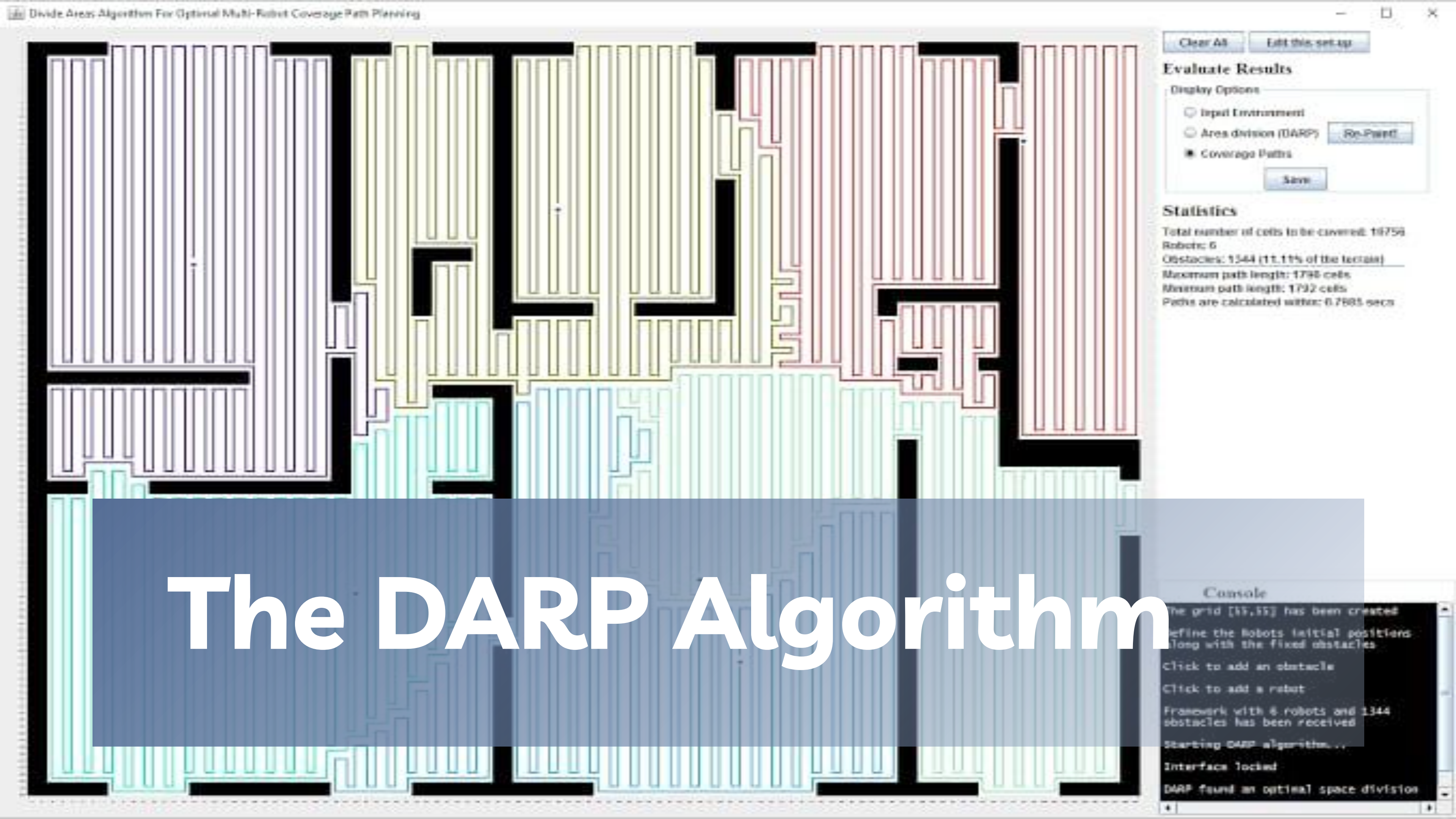
Erroneous Navigation

- **CAUSE :** The cumulative errors in the position of the bot leads to a significant deviation from the intended path
- **EFFECT :** Fallacious values being fed into the path planning algorithm further reducing the efficiency of bot
- **ACTION :** The following solutions are proposed :
    - Implementation of an Extended Kalman filter algorithm to fuse the sensor data obtained from inertial measurement unit and LIDAR. (Can be achieved via ROS Packages)
    - Setting up a Proportional-Integral-Derivative controller in the navigation ROS stack to fine tune the control.

Failure Modes and Effects Analysis (FEMA)

Abnormal Temperature

- **CAUSE :** Due to overload and continuous operations, the Lithium batteries might get over-discharged
- **EFFECT :** Significant risk of abnormal temperatures and the batteries getting exploded
- **DETECTION :** The internal temperature sensors monitor the temperature of the batteries and the system continuously
- **ACTION :** The battery pack comes with a BMS which should handle this case forcing a shutdown of the power circuit. Further to prevent reaching abnormal temperatures, a proper ventilation system is considered while designing the model.


Divide Areas Algorithm For Optimal Multi-Robot Coverage Path Planning
Clear All
Edit this set-up
Display Options
Re-Paint!
Save

The DARP Algorithm

Evaluate Results

( ) Input Environment
( ) Area division (DARP)
(●) Coverage Paths

Statistics

Total number of cells to be covered: 10758
Robots: 6
Obstacles: 1344 (11.11% of the terrain)
Maximum path length: 1798 cells
Minimum path length: 1792 cells
Paths are calculated within: 0.7885 secs

Console

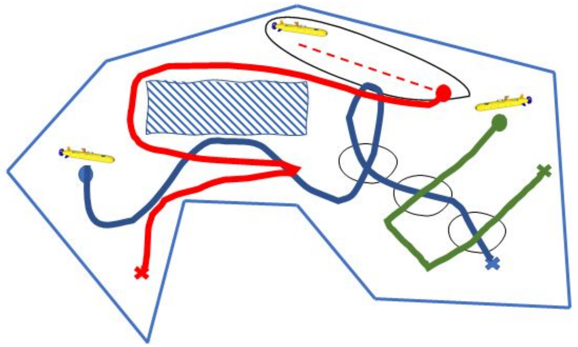
```
The grid [55,55] has been created

Define the Robots initial positions
along with the fixed obstacles

Click to add an obstacle

Click to add a robot

Framework with 6 robots and 1344
obstacles has been received

Starting DARP algorithm...

Interface locked

DARP found an optimal space division
```

DARP Working

Objective:

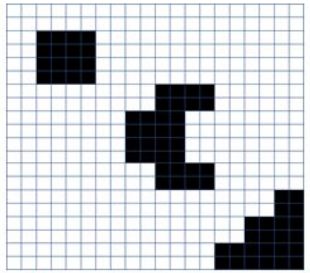
- Design robot paths that completely cover this area of interest in the minimum possible time called coverage path planning problem (CPP).

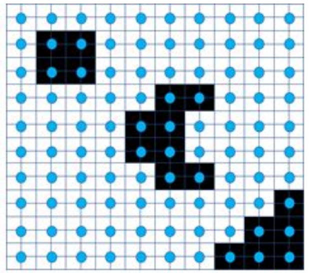
- *Eliminate backtracking*
- *Leave no area uncovered*
- *Equal robots' trajectories*
- *Avoid mismatch between current position and starting location*

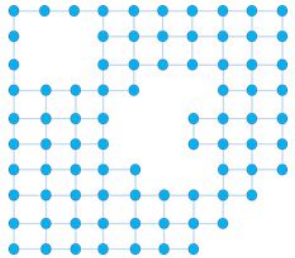


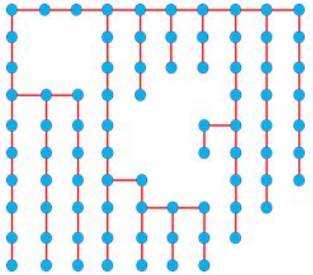
Basic Building Blocks

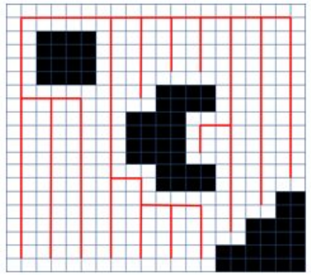
- Cellular decomposition
- Spanning tree coverage (STC)

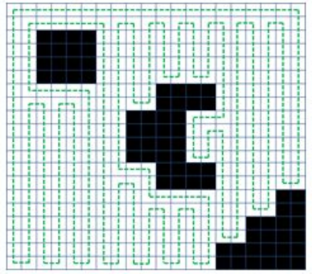

(1) Area discretization


(2) Convert cells to nodes


(3) Discard nodes and edges that correspond to obstacles

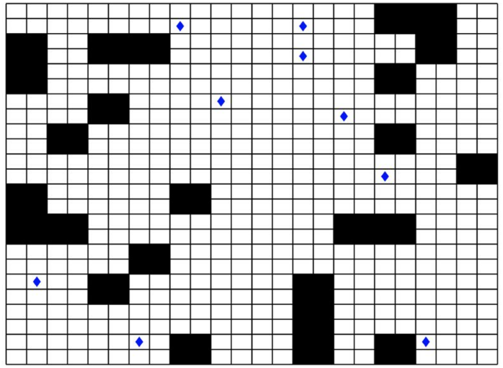

(4) Construct a Minimum Spanning Tree

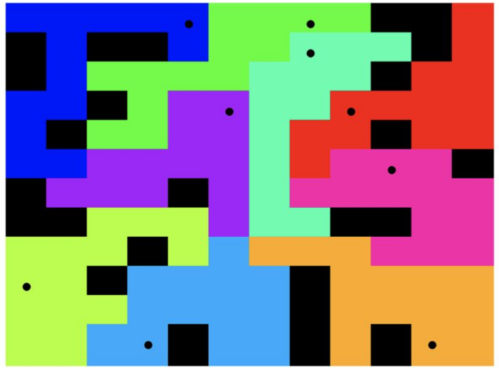

(5) The resulting MST is "our guide" for the navigation

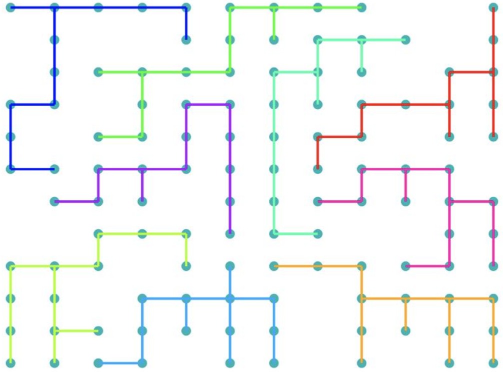

(6) The final path that circumnavigates the MST

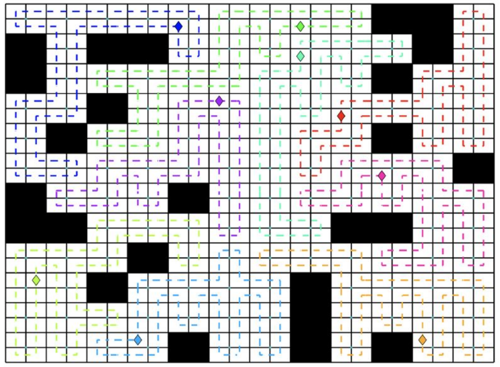
mCPP (multiple CPP)

- So, the main idea of DARP (Divide Areas based on Robot's Initial Positions) is to apply STC individually for each robot by first assigning a certain area of the map to each robot.


(a) Initial cells discretization, robots cell and obstacles

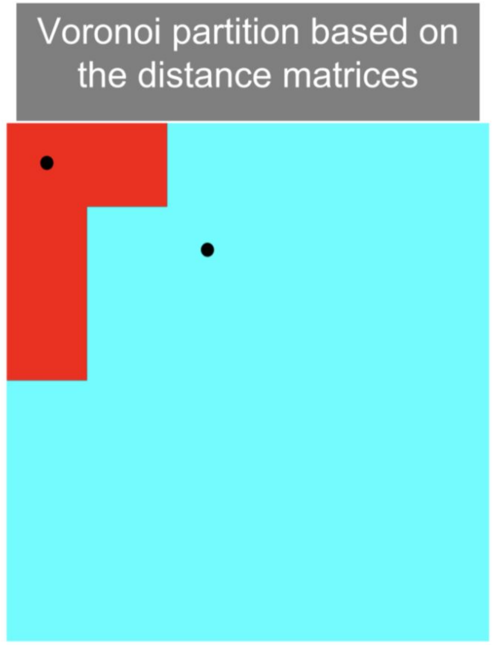

(b) DARP outcome - robots' exclusive areas


(c) Constructing Minimum Spanning Trees for each one of the robots sets


(d) Final Paths, designed to circumnavigate the MSTs

How to assign regions ?

- The most natural way seems to use the idea of Voronoi partitioning, i.e. assign a block to the robot with the closest initial position.
- We again run into the problem where the area that each bot covers becomes dependent on the initial position of each bot.



DARP 0.5

- For every ith operational robot, an evaluation matrix Ei is maintained. This evaluation matrix Ei expresses the level of reachability (e.g. distance) between the cells of L (the map) and ith robot's initial position xi(t0).

$$A_{x,y} = argmin_i(E_{i|x,y}), \forall\, i$$

- The DARP algorithm's core idea is that each evaluation matrix Ei can be appropriately "corrected" by a term mi as follows:

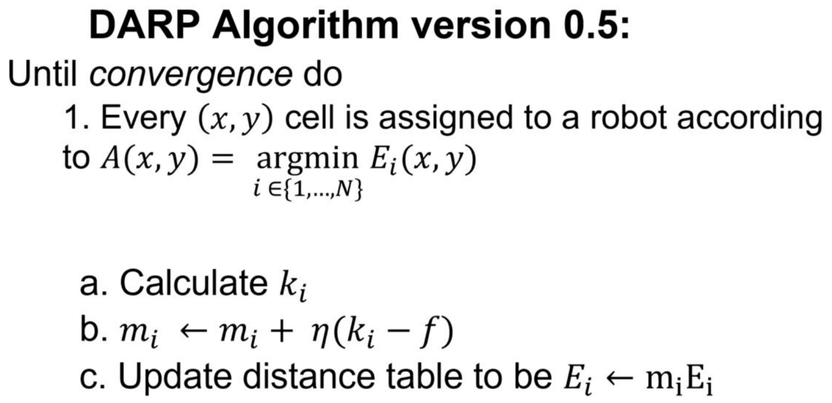
$$E_i = m_i E_i$$

- Error function

$$J = \sum_{i=1}^{N} (k_i - f_i)^2$$

- Where ki denotes the number of assigned cells to the ith robot, while fi denotes the number of cells that the ith bot has to cover.

$$m_i = m_i - \eta \frac{\partial J}{\partial m_i}$$

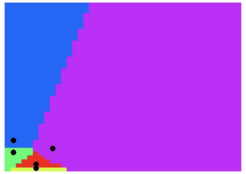


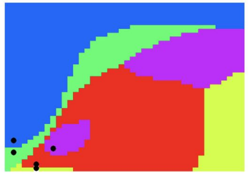
DARP Algorithm version 0.5:

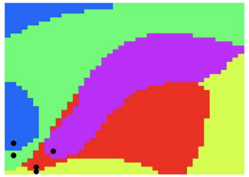
```
Until *convergence* do
    1. Every $(x, y)$ cell is assigned to a robot according
    to $A(x,y) = \underset{i\in\{1,\dots,N\}}{\operatorname{argmin}} E_i(x,y)$

        a. Calculate $k_i$
        b. $m_i \leftarrow m_i + \eta(k_i - f)$
        c. Update distance table to be $E_i \leftarrow \mathrm{m_i E_i}$
```

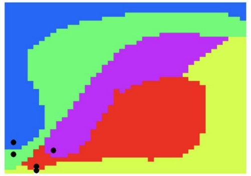
DARP 1.0

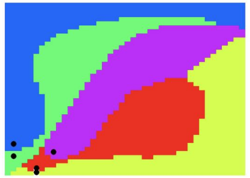
- Spatial Disconnectivity problem:

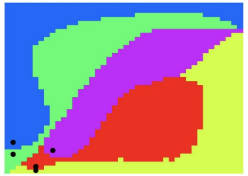

(a) $T = 0$


(b) $T = 40$

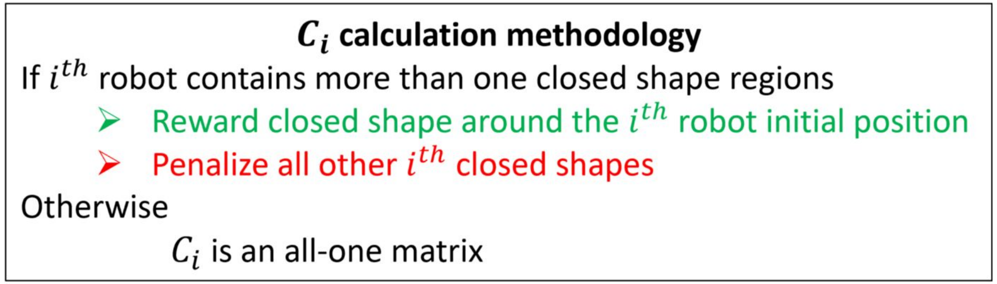

(c) $T = 80$


(d) $T = 120$


(e) $T = 200$


(f) $T = 260$

- To tackle this issue, an extra connectivity matrix is introduced.


C_i calculation methodology

If i^{th} robot contains more than one closed shape regions

- ➢ Reward closed shape around the i^{th} robot initial position
- ➢ Penalize all other i^{th} closed shapes

Otherwise

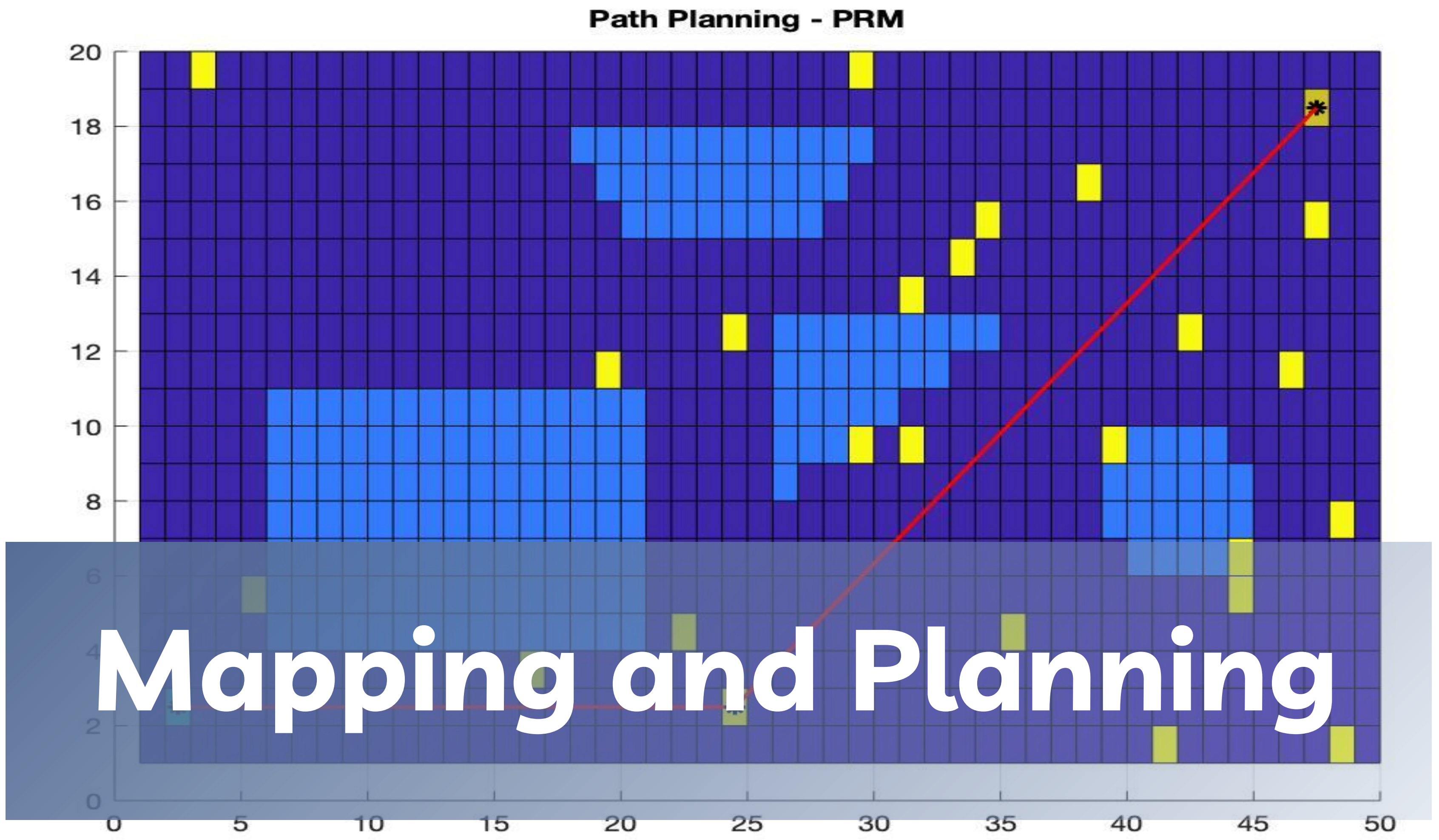
C_i is an all-one matrix

DARP Algorithm version 1.0:

```
Until all area division objectives are met do
    1. Every $(x, y)$ cell is assigned to a robot
    according to $A(x, y) = \underset{i \in \{1,\dots,N\}}{\operatorname{argmin}} E_i(x, y)$
    2. For each $i^{th}$ robot do
      a. Calculate $k_i$
      b. $m_i \leftarrow m_i + \eta(k_i - f)$
      c. Calculate connectivity matrix $\boldsymbol{C_i}$
      d. $E_i \leftarrow \boldsymbol{C_i} \odot \mathrm{m_i E_i}$
```

DARP Failure Analysis

- Failure of algorithm to converge due to close initial position of robots.
  - To tackle this problem we introduced a filter in the sense that only if all pairs of robots were separated a constant minimum distance will we try and find the optimal path using DARP.
- In some other cases, even though the bots were far spaced, the algorithm was unable to converge.
  - If we give some leeway in terms of area covered by each bot the algorithm is able to converge.



Mapping and Planning

Mapping and Planning

With Mapping being introduced in the problem, the base problem changes from an offline path planning problem to an online path planning problem.

In offline mode, the path planning task starts from a point where the agent has a complete knowledge about: its environment with obstacles (The complete map is given), its initial position, and its final goal within this environment. In online mode, path planning is carried out in parallel while (a) moving towards the goal, and (b) perceiving the environment including its changes.

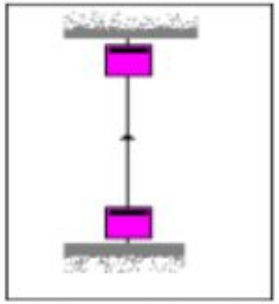
For Online Complete Coverage Planning, we can use either of the 2 established algorithms.

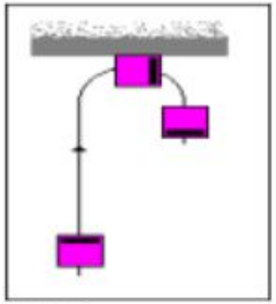
- Template Based Methodology
- Epsilon star

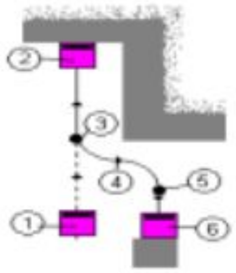
Template Based Algorithms

This system, when implemented on a commercially available mobile platform, exclusively relies on odometry and ultrasonic data, aiming at providing low-cost navigation modules that could be implemented.

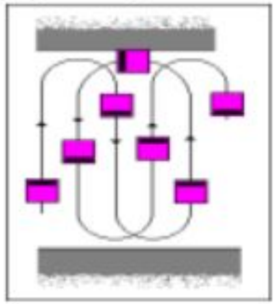
The complete coverage trajectory is planned as a sequence of predefined trajectories denoted as templates along the lines proposed. A set of 5 templates is proposed as the minimum number to achieve a satisfactory oor coverage of an area which contains obstacles in its interior. Wall tracking is also used.

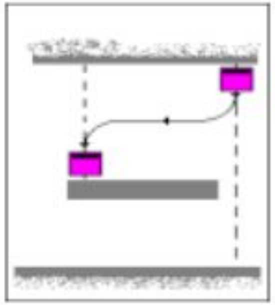

(a) Template TM


(b) Template UT

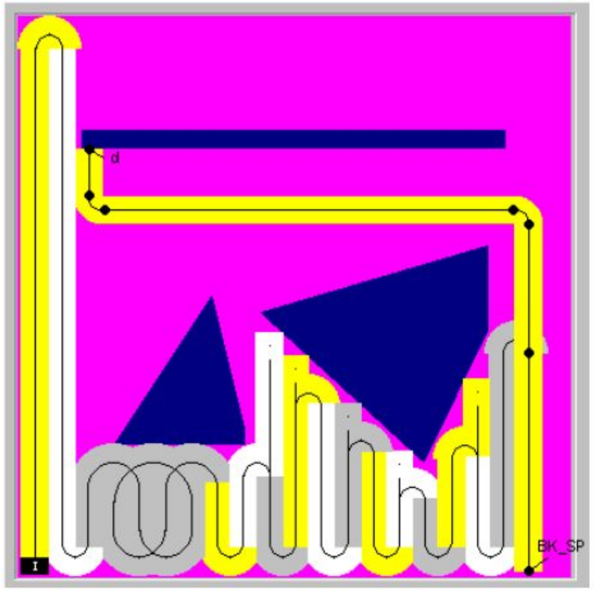


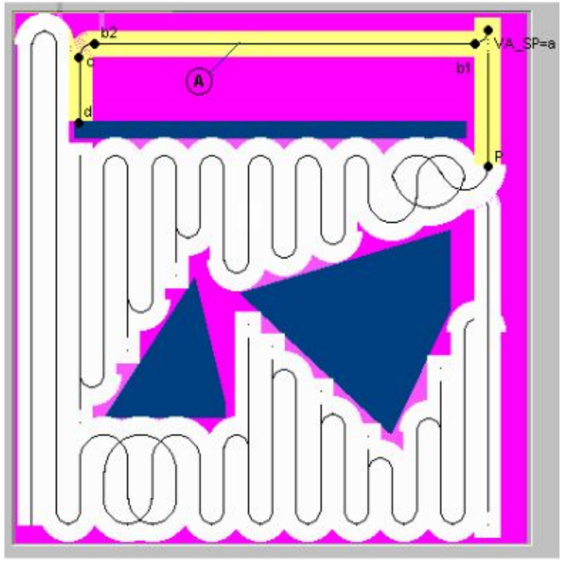
(c) Template SS

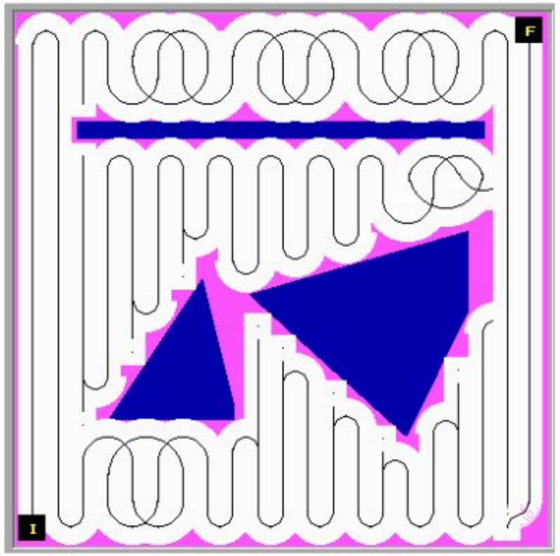

(d) Template UTI


(e) Template BT

Final Robot Paths







Epsilon Star

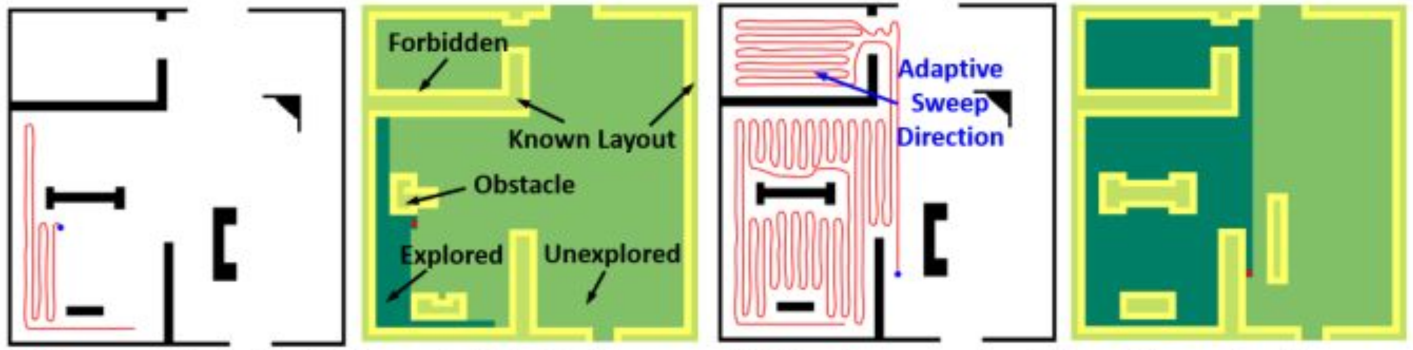
The Epsilon Star Algorithm works on a grid based approach.

The algorithm is built upon the concept of an Exploratory Turing Machine (ETM) which acts as a supervisor to the autonomous vehicle to guide it with adaptive navigation commands. The ETM generates a coverage path online using Multiscale Adaptive Potential Surfaces (MAPS).

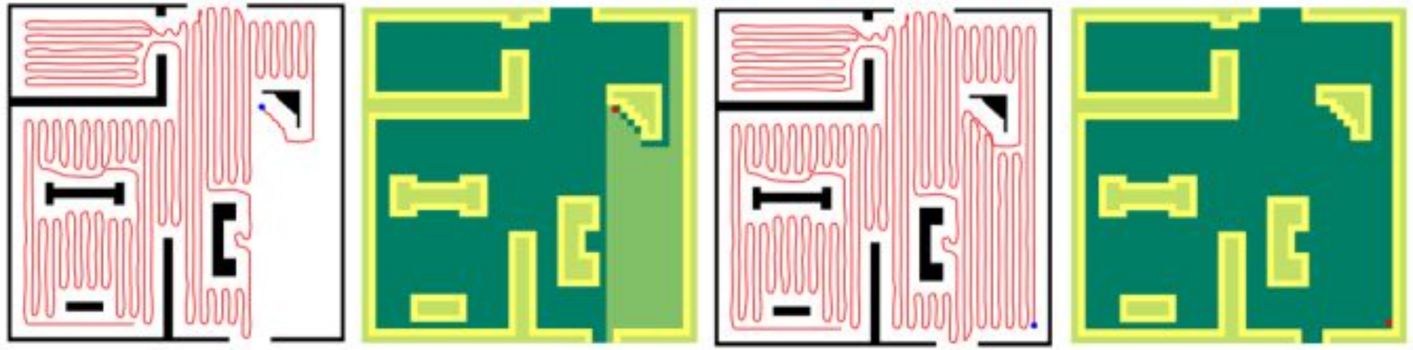
The Epsilon Star Algorithm can be understood as :

- Dividing Coverage Area into Tiles
- Dynamically Constructing Multi-scale Potential Surfaces (MAPS)
- Supervisory Control Structure of the execution
- Completing the task with no cells left to unattended on a global level.
    - This means that a global extrema has been reached

Final Robot Path



(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

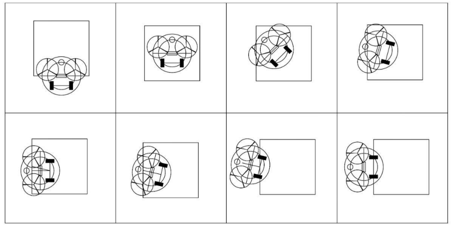
(a) Trajectories and corresponding color coded symbolic encodings for ϵ^*-algorithm

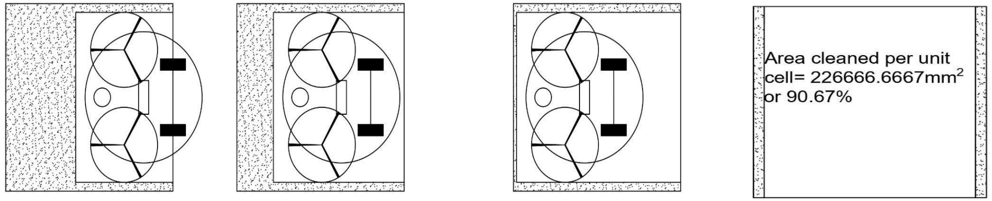
Cleaning Efficiency

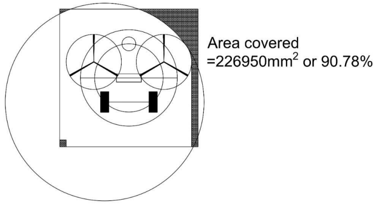
Coverage depends on the value of the orientation of the obstacle and epsilon. For grid obstacles and Smaller epsilon, we can achieve a cleaning efficiency of more than 90%. Having said that, since we are using Online Path Planning, the exact cleaning efficiency varies from map to map.

From Round 1 Report, it is clear that the bot covers the grid cell with more than 90% efficiency and hence any complete coverage algorithm used with the bot would give a cleaning efficiency of more than 90%.

Furthermore, with smaller and smaller values of epsilon and wall tracking, the cleaning efficiency is bound to go higher.







Deployment Strategy

1.Template Based Methodology

Deploying multiple bots in such a scenario can save the repetition of cleaned areas and the travel time from one point to another.

- **Back-tracking**
  We can evaluate the extent of backtracking and deploy a bot in order to avoid having to do that. When a situation of extreme backtracking occurs, we can deploy another bot at the new place instead of having to backtrack.

- **Traveling**
  As seen in the Final Robot Path Example, in the latter part of the path in image 1, we can see that in order to resume it's cleaning after having dealt with the obstacles, the robot travels back to its intended position before encountering the obstacle.In case of large obstacles, this travel time can be significant. Thus as as soon as the obstacle crosses a certain threshold, a new bot can be instantly deployed at the intended position.

  This can be carried out efficiently with 5 bots by cyclic allocation of intended positions.
  For better efficiency, if more than one bots are idle, the bot nearest to it (path-distance, not euclidean), will be assigned to travel to that position and continue the cleaning.

2. Epsilon Star

In any Template based Algorithm, there might be back-tracking and the need for the robot to go back to another location (for complete coverage).

Deploying multiple bots in such a scenario can save the repetition of cleaned areas and the travel time from one point to another.

Using the same approach as above, we can keep the idle robots on stand by and deploy them into the map when needed.

Thank You!
Open to Questions