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. The offline path planning task simply connects the initial position to the final position, then the agent will execute the generated path.

In online mode, path planning is carried out in parallel while (a) moving towards the goal, and (b) perceiving the environment including its changes.

However, Considering the interactions among (i) Agent, (ii) Environment, and (iii) goal, the difference between online and offline path planning can be seen from many points of view:

Offline path planning is generally used for static environments (or slowly changing). and only when a global map of the environment is initially available (given or built). And thus, this approach can ensure global optimum path (in terms of safety, shortest path, time, energy, ...).

For dynamic environments, online path planning must be used since the path must be updated according to environment changes. This approach is also necessary for motion in an incompletely known environment map, since the agent is discovering its map while moving and needs to update the path according to any new knowledge. However, with this partial knowledge, this approach is time consuming, requires extra sensors, and it doesn't generate globally optimum paths, it can even fail to reach the target. but again, it is still a necessity.

Cleaning tasks require a special kind of trajectory able to cover all the unoccupied areas in the specified cleaning environments. Paths that comply with this requirement are known as paths of total coverage.Some methods for generating a path of total coverage, based on a map of the area to be cleaned have been mentioned in the previous report. They use grid-based or template based approaches.

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

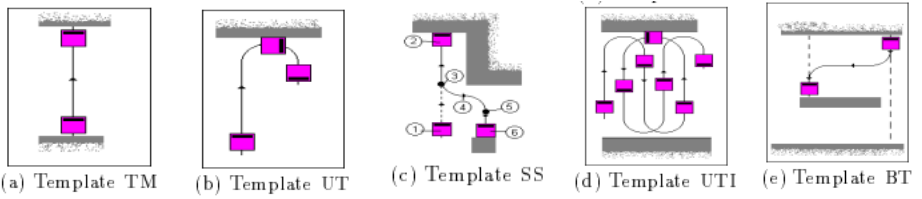
* Template Based Methodology
* Epsilon star

# 1.Template Based Methodology

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.

Unlike the previously used DARP algorithm, for a satisfactory coverage of the entire space, neighbour paths may have overlapping areas. Sequences of maneuvers consisting of a predefined number of line segments and arcs will be employed to generate the total coverage path. Each elementary path type is denoted as a template, the total path being a sequence of templates. All templates take into consideration the parameters of the robot such as the minimum turning radius, the robot width and the cleaning area.To account for vehicle dimensions and simplify the path planning, the robot will be considered as a point object.

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.



The use of the BT template is to cope with obstacles in the middle of the environment.

* **Template Towards Marker, TM**

Line segment that links two given points. This template is the backbone of nearly all the other templates

* **Template Uturn, UT**

Line segment followed by a U turn. With the exception of TM the execution of a UT template is fast when compared with all the other templates due to the simplicity of the included maneuvers The successive application of UT templates creates a snake trail pattern as represented above. The platform minimum turning radius and the required overlap between adjacent trails are the main geometric parameters associated with UT

* **Template Side Shift, SS**

This template provides a tool for changing between adjacent tracks when the mobile robot has a minimum turning radius that prevents the use of a UT template to achieve that change A Side Shift template can also be used when an environmental barrier such as a wall prevents the use of a UT template, this being the situation displayed. The main drawback of this template is that high coverage is redundant.

* **Template U turn Interlaced, UTI**

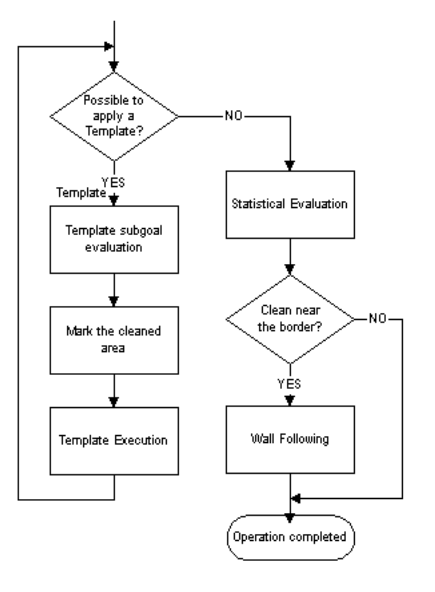
It is well suited for robots with a large minimum turning radius and its application reduces the wheel slippage when compared to the UT template This fact was experimentally observed.

* **Template Backtracker BT**

This template is extremely useful when there are obstacles lying in the middle of the area to be cleaned The backtracking movements supported by template BT allow the mobile robot to clean areas not yet covered that would otherwise be left behind uncleaned, this being one of the novelties of the proposed method.

The complete coverage path planning methodology is implemented iteratively after an initial localization procedure. After a human operator drives the platform to its initial location and an estimated location is evaluated based on ultrasonic data. At each iteration and based on the previously cleaned area and the a priori map the algorithm chooses the template to be applied as explained later in this section Before execution the trajectory that will be generated by the chosen template has to be further decomposed in such way that simple motion commands.such as Move(x,y,forward) or Turn(angle,left,radius) can be dispatched to the mobile robot This decomposition achieved by the path tracking module breaks up the templates into subgoals The subgoals of each template are the boundary points that separate line segments from turns thus decomposing the template in a set of subtemplates see for details Following template subgoals evaluation a prediction of the new cleaned area is carried out followed by the template execution which is achieved by the execution of each of its sub templates.

If no template can be applied this means that all the area has already been cleaned or there is a deadlock situation. In both cases a further procedure for wall following is implemented This procedure fully supported on ultrasonic data.



For obstacles like the ones mentioned in Round-1, (All those that can be shows as blocks on a grid),A template will always be applicable, except at the end, or if the robot is not aligned properly. In cases with slat edged obstacles, the wall following is implemented.

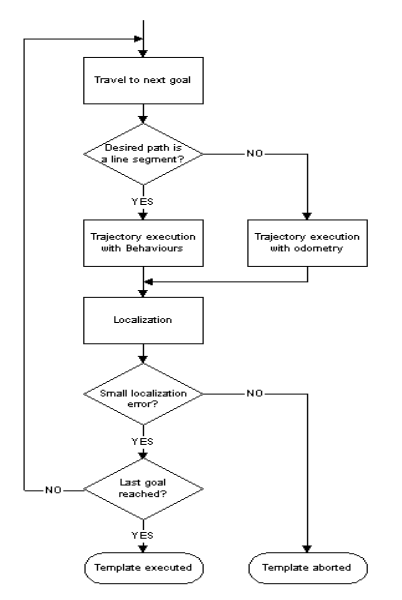
A heuristic formula is used for choosing the best template to apply at each stage aiming at maximizing the total covered area The template BT is the 1st one whose application is checked by the planner because if any uncleaned space is left behind it must be covered before moving on otherwise it will be left uncleaned.

Next the choice of an interlaced template UTI is trade aiming at reducing wheel slippage and consequently reducing odometry errors Due to a fast execution time the use of a UT template is checked next The Side Shift template is the last one to be considered due to its high coverage redundancy The TM template is used as a major component of all the other templates given that all line segments are executed by TM templates

Thus, the order or checking of templates is as follows: BT →UTI →UT →SS

## Path Tracking

For Path Tracking and Path Correction using Odometry, the following algorithm is used to track the Path of the robot.



## Localization

Localization is essential to correct the cumulative errors inherent to the odometry system. Using the a priori map and the data from the LiDAR attached on the front of the bot’s body, a test is carried out at each subgoal so as to determine if the real data acquired by the sensors matches the bot’s position and orientation estimate given by odometry. If the deviation between the predicted and the real measurements is more than a certain threshold the real data is used to recalibrate the overall location of the mobile robot. Also, to make localization more robust, we intend to fuse both the aforementioned data sources using Kalman filters, allowing to reduce the effect of noise and errors in the values obtained from these sources.

This localization approach, based on low cost sensors and on the a priori map, requires no special modification on the environment (e.g. installation of bar codes). However, the implemented methodology introduces an overhead on path execution duration because localization has to be done with the vehicle stopped near an obstacle.

# 2. 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) (hierarchical potential surfaces) which are hierarchically structured and dynamically updated based on sensor information. The ε ⋆ -algorithm is computationally efficient, guarantees complete coverage, and does not suffer from the local extrema problem. Its performance is validated by:

i) high-fidelity simulations on Player/Stage and

ii) actual experiments in a laboratory setting on autonomous vehicles.

## Vehicle Requirements

1. Localization System

Provides vehicle location (e.g., GPS), and heading (e.g.,Compass)

2. Range Detector with Sensing Radius Rs

Allows the vehicle to detect obstacles in the local neighborhood (e.g., laser)

3. Tasking Sensor with Radius rt

Allows the vehicle to carry out certain tasks (e.g., cleaning, target detection, crops cutting) while it operates in the field

## Step 1 : Dividing Coverage Area into Tiles

The tiling formed by square tiles of side ε is called an ε -cell tiling. It is recommended that an ε -cell should be atleast big enough to contain the autonomous vehicle and small enough for the tasking sensor to be able to cover it when the vehicle passes through it. Within these two bounds, the choice of ε depends on the following factors.

* A smaller ε provides a better approximation of the search area and its obstacles.
* On the other hand, a 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.

The tiling is partitioned into three subsets:

* Obstacle cells (To) : they are detected online.
* Forbidden cells (Tf ) : create buffer around obstacles
* Allowed cells (Ta) : these are the target cells to cover

## Step 2 : Dynamically Constructed Multi-scale Potential Surfaces (MAPS)

With the Movement of Bot and detection of obstacles, the potential allotment of the maps is dynamic and thus this is useful in online path planning.

These are Multi-level to avoid getting the robot into a place from which it has to back track it’s way out to reach other uncleaned cells.

The Level 0 of the Map is where the boundaries of the map are set initially and the location of the known obstacles. Higher Level of Mapping is used to avoid the bot from getting cornered between cleaned areas. Thus, higher the Level of mapping, better the tendency of the robot to avoid back-tracking. This is termed as reaching a local extrema in the Potential Surface language.

Higher levels of MAPS are used to prevent the local extrema problem. Local Extrema implies no unexplored cells are available in the local neighborhood on Level 0. The potentials of unexplored blocks are adjusted as per the inputs and feedback given the detection system of the robot.

## Step 3 : Supervisory Control Structure

The Exploratory Turing Machine (ETM) is used to control the usage of Machine States.

### Machine States

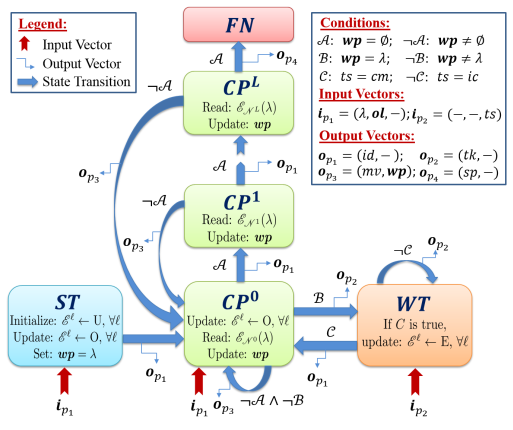
* **The Start State** **(ST)**: start the machine and initialize the MAPS with all e-cells as unexplored.
* **The Computing States (CP):**
  + **CP0**: compute waypoint **wp** using Level 0 of MAPS, and send navigation command cd.
  + **CP1, CP2….CPL** : sequentially used to compute **wp** in case of a local extremum.

Local extremum is when no waypoint is found.

* **The Waiting State (WT)**: wait for the vehicle to complete specific task (e.g., cleaning) in the current

cell, until the status turns to complete.

* **The Finished State (FT):** terminate the operation upon complete coverage



## 

## 

## 

## 

## 

## Operation of the ETM

| **Machine State** | **When is it reached?** | **What does it do?** |
| --- | --- | --- |
| The ST State | As soon as the autonomous vehicle is turned on. | Initialization of the system. |
| The CP0 State | Either after system initialization, or when the current cell has just been tasked and needs a new 01. | Default state to compute for 01 on Level 0 of MAPS. |
| The CP1, CP2….CPL States | When waypoint wp cannot be found in CP0 state. | Sequentially switches to higher levels of MAPS, until wp can be found at some Level l ≤ L |
| The WT State | When the autonomous vehicle reaches the computed wp. | Command the vehicle to perform tasking (e.g., cleaning) in  the current cell. |
| The FT State | When wp cannot be found in CPL  state. | Terminate operation since no unexplored cells are left. |

That is,in state ST , the ETM initializes the MAPS. Since the whole area is initially unexplored, all ε -cells are assigned the state U, thus MAPS are constructed using only the potential field B. Then, the ETM cycles on and between the states CP0 and W T , as follows. In each iteration of state CP0 , the ETM takes input from the autonomous vehicle about the newly discovered obstacle locations and its current position (λ). Then, it moves the head on the tape to λ and updates the MAPS in accordance with the discovered obstacles, and performs the following operations:

i) reads the potentials from the local neighborhood N0(λ ) of λ to compute the new waypoint,

ii) changes the head state to W T if waypoint is reached otherwise stays in CP0 , and

iii) generates an output vector for the vehicle containing the operational command and the new waypoint.

In each iteration of state WT, it receives the task status from the vehicle, and continues to send tasking

command until it is complete. Once the current cell is tasked, it updates the MAPS and returns to the state CP0.

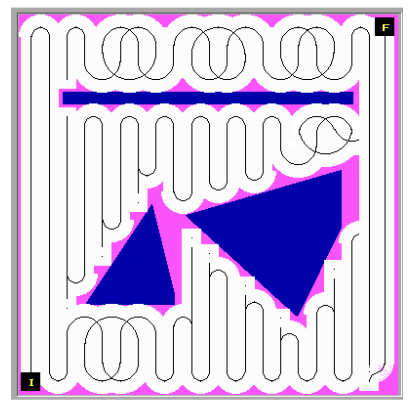
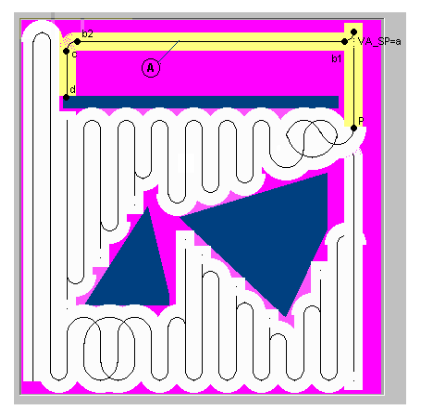
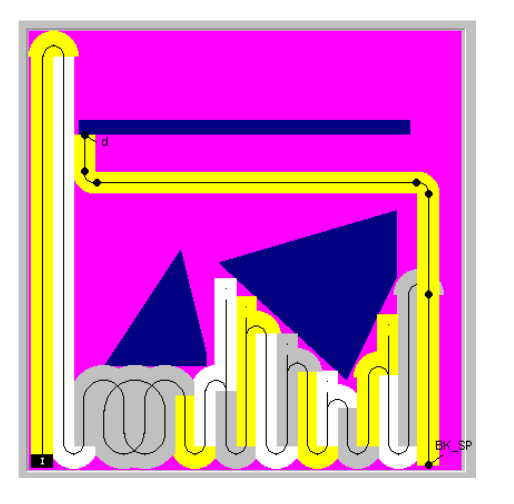
If the head gets stuck in a local extremum in state CP0 , i.e. no waypoint could be found in the local neighborhood at Level 0 of the MAPS, then it switches to CP1 and operates on Level 1. Here it searches for the coarse cell with the highest positive potential in a local neighborhood N1(λ ) to find a waypoint. If no waypoint is found even at Level 1, then it switches to state CP2 and so on until it finds one, then it comes down to state CP0 and continues. If no waypoint is found even at the highest level then the ETM halts in state FN and the coverage is complete.

Here also backtracking occurs at extremas but it is something that can not be avoided in Online Complete Coverage Path Planning.

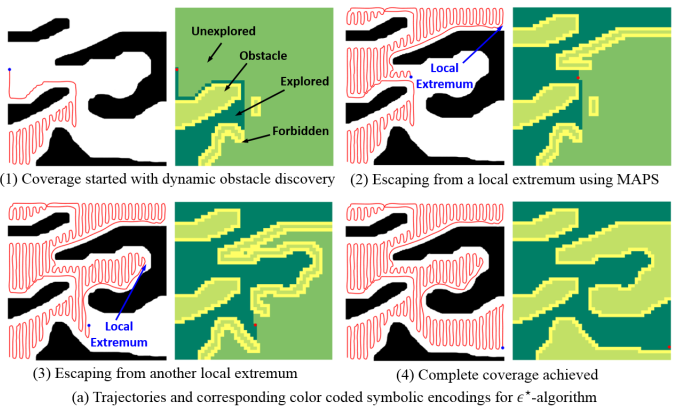
Final Robot Paths

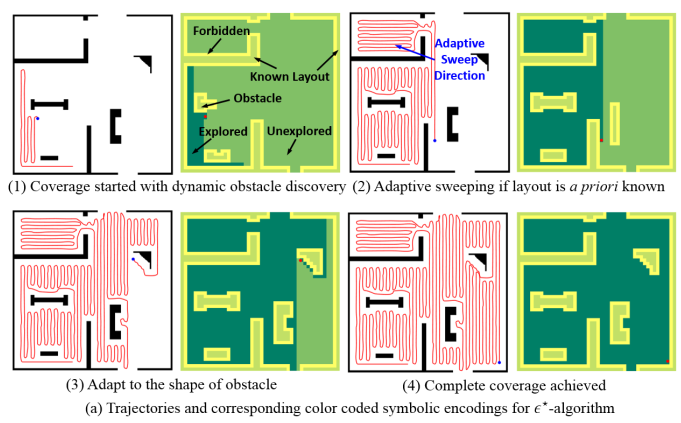
Examples of Final Robot Paths :

**1.Template Based Methodology**



**1.Epsilon Star**

****

****

Cleaning Efficiencies

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.