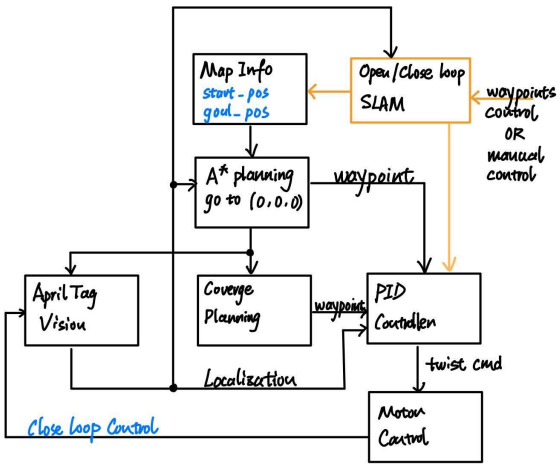
CSE276A HW5 Report

Shusen Lin A59010006, ECE
Vaibhav Bishi A59009958, ECE

1. Architecture

This is the course assignment for CSE276A Introduction to Robotics. The purpose of this project is to implement a coverage planning algorithm in a 2D Apriltag feature based environment. The following figure is the system architecture:



With pre-built map: Shown above with black blocks, with the knowledge of map, we must put the robot at map origin (0,0,0) and then, implement the coverage planning based on Apriltag detection localization.

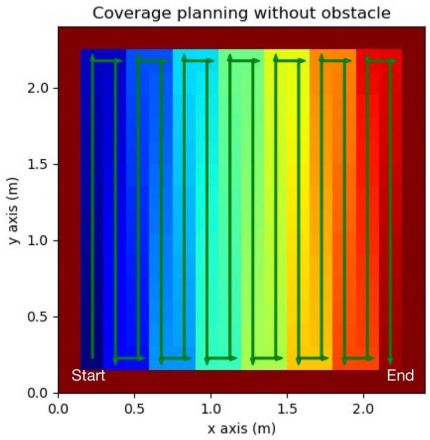
With SLAM and A:* SLAM block is shown in orange, either can be done by using set waypoints or manual control for obtaining the landmark position in world frame. Once several SLAM iterations are done, the robot executes A* algorithm to reach the start point (0,0,0), and then implement the coverage planning based on the constructed map.

By this architecture, SLAM and A* combination has the ability to handle the avoidance of obstacles, and regenerate the map if the environment is changed.

2. Modules

Environment Representation: Same as our previous work, we are going to represent the environment with a gridmap, since gridmaps have the ability to cover all the places in an environment. The goal of coverage planning is to cover all the free cells of the map. Also, the gridmap is convenient for planning either A* or coverage planning.

Coverage Planning: For the coverage, we are using the **Boustrophedon Planner**. The reason why we choose this is because Boustrophedon Cell Decomposition has the strength that allows for more diverse, non-polygonal obstacles within a configuration space. The coverage for an free rectangular space is as follows:

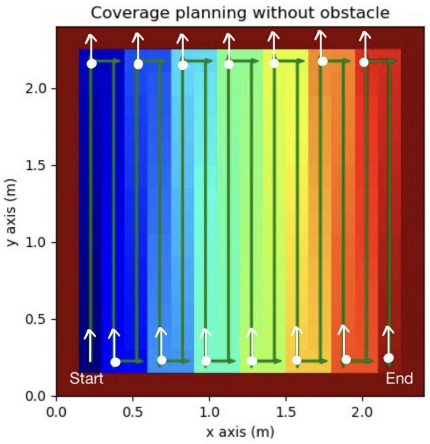


Our map decomposition is based on the world scale and resolution, the number of cells is equal to:

$$cells\ amount = \frac{row\ length(m)}{RES(m)} * \frac{col\ length(m)}{RES(m)}$$

Since the width for the Mega bot is around 0.15 meters, The resolution we choose here is *RES=0.2m*.

Waypoints Choice: Based on the PID controller property, physical limitation of the robot, and the experiment observation, the robot moving straight (no angular variation) returns us a better coverage behavior. Hence, we choose the waypoints as follow:



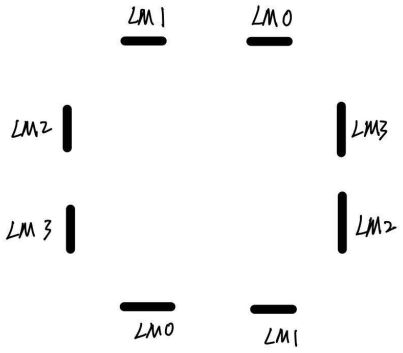
The waypoints may be different as shown in the video.

Boustrophedon Planner Details: Our algorithm has the preference to search the space from the left side.

- The planner will start the **block-search** form the original point, and will **up-search** first until we hit any obstacle or left cell is unsearched. In the first case, it will start one unit **right-search** or it will **left-search** until it hits an obstacle.
- After the **left-search** is done, it will repeat the **up-search** again.
- After the **right-search** it will do **down-search** with the same logic as **up-search** until it hits any obstacle.
- If no more reachable cell is nearby, the **block-search** is done.
- The planner will find the closest unsearched point as the next original point **until all the cells are searched.**

Localization: For localization while following the waypoints generated by the planner, instead of taking a single transformation from the first Apriltag that the robot notices, we take the average location of the robot from all the tags that the robot can read. This drastically improves the localization and the jittery motion is considerably reduced.

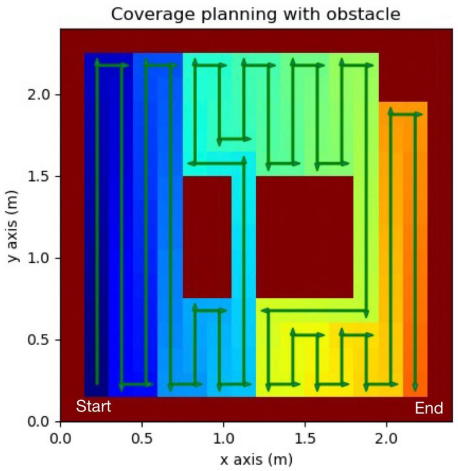
Ambiguity Checker: The landmarker set up we use as shown below:



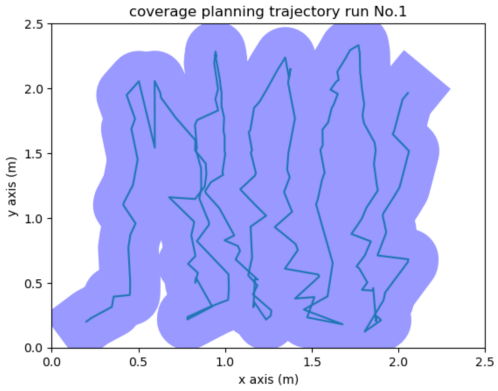
We used eight markers in total which four of them are unique. The strategy we applied for ambiguity check is instead of setting the landmarker static transformation in the beginning. Whenever a robot sees a landmark, it will use its current position and transformation from camera to marker to estimate the location of the marker. **And then compare the norm error between for say LM1 up and LM1 down, the marker with the smaller error will be determined and the static transformation will be sent.** (assume no same markers will be seen simultaneously.)

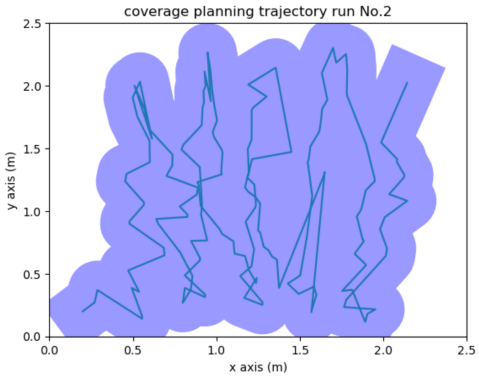
3. General Performance

As shown before, the Planner finished the free space coverage search successfully. For the environment with obstacle, the planner performance is shown below:



We can notice that the Planner successfully covers all the cells within one **block-search.** And the real-time experiment trajectory of coverage is shown below:

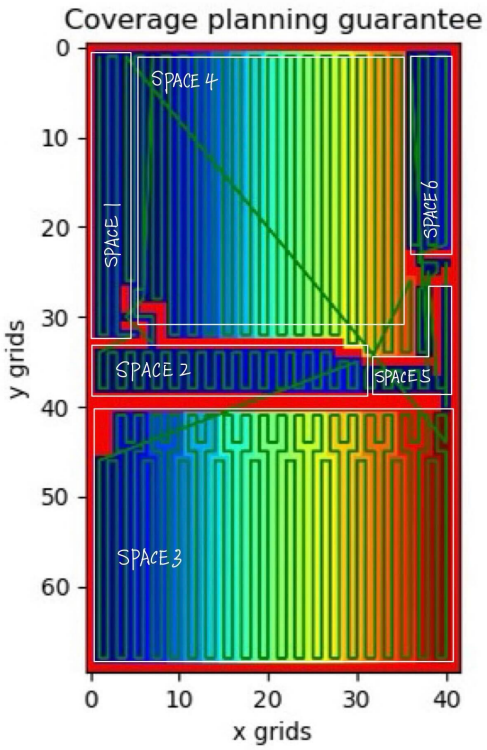




More Complicated Case: With the aim to discuss the guarantee coverage (more details will be mentioned later), we are going to test our planner in a non-polygonal environment with more obstacles and blocks, the performance of as shown below.

From this figure, we can observe that the Planner successfully covers all the cells in the map. It starts the searching from left and up corner SPACE1 and finally finishes the planning on SPACE6. The green line across the SPACEs represents the planner choosing the nearest unsearched point. And we will implement this function via A* in real time.

Since this environment is more maze-like. Our nearest points choice strategy is not smart enough for this scenario. Hence, the searching spaces are not very consecutive but still cover 100% cells.

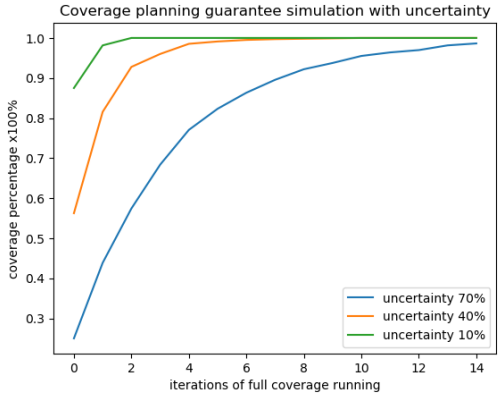


4. Guarantee Discussion

We test the coverage performance guarantee of our Boustrophedon algorithm implementation in simulation by running several iterations of the planner starting from a random position in the gridmap. The premise is that our robot covers all the unit cells as considered by our map, however depending on the cell size we choose and any potential error during motion, the robot may not cover the entire cell area, but rather some parts of it.

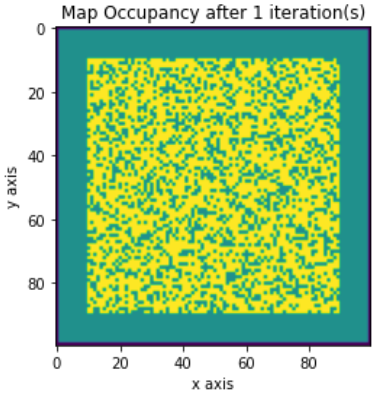
Therefore, for our original map which can be considered coarse by nature since the robot has a finite size, we get a finer map of the same dimensions. Whenever the robot explores one unit cell, we randomly select a patch inside the large unit cell such that only certain coordinates of that region are actually traversed by the robot depending on a threshold value. This process takes place for each of the explored cells until the coverage terminates and the actual density of the traversed area is calculated for this iteration. We then choose another starting point and rerun the coverage planner. After every iteration, the actual area covered by the robot increases and this information for density with respect to number of coverage runs is stored.

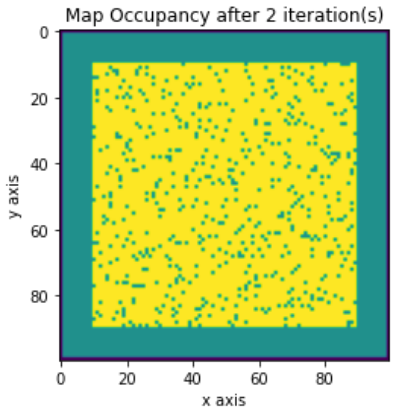
The plot for density of covered area against number of runs for a square map with no obstacles is shown below, where we take different threshold values which refers to the uncertainty in the area covered per unit cell.



We can infer from the plot above that our algorithm guarantees convergence for full-area coverage for an open map at different uncertainty values.

At 70% uncertainty, the plots for the occupancy map after 1, 2 and 3 iterations are shown below:



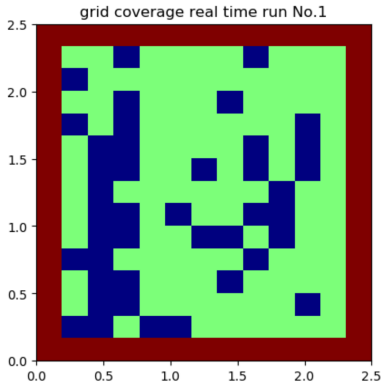


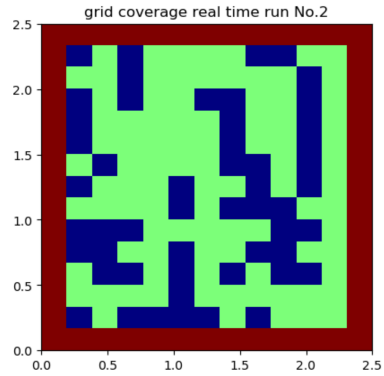
*The **teal** cells are the unexplored or inaccessible regions (walls) and the **yellow** cells are the explored regions. It should be noted that the thick **teal boundary** (0.2 m wide) around the accessible region indicates the walls.*

It is evident from the plots that the density of explored regions increases with the number of runs until we achieve convergence eventually.

In case of a more complicated map which is non-convex or has multiple obstacles, we can use the coverage planner to traverse maximum possible area as permitted by our planner and then use A* to generate the shortest path from the current robot location to the bottom-left corner of the next unexplored patch and move our robot there. Once we reach that location, we run coverage for that patch and repeat this process if necessary. For this scenario, we can take several reruns and we end up with the same conclusion as our coverage increases until full convergence as demonstrated in the simple case.

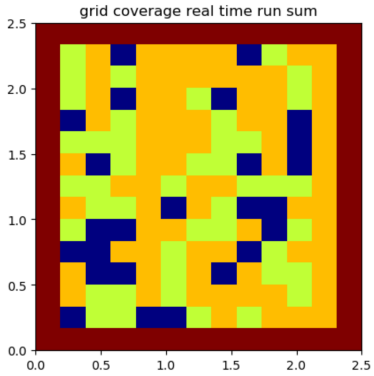
Real Time Experiment of Guarantee: We also test the above theory in real time. We collected the trajectory data and transferred it to the occupancy map based on its resolution. The result of two runs are shown below:





*The **red** boundary represents the configuration space boundary with width 0.2 meters, **green** cells represent the coverage space, and **blue** cells represent the uncoverage space.*

If we combine the grid occupancy map, we can get the same trend as the theory mentioned:



*The **red** boundary represents the configuration space boundary with width 0.2 meters, light **green** and*

orange *cells represent the covered space, and* ***blue*** *cells represent the uncovered space.*

We can conclude that, as the iteration increases, the coverage rate will approach 100%.

5. SLAM and A*

We implement SLAM for building the map and for localization of the robot using the Kalman Filter for prediction and update steps. The map includes the location and orientation of each of the Apriltags which are used as landmarks.

The motion and observation models and the prediction and update steps for the Kalman Filter are outlined below.

Motion model: $\mathbf{x}_{t+1} = F\mathbf{x}_t + G\mathbf{u}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(0, W)$

Observation model: $\mathbf{z}_t = H\mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(0, V)$

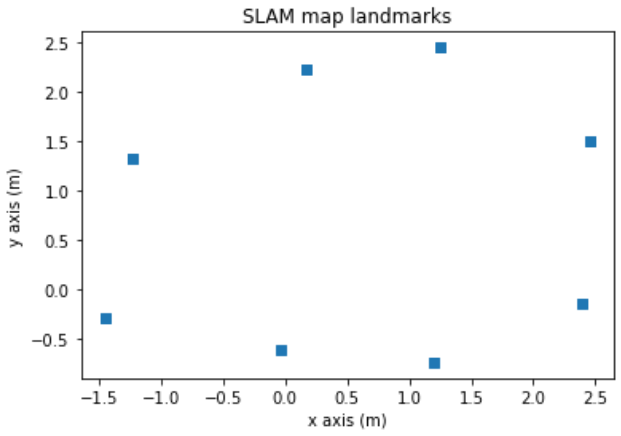
Prior: $\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$

Prediction:
$$\mu_{t+1|t} = F\mu_{t|t} + G\mathbf{u}_t$$
$$\Sigma_{t+1|t} = F\Sigma_{t|t}F^\top + W$$

Update:
$$\mu_{t+1|t+1} = \mu_{t+1|t} + K_{t+1|t}(\mathbf{z}_{t+1} - H\mu_{t+1|t})$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1|t}H)\Sigma_{t+1|t}$$

Kalman Gain: $K_{t+1|t} := \Sigma_{t+1|t}H^\top \left(H\Sigma_{t+1|t}H^\top + V\right)^{-1}$

The SLAM map we obtain with the landmark locations is shown below:



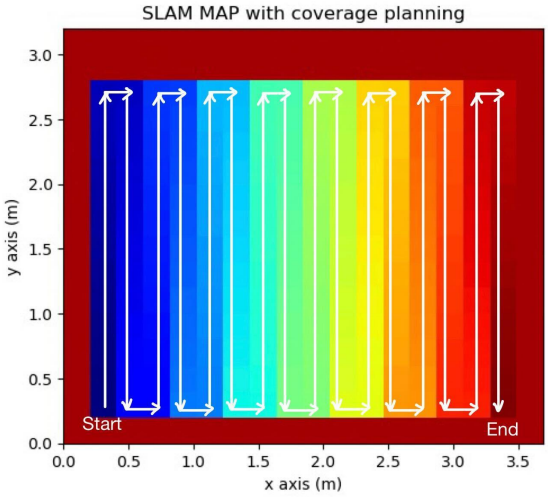
The landmark locations obtained are as follows:

[(2.39546109 m, -0.14429049 m),
(2.458945660 m, 1.49745670 m),
(1.195821120 m, -0.74576878 m),
(1.243553235 m, 2.45455325 m),
(0.530087700 m, 2.23419459 m),
(-1.23412456 m, 1.32187435 m),
(-1.45321801 m, -0.2860211 m),
(-0.04124464 m, -0.4310783 m)]

Based on the map generated by SLAM, we use these landmark locations to create a square gridmap by considering the minimum possible distance between the markers that would give a valid rectangular environment.

We start our planner by moving the robot to the appropriate start position at (0, 0, 0) using Astar. The A* algorithm uses a combination of cost-to-arrive values and heuristic searching to obtain the shortest path. So we feed the path coordinates to the PID controller and move the robot to the required position.

Once the robot is at (0, 0, 0), we implement the coverage planner on the SLAM map to obtain the following trajectory behavior:



Since our map has no obstacles, the robot simply sweeps the whole area by moving from one end to the other. We choose the waypoints in a way that we only care about the start and the end of a linear movement from one end to the other, and the point at which the robot turns right to go in the opposite direction.

6. Discussion

We are successfully able to achieve autonomous navigation and perform Boustrophedon coverage planning with convergence guarantee for the robot in case of both known-map and SLAM generated map with A*. In order to improve the SLAM performance and reduce localization error, we can use the Extended Kalman Filter to account for the non-linearization of the model. We can also improve the awareness of the robot during navigation or while implementing coverage by simultaneously running

SLAM and planning online every few time steps. This can help us deal with any unexpected obstacle that the robot may encounter in the configuration space. We can simply set the newly observed obstacle as inaccessible in the map and update the map. Then we can have a newly planned path going around the obstacle if it is stationary for some amount of time. If the obstacle is mobile, like a person passing through the region, we can get the robot to wait for some time and update the map again. If there is no significant change in the accessibility of that region, we can simply follow the previously planned route.

7. Video Link

https://drive.google.com/file/d/1zD3UddKC_AJyw-Dpal9EzexaNpThAgpF/view?usp=sharing