CSE 276A - Introduction to Robotics

HW5

Drishti Megalmani A59001657
Sai Ashritha Kandiraju A59001630

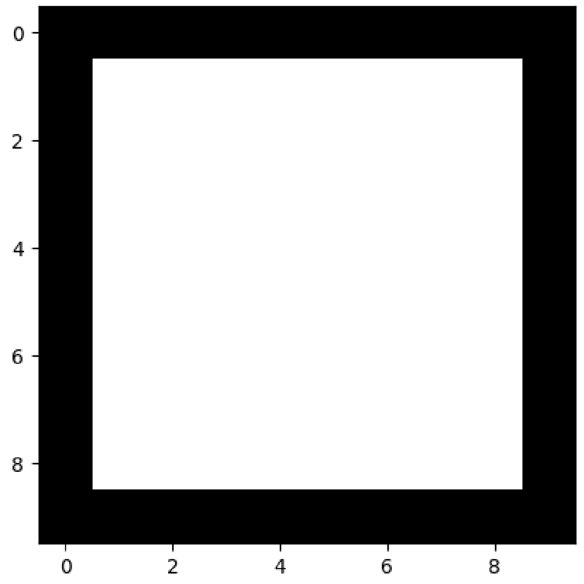
In this homework, we aim to implement a coverage path planning algorithm to enable our robot to navigate through an environment and provide a level of coverage of the area.

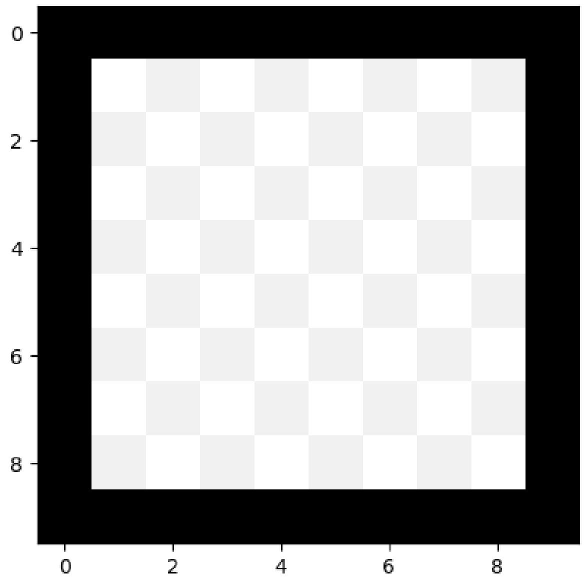
METHODOLOGY

We set up an environment in a 7ft x 7ft area with 8 distinct landmarks and 4 duplicates (to introduce ambiguity in the map). The map has 3 apriltags on each side. We assume full knowledge of the map and the landmarks. For coverage we implement the offline version of Spanning Tree Based Coverage (STC) path planning algorithm [1].

ALGORITHM

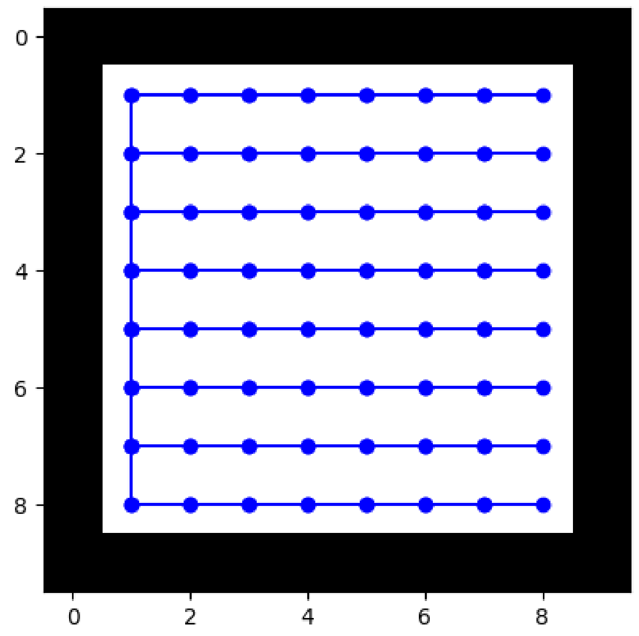
Input - The input to the offline STC algorithm is a geometrical description of the environment with boundaries and obstacles. We first divide the entire coverage area into cells of size 2D x 2D. Each cell contains 4 sub-cells of size D x D. We then discard all the cells that are partially/fully covered by obstacles. Figures below show the map with boundaries (left) and the coverage area divided into cells of size 2D (right)



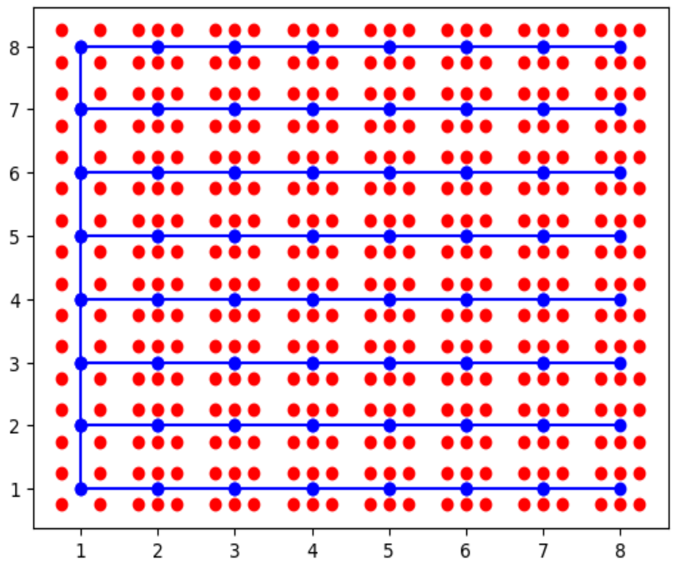


Graph Representation - We then represent these cells as a graph G. The nodes of the graph are the center points of the cells and the edges of the graph connect centers of adjacent cells in the horizontal and vertical direction.

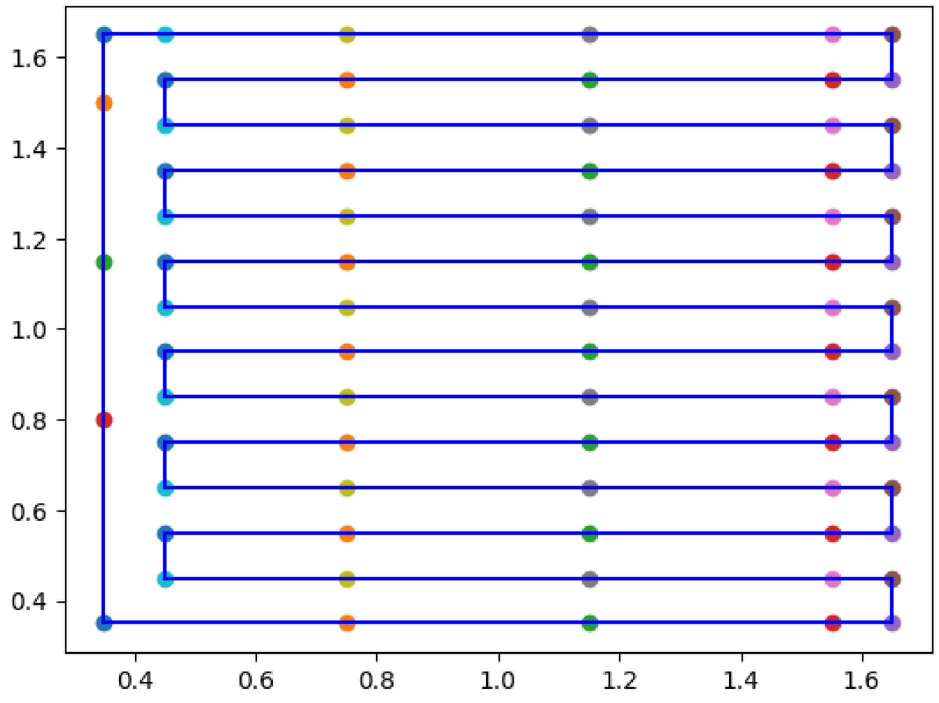
Spanning Tree Construction - We pick a starting cell S (from where the robot starts navigation) and construct a minimum spanning tree T for the graph G using Kruskal's minimum spanning tree construction algorithm. In this case, all edges are considered to have an equal weight of 1.



Circumnavigation - After we have the spanning tree T, we construct a path which *circumnavigates* the spanning-tree along the counterclockwise direction. To obtain the points in this path, for each node in the spanning tree, we find the points around it (in all 8 directions) at a small distance of 0.05m given that the points don't lie on the spanning tree edges as shown in the image below. We got a total of 368 points, but finally subsampled 75 waypoints for the final path to be used for our PID controller.



Coverage - We know that the minimum spanning tree of a graph is an undirected graph that connects all the nodes without any cycles with the smallest possible total edge weight. So when we circumnavigate the spanning tree, we can guarantee complete coverage of the map (performance guarantees discussed later). We halt the navigation when we encounter the starting subcell S again. The path for coverage is shown below. Here, we see that the defined path covers the distance from 0.35m to 1.65m in both x and y axes. Our original map size is of 2x2m in which 0.2m is taken as the space for april tag box placements on both the sides of the grid as shown in the grid image in the Input section. This leaves an internal space ranging from 0.2 to 1.8m in both axes. The size of the robot is about 0.17x0.13m. Due to this and the april tag boxes placed on the boundary, we have considered a safe distance around the boundary of the grid to avoid collisions. This explains why our path covers only from 0.35m to 1.65m in both directions.



MOTIVATION

In the beginning, we tried different approaches to cover the map area. We experimented with patterns like zig-zag and spiral traversal which helped us obtain complete coverage. However, we observed that traversing in a fixed pattern needed us to make certain assumptions regarding the map.

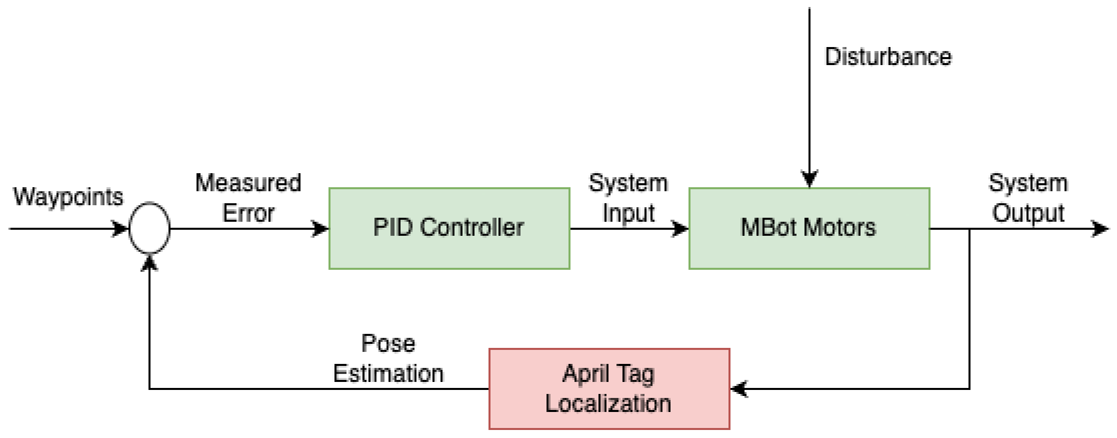
1) We had to assume that there were no obstacles present within the map
2) We had to make assumptions regarding the shape of the map. These patterns only worked for a map which was a square/rectangle

To avoid this, we decided to represent the map as a grid of cells. The cells would then be marked as accessible or not based on the obstacles/boundaries. This also means that the designed coverage algorithm would work for a map of any shape.

To obtain complete coverage of the map, we decided to implement the Spanning Tree Based Coverage (STC) path planning algorithm. This algorithm first constructs a spanning tree for the cells represented as a graph and then circumnavigates this tree in a clockwise/counterclockwise manner until we get back to the starting position. Since the constructed spanning tree lays out a path that connects all 2D x 2D cells in the graph and circumnavigating this path ensures that we visit all the D x D subcells, we can thus guarantee complete coverage using this algorithm.

IMPLEMENTATION DETAILS

Architecture



We used the same framework as in the HW2 solution with some changes to make it compatible with the april detection node (The HW2 solution provided didn't really work without modifications due to the discrepancy in publish and subscribe messages of the tf.transform).
In the planner node (hw5_solution), we use a listener to obtain the measurement from april tags and transform this to get the current pose of the robot in the world. The planner node then publishes a twist message to the mpi control node which moves the motors.

Spanning Tree Construction

To construct the minimum spanning tree for the graph G, we used NetworkX's *`minimum_spanning_tree(G[, weight])`* which computes the minimum spanning tree using Kruskal's algorithm.

Handling Same Tag Ambiguity

Since the map has the same tags placed at multiple locations, choosing the right pose matrix for the transformations and localization is essential.

To handle this ambiguity, we followed these steps:

- The *`current_state`* of the robot (from the open loop update step or last known current_state after localization) corresponds to *`wTr`*.
- The measurement from the april tag detection will give us `cTa`.
- We already know the transformation matrix `rTc`.
- We perform the following transformations:

```
rTa = rTc x cTa
wTa = wTr x rTa
```

- `wTa` april tag in the world coordinates which is nothing but our values in the april tag pose matrices.
- For each tag id, we maintain a list of possible pose matrices in the map based on each occurrence. We find the error in terms of the euclidean distance between the wTa's translation coordinates and the coordinates of each tag in the april_tag pose matrices. The tag with least error, is our required tag. Here, we are basically finding the error between our estimate of the april tag in the world with the actual pose of that april tag.

PERFORMANCE GUARANTEE

We analyze the performance guarantees of this algorithm using the following theorems

Lemma 1 - The STC algorithm circumnavigates all cells of the map

Proof - In the STC algorithm we represent the map cells as a graph where each cell of size 2D x 2D is a node. The algorithm first constructs a minimum spanning tree of the graph and then circumnavigates the tree. By definition, a minimum spanning tree is an undirected graph that connects all nodes of the graph without any cycles. Since a spanning tree guarantees the inclusion of all vertices of the graph, this proves that the STC algorithm circumnavigates all cells of the map.

Lemma 2 - The STC algorithm covers all the subcells of a particular cell in the map in a counterclockwise but not necessarily contiguous order

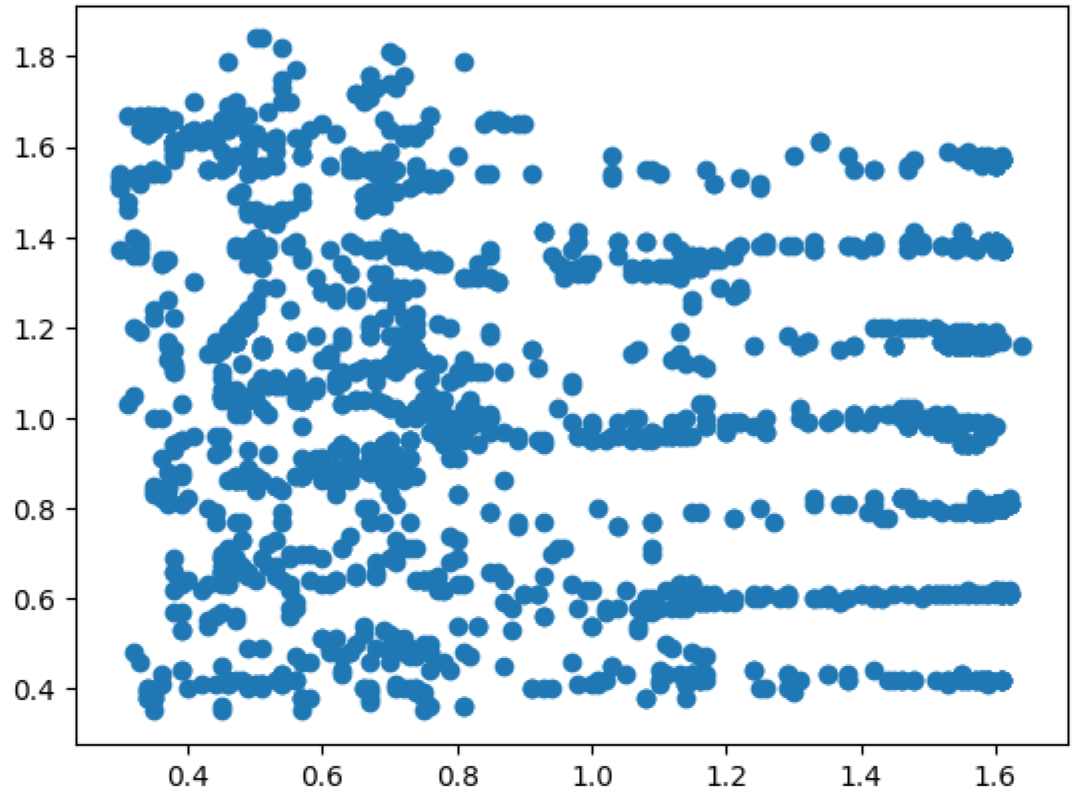
Proof - While traversing a node (say X), if it has no neighbors (leaf node) the robot circumnavigates the cell in the counterclockwise direction. If the node X has neighbors, the algorithm selects its *first* neighbor (say Y) in the counterclockwise direction, circumnavigates all the subcells of Y before visiting the rest of the subcells of node X. This shows that the order in which subcells of a cell are visited is always counterclockwise. By definition, to circumnavigate a node means to go around the node in all directions. Thus, while obtaining the path for circumnavigation, we picked all the 4 subcells belonging to a particular cell. This proves that the STC algorithm covers all the subcells of a particular cell in the map during circumnavigation.

Lemma 3 - The STC algorithm covers every cell which is accessible in the map in an optimal manner without repetitive coverage

Proof - From Lemmas 1 and 2, we can argue that the STC algorithm visits all cells and its subcells before returning to the starting subcell. From Lemma 2 it is also clear that since we enforce a counterclockwise order of traversal for the subcells, we will never revisit a subcell. This shows that there would not be any repetitive coverage. Thus, the STC algorithm covers every cell which is accessible in the map in an optimal way. This proves that the coverage using this algorithm is 100%.

RESULTS AND ANALYSIS

We started from point (0.35, 0.35) and performed a motion with the waypoints derived from our algorithm. The robot's orientation was 0 radian. The robot performed mainly forward, backward and sliding operations thereby taking feedback from the april tags mainly on the right side of the grid. The plot below displays the actual trajectory points the robot followed while taking continuous feedback from the april tags in the vicinity. We see that the path is much towards the right side of the point close to the april tags it was taking feedback from and the trajectory is jerky. This can be due to feedback from multiple tags at a time, possible errors in localizations and possible human errors in placement of the april tags.



CHALLENGES

1. Due to the sheer number of the april tags, it was time consuming to accurately place each tag in the defined position in *pose_ma.*
2. We attempted motion with the waypoints by our algorithm initially with a rotation each time we reached the boundary, but the rotation was not always accurate resulting in a non-reliable solution. Hence, we switch to only forward, backward and sliding motions.
3. We also had to slightly tune the PID values again.

CONTRIBUTION

Both of us contributed equally on all sections of the homework (from researching algorithms, design thinking, implementation, testing and report).

REFERENCES

[1] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. Technical report, Dept. of Mechanical Engineering, Technion, Israel Institute of Technology, December 1999.