RBE 550 -Motion Planning
HW3-Advanced Search algorithms

Krishna Sathwik Durgaraju

Code and algorithm explanation:

Q1. Explain in your own words, how does D* replan a path by updating the cost?

D*:

With the given Initial static map D* calculates the feasible path like Dijkstra but it adds the nodes in the open list from Goal to start position.

Each node is assigned with different state tag -NEW, OPEN, CLOSED.

D* replan a path by updating the cost with the following steps.

1.Process state

This function calculates the optimal path to the goal. Initially set heuristic of the goal be zero and insert it into the OPEN list. Repeatedly call this function until the node removed from the list. The process state starts by selecting the minimum node from the open list then finds the neighbours of this node. Compare the minimal K value from the open list with the heuristic of the selected node to determine RAISE and LOWER states. If h>k, then raise state and h<k assigns as lower state. Finally, the minimum k value is updated.

2.Prepare repair

If any of the neighbour node is a dynamic obstacle cost from the adjacent node to the dynamic obstacle node should be modified. The Modify_cost function updates cost from this neighbour node to all this neighbour's neighbours.

3.Repair Replan

Replan the trajectory until no better path is possible or the open list is empty. Call process_state until it returns k_min >= h(Y) or open list is empty. The cost change will be propagated.

Steps:

1. Initially, the heuristic of goal is set to zero.
2. Goal node is added to the open node and basic path finding algorithm such Dijkstra or A* is used to find a feasible path from goal to the start.
3. Once the path is found, the robot is moved along the path while taking the sensor inputs from the sensors. At each state, the nodes are repaired based on the affected nodes and paths and parents of each affected node is evaluated.
4. This process is repeated until the robot reaches the goal if a feasible path is found.

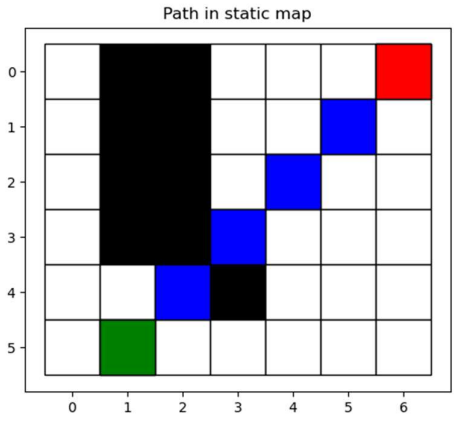
Q2) Why does D* can replan faster than A* or Dijkstra?

- In Dynamic environment D* repairs and replans the path locally whereas A* and Dijkstra finds the path by repeating the entire process.
- D* takes advantage of previous calculated path if the obstacles in the map changes which makes it computationally less expensive.
- Since A* and Dijkstra calculates from the beginning, they are computationally expensive.
- D* is used for dynamic environment for faster replan whereas A* and Dijkstra are used for static environment.

Results:

Map-1:

Static Map



1-Dynamic Obstacle is detected (3,3)

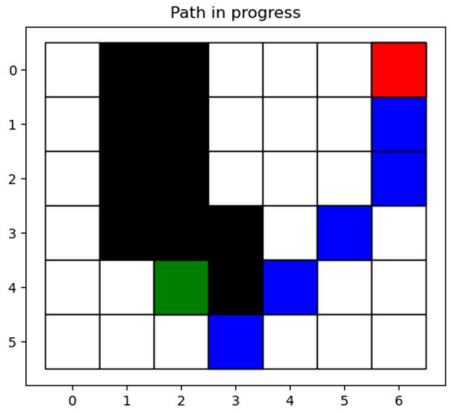
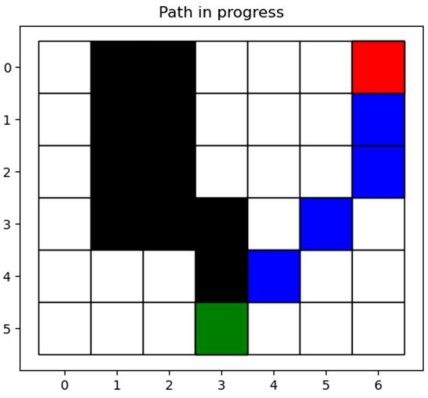


Figure 2 New path is calculated



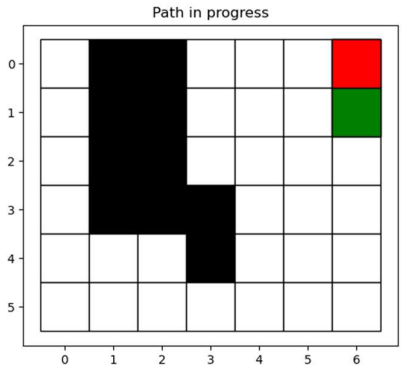
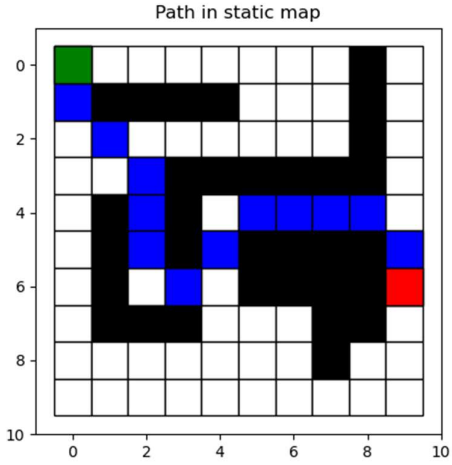


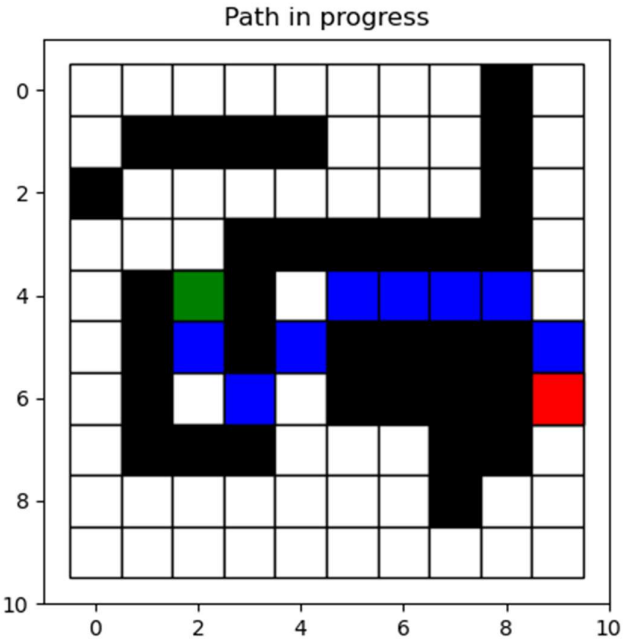
Figure 1 Reached Goal

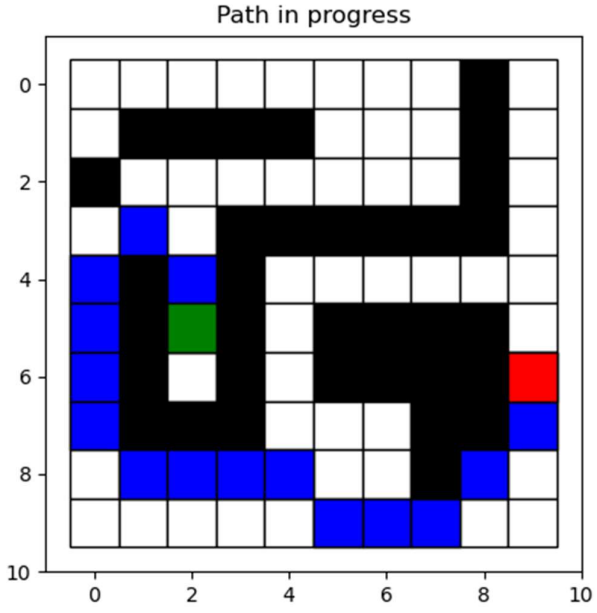
Map-2:

Static map

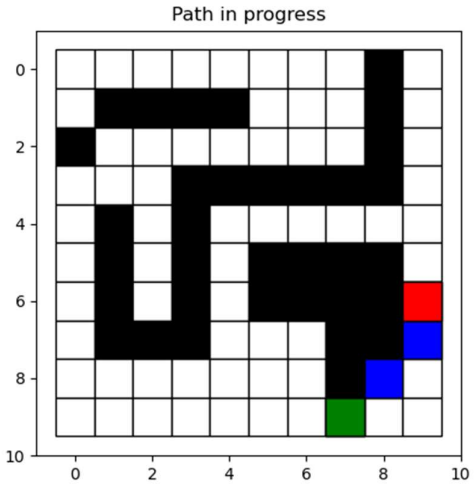


Dynamic Obstacle at (3,6):





Dynamic Obstacle at (10,7):



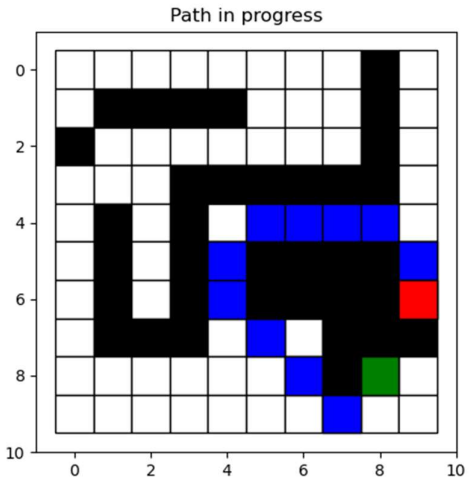


Figure 4 New path is calculated when an obstacle is detected

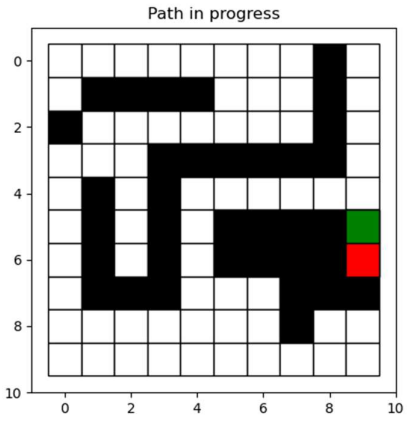


Figure 3 Final Goal is reached in Dynamic map

Map 3:

Figure 5 Dynamic Obstacles makes no path to the goal

Q3) What is the key differences between regular RRT* and informed RRT*?

Informed RRT*	RRT*
Sampling is performed in Local region (ellipsoid subset)	Sampling is performed on entire workspace.
Focused sampling is performed once initial path is found.	Irrespective of Path, sampling will be performed.
It provides Optimal path with a smaller number of iterations	It provides optimal path with a greater number of iterations
It generates Tree structure with a smaller number of nodes because of focused sampling	More number of nodes generated because of explored sampling.
Path exploration is controlled in finite space.	High exploration on the sample space

Implementation of informed RRT*:

Pseudo code used to implement Ellipsoid Sampling:

Algorithm 2: $\texttt{Sample}\left(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\max}\right)$

```
if c_max < ∞ then
    c_min ← ||x_goal − x_start||_2;
    x_centre ← (x_start + x_goal) /2;
    C ← RotationToWorldFrame(x_start, x_goal);
    r_1 ← c_max /2;
    {r_i}_{i=2,...,n} ← (sqrt(c_max^2 − c_min^2)) /2;
    L ← diag{r_1, r_2, ..., r_n};
    x_ball ← SampleUnitNBall;
    x_rand ← (CLx_ball + x_centre) ∩ X;
else
    x_rand ~ U(X);
return x_rand;
```

Rotation to World frame:

1. Transverse axis in the world frame-a1
2. first column of the identity matrix-I1
3. Matrix M = outer product of the a1 and I1
4. Singular value Decomposition to find eigen vectors
5. Calculate Rotation Matrix for Hyper ellipsoid frame to the world frame

4.By showing and comparing the results of RRT* and informed RRT*, what are the advantages of using the latter?

Results:

```
In [16]: runfile('C:/Users/krish/Desktop/Motion_Planning/01_Assignments/06_Advanced/Advanced Search Algorithms/informed_RRT/main.py',
wdir='C:/Users/krish/Desktop/Motion_Planning/01_Assignments/06_Advanced/Advanced Search Algorithms/informed_RRT')
********* RRT Star *******
Iterations = 500 Cbest = 283.4082892961946
Iterations = 1000 Cbest = 280.78825192347415
Iterations = 1500 Cbest = 276.5091508358463
Iterations = 2000 Cbest = 276.5091508358463
RRT star: It took 1445 nodes to find the current path
The path length is 264.93
*********Informed RRT Star*******
Iterations = 0 Cbest = 0
Iterations = 500 Cbest = 266.5341507971448
Iterations = 1000 Cbest = 255.5003379727665
Iterations = 1500 Cbest = 251.45117262857983
Iterations = 2000 Cbest = 251.32803837377676
Informed RRT star:It took 1135 nodes to find the current path
The path length is 251.33
```

Figure 6 Results of Path convergence over iterations

RRT * Output:

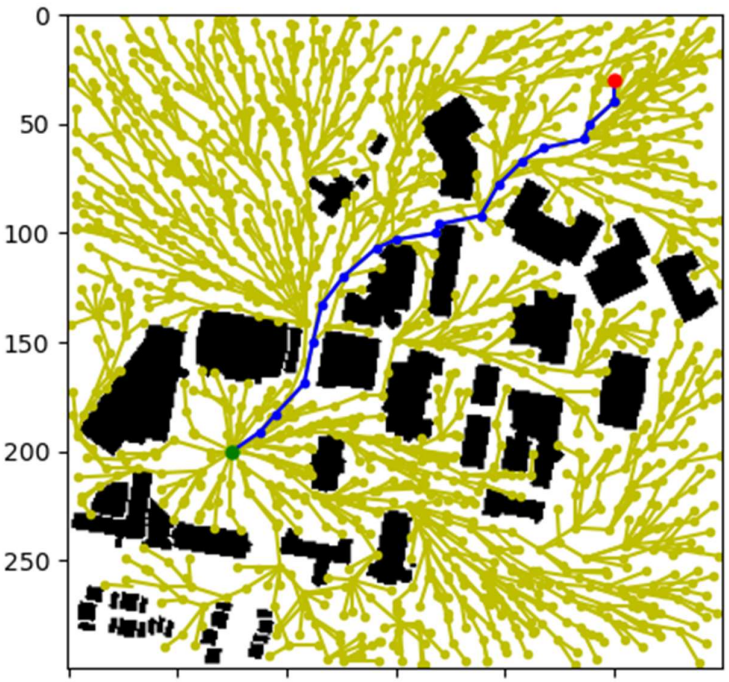


Figure 7: RRT Path - 1445 nodes and Path length =264.93*

Informed RRT* Result:

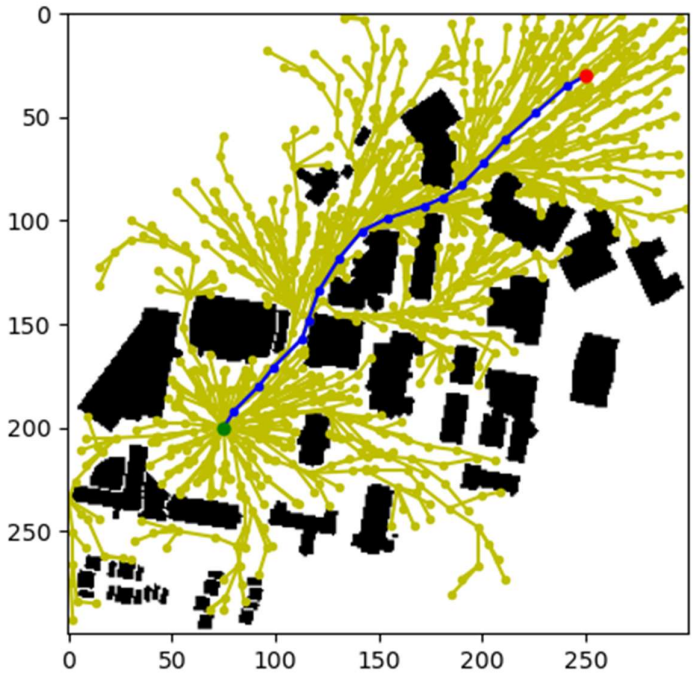


Figure 8: Informed RRT Path - 1135 nodes and Path length = 251.33*

Advantages of Informed RRT* over RRT*:

From Figure-1, we can conclude that

1.Informed Search converges faster with less iterations -High Speed

RRT* -1000 iterations – C best = 280

Informed RRT* -1000 iterations – C best = 255

2. Informed RRT* results in Low-cost path with fewer number of nodes which makes use of less memory.

For the 2000 iterations,

RRT* produces 1445 nodes with a path length

Informed RRT* produces 1135 nodes

3.From Path length we can conclude that Informed RRT provides converging Optimal path.

References:

Implemented code by understanding concepts from below link

1. https://www.cs.cmu.edu/~motionplanning/lecture/AppH-astar-dstar_howie.pdf