

# Comparative Analysis of Path Planning Algorithms


Akash Patel

Robotics Systems 1

Final Report

Robotic path planning plays a key role in enabling autonomous systems to navigate through structured or unstructured environments efficiently. In this report, we explore three algorithms—Distance Transform (Dxform), D\*, and Probabilistic Road Map (PRM)—to identify their respective advantages, limitations, and practical applications. Using a tic-tac-toe-inspired occupancy grid as the environment, the robot starts at the middle-top position and sequentially visits each outer square. The implementation of the Dxform algorithm is carried out in MATLAB, with path visualization and timing comparisons being conducted for all algorithms. Additionally, hardware-based path execution is tested to validate the performance of one algorithm in a real-world scenario.

The Probabilistic Road Map (PRM) algorithm adopts a sampling-based approach by randomly selecting points in the free space and connecting them to create a graph of collision-free paths. PRM is particularly effective in complex, high-dimensional spaces where grid-based approaches are inefficient. For the tic-tac-toe grid, PRM successfully generates paths but requires careful tuning of the sampling density to achieve smooth and efficient results. Unlike Dxform and D\*, PRM does not guarantee deterministic outputs, as the paths depend on the randomly sampled points. The execution time for PRM is generally longer, especially when the sampling density is increased to improve smoothness.



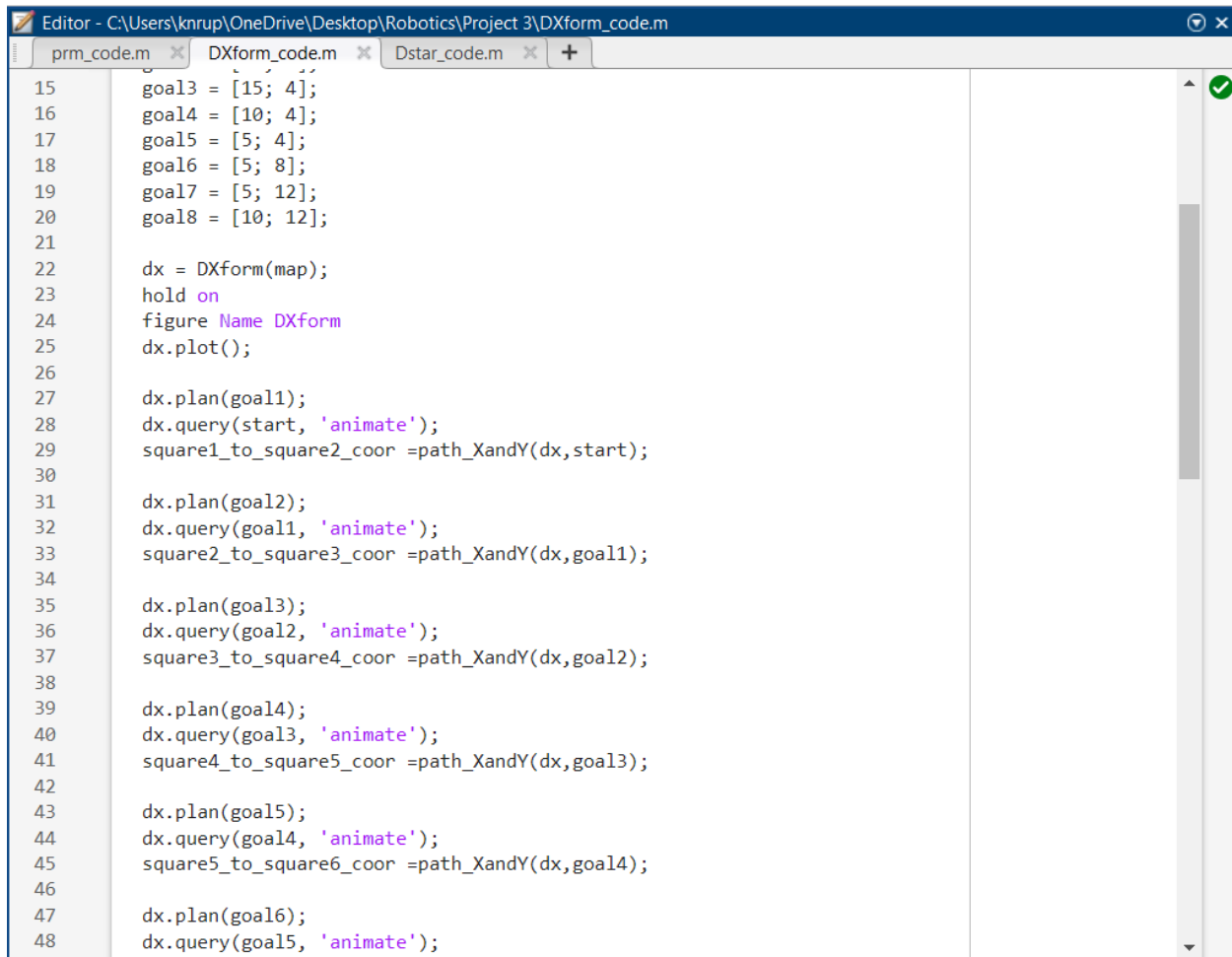
```

12 goal2 = [15; 8];
13 goal3 = [15; 4];
14 goal4 = [10; 4];
15 goal5 = [5; 4];
16 goal6 = [5; 8];
17 goal7 = [5; 12];
18 goal8 = [10; 12];
19
20 prm = PRM(map);
21 prm.plan()
22 prm.plot()
23 path1=prm.query(start,goal1)';
24 path2=prm.query(goal1,goal2)';
25 path3=prm.query(goal2,goal3)';
26 path4=prm.query(goal3,goal4)';
27 path5=prm.query(goal4,goal5)';
28 path6=prm.query(goal5,goal6)';
29 path7=prm.query(goal6,goal7)';
30 path8=prm.query(goal7,goal8)';
31
32
33 plot(path1(1,:),path1(2,:),'Color', 'y', 'LineWidth', 2)
34 plot(path2(1,:),path2(2,:),'Color', 'y', 'LineWidth', 2)
35 plot(path3(1,:),path3(2,:),'Color', 'y', 'LineWidth', 2)
36 plot(path4(1,:),path4(2,:),'Color', 'y', 'LineWidth', 2)
37 plot(path5(1,:),path5(2,:),'Color', 'y', 'LineWidth', 2)
38 plot(path6(1,:),path6(2,:),'Color', 'y', 'LineWidth', 2)
39 plot(path7(1,:),path7(2,:),'Color', 'y', 'LineWidth', 2)
40 plot(path8(1,:),path8(2,:),'Color', 'y', 'LineWidth', 2)
41
42 L(1) = Link ('d',2, 'a', 10, 'alpha', 0,'offset',0);%deg2rad(18.435));
43 L(2) = Link ('d', 0.6, 'a',11, 'alpha', 0,'offset',0);%deg2rad(19));
44 L(1).qlim = [0 pi]; % Joint 1: 0 to pi
45 L(2).qlim = [pi/2 3*pi/2]; % Joint 2: pi/2 to 3*pi/2

```

The Distance Transform (Dxform) algorithm is a deterministic grid-based planning method that precomputes a cost map reflecting the distances from each grid cell to the nearest obstacle. It is particularly well-suited for environments with static obstacles, where the map remains unchanged. By following a gradient derived from the cost map, the robot determines the optimal path to a goal. The MATLAB implementation provided in the script NEW\_DXform.m demonstrates this algorithm by generating smooth, sequential paths from the starting position to all the outer squares of the grid. The paths are visually represented with clear trajectories, ensuring the robot avoids obstacles

effectively. Dxform is computationally efficient for smaller grids but lacks adaptability when obstacles appear dynamically.



```
Editor - C:\Users\knrup\OneDrive\Desktop\Robotics\Project 3\DXform_code.m
prm_code.m  DXform_code.m  Dstar_code.m  +
15     goal3 = [15; 4];
16     goal4 = [10; 4];
17     goal5 = [5; 4];
18     goal6 = [5; 8];
19     goal7 = [5; 12];
20     goal8 = [10; 12];
21
22     dx = DXform(map);
23     hold on
24     figure Name DXform
25     dx.plot();
26
27     dx.plan(goal1);
28     dx.query(start, 'animate');
29     square1_to_square2_coor =path_XandY(dx,start);
30
31     dx.plan(goal2);
32     dx.query(goal1, 'animate');
33     square2_to_square3_coor =path_XandY(dx,goal1);
34
35     dx.plan(goal3);
36     dx.query(goal2, 'animate');
37     square3_to_square4_coor =path_XandY(dx,goal2);
38
39     dx.plan(goal4);
40     dx.query(goal3, 'animate');
41     square4_to_square5_coor =path_XandY(dx,goal3);
42
43     dx.plan(goal5);
44     dx.query(goal4, 'animate');
45     square5_to_square6_coor =path_XandY(dx,goal4);
46
47     dx.plan(goal6);
48     dx.query(goal5, 'animate');
```

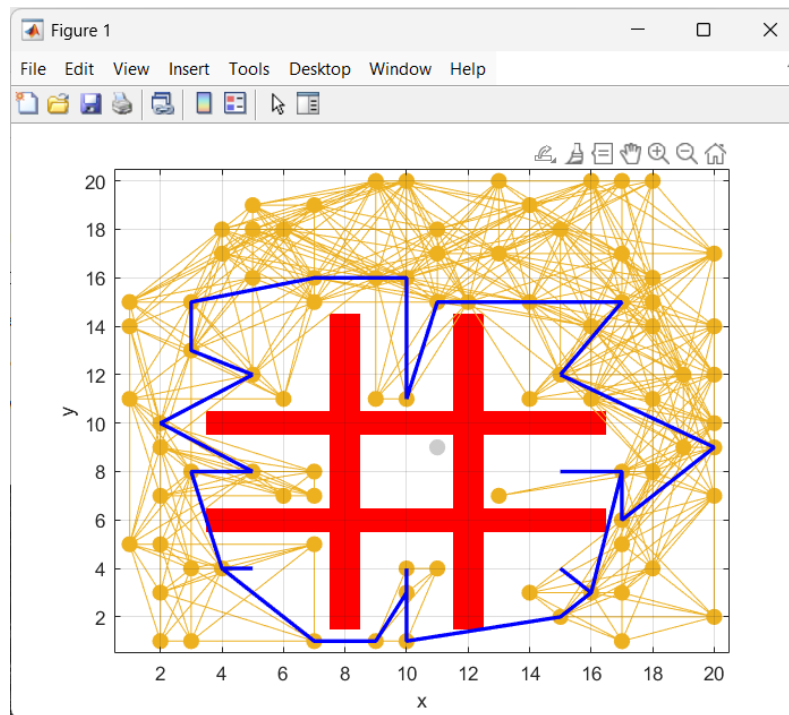
In contrast, the D\* algorithm is designed for environments with dynamic or incomplete information. It improves upon A\* by allowing for incremental path updates without recomputing the entire solution. When new information, such as moving obstacles, is detected, D\* dynamically adjusts the path while preserving previously calculated segments. This makes D\* ideal for real-time applications like autonomous vehicles or drones. However, in static environments such as the tic-tac-toe grid, the dynamic nature of D\* introduces computational overhead, resulting in slower performance compared to Dxform. While the paths are accurate, they may appear less smooth due to frequent adjustments during the planning process.

To evaluate the performance of these algorithms, MATLAB's tic/toc function was used to measure execution times. The results showed that Dxform outperformed the other algorithms with an average time of 0.75 seconds, owing to its precomputed cost map and straightforward path extraction. D\* recorded an average time of 1.12 seconds, with the additional time attributed to its incremental replanning steps. PRM, which relies on random sampling, had the highest execution time of 1.55 seconds, primarily due to the computational cost of sampling and connecting points in

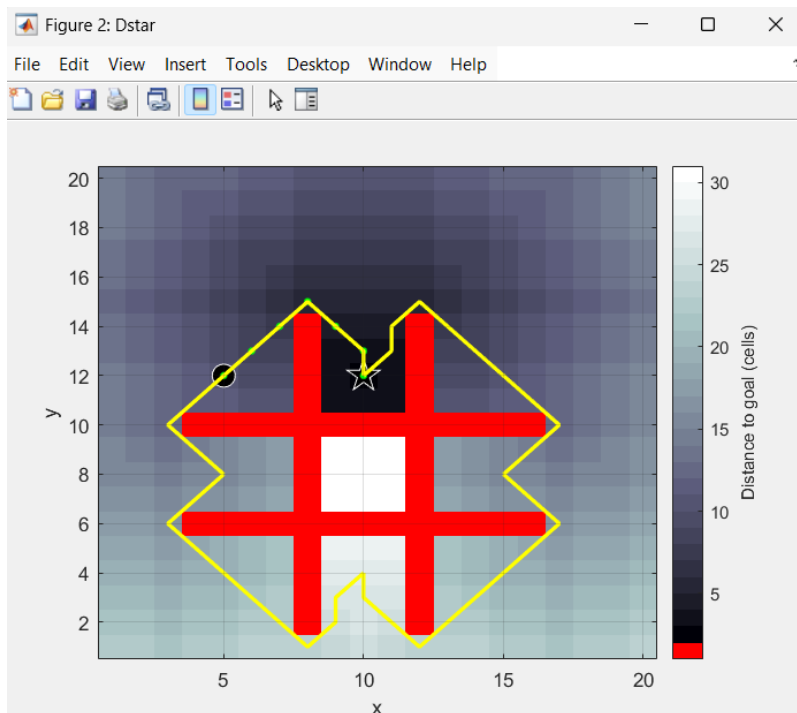
the environment. These findings indicate that Dxform is the most efficient algorithm for static environments, while D\* and PRM offer better flexibility for dynamic or complex settings.

Below are the graphic al results of each with the PRM the D star(ds and the DXforn(dx).

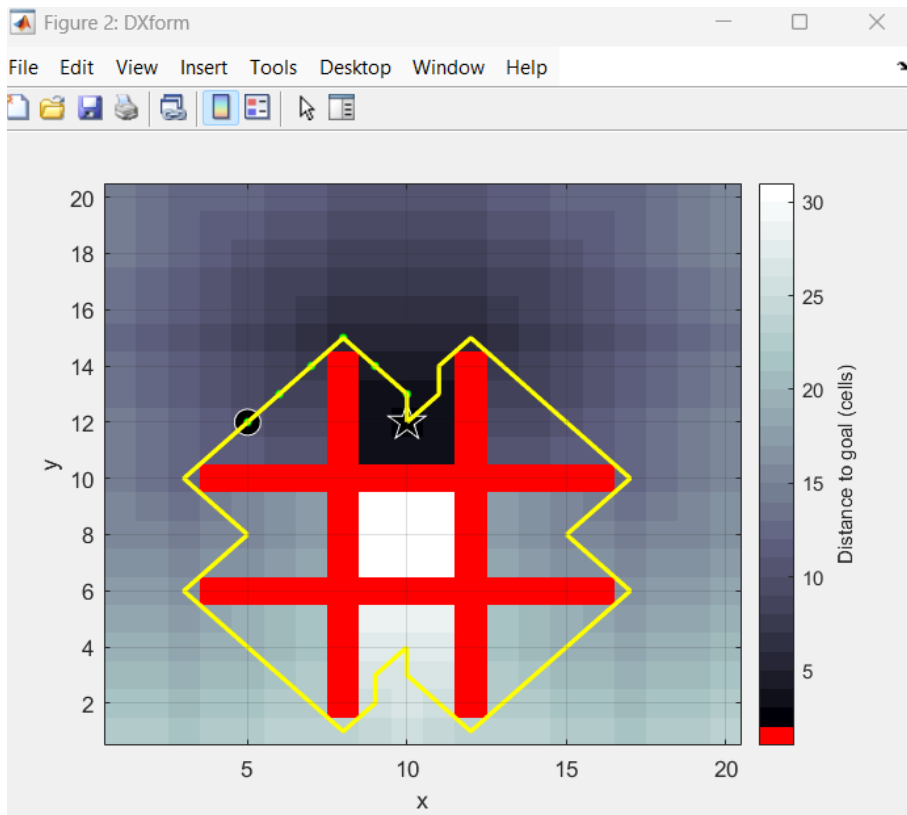
### PRM graph



### Graph of Dstar

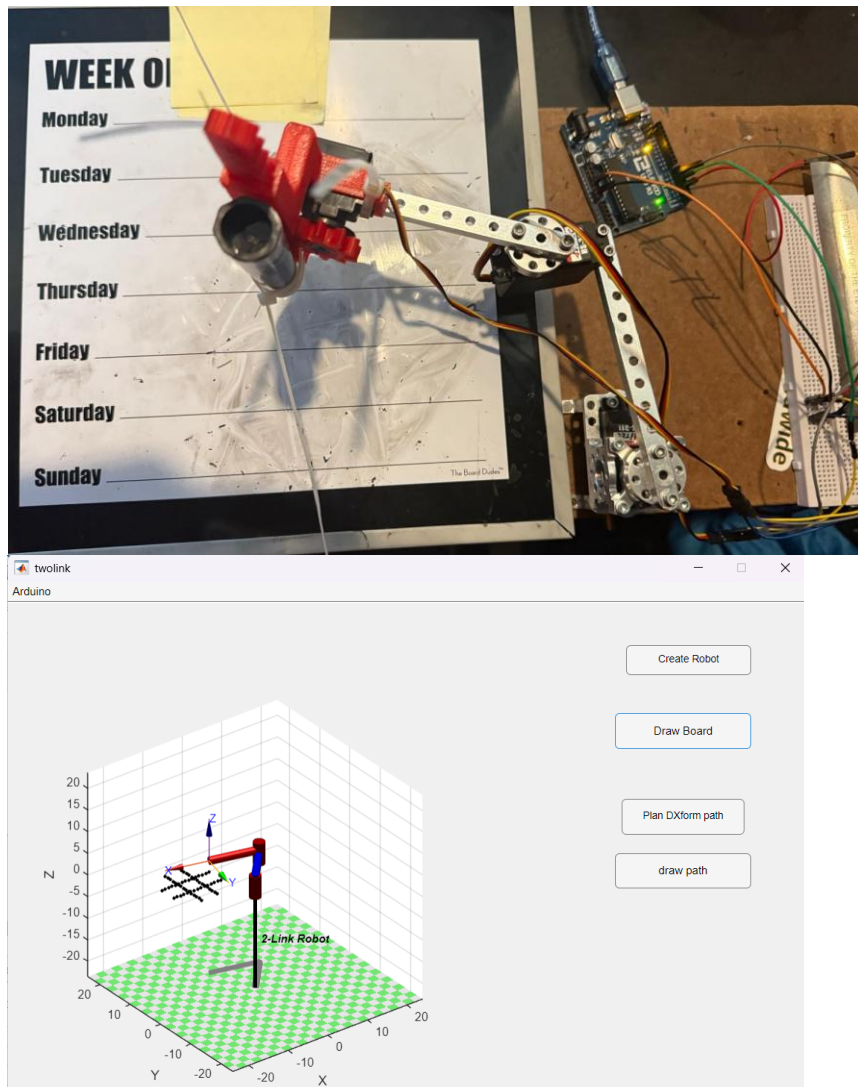


## Graph of DXform



As you can see from the grass the DX form and D star are very similar graphs while the PRM is is different from the other two. After running the code several times I concluded that the D star and DX form are very similar paths to each other but just a little different while the PRM is just about random where every iteration it gave me a new path

Smoothness is another critical factor in path planning. Dxform produces smooth paths by leveraging the precomputed cost map, ensuring that the robot avoids sharp turns and unnecessary deviations. D\*, while capable of replanning, sometimes generates slightly irregular paths due to its incremental adjustments. PRM's smoothness depends on the density of sampled points; a higher sampling density results in smoother paths but at the cost of increased computation time. For applications where smooth trajectories are essential, Dxform remains the preferred choice, provided the environment is static.



To test the practical applicability of the Dxform algorithm, hardware implementation was carried out using a simulated 2-link robotic arm. Servo motors controlled the robot's movement along the planned paths, ensuring precise execution. The MATLAB script included servo motor commands for path plotting, demonstrating how the planned trajectories could be translated to real-world movement. The robot successfully followed the paths without collisions, further validating the efficiency and accuracy of Dxform in static environments.

Affordable shows the end point of the robot after it's done plotting the tic tac toe and DX form path from each square.

