Navigation

KOM4520 Fundamentals of Robotic Vision

Robotic navigation problem

- Robot navigation is the problem of guiding a robot towards a goal.
- The human approach to navigation is to make maps and erect signposts, and at first glance it seems obvious that robots should operate the same way.
- However many robotic tasks can be achieved without any map at all, using an approach referred to as **reactive navigation**.
- e.g: navigating by heading towards a light, following a white line on the ground, moving through a maze by following a wall.
- In these applications the robot is reacting directly to its environment: the intensity of the light, the relative position of the white line or contact with a wall.

- Robotic vacuum cleaners are cleaning floors and most of them do so without using any map of the rooms in which they work.
- Instead they do the job by making random moves and sensing only that they have made contact with an obstacle.

Robotic navigation problem

- Human-style **map-based navigation** is used by more sophisticated robots and is also known as motion planning.
- This approach supports more complex tasks but is itself more complex. It imposes a number of requirements, not the least of which is having a map of the environment. It also requires that the robot's position is always known.

- The reactive and map-based approaches to robot navigation with a focus on wheeled robots operating in a planar environment will be discussed here.

By graduate students of MIT:
An autonomous robot with "socially aware navigation," that can keep pace with foot traffic while observing these general codes of pedestrian conduct.

Robotic navigation problem

dealdig-robvacuum-8-smart-robot-vacuum-cleaner

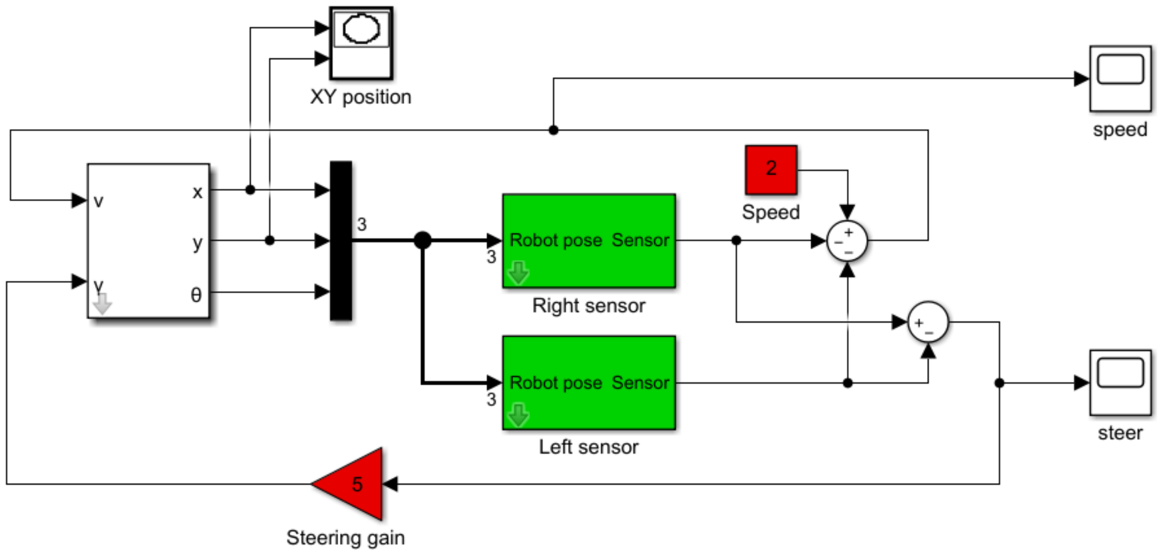
Reactive navigation

- Complex tasks can be performed by a robot even if it has no map and no real *idea* about where it is.
- Insects such as ants and bees gather food and return it to their nest based on input from their senses, they have far too few neurons to create any kind of mental map of the world and plan paths through it.
- Even single-celled organisms exhibit goal-seeking behaviors. In this case we need to temporarily modify our earlier definition of a robot to *a goal oriented machine that can sense, plan and act.*
- Grey Walter’s robotic tortoise demonstrated that it could moves toward a light source, a behavior known as phototaxis.
- This was an important result in the then emerging scientific field of cybernetics.

Reactive navigation

Braitenberg Vehicles:

- A very simple class of goal achieving robots are known as Braitenberg vehicles and are characterized by direct connection between sensors and motors.
- They have no explicit internal representation of the environment in which they operate and nor do they make explicit plans.



sl_braitenberg

Reactive navigation

- To ascend the gradient we need to estimate the gradient direction at the current location and this requires at least two measurements of the field.
- In this example two sensors are used, bilateral sensing, with one on each side of the robot's body.
- The sensors are parameterized by the position of the sensor with respect to the robot's body, and the sensing function which returns the sensor value $s(x, y) \in [0, 1]$ which is a function of the sensor's position in the plane. This particular function has a peak value at the point (60, 90).

The vehicle speed is

$$v = \frac{s_R + s_L}{2}$$

where s_R and s_L are the right and left sensor readings respectively. At the goal, where

$$s_R = s_L = 1$$

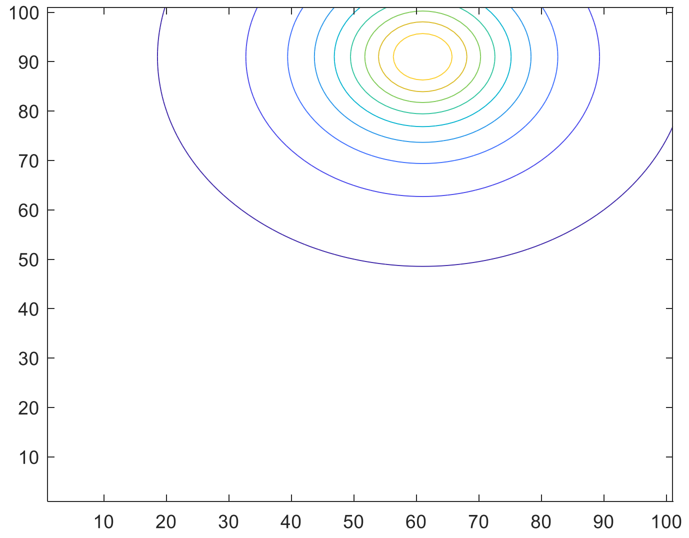
the velocity becomes zero.

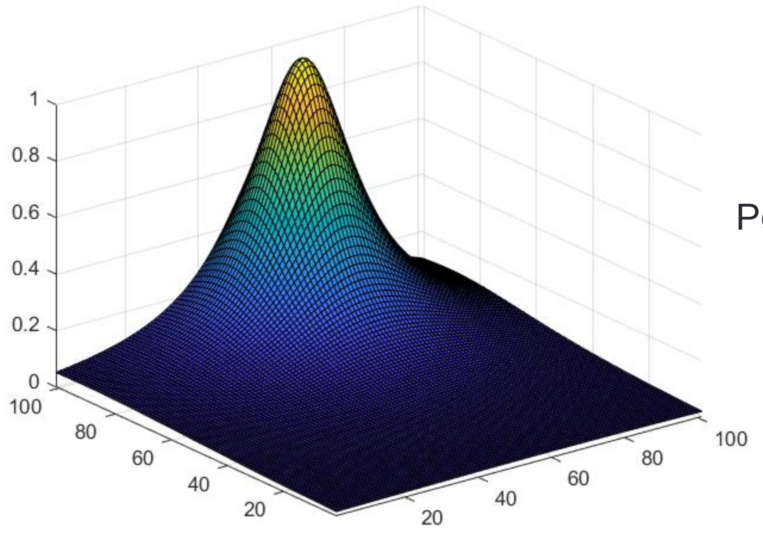
Steering angle is based on the difference between the sensor readings

$$\gamma = K(s_R - s_L)$$

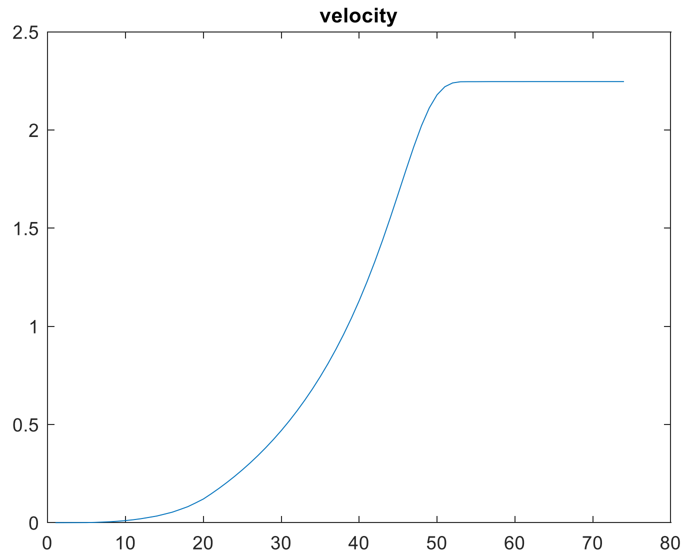
so when the field is equal in the left- and right-hand sensors the robot moves straight ahead.

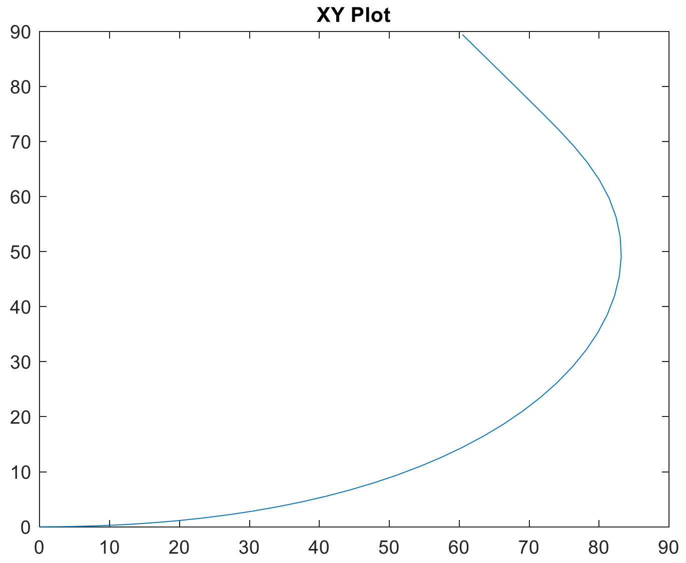
Reactive navigation





Potential Field





Map-Based Planning

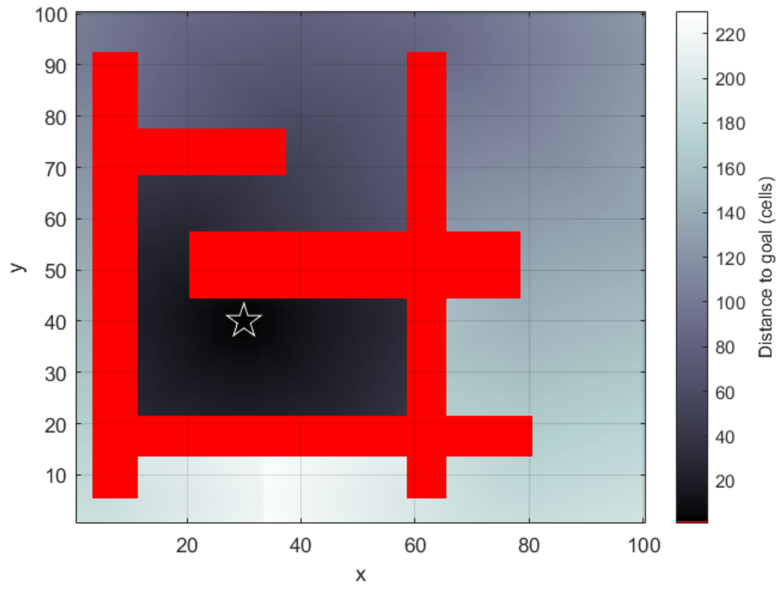
- The key to achieving the *best* path between points A and B, as we know from everyday life, is to use a map.
- Typically best means the shortest distance but it may also include some penalty term or cost related to traversability which is how easy the terrain is to drive over – it might be quicker to travel further but faster over better roads.
- A more sophisticated planner might also consider the size of the robot, the kinematics and dynamics of the vehicle and avoid paths that involve turns that are tighter than the vehicle can execute.

- Definition of a robot: “goal oriented machine that can sense, plan and act”
- Map-Based Planning focuses on planning

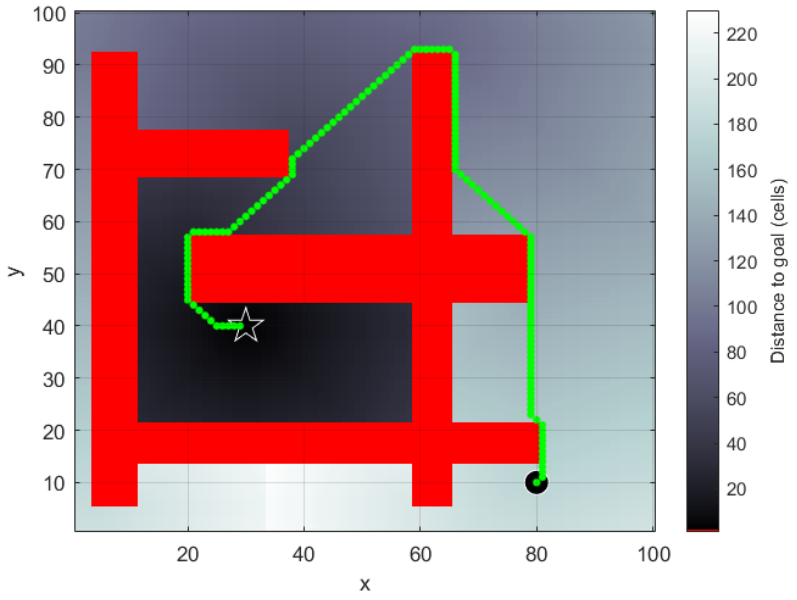
- There are many ways to represent a map and the position of the vehicle within the map. Graphs can be used to represent places and paths between them. Graphs can be efficiently searched to find a path that minimizes some measure or cost, most commonly the distance traveled.

Map-Based Planning

- A simpler and very computer friendly representation is the occupancy grid which is widely used in robotics.



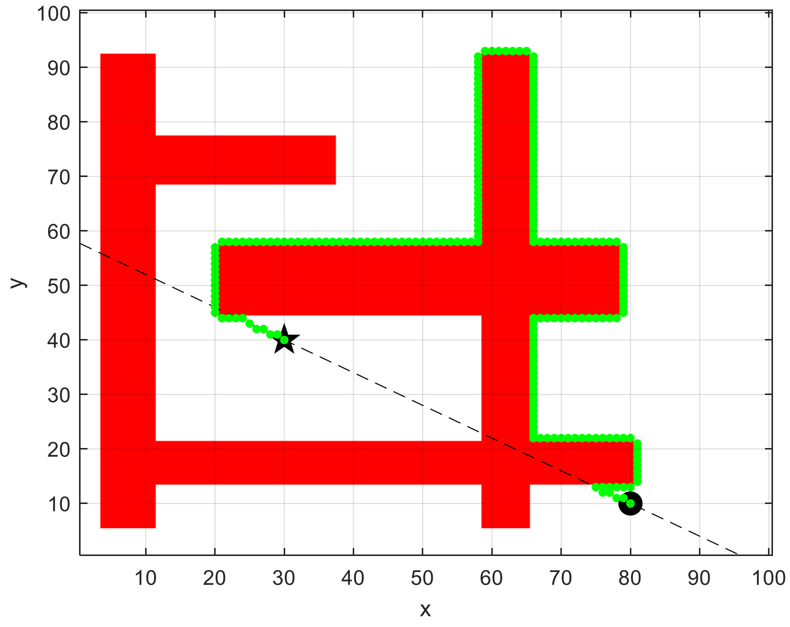
```
%map = makemap(100)
load map1
dx =  DXform(map);
dx.plan([30,40]);
dx.plot();
```



```
p1 = dx.query([80,10]);
dx.plot(p1);
```

Map-Based Planning

- We see the obstacle regions in red overlaid on the distance map whose grey level at any point indicates the distance from that point to the goal, in grid cells, taking into account travel around obstacles.
- The hard work has been done and to find the shortest path from any point to the goal we simply consult or query the plan.
- Although the plan is expensive to create, once it has been created it can be used to plan a path from *any* initial point to that goal

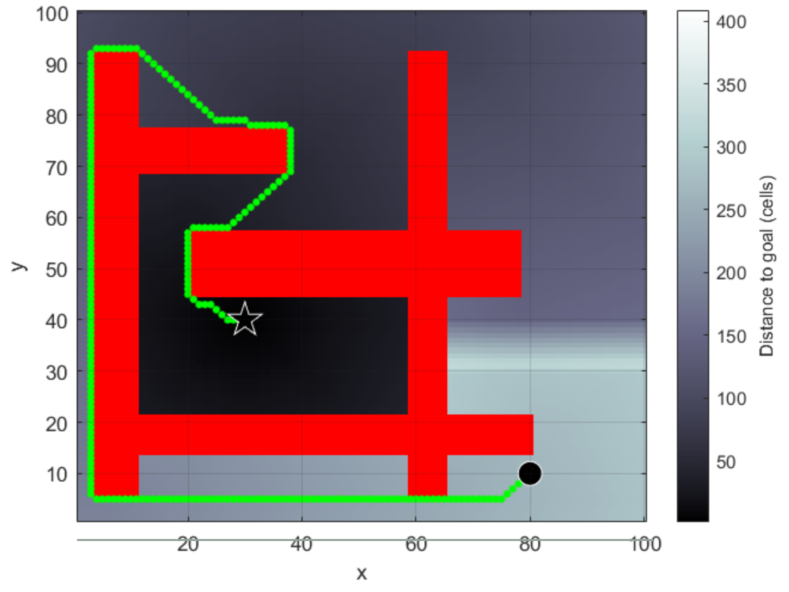


```
bug=Bug2(map);
p2=bug.query([80,10], [30,40]);
bug.plot(p2)
```

considerably longer than the path with map based planning.

Map-Based Planning

- A popular algorithm for robot path planning is D$*$ which finds the best path through a graph, which it first computes, that corresponds to the input occupancy grid. D* has a number of features that are useful for real-world applications. Firstly, it generalizes the occupancy grid to a cost map which represents the cost $c \in R, c > 0$ of traversing each cell in the horizontal or vertical direction



```
ds=Dstar(map);
c=ds.costmap( );
ds.plan([30,40]);
ds.modify_cost([60, 100; 30, 40], 20);
ds.plan();
p3=ds.query([80,10]);
ds.plot(p3);
```

Roadmap Methods

- In robotic path planning the analysis of the map is referred to as the *planning phase*.
- The *query phase* uses the result of the planning phase to find a path from A to B. The two previous planning algorithms, distance transform and D$*$, require a significant amount of computation for the planning phase, but the query phase is very cheap.

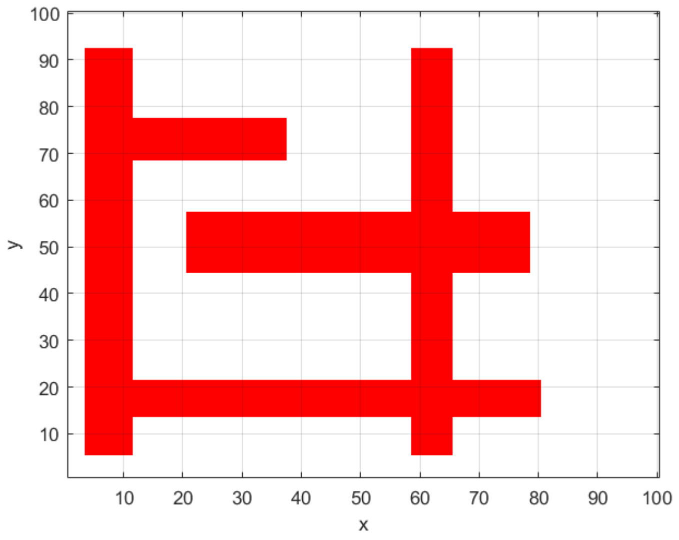
- If the goal changes the expensive planning phase must be re-executed.
- The disparity in planning and query costs has led to the development of roadmap methods where the query can include both the start and goal positions. The planning phase provides analysis that supports changing starting points and changing goals. A good analogy is making a journey by train.

- We first find a local path to the nearest train station, travel through the train network, get off at the station closest to our goal, and then take a local path to the goal. The train network is invariant and planning a path through the train network is straightforward.

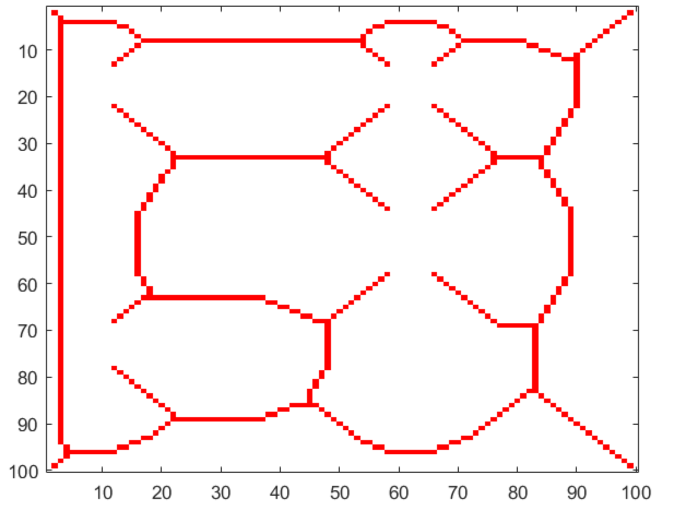
- Planning paths to and from the entry and exit stations respectively is also straightforward since they are, ideally, short paths. The robot navigation problem then becomes one of building a network of obstacle free paths through the environment which serve the function of the train network.

Roadmap Methods

- The topological skeleton of the free space is computed by a morphological image processing algorithm known as **thinning** applied to the free space/
- The obstacles have grown and the free space, the white cells, have become a thin network of connected white cells which are equidistant from the boundaries of the original obstacles.



```
free = 1 - map
free(1,:) = 0; free(end,:) = 0;
free(:,1) = 0; free(:,end) = 0;
skeleton = ithin(free);
```

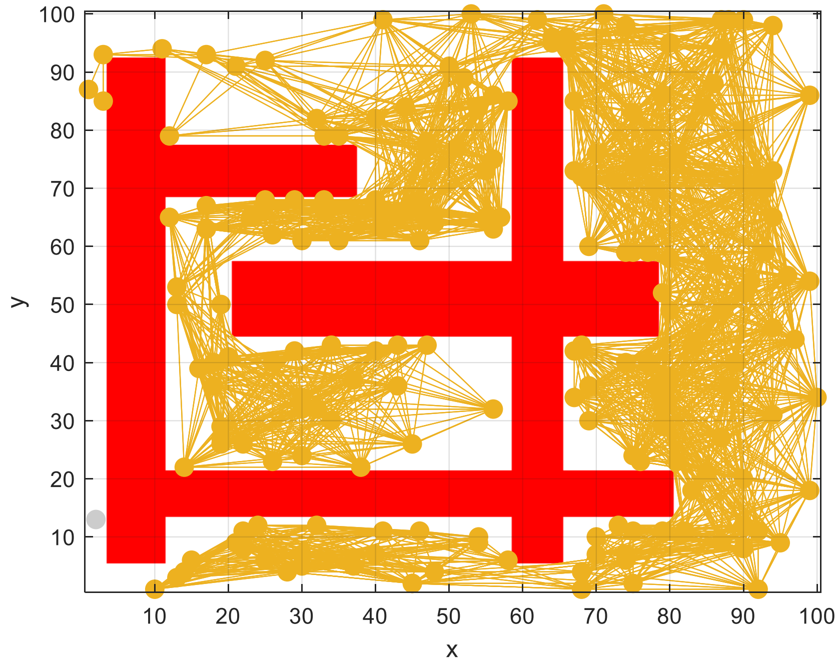


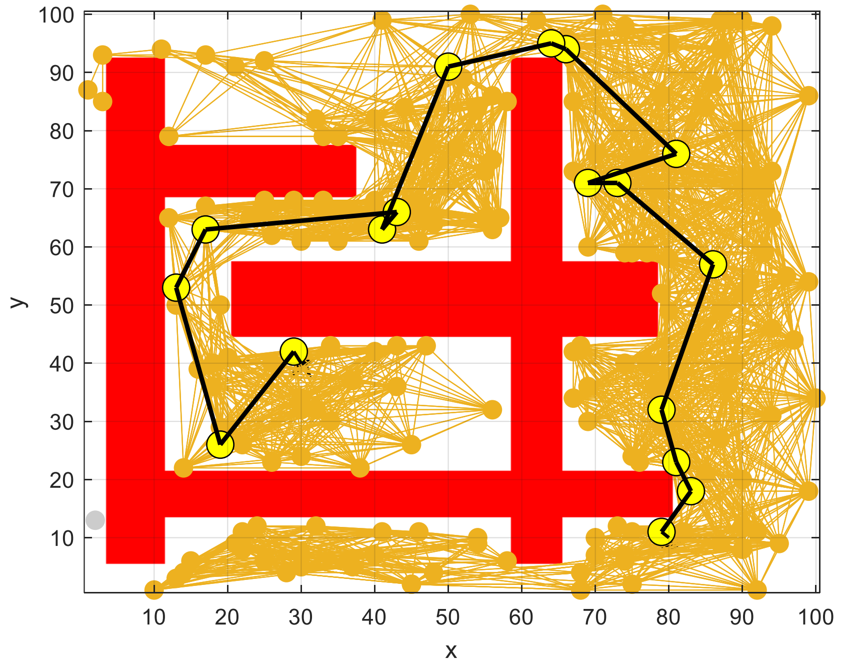
Probabilistic Roadmap Method (PRM)

- The high computational cost of the distance transform and skeletonization methods makes them infeasible for large maps and has led to the development of probabilistic methods.
- These methods sparsely sample the world map and the most well known of these methods is the probabilistic roadmap or PRM method.

```
prm=PRM(map)
prm.plan('npoints', 200)
prm.plot()
```

```
prm.query([80,10], [30,40])
prm.plot()
```



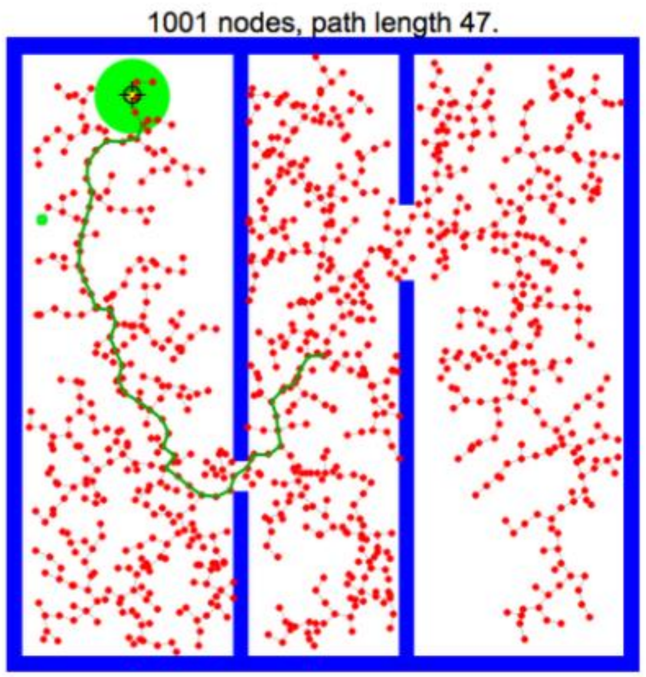


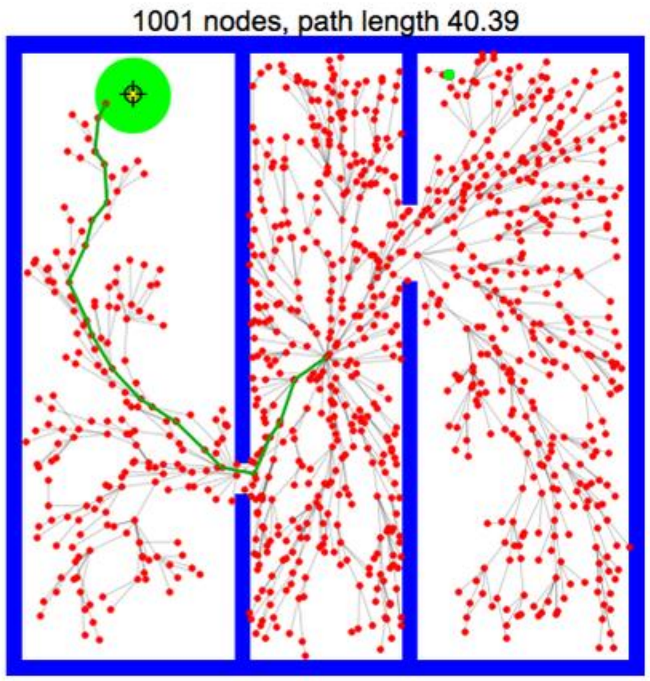
Probabilistic Roadmap Method (PRM)

- There are some important tradeoffs in achieving this computational efficiency.
- Firstly, the underlying random sampling of the free space means that a different roadmap is created every time the planner is run, resulting in different paths and path lengths.
- Secondly, the planner can fail by creating a network consisting of disjoint components.
- If the start and goal nodes are not connected by the roadmap, that is, they are close to different components the query method will report an error. The only solution is to rerun the planner and/or increase the number of nodes.
- Thirdly, long narrow gaps between obstacles such as corridors are unlikely to be exploited since the probability of randomly choosing points that lie along such spaces is very low.

Rapidly-Exploring Random Tree (RRT)

- A rapidly exploring random tree (RRT) grows a tree rooted at a start node. RRTs are designed to efficiently explore paths in a high-dimensional space.
- An RT selects a node at random from the tree and adds an edge in a random direction, but an RRT first selects a goal point, then tries to add an edge from the closest node in the tree toward the goal point. RRT* improves on this by rewiring the tree to form shortest paths.

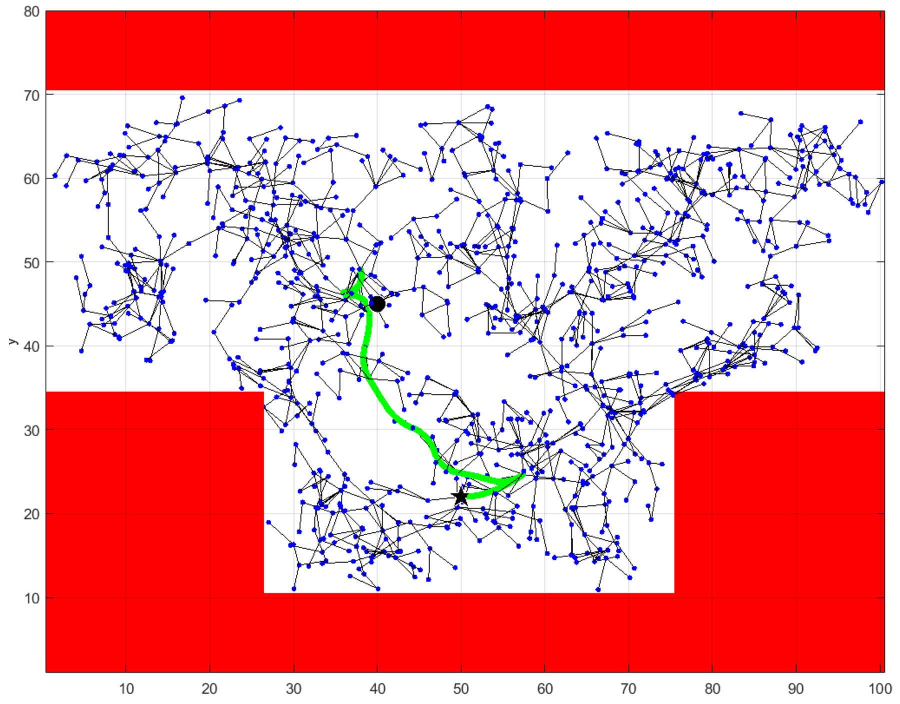




http://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT

Rapidly-Exploring Random Tree (RRT)

```
load road
car = Bicycle('steermax', 0.5);
rrt = RRT(car, road, 'npoints', 1000, 'root',
[50 22 0], 'simtime', 4)
rrt.plan();
p = rrt.query([40 45 0], [50 22 0]);
figure;
rrt.plot(p)
```



S. M. LaValle, *Rapidly-Exploring Random Trees: A New Tool for Path Planning*, Computer Science Department, Iowa State University (TR 98-11), 1998.