BRAC UNIVERSITY

Inspiring Excellence

Course : CSE461
Class Note : Robot Navigation

Navigation in robotics refers to the robot's ability to determine its own position in its frame of reference and then plan a path towards some goal location.

Navigation Problem:

Navigation problems can arise in different types of environments. Environments can be known or unknown, static or dynamic, and structured or unstructured (indoors or outdoors). These factors can affect the robot's ability to navigate effectively.

In known environments, the robot has prior knowledge of its surroundings, while in unknown environments, the robot must explore and gather information about its surroundings. In static environments, the surroundings do not change, while in dynamic environments, the surroundings can change over time. In structured environments, such as indoors, there are clear boundaries and obstacles, while in unstructured environments, such as outdoors, the surroundings can be more unpredictable.

Path Planning:

Path planning is an essential component of robot navigation. It refers to the process of determining a path for the robot to follow from its current position to a goal location, while avoiding obstacles and unsafe conditions. Path planning requires the determination of the robot's current position and a position of a goal location, both within the same frame of reference or coordinates.

One of the criteria for deciding on a path can be the speed at which the robot reaches its goal. In some situations, it may be important for the robot to reach its goal as quickly as possible, while in other situations, speed may not be as important. For example, in a search and rescue operation, it may be critical for the robot to reach its goal as quickly as possible, while in a manufacturing setting, speed may not be as important as precision.

The smoothness of the path can also be a factor in path planning. A smoother path may be easier for the robot to follow and may result in less wear and tear on the robot's components. For example, if the robot is carrying a delicate payload, it may be important for the path to be as smooth as possible to avoid damaging the payload.

The amount of compute power available can also affect path planning. Some path planning algorithms may require more computational resources than others, so the available compute power may determine which algorithms can be used. For instance, if the robot is operating in a complex environment with many obstacles, a more sophisticated path planning algorithm may be required, which may require more computational resources.

The precision of the robot's motion control can also affect path planning. If the robot has very precise motion control, it may be able to follow a more complex path than a robot with less precise motion control. Suppose, if the robot is operating in a tight space with many obstacles, precise motion control may be required to navigate successfully.

Path Planning mostly depends on the type of environment we have. There can be two types of environments:

1. Map is unknown : Visual Homing(Purely Reactive Navigation), Bug-based Path Planning
2. Map is known : Global Path Planning (Feature based map, Graph Based map).

Visual Homing(Purely Reactive Navigation): [no map]

Visual homing navigation relies on the robot's ability to recognize visual features in its environment and use them to determine its position relative to a goal location. The robot takes snapshots of its surroundings and compares them to a stored image of the goal location to determine how to move towards the goal.

Bug-based Path Planning: [no map]

Bug-based path planning is a type of path planning algorithm for mobile robots that is inspired by the way insects navigate. These algorithms are designed to work in environments where the robot has only local knowledge of its surroundings and a global goal.

Bug algorithms work by having the robot move towards its goal until it encounters an obstacle. When an obstacle is encountered, the robot follows the obstacle's boundary until it can continue towards its goal. This process is repeated until the robot reaches its goal.

Global Path Planning:

Metric or global path planning refers to path planning algorithms that use a global representation of the robot's environment to plan a path from the robot's current position to a goal location. This is in contrast to local or reactive path planning algorithms, which only use local sensor information to plan a path.

There are many different algorithms that can be used for global path planning, including Dijkstra's algorithm, A* , Probabilistic RoadMaps (PRMs), Visibility Graphs (VGs), and Rapidly-exploring Random Trees (RRTs). These algorithms differ in the way they search for a path and the type of paths they generate.

Localization:

Localization is the process of determining the position and orientation of a robot within its environment. It is one of the most fundamental competencies required by an autonomous robot, as the knowledge of the robot's own location is essential for making decisions about future actions.

There are many types of robot localization. We will study only three of them.

1. **Dead-Reckoning(Motion):** Dead-reckoning is a method of estimating an object's current position based on its previous position, speed, heading (or direction or course), and elapsed time. In robotics, dead-reckoning can be used to estimate a robot's position based on its motion. For example, if a robot knows its starting position and can measure its speed and heading, it can use dead-reckoning to estimate its current position by calculating how far it has traveled in a given direction over a given period of time. Dead-reckoning can be useful for estimating a robot's position when other localization methods are not available or not reliable. However, it is subject to cumulative errors due to uncertainties in the robot's motion and sensor measurements. As a result, dead-reckoning is typically used in combination with other localization methods to improve the accuracy of the robot's position estimate.
2. **LandMark(Sensing) Based Localization:** Landmark-based localization is a method of determining a robot's position within its environment by using known landmarks as reference points. The robot uses its sensors to detect and recognize landmarks in its environment and then uses their known positions to estimate its own position. Landmark-based localization can be very effective when the robot is operating in an environment with distinctive and easily recognizable landmarks. However, it can be challenging in environments where landmarks are not easily distinguishable or where the robot's sensors have difficulty detecting them.
3. **State Estimation (uncertainty in motion & sensing):** State estimation is the process of determining the state of a system based on incomplete or uncertain information. In robotics, state estimation is used to estimate the position, orientation, and other relevant quantities of a robot based on sensor measurements and knowledge of the robot's motion. Uncertainty in motion and sensing can affect the accuracy of state estimation for localization. For example, if a robot's sensors are noisy or if its motion is uncertain, it can be difficult to accurately estimate its position and orientation.

   To deal with uncertainty in motion and sensing, state estimation algorithms for localization typically use probabilistic methods to represent and reason about uncertainty. For example, a Kalman filter can be used to estimate the position and orientation of a robot based on noisy sensor measurements and uncertain motion models.

Mapping:

Mapping is an important aspect of robot navigation, which involves creating a representation of the environment that the robot can use to plan its movements. Mapping allows robots to navigate autonomously by providing them with information about the location of obstacles, landmarks, and other features in their environment.

Occupancy Grid :

An occupancy grid is a popular mapping technique used in robotics and computer vision to represent an environment as a two-dimensional grid of cells, where each cell represents a small area of the environment. Each cell in the grid is assigned a probability value indicating the likelihood of that cell being occupied by an obstacle or object.

The occupancy grid is typically generated by a robot's sensors, such as lidar or sonar, which scan the environment and measure the distance to nearby objects. The data from the sensors is then used to estimate the likelihood of each cell being occupied by an obstacle.

The occupancy grid can be represented as a matrix, where each element of the matrix represents the probability that the corresponding cell is occupied. A value of 0 indicates that the cell is definitely free, while a value of 1 indicates that the cell is definitely occupied. Values between 0 and 1 represent uncertain or unknown states.

Sensor Model:
A sensor model takes into account the characteristics and limitations of the sensor, as well as the uncertainties and noise in the sensor measurements. For example, a range sensor, such as a laser rangefinder or sonar, might have a sensor model that describes the probability of obtaining a particular range measurement given the distance to an object and the sensor's noise characteristics.

Sensor models are often used in conjunction with mapping techniques, such as occupancy grids, to estimate the probability of an environment being occupied by an obstacle based on sensor measurements. By combining sensor measurements with the sensor model, the robot can generate a more accurate and reliable estimate of the environment's state.

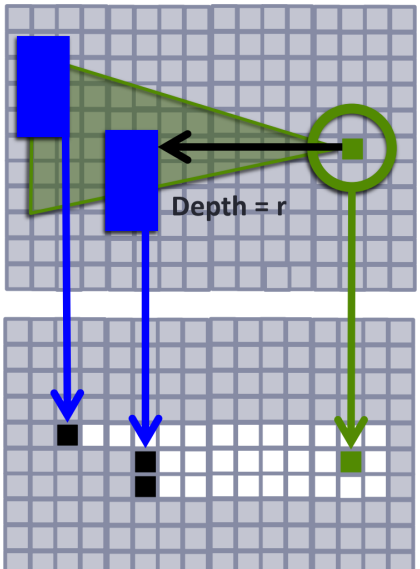
Depth Sensor Model :

Assuming that the depth sensor is mounted at a point p and has a maximum range R and maximum angle B, the sensor model can be defined as follows:

Region 1: If the measured distance is less than r, the cell is likely to be empty.
where dist is the distance from the cell to the sensor p.

Region 2: If the measured distance is equal to r, the cell is likely to be occupied by an obstacle. The probability of the cell being occupied by an obstacle is given by the function:
P(obstacle) = 1

Region 3: If the measured distance is greater than r, the cell is considered unknown or obscured.
where R is the maximum range of the sensor.



A Simple OG Mapping Algorithm:

Initialize a Grid: Create an occupancy grid by dividing the environment into a regular grid of cells. Each cell is initially set to an unknown state.

Update the Grid: As the robot moves through the environment, update the occupancy grid by using sensor measurements to determine whether each cell is occupied or free. For example, if a range sensor detects an obstacle within a cell, that cell is marked as occupied.

Pick a Next Move: Based on the current occupancy grid, select the next move for the robot. This can be done by using a path planning algorithm, such as A* or Dijkstra's algorithm, to find a path to the robot's goal while avoiding obstacles.

Loop forever: Repeat steps 2-3 indefinitely, continuously updating the occupancy grid and selecting the next move for the robot.

Exploration :
The goal of exploration is to visit as many unknown areas of the environment as possible while minimizing the time and energy required to do so. To achieve this goal, robots use a variety of techniques, such as frontier-based exploration, information gain-based exploration, and uncertainty-based exploration.

Frontier Based Exploration :

A frontier cell is a cell adjacent to an unknown or occupied cell. The idea behind frontier-based exploration is that by exploring frontier cells, the robot can quickly cover large portions of the environment while minimizing the amount of backtracking required. This is because frontier cells represent the boundary between known and unknown areas of the environment, and exploring them can reveal new information about the environment's layout and features.

To implement frontier-based exploration, the robot first creates a map of the environment and divides it into a grid of cells. Each cell is initially marked as unknown, free, or occupied based on sensor measurements. The robot then identifies the frontier cells by searching for cells adjacent to unknown or occupied cells.