ROBOTICS AND AUTOMATION

Unit IV — Autonomous Navigation & Path Planning

Topics Covered (03 Hours)

Environment Representation: Occupancy Grids & Maps
Configuration Space (C-Space) Dimensions
Path Planning: Dijkstra's Shortest Path Search
Path Planning: A* Heuristic-Based Pathfinding
Path Planning: Rapidly-exploring Random Trees (RRT)
Localization: Odometry Mechanics & Drift Errors
Sensor Fusion & Kalman Filters (Linear & Extended)
Particle Filters (Monte Carlo Localization Rules)
Obstacle Avoidance: Artificial Potential Field Methods
Obstacle Avoidance: Reactive Control Loops
Case Study: Autonomous Mobile Robots in Hospitals

Compiled in a simple, easy-to-understand, book style format
for students of Computer Science and Engineering

TABLE OF CONTENTS

4.1 Environment Representation & C-Space 1
4.2 Path Planning Algorithms 1
Dijkstra's & A* Algorithms 1
Rapidly-exploring Random Trees (RRT) 1
4.3 Localization & Obstacle Avoidance 1
Localization Techniques 1
Obstacle Avoidance (Potential Fields) 1
4.4 Case Study: AMR Navigation in Hospitals 1
Unit Summary 1

4.1 Environment Representation & C-Space

To navigate autonomously, a mobile robot must have a digital model of its environment. The robot maps raw sensor data into three primary abstract representations:

- **1. Occupancy Grids:** Divides the environment into a grid of binary or probabilistic cells. Each cell contains a value representing the probability that the cell is occupied by an obstacle (ranging from 0 for completely free to 1 for completely occupied). These maps are built using range sensors (LiDAR).
- **2. Topological Maps:** Represent the environment as a mathematical graph. Nodes represent key landmarks or locations (e.g., rooms, hallway intersections), and edges represent paths connecting them. They discard fine geometric details, saving memory.
- **3. Configuration Space (C-Space):** The set of all possible positions and orientations the robot can assume. Because the robot has a physical size and shape, we simplify path planning by shrinking the robot to a single mathematical point, while simultaneously expanding all obstacles by the radius of the robot's footprint. The resulting space is the C-Space, divided into C-free (safe areas) and C-obstacle (collision areas).

4.2 Path Planning Algorithms

Once the environment is mapped, the robot's controller runs path planning algorithms to compute a collision-free path from a start position to a goal position.

Dijkstra's & A* Algorithms

- **Dijkstra's Algorithm:** A graph search algorithm that explores nodes incrementally from the start location, finding the absolute shortest path. However, it explores nodes in all directions blindly, making it slow.
- **A* (A-Star) Algorithm:** An optimized version of Dijkstra's algorithm. It uses a Heuristic Function h(n) to guide the search towards the goal. For each candidate node n, it computes:

f(n) = g(n) + h(n)

Where g(n) is the actual cost to reach node n from the start, and h(n) is the estimated heuristic cost (typically straight-line Euclidean distance) from n to the goal. A* is faster because it focuses its node search directionally towards the target.

Rapidly-exploring Random Trees (RRT)

A* works well on 2D grids, but for complex, high-dimensional spaces (like a 6-DoF robotic arm avoiding self-collision), grid-based search is too slow. RRT is a sample-based path planning algorithm:

- 1. Start with a tree containing only the start node.
- 2. Sample a random point in C-space.
- 3. Find the nearest node in the existing tree to this random point.
- 4. Extend the tree from the nearest node a small step toward the random point.
- 5. Check for collisions. If safe, add the new point to the tree. Repeat until the tree reaches the goal.

4.3 Localization & Obstacle Avoidance

A robot must know its exact position (Localization) and avoid dynamic obstacles during navigation.

Localization Techniques

- **Odometry:** Calculates position by integrating wheel rotation data from motor encoders over time. Highly accurate over short periods, but suffers from cumulative drift errors due to wheel slip and uneven surfaces.
- **Sensor Fusion (Kalman Filters):** Combines noisy measurements from multiple sensors (e.g., merging wheel encoders with an Inertial Measurement Unit (IMU) and LiDAR landmarks). The Kalman Filter (and Extended Kalman Filter for non-linear systems) uses a mathematical predict-correct cycle to estimate the robot's state with high precision.
- **Particle Filters (Monte Carlo Localization):** Represents the robot's position as a cloud of particles (hypotheses). As the robot moves, particles are updated based on odometry. Particles that match range sensor observations are given high weight, and the cloud is resampled. Eventually, the particles converge on the robot's true location.

Obstacle Avoidance (Potential Fields)

Autonomous robots use reactive control loops to adjust their paths in real-time when dynamic obstacles appear. A common method is the Artificial Potential Field:

The goal is modeled as an attractive pole, pulling the robot towards it. Obstacles are modeled as repulsive poles, pushing the robot away. The robot calculates the vector sum of these forces to determine its immediate steering direction. While simple and fast, it can get stuck in Local Minima traps (e.g., a U-shaped obstacle where the forces balance out to zero before the goal is reached).

4.4 Case Study: AMR Navigation in Hospitals

Modern hospitals deploy Autonomous Mobile Robots (AMRs) to deliver medicine, linens, and meals between wards. The hospital environment is highly dynamic, containing patients, staff, stretchers, and moving equipment.

The AMR uses LiDAR sensors to match active laser scans against a pre-loaded topological floor plan. It localizes using Particle Filters, combining wheel encoders and IMU data to track movement between wards. To plan a route, it runs the A* algorithm on the graph map.

When a dynamic obstacle (like a person walking) blocks the path, the AMR's laser scanner detects the obstacle, updates its local occupancy grid, and calculates a detour path in real-time. If the path remains blocked, the robot stops, plays a voice warning, and recalculates a new global path using alternative corridors.

Unit Summary

This unit explored robot localization, path planning, and navigation.

- Environment maps include occupancy grids (probabilities), topological graphs, and C-space geometry.
- Path planning algorithms include Dijkstra's (shortest path), A* (heuristics), and RRT (sample trees).
- Localization combines wheel odometry, sensor fusion (Kalman filters), and particle filters.
- Obstacle avoidance uses potential field forces or reactive control loops to navigate around dynamic blocks.
- Real-world case studies demonstrate AMRs navigating dynamic hospital environments safely.