Navigation and Planning for Mobile Robots

Ilia Nechaev

06.12.2024

Introduction

What do we have:

- Map of the environment
- Start position
- Goal position

What do we want:

- Find a path from start to goal
- Avoid known and unknown obstacles

What is environment?

- Map of the world with known obstacles
- Set of robot's parameters that can be reached without collision with obstacles

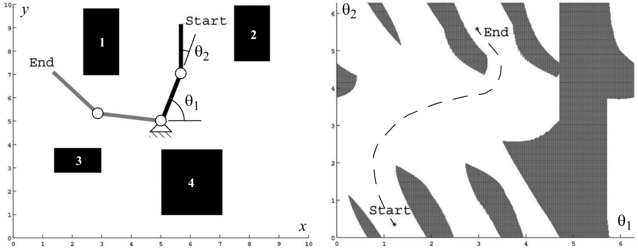


Figure: World map and configuration space

How can we represent the environment?

- Graph-based representation (discrete)
- Potential field representation (continuous)

Visibility Graph

- Nodes at obstacle vertices + start/goal
- Edges represent line-of-sight connections
- Useful for shortest path planning

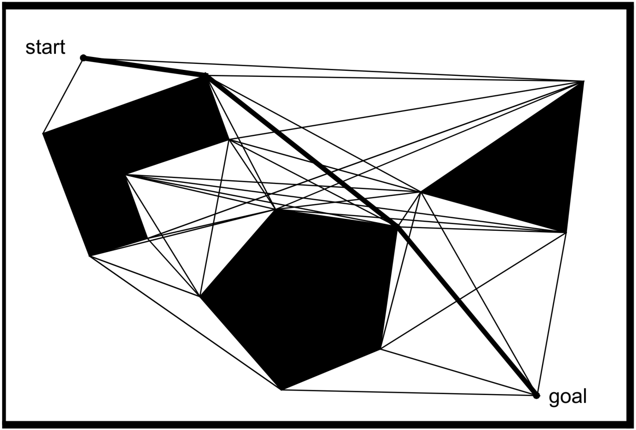


Figure: Example of a Visibility Graph

Voronoi Diagram

- Decomposes space based on proximity to obstacles
- Path maximizes clearance from obstacles
- Suitable for safe navigation

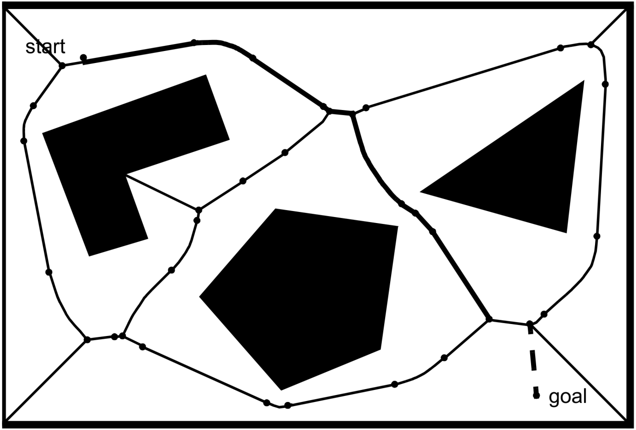
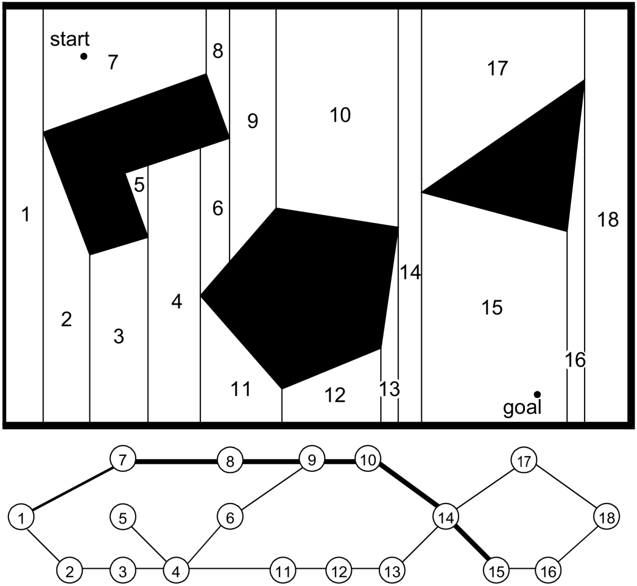


Figure: Example of a Voronoi Diagram

Exact Cell Decomposition

- Divides free space into non-overlapping cells
- Cells exactly cover the free space
- Graph nodes represent cells; edges represent adjacency

Exact cell decomposition



Approximate Cell Decomposition

- Uses regular grids (e.g., quadtrees)
- Approximate representation of free space
- Balances computational efficiency and accuracy

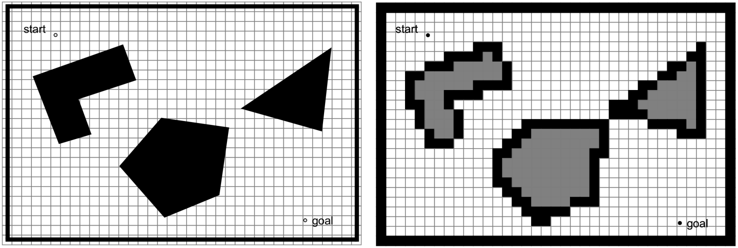


Figure: Approximate cell decomposition

Breadth-First Search (BFS)

- Explores neighbor nodes level by level
- Guarantees the shortest path in unweighted graphs
- Suitable for simple pathfinding tasks

Depth-First Search (DFS)

- Explores as far as possible along each branch
- Does not guarantee the shortest path
- Useful for exploring all possible paths

Dijkstra's Algorithm

- Finds the shortest path in weighted graphs
- Uses a priority queue to select the next node
- Guarantees the optimal path

A* Algorithm

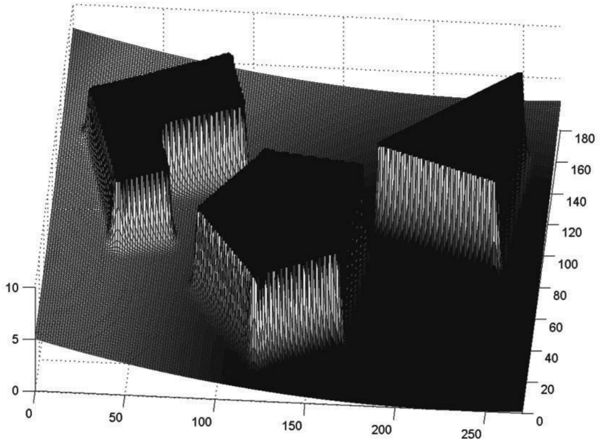
- Enhances Dijkstra's algorithm with heuristics
- Estimates cost to goal using a heuristic function
- More efficient pathfinding in large graphs

D* Algorithm

- Dynamic version of A* for changing environments
- Replans paths when obstacles are detected
- Suitable for real-time navigation

Potential Field Path Planning

- Treats robot as a particle in a field
- Goal exerts attractive force; obstacles exert repulsive forces
- Results in smooth and reactive paths



How to use?

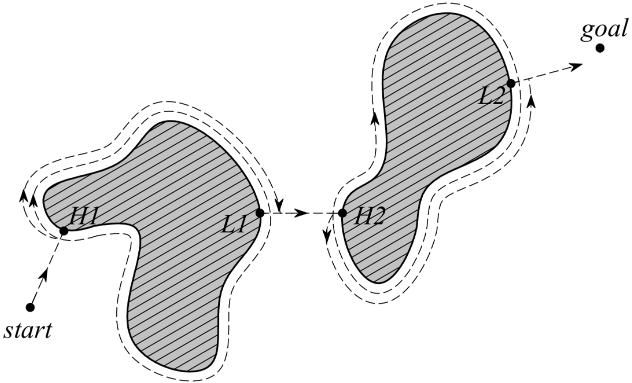
- **Potential field**
  - Potential field can be described as function $U(x, y) = U_{\text{attr}}(x, y) + U_{\text{rep}}(x, y)$
  - $U_{\text{attr}}(q) = \frac{1}{2} k_{\text{att}} \cdot \|q - q_{\text{attr}}\|$
  - $U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \cdot \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$
- **Potential force**
  - $F(q) = -\nabla U(q) = -\begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix} = -\nabla U_{\text{attr}}(q) - \nabla U_{\text{rep}}(q) = F_{\text{attr}}(q) + F_{\text{rep}}(q)$
  - $F_{\text{attr}}(q) = -k_{\text{attr}} \cdot (q - q_{\text{goal}})$
  - $F_{\text{rep}}(q) = \begin{cases} k_{\text{rep}} \cdot \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \cdot \frac{1}{\rho^2(q)} \cdot \nabla \rho(q) & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases}$
- Potential force can be used to control the robot's direction

Bug Algorithms Overview

- Simple and efficient obstacle avoidance methods
- Based on local sensing and minimal global information
- Types include Bug1, Bug2, and Tangent Bug

Bug1 Algorithm

- Move toward the goal until an obstacle is encountered
- Circumnavigate the obstacle while recording the closest point to the goal
- Resume moving toward the goal from the closest point
- Ensures reaching the goal if the environment is simply connected



Bug2 Algorithm

- Define a straight line (M-line) from start to goal
- Follow the M-line until an obstacle is hit
- Traverse the obstacle boundary until the M-line is re-encountered closer to the goal
- More efficient than Bug1 in many scenarios

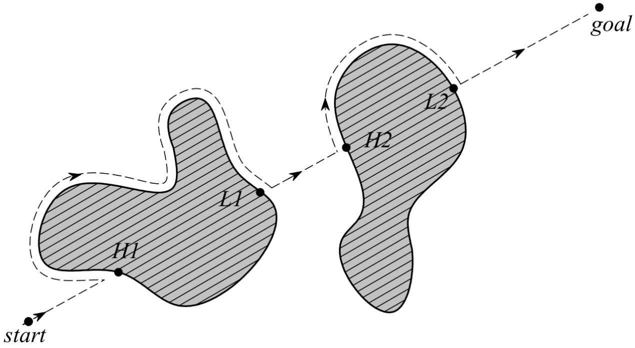


Figure: Bug2 Algorithm Path

Conclusion

- Summarized graph construction techniques
- Reviewed key graph search algorithms
- Discussed potential field path planning
- Discussed bug algorithms for obstacle avoidance

References

Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots. Second edition*. MIT Press, 2011.