Planning and Navigation
Where am I going? How do I get there?

Competencies for Navigation I

• Cognition / Reasoning:
  ➢ is the ability to decide what actions are required to achieve a certain goal in a given situation (belief state).
  ➢ decisions ranging from what path to take to what information on the environment to use.
• Today’s industrial robots can operate without any cognition (reasoning) because their environment is static and very structured.
• In mobile robotics, cognition and reasoning is primarily of geometric nature, such as picking safe path or determining where to go next.
  ➢ already been largely explored in literature for cases in which complete information about the current situation and the environment exists (e.g. traveling salesman problem).

Competencies for Navigation II

• However, in mobile robotics the knowledge of about the environment and situation is usually only partially known and is uncertain.
  ➢ makes the task much more difficult
  ➢ requires multiple tasks running in parallel, some for planning (global), some to guarantee “survival of the robot”.
• Robot control can usually be decomposed into various behaviors or functions
  ➢ e.g. wall following, localization, path generation or obstacle avoidance.
• In this chapter we are concerned with path planning and navigation, (and assume low level motion control and localization).
• We can generally distinguish between (global) path planning and (local) obstacle avoidance.

Global Path Planning

• Assumption: there exists a good enough map of the environment for navigation.
  ➢ Topological or metric or a mixture between both.
• First step:
  ➢ Representation of the environment by a road-map (graph), cells or a potential field. The resulting discrete locations or cells allow then to use standard planning algorithms.
• Examples:
  ➢ Visibility Graph
  ➢ Voronoi Diagram
  ➢ Cell Decomposition -> Connectivity Graph
  ➢ Potential Field
Path Planning: Configuration Space

- State or configuration $q$ can be described with $k$ values $q_i$.

$\text{what is the configuration space of a mobile rotor?}$

Path Planning Overview

1. Road Map, Graph construction
   - Identify a set of routes within the free space
   - Where to put the nodes?
     - Topology-based: at distinctive locations
     - Metric-based: where features disappear or get visible

2. Cell decomposition
   - Discriminate between free and occupied cells
   - Where to put the cell boundaries?
     - Topology- and metric-based: where features disappear or get visible

3. Potential Field
   - Impose a mathematical function over the space

Road-Map Path Planning: Visibility Graph

- Shortest path length
- Grow obstacles to avoid collisions

Road-Map Path Planning: Voronoi Diagram

- Easy executable: Maximize the sensor readings
- Works also for map-building: Move on the Voronoi edges
Road-Map Path Planning: Cell Decomposition

- Divide space into simple, connected regions called cells
- Determine which open cells are adjacent and construct a connectivity graph
- Find cells in which the initial and goal configuration (state) lie and search for a path in the connectivity graph to join them.
- From the sequence of cells found with an appropriate search algorithm, compute a path within each cell.
  - e.g. passing through the midpoints of cell boundaries or by sequence of wall following movements.

Road-Map Path Planning: Exact Cell Decomposition

Road-Map Path Planning: Approximate Cell Decomposition

Road-Map Path Planning: Adaptive Cell Decomposition
Road-Map Path Planning: Path / Graph Search Strategies

- Wavefront Expansion NF1 (see also later)
- Breadth-First Search
- Depth-First Search
- Greedy search and A*

Potential Field Path Planning:

- Potential Field Generation
  - Generation of potential field function $U(q)$
    - attracting (goal) and repulsing (obstacle) fields
    - summing up the fields
    - functions must be differentiable
  - Generate artificial force field $F(q)$
    $$ F(q) = -\nabla U(q) = -\nabla U_{at}(q) - \nabla U_{rep}(q) = \nabla [\frac{\partial U}{\partial q_x} \frac{\partial U}{\partial q_y}] $$
  - Set robot speed $(v_x, v_y)$ proportional to the force $F(q)$ generated by the field
    - the force field drives the robot to the goal
    - if robot is assumed to be a point mass

Potential Field Path Planning: Attractive Potential Field

- Parabolic function representing the Euclidean distance $|q - q_{goal}|$ to the goal
  $$ U_{at}(q) = \frac{1}{2} k_{at} \cdot \|q - q_{goal}\|^2 $$
- Attracting force converges linearly towards 0 (goal)
  $$ F_{at}(q) = -\nabla U_{at}(q) = -k_{at} \cdot \nabla q_{goal} \cdot (q - q_{goal}) $$
Potential Field Path Planning: **Repulsing Potential Field**

- Should generate a barrier around all the obstacle
  - strong if close to the obstacle
  - no influence if far from the obstacle

\[
U_{rep}(q) = \begin{cases} 
\frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \leq \rho_0 \\
0 & \text{if } \rho(q) \geq \rho_0
\end{cases}
\]

- \(\rho(q)\): minimum distance to the object
- Field is positive or zero and tends to infinity as \(q\) gets closer to the object

\[
F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} 
\frac{k_{rep}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\
0 & \text{if } \rho(q) \geq \rho_0
\end{cases}
\]

Potential Field Path Planning: **Extended Potential Field Method**

- Additionally a rotation potential field and a task potential field in introduced

- Rotation potential field
  - force is also a function of robot’s orientation to the obstacle

- Task potential field
  - Filters out the obstacles that should not influence the robot’s movements, i.e., only the obstacles in the sector Z in front of the robot are considered

Obstacle Avoidance (Local Path Planning)

- The goal of the obstacle avoidance algorithms is to avoid collisions with obstacles
- It is usually based on local map
- Often implemented as a more or less independent task
- However, efficient obstacle avoidance should be optimal with respect to
  - the overall goal
  - the actual speed and kinematics of the robot
  - the on board sensors
  - the actual and future risk of collision

- Example: Alice

Obstacle Avoidance: **Bug1**

- Following along the obstacle to avoid it
- Each encountered obstacle is once fully circled before it is left at the point closest to the goal
Obstacle Avoidance: Bug2

- Following the obstacle always on the left or right side
- Leaving the obstacle if the direct connection between start and goal is crossed

Obstacle Avoidance: Vector Field Histogram (VFH)

- Environment represented in a grid (2 DOF)
- Cell values equivalent to the probability that there is an obstacle
- Reduction in different steps to a 1 DOF histogram
- Calculation of steering direction
- All openings for the robot to pass are found
- The one with the lowest cost function $G$ is selected

Obstacle Avoidance: Video VFH

- Notes:
  - Limitation if narrow areas (e.g., doors) have to be passed
  - Local minimum might not be avoided
  - Reaching of the goal can not be guaranteed
  - Dynamics of the robot not really considered
Obstacle Avoidance: The Bubble Band Concept

- Bubble = Maximum free space which can be reached without any risk of collision
  - generated using the distance to the object and a simplified model of the robot
  - bubbles are used to form a band of bubbles which connects the start point with the goal point

Obstacle Avoidance: Basic Curvature Velocity Methods (CVM)

- Adding physical constraints from the robot and the environment on the velocity space \((v, \omega)\) of the robot
  - Assumption that robot is traveling on arcs \(c = \omega / v\)
  - Acceleration constraints:
  - Obstacle constraints: Obstacles are transformed in velocity space
  - Objective function to select the optimal speed

Obstacle Avoidance: Lane Curvature Velocity Methods (CVM)

- Improvement of basic CVM
  - Not only arcs are considered
  - lanes are calculated trading off lane length and width to the closest obstacles
  - Lane with best properties is chosen using an objective function

- Note:
  - Better performance to pass narrow areas (e.g., doors)
  - Problem with local minima persists

Obstacle Avoidance: Dynamic Window Approach

- The kinematics of the robot is considered by searching a well chosen velocity space
  - velocity space -> some sort of configuration space
  - robot is assumed to move on arcs
  - ensures that the robot comes to stop before hitting an obstacle
  - objective function is chosen to select the optimal velocity

\[ O = a \cdot \text{heading}(v, \omega) + b \cdot \text{velocity}(v, \omega) + c \cdot \text{dist}(v, \omega) \]
Obstacle Avoidance: Global Dynamic Window Approach

- Global approach:
  - This is done by adding a minima-free function named NF1 (wave-propagation) to the objective function $O$ presented above.
  - Occupancy grid is updated from range measurements.

Obstacle Avoidance: The Schlegel Approach

- Some sort of a variation of the dynamic window approach
  - Takes into account the shape of the robot
  - Cartesian grid and motion of circular arcs
  - NF1 planner
  - Real-time performance achieved by use of precalculated table.
Control decomposition

- Pure serial decomposition

- Pure parallel decomposition

Sample Environment

Our basic architectural example

General Tiered Architecture

- Executive Layer
  - activation of behaviors
  - failure recognition
  - re-initiating the planner
- Deep Space One
A Two-Tiered Architecture for Off-Line Planning

Executive

Real-time controller
behavior 1  behavior 2  behavior 3
PID motion control

Robot Hardware

A Three-Tiered Episodic Planning Architecture.

Path planning

Local knowledge

Executive

Real-time controller
behavior 1  behavior 2  behavior 3
PID motion control

Robot Hardware

• Planner is triggered when needed: e.g. blockage, failure

An integrated planning and execution architecture

Global knowledge, map

Global Executive

Real-time controller
behavior 1  behavior 2  behavior 3
PID motion control

Robot Hardware

Example: The RoboX Architecture

• All integrated, no temporal decomposition between planner and executive layer
Summary

• This lecture looked at:
  ➢ Path planning
  ➢ Obstacle avoidance
  ➢ Navigation
• We revisited some of the map representations from previous lectures and showed how paths through these maps could be determined.
• We discussed obstacle avoidance, and described how obstacle avoidance techniques could be integrated into path execution.
  ➢ Combined, path planning, execution and obstacle avoidance equal navigation
• Finally, we looked at robot control architectures and how they can implement navigation.