Planning and Navigation

Where am I going? How do I get there?

- Localization
  - Environment Model
  - Local Map

- Perception
  - Real World Environment

- "Position" Global Map
- Cognition
  - Path
  - Motion Control
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. sales man 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 in 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, except the low lever motion control and localization.

- We can generally distinguish between (global) path planning and (local) obstacle avoidance.
Global Path Planing

• 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 $q_i$.

- What is the configuration space of a mobile robot?
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
   - Imposing a mathematical function over the space
• 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: Voronoi, Sysquake Demo
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

- Robot is treated as a point under the influence of an artificial potential field.
  - Generated robot movement is similar to a ball rolling down the hill
  - Goal generates attractive force
  - Obstacle are repulsive forces
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 $F(q)$
  
  \[
  F(q) = -\nabla U(q) = -\nabla U_{\text{att}}(q) - \nabla U_{\text{rep}}(q) = \begin{bmatrix}
  \frac{\partial U}{\partial x} \\
  \frac{\partial U}{\partial y}
  \end{bmatrix}
  \]

- 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

$$U_{att}(q) = \frac{1}{2} k_{att} \cdot \rho_{goal}^2(q)$$

- Attracting force converges linearly towards 0 (goal)

$$F_{att}(q) = -\nabla U_{att}(q)$$

$$= -k_{att} \cdot \rho_{goal}(q) \nabla \rho_{goal}(q)$$

$$= -k_{att} \cdot (q - q_{goal})$$
Potential Field Path Planning: Repulsing Potential Field

- Should generate a barrier around all the obstacle
  - strong if close to the obstacle
  - not influence if fare 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

\[ F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} k_{rep} \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} \frac{q - q_{goal}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases} \]
Potential Field Path Planning: Sysquake Demo

• Notes:
  - Local minima problem exists
  - Problem is getting more complex if the robot is not considered as a point mass
  - If objects are convex there exists situations where several minimal distances exist → can result in oscillations
Potential Field Path Planning: **Extended Potential Field Method**

**Khatib and Ch**

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

**Rotation potential field**

- force is also a function of robots orientation to the obstacle

**Task potential field**

- Filters out the obstacles that should not influence the robots movements, i.e. only the obstacles in the sector Z in front of the robot are considered
Potential Field Path Planning: Potential Field using a Dyn. Model

- Forces in the polar plane
  - no time consuming transformations

- Robot modeled thoroughly
  - potential field forces directly acting on the model
  - filters the movement -> smooth

- Local minima
  - set a new goal point
Potential Field Path Planning: Using Harmonic Potentials

- Hydrodynamics analogy
  - robot is moving similar to a fluid particle following its stream
- Ensures that there are no local minima

- Note:
  - Complicated, only simulation shown
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 boards 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.

![](image.png)
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 lowest cost function $G$ is selected

$$G = a \cdot \text{target\_direction} + b \cdot \text{wheel\_orientation} + c \cdot \text{previous\_direction}$$
Obstacle Avoidance: Vector Field Histogram + (VFH+)

- Accounts also in a very simplified way for the moving trajectories (dynamics)
  - robot moving on arcs
  - obstacles blocking a given direction also blocks all the trajectories (arcs) going through this direction

© R. Siegwart, I. Nourbakhsh
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 cannot 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

\[ B(p) = B(p, d(p, O)) \]
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

![Diagram showing obstacle avoidance in velocity space.](image)
Obstacle Avoidance: **Lane Curvature Velocity Methods (CVM)**

Simmons et al.

- 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

Fox and Burgard, Brock and Khatib

- 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
Obstacle Avoidance: The EPFL-ASL approach

- Dynamic window approach with global path planning
  - Global path generated in advance
  - Path adapted if obstacles are encountered
  - Dynamic window considering also the shape of the robot
  - Real-time because only max speed is calculated

- Selection (Objective) Function:
  \[
  \text{Max}( a \cdot \text{speed} + b \cdot \text{dist} + c \cdot \text{goal}\_\text{heading} )
  \]
  - \( \text{speed} = \frac{v}{v_{\text{max}}} \)
  - \( \text{dist} = \frac{L}{L_{\text{max}}} \)
  - \( \text{goal}\_\text{heading} = 1 - (\alpha - \omega T) / \pi \)

- Matlab-Demo
  - start Matlab
  - cd demoJan (or cd E:\demo\demoJan)
  - demoX

![Diagram of the EPFL-ASL approach]
Obstacle Avoidance: The EPFL-ASL approach
Obstacle Avoidance: Other approaches

• Behavior based
  - difficult to introduce a precise task
  - reachability of goal not provable

• Fuzzy, Neuro-Fuzzy
  - learning required
  - difficult to generalize
### Obstacle Avoidance

<table>
<thead>
<tr>
<th></th>
<th>Bug</th>
<th>Bubble band</th>
<th>Vector Field Histogram (VFH)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>kinematics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>dynamics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>view</td>
</tr>
<tr>
<td>point</td>
<td>point</td>
<td>point</td>
<td>C-space</td>
</tr>
<tr>
<td>exact</td>
<td>basic</td>
<td>basic</td>
<td>simplistic</td>
</tr>
<tr>
<td>local</td>
<td>local</td>
<td>local</td>
<td>local</td>
</tr>
<tr>
<td>local tangent graph</td>
<td>histogram grid</td>
<td>histogram grid</td>
<td>histogram grid</td>
</tr>
<tr>
<td>polygonal</td>
<td>polygonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>required</td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>range</td>
<td>tactile</td>
<td>tactile</td>
<td>sonars</td>
</tr>
<tr>
<td>various</td>
<td>various</td>
<td>nonholonomic (GuideCane)</td>
<td>nonholonomic (GuideCane)</td>
</tr>
<tr>
<td>6 ... 242 ms</td>
<td>6 ms</td>
<td>27 ms</td>
<td></td>
</tr>
<tr>
<td>66 MHz, 486 PC</td>
<td>66 MHz, 486 PC</td>
<td>20 MHz, 386 AT</td>
<td></td>
</tr>
<tr>
<td>efficient in many cases, robust</td>
<td>inefficient, robust</td>
<td>very inefficient, robust</td>
<td>fewer local minima</td>
</tr>
<tr>
<td>Dynamic window</td>
<td>Curvature velocity</td>
<td>method</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>circle</td>
<td>circle</td>
<td>circle</td>
<td>circle</td>
</tr>
<tr>
<td>(holonomic)</td>
<td>exact</td>
<td>exact</td>
<td>exact</td>
</tr>
<tr>
<td>basic</td>
<td>basic</td>
<td>basic</td>
<td>basic</td>
</tr>
<tr>
<td>global</td>
<td>local</td>
<td>local</td>
<td>local</td>
</tr>
<tr>
<td>obstacle line field</td>
<td>histogram grid</td>
<td>histogram grid</td>
<td></td>
</tr>
<tr>
<td>C-space grid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NF1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180° FOV SCK laser scanner</td>
<td>24 sonars ring, 56 infrared ring, stereo camera</td>
<td>24 sonars ring, 30° FOV laser</td>
<td>24 sonars ring, 30° FOV laser</td>
</tr>
<tr>
<td>holonomic (circular)</td>
<td>synchro-drive (circular)</td>
<td>synchro-drive (circular)</td>
<td>synchro-drive (circular)</td>
</tr>
<tr>
<td>6.7 ms</td>
<td>250 ms</td>
<td>125 ms</td>
<td>125 ms</td>
</tr>
<tr>
<td>450 MHz, PC</td>
<td>486 PC</td>
<td>200 MHz, Pentium</td>
<td>68 MHz, 486 PC</td>
</tr>
<tr>
<td>turning into corridors</td>
<td>local minima</td>
<td>local minima</td>
<td>local minima, turning into corridors</td>
</tr>
</tbody>
</table>

© R. Siegwart, I. Nourbakhsh
### Table: Gradient Method

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>circle</td>
<td>circle (but general formulation)</td>
<td>circle (but general formulation)</td>
<td>polygon</td>
<td>polygon</td>
<td>shape</td>
</tr>
<tr>
<td>exact</td>
<td>(holonomic)</td>
<td>(holonomic)</td>
<td>exact</td>
<td>exact</td>
<td>kinematics</td>
</tr>
<tr>
<td>basic</td>
<td></td>
<td></td>
<td>basic</td>
<td>basic</td>
<td>dynamics</td>
</tr>
<tr>
<td>global</td>
<td>global</td>
<td>local</td>
<td>local</td>
<td>global</td>
<td>view</td>
</tr>
<tr>
<td></td>
<td>grid</td>
<td>grid</td>
<td></td>
<td>grid</td>
<td>local map</td>
</tr>
<tr>
<td></td>
<td>local perceptual space</td>
<td>NF1</td>
<td></td>
<td>grid</td>
<td>global map</td>
</tr>
<tr>
<td>fused</td>
<td></td>
<td></td>
<td>graph (topological), NF1</td>
<td>wavefront</td>
<td>path planner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>180° FOV SCK laser scanner</td>
<td>sensors</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>180° FOV SCK laser scanner</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>nonholonomic (approx. circle)</td>
<td>holonomic (circular)</td>
<td>differential drive (octagonal, rectangular)</td>
<td>synchrodrive (circular), tricycle (forklift)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 ms (core algorithm: 10 ms)</td>
<td>180° FOV SCK laser scanner</td>
<td>2x 180° FOV SCK laser scanner</td>
<td>360° FOV laser scanner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>266 MHz, Pentium</td>
<td></td>
<td>100 ms (core algorithm: 22 ms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>380 MHz, G3</td>
<td>allows shape change</td>
<td>cycle time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>local minima</td>
<td>turning into corridors</td>
<td>architecture</td>
</tr>
</tbody>
</table>

© R. Siegwart, I. Nourbakhsh
Generic temporal decomposition

off-line

strategic

tactical

real-time

hard real-time
4-level temporal decomposition

Path planning 0.001 Hz

Range-based obstacle avoidance 1 Hz

Emergency stop 10 Hz

PID velocity control 150 Hz
Control decomposition

- Pure serial decomposition

- Pure parallel decomposition
Sample Environment
Our basic architectural example

- **Localization**
  - Environment Model
  - Local Map

- **Perception**
  - Real World Environment

- **Position**
- **Local Map**

- **Cognition**
  - Perception to Action
  - Obstacle Avoidance
  - Position Feedback
  - Path

- **Motion Control**
General Tiered Architecture

- Executive Layer
  - activation of behaviors
  - failure recognition
  - re-initiating the planner

```
Path planning

Executive

Real-time controller

<table>
<thead>
<tr>
<th>behavior 1</th>
<th>behavior 2</th>
<th>behavior 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PID motion control

Robot Hardware
```
A Tow-Tiered Architecture for Off-Line Planning

Executive

Real-time controller

<table>
<thead>
<tr>
<th>behavior 1</th>
<th>behavior 2</th>
<th>behavior 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID motion control</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Robot Hardware

© R. Siegwart, I. Nourbakhsh
A Three-Tiered Episodic Planning Architecture.

- Planner is triggered when needed: e.g. blockage, failure
An integrated planning and execution architecture

- All integrated, no temporal between planner and executive layer
Example: The RoboX Architecture
Example: RoboX @ EXPO.02