Lecture Overview

Mobile Robot Control Scheme

- Localization Map Building
  - Environment model
  - Local map
  - "Position"
global map

- Cognition Path Planning
  - Mission commands
  - Path

- Information Extraction
  - Raw data

- Sensing

Perception

- Real World Environment

Motion Control

- Path Execution
  - Actuator commands
  - Acting
ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Kinematics

Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza
Kinematics

Definitions

- Kinematics
  - Origin: *kinein* (Greek) – *to move*
  - The subfield of Mechanics dealing with motions of bodies

- Forward kinematics
  - Given is a set of actuator positions
  - Determine corresponding reference pose

- Inverse kinematics
  - Given is a desired reference pose
  - Determine corresponding actuator positions
From forward kinematics we know
\[
\begin{bmatrix}
x_g \\ y_g \\ z_g
\end{bmatrix}^T = h(\theta_1, \cdots, \theta_n)
\]

\(h\) is often not easily invertible in closed form

Approach: \textit{iteratively} perform the following steps

1. Start from a known forward-kinematic solution (e.g., via sampling) \(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}^T = h(\theta_1, \cdots, \theta_n)\)
2. Linearize \(h\) around \((\theta_1, \cdots, \theta_n)\), resulting in the Jacobian
\[
J = \begin{bmatrix}
\frac{\partial h_1}{\partial \theta_1} & \cdots & \frac{\partial h_1}{\partial \theta_n} \\
\cdots & \ddots & \cdots \\
\frac{\partial h_m}{\partial \theta_1} & \cdots & \frac{\partial h_m}{\partial \theta_n}
\end{bmatrix}_{\theta=\theta_i}
\]
3. Invert the Jacobian to obtain \(\begin{bmatrix} \Delta \theta_1 \\ \cdots \\ \Delta \theta_n \end{bmatrix}^T = J^{-1} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ \Delta z_i \end{bmatrix}^T\)
4. Move by \(\Delta\) in direction \(\begin{bmatrix} x_g - x_i \\ y_g - y_i \\ z_g - z_i \end{bmatrix}^T\)
Kinematics *(Wheeled) Non-Holonomic Systems*

- Wheels
  - Are often subject to motion constraints
  - Often do not allow to compute kinematics directly

- Consequently, for most wheeled robots, actuator positions do not map to unique reference poses
  - There is *no direct* (i.e., instantaneous) *way to measure a robot’s position*
  - *Position must be integrated over time*, depends on the path taken

- Understanding mobile robotic motion requires an understanding of wheel constraints placed on the robot’s mobility
Fixed standard wheel

\[
\begin{align*}
\hat{\xi}^R &= \begin{bmatrix} \dot{x}^1 \\ \dot{y}^1 \\ \dot{\theta}^1 \end{bmatrix}^T \\
\begin{bmatrix} \sin(\alpha + \beta) & -\cos(\alpha + \beta) & -l \cos \beta \\
\cos(\alpha + \beta) & \sin(\alpha + \beta) & l \sin \beta \end{bmatrix} R(\theta)^T \hat{\xi}^I - r \dot{\phi} &= 0 \\
\begin{bmatrix} \cos(\alpha + \beta) & \sin(\alpha + \beta) & l \sin \beta \end{bmatrix} R(\theta)^T \hat{\xi}^I &= 0
\end{align*}
\]
Kinematics

Forward Kinematics for Wheels

\[
\dot{\theta} (-l)\cos(\beta)
\]

\[
\dot{\theta} (-l)\sin(\beta)
\]

\[
\nu = r \cdot \phi
\]

\[
\dot{\xi}^R = [\dot{x}^1 \, \dot{y}^1 \, \dot{\theta}^1]^T
\]

\[
\begin{bmatrix}
\sin(\alpha + \beta) - \cos(\alpha + \beta) - l \cos \beta
\end{bmatrix} R(\theta)^T \ddot{\xi}^l - r \dot{\phi} = 0
\]

\[
\begin{bmatrix}
\cos(\alpha + \beta) \sin(\alpha + \beta) \, l \sin \beta
\end{bmatrix} R(\theta)^T \ddot{\xi}^l = 0
\]
Differential Forward Kinematics
Concatenation of Constraints

- Given a wheeled robot
  - Each wheel imposes $\geq 0$ constraints on its motion
  - Only fixed and steerable standard wheels impose no-sliding constraints

- Suppose the robot has $N_f + N_s$ standard wheels of radius $r_i$, then the individual wheel constraints can be concatenated in matrix form
  - Rolling constraints
    \[ J_1(\beta_s)R(\theta)\ddot{\xi}_l + J_2\dot{\varphi} = 0 \]
    \[ \varphi(t) = \begin{bmatrix} \varphi_f(t) \\ \varphi_s(t) \\ (N_f+N_s) \times 1 \end{bmatrix}, \quad J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix}, \quad J_2 = \text{diag}(r_1 \cdots r_N) \]
  - No-sliding constraints
    \[ C_1(\beta_s)R(\theta)\ddot{\xi}_l = 0 \]
    \[ C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix}, \quad C_{1s}(\beta_s) \] \hspace{1cm} (N_f+N_s) \times 3 \]

- Solving for $\ddot{\xi}_l$ results in an expression for differential forward kinematics
Five Basic Types of Three-Wheel Configurations

- Degree of mobility $\delta_m = 3$ - Number of independent wheel constraints
- Degree of steerability $\delta_s$
- Robots maneuverability $\delta_M = \delta_m + \delta_s$

<table>
<thead>
<tr>
<th>Configuration</th>
<th>$\delta_M$</th>
<th>$\delta_m$</th>
<th>$\delta_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omnidirectional</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Differential</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Omni-Steer</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Tricycle</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Two-Steer</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
ETH Master Course: 151-0854-00L

Autonomous Mobile Robots
Perception: Sensors Overview

Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza
Sensors Overview
Sensor Outline

- Optical encoders
- Heading sensors
  - Compass
  - Gyroscopes
- Accelerometer
- IMU
- GPS
- Range sensors
  - Sonar
  - Laser
  - Structured light
- Vision
Sensors Overview

Inertial Measurement Unit (IMU)

- **Definition**
  - An inertial measurement unit (IMU) is a device that uses measurement systems such as gyroscopes and accelerometers to estimate the relative position (x, y, z), orientation (roll, pitch, yaw), velocity, and acceleration of a moving vehicle with respect to an inertial frame.

- In order to estimate motion, the gravity vector must be subtracted. Furthermore, initial velocity has to be known (this is done by starting moving from a rest position).
Sensors Overview

Range sensors

- Sonar
- Laser range finder
- Time of Flight Camera
- Structured light
Robot Vision

*From World to Pixel coordinates*

- Mapping of world points to pixels in the image

- Perspective projection
  - Convert a 3D point in camera coordinates $P_c$ to pixel coordinates
  - Generalize the projection for any 3D point $P_w$ in world coordinates
  - Use projection equations for *camera calibration*
- **Summary**
  - Stereo imaging can give us *scale*
  - "Triangulation": with known stereo-camera configuration, we can compute the 3D coordinates of a point seen in both images
  - *Epipolar constraint* for efficient & robustified search for correspondences
  - Use stereo processing for 3D scene reconstruction or computation of disparity maps
  - Generalize stereo processing to multiple cameras for *structure from motion*
Robot Vision

Salient Image Regions

- Correlation vs. Convolution
  - Use in template matching, smoothing and taking the derivate of an image

- Image filtering for *edge detection*

- Point Features: *Haris, Sift, FAST, BRIEF, BRISK* and their characteristics e.g. scale/rotation invariance, computational time

- Building and using the visual vocabulary for place recognition
Robot Vision

Error Propagation and Line Fitting

- Representing uncertainty for real-world data
- The *error propagation law* and its significance

- Line Fitting algorithms for image/laser point clouds:
  - Split-and-merge, Line regression, RANSAC, Hough Transform, …
  - How they work and their characteristics and applications
ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Localization

Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza
Consider a mobile robot moving in a known environment.

As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.

Due to odometry uncertainty, after some movement the robot will become very uncertain about its position.

To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot uses its on-board exteroceptive sensors (e.g. ultrasonic, laser, vision sensors) to make observations of its environment.

The robot updates its position based on the observation. Its uncertainty shrinks.

*Localization*

*The probabilistic localization problem*
In robot localization, we distinguish two update steps:

1. **ACTION** (or prediction) update:
   - the robot moves and estimates its position through its *proprioceptive* sensors. During this step, the robot uncertainty grows.

2. **PERCEPTION** (or meausurement) update:
   - the robot makes an observation using its *exteroceptive* sensors and corrects its position by opportunely combining its belief before the observation with the probability of making exactly that observation. During this step, the robot uncertainty shrinks.
Localization

Solution to the probabilistic localization problem

How do we solve the Action and Perception updates?

- Action update uses the Theorem of Total probability

\[
\text{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \text{bel}(x_{t-1}) \, dx_{t-1}
\]

- Perception update uses the Bayes rule

\[
\text{bel}(x_t) = \eta \cdot p(z_t \mid x_t) \overline{\text{bel}}(x_t)
\]

(because of the use of the Bayes rule, probabilistic localization is also called *Bayesian localization*)
Localization
*Markov versus Kalman*

Two approaches exist to represent the probability distribution and to compute the Total Probability and Bayes Rule during the Action and Perception phases.

<table>
<thead>
<tr>
<th>Markov</th>
<th>Kalman</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The configuration space is divided into many cells. The configuration space of a robot moving on a plane is 3D dimensional ( (x, y, \theta) ). Each cell contains the probability of the robot to be in that cell.</td>
<td>• The probability distribution of both the robot configuration and the sensor model is assumed to be continuous and <strong>Gaussian</strong>!</td>
</tr>
<tr>
<td>• The probability distribution of the sensors model is also discrete.</td>
<td>• Since a Gaussian distribution is only described by its mean value ( \mu ) and covariance ( \Sigma ), we need only to update ( \mu ) and ( \Sigma ). Therefore the computational cost is very low!</td>
</tr>
<tr>
<td>• During Action and Perception, all the cells are updated. Therefore, the computational cost is very high</td>
<td></td>
</tr>
</tbody>
</table>
### Localization

**Markov versus Kalman**

<table>
<thead>
<tr>
<th>Markov</th>
<th>Kalman</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROS</strong></td>
<td><strong>PROS</strong></td>
</tr>
<tr>
<td>• localization starting from any unknown position</td>
<td>• Tracks the robot and is inherently very precise and efficient</td>
</tr>
<tr>
<td>• recovers from ambiguous situation</td>
<td></td>
</tr>
<tr>
<td><strong>CONS</strong></td>
<td><strong>CONS</strong></td>
</tr>
<tr>
<td>• However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and calculation power can thus become very important if a fine grid is used.</td>
<td>• If the uncertainty of the robot becomes too large (e.g. collision with an object) the Kalman filter will fail and the position is definitively lost</td>
</tr>
</tbody>
</table>
Simultaneous Localization and Mapping

**SLAM - Summary**

- What is SLAM and how does it work?
- Graphical representation of SLAM and approaches to solve it
  - *Full graph optimization*
  - *Filtering*
  - *Keyframe-based* approaches
- Popular techniques, working principles and relative merits
  - *EKF SLAM* (via the MonoSLAM system)
  - *Particle Filtering SLAM*
  - *GraphSLAM*
ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Motion Planning
Work space
- The *physical space* we live and operate in
- Usually 3D on flat ground and 6D for flying robots

State space (Configuration space)
- The *space of configurations* an agent operates in
- One DoF for each non-redundant actuator

Application: manipulator arm
Application: wheeled robot
- Operation on ground results in a 3D config space: \((x, y, \theta)\)
- In practice / for simplicity often a holonomic point-mass is assumed
- Obstacles are inflated by the robot radius
### Motion Planning

**Dynamic Window Approach (DWA)**

- **Working Principle**
  - 2D evidence grid transformed into \((v, \omega)\) space based on inevitable collision states
  - Circular arcs and acceleration window \(V_d\) account for vehicle kinematics
  - Selection of \((v, \omega)\)-pair within \(V_r = V_s \cap V_a \cap V_d\) that maximizes objective function
    \[
    G(v, \omega) = \sigma(\alpha \cdot \text{heading}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{velocity}(v, \omega))
    \]

- **Properties**
  - Prone to local minima

---

![Diagram showing working principle and properties of DWA](image)
Overview
- Graph structure is a *discrete representation* $G(N,E)$
- Solves a least cost problem between two states on a (directed) graph

Limitations
- State space is discretized, hence completeness is at stake
- *Feasibility of paths* is often *not inherently encoded*

Algorithms Covered
- Breadth first
- Depth first
- Dijkstra
- A* and variants
- (D* and variants)
Global Planning

Breadth First Graph Search

- Working principle
  - Operates on a *FIFO queue* and a “*closed*” list
  - Backtracks optimal solution starting from goal state

- Properties
  - First goal state encountered is optimal, iff edge costs are identical
  - Optimal for arbitrary edge costs, iff expansion is continued until FIFO queue is empty
Discussion:
Exam

- Oral, 30 minutes

- 3 Questions selected drawn by dices
  - Application
  - Basics 1
  - Basics 2

- Questions given beforehand (Webpage, sent to all participants)

- Example:

| All terrain demining in unstructured environments | 3.2.3 Wheel kinematic constraints of the 5 wheel types, pro/cons of wheel types | 5.2.4 Odometric position estimation and error model for a differential drive robot and their use in Markov and EKF localization |