Mobile Robots
Localization

Gerald Steinbauer
Institute for Software Technology
Today’s Agenda

• Motivation for Localization
• Odometry
• Odometry Calibration
• Error Model
Robotics is Easy …
“self-localization the most fundamental problem to providing a mobile robot with autonomous capabilities” [Cox 1991]
RoboCup Four-Legged League
RoboCup Four-Legged League
Geometry Again

- we are looking for a **transformation** between
  - a **reference** frame (the environment)
  - a **robot** frame
- we use the following terms **interchangeably**
  - **transformation**
  - **pose** (position + orientation)
  - **location**
- **pose**
  - 2D space: 2D position + orientation (1 angle)
  - 3D space: 3D position + orientation (quaternion, Euler angles, roll-pitch-yaw)
Taxonomy of Localization Problems

- **position tracking**
  - initial pose is known, “only” track the change of pose

- **global localization**
  - the initial is unknown, the pose has to be estimated from scratch

- **kidnapped robot problem**
  - robot is teleported without telling it

- **passive/active localization**
  - passive: robot estimates its pose on the fly
  - active: robot is able to set actions to improve its pose estimation

- **dynamic versus static environment**
  - the environment may change over time

- **single robot versus multi robots**
Why Localization is Hard?

- in general the pose **cannot** be directly estimated
  - integration of local motions
  - match against landmarks
- landmarks or observations can be **ambiguous**
- sensors have systematic/nonsystematic errors
- action execution is not completely deterministic
General Concept

- Global position information (e.g., GPS)
- Robot-mounted local motion sensors (e.g., encoder, IMU, …)
- Environment knowledge (e.g., map, landmarks)
- Local observations (e.g., vision, laser scanner)
- Prediction of the position change (e.g., odometry)
- Fusion
- Matching
- Position update
Dead Reckoning/Odometry

- **simple** solution for position tracking or pose prediction
- **Odometry**
  - in general for *wheeled* robots
  - use *wheel encoder* to estimate change of pose
- **Dead Reckoning**
  - *historic* term – also used for marine application
  - means the *combination* of actual pose, heading, speed and time
  - in robotics use a *heading* sensor, e.g. compass or IMU
Ideal and Continuous Time

- we assume
  - a differential drive robot
  - a function over time for robot’s linear and rotational velocity, $v(t), \omega(t)$
  - ideal execution
  - continuous time

- then we can simply integrate the functions in two steps to get the pose
  - $\theta(t) = \int_0^t \omega(\tau) d\tau + \theta_0$
  - $x(t) = \int_0^t v(\tau) \cos(\theta(\tau)) d\tau + x_0$
  - $y(t) = \int_0^t v(\tau) \sin(\theta(\tau)) d\tau + y_0$
Example Constant Velocities

• constant functions:
  • \( v(t) = v \)
  • \( \omega(t) = \omega \)

• assume \( x_0 = y_0 = \theta_0 = 0 \)

• lead to a circle with radius \( \frac{v}{\omega} \)
  • \( x(t) = \frac{v}{\omega} \sin(\omega t) \)
  • \( y(t) = \frac{v}{\omega} - \frac{v}{\omega} \cos(\omega t) \)
  • \( \theta(t) = \omega t \)
Differential Drive Odometry

• in reality we cannot rely on motion execution
• solution: measure the motion
• equip both wheel with a wheel encoder
• the odometry sensor delivers
  • the traveled distance of the left and right wheel during the last sampling period: $\Delta s_l, \Delta s_r$
• measure the distances for a small time interval
• assume the robot drives on a circle during the interval
Discrete Time

- if we have **discrete** time
  - we assume equidistance time steps $t_0 = 0, t_{i+1} = t_i + \Delta T$
  - we assume **piecewise** constant functions for
  - robot travels on a constant **arc** between time steps
- we **discretely** update the poses
  - $x_{i+1} = x_i - \frac{v}{\omega} \sin(\theta_i) + \frac{v}{\omega} \sin(\theta_i + \omega \Delta T)$
  - $y_{i+1} = y_i + \frac{v}{\omega} \cos(\theta_i) - \frac{v}{\omega} \cos(\theta_i + \omega \Delta T)$
  - $\theta_{i+1} = \theta_i + \omega \Delta T$
Motion Model I

\[
\begin{align*}
\Delta y & = 0 \\
\end{align*}
\]
Motion Model II

- the average travel distance of the robot is
  \[ \Delta s = \frac{\Delta s_l + \Delta s_r}{2} \]
- the change of orientation is
  \[ \Delta \theta = \frac{\Delta s_r - \Delta s_l}{b} \]
- if the distances are small we can assume
  \[ \Delta d \approx \Delta s \]
- then the pose update is
  \[
  x_t = f(x_{t-1}, u_t) = \begin{bmatrix}
  x_{t-1} \\
  y_{t-1} \\
  \theta_{t-1}
\end{bmatrix} + \begin{bmatrix}
  \Delta s \cos\left(\theta_{t-1} + \frac{\Delta \theta}{2}\right) \\
  \Delta s \sin\left(\theta_{t-1} + \frac{\Delta \theta}{2}\right) \\
  \Delta \theta
\end{bmatrix}
  \]
Relation to Transformations

- odometry can also be represented as transformation
  - from a odometry coordinate system in the world
  - to a fixed robot-centric coordinate system
- ROS provides odometry as 3D transformation between the frames `odom` and `base_link`, rotation is represented as quaternion, usually `z`, `roll` and `pitch` assumed 0

- **2D example**

\[
A_{t-1,t} = \begin{bmatrix}
\cos\Delta\theta & -\sin\Delta\theta & \Delta s \cos\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) \\
\sin\Delta\theta & \cos\Delta\theta & \Delta s \sin\left(\theta_{t-1} + \frac{\Delta\theta}{2}\right) \\
0 & 0 & 1
\end{bmatrix}
\]

- discrete transformations can be easily combined

\[
A_{t-2,t} = A_{t-2,t-1}A_{t-1,t}
\]
Integration of Odometry

• odometry information is only available at distinct time steps

• we integrate the information to estimate the pose
  • we make an additional error due to integration
  • errors accumulate boundless
Systematic Errors in Odometry

• can be determined and corrected in general

• sources
  • unequal wheel diameters
  • actual diameter differs from nominal diameter
  • actual wheelbase differs from nominal wheelbase
  • misaligned wheels
  • finite encoder resolution
  • finite encoder sampling rate
Non-Systematic Errors in Odometry

- stochastic in their nature
- can not be corrected
- can be probabilistically modelled
- sources
  - travel over uneven floor
  - travel over unexpected obstacles on the floor
  - wheel slippage
    - slippery floor
    - to high acceleration
    - internal forces
    - non-point contact of wheel, e.g. tracks instead of wheels
Reasons for Odometry Errors

- ideal case
- different wheel diameters
- bump
- carpet

and many more ...
Calibration of Odometry

• deals with \textit{systematic} errors
• determine correction \textit{factors} to minimize the error
• classical approach UMBmark (\textit{square path experiment}) [Borenstein \& Feng 1996]
• for \textit{differential drive} robots mainly
Two Major Types of Errors

- **Type A** – uncertain wheelbase
  - \( E_b = \frac{b_{\text{actual}}}{b_{\text{nominal}}} \)
  - leads to an incorrect estimation of orientation
Two Major Types of Errors

- Type B – unequal wheel diameters
  - \( E_d = \frac{D_R}{D_L} \)
  - leads to an incremental orientation error
  - only relative not absolute in respect to the diameter
UMBmark I

- **standard** test to calibrate odometry

1. determine absolute **initial** position \((x_0, y_0)\) of the robot, reset odometry
2. drive the robot through a 4m x 4m square path in **CW** direction
   a) stop after a 4m straight line
   b) turn 90° on the spot
   c) drive **slowly**
3. record the absolute **end** position \((x_{abs}, y_{abs})\) and the odometry \((x_4, y_4)\)
4. repeat 1-3 \(n\) times, usually \(n = 5\)
   - repeat the procedure **CCW**
UMBmark II

• The procedure leads to $n$ positions differences for each CW and CCW
  • $\epsilon_x = x_{abs} - x_{odo}$
  • $\epsilon_y = y_{abs} - y_{odo}$

• The repetition allows to limit the influence of nonsystematic errors in the procedure

• The position usually form two clusters:
  • $x_{c.g.,cw/ccw} = \frac{1}{n} \sum_{i=1}^{n} \epsilon x_i,cw/ccw$
  • $y_{c.g.,cw/ccw} = \frac{1}{n} \sum_{i=1}^{n} \epsilon y_i,cw/ccw$

[Borenstein, Feng 1996]
Results for Type A Errors Only

- consider first a robot only affected by type A errors
- assume the initial position (0,0)
  - $\epsilon_x, \epsilon_y$ simply becomes $x_{c.g.}$ respectively $y_{c.g.}$
- approximate trigonometry for small angles: $L\sin\gamma \approx L\gamma$, $L\cos\gamma \approx L$
- CCW
  - $x_4 \approx -2La$
  - $y_4 \approx 2La$
- CW
  - $x_4 = -2La$
  - $y_4 = -2La$

[Borenstein, Feng 1996]
Results for Type B Error Only

- consider first a robot only affected by type B errors
- CCW
  - $x_4 \approx 2L\beta$
  - $y_4 \approx -2L\beta$
- CW
  - $x_4 = -2L\beta$
  - $y_4 = -2L\beta$

[Borenstein, Feng 1996]
Determine the Angles

- **superimpose** the results for the individual errors
  - \( x, \text{ CW} \): \(-2L\alpha - 2L\beta = -2L(\alpha + \beta) = x_{c.g,\text{cw}} \)
  - \( x, \text{ CCW} \): \(-2L\alpha + 2L\beta = -2L(\alpha - \beta) = x_{c.g,\text{ccw}} \)
  - \( y, \text{ CW} \): \(-2L\alpha - 2L\beta = -2L(\alpha + \beta) = y_{c.g,\text{cw}} \)
  - \( y, \text{ CCW} \): \(2L\alpha - 2L\beta = -2L(-\alpha + \beta) = y_{c.g,\text{ccw}} \)

- **calculate the angles** by combining
  - \( \beta = \frac{x_{c.g,\text{cw}} - x_{c.g,\text{ccw}}}{-4L} \)
  - \( \beta = \frac{y_{c.g,\text{cw}} + y_{c.g,\text{ccw}}}{-4L} \)
  - \( \alpha = \frac{x_{c.g,\text{cw}} + x_{c.g,\text{ccw}}}{-4L} \)
  - \( \alpha = \frac{y_{c.g,\text{cw}} - y_{c.g,\text{ccw}}}{-4L} \)
Determine the Correction Factors for $E_d$

- determine the ratio of wheel diameters $E_d$
- determine the radius of the curve traveled by the robot on the “straight” segment
  - \[ R = \frac{L/2}{\sin(\beta/2)} \]
- determine actual ratio
  - \[ E_d = \frac{D_R}{D_L} = \frac{R+b/2}{R-b/2} \]
- correct the odometry
  - assume the average wheel diameter $D_a = \frac{(D_L+D_R)}{2}$ stays constant
  - solving a system of two equations:
    - \[ D_L = \frac{2}{E_d+1} D_a, \quad D_R = \frac{2}{(1/E_d)+1} D_a \]
Determine the Correction Factors for $E_b$

- determine the ratio for the actual wheelbase $E_b$
- the wheelbase is inversely proportion to the amount of rotation
  - we can state the following proportion
    - $\frac{b_{\text{actual}}}{\pi/2} = \frac{b_{\text{nominal}}}{\pi/2-\alpha}$
- determine actual ratio
  - $E_b = \frac{\pi/2}{\pi/2-\alpha}$
- correct the odometry
  - $b_{\text{actual}} = \frac{\pi/2}{\pi/2-\alpha} b_{\text{nominal}}$
Description of Random Errors

- **modeling** of the nonsystematic (random) error is important for consistent perception
- **simple solution**: treat the observation/estimation $m$ as normal distributed random variable $m \sim \mathcal{N}(\mu_m, \sigma_m)$

\[
PDF(m) = \frac{1}{\sigma_m \sqrt{2\pi}} e^{-\frac{(m-\mu_m)^2}{2\sigma_m^2}}
\]

\[
P(a < m \leq b) = \int_a^b PDF(x)dx
\]

- if probability density function (PDF) is not Gaussian other means have to be used, e.g. sample-based
Nonlinear Error Propagation

- **question**: how is the error propagated by a nonlinear function $f$
  - assume vectors of normal distributed random variables, $x \sim \mathcal{N}(\mu_x, \Sigma_x)$, $y \sim \mathcal{N}(\mu_y, \Sigma_y)$, $\Sigma$ ... covariance matrix
  - assume function $f$ as vector of functions $y_i = f_i(x)$
- use the **Taylor-approximation** of $f$
Nonlinear Error Propagation

- Taylor-approximation of error propagation

\[ \mu_y = f(\mu_x) \]
\[ \Sigma_y = F_x \Sigma_x F_x^T \]

\[ F_x = \nabla f = [\nabla x f(x)^T]^T = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \]

\( F_x \) ... Jacobean Matrix
Differential Drive

\[
x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}
\]

\[
x_t = x_{t-1} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} = f(x_{t-1}, u_t)
\]
Differential Drive Odometry

- the odometry sensor delivers
  - the traveled distance of the left and right wheel during the last sampling period: $\Delta s_l, \Delta s_r$
- the average travel distance of the robot is
  - $\Delta s = \frac{\Delta s_l + \Delta s_r}{2}$
- the change of orientation is
  - $\Delta \theta = \frac{\Delta s_r - \Delta s_l}{b}$
- the pose update is
  $$x_t = f(x_{t-1}, u_t) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta s \cos \left( \theta_{t-1} + \frac{\Delta \theta}{2} \right) \\ \Delta s \sin \left( \theta_{t-1} + \frac{\Delta \theta}{2} \right) \\ \Delta \theta \end{bmatrix}$$
Error Model I

• we treat the pose $x$ as normal distributed vector of random variables
  • $x \sim \mathcal{N}(\mu_x, \Sigma_x)$

• we assume a covariance matrix to represent the nonsystematic errors of $\Delta s_l, \Delta s_r$
  • $\Sigma_\Delta = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$
  • the errors of $\Delta s_l$ and $\Delta s_r$ assumed as independent
  • the error is proportional the distance, $k_r$ and $k_l$ are error constants

• using Taylor-approximation we get the following error propagation
  • $\Sigma_{x_t} = F_{x_{t-1}} \Sigma_{x_{t-1}} F_{x_{t-1}}^T + F_{\Delta s} \Sigma_\Delta F_{\Delta s}^T$
Error Model II

• we use the following Jacobean matrices:

\[ F_{x_{t-1}} = \nabla f_{x_{t-1}} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta_s \sin(\theta + \theta/2) \\ 0 & 1 & \Delta_s \cos(\theta + \theta/2) \\ 0 & 0 & 1 \end{bmatrix} \]

• \[ F_{\Delta s} = \nabla_{\Delta s} f = \begin{bmatrix} \frac{\partial f}{\partial \Delta_r} & \frac{\partial f}{\partial \Delta_l} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \cos \left( \theta + \frac{\Delta \theta}{2} \right) - \frac{\Delta s}{2b} \sin \left( \theta + \frac{\Delta \theta}{2} \right) & \frac{1}{2} \cos \left( \theta + \frac{\Delta \theta}{2} \right) + \frac{\Delta s}{2b} \sin \left( \theta + \frac{\Delta \theta}{2} \right) \\ \frac{1}{2} \sin \left( \theta + \frac{\Delta \theta}{2} \right) + \frac{\Delta s}{2b} \cos \left( \theta + \frac{\Delta \theta}{2} \right) & \frac{1}{2} \sin \left( \theta + \frac{\Delta \theta}{2} \right) - \frac{\Delta s}{2b} \cos \left( \theta + \frac{\Delta \theta}{2} \right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \]
Growth of Uncertainty – Straight Line

errors perpendicular to the direction of movement are growing much faster!
Growth of Uncertainty – Circle

errors ellipse does not remain perpendicular to the direction of movement!

[Siegwart, Nourbakhsh, Scaramuzza, 2011, MIT Press]
Non-Gaussian Error Model

errors are not shaped like ellipses! use non-parametric representations, e.g. particles

[Fox, Thrun, Burgard, Dellaert, 2000]
Literature

Questions?
Thank you!