Humanoid Robotics

6D Localization of Humanoid Robots

Maren Bennewitz
Goal of this Chapter

- Motivation:
  - To perform useful service, a robot needs to know its pose within the environment
  - Motion commands are only executed inaccurately
- Know how to estimate the **6D pose of a humanoid** in a **given 3D model** of the environment
- **Based on onboard sensor data and odometry**
- Sensor data: distance information from depth images or laser range readings
Problem Description

- Estimate the robot’s 6D pose in a given 3D model of the environment
- Based on distance and odometry measurements
Recap: Basic Probability Rules

\[0 \leq p(x) \leq 1\]

Product rule (if \(x\) and \(y\) independent):
\[p(x, y) = p(x) \cdot p(y)\]

Bayes’ rule:
\[p(x | y) = \frac{p(y | x) \cdot p(x)}{p(y)}\]

Often written as:
\[p(x | y) = \eta \cdot p(y | x) \cdot p(x)\] Normalizing constant ensures that the posterior sums up to 1 over all possible values of \(x\)

In case of background knowledge, Bayes' rule turns into
\[p(x | y, e) = \eta \cdot p(y | x, e) \cdot p(x | e)\]
Recap: Basic Probability Rules

Law of total probability:

\[ p(x) = \int_y p(x \mid y) \cdot p(y) \, dy \]  \hspace{1cm} \text{continuous case}

\[ p(x) = \sum_y p(x \mid y) \cdot p(y) \]  \hspace{1cm} \text{discrete case}
Markov Assumption

Markov assumption

\[
p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)
\]

\[
p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)
\]
State Estimation

- Estimate the state $x$ given observations $z$ and odometry measurements/actions $u$

- **Goal:** Calculate the distribution

  $$p(x \mid z, u)$$

- Apply the recursive Bayes’ filter
Recursive Bayes Filter 1

\[ \text{bel}(x_t) = p(x_t | z_{1:t}, u_{1:t}) \]

Definition of the belief

all data up to time \( t \)
Recursive Bayes Filter 2

\[ \text{bel}(x_t) = p(x_t | z_{1:t}, u_{1:t}) \]
\[ = \eta \ p(z_t | x_t, z_{1:t-1}, u_{1:t}) \ p(x_t | z_{1:t-1}, u_{1:t}) \]

Bayes' rule
Recursive Bayes Filter 3

\[
bel(x_t) = p(x_t | z_{1:t}, u_{1:t})
= \eta p(z_t | x_t, z_{1:t-1}, u_{1:t}) p(x_t | z_{1:t-1}, u_{1:t})
= \eta p(z_t | x_t) p(x_t | z_{1:t-1}, u_{1:t})
\]

Markov assumption
Recursive Bayes Filter 4

\[
\text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \\
= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
\, dx_{t-1} p(x_{t-1} \mid z_{1:t-1}, u_{1:t})
\]

Law of total probability
Recursive Bayes Filter 5

\[ \text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \]

Markov assumption
Recursive Bayes Filter 6

\[ bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \]
\[ = \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \]
\[ = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \]
\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \]
\[ \cdot p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \]
\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \]
\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \, dx_{t-1} \]
Recursive Bayes Filter 7

\[ bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \]

\[ p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \]

\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \, dx_{t-1} \]

\[ = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \, dx_{t-1} \]

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

recursive term
Recursive Bayes Filter 7

\[ \text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \]

\[ = \eta \ p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) \ p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta \ p(z_t \mid x_t) \ p(x_t \mid z_{1:t-1}, u_{1:t}) \]

\[ = \eta \ p(z_t \mid x_t) \ \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \]

\[ \hspace{1cm} p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \ dx_{t-1} \]

\[ = \eta \ p(z_t \mid x_t) \ \int p(x_t \mid x_{t-1}, u_t) \ p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) \ dx_{t-1} \]

\[ = \eta \ p(z_t \mid x_t) \ \int p(x_t \mid x_{t-1}, u_t) \ p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \ dx_{t-1} \]

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

\[ \text{observation model} \quad \text{motion model} \]
Probabilistic Motion Model

- Robots execute motion commands only inaccurately
- The motion model specifies the probability that action $u_t$ moves the robot from $x_t$ to $x_{t-1}$:

$$p(x_t \mid u_t, x_{t-1})$$

- Defined for each robot type individually
Observation Model for Range Readings

- Sensor data consists of \( k \) measurements (subsample)

\[
\mathbf{z}_t = \{z^1_t, \ldots, z^k_t\}
\]

- The individual measurements are independent given the robot’s pose

\[
p(\mathbf{z}_t \mid x_t) = \prod_{i=1}^{k} p(z^i_t \mid x_t)
\]

- “How well can the distance measurements be explained given the pose and the map”
Particle Filter

- One implementation of the recursive Bayes’ filter
- Non-parametric framework (not only Gaussian)
- Arbitrary models can be used as motion and observation models
Key Idea: Samples

Use a set of weighted samples to represent arbitrary distributions
Particle Set

Set of weighted samples (particles)

\[ \mathcal{X} = \left\{ \left\langle x[j], w[j] \right\rangle \right\}_{j=1,\ldots,J} \]
Particle Filter

- The set of weighted particles approximates the belief about the robot’s pose
- **Prediction**: Sample from the motion model (propagate particles forward)
- **Correction**: Weigh the samples based on the observation model
Monte Carlo Localization

- **Each particle is a pose hypothesis**

- **Prediction**: For each particle, sample a new pose from the motion model

  \[ x_t^{[j]} \sim p(x_t \mid x_{t-1}^{[j]}, u_t) \]

- **Correction**: Weigh samples according to the observation model

  \[ w_t^{[j]} \propto p(z_t \mid x_t^{[j]}) \]

- **Resampling**: Draw sample \( i \) with probability \( w_t^{[i]} \) and repeat times (\( J = \# \) particles)
MCL – Correction Step

Image Courtesy: S. Thrun, W. Burgard, D. Fox
MCL – Resampling & Prediction

Image Courtesy: S. Thrun, W. Burgard, D. Fox
MCL – Correction Step

Image Courtesy: S. Thrun, W. Burgard, D. Fox
MCL – Resampling & Prediction

Image Courtesy: S. Thrun, W. Burgard, D. Fox
Wrap Up: Particle Filter for Localization

- Particle filters are non-parametric, recursive Bayes filters
- The belief about the state is represented by a set of weighted samples
- The motion model is used to draw the samples for the next time step
- The weights of the particles are computed using the observation model
- Resampling is carried out based on weights
- Also called: Monte-Carlo localization (MCL)
6D Localization for Humanoids

3D environments require a 6D pose estimate

\[ x = (x, y, z, \varphi, \theta, \psi) \]

2D position  height  yaw, pitch, roll

estimate the 6D torso pose
6D Localization for Humanoids

- Estimate the belief about the robot’s pose using MCL
- The probability distribution is represented by pose hypotheses (particles)
- Needed: motion model and observation model
Motion Model \( p(x_t \mid x_{t-1}, u_t) \)

- The odometry estimate \( u_t \) corresponds to the incremental motion of the torso
- \( u_t \) is computed by forward kinematics (FK) from the current stance foot while walking
Kinematic Walking Odometry

- **Goal:** Estimate the robot’s torso pose while walking
- Keep track of the transform to the current stance foot

![Diagram of a biped robot](image)

frame of the current stance foot

frame of the torso, transform can be computed with FK over the right leg

Figure 3.1: Coordinate frames for kinematic walking odometry of a biped robot under ideal conditions. Initially, the frame $F_{\text{rfoot}}$ of the right stance foot is fixed in the world frame $F_{\text{odom}}$ and the torso frame $F_{\text{torso}}$ can be computed with forward kinematics over the right leg. In the next step, the left swing leg touches the ground. While both feet remain on the ground, the left foot frame $F_{\text{lfoot}}$ is computed with forward kinematics over both leg chains. Finally, for the next stepping phase the left leg becomes the stance leg and $F_{\text{lfoot}}$ remains static with respect to $F_{\text{odom}}$, which creates a new reference to compute $F_{\text{torso}}$. Filter can be easily parallelized and it can directly benefit from more computational power by increasing the number of particles. Thus, the possible estimation accuracy directly scales with the available computational resources.

3.2. Motion Model

In the prediction step of MCL, a new pose is drawn for each particle according to the motion model $p(x_t | x_{t-1}, u_t)$. In our case, $u_t$ corresponds to the incremental motion of the humanoid’s torso in a local coordinate frame and is computed with forward kinematics from the current stance foot while walking. This kind of odometric estimate is usually referred to as kinematic odometry.

Figure 3.1 shows a single stepping phase and the coordinate frames used to compute the kinematic odometry. As illustrated in Figure 3.2, $u_t$ can then be computed from two consecutive torso poses $e_x^{t-1}$ and $e_x^t$, which are 6D rigid body transforms in the odometry coordinate frame $F_{\text{odom}}$:

$$T(u_t) = (e_x^{t-1})^1_{\text{odom}} e_x^t.$$  \hspace{1cm} (3.7)

Here, $T(u_t)$ denotes the 6D rigid body transform of the odometry motion $u_t$. On the other hand, we can think of the humanoid’s odometric pose estimates as a concatenation of the incremental motions $e_x^{t-1} = e_x^{t-1} 1 T(u_t)$.

Under ideal conditions the stance foot of the robot rests firmly and precisely on the ground, leading to an accurate 6D pose estimate with kinematic odometry. However, the feet of the robot typically slip on the ground due to inaccurate execution of motions. Additionally, gear backlash in individual joints can aggravate the problem. Hence, in practice,
**Kinematic Walking Odometry**

Both feet on the ground, compute the transform to the frame of the left foot with FK

![Diagram of kinematic walking odometry](image)
Kinematic Walking Odometry

The left leg becomes the stance leg and is the new reference to compute the transform to the torso frame.
Kinematic Walking Odometry

- Using FK, the poses of all joints and sensors can be computed **relative to the stance foot** at each time step.
- The **transform from the odometry frame to the stance foot** is updated whenever the swing foot becomes the new stance foot.
Odometry Estimate

Odometry estimate $u_t$ from two consecutive torso poses
Odometry Estimate

- The incremental motion $u_t$ of the torso is computed from kinematics of the legs.
- Typical error sources: slippage on the ground and backlash in the joints.
- Accordingly, we have only noisy odometry estimates while walking and the drift accumulates over time.
- The particle filter has to account for that noise within the motion model.
**Motion Model**  \( p(x_t \mid x_{t-1}, u_t) \)

- Noise modeled as a Gaussian with systematic drift on the 2D plane
- Prediction step samples a new pose for each particle \( i \) according to

\[
x_t^{[i]} = x_{t-1}^{[i]} \oplus \mathcal{N}(Mu_t, \Sigma_u)
\]

- \( M \) learned with least squares optimization using ground truth data (as in Ch. 2)
Sampling from the Motion Model

We need to **draw samples from a Gaussian**

\[ x_t[i] = x_{t-1}[i] \oplus \mathcal{N}(\mu, \Sigma) \]

Gaussian
How to Sample from a Gaussian

- Drawing from a 1D Gaussian can be done in closed form

\[ \mu + \frac{1}{2} \sum_{i=1}^{12} \text{rand}(-\sigma, \sigma) \]

Example with $10^6$ samples
How to Sample from a Gaussian

- Drawing from a 1D Gaussian can be done in closed form

\[
\mu + \frac{1}{2} \sum_{i=1}^{12} \text{rand}(-\sigma, \sigma)
\]

- If we consider the individual dimensions of the motion to be independent, we can draw each dimension using a 1D Gaussian

\[
x_t^{[i]} = x_{t-1}^{[i]} \oplus \mathcal{N}(\mu, \Sigma)
\]

draw each of the dimensions independently
Sampling from the Odometry

- We sample an **individual motion for each particle**
  \[ u_t^{[i]} \sim \mathcal{N}(M u_t, \Sigma u_t) \]

- We then incorporate the **sampled** motion \( u_t^{[i]} \) into the pose of particle \( i \)
  \[ x_t^{[i]} = x_{t-1}^{[i]} \oplus u_t^{[i]} \]
**Observation Model** \( p(o_t \mid x_t) \)

- Observation \( o_t \) consists of independent
  - range measurements \( r_t \),
  - height \( \tilde{z}_t \) (computed from the values of the joint encoders),
  - and roll \( \tilde{\varphi}_t \) and pitch \( \tilde{\psi}_t \) measurements (by IMU)

- Observation model:
  \[
p(o_t \mid x_t) = p(r_t, \tilde{z}_t, \tilde{\varphi}_t, \tilde{\psi}_t \mid x_t)
  \]
Observation Model \( p(o_t \mid x_t) \)

\[
p(o_t \mid x_t) = p(r_t, \tilde{z}_t, \tilde{\varphi}_t, \tilde{\psi}_t \mid x_t) = \\
p(r_t \mid x_t) p(\tilde{z}_t \mid x_t) p(\tilde{\varphi}_t \mid x_t) p(\tilde{\psi}_t \mid x_t)
\]
Observation Model \( p(o_t | x_t) \)

\[
p(o_t | x_t) = p(r_t, \tilde{z}_t, \tilde{\phi}_t, \tilde{\psi}_t | x_t) = \prod_{i} p(r_{ti} | x_t) \, p(\tilde{z}_t | x_t) \, p(\tilde{\phi}_t | x_t) \, p(\tilde{\psi}_t | x_t)
\]

**Range data**: Ray-casting or endpoint model in 3D map
Observation Model: Simple Ray-Cast Model

- Compare the measured distances with the expected distances from the particle pose to obstacles in the map.
- Consider the first obstacle **along the ray** in the map.
- Use a Gaussian to evaluate the difference.

![Diagram showing a simple ray-cast model with measured distance and object in map.]
Observation Model: Beam-Endpoint Model

Evaluate the distance of the hypothetical beam end point to the closest obstacle in the map with a Gaussian.
Observation Model \( p(\mathbf{o}_t | \mathbf{x}_t) \)

\[
p(\mathbf{o}_t | \mathbf{x}_t) = p(\mathbf{r}_t, \tilde{\mathbf{z}}_t, \tilde{\varphi}_t, \tilde{\psi}_t | \mathbf{x}_t) =
\]

\[
p(\mathbf{r}_t | \mathbf{x}_t) p(\tilde{\mathbf{z}}_t | \mathbf{x}_t) p(\tilde{\varphi}_t | \mathbf{x}_t) p(\tilde{\psi}_t | \mathbf{x}_t)
\]

- **Torso height**: Compare measured value from kinematics to predicted height from motion model
- **IMU data**: Compare measured roll and pitch to the predicted angles
- **Use individual Gaussians to evaluate the difference**
Likelihoods of Measurements

Gaussian distribution

\[ \phi(d, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left( -\frac{d^2}{2\sigma^2} \right) \]

height \hspace{1cm} p(\tilde{z}_t \mid x_t) = \phi(z_t - \tilde{z}_t, \sigma_z) \hspace{1cm} \text{standard deviation corresponding to noise characteristics of joint encoders and IMU}

roll \hspace{1cm} p(\tilde{\varphi}_t \mid x_t) = \phi(\varphi_t - \tilde{\varphi}_t, \sigma_\varphi)

pitch \hspace{1cm} p(\tilde{\theta}_t \mid x_t) = \phi(\theta_t - \tilde{\theta}_t, \sigma_\theta)
Localization Evaluation: 2D Laser Scanner

- Trajectory of 5m, 10 runs each
- Ground truth from external motion capture system
- Raycasting results in a significantly smaller error
- Calibrated motion model requires fewer particles
Summary

- Estimation of a humanoid’s 6D torso pose in a given 3D model using a particle filter
- Motion estimate from kinematic walking odometry
- Sample a motion for each particle individually using the motion model
- Use individual Gaussians in the observation model to evaluate the differences between measured and expected values
- Particle filter allows to locally track and globally estimate the robot’s pose
Literature

- Book: Probabilistic Robotics, S. Thrun, W. Burgard, and D. Fox

Acknowledgment

- Previous versions of parts of the slides have been created by Wolfram Burgard, Armin Hornung, and Cyrill Stachniss