Broad Overview & Sample Exam Questions

Maren Bennewitz
About These Slides

- Most important topics
- No guarantee for completeness regarding the exam
- Can be seen as basis for a good grade
Least Squares
Problem Definition

- Given a system described by a set of \( n \) observation functions

\[
\{f_i(x)\}_{i=1:n}
\]

- Let
  - \( x \) be the state vector (to be estimated)
  - \( z_i \) be a measurement of the state \( x \)
  - \( \hat{z}_i = f_i(x) \) be a function that maps \( x \) to a predicted measurement \( \hat{z}_i \)

- **Given** \( n \) noisy measurements \( z_{1:n} \) about the state \( x \)
- **Goal:** Estimate the state \( x \) that best explains the \( z_{1:n} \) measurements
Error Function

- Error $e_i(x)$ is the **difference** between the **predicted** and the **actual** measurement

\[ e_i(x) = z_i - f_i(x) \]

- Assumption: The error has **zero mean** and is **normally distributed**
- Gaussian error with information matrix $\Omega_i$
- The **squared error** of a measurement depends on the state and is a **scalar**

\[ e_i(x) = e_i(x)^T \Omega_i e_i(x) \]
Goal: Minimize the Squared Error

- Find the state $x^*$ that minimizes the error over all measurements

$$
x^* = \arg\min_x F(x) \quad \text{global error (scalar)}
$$

$$
= \arg\min_x \sum_i e_i(x) \quad \text{squared error terms (scalar)}
$$

$$
= \arg\min_x \sum_i e_i^T(x) \Omega_i e_i(x) \quad \text{error terms (vector)}
$$
Solve via Iterative Local Linearizations: Gauss-Newton

- Linearize the error terms around the current /initial guess
- Compute the first derivative of the approximated global error function
- Set it to zero and solve the linear system
- Obtain the new state
- Iterate
Minimize the Global Error

- Approximate the error functions around the current guess $x$ via **Taylor expansion**
  \[ e_i(x + \Delta x) \approx e_i(x) + J_i(x) \Delta x \]

  Jacobian of the error function

- Write the global error as a quadratic form
- Compute the derivative, obtain linear system
- Set it to zero, solve the linear system
Questions

- Describe the class of problems to which least squares can be applied
- How is the error defined?
- Sketch the error minimization process with Gauss Newton
- Write down the equations and explain how the best increment is obtained
Projective Geometry
Homogenous Coordinates
Homogeneous Coordinates

- H.C. can simplify mathematical expressions
- Allow for easy chaining and inversion of transformations
- Modeled through an extra dimension
Questions

- How can homogeneous coordinates be obtained from Euclidean coordinates, and vice versa?
- Why do we use homogeneous coordinates instead of Euclidean coordinates?
- How does a rigid body transformation look like?
Camera Calibration
Calibration

- Assumption: pinhole camera model
- A camera projects 3D world points onto the 2D image plane
- Calibration: Finding the parameters of the camera that affect this process
- **Extrinsics**: describe the pose of the camera in the world
- **Intrinsics**: describe the mapping of the scene in front of the camera to the pixels on the sensor
Questions

- Describe the pinhole camera model
- What are the assumptions in this model?
- Which parameters do we need to calibrate?
- Write down the calibration matrix for the intrinsic parameters
Question: Explain the mapping
Humanoid Calibration
Parameters to Estimate

- True position of a joint: \[ q = \hat{q} + q^{off} \]
- Estimated joint offsets:
  \[ q^{off} = \begin{pmatrix} q_1^{off} \\ q_2^{off} \\ \vdots \\ q_n^{off} \end{pmatrix} \]
- Camera extrinsics (wrt. the reference frame neck joint): \( R, t \)
- Camera intrinsics: \( c, m, x_H, y_H, q \)
Least-Squares Optimization Problem

- Marker observations in camera image

- Optimize the parameters so that the reprojection error is minimized

projection of robot’s hand based on the estimated parameters
Formulation as Least-Squares Optimization

- Consider a set of $n$ different robot configurations with encoder readings $\hat{q}_i$

- The error function is

$$e_i(\theta, z_i, \hat{q}_i) = z_i - \text{predictmarker}_{M_{EEF}}(\theta, \hat{q}_i)$$

- Measurement = observed marker position in the image
- Predicted marker position in the image given the robot's kinematic structure
- Current values of calibration parameters
- Joint readings
Predicted Marker Location

The predicted position of a marker $M_{EEF}$ attached to the end-effector $EEF$ is given by

$$\text{predictmarker}_{M_{EEF}}(\theta, \hat{q}_i) = K_\theta[R_\theta| - R_\theta t_\theta]F^N_{EEF}(\theta, \hat{q}_i)\tilde{m}_{EEF}$$

- **calibration matrix with current estimates of intrinsic camera parameters**
- **current estimates of extrinsic camera parameters**
- **forward kinematics**: transformation from $EEF$ frame into neck frame from joint encoder readings and estimates of the offsets
- **homogenous coordinates of marker position**
Forward Kinematics: Transformation

The complete transform from $E$ to $B$ corresponding to forward kinematics:

$$\mathcal{F}_E^B(\theta, \hat{q}) = T_1^B \left( \prod A_i^{i-1}(q_i) \right) T_E^n$$

- Transformation depending on joint encoder readings and current estimates of the offsets
- Transformation matrix for the transform between the joints $i$ and $i - 1$ given $q_i$
- Position of joint $i$ according to $\hat{q}$ and $\theta$
Questions

- What kind of parameters to estimate?
- Define the error function
- Explain the problem of calibration using the image below
- How is the predicted marker location computed?
- Explain forward kinematics
3D World Representations
Questions

- Which 3D world representations exist? Briefly explain how to build them.
- What are their advantages / drawbacks?
- Explain the ICP algorithm and how the error function is defined
- How can the SVD be used to find the best rotation and translation?
6D Localization of Humanoid Robots
Recursive Bayes Filter

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

Definition of the belief

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

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

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

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

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

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

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

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

\[ = \eta p(z_t | x_t) \int p(x_t | x_{t-1}, u_t) \, bel(x_{t-1}) \, dx_{t-1} \]
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 (\(J=\#\text{particles}\))
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
### Kinematic Walking Odometry

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

![Diagram of robot walking with labeled frames](image)

- Frame of the torso, transform can be computed with FK over the right leg
- Frame of the current stance foot
Kinematic Walking Odometry

Both feet on the ground, compute the transform to the frame of the left foot with FK.
Kinematic Walking Odometry

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

Odometry estimate $u_t$ from two consecutive torso poses

Figure 3.2: Odometry estimate from two consecutive torso poses during the walking cycle.

Only noisy estimates of the odometry are available while walking and a substantial amount of drift accumulates over time. Consequently, a particle filter has to account for that noise with a higher variance, requiring a higher number of particles and thus more computational power for successful pose estimation. By learning the motion model parameters instead, the localization performance can be increased, both in terms of computational load as well as accuracy.

Here, we consider the most general case of any kind of 3D positional and rotational displacement, for instance originating from an omnidirectional walking engine. We furthermore assume that systematic drift affects the motion reported by odometry in the 2D plane, i.e., only $(e_x, e_y, e_y)$ are affected from $e_x = (e_x, e_y, e_z, e_j, e_q, e_y)$. This is not a strong restriction as long as the humanoid walks on a solid surface, since its motion is constrained by this surface and it cannot leave the ground. Even when climbing challenging terrain such as stairs, the drift of the motion occurs in the 2D plane of the stance leg as long as the robot does not fall or slide down a slope. General noise in the kinematic estimate of the humanoid's height above the ground does not lead to a systematic drift.

3.2.1. Motion Model Calibration

For odometry calibration, we will refer to the reduced state vectors containing 2D position and orientation as $x_0 = (x, y, y)$. Corresponding to Eq. (3.8), $u_0_t = (u_x_t, u_y_t, u_y_t)$ estimates the displacement between two poses reported by odometry $e_x_0_t = e_x_0_t + u_0_t$.

To calibrate the drift of $u_0_t$, we assume that a ground truth pose $x_0_0_t$ is available in a prior learning phase, e.g., from an external motion capture system, scan matching, or visual odometry. Based on the deviations from the ground truth, values of a calibration matrix $M_{2 \times 3}$ can be determined to correct the 2D drift of odometry, such that $x_0_t = x_0_0_t + Mu_0_t$.
Observation Model $p(o_t \mid x_t)$

$$p(o_t \mid x_t) = p(r_t, \tilde{z}_t, \tilde{\phi}_t, \tilde{\psi}_t \mid x_t) =$$

$$p(r_t \mid x_t) p(\tilde{z}_t \mid x_t) p(\tilde{\phi}_t \mid x_t) p(\tilde{\psi}_t \mid x_t)$$

- **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**
Questions

- Derive the recursive Bayes’ filter
- What are the three steps of the particle filter?
- What are the components of the state space?
- How is the odometry estimate from two consecutive torso poses computed?
- What are the components of the observation model?
- How are the individual terms computed?
Footstep Planning
Footstep Planning with A*

- Construct a search tree of successor states
- Fixed set of possible footsteps
- Check foot placements for collisions with obstacles during expansion

source: Kuffner et al.
Heuristics for Footstep Planning

- Heuristic highly influences the A* performance
- Estimate the costs to the goal scaled with the maximum forward step size
- Typical heuristic functions are based on:
  - Euclidean distance (straight line)
  - Shortest 2D path with safety margin around obstacles

In general, shortest 2D path heuristic is inadmissible for humanoids
Heuristics for Footstep Planning

- Shortest 2D path heuristics yields good results in practice
- Preprocessing required
  - “Erasing” of small, oversteppable objects from grid map in a preprocessing step
  - Compute heuristics using remaining objects
  - Consider the oversteppable objects in collision check during node expansion
Anytime Repairing A* (ARA*)

- Weighted A* (wA*): Heuristics “inflated” by a factor w
- ARA* runs a series of wA* searches, iteratively lowering w as long as a given time limit is not met
- The resulting paths are guaranteed to cost no more than w times the costs of the optimal path
- ARA* reuses information from previous searches

[Likhachev et al. ’04]
Speeding Up Footstep Planning (1)

Combine global, fast 2D path planning with footstep planning in a bounded local map.
Speeding Up Footstep Planning (2)

- Computing a path of 30 footsteps takes long running times (seconds)
- Footstep planning with **aborting A***, to achieve a fast, guaranteed replanning rate of ~50 Hz
- Use the shortest 2D path heuristic
Footstep Planning with A*

- Standard approach: Use a fixed set of actions
- How to choose the set of actions?
Footstep Planning with A*

Small set $\rightarrow$ fast planning
limited search space

Large set $\rightarrow$ large coverage
long planning time
Adaptive Node Expansion

- Add only a small set of nodes at each expansion step
- Systematically search for valid successors in the reachable area
- Leads to a high success rate, short paths, and fast planning times
Questions

- How does A* for footstep planning work?
- What are possible heuristics, what are their advantages/drawbacks?
- What is the idea of ARA*?
- How else can we speed up footstep planning?
- What is the idea of adaptive node expansion?
Bipedal Walking
Bipedal Walking

- Abstract kinematics (AK) of the leg
- Realize walking using motion pattern generation and AK
- Inverted pendulum model
- ZMP and CoM concepts
- Static and dynamic walking
- Inverted pendulum equation of motion and linearization
- ZMP preview control (basic concept)
- Capture step control (basic concept)
- Cart-pole model equation of motion (basic concept)
Questions

- Why is abstract kinematics simpler as IK?
- How does a motion pattern generator use abstract kinematics?
- Explain the inverse pendulum model (IPM) and its linearization
- What is the relation between the IPM and the ZMP?
- How does ZMP preview control work?
- What is the main idea of the capture step framework?
Inverse Kinematics and Whole-Body Motion Planning
while (e is too far from g) {
  Compute $J(e, q)$ for the current configuration $q$
  Compute $J^{-1}$
  $\Delta e = \alpha (g - e)$ // choose a step to take
  $\Delta q = J^{-1} \Delta e$ // compute required change in joints
  $q = q + \Delta q$ // apply change to joints
  Compute resulting $e$ // by FK
}
Questions

- How does the basic Jacobian IK technique work? Write down the algorithm.
- How is the Jacobian defined?
- What are the limitations of Jacobian control techniques?
RRTs: Tree Extension with a Small, Fixed Step Size

The algorithm terminates when $q_{new}$ is near the goal

[Kuffner&Lavalle, ICRA '00]
Questions

- How does the tree extension step in RRTs work?
- What is the idea of RRT-connect and how does the algorithm work?
- How does the RRT approach consider constraints?
- What are the properties of RRTs?
Reachability Map (RM)

- Constructed by systematic **sampling joint configurations** of a kinematic chain
- Apply **FK** to determine the **corresponding voxel** containing the end-effector pose
- Configurations are only added to the RM if they are statically-stable and self-collision free
- Result: Representation of reachability, **each voxel contains configurations** and a **corresponding manipulability measure**
Inverse Reachability Map (IRM)

- The IRM represents the set of potential stance poses relative to the EE frame
- Allows for selecting an optimal stance pose for a given grasping target
- Computed once offline
- Queried online

Cross section through the IRM showing potential feet locations

red=low
green=high
Determining the Optimal Stance Pose Given a Grasp Pose

- Given a desired 6D end-effector pose with transform $\mathcal{F}_{\text{grasp}}$
- How to determine the optimal stance pose?
Questions

- What is a singular configuration?
- How is the reachability map constructed and what does it contain?
- How can an IRM be obtained?
- How can the IRM be implemented?
- How can we select an optimal stance pose from the IRM given an end-effector pose?
Foresighted Robot Navigation in Human Environments
Key Idea to Speed-Up Planning

- Simplify planning by splitting the whole plan into several parts
- Exploit knowledge about different obstacle classes to choose appropriate robot actions
Our Approach - Overview

- **Segment objects** in the RGB image using a convolutional neural network (CNN)
- For each object class, **estimate the costs of possible actions** to overcome the obstacle based on execution time
- Encode the **action costs** corresponding to detected objects in a **2D cost grid** of the environment
- **Search for a 2D path** in this cost grid, which implicitly contains the necessary actions
Questions

- Why use obstacle class information during path planning?
- How are actions planned?
- Why is this an efficient approach for planning?
Our Approach - Overview

- Learn **transition probabilities** for **object interactions**
- Use a map of the environment with the **locations of relevant objects**
- Integrate observations about the user and **detected object interactions** to predict their **next navigation goal**
- Realize **foresighted robot navigation** by choosing poses close to the user’s goal while avoiding interferences
Questions

- How can the probabilities of possible navigations goals be computed?
- How can the robot determine the best position for itself?
- Which human comfort constraints do exist for robot navigation?
Todo

- Please complete the survey and leave your comments:
  https://limesurvey.informatik.uni-bonn.de/index.php/597431?lang=en

External students:

- Fill out the registration form for the exam on the webpage and give it to us!
- Send an email to one of the tutors