Humanoid Robotics

Least Squares

Maren Bennewitz
Goal of This Lecture

- Introduction to least squares error minimization
- Application to odometry calibration and later also to camera calibration and whole-body self-calibration of humanoids

- **Odometry:** Use data from motion sensors to estimate the change in position over time
- Robots typically execute motion commands inaccurately, *systematic errors* might occur, i.e., due to wear
Motion Drift

Odometry calibration with least squares helps to reduce such systematic errors.
Least Squares (LSQ)

- Approach for computing a solution of an overdetermined system

- Linear system $Ax = b$

- **Overdetermined system**: more independent equations than unknowns, i.e., no exact solution exists
Least Squares (LSQ)

- Approach for computing a solution of an overdetermined system

- Linear system $Ax = b$

- **Overdetermined system**: more independent equations than unknowns, i.e., **no exact solution exists**

- LSQ minimizes the **sum of the squared errors** in the equations
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 (unknown)
  - $z_i$ 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 measurements $z_{1:n}$
Problem Definition

\[ f_1(x) = \hat{z}_1 \quad z_1 \]
\[ f_2(x) = \hat{z}_2 \quad z_2 \]
\[ \ldots \]
\[ f_n(x) = \hat{z}_n \quad z_n \]

- state (unknown)
- predicted measurements
- real measurements
Example

\[ f_1(x) = \hat{z}_1 \quad z_1 \]
\[ f_2(x) = \hat{z}_2 \quad z_2 \]
\[ \ldots \]
\[ f_n(x) = \hat{z}_n \quad z_n \]

- **x**: position of a 3D feature in space
- \( \hat{z}_i \): coordinate of the 3D feature projected into camera image (prediction)
- Estimate the most likely 3D position of the feature based on the predicted image projections and the actual measurements \( z_i \)
Error Function

- Error $e_i$ 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: Find the Minimum

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

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

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

$$= \arg\min_x \sum_i e_i^T(x) \Omega_i e_i(x) \quad \text{error terms (vector)}$$
Goal: Find the Minimum

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

$$x^* = \arg\min_x \sum_i e_i^T(x)\Omega_i e_i(x)$$

- Possible solution: compute the first derivative of the global error function and find its zero

- Typically non-linear functions, no closed-form solution

- **Use a numerical approach**
Assumptions

- A “good” initial guess is available
- The error functions are “smooth” in the neighborhood of the (hopefully global) minimum

- Then: Solve the problem by iterative local linearizations
Solve via Iterative Local Linearizations (Gauss-Newton)

- Linearize the error terms around the current solution/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 (that is hopefully closer to the minimum)
- Iterate
Linearizing the Error Function

Approximate the error functions around an initial guess $x$ via **Taylor expansion**

$$e_i(x + \Delta x) \approx e_i(x) + J_i(x) \Delta x$$
Reminder: Jacobian Matrix (1)

- Given a vector-valued function

\[ g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix} \]

- The Jacobian matrix is defined as

\[ J_g(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial x_1} & \frac{\partial g_1(x)}{\partial x_2} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \frac{\partial g_2(x)}{\partial x_1} & \frac{\partial g_2(x)}{\partial x_2} & \cdots & \frac{\partial g_2(x)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m(x)}{\partial x_1} & \frac{\partial g_m(x)}{\partial x_2} & \cdots & \frac{\partial g_m(x)}{\partial x_n} \end{pmatrix} \]
Reminder: Jacobian Matrix (2)

- Generalizes the gradient of a scalar-valued function
- Orientation of the tangent plane wrt the vector-valued function at a given point
Linearizing the Error Function

Approximate the error functions around an initial guess $x$ via **Taylor expansion**

$$e_i(x + \Delta x) \approx e_i(x) + J_i(x) \Delta x$$
Squared Error

- With the previous linearization, we can fix $\mathbf{x}$ and carry out the minimization in the increments $\Delta \mathbf{x}$
- We use the Taylor expansion in the squared error terms:

$$e_i(x + \Delta x) = \ldots$$
Squared Error

- With the previous linearization, we can fix $X$ and carry out the minimization in the increments $\Delta x$
- We use the Taylor expansion in the squared error terms:

$$e_i(x + \Delta x) = e_i^T (x + \Delta x) \Omega_i e_i (x + \Delta x)$$
$$\simeq (e_i + J_i \Delta x)^T \Omega_i (e_i + J_i \Delta x)$$
$$= e_i^T \Omega_i e_i +$$
$$e_i^T \Omega_i J_i \Delta x + \Delta x^T J_i^T \Omega_i e_i +$$
$$\Delta x^T J_i^T \Omega_i J_i \Delta x$$
Squared Error (cont.)

- All summands are scalar so the transposition of a summand has no effect
- By grouping similar terms, we obtain:

$$e_i(x + \Delta x)$$

$$\simeq e_i^T \Omega_i e_i + e_i^T \Omega_i J_i \Delta x + \Delta x^T J_i^T \Omega_i e_i + \Delta x^T J_i^T \Omega_i J_i \Delta x$$

$$= \underbrace{e_i^T \Omega_i e_i}_{c_i} + 2 \underbrace{e_i^T \Omega_i J_i}_{b_i^T} \Delta x + \Delta x^T \underbrace{J_i^T \Omega_i J_i}_{H_i} \Delta x$$

$$= c_i + 2b_i^T \Delta x + \Delta x^T H_i \Delta x$$
Global Error

- Global error = sum of the squared error terms corresponding to the individual measurements
- Approximate the global error in the neighborhood of the current solution $\mathbf{x}$

$$F(\mathbf{x} + \Delta \mathbf{x}) \approx \sum_i \left(c_i + 2b_i^T \Delta \mathbf{x} + \Delta \mathbf{x}^T H_i \Delta \mathbf{x}\right)$$

$$= \sum_i c_i + 2(\sum_i b_i^T)\Delta \mathbf{x} + \Delta \mathbf{x}^T (\sum_i H_i) \Delta \mathbf{x}$$
Global Error (cont.)

\[ F(x + \Delta x) \approx \sum_i \left( c_i + 2b_i^T \Delta x + \Delta x^T H_i \Delta x \right) \]

\[ = \sum_i c_i + 2 \left( \sum_i b_i^T \right) \Delta x + \Delta x^T \left( \sum_i H_i \right) \Delta x \]

\[ = c + 2b^T \Delta x + \Delta x^T H \Delta x \]

with

\[ b^T = \sum_i e_i^T \Omega_i J_i \]

\[ H = \sum_i J_i^T \Omega_i J_i \]
Quadratic Form

- Thus, we can write the global error as a quadratic form in $\Delta x$

$$F(x + \Delta x) \simeq c + 2b^T \Delta x + \Delta x^T H \Delta x$$

- We need to compute the derivative of $F(x + \Delta x)$ wrt. $\Delta x$ (given $x$)
Deriving a Quadratic Form

- Given a quadratic form

\[ f(x) = x^T H x + b^T x + c \]

- The first derivative is

\[ \frac{\partial f}{\partial x} = H^T x + H x + b = (H + H^T) x + b \]
Quadratic Form

- Global error as quadratic form in $\Delta x$
  \[ F(x + \Delta x) \simeq c + 2b^T \Delta x + \Delta x^T H \Delta x \]

- The derivative of the approximated global error $F(x + \Delta x)$ wrt. $\Delta x$ is then:
  \[ \frac{\partial F(x + \Delta x)}{\partial \Delta x} \simeq 2b + 2H\Delta x \]

Note: $H$ is symmetric
Minimizing the Quadratic Form

- Derivative of $F(x + \Delta x)$
  \[
  \frac{\partial F(x + \Delta x)}{\partial \Delta x} \approx 2b + 2H\Delta x
  \]
- Setting it to zero leads to
  \[
  0 = 2b + 2H\Delta x
  \]
- Which leads to the linear system
  \[
  H\Delta x = -b
  \]
- The solution for the increment $\Delta x^*$ is
  \[
  \Delta x^* = -H^{-1}b
  \]
Gauss-Newton Solution

Iterate the following steps:

- Linearize around $x$ and compute for each measurement
  \[
  e_i(x + \Delta x) \approx e_i(x) + J_i(x)\Delta x
  \]

- Compute the terms for the linear system
  \[
  b^T = \sum_i e_i^T \Omega_i J_i \quad H = \sum_i J_i^T \Omega_i J_i
  \]

- Solve the linear system
  \[
  \Delta x^* = -H^{-1}b
  \]

- Update state
  \[
  x \leftarrow x + \Delta x^*
  \]
How to Efficiently Solve the Linear System?

- Linear system $H \Delta x = -b$
- Can be solved by matrix inversion (in theory)
- In practice:
  - Cholesky factorization
  - QR decomposition
  - Iterative methods such as conjugate gradients (for large systems)
Next Steps for You

- Register for the tutorial, see webpage for details
- Download the first exercise sheet from the webpage
- Join the tutorial next Tuesday at 8.30am via Zoom, link on webpage
Thanks for joining!

Now:
Job announcement and robot demo by Nils Dengler
Gauss-Newton Summary

Method to minimize a squared error:

- Start with an initial guess
- Linearize the individual error functions
- This leads to a quadratic form
- Setting its derivative to zero leads to a linear system
- Solving the linear systems leads to a state update
- Iterate
Example: Odometry Calibration

- Odometry measurements $\mathbf{u}_i = (u_{i,x}, u_{i,y}, u_{i,\theta})^T$
- Eliminate systematic errors through calibration
- Assumption: Ground truth $\mathbf{u}_i^*$ is available
- Ground truth by external motion capture, scan-matching, or a SLAM system
Example: Odometry Calibration

- We are looking for a function $f_i(x)$ that, given its parameters $x$, returns a corrected odometry:

$$u'_i = f_i(x) = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} u_i$$

- We need to find the parameters $x$
Odometry Calibration (cont.)

- The state vector is
  \[ \mathbf{x} = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} \end{pmatrix}^T \]

- The error function is
  \[ \mathbf{e}_i(\mathbf{x}) = \mathbf{u}_i^* - \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \mathbf{u}_i \]

- Accordingly, its Jacobian is
  \[ \mathbf{J}_i = \frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial \mathbf{x}} = - \begin{pmatrix} u_{i,x} & u_{i,y} & u_{i,\theta} \\ u_{i,x} & u_{i,y} & u_{i,\theta} \\ u_{i,x} & u_{i,y} & u_{i,\theta} \end{pmatrix} \]

Does not depend on \( \mathbf{x} \), why? What are the consequences?  \[ \mathbf{e} \text{ is linear, no need to iterate!} \]
Questions

- How do the parameters look like if the odometry is perfect?
- How many measurements are at least needed to find a solution for the calibration problem?
Reminder: Rotation Matrix

- 3D rotations along the main axes

\[
R_x(\theta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix} \quad R_y(\theta) = \begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix}
\]

\[
R_z(\theta) = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

- IMPORTANT: Rotations are not commutative!

\[
R_x(\frac{\pi}{4}) \cdot R_y(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \neq R_y(\frac{\pi}{4}) \cdot R_x(\frac{\pi}{4}) \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}
\]

- The inverse is the transpose (can be computed efficiently)
Matrices to Represent Affine Transformations

- Describe a 3D transformation via matrices

\[
A = \begin{pmatrix}
R & t \\
0 & 1
\end{pmatrix}
\]

- Such transformation matrices are used to describe transforms between poses in the world
Example: Chaining Transformations

- Matrix $A$ represents the pose of a robot in the world frame
- Matrix $B$ represents the position of a sensor on the robot in the robot frame
- The sensor perceives an object at a given location $p$, in its own frame
- Where is the object in the world frame?
Example: Chaining Transformations

- Matrix $A$ represents the pose of a **robot** in the **world** frame
- Matrix $B$ represents the position of a **sensor** on the robot in the **robot** frame
- The **sensor** perceives an **object** at a given location $p$, in its **own** frame
- Where is the object in the **world** frame?

$Bp$ gives the pose of the object wrt the robot
Example: Chaining Transformations

- Matrix $A$ represents the pose of a robot in the world frame.
- Matrix $B$ represents the position of a sensor on the robot in the robot frame.
- The sensor perceives an object at a given location $p$, in its own frame.
- Where is the object in the world frame?

$Bp$ gives the pose of the object wrt the robot.

$ABp$ gives the pose of the object wrt the world.
Summary

- Technique to minimize squared error functions
- Gauss-Newton is an iterative approach for solving non-linear problems
- Uses linearization (approximation!)
- Popular method in a lot of disciplines
- Transformation matrices describe transforms between poses
- Next chapters: Application of least squares to camera and whole-body self-calibration
Literature

Least Squares and Gauss-Newton

- Basically every textbook on numeric calculus or optimization
- Wikipedia (for a brief summary)
- Slides partially based on the course on Robot Mapping by Cyrill Stachniss