Robotic 3D Vision
Prof. Dr. Jörg Stückler, Rui Wang
Winter Semester 2017/2018

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 4

Topics: EKF-SLAM, Direct Visual SLAM, Pose Graph Optimization

Submission deadline: Wednesday, 10.01.2018, 23:59
Hand-in via email to rob3dvis-ws17@vision.in.tum.de

General Notice

All exercises can be done in teams of up to three students. Please hand-in your solution before the submission deadline, indicating names and matriculation numbers of your team members. Teams are encouraged to present their submitted solution during the exercise sessions.

Exercise 4.1: EKF-SLAM

In this exercise, you will implement an EKF-SLAM algorithm. We assume the robot moves in the 2D plane, for example, a wheeled robot with differential drive that moves on the floor inside a building. This means the robot state $\boldsymbol{\xi}_t = (x_t, y_t, \theta_t)^\top$ is 3-dimensional and composed of the 2-dimensional position x_t, y_t in the plane and the robot heading θ_t. We model the robot motion with an odometry-based motion model in this exercise, i.e. the state-transition model is

$$\boldsymbol{\xi}_t = g(\boldsymbol{\xi}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t := \boldsymbol{\xi}_{t-1} + \begin{pmatrix} u_{tr}\cos(\theta_{t-1} + u_{r_1}) \\ u_{tr}\sin(\theta_{t-1} + u_{r_1}) \\ u_{r_1} + u_{r_2} \end{pmatrix} + \boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t}). \quad (1)$$

The action $\mathbf{u}_t = (u_{tr}, u_{r_1}, u_{r_2})^\top$ is given by translational (u_{tr}) and rotational (u_{r_1}, u_{r_2}) motion measurements obtained from wheel odometry. For the noise covariance of the state-transitions, we assume

$$\boldsymbol{\Sigma}_{d_t} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{pmatrix} \quad (2)$$

The robot measures the range r and bearing ϕ to 2D landmark points $\mathbf{l}_j = (l_{j,x}, l_{j,y})$ in the environment in the horizontal plane. The full state vector $\mathbf{x}$ of the EKF is the vector stacked from robot state $\boldsymbol{\xi}$ and all landmark points $\mathbf{l}_j$. The robot measures multiple landmarks in a time step for which we assume the association $c_{t,i} = j$ of measurements $\mathbf{z}_{t,i} = (r_{t,i}, \phi_{t,i})^\top$ to landmarks j known. The observation model is

$$\mathbf{z}_{t,i} = h(\mathbf{x}_t, c_{t,i}) + \boldsymbol{\delta}_{t,i} := \begin{pmatrix} \left\| (x_t, y_t)^\top - (l_{j,x}, l_{j,y})^\top \right\|_2 \\ \operatorname{atan2}(l_{j,y} - y_t, l_{j,x} - x_t) - \theta_t \end{pmatrix} + \boldsymbol{\delta}_{t,i}, \boldsymbol{\delta}_{t,i} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t,i}). \quad (3)$$

For the observation noise of an individual landmark measurement, we assume

$$\mathbf{\Sigma}_{m_t,i} = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}. \tag{4}$$

The complete observation model in each time step, $\mathbf{z}_t = h(\mathbf{x_t}) + \boldsymbol{\delta}_t$ with $\boldsymbol{\delta}_t = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{m_t})$ stacks the M measurements in a single vector $z_t = (z_{t,0}^\top, \ldots, z_{t,M-1}^\top)^\top \in \mathbb{R}^M$. Analogously, we write $h(\mathbf{x}_t) := (h(\mathbf{x}_t, c_{t,0})^\top, \ldots, h(\mathbf{x}_t, c_{t,M-1})^\top)^\top$. The covariance $\mathbf{\Sigma}_{m_t}$ is formed from the individual measurement covariances,

$$\mathbf{\Sigma}_{m_t} = \begin{pmatrix} \mathbf{\Sigma}_{m_t,0} & 0 & \cdots & 0 \\ 0 & \mathbf{\Sigma}_{m_t,1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{\Sigma}_{m_t,M-1} \end{pmatrix}. \tag{5}$$

(a) Determine the analytic Jacobians of the state-transition function $g(\mathbf{x}_t, \mathbf{u}_t)$ and the observation function $h(\mathbf{x}_t)$ for the robot pose $\boldsymbol{\xi}$ and landmark positions $\mathbf{l}_j$.

(b) Obtain the code sample and data for this part of the exercise from the course webpage. The archive contains three folders: data, matlab, plots. Implement EKF prediction and correction to localize the robot and map the landmarks by finalizing the code in the files `prediction_step.m` and `correction_step.m`.

(c) Run your code and report the final robot pose and landmark position estimates after processing the whole dataset. Visualize your final result using the provided plotting functions. Plot the current state estimate in each time step into a png image and generate a video from the result sequence using avconv.

Exercise 4.2: Direct Visual SLAM, Pose Graph Optimization

In this exercise, you will implement a direct RGB-SLAM approach using direct image alignment and pose graph optimization.

(a) Extract the exercise archive to obtain the provided code. The archive contains code for direct image alignment as implemented in Exercise 3.2. Download the fr2_desk sequence from the TUM RGB-D benchmark from the following website: https://vision.in.tum.de/data/datasets/rgbd-dataset/download The file formats are described here: https://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats

Note: Convert the RGB images to floating point grayscale images before processing them. The depth images represent depth values by 16-bit integer values and need to be scaled by a factor of 1/5000 to obtain metric depth. Convert the depth images to floating point metric values before further processing them.

Use the provided `downscale` function to downsample the images 4 times with sampling factor 2. Display the images in original resolution and their downsampled versions.

(b) Test the solution code for Exercise 3.2 and verify that it works with the provided data from Ex. 3.2. The target result is specified in Ex. 3.2.

(c) Implement keyframe-based camera tracking through direct image alignment. You can process the RGB-D images only up to a downsampled size in the image pyramid to make the image alignment faster. Choose thresholds on the rotational and translational distance to create new keyframes, when the image overlap gets too small for camera tracking. Process the complete sequence and evaluate the relative pose error (RPE) using the provided evaluation tool in the TUM RGB-D benchmark (https://vision.in.tum.de/data/datasets/rgbd-dataset/tools). Plot the resulting trajectory using the absolute trajectory error (ATE) tool.

(d) Implement pose-graph optimization for the keyframe poses using the Gauss-Newton algorithm and left-multiplied increments on the poses. Create relative pose constraints between keyframes that are in successive order through direct image alignment. At the end of the trajectory, detect a loop closure manually between a pair of keyframes and find their relative pose constraint through direct image alignment. Evaluate the relative pose error (RPE) and absolute trajectory error (ATE) using the provided evaluation tool in the TUM RGB-D benchmark (https://vision.in.tum.de/data/datasets/rgbd-dataset/tools). Plot the resulting trajectory using the absolute trajectory error (ATE) tool. Compare your result with the direct visual odometry result from the previous exercise part.

Submission instructions

A complete submission consists both of a PDF file with the solutions/answers to the questions on the exercise sheet and a ZIP file containing the source code that you used to solve the given problems. Note all names and matriculation numbers of your team members in the PDF file. Make sure that your ZIP file contains all files necessary to compile and run your code, but it should not contain any build files or binaries. Please submit your solution via email to `rob3dvis-ws17@vision.in.tum.de`.