Albert-Ludwigs-Universität Freiburg, Institut für Informatik
PD Dr. Cyrill Stachniss
Lecture: Robot Mapping
Winter term 2013

Sheet 9

Topic: Least-Squares

Submission deadline: January, 27

Submit to: `robotmappingtutors@informatik.uni-freiburg.de`

Exercise: Odometry Calibration

Implement an odometry calibration tool based on a least-squares method as presented in the lecture. To support this task, we provide a small *Octave* framework (see course website). The framework contains the following folders:

data contains the recorded raw odometry and the motion estimated by a scan-matcher for each time step.

octave contains the Octave framework with stubs to complete.

plots this folder is used to store images.

The below mentioned tasks should be implemented inside the framework in the directory `octave` by completing the stubs:

- Implement the functions in `ls_calibrate_odometry.m` for constructing and solving the least-squares system.
- Implement the function in `apply_odometry_correction.m` for applying the calibration matrix to a set of odometry measurements.
- Implement the function in `compute_trajectory.m` for chaining up the affine transformation matrices of the relative odometry measurements.

After implementing the missing parts, you can run the framework. To do that, change into the directory octave and launch *Octave*. To start the main loop, type `LSCalibrateOdometry`. The script will produce a plot showing the trajectory of the raw odometry measurements, the estimate obtained by scan-matching, and the odometry after applying the calibration. This plot will be saved in the `plots` directory. Figure 1 shows the result that you should obtain.

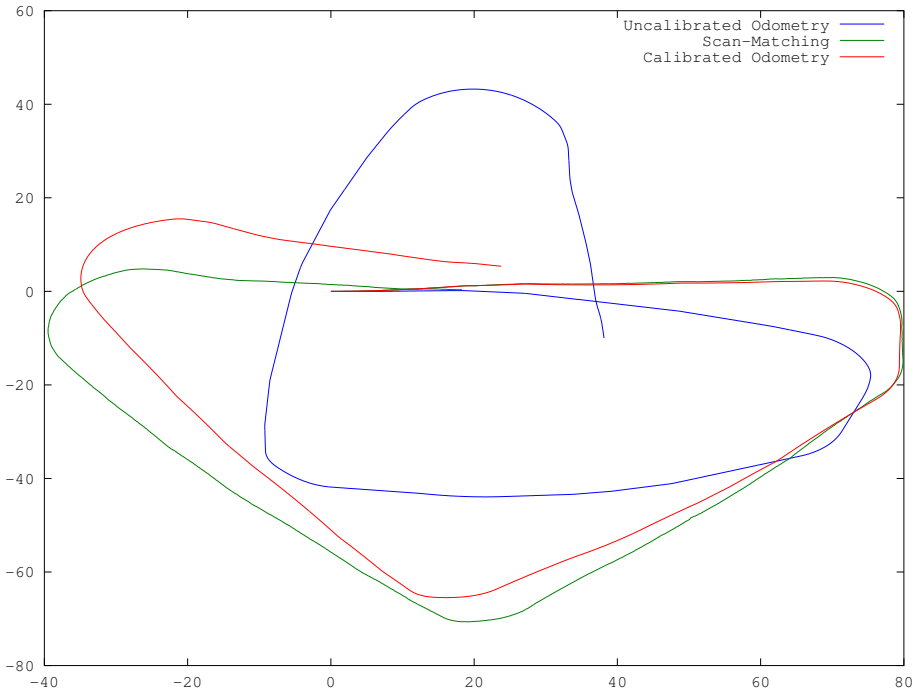

Figure 1: Visualization of the uncalibrated odometry, scan-matching, and the odometry after calibration.

Some implementation tips:

- The functions `v2t` and `t2v` are available within the framework and allow to convert between a vector representing the pose of a robot and its corresponding affine transformation matrix.
- The function `reshape` returns a matrix with specified dimensions whose elements are taken from another matrix. It can, for example, convert a vector into a matrix.
- Many of the functions in *Octave* can handle matrices and compute values along the rows or columns of a matrix. Some useful functions that support this are `sum`, `log`, `sqrt`, `sin`, `cos`, and many others.