Albert-Ludwigs-Universität Freiburg, Institut für Informatik
PD Dr. Cyrill Stachniss
Lecture: Robot Mapping
Winter term 2012

Sheet 10

Topic: Graph-Based SLAM

Submission deadline: February, 11
Submit to: `robotmappingtutors@informatik.uni-freiburg.de`

Exercise: Max-Mixture Least Squares SLAM

Implement the max-mixture approximation for addressing multi-modal constraints in the context of least-squares, graph-based SLAM. To support this task, we provide a small *Octave* framework (see course website). The framework contains the following folders:

data contains several datasets, each gives the measurements of one SLAM problem

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 function in `compute_best_mixture_component.m` for selecting the most likely mode of a constraint based on the max-mixture formulation.
- Implement the function in `compute_global_error.m` for computing the total squared error of a graph.
- Implement the function in `linearize_and_solve.m` for constructing and solving the linear approximation.

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 `maxmixlsSlam`. The script will produce a plot showing the positions of the robot in each iteration. These plots will be saved in the `plots` directory.

The file ⟨name of the dataset⟩.png depicts the result that you should obtain after convergence for each dataset. Additionally, the initial and the final error ($\sum_i e_i^T \Omega_i e_i$) for each dataset should be approximately:

dataset	initial error	final error
manhattan250.dat	111	6
manhattan500.dat	322	16
manhattan1000a.dat	3932	32
manhattan1000b.dat	3932	32

Use the following criterion when computing the most likely mixture component, k^*, of a constraint:

$$k^* = \underset{k}{\operatorname{argmin}}\left(\frac{1}{2} e_{ij_k}^T \Omega_{ij_k} e_{ij_k} - \log(w_k) + \frac{1}{2}\log(|\Sigma_{ij_k}|)\right)$$

Some implementation tips:

- You can use the `compute_error_pose_pose_constraint.m` function available to you to compute the error vector of a pose-pose constraint.
- You can use the `linearize_pose_pose_constraint.m` function available to you to compute the Jacobian of a pose-pose constraint.
- 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 `max`, `abs`, `sum`, `log`, `sqrt`, `sin`, `cos`, and many others.