MRPT
Mobile Robot Programming Toolkit
Version: 2.5.8

graphslam-engine: execute graphSLAM using rawlog files in MRPT

Nikos Koukis
nickkouk@gmail.com
National Technical University of Athens

Document build: Saturday 7th January, 2023

Contents

List of Figures

Abstract

Current report delves into the new capabilities of mrpt-graphslam library and provides a complete guide for using the `graphslam-engine` application. More specifically it deals with the following:

- Description of major mrpt-graphslam library structure and concepts (e.g. CGraphSlamEngine, decider, optimizers classes)
- `graphslam-engine` usage information (e.g. command line arguments)

The extension of mrpt-graphslam library as well as the coding of the corresponding graphslam-engine application was implemented as part of the *Google Summer of Code 2016 program.*

1   Introduction to graphSLAM

An autonomous robot needs to address two critical problems to survive and navigate within its surroundings: mapping the environment and finding its relative location within the map. Simultaneous localization and mapping (SLAM) is a process that aims to localize an autonomous mobile robot in a previously unexplored environment while constructing a consistent and incremental map of its environment [1]. While filtering methods (extended Kalman filtering, information-form filtering, particle filtering) used to dominate the SLAM literature, recently (2006) graph-based approaches have made a comeback. Introduced by Lu and Milios in 1997 [2] graph-based approaches formulate SLAM as a least-squares minimization problem.

2   Application Usage

Aim of the app is to perform 2D graphSLAM: Robot localizes itself in the environment while, at the same time builds a map of that environment. App currently executes SLAM using MRPT rawlog files (both MRPT rawlog formats are supported) as input which should contain (some of) the following observation types:

- CObservationOdometry
- CObservation2DRangeScan
- CObservation3DRangeScan
  Working with 3DRangeScans is currently in an experimental phase.

The majority of the graphslam-engine parameters in each case should be specified in an external .ini file which is to be given as a command-line argument. The following parameters can also be specified as command-line arguments:

.ini-file [REQUIRED]
> Specify the .ini configuration file using the `-i`, `--ini-file` flags. Configuration file parameters are read by the main CGraphSlamEngine class as well as the node/edge registration schemes, and the optimization scheme.

rawlog-file [REQUIRED]
> Specify the rawlog dataset file using the `-r`, `--rawlog` flags.

ground-truth
> Specify a ground truth file with `-g`, `--ground-truth` flags. Ground truth has to be specified if user has set visualize_slam_metric or visualize_ground_truth to true in the .ini file, otherwise an exception is raised.

node/edge registration deciders
> Specify the node registration or/and edge registration decider classes to be used using `-n`, `--node-reg`, `-e`, `--edge-reg` flags. If not specified the default CFixedIntervalsNRD, CICPCriteriaERD are used as node and edge registration decider schemes respectively.

optimizer class to be used
> Specify the class to be used for the optimization of the pose-graph using the `-o`, `--optimizer` flags. Currently the only supported optimization scheme is Levenberg-Marquardt non-linear graph optimization defined in optimize_graph_spa_levmarq. If not specified, the default CLevMarqGSO is used.

Furthermore, application offers the following capabilities to the user:

- Support of both kinds of MRPT rawlog formats, action-observations and observation-only format. However, users should also make sure that the deciders/optimizer classes that are to be used also support the rawlog format that they are interested in.
- Visual inspection of graphSLAM execution. Apart from others the visualization window can be configured, by the user, to include the following:
    - graphSLAM estimated robot trajectory
    - Odometry-only trajectory

        – ground-truth trajectory (if the corresponding ground-truth is available
        – Current robot 2D laser scan footprint
        – Draft version of the map
        – Currently constructed graph as edges between the registered nodes
- Hotkeys support for triggering various visualization features on/off. [1]

2.1   command-line arguments

The available command line argument are also listed below, along with a description of their usage:

```
graphslam-engine [--disable-visuals] [--list-optimizers]
                 [--list-regs] [--list-edge-regs]
                 [--list-node-regs][-o <CLevMarqGSO>][-e
                 <CICPCriteriaERD>][-n <CICPCriteriaNRD>][-g
                 <contents.rawlog.GT.txt>][-r <contents.rawlog>] [-i
                 <config.ini>] [--][--version] [-h]


--disable-visuals
  Disable Visualization - Overrides related visualize* directives of the
  .ini file

--list-optimizers
  List (all) available graphslam optimizer classes

--list-regs
  List (all) available registration decider classes

--list-edge-regs
  List available edge registration decider classes

--list-node-regs
  List available node registration decider classes

-o <CLevMarqGSO>,  --optimizer <CLevMarqGSO>
  Specify GraphSlam Optimizer

-e <CICPCriteriaERD>,  --edge-reg <CICPCriteriaERD>
  Specify Edge registration decider

-n <CICPCriteriaNRD>,  --node-reg <CICPCriteriaNRD>
  Specify Node registration decider

-g <contents.rawlog.GT.txt>,  --ground-truth <contents.rawlog.GT.txt>
  Ground-truth textfile

-r <contents.rawlog>,  --rawlog <contents.rawlog>
  Rawlog dataset file

-i <config.ini>,  --ini_file <config.ini>
  .ini configuration file
```

---

[1]Any registration decider / optimizer can get notified of keyboard/mouse events and can modify the visuals when these occur. For more on this see the corresponding method

```
--, --ignore_rest
  Ignores the rest of the labeled arguments following this flag.

--version
  Displays version information and exits.

-h, --help
  Displays usage information and exits.
```

2.2   Application output

By default graphslam-engine execution generates an output directory by the name *graphslam_results* within the current working directory. The output directory contains the following files:

CGraphSlamEngine.log
  Logfile containing the activity of the CGraphSlamEngine class instance. Activity involves output logger messages, time statistics for critical parts of the application, and summary statistics about the constructed graph (e.g. number of registered nodes, edges).

node_registrar.log, edge_registrar.log, optimizer.log
  Logfiles containing the activity of the corresponding class instances. File contents depend on the implementation of the corresponding classes but in most cases they contain output logger messages, time statistics for critical parts of the class execution.

output_graph.graph
  File contains the constructed graph in the VERTEX/EDGE format. The latter can be visualized using the MRPT graph-slam application for verification reasons. For more information on this, consult the following pages:
  - `http://www.mrpt.org/Graph-SLAM_maps`
  - `http://www.mrpt.org/list-of-mrpt-apps/application-graph-slam/`

output_scene.3DScene
  File contains the 3DScene that was generated in the end of the graphslam-engine execution. The latter can be visualized using the MRPT SceneViewer3D tool. For more information on this, see: `http://www.mrpt.org/list-of-mrpt-apps/application-sceneviewer3d/`

  Note: File is generated only when the visualization of the graph construction is enabled in the .ini configuration file. See .ini parameters as well as the `--disable` flag for more on this.

SLAM_evaluation_metric.log
  File contains the differences between the estimated trajectory increments and the corresponding ground-truth increments and can be used to verify and evaluate the performance of the SLAM algorithm. For more information on the metric, see [3]
  Note: File is generated only when the ground-truth of the corresponding dataset is given.

Note: In case a directory named graphslam_results, generated during a previous execution, already exists, it is, by default, overwritten. If this is not the desired behavior, user can set the user_decides_about_output_dir .ini parameter to true so that they are asked about this naming conflict during program execution.

3   Library Design

In this section insight into the graphslam-engine application - and corresponding library - is provided.

CGraphSlamEngine is the main class executing graphslam. CGraphSlamEngine delegates most of the graph manipulation tasks to node/edge registration deciders and optimizer classes. This makes up for independence between the different tasks as well as for a reconfigurable setup, as the user can select

to use different decider/optimizer classes depending on the situation. Users can also write their own decider/optimizer classes by inheriting from one of the CNodeRegistrationDecider, CEdgeRegistrationDecider, CGraphSlamOptimizer interfaces depending on what part they want to implement.

3.1   Registration Deciders

The registration decider classes are divided into node and edge registration deciders. The former are responsible of adding new nodes in the graph while the latter add additional edges between already registered graph nodes. These nodes can be consecutive or non-consecutive.

3.1.1   Node Registration Deciders (NRD)

Node registration decider schemes add nodes to the graph according to a specific criterion. Node deciders should implement the methods defined in the CNodeRegistrationDecider abstract class. The latter provides the basic methods that have to exist in every node registration decider class and which are called from the main CGraphSlamEngine instance. For an example of inheriting from this class see CFixedIntervalsNRD.

Currently two specific node registration schemes have been implemented:

- CFixedIntervalsNRD
  Decider registers a new node in the graph if the distance or the angle difference with regards to the previous registered node surpasses a corresponding fixed threshold. Decider makes use only of the CObservationOdometry instances in the rawlog file.
- CICPCriteriaNRD
  Decider registers a new node in the graph if the distance or the angle difference with regards to the previous registered node surpasses a corresponding fixed threshold. Decider measures the distance from the current position to the previous registered node using ICP (i.e. matches the current range scan against the range scan of the previous node). In case of noisy 2D laser scans, decider can also use odometry information to locally correct and smoothen the robot trajectory. Decider makes use of 2DRangeScans or 3DRangeScans.

Note: As a naming convention, all the implemented node registration deciders are suffixed with the NRD acronym. For more information on this see the

node registration deciders interface

3.1.2   Edge Registration Deciders (ERD)

Edge registration decider schemes add edges between already added nodes in the graph according to a specific criterion. Edge deciders should implement the methods defined in CEdgeRegistrationDecider abstract class. CEdgeRegistrationDecider provides the basic methods that have to exist in every edge registration decider class. For an example of inheriting from this class see CICPCriteriaERD.

Currently two specific edge registration schemes have been implemented:

- CICPCriteriaERD
  Register new edges in the graph with the last inserted node. Criterion for adding new edges should be the goodness of the candidate ICP edge. The nodes for ICP are picked based on the distance from the last inserted node. Decider makes use of 2DRangeScans or 3DRangeScans.
- CLoopCloserERD
  Evaluate sets of potential loop closure edges in the graph based on their pairwise consistency matrix.Decider first splits the graph into partitions based on the 2D laser scans of the nodes and then searches for potential loop closure edges within the partitions. Goal is to register only a subset of the potential loop closure edges that maximally agree with each other. Decider is implemented based on [4], [5]

Note: As a naming convention, all the implemented edge registration deciders are suffixed with the ERD acronym.

For more information on this see the edge registration deciders interface

3.2   graphSLAM Optimizers (GSO)

Optimizer classes optimize an already constructed graph so that the registered edges maximally agree with each other. Optimizer schemes should implement the methods defined in CGraphSlamOptimizer abstract class. For an example of inheriting from this class see CLevMarqGSO.

Note: As a naming convention, all the implemented optimizer classes are suffixed with the GSO acronym.

For more information on this see the graphSLAM optimizers interface

3.3   Notable links

Below important links for the library/application are provided:

- graphslam-engine - Application page
- Demonstration video
- CGraphSlamEngine documentation
- Registration Deciders / Optimizers documentation
- GSoC 2016 graphslam-engine discussion
- Directory of .ini sample files: `../../share/mrpt/config_files/graphslam-engine/`
- Directory of demo datasets: `../../share/mrpt/datasets/graphslam-engine-demos/`

4   Further Improvements

As of now, the mrpt-graphslam library is still under active development. A list of features that have been scheduled for implementation are listed below:

- Cleanup and Fix code bugs
- Integrate with ROS — Add support for online execution
- Add support for multi-robot graphSLAM
- Add nodes in the graph in an adaptive fashion
- Implement node reduction scheme
- Add support for visual sensors (e.g. working with point features as SIFT)
- Integrate with 3rd party optimization libraries

For more details on the progress of the aforementioned improvements, see the corresponding github milestone

References

[1] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review," *Journal of Field Robotics*, 2016.

[2] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 249–275, 1994.

[3] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós, "A Comparison of SLAM Algorithms Based on a Graph of Relations," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2089–2095, 2009.

[4] J. L. Blanco, J. Gonzalez, and J. A. Fernandez-Madrigal, "Consistent observation grouping for generating metric-topological maps that improves robot localization," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. May, pp. 818–823, 2006.

[5] E. Olson, "Recognizing places using spectrally clustered local matches," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157–1172, 2009.

[6] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[7] J. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization," *University of Malaga, Tech. Rep*, no. 3, 2010.