NA 568 Mobile Robotics: Methods & Algorithms Winter 2022 – Homework 7 – SLAM

Maani Ghaffari
University of Michigan

March 19, 2022

This is a reminder that no late HW is accepted. We drop your lowest grade from HW 1-6. It is perfectly fine to drop a zero as a HW grade. We are using Gradescope for turning in HW; see relevant information on the course Canvas site.

This problem set counts for about 7% of your course grade. You are encouraged to talk at the conceptual level with other students, but you must complete all work individually and may not share any non-trivial code or solution steps. See the syllabus for the full collaboration policy.

Submission Instructions

Your assignment must be received by 11:55 pm on Friday, April 1st (Anywhere on Earth Time). This corresponds to 7:55 AM on April 2nd in Eastern Time. This is selected out of fairness to all our students, including those who take the course remotely. You are to upload your assignment directly to the Gradescope website as two attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your uniqname with the structure shown below.

   ```
   alincoln_hw7.zip:
   alincoln_hw7/
   alincoln_hw7/YOUR_WORKING_CODE
   ```
   You may use C++, Python, or MATLAB for your programming.

2. A PDF with the written portion of your write-up. Scanned versions of hand-written documents, converted to PDFs, are perfectly acceptable. No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `alincoln_hw7.pdf`.

3. Submit your complete source code. We will compile and run your code for evaluation.

Pose Graph SLAM using GTSAM Library

In this homework, you're going to solve the pose graph SLAM problem using the GTSAM library. If you are not familiar with GTSAM, a detailed tutorial is on their website: https://gtsam.org/tutorials/intro.html. To install GTSAM in c++, you'll have to clone the code from the repository: https://github.com/borglab/gtsam, checkout (using git) the latest version and build the library following the instruction.

After you successfully install GTSAM, write a function to read G2O [1] files and solve the graph optimization problem for both 2D and 3D cases using GTSAM. In this assignment, we use datasets provided at https://lucacarlone.mit.edu/datasets/.

While GTSAM is developed using C++, it also provides both MATLAB and Python wrapper. In this assignment, you're free to use any of those languages.

GTSAM Installation Guide

Below we provide an installation guide for GTSAM libraries, then we introduce C++, MATLAB, and python versions of them respectively.

GTSAM library

The first step is to clone and install GTSAM library. Detailed instruction can be found in the repository.

Remark 1. *If you are planning to use MATLAB or python wrapper, you may skip this part.*

Remark 2. *GTSAM requires the Eigen library. It can be downloaded and installed from here:*
http://eigen.tuxfamily.org/index.php?title=Main_Page

Remark 3. *Please notice prerequisites: Boost >= 1.58 (Ubuntu: sudo apt-get install libboost-all-dev); CMake >= 3.0 (Ubuntu: sudo apt-get install cmake); A modern compiler, i.e., at least gcc 4.7.3 on Linux.*

- Go to this repository and click the green *clone or download* button on the right hand side to clone the repository to your machine.

- In the terminal, enter command (this will allow you to navigate to the repository)
  ```
  cd <path_to_your_repository>
  ```

- Then create a new folder named build by
  ```
  mkdir build
  ```

- Navigate into the build folder
  ```
  cd build
  ```

- run cmake to create essential links for building files (**Note:** if you want to use the python wrapper, you will have to do something different here. Please jump to the python wrapper section)
  ```
  cmake ..
  ```

  Remark 4. *If you have error saying could not find Boost, you may try manually install boost following this tutorial and make sure you are pointing to the correct directory by adding following commands in CMakeList.txt :*

---

[1] https://github.com/RainerKuemmerle/g2o/wiki/File-Format

```
SET (BOOST_ROOT "<your_boost_path>")
SET (BOOST_INCLUDEDIR "<your_boost_path>/boost")
SET (BOOST_LIBRARYDIR "<your_boost_path>/libs")
SET (BOOST_MIN_VERSION "1.58.0")
set (Boost_NO_BOOST_CMAKE ON)
```

- make the file to the build folder (The command -j10 indicates the number of threads to be used during make. It'll make the compilation faster. I recommend using your max number of cores - 2, otherwise your machine may get stuck. I have 12 threads on my machine, so I use -j10.)
  `make -j10`

- install the gtsam library to your machine.
  `sudo make install -j10`

- At this step, you will be able to see the path where all the packages are installing to. If later you somehow cannot link the install path to your code, you can come back to this step and see where it is installed to.

C++

If you want to use C++, you don't have to install anything else. All you need to do is to find and link GTSAM library by adding the below context to your `CMakeList.txt` file.

```
find_package(GTSAM REQUIRED)
include_directories(${GTSAM_INCLUDE_DIR})
target_link_libraries(<your_project> <your_project_lib> gtsam)
```

You may refer to some c++ gtsam examples here.

MATLAB Wrapper

If you have a newer Ubuntu system (later than 10.04), you must make a small modification to your MATLAB installation, due to MATLAB being distributed with an old version of the C++ standard library. Delete or rename all files starting with libstdc++ in your MATLAB installation directory, in paths:

- `/usr/local/MATLAB/[version]/sys/os/[system]/`

- `/usr/local/MATLAB/[version]/bin/[system]/`

For MATLAB wrapper, you can do it in two ways:

1. download the precompiled wrapper from here,

2. or install from sources following the instructions in here.

We recommend using the pre-compiled wrapper if you are not familiar with linux systems. Below are instructions on how to add the pre-compiled MATLAB wrapper to your toolbox.

- Download the pre-compiled wrapper here. Download precompiled MATLAB toolbox (Works with MATLAB R2011a and later) depends on your system OS (Mac OS 64-bit / Linux 64-bit / Windows 64-bit).

- Extract the folder. (**Note: for linux system**, after extracting, you'll have to right click on the extracted file → Open With → Archive Mounter. Your will then see a `gtsam-toolbox-3.2.0-lin64` on your left side along with Computer, OS... The folder inside named `gtsam_toolbox` is the folder we want to use.)

- Put the `gtsam_toolbox` folder under `<YOUR_MATLAB_INSTALL_PATH>/toolbox/`

- Every time you open your MATLAB, at the left side of your GUI, navigate to your `gtsam_toolbox` → right click → add to path → selected folder. Then you should be able to run the example codes under the example folder. Or you can add the following commends as your first part of code.
  `addpath('<your_gtsam_toolbox_path>')`
  `import gtsam.*`

You may refer to some matlab gtsam examples in `gtsam_toolbox/gtsam_examples` folder.

Python Wrapper

For Python wrapper, you can do it in two ways:

1. Using pip to install:

   - `pip install gtsam`

2. Build from source. Follow the instructions here to install. Below are some guides on how to install the Python wrapper.

   - After you successfully clone the repository and create the build folder, you'll have to first go into the cython folder and install the required dependencies.
     `pip install -r <gtsam_folder>/python/requirements.txt`
   - Then you'll have to do cmake differently by specifying the the Python version you want to use and enable Python wrapper:
     `cmake ..  -DGTSAM_BUILD_PYTHON=1 -DGTSAM_PYTHON_VERSION=<your_python_version>`
   - Compile the files in build (-j means using multi-thread. 10 is the number of threads you want to use)
     `make -j10`
   - Install it to your machine
     `sudo make python-install`

You may refer to some python gtsam examples here.

1   2D Graph SLAM (50 points)

A. (10 pts) Write a function to read 2D `Intel dataset` [2] from G2O format and output poses and edges. These poses and edges are used in later problems. It can be any form you like as long as you can use it to generate the correct result.

For 2D data, the pose in G2O format is [`VERTEX_SE2 i x y theta`] and the edge in G2O format is [`EDGE_SE2 i j x y theta info(x, y, theta)`], where `info(x, y, theta)` is a 1×6 vector

---

[2] https://www.dropbox.com/s/vcz8cag7bo0zlaj/input_INTEL_g2o.g2o?dl=0

$[q_{11}\ q_{12}\ q_{13}\ q_{22}\ q_{23}\ q_{33}]$ where the elements are the upper-triangle matrix of the 3×3 information matrix $\Omega = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix}$. By inverting this information matrix, you can obtain the covariance matrix for the noise model.

You may look into detail in the g2o repository [3].

Remark 5. *If you use `readG2o()` and `load2D()` functions provided by GTSAM, then you will not be able to solve the graph incrementally (task C in this homework).*

Hint: You may use `fscanf()`, `textscan()`, or `readcell()` functions with proper formatSpec from MATLAB and check if the first element in the input is `VERTEX_SE2` or `EDGE_SE2`.

B. (20 pts) **Batch Solution:** A batch solution means when we construct the entire graph and solve it at the end altogether. Load `data/input_INTEL_g2o.g2o` and construct a 2D nonlinear factor graph using GTSAM. Use the Gauss-Newton solver. Visualize and compare the optimized trajectory against the initial trajectory. Include the plot in your pdf. Describe the graph construction process and its parameters.

Remark 6. *For this problem, Gauss Newton solver will fall into a local minimum if we don't give any perturbation. It is okay to submit a plot that doesn't work as expected, but please include discussions about why is this happening.*

Hint: You may use `NonlinearFactorGraph` as your graph, use `GaussNewtonOptimizer` as you optimizer, use `Values` for your initial estimation, `noiseModel.Gaussian.Covariance()` for your noise model, `graph.add()` and `initial.insert()` functions as you see fit. However, function names might be different for different versions of gtsam.

C. (20 pts) **Incremental Solution:** Use ISAM2 solver to optimize the trajectory incrementally (as you build the graph gradually). A detailed algorithms is described in Algorithm 1. Visualize and compare the optimized trajectory against the initial trajectory. Include the plot in your pdf. Describe the graph construction process and its parameters.

---

[3]https://github.com/RainerKuemmerle/g2o/wiki/File-Format-SLAM-2D

Algorithm 1 `incremental_solution_2d`(poses, edges)

Require: poses: a $N \times 4$ array that each row is $pose = (id_p, x, y, \theta)$; edges: a $M \times 11$ array that each row
is $edge = (id_{e1}, id_{e2}, dx, dy, d\theta, info)$

1: $isam \leftarrow$ gtsam.ISAM2()   ▷ Initialize isam solver
2: **for** every pose in poses **do**
3:      $graph \leftarrow$ NonlinearFactorGraph   ▷ Initialize the factor graph
4:      $initialEstimate \leftarrow$ Values   ▷ Initialize the initial estimation
5:      $(id_p, x, y, \theta) \leftarrow pose$ ▷ Extract information from the current pose
6:      **if** $id_p == 0$ **then**   ▷ The first pose
7:          $priorNoise \leftarrow$ some noiseModel   ▷ Use a predefined noise model
8:          $graph$.add(PriorFactorPose2(0, Pose2(x, y, θ), $priorNoise$))
9:          $initialEstimate$.insert(id_p, Pose2(x, y, θ))
10:     **else**   ▷ Not the first pose
11:         $prevPose \leftarrow result$.at($id_p - 1$)   ▷ Use last optimized pose
12:         $initialEstimate$.insert(id_p, $prevPose$)
13:         **for** every edge in edges **do**
14:             $(id_{e1}, id_{e2}, dx, dy, d\theta, info) \leftarrow edge$ ▷ Extract information from the current edge
15:             **if** $id_{e2} == id_p$ **then**
16:                 $cov = construct\_covariance(info)$ ▷ Construct a covariance matrix from the
    information vector.
17:                 $Model \leftarrow$ noiseModel.Gaussian.Covariance(cov)
18:                 $graph$.add(BetweenFactorPose2(id_{e1}, id_{e2}, Pose2($dx, dy, d\theta$), $Model$))
19:             **end if**
20:         **end for**
21:     **end if**
22:     $isam$.update($graph, initialEstimate$)
23:     $result = isam$.calculateEstimate()
24: **end for**

Hint: You may use `NonlinearFactorGraph` as your graph, use `gtsam.ISAM2()` as your update algo-
rithm, use `Values` for your initial estimation, and use `graph.add()`, `initial.insert()`, `isam.update()`,
and `isam.calculateEstimate()` functions as you see fit. However, function names might be different
for different versions of gtsam.
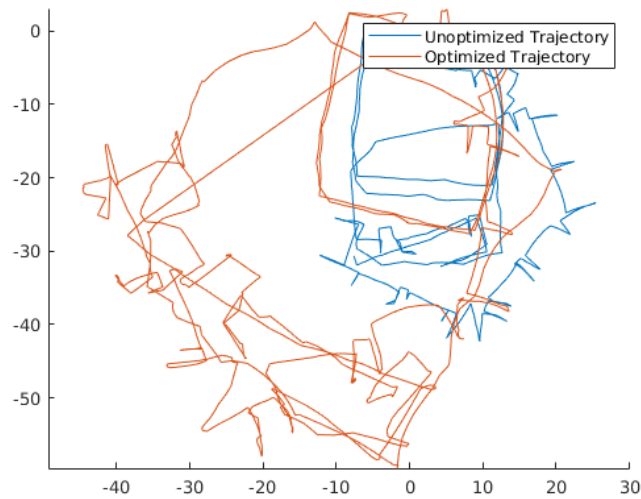
2   3D Graph SLAM (50 points)

A. (10 pts) Write a function to read 3D Garage G2O file [4] from G2O format and output poses and edges.

For 3D data, the pose in G2O format is [`VERTEX_SE3:QUAT i x y z qx qy qz qw`] where (x, y, z)
represents the translation and (qx, qy, qz, qw) the rotation as a quaternion. The edge in G2O format
is [`EDGE_SE3:QUAT i j x y z qx qy qz qw info(x, y, z, qx, qy, qz)`], where `info(x, y,
z, qx, qy, qz)` is a 1×21 vector of the 6×6 information matrix. After similar process in task 1 A,
you can obtain the covariance matrix. You may look into detail in the g2o repository [5].

---

[4]https://www.dropbox.com/s/zu23p8d522qccor/parking-garage.g2o?dl=0
[5]https://github.com/RainerKuemmerle/g2o/wiki/File-format-slam-3d

Figure 1: Expected result for task 1 B.

Remark 7. *Please notice that the quaternion in MATLAB is in the order of* `[qw qx qy qz]` *and is different from the order in g2o files which is* `[qx qy qz qw]`. *You may use* `quat2rotm()` *function in matlab to construct a rotation matrix from quaternion or use* `quat2tform()` *function in matlab to construct a transformation matrix.*

B. (20 pts) **Batch Solution:** Load `data/parking-garage.g2o` and construct a 3D nonlinear factor graph using GTSAM. Use the Gauss-Newton solver. Visualize and compare the optimized trajectory against the initial trajectory. Include a 3D plot or two 2D plots in your pdf. Describe the graph construction process and its parameters.

C. (20 pts) **Incremental Solution:** Use ISAM2 solver to optimize the trajectory incrementally. Visualize and compare the optimized trajectory against the initial trajectory. Include a 3D plot or two 2D plots in your pdf. Describe the graph construction process and its parameters.
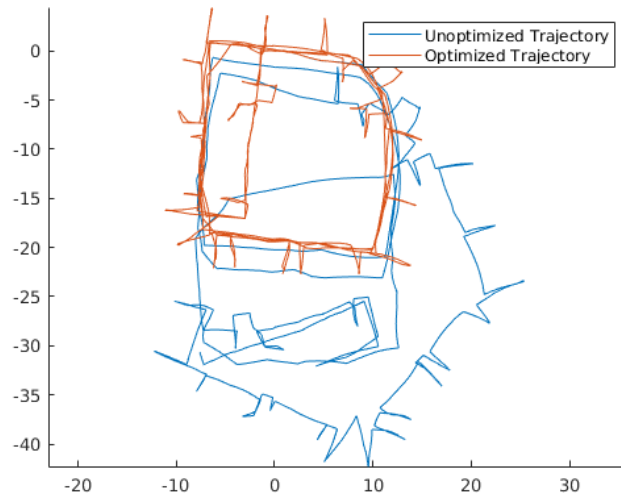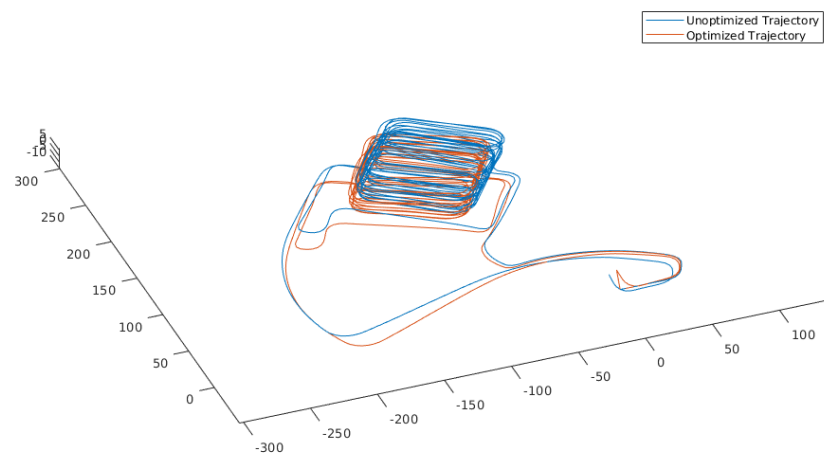
Figure 2: Expected result for task 1 C.
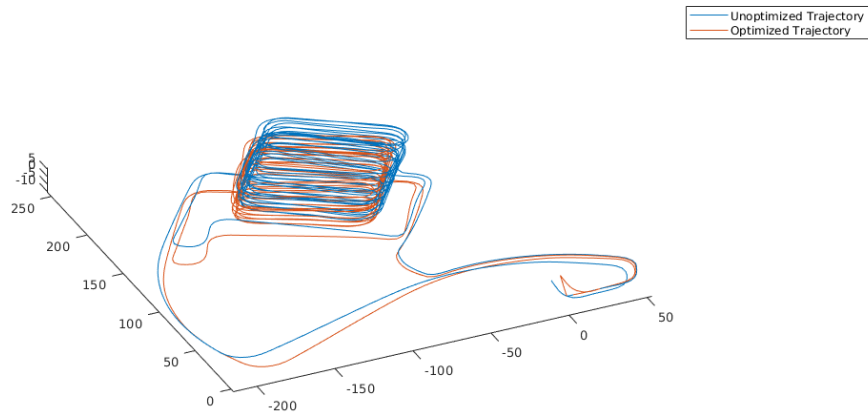
Figure 3: Expected result for task 2 B.

Figure 4: Expected result for task 2 C.