NA 568 Mobile Robotics: Methods & Algorithms Winter 2023 – Homework – SLAM

Maani Ghaffari
University of Michigan

January 4, 2023

This is a reminder that no late HW is accepted. We are using Gradescope for turning in HW; see relevant information on the course Canvas site. This Homework is mandatory and is not part of Homeworks 1-6. This means your grade will remain as is.

This problem set counts for about 11% 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, February 24, 2023 (Anywhere on Earth Time). This corresponds to 7:55 AM on February 25, 2023, 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_hw-slam.zip`:
   `alincoln_hw-slam/`
   `alincoln_hw-slam/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_hw-slam.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.

Read the Following before Getting Started!

Some students had many problems with this assignment. If the following doesn't solve your problems, feel free to post your questions on Piazza!

1. For everyone:
   - For `1A` and `2A` you are required to find a good way to load the G2O file we provided. It is suggested to make it a function which can be called to solve later problems. We don't have specific formatting requirement on the function so you can decide the input/output of that function. You don't need to write in your submitted writeup, but including some comments in the code would be highly appreciated since it could make grading process easier!

- In the G2O file we provide entries of information matrix. You can use GTSAM built-in method called `noiseModel.Gaussian.Information()` method to load it.

2. For C++ users:

   - If you are using C++ for this assignment, make sure to use `double` data type instead of `float`.
   - To plot your result, you can choose either save all the data and use plot libraries in either Python or MATLAB, or use C++ plot libraries like matplotlib-cpp. Feel free to choose the most comfortable way to do this.
   - To compile your C++ code you might want to write the `CMakeLists.txt`. We don't provide it so you might want to do this by yourself. There are many examples online for reference. But if you are not familiar with C++ and CMake feel free to consider using Python or MATLAB for this assignment.

3. For Python users:

   - If you are using Python wrappers, make sure to construct the GTSAM Pose correctly by following the official documentation. When you encountered issue like `Thrown when a linear system is ill-posed`, make sure to check the following. We noticed some students use the code shown below experienced imperfect plot caused by precision issue.
     `gtsam.Pose2(edge[2:5])`.

     Instead, the correct construction of the `Pose2` is
     `gtsam.Pose2(edge[2],edge[3],edge[4])`.

     You should follow the same pattern when constructing `Pose3`.

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 $[q_{11}\ q_{12}\ q_{13}\ q_{22}\ q_{23}\ q_{33}]$ 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).*

[2] https://www.dropbox.com/s/vcz8cag7bo0zlaj/input_INTEL_g2o.g2o?dl=0
[3] https://github.com/RainerKuemmerle/g2o/wiki/File-Format-SLAM-2D

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.

Algorithm 1 `incremental_solution_2d`(poses, edges)

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

```
isam ← gtsam.ISAM2()                                   ▷ Initialize isam solver
for every pose in poses do
    graph ← NonlinearFactorGraph                       ▷ Initialize the factor graph
    initialEstimate ← Values                           ▷ Initialize the initial estimation
    (id_p, x, y, θ) ← pose                             ▷ Extract information from the current pose
    if id_p == 0 then                                  ▷ The first pose
        priorNoise ← some noiseModel                   ▷ Use a predefined noise model
        graph.add(PriorFactorPose2(0, Pose2(x, y, θ), priorNoise))
        initialEstimate.insert(id_p, Pose2(x, y, θ))
    else                                               ▷ Not the first pose
        prevPose ← result.at(id_p − 1)                 ▷ Use last optimized pose
        initialEstimate.insert(id_p, prevPose)
        for every edge in edges do
            (id_e1, id_e2, dx, dy, dθ, info) ← edge    ▷ Extract information from the current edge
            if id_e2 == id_p then
                cov = construct_covariance(info)       ▷ Construct a covariance matrix from the information vector.
                Model ← noiseModel.Gaussian.Covariance(cov)
                graph.add(BetweenFactorPose2(id_e1, id_e2, Pose2(dx, dy, dθ), Model))
            end if
        end for
    end if
    isam.update(graph, initialEstimate)
    result = isam.calculateEstimate()
end for
```

Hint: You may use `NonlinearFactorGraph` as your graph, use `gtsam.ISAM2()` as your update algorithm, 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

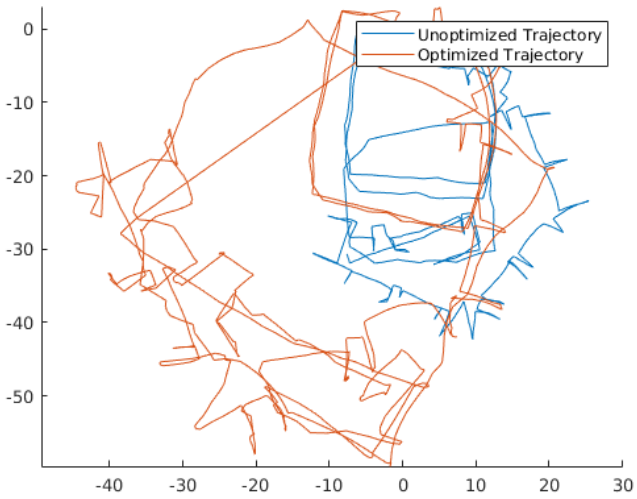

Figure 1: Expected result for task 1 B.

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.

FAQ

1. Q: How can I resolve the issues when installing the libraries?
   A: Google your error information first, and try to find the answer on the GitHub issues page and piazza. If it still exists, please send a new piazza post with your system information, the version of your coding language, and your error information.

2. Q: I'm familiar with C++ and Python. Which language should I choose for the implementation?
   A: The key goal of this assignment is to understand the algorithm and implement it using GTSAM. So choose the one you are most comfortable with! Generating plots is easier in Python/MATLAB, while C++ has better documentation of GTSAM. Mixing these is also acceptable, you can save your optimization result from C++ and write a script to visualize it in Python/MATLAB. Also, check this nice library for plotting in C++ `https://github.com/lava/matplotlib-cpp`.

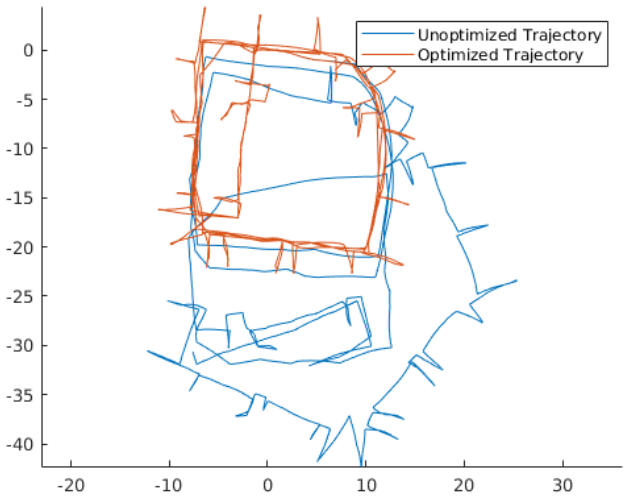

Figure 2: Expected result for task 1 C.

In practice, when the computational time matters in the application, which is mostly true for robotics projects if we want to deploy the algorithms on the robot eventually, C++ is preferred among the three languages. Writing GTSAM in C++ is a more common practice for real applications. Thus, the advantage of solving this homework in C++ is that you'll better understand the library in a way closer to future applications in industry and research.

3. Q: I'm a Python user, but GTSAM only has C++ Doxygen documentation. How do I know how to implement the algorithms?
   A: Great question! Looking at GTSAM python examples from Github would be sufficient for the Homework! Also, if you wish to look into detailed function explanation and usage, unfortunately, you have to refer to C++ documentation.

4. Q: Why I can't upload my whole working space folder to gradescope?
   A: It might not be possible to upload MATLAB toolbox or data files. Try to upload the scripts only, and please leave instructions about where and how to set the data path in your code so that our grader can modify and run your code easily.

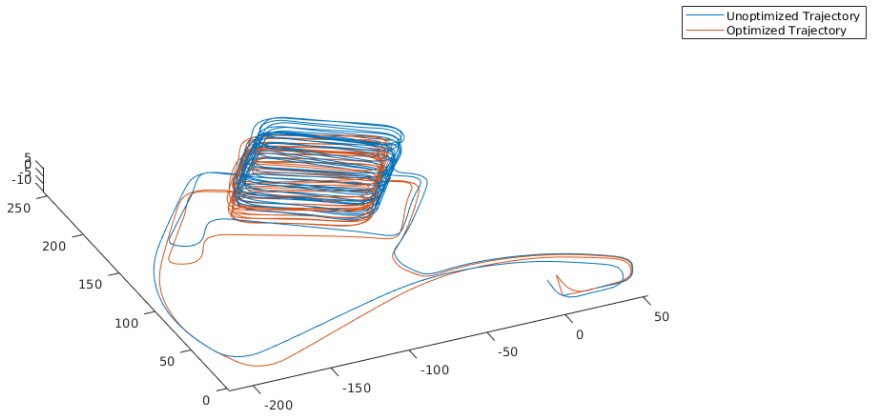

Figure 3: Expected result for task 2 B.

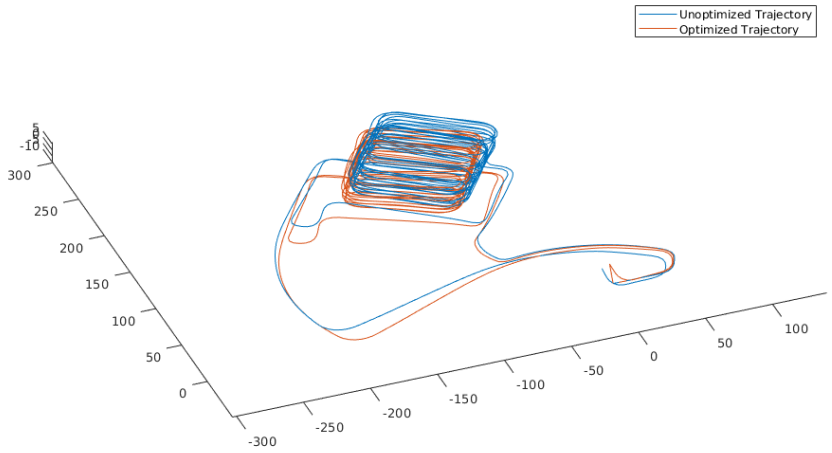

Figure 4: Expected result for task 2 C.