Department of Computer Science, University College London

COMPGX04 Worksheet 04: Introduction to Graph-based Optimisation

February 21, 2018

1 Getting started

1.1 Introduction

In this week we are going to use GTSAM [1], which is a toolbox that makes working with factor graphs easy. Factor graphs are graphical models, where unknown variables (referenced by a *key*) are nodes that are connected by *factors*. Assume, for instance, that you know the odometry of an object and you want to estimate the poses at three timestamps. In this case, the poses would correspond to variables, which are subject to estimation, and the factors would be based on a probabilistic motion model. Please have a look at the first pages of **gtsam.pdf** (see Moodle or GTSAM-website), which provides a nice introduction.
GTSAM is implemented in C++ but also provides a Matlab wrapper that allows faster prototyping.

1.2 Installation

Unfortunately the versions available on GTSAM's website [1] contains bugs and does not work "out-of-the-box". We have a locally modified version of GTSAM which fixes the errors. Pull requests have been made for these, but they have not been accepted into the code yet. Therefore, you will need build the code locally.
The basic system requirements are needed (all systems):

1. Matlab.
2. A C++ 14 compliant compiler.
3. cmake (version 2.6 or above).
4. The boost libraries (greater than version 1.43).

1.2.1 Linux and macOS

The installation process is similar in both cases and requires building from source. Download the file from moodle and uncompresses:

```
PPLayouts:TMP2 ucacsjj$ unzip -q gtsam.zip
PPLayouts:TMP2 ucacsjj$ cd gtsam
PPLayouts:gtsam ucacsjj$ ./install.bash
Creating install directory /opt/gtsam/compgx04
Creating the build subdirectory
Creating the build subdirectory
-- The CXX compiler identification is AppleClang 9.0.0.9000039
-- The C compiler identification is AppleClang 9.0.0.9000039

...

<<<LOTS OF TEXT; YOU CAN SAFELY IGNORE BOOST WARNINGS>>>

...

-- Configuring done
-- Generating done
-- Build files have been written to: /Users/ucacsjj/Proj/Other/TMP2/build

...

<<LOTS OF TEXT BUILDING>>

...

[100%] Generating ../wrap/gtsam/gtsam_wrapper.cpp
Scanning dependencies of target gtsam_matlab_wrapper
[100%] Building CXX object gtsam/CMakeFiles/gtsam_matlab_wrapper.dir/
__/wrap/gtsam/gtsam_wrapper.cpp.o
[100%] Linking CXX shared module ../wrap/gtsam_mex/gtsam_wrapper.mexmaci64
[100%] Built target gtsam_matlab_wrapper
Install the project...
-- Install configuration: "Release"


...

<<LOTS OF TEXT INSTALLING>>

...
```

This will automatically build and install the libraries. By default, these are placed in `/opt/gtsam/compgx04`,

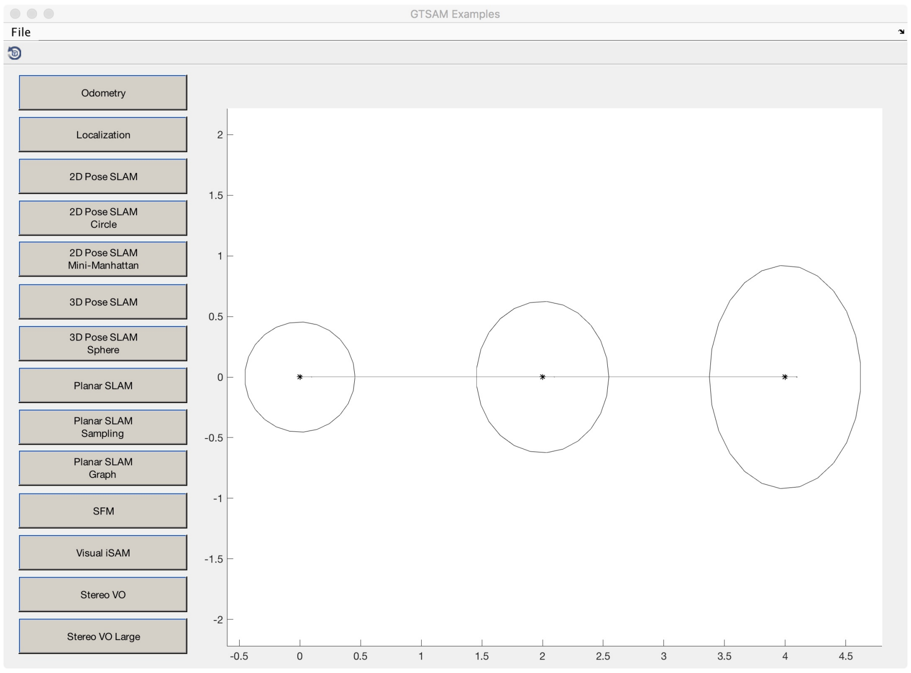


Figure 1.1: The GUI opened by `gtsamExamples`. The buttons on the left run various tests, and the window of the right shows the the output of the estimator.

which shouldn't interfere with any package management systems.

NOTES FOR LINUX USERS If you get an error like *Invalid MEX-file [...] libstdc++.so.6: version 'GLIBCXX_3.4.21' not found [...]* when starting Matlab, make sure that you select the correct standard library version. For example by running: **env LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libstdc++.so.6 matlab**.

1.2.2 Windows

Instructions coming! However, you will need to use the cmake GUI and download a binary distribution of the boost libraries.

1.3 Testing

GTSAM comes with a number of examples which are very useful to understand the code; we highly recommend you explore them. To gain access to them, start matlab. You then need to inform matlab where GTSAM has been installed. After that, you can run the example:

```
>> addpath(genpath('/opt/gtsam/compgx04'))
>> gtsamExamples
```

If this is working correctly you should see the window appear in Figure 1.1.

2 Pose estimation

These tasks are designed to familiarise you with GTSAM.

2.1 Odometry Example

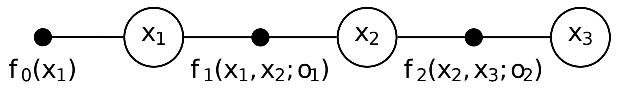


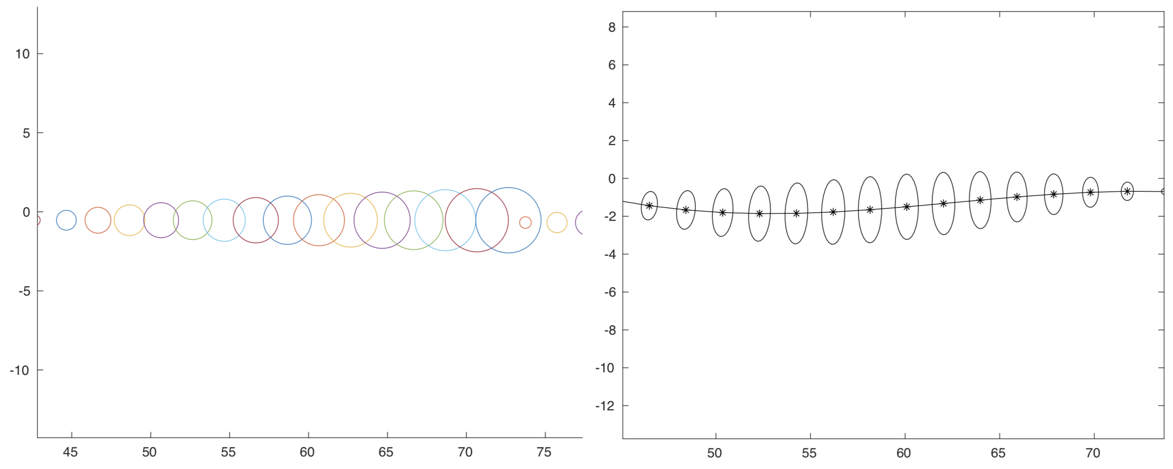
Figure 2.1: Odometry Example

The odometry example generates a graph whose structure and estimates are illustrated in Figure 2.1.
First look at the odometry example source code. You can access it by typing:

```
>> edit OdometryExample.m
```

Read the source code and, comparing it with the material in **gtsam.pdf**, make sure that you understand what the different method means.
Modify the code so that it can use a for loop and create 10 poses and link them together in a chain. See what happens if you set the `initialEstimate` on each pose to be the origin, versus stating it in roughly the right place (the robot moves 2 steps to the right each time).

2.2 Comparison Between Filtering and Smoothing



(a) Filter (b) Smoother

Figure 2.2: test

You have already learned how the Kalman-Filter works. In this task, we want you to understand the difference between *smoothing* and *filtering*, by solving the same problem using a Kalman-Filter and a factor graph. The problem involves 2D pose estimation given noisy odometry and pose measurements with missing data.

2.2.1 System Description

The state of vehicle at time step k is

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix} \tag{2.1}$$

The control input is the distance travelled d_k and the angle turned $\Delta\psi_k$ between one step and the next. The process model is

$$\mathbf{x}_{x+1} = \mathbf{x}_k + \begin{bmatrix} d_k \cos\left(\psi_k + 0.5\Delta\psi_k\right) \\ d_k \sin\left(\psi_k + 0.5\Delta\psi_k\right) \\ \Delta\psi_k \end{bmatrix}. \tag{2.2}$$

The process noise is modelled as additive errors on the measurements of d_k and $\Delta\psi_k$.
The sensor directly measures the position and orientation of the robot,

$$\mathbf{z}_k = \begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix}. \tag{2.3}$$

The measurement errors are additive and zero-mean. The measurements are only available infrequently. Therefore, both your filter and your GTSAM implementation will have to handle missing data. For the Kalman filter you will have to use an EKF. For GTSAM, you might find the source code for `LocalizationExample.m` particularly useful (which you can view using the same edit command above).
The source code `task2_2_skeleton.m` provides a skeleton. It provides an implementation of the simulation model and provides hooks where you should include your Kalman filter and GTSAM implementations.

2.2.2 Tasking

We would like you to do the following:

1. Complete the implementation of the EKF-based filter for this example.
2. Complete the GTSAM implementation for this example.
3. By looking at plots with covariance ellipses as well as the graphs which show how the errors and covariances behave over time, compare the level of accuracy of the two approaches. See if you can determine why the filter exhibits a sudden drop in covariance, but the smoother does not.
4. Time permitting, experment with other properties of GTSAM, such as the performance of it versus the Kalman filter when there is a lot of noise, or when there are different combinations of measurements available.

References

[1] GTSAM: `https://research.cc.gatech.edu/borg/gtsam`