Sparse Feature-Based Visual SLAM

COMP0130 Coursework 2

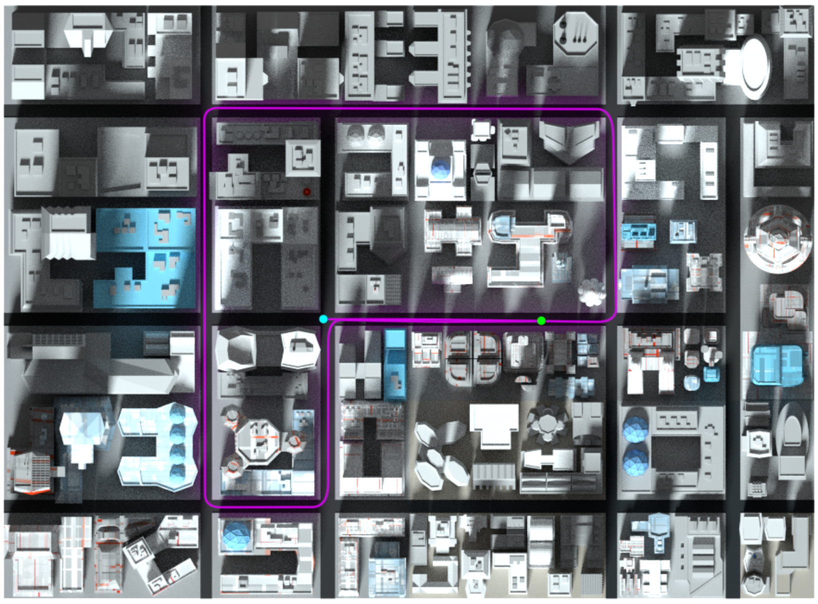
Simon Julier (s.julier@ucl.ac.uk), Sebastian Friston (sebastian.friston.12@ucl.ac.uk), Seb Kay (sebastian.kay.15@ucl.ac.uk)

Overview

Assignment Release Date: Thursday 28st February, 2019
Assignment Submission Date: 23:55 Friday 15th March, 2019
Weighting: 30% of module total
Final Submission Format: each group submits a single zip file which contains both the source code and a report in PDF format.

Assignment Description

In this assignment, you will investigate and develop the main components of a vehicle-based SLAM system which is implemented in a highly optimised version of GTSAM called iSAM.



Plan view of the simulated urban environment. The purple lines show the trajectory taken by the robot as it drives around the city. The robot starts at the green dot (slightly to the right of the middle of the map) and drives the route in a clockwise direction. The robot finished at the cyan point.

The robot is equipped with three types of sensors: a vehicle odometry sensor (relative transformations), GPS (modelled as direct measurement of position) and a range-bearing sensor which measures range to the landmarks. For simplicity, all sensors operate in 2D. The models for each sensor are described in the appendix.

We assume that data association ambiguity has already been addressed. Therefore, each measurement you receive will have a unique (and correct) landmark ID. You can view your work as, in effect, looking at the backend behaviour of a SLAM system.

An event-based matlab framework, has been provided. More details about how this code works is included at the end of this report.

The assignment consists of two parts:

1. Implement odometry and GPS-based fusion and compare the performance with a Kalman filter. This will use similar code and analysis that you investigated for the lab exercise. (50% mark.)
2. Complete the implementation of SLAM system using the range bearing sensor and assess the quality and behaviour of the map. (50% mark.)

Your submission will consist of a single zip file which contains two things: a written PDF report, and your code.

Report. This should be separated into two sections, one per part. The report should be no more than 8 pages of A4, including all diagrams, equations and references (should you choose to include any).

Code. The code, written in matlab, must be executable for your answers for each task. **If the code does not run, you may be penalised up to 20% of your final mark.**

Part 1: Odometry and GPS Fusion

Task 1.1: Odometry

Using your code from the last workshop, complete the implementation of the odometry for both the Kalman filter-based and iSAM based localization systems. For the Kalman filter, you will need to complete the implementation of `predictMeanAndJacobians`
In the `answers.kalman.OdometryOnlyLocalizationSystem` class. For the GTSAM system, you will finish the implementation of the `computeRelativeVehicleTransformAndCovariance` method in `answers.gtsam.OdometryOnlyLocalizationSystem`. Both classes contain documentation of what is required.

By running both localization systems on the script `task11CompareOdometry`, compare the computed covariances of both approaches. What do you notice about the differences in the vehicle covariance of each over time? For iSAM, plot the time required to run the optimiser. How does it grow with time? Why do you think it grows?

Task 1.2: GPS

Complete the Kalman filter and GTSAM implementations of the `GPSLocalizationSystem` classes which can be found in `answers.kalman` and `answers.gtsam`. Note that, unlike the labwork, the GPS does not provide orientation information. For the Kalman filter, you can use the regular Kalman filter update rules. For GTSAM, you might find the `LocalizationExample` useful.

You should use the script `task12CompareGPS`

Part 2: SLAM System

In this part, you will extend the GTSAM based localization system to support the creation of a full map which is built using SLAM. You will investigate the properties of SLAM as other sensors become available.

Task 2.1: SLAM System Development

Implement the method `processLaserObservation in answers.gtsam.LaserSensor2DSLAMSystem`. You might find GTSAM's `PlanarSLAMExample` to be helpful. This method implements both the initialisation of new landmarks and the update of existing landmarks with new measurements. Explain your code and provide a plot to illustrate the performance of the SLAM system.

The script `task21SLAMDevelopment` can be used for development. This is a small example (the same as in part 1).

Task 2.2: Comparison with Odometry and GPS

Experiment with the value added effects of odometry and GPS on your SLAM solution. To do this, use the script `task22SLAMComparison`. This introduces a larger map, but your code from task 2.1 should be directly applicable. Note that plotting can become very slow, and you might want to change how often the graphics output is generated.

You should compare:

1. What are the effects of enabling / disabling odometry. Why do you think the differences arise?
2. What is the effect of adding GPS measurements? You should investigate if a GPS measurement is added once per second, once ever two seconds and once every five seconds.

A qualitative analysis in which you plot results is sufficient.

One problem with graphs is that as they get bigger they get slower. Can you propose a method to reduce the computational cost?

Materials Description

All the software provided to support this coursework can be obtained from a public branch from github. The URL is:

```
https://github.com/ucl/comp0130.git
```

This consists of the matlab code and data. The code uses the version of GTSAM you have installed already.

Getting Help

You are encouraged to use the assignment discussion forum to help when you get stuck. Please check the forum regularly and try and answer each other's questions. We will be checking the forum as often as I can and will be answering any questions that have remained unanswered. Please note that if you have questions about coursework, please post them to the forum directly rather than emailing me. The reason is that all students will be able to see the reply.

Submission Format and Structure

Each group will submit a single zip file. The name of the zip file will be of the form "Group_XX_CW2.zip", where XX is the group name. The zip file, when uncompressed. The code will not be marked independently, however, the code may be tested to ensure that it works correctly and supports the results presented in the report. As explained above, if the submitted code does not run, your final mark may be penalised.

Marking Guidelines

All reports will be marked against the marking rubric, which is downloadable from the course's Moodle page. The mark weighting for each section is as follows:

1. Algorithms Description (40%)
   How well are the different algorithms described and contrasted?

2. Implementation, Analysis and Presentation of Results (40%)
   How thorough is the analysis and discussion of the results? How well do they compare between the different algorithms and with properties of those algorithms? Are the algorithms well-supported by quantitative evidence?

3. Format, structure, referencing and clarity of writing (20%)
   Is your report well laid out and does the write-up follow a clear structure? Have you included any references to show background research/reading? Is your writing free from spelling, punctuation, and grammatical errors?

Submission and Feedback

The deadline for submission is **23:55 on Friday 15th March, 2019**. As explained above, please submit a single zip file per group. Please pay attention to both the requested file name and the file structure.

Details of the Scenario

Coordinate Systems Used

An East-North-Up coordinate system is used. This is a right-handed coordinate system. Looking at the plan view of the scenario in Figure 1, the +ve x points to the right, +ve y axis points up and +ve z points out of the page. Angles are similarly defined.

Vehicle Motion Model Assumption

To work with the camera-based types, the vehicle state is represented as a Pose3, which consists of 3 dimensional position (x, y, z) and 3 axis rotation (φ, θ, ψ). However, you may assume that the vehicle drives on ground which is flat and level. Therefore, $z = \varphi = \theta = 0$ and it is not necessary to use any sophisticated analysis of 3D rotating frames. We do not discuss these degrees of freedom further. (However, they do introduce some implementation issues in the code – please see there for details).

The equations of motion which describe the movement of the vehicle are as follows. At time step k the vehicle odometry consists of a (signed) speed V_k and a rate of heading change $\dot{\psi}_k$. Given that the time interval is ΔT_k, the new position and orientation of the robot is approximated by

$$\begin{aligned} x_k &= x_{k-1} + V_k \Delta T_k \cos(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k) \\ y_k &= y_{k-1} + V_k \Delta T_k \sin(\psi_{k-1} + 0.5\Delta T_k \dot{\psi}_k) \\ \psi_k &= \psi_{k-1} + \Delta T_k \dot{\psi}_k \end{aligned}$$

The process noise is approximated as additive errors on the measured speed and rate of change. The covariance on the speed and control inputs is provided with each message.

Sensing Systems

The odometry system returns the wheel speed and rate of change of heading. These are both corrupted by zero-mean noises.

The GPS returns a measurement of the (x,y) position of the vehicle in global coordinates. It does not provide orientation information.

The range-bearing sensor returns the azimuth, elevation and range to a 3D landmark. Since we are only doing 2D SLAM in this coursework, this has to be converted to a 2D (range-bearing) measurement. The equations to do this are already provided for you in `answers.gtsam.LaserSensor2DSLAMSystem`.