Department of Computer Science, University College London

COMP0249 Coursework 01: Graph-based Optimisation and SLAM

Current Revision: 20250212

1 Overview

- **Assignment Release Date: Monday 10st February, 2025**
- **Assignment Submission Date: Monday 24th February, 2025**
- **Weighting: 50% of module total**
- **Final Submission Format. Each group must submit two things:**
  1. **A zip file (which contains the source code implemented to tackle this coursework). The name of the zip file must be of the form** `COMP0249_CW1_GROUP_g.zip`**, where** `g` **is your code group.**
  2. **A report in PDF format. It will be named** `COMP0249_CW1_GROUP_g.pdf` **where again** `g` **is your group code.**

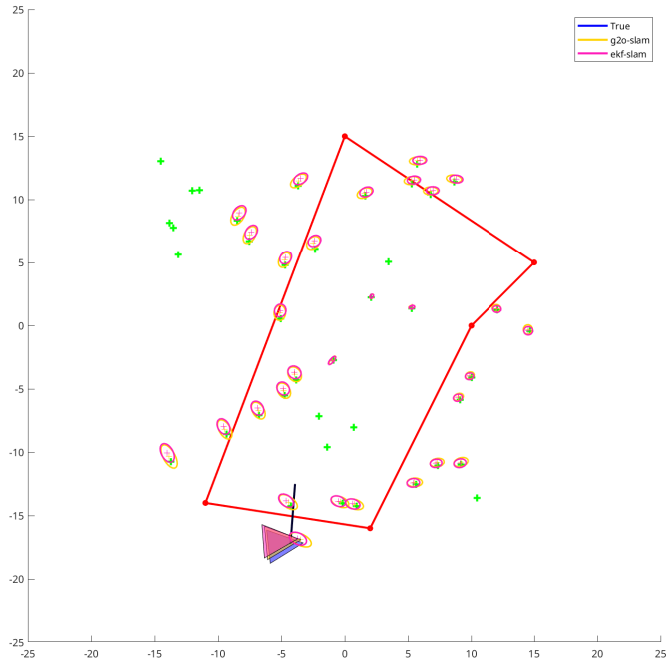

Figure 1: The scenario. The vehicle drives a path (purple line) through an environment populated by landmarks (green crosses). Various landmark estimates, with their associated covariance ellipses, are shown. Two SLAM systems: an EKF-based one (purple) and a factor graph based system (yellow) are shown.

Note that a penalty of 2% will be applied if the files are submitted with the wrong name.

2 Coursework Description

The goal of this coursework is to gain experience with how a graphical-model based SLAM system is implemented and assess its properties. Note this coursework heavily relies on the code and experience you gained in the first four labs for this module. The scenario is shown in Figure 1. A wheeled robot (DriveBot) drives through a two-dimensional environment which is populated by a set of landmarks. The number of landmarks and their locations are not known, and must be estimated by the SLAM system.

The vehicle is equipped with three types of sensors:

1. **A vehicle odometry sensor, which records speed and heading.**
2. **A GPS sensor which measures vehicle position.**
3. **A 2D range bearing sensor which measures the range and bearing to the landmarks in 2D.**

In addition, the simulator provides information which is used to specify the initial position and orientation of the robot.
The mathematical models for the vehicle and landmark models, data sources and sensors are described in Appendix A.

3 Questions

1. **In this question you will implement the prediction step (using the process model in Subsection A.2).**

    a. **Describe and illustrate the structure of factor graph which is used to address this problem (a clear hand drawn sketch or a screenshot with attribution is acceptable). Your description should identify the different types of vertices as well as the different types of edges required. For the vertices, identify the state and addition operator. For the edges, identify their type, the vertices that they need to connect to, the equations of the models they contain and the measurements they encapsulate. It is sufficient to cite equations from the appendix rather than write equations in full.**

[15 marks]

b. Complete the implementation of `PlatformPredictionEdge` by providing implementations of `initialEstimate`, `computeError` and `linearizeOplus`. Since events can arrive at different times, you must ensure that timestamps are handled correctly. Provide code snippets for these methods in your text to briefly explain how you implemented each method and provide results diagrams to show that the position and heading errors are consistent with the predicted covariance values.

To test your implementation, you can use the script:

```
>> cw1.q1_b
```

[9 marks]

c. In this subquestion we are going to have a quick look at how the graph optimizes. Run the script:

```
>> cw1.q1_c
```

This optimizes the graph every time step (which is a very inefficient thing to do). Once the code has finished, it will plot a pair graphs using internal performance data which show the chi2 values (the value of the cost function when the optimizer finishes at each time step) and the optimization values over time. What trend do you see in both graphs? How do you think they might relate to one another? What do you think might be the underlying cause of this?

[7 marks]

[Question total 31 marks]

2. **In this question you will complete the implementation of the factor graph SLAM system by implementing the rest of the** `LandmarkRangeBearingEdge` **class using the model in Section A.4.**

   a. **Describe and illustrate the structure of factor graph which is used to address the full SLAM problem (a clear hand drawn sketch or a screenshot with attribution is acceptable). Any new vertices and edges you need to supplement the graph from question 1. As before, your description should identify the different types of vertices as well as the different types of edges required. For the vertices, identify the state and addition operator. For the edges, identify their type, the vertices that they need to connect to, the equations of the models they contain and the measurements they encapsulate. It is sufficient to cite equations from the appendix rather than write equations in full.**

   [9 marks]

   b. i. **Complete the implementation of** `LandmarkRangeBearingEdge` **by providing implementations of** `initialEstimate`, `computeError` **and** `linearizeOplus`. **Provide listings of the code snippets for these methods in your text and briefly describe how you implemented them.**

   To test your implementation, you can use the script:

   ```
   >> cw1.q2_b
   ```

   This script enables an EKF-SLAM system and disables all noises. As a result, both your implementation and the EKF should provide exaclty the same correct results. To test with noise, change the value of `perturbWithNoise` **in** `q2_b.json` **to** `true`.

   Provide code snippets for these methods in your text to briefly explain how you implemented each method.

   [7 marks]

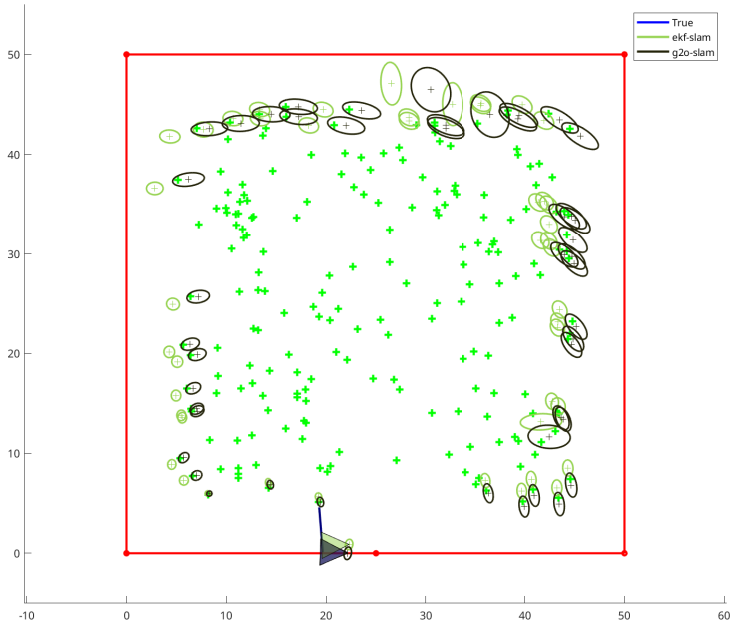

Figure 2: The scenario for Q2c. The vehicle drives a path (straight red line with small circles) through an environment populated by landmarks (green crosses). Various landmark estimates, with their associated 2σ covariance ellipses, are shown for the two SLAM systems shown in the legend: EKF (light green) and G2O (dark grey).

ii. Plot the timing and chi2 and error plots for the system running in this case. What trend do you see? How does this related to what you saw in Q1b?

[7 marks]

[Subquestion total 14 marks]

c. In this subquestion, we'll look at a larger scenario which is designed to be more punishing for the SLAM algorithms. In particular, the robot has to operate over a larger area, and the sensor detection range is smaller.

Run the script:

```
>> cw1.q2_c
```

i. You should see something like the plot show in Figure 2 (however, the results will vary from run-to-run). This figure illustrates that the EKF is not producing a consistent map but the factor graph is. What evidence in this figure shows this to be the case?

[5 marks]

ii. As noted, the sensor detection range is smaller. Why do you think this might cause a significant change in the performance of the two algorithms?

[5 marks]

iii. By increasing the value of `detectionRange` **in** `+cw1s/config/q2_c/scenario.json`**, can you identify a value at which they both produce similar estimates? Provide the evidence you were successful. Why do you think this produces better results?**

[3 marks]

Note it is sufficient to find a single value. You do not have to exhaustively search to find the minimum detection length required.

[Subquestion total 13 marks]

[Question total 36 marks]

3. In the last two questions, you have see the graph is running very slow. This is partially due to implementation issues (if you profile, you will find that the actual code doing the maths takes up less than 0.1% of the time required to run the operations — this is the overhead of using an interpreted language). However, it is also because we have made no attempt to optimise the operation of the graph. We will look at two strategies. The first is graph pruning, which deletes edges. The second is conditioning — we (temporarily) assume that some states are known and do not bother updating them.

For this question, all the code to implement the pruning and fixing strategies is provided and the questions are about interpreting the results.

a. Run the script:

```
>> cw1.q3_a
```

This script compares two versions of the graph (and an EKF): the original g^{2}oSLAM system, and a modified one in which landmarks can have at most 4 observations of any landmark.

i. Plot the generated map and the error plots for the different SLAM systems. For each, identify if the SLAM system seems to be consistent or not. Do you notice a trend in the standard deviation bounds for the different filters in the different graphs?

[4 marks]

ii. Plot the chi2 and optimization times for the the g^{2}o-related estimators. What differences do you see between them for both measures? Provide an explanation of why you think these differences might arise.

[4 marks]

iii. **By analysing the structure and the count of the numbers of edges and vertices in the graph, how effective do you think the strategy is in this case? Are there other environments where this strategy might be more effective? If so, describe what they would be and why you think they would be better.**

[6 marks]

[Subquestion total 14 marks]

b. **An alternative approach is to fix vertices. Fixing a vertex means that the value does not change during the optimization step. In turn, this means that the vertex can be excluded from the optimization step, saving both time and storage. From a probabilistic point of view, fixing a vertex is the same as conditioning on its value — you assume it's value is known perfectly.**

The proposed scheme here fixes vertices which correspond to vehicle poses from more than a few seconds ago. It can be activated by running the script:

```
>> cw1.q3_b
```

Note that at the very end of the run, all the vertices which were fixed are unfixed, and the optimization is run one final time.

i. **Run the script. What chi2 values and optimization times are obtained with this approach? How do these behaviours differ from those for the original g^{2}o SLAM solution? What do you think is the reason?**

[9 marks]

ii. **Examine the estimated position estimates and the covariance estimates of the landmarks during the run and at the end after the SLAM system has finished. What do you observe about the behaviour of them? Explain what you think the underlying cause might be.**

[6 marks]

iii. How successful do you thin this strategy is in this case? What are your reasons? Remember you should discuss the behaviour both during the run and at the end of it.

[4 marks]

[Subquestion total 19 marks]

[Question total 33 marks]

[Coursework total 100 marks]

A System Models

A.1 State Description

The vehicle state is described by its position and orientation in 2D:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \psi_k \end{bmatrix}. \tag{1}$$

A standard right handed coordinate system is used: x points to the right, y points up and a positive rotation is in an anticlockwise direction. Angles are in radians.
Landmarks are in 2D. The state of the ith landmark is given by

$$\mathbf{m}^i = \begin{bmatrix} x^i \\ y^i \end{bmatrix}. \tag{2}$$

A.2 Process Model

The process model is the same as one you have seen in earlier labs,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{M}(\psi_k)\left(\mathbf{u}_{k+1} + \mathbf{v}_{k+1}\right). \tag{3}$$

Note that ΔT_{k+1} can vary from timestep to timestep.
The matrix

$$\mathbf{M}_k = \Delta T_{k+1} \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 \\ \sin\psi_k & \cos\psi_k & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

is the rotation from the vehicle-fixed frame to the world-fixed frame and ΔT_{k+1} is the length of the prediction interval. Note that the prediction interval is governed by when various sensors are available, and can vary through the simulation.
The control input consists of the speed of the vehicle (which is oriented along a body-fixed x axis) together with an angular velocity term,

$$\mathbf{u}_k = \begin{bmatrix} s_k \\ 0 \\ \dot{\psi}_k \end{bmatrix}.$$

The process noise is zero mean, Gaussian and additive on all three dimensions,

$$\mathbf{v}_k = \begin{bmatrix} v_k \\ v_y \\ v_\psi \end{bmatrix}.$$

The noise in the body-fixed y direction allows for slip and the fact, as discussed in the lectures, that the velocity is related to the front wheel and not the body orientation. The process noise covariance $\mathbf{Q}_k$ is diagonal.

A.3 GPS Observation Model

The GPS sensor is a highly idealized one which provides direct measurements of the position of the robot. The observation model is

$$\mathbf{z}^{\mathrm{G}}_{k+1} = \begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} + \mathbf{w}^{\mathrm{G}}_{k+1},$$

where $\mathbf{w}^{\mathrm{G}}_{k+1}$ is the observation noise which is Gaussian, independent, zero-mean and with covariance $\mathbf{R}^{\mathrm{G}}$. This matrix is diagonal and constant.

A.4 Landmark Observation Model

The landmark observation model measures the range, azimuth and elevation of a landmark relative to the platform frame,

$$\mathbf{z}^{\mathrm{L}}_{k+1} = \begin{bmatrix} r^i_{k+1} \\ \beta^i_{k+1} \end{bmatrix} + \mathbf{w}^{\mathrm{L}}_{k+1},$$

where

$$r^i_{k+1} = \sqrt{(x^i - x_{k+1})^2 + (y^i - y_{k+1})^2}$$
$$\beta^i_{k+1} = \tan^{-1}\left(\frac{y^i - y_{k+1}}{x^i - x_{k+1}}\right) - \phi_{k+1}$$

The covariance of $\mathbf{w}^{\mathrm{L}}_k$, $\mathbf{R}^{\mathrm{L}}_k$ is diagonal and is assumed to be time invariant and the same for all landmarks.