Department of Computer Science, University College London

COMP0130 Coursework 02: Graph-based Optimisation and SLAM

Current Revision: 202402225

1 Overview

- Assignment Release Date: Sunday 25th February March, 2024
- Assignment Submission Date: 13th March, 2024
- Weighting: 33% of module total
- **Final Submission Format**. Each group must submit *two* separate files:
  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 `COMP0130_CW2_GROUP_g.zip`, where `g` is the letter code of your group.
  2. A report in PDF format. It will be named `COMP0130_CW2_GROUP_g.pdf` where again `g` is the letter code of your group.

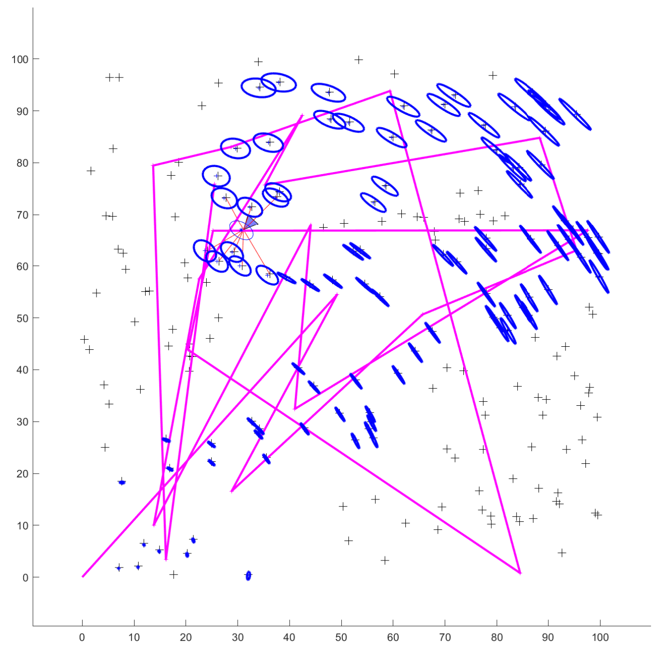

Figure 2.1: The scenario. The vehicle (small blue triangle,starting from (0,0) at bottom left) drives a path (purple line) through an environment populated by landmarks (blue crosses). Various landmark estimates, with their associated covariance ellipses, are shown.

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. The system will be implemented using the MiniSLAM event-based framework you met in Topic 02 Lab 01 and the MATLAB port of the g^2o port you met in Topic 02 Lab 02. We assume you have familiarity with the questions and worked answers to those workshops.

The scenario is shown in Figure 2.1. 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 four types of sensors:

1. A vehicle odometry sensor, which records speed and heading.
2. A compass, which measures the heading of the robot platform.
3. A GPS sensor which measures vehicle position.
4. 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. Please note the models for the GPS are much simpler than those introduced in part 1 of this module. In principle these full models can be incorporated into a factor graph to produce a tightly coupled solution, but we do not do this to focus more on graph construction and use.

A simulation system (`DriveBotSimulator`) creates a stream of events which the SLAM system processes. Data association has already been addressed. Therefore, each landmark measurement you receive will have a unique (and correct) landmark ID.

The coursework consists of three questions:

- Q1: Implement the prediction and update steps using the vehicle process model, together with compass and GPS sensors. You will also investigate some general properties about how the graph optimizers work. (35.5%)
- Q2: Extend your implementation to undertake SLAM and explore some characteristics of how the algorithm behaves. (35.5%)
- Q3: Discuss ways improve the performance of the SLAM system. This also provides further insight into how the graphs operate. (29.0%)

Please note that different subparts of different questions have their own launcher scripts of the form `qx_y.m`. Although these scripts look very similar to one another, there are subtle differences between them to support the different subquestions (for example, some use datasets which are considerably longer than others, and the combinations of sensors used for different subparts varies). **Furthermore, to mark your coursework, we will execute each script in turn and so the changes required to answer each question should not interfere with us being able to run the launcher code for a previous question.**

3 Questions

1. In this question you will implement the prediction step (using the process model in Subsection A.2), debug the implementation of the compass (using the observation model in Subsection A.3) and implement the GPS sensor (using the observation model in Subsection A.4). Note that the compass, GPS and odometry data all arrive at different rates, and *you must ensure that timestamps are handled correctly in all of your answers*. The graph must be initialized using a set of initial conditions.

   a. Provide and describe a diagram of the structure of factor graph which can be used to just predict the robot pose over time, and update it using the GPS and compass sensors. Your description should list the different types of vertices and edges. Write down the equations needed to implement the different types of edges and vertices. You will be penalized for including edges or vertices which are not required for this task.

      Hint: Think about how angular discontinuities affect your solution and how these are related to the $\boxplus$ operator.

      [12 marks]

   b. Implement the prediction step in the `DriveBotSLAMSystem`.

      i. In your report describe which methods you edited to complete the functionality. In the code, please provide comments which explain what your implementation does. These should be prefixed by "Q1b".

         [6 marks]

      ii. Using the script `q_1b.m`, plot the graphs of the state error and the covariances computed by the estimator. Describe the behaviours you see. Also describe, but do not plot, the behaviour of the residual values. Explain why you think the state, error and chi2 values behave the way you observe.

          [6 marks]

c. The robot will use a compass measurement. An initial implementation of the appropriate edge has been provided but this contain bugs.

   i. Modify the script `q1_c.m` so that the simulator will generate compass measurements. By studying the error characteristics and other behaviours of the graph, present the evidence which support the fact that the implementation of the edge is, in fact, buggy and explain what you think the error is.
   [6 marks]

   ii. Modify the code to produce the correct solution. Explain your implementation with the source code and present results to confirm that the error has been fixed.
   [4 marks]

d. The robot will now use data from the GPS sensor. Your solution *must* treat the GPS measurement as a 2D observation for the answer to be acceptable.

   i. In your report describe which methods you edited to complete the functionality. In the code, please provide comments which explain what your implementation does. These should be prefixed by "Q1d".
   [4 marks]

   ii. Using the script `q1_d.m`, generate results to show
   generate plots of the state error and the covariance. Present these graphs. Discuss the behaviours you see and why they arise.
   [8 marks]

e. The results so far have only shown the behaviour at the end of the run. In this final part, we will look at bit more at the operation of the graph.

   i. Modify the script `q1_e.m` so that the optimizer runs once every timestep instead of at the end of the run. Plot the curves of the optimization time and the `chi2` over the run. Describe the behaviours you observe and propose an explanation for why they arise.
   [6 marks]

   ii. Enable the compass in `q1_e.m`. Explain why the time and chi2 behaviours now differ from those when just the GPS was enabled.
   [4 marks]

[Total 56 marks]

2. In this question you will extend the `DriveBotSLAMSystem` to support SLAM. The observation model for these landmarks is given in Subsection A.5. This system will build directly off of the system you built in Q1. Note that the GPS and the compass will only be used in the optional unmarked task at the end of this question.

   a. Provide and describe a diagram of the structure of factor graph which can be used to perform SLAM. Your description should list the different types of vertices and edges. Write down the equations needed to implement the different types of edges and vertices. For edges or vertices which you have described already, it is sufficient to cite back to the earlier answers. You will be penalized for including edges or vertices which are not required for this task.

      Hint: Think about how angular discontinuities affect your solution and how these are related to the $\boxplus$ operator.

      [7 marks]

   b. Implement SLAM in `DriveBotSLAMSystem`.

      i. In your report describe which methods you edited to complete the functionality. In the code, please provide comments which explain what your implementation does. These should be prefixed by "Q2b".

         [9 marks]

      ii. Run the script `q2_b.m` which runs a SLAM example on a single closed loop. Plot the vehicle covariances, `chi2` values and the amount of time to complete an optimization step. Discuss the behaviours you see and why you think they arise.

         [15 marks]

c. We will now look at some more details about the properties of the constructed graph.

   i. By modifying the script `q2_c.m`, investigate the properties of the structure of the graph. You should obtain information on: the number of vehicle poses stored, the number of landmarks initalized, the average number of observations made by a robot at each timestep, and the average number of observations received by each landmark.

      In your report, describe how you calculated these quantities. In the code, please prefix your changes by "Q2c".

      Hint: You can obtain all the necessary information from querying the constructed graph, Matlab's `isa` function, and using methods in the `BaseVertex` and `BaseEdge` classes.

      [7 marks]

   ii. Present the results you obtained from analysing the graph and discuss what these quantities imply.

      [8 marks]

d. One of most important advantages of SLAM is the ability for it to perform loop closure.

   Using the script `q2_d.m`, generate figures which show the map, vehicle and landmark states just before and just after the loop closure event. Explain the form of the covariance matrices you see on the landmark and vehicle estimates immediately before and after the update. By looking at the determinant of the covariance of each landmark, compute the amount by which the uncertainty has changed for each landmark.

   [10 marks]

[Total 56 marks]

e. This is a purely optional activity for those who are interested. No marks are awarded for it.

   Run the script `q2_large_map.m`. This tests the code on a much more substantial example. The landmark covariances matrices look like they are oriented as tangents on a set of concentric circles. Where do you think the origin might be, and what do you think might cause it?

3. This question explores several ways to improve the performance of a SLAM system by deleting parts of the factor graph. Neither the GPS nor the compass are used.

   a. One way to improve the performance of a SLAM system is to maintain all the sensor observation data, but to remove all the vehicle prediction edges. As a result, the SLAM system will act more like a structure-from-motion system.

      i. By referring to the structure of the factor graph, why do you think this might be a sensible strategy to take? When do you think this is likely to be a successful strategy? When is it likely to be unsuccessful?

         [4 marks]

      ii. Modify the `DriveBotSLAMSystem` system so that, when requested, it will delete the vehicle edges before optimization. Your code should support two cases: when all of the prediction edges are deleted, and when all but the first prediction edge is deleted.

         In your report, describe which methods you edited to provide this functionality. In the code, please provide comments which explain what your implementation does. These should be prefixed by "Q3a".

         [4 marks]

      iii. Using the script `q3_a.m`, test the cases when all prediction edges are removed, and all but the first prediction edge is removed. You should find one case will fail. Which case is it, and why do you think this failure occurs?

         [8 marks]

      iv. Plot the optimization time, covariance and chi2 values for the working case where the edges have been removed. How do the results differ from the full SLAM system? Given the difference in performance, discuss if you this strategy on its own successful.

         [8 marks]

b. One approach simplifying graphs is graph pruning.

   i. Describe what the graph pruning approach is and describe one heuristic for graph pruning which you will implement to use with the SLAM system described here.

   [4 marks]

   ii. Implement your solution. You can either modify the existing code (but ensure that it will run the earlier questions correctly) or you can create new classes to implement it (either by subclassing by simply copying the existing code to create a new class). To accompany it, you should write your own script `q3_b.m`, following the pattern of the other scripts, to run your revised SLAM system with the necessary settings.

   In your report, describe how you implemented the functionality. In the code, please provide comments which explains what your implementation does. These should be prefixed by "Q3b".

   [10 marks]

   iii. Evaluate the effectiveness of your graph pruning strategy. Potential measures you might want to include are the graph properties (such as number of vertices), chi2 performance, and error performance. Your evaluation should show whether your changes have improved performance, and provide an explanation of why you think you achieved the results presented.

   [8 marks]

[Total 46 marks]

[Coursework Total 158 marks]

A System Models

A.1 State Description

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

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

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

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

A.2 Process Model

The vehicle process model is the similar to the one you saw in Topic 02 Workshop 02. The model has the form,

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

The matrix

$$\mathbf{M}(\psi_k) = \begin{bmatrix} \cos\psi_k & -\sin\psi_k & 0 \\ \sin\psi_k & \cos\psi_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

is the rotation from the vehicle-fixed frame to the world-fixed frame and ΔT_k 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,

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

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 Compass Observation Model

The compass sensor is a highly idealized one which provides a direct measurement of the orientation of the robot platform frame in a world-fixed coordinate frame. The observation model is

$$\mathbf{z}_k^{\mathrm{C}} = \begin{bmatrix} \psi_k \end{bmatrix} + \begin{bmatrix} \Delta\psi_C \end{bmatrix} + \mathbf{w}_k^{\mathrm{C}},$$

where $\Delta\psi_C$ is the compass angular offset and $\mathbf{w}_k^{\mathrm{C}}$ is the observation noise which is Gaussian, independent, zero-mean and with constant covariance $\mathbf{R}^{\mathrm{C}}$.

A.4 GPS Observation Model

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

$$\mathbf{z}_k^{\mathrm{G}} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \mathbf{M}(\psi_k) \begin{bmatrix} \Delta x_G \\ \Delta y_G \end{bmatrix} + \mathbf{w}_k^{\mathrm{G}},$$

where $\mathbf{M}(\psi_k)$ is given by Eq. A.4, $[\Delta x_G \ \Delta y_G]^\top$ is the GPS position offset (in the platform-fixed frame) and $\mathbf{w}_k^{\mathrm{G}}$ is the observation noise which is Gaussian, independent, zero-mean and with covariance $\mathbf{R}^{\mathrm{G}}$. This matrix is diagonal and constant.

A.5 Landmark Observation Model

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

$$\mathbf{z}_k^{\mathrm{L}} = \begin{bmatrix} r_k^i \\ \beta_k^i \end{bmatrix} + \mathbf{w}_k^{\mathrm{L}},$$

where

$$\begin{aligned} r_k^i &= \sqrt{(x^i - x_k)^2 + (y^i - y_k)^2} \\ \beta_k^i &= \tan^{-1}\left(\frac{y^i - y_k}{x^i - x_k}\right) - \phi_k \end{aligned}$$

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