Lab 9: Data association

MCHA4400

Semester 2 2025

Introduction

In this lab, you will explore geometric compatibility and geometric matching that will serve as a foundation for the landmark SLAM data association aspects of Assignment 1.

GenAI Tip

You are strongly encouraged to explore the use of Large Language Models (LLMs), such as GPT, Gemini or Claude, to assist in completing this activity. If you do not already have access to an LLM, bots are available to use via the UoN Mechatronics Slack team, which you can find a link to from Canvas. If you are are unsure how to make the best use of these tools, or are not getting good results, please ask your lab demonstrator for advice.

1 Feature bundle density

For SLAM with point landmarks, we can consider the following state vector:

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\eta} \\ \mathbf{r}^n_{1/N} \\ \mathbf{r}^n_{2/N} \\ \vdots \\ \mathbf{r}^n_{n_L/N} \end{bmatrix}, \tag{1}$$

where

$$\boldsymbol{\nu} = \begin{bmatrix} \mathbf{v}^b_{B/N} \\ \boldsymbol{\omega}^b_{\mathcal{B}/\mathcal{N}} \end{bmatrix}, \tag{2}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \mathbf{r}^n_{B/N} \\ \boldsymbol{\Theta}^n_b \end{bmatrix}, \tag{3}$$

are the body-fixed velocities and world-fixed pose of the body, respectively, $\mathbf{r}^n_{j/N}$ is the position of the j^{th} landmark and n_L is the number of landmarks in the map.

Let $\mathbf{h}_j(\mathbf{x}) \in \mathbb{R}^2$ denote the pixel coordinates of the feature corresponding to the j^{th} landmark as a function of the state $\mathbf{x}$, i.e.,

$$\mathbf{h}_j(\mathbf{x}) = \texttt{w2p}(\mathbf{r}^n_{j/N}; \mathbf{T}^n_b, \boldsymbol{\theta}) \tag{4a}$$

$$= \texttt{v2p}\Big((\mathbf{R}^b_c)^\mathsf{T}\big((\mathbf{R}^n_b)^\mathsf{T}(\mathbf{r}^n_{j/N} - \mathbf{r}^n_{B/N}) - \mathbf{r}^b_{C/B}\big); \boldsymbol{\theta}\Big), \tag{4b}$$

where $\texttt{w2p}(\cdot)$ and $\texttt{v2p}(\cdot)$ are the `worldToPixel` and `vectorToPixel` functions, respectively, and the camera pose is related to the body pose as follows:

$$\mathbf{r}^n_{C/N} = \mathbf{R}^n_b \mathbf{r}^b_{C/B} + \mathbf{r}^n_{B/N}, \tag{5a}$$

$$\mathbf{R}^n_c = \mathbf{R}^n_b \mathbf{R}^b_c. \tag{5b}$$

To propagate the uncertainties in the landmark position and camera pose through to uncertainty in the predicted feature location in an image using an affine transform, we will need to compute the Jacobian of $\mathbf{h}_j(\mathbf{x})$ with respect to $\mathbf{x}$. This Jacobian has the following block structure:

$$\frac{\partial \mathbf{h}_j}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{h}_j}{\partial \mathbf{r}^n_{B/N}} & \frac{\partial \mathbf{h}_j}{\partial \boldsymbol{\Theta}^n_b} & \mathbf{0}_{2\times 3} & \cdots & \frac{\partial \mathbf{h}_j}{\partial \mathbf{r}^n_{j/N}} & \cdots & \mathbf{0}_{2\times 3} \end{bmatrix} \in \mathbb{R}^{2\times(12+3n_L)}. \tag{6}$$

Assuming each feature measurement is independent and Gaussian, we can assume the measurement likelihood for visible landmarks $p(\mathbf{y}|\mathbf{x})$ is given as a product of Gaussian likelihoods over the association set $\mathcal{A}$ multiplied by the likelihoods for the unassociated landmark set $\mathcal{U}$ as follows:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \prod_{(i,j)\in\mathcal{A}} p_{\mathcal{A}}(\mathbf{y}_i|\boldsymbol{\eta}, \mathbf{m}_j) \prod_{j\in\mathcal{U}} p_{\mathcal{U}}(\mathbf{y}_\emptyset|\boldsymbol{\eta}, \mathbf{m}_j) \\ &= \prod_{(i,j)\in\mathcal{A}} \mathcal{N}(\mathbf{y}_i; \mathbf{h}_j(\mathbf{x}), \mathbf{R}) \prod_{j\in\mathcal{U}} \log \frac{1}{|\mathcal{Y}|}, \end{aligned}$$

where $|\mathcal{Y}|$ is the image area in pixel units and $\mathbf{R} \in \mathbb{R}^{2\times 2}$ is the feature measurement covariance matrix.

The association set $\mathcal{A}$ should satisfy *geometric compatibility* between the bundle of all feature measurements and feature predictions[1] in the image. To determine compatibility, it will be helpful to have access to the bundle of predicted feature locations,

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \mathbf{h}_1(\mathbf{x}) \\ \mathbf{h}_2(\mathbf{x}) \\ \vdots \\ \mathbf{h}_{n_L}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{2n_L}. \tag{7}$$

and the bundle Jacobian,

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}} \\ \vdots \\ \frac{\partial \mathbf{h}_{n_L}}{\partial \mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{h}_1}{\partial \mathbf{r}^n_{B/N}} & \frac{\partial \mathbf{h}_1}{\partial \boldsymbol{\Theta}^n_b} & \frac{\partial \mathbf{h}_1}{\partial \mathbf{r}^n_{1/N}} & \mathbf{0}_{2\times 3} & \cdots & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{h}_2}{\partial \mathbf{r}^n_{B/N}} & \frac{\partial \mathbf{h}_2}{\partial \boldsymbol{\Theta}^n_b} & \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{h}_2}{\partial \mathbf{r}^n_{2/N}} & \cdots & \mathbf{0}_{2\times 3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{h}_{n_L}}{\partial \mathbf{r}^n_{B/N}} & \frac{\partial \mathbf{h}_{n_L}}{\partial \boldsymbol{\Theta}^n_b} & \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \cdots & \frac{\partial \mathbf{h}_{n_L}}{\partial \mathbf{r}^n_{n_L/N}} \end{bmatrix} \in \mathbb{R}^{2n_L\times(12+3n_L)}. \tag{8}$$

Tasks

a) Copy or merge the following files from your Lab 8 solution into your Lab 9 working directory:

- `data/camera.xml`
- `src/Camera.h`
- `src/Camera.cpp`
- `src/GaussianBase.hpp`
- `src/GaussianInfo.hpp`

[1] obtained from knowledge of the landmarks and camera pose

- `src/rotation.hpp`
- `src/MeasurementSLAMPointBundle.h`
- `src/MeasurementSLAMPointBundle.cpp`

b) Configure the application for a `Debug` build to ensure all needed files are present and run-time assertions are enabled as follows:

Terminal

```
nerd@basement:~/MCHA4400/lab9$ cmake -G Ninja -B build -DCMAKE_BUILD_TYPE=Debug && cd build
nerd@basement:~/MCHA4400/lab9/build$ ninja
[Build and unit test results]
```

c) Complete the implementation of the template member function `MeasurementSLAMPointBundle::predictFeatureBundle` within `src/MeasurementSLAMPointBundle.h`, which implements (7) and the member function `MeasurementSLAMPointBundle::predictFeatureBundle` within `src/MeasurementSLAMPointBundle.cpp`, which also implements (8).

Hint

Use `MeasurementSLAMPointBundle::predictFeature` to compute each block of (7) and (8) and assemble the result.

2 Geometric compatibility

Consider a marginal likelihood for a candidate feature/landmark pair (i, j) that is well approximated by a Gaussian, i.e.,

$$p(\mathbf{y}_i) = \int_{\mathcal{X}} p_{\mathcal{A}}(\mathbf{y}_i|\boldsymbol{\eta}, \mathbf{m}_j)\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \approx \mathcal{N}(\mathbf{y}_i; \boldsymbol{\mu}_j, \mathbf{P}_j), \tag{9}$$

where $\boldsymbol{\mu}_j$ and $\mathbf{P}_j$ are the marginal mean and covariance, respectively, of the predicted feature corresponding to landmark j in image coordinates and $\mathbf{y}_i$ is the measured feature.

The **individual compatibility** of feature i and landmark j is given by

$$(\mathbf{y}_i - \boldsymbol{\mu}_j)^{\mathsf{T}} \mathbf{P}_j^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_j) \leq \chi^2_2(c). \tag{10}$$

An association set $\mathcal{A}$ satisfies individual compatibility if it satisfies (10) for all $(i, j) \in \mathcal{A}$.

Let $\boldsymbol{\Xi}_j$ be an upper-triangular matrix such that $\boldsymbol{\Xi}_j^{\mathsf{T}} \boldsymbol{\Xi}_j = \mathbf{P}_j^{-1}$ and let $\boldsymbol{\nu}_j = \boldsymbol{\Xi}_j \boldsymbol{\mu}_j$. Then, we can test the individual compatibility of feature i with landmark j as follows:

$$\mathbf{w}_{i,j}^{\mathsf{T}} \mathbf{w}_{i,j} \leq \chi^2_2(c), \tag{11}$$

where $\mathbf{w}_{i,j} = \boldsymbol{\Xi}_j \mathbf{y}_i - \boldsymbol{\nu}_j$.

Consider a marginal likelihood for the association set $\mathcal{A}$ that is well-approximated by a Gaussian, i.e.,

$$\mathbf{p}(\mathbf{y}_{\mathcal{A}}) \approx \mathcal{N}(\mathbf{y}_{\mathcal{A}}; \boldsymbol{\mu}_{\mathcal{A}}, \mathbf{P}_{\mathcal{A}}), \tag{12}$$

where $\boldsymbol{\mu}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{A}}$ are the marginal mean and covariance, respectively, of the predicted feature bundle of all associated landmarks in image coordinates and $\mathbf{y}_{\mathcal{A}}$ is the measured feature bundle of all associated landmarks. Then, the association set $\mathcal{A}$ satisfies **joint compatibility** if

$$(\mathbf{y}_{\mathcal{A}} - \boldsymbol{\mu}_{\mathcal{A}})^{\mathsf{T}} \mathbf{P}_{\mathcal{A}}^{-1} (\mathbf{y}_{\mathcal{A}} - \boldsymbol{\mu}_{\mathcal{A}}) \leq \chi^2_{2|\mathcal{A}|}(c), \tag{13}$$

where $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$ (number of associations).

Let $\boldsymbol{\Xi}_{\mathcal{A}}$ be an upper-triangular matrix such that $\boldsymbol{\Xi}_{\mathcal{A}}^{\mathsf{T}} \boldsymbol{\Xi}_{\mathcal{A}} = \mathbf{P}_{\mathcal{A}}^{-1}$ and let $\boldsymbol{\nu}_{\mathcal{A}} = \boldsymbol{\Xi}_{\mathcal{A}} \boldsymbol{\mu}_{\mathcal{A}}$. Then, we can test the joint compatibility of $\mathcal{A}$ as follows:

$$\mathbf{w}_{\mathcal{A}}^{\mathsf{T}} \mathbf{w}_{\mathcal{A}} \leq \chi^2_{2|\mathcal{A}|}(c), \tag{14}$$

where $\mathbf{w}_{\mathcal{A}} = \boldsymbol{\Xi}_{\mathcal{A}} \mathbf{y}_{\mathcal{A}} - \boldsymbol{\nu}_{\mathcal{A}}$.

An association set $\mathcal{A}$ satisfies **geometric compatibility** if it satisfies both individual compatibility (10) for all elements of $\mathcal{A}$ and joint compatibility (13) for $\mathcal{A}$.

The total **surprisal** of the measurement bundle is given by

$$\begin{aligned} s(\mathbf{y}) &= -\log p(\mathbf{y}) \\ &= -\log \int_{\mathcal{X}} \prod_{(i,j)\in\mathcal{A}} p_{\mathcal{A}}(\mathbf{y}_i | \boldsymbol{\eta}, \mathbf{m}_j) \prod_{j\in\mathcal{U}} p_{\mathcal{U}}(\mathbf{y}_{\emptyset} | \boldsymbol{\eta}, \mathbf{m}_j)\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}, \end{aligned}$$

where

$$p_{\mathcal{U}}(\mathbf{y}_{\emptyset} | \boldsymbol{\eta}, \mathbf{m}_j) = \mathcal{U}(\mathbf{y}; \mathcal{Y}) = \begin{cases} \frac{1}{|\mathcal{Y}|} & \text{if } \mathbf{y} \in \mathcal{Y}, \\ 0 & \text{if } \mathbf{y} \notin \mathcal{Y}, \end{cases}$$

is the measurement likelihood if landmark j is unassociated with a feature, where $\mathcal{Y}$ is the image domain. Since this likelihood is independent of the state, we can move these terms outside the integral,

$$\begin{aligned} s(\mathbf{y}) &= -\log \int_{\mathcal{X}} \prod_{(i,j)\in\mathcal{A}} p_{\mathcal{A}}(\mathbf{y}_i | \boldsymbol{\eta}, \mathbf{m}_j) \prod_{j\in\mathcal{U}} \mathcal{U}(\mathbf{y}; \mathcal{Y})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= -\log \prod_{j\in\mathcal{U}} \mathcal{U}(\mathbf{y}; \mathcal{Y}) \int_{\mathcal{X}} \prod_{(i,j)\in\mathcal{A}} p_{\mathcal{A}}(\mathbf{y}_i | \boldsymbol{\eta}, \mathbf{m}_j)\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= -\sum_{j\in\mathcal{U}} \log \mathcal{U}(\mathbf{y}; \mathcal{Y}) - \log \int_{\mathcal{X}} p(\mathbf{y}_{\mathcal{A}} | \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &= -\sum_{j\in\mathcal{U}} \log \frac{1}{|\mathcal{Y}|} - \log p(\mathbf{y}_{\mathcal{A}}) \\ &= |\mathcal{U}| \log |\mathcal{Y}| + s(\mathbf{y}_{\mathcal{A}}), \end{aligned}$$

where $|\mathcal{U}|$ is the number of landmarks expected to be visible that weren't associated with any features.

Tasks

a) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/individualCompatibility.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementations provided.

b) Within `src/association_util.cpp` complete the implementation of `individualCompatibility`, which checks the compatibility of image feature i with landmark j.

> **Hint**
> Use `GaussianInfo::isWithinConfidenceRegion` to evaluate (11).

c) Run `ninja` and ensure all unit tests within `test/src/individualCompatibility.cpp` pass.

d) Remove the `doctest::skip` decorator from each `SCENARIO` in `test/src/jointCompatibility.cpp` to enable these unit tests and run `ninja` to verify that they *fail* due to the incomplete implementations provided.

e) Within `src/association_util.cpp` complete the implementation of `jointCompatibility`, which checks the joint compatibility of a given association set and computes the total surprisal[2] of the measurements. Note that the measured feature set $\mathbf{Y}$ is stored as the following *matrix*:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \end{bmatrix} \in \mathbb{R}^{2\times m}. \tag{15}$$

> **Hint**
> Use `GaussianInfo::marginal` to extract the marginal distribution of the predicted feature bundle of the associated landmarks from the provided joint bundle distribution and then use `GaussianInfo::isWithinConfidenceRegion` to evaluate (14). Use `GaussianInfo::log` to compute the surprisal of the associated landmarks and add this to the surprisal of the unassociated landmarks.

f) Run `ninja` and ensure all unit tests within `test/src/jointCompatibility.cpp` pass.

3 Geometric data association

Use the surprisal nearest neighbours (SNN) data association algorithm to produce results similar to Figure 1.

Tasks

a) In `src/image_features.cpp` complete the implementation of `detectFeatures`, which returns the location and feature scores within a `std::vector<PointFeature>`.

[2] i.e., the surprisal for unassociated landmarks plus the surprisal for associated landmarks.

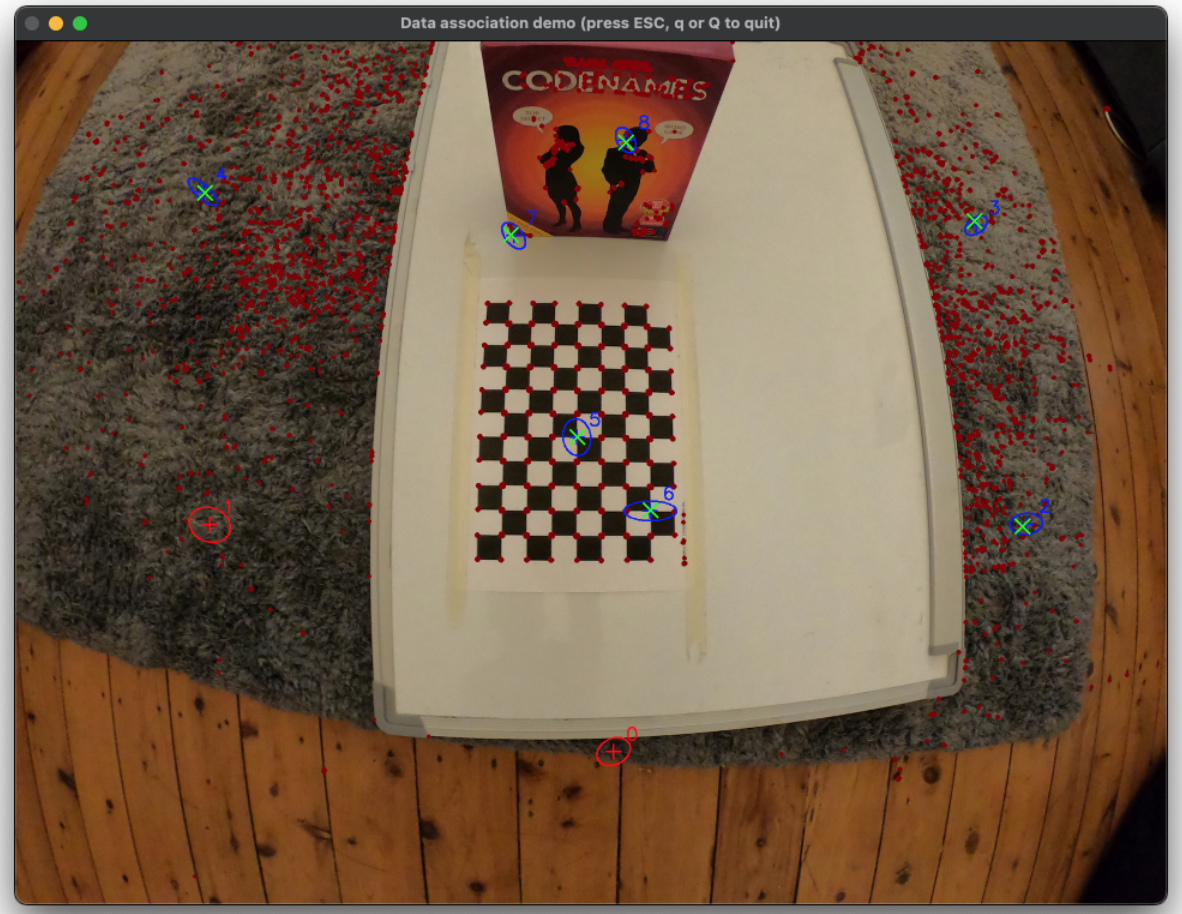

Figure 1: Example of geometric data association in self-similar texture environment. The predicted feature locations of previously established landmarks are shown ('+' markers) with the 3σ confidence regions (ellipses). Blue indicates landmarks associated with a feature. Bright red indicates landmarks not associated with any feature. Dark red dots indicate features not associated with any landmark. Green '×' markers indicate features associated with a landmark.

> **Tip**
>
> A point feature detector, such as the Harris, Shi & Tomasi or FAST corner detector is recommended.

b) Review the implementation of `associationDemo` within `src/association_demo.cpp` and `snn` within `src/association_util.cpp`, and familiarise yourself with the data association algorithm and its components. These functions are complete as provided, but use many of the functions you have implemented in earlier parts of this lab.

c) Build and run the application and you should see console output similar to the following:

Terminal

```
nerd@basement:~/MCHA4400/lab9/build$ ninja && ./lab9 ../data/config.xml
[snip]
??? features found in GOPR9348.JPG
Landmark 0 is expected to be within camera FOV
Landmark 1 is expected to be within camera FOV
Landmark 2 is expected to be within camera FOV
Landmark 3 is expected to be within camera FOV
Landmark 4 is expected to be within camera FOV
Landmark 5 is expected to be within camera FOV
Landmark 6 is expected to be within camera FOV
Landmark 7 is expected to be within camera FOV
Landmark 8 is expected to be within camera FOV
No feature associated with landmark 0.
No feature associated with landmark 1.
Feature 9432 located at [1679  810] in image matches landmark 2.
Feature 9288 located at [1599  301] in image matches landmark 3.
Feature 9707 located at [316 254] in image matches landmark 4.
Feature 585 located at [937 661] in image matches landmark 5.
Feature 1116 located at [1059  783] in image matches landmark 6.
Feature 6858 located at [826 324] in image matches landmark 7.
Feature 3366 located at [1018  170] in image matches landmark 8.
[Display cv::imshow window and wait for keypress]
```

and a plot similar to Figure 1.

d) Once you are happy with the results, switch to a `Release` build and compare the time taken to perform geometric data association against a `Debug` build.