Robotic Vision Summer School
Fusing map data

February 4, 2020

To identify the best estimate of the location of the animals, there are two key analysis problems that must be resolved. The first is to register the SLAM maps obtained by the different robots. That is to align the maps generated by different robots so that they share a common reference and the information about the animals can be combined. The second is to estimate the position of an animal from multiple independent measurements of bearings taken at different points in the terrain. Both these problems can be solved using stochastic data fusion techniques.

1 Notation

1.1 Kroneker products

For two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{p \times q}$ then

$$A \otimes B = \begin{pmatrix} A_{11}B & \cdots & A_{1m}B \\ \vdots & & \vdots \\ A_{n1}B & \cdots & A_{nm}B \end{pmatrix} \in \mathbb{R}^{np \times mq}$$

1.2 SLAM configuration state

The state of a differential drive robot is modelled as position (x, y) and orientation θ. That is the state $\xi = (x, y, \theta) \in \mathbb{R}^3$ where we unwrap the angle variable θ from the circle $[0, 2\pi)$ to the real line. This state representation is nice for formulating the extended Kalman filter but poor for computing geometric transformations.

The geometric pose of a robot is represented[1] as an element of $P \in \mathbf{SE}(2)$ Fig. 1. This pose representation the position and orientation of the robot as a rigid transformation from an fixed reference frame $\{0\}$ to the robot body-fixed frame of reference $\{P\}$.

[1] Using $\mathbf{SE}(3)$ to represent the pose is not formally correct, however, it is a convenient and accepted notation where the rotation columns encode the directions of the axes of the robot body-fixed frame of reference while translation column encodes the robot position.

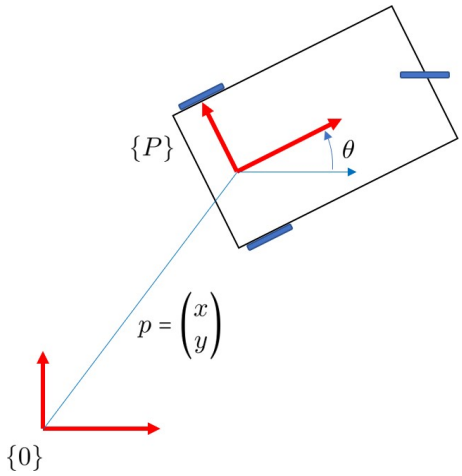

Figure 1: Geometric pose of a robot moving on a plane.

An element of $\mathbf{SE}(2)$ is written

$$P = \begin{pmatrix} R_P & x_P \\ 0 & 1 \end{pmatrix} \in \mathbf{SE}(2)$$

with $R_P \in \mathbf{SO}(2)$ a rotation and $x_P \in \mathbb{R}^2$ its location. The rotation matrix is written

$$R_P = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

where θ is the relative orientation angle of the robot.[2] Thus, the geometric pose P is constructed from the robot state (x, y, θ) by

$$P = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

The inverse of a matrix $A \in \mathbf{SE}(2)$ is

$$A^{-1} = \begin{pmatrix} R_A^\top & -R_A^\top x_A \\ 0 & 1 \end{pmatrix}.$$

That is $AA^{-1} = I_3 = A^{-1}A$.

Landmarks are 2D points on the plane $p_i \in \mathbb{R}^2$ for $i = 1, \ldots, n$ represented with respect to the reference frame $\{0\}$. A typical state configuration including geometric robot pose and landmark points are shown in Figure 2

[2]This expression for rotation uses the robotics convention - representing the robots state from the perspective of a reference frame. The Computer Vision community often uses the inverse convention, representing an object viewed in the world from the perspective of the camera.

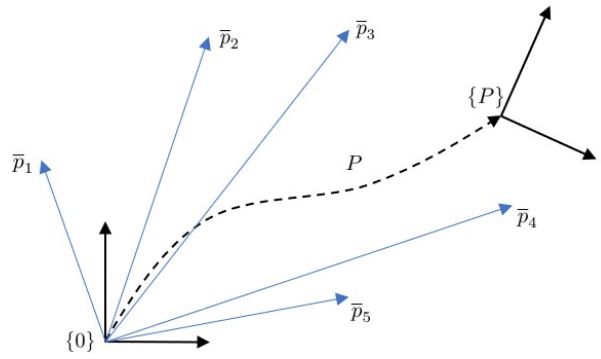

Figure 2: The state of a SLAM problem with 5 landmarks.

1.3 Rigid body transformations

A rigid-body transformation is a transformation of space that preserves distances and angles between all points. Rigid-body transformations consist of rotations and translations and a general rigid-body transformation of $\mathbb{R}^2$ is an element of $\mathbf{SE}(2)$. Landmarks transform under rigid body transformation by $A \in \mathbf{SE}(2)$ by

$$p \mapsto R_A p + x_A$$

The algebra of homogeneous coordinates are often used to simplify this notation. Let the homogeneous vector for $p \in \mathbb{R}^2$ be defined as

$$\overline{p} = \begin{pmatrix} p \\ 1 \end{pmatrix} \hookrightarrow \mathbb{R}^3$$

where we use the hook right arrow to indicate the embedding of homogeneous coordinates into $\mathbb{R}^3$, the vector has three entries. With this algebra one has

$$\overline{R_A p + x_A} = \begin{pmatrix} R_A p + x_A \\ 1 \end{pmatrix} = \begin{pmatrix} R_A & x_A \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix} = A\overline{p}$$

or $\overline{p} \mapsto A\overline{p}$.

The matrix $P \in \mathbf{SE}(3)$ representing the position and orientation of a robot will also transform by a rigid-body transformation $A \in \mathbf{SE}(2)$ by

$$P \mapsto AP.$$

That is the old P is translated and rotated by A to a new pose AP.

1.4 Probability

The multi-variate Gaussian distribution in a random variable $x \in \mathbb{R}^N$ is given by

$$N_{(\underline{x},\Sigma)}(x) = \frac{1}{(2\pi)^{\frac{N}{2}}} \frac{1}{(\det(\Sigma))^{\frac{N}{2}}} \exp\left(-(x-\underline{x})^\top \Sigma^{-1}(x-\underline{x})\right).$$

where[3]

$$\text{mean} = \underline{x} \in \mathbb{R}^N$$
$$\text{variance} = \Sigma \in \mathbb{R}^{N \times N}$$

and $\Sigma^\top = \Sigma$ is a positive definite matrix. Where the random variable is clear we will write $N_{(\underline{x},\Sigma)}(x) = N(\underline{x}, \Sigma)$. Where the parameters are clear or not specified explicitly we will write just $N(x)$ for a Gaussian distribution. In particular, in the case where a random variable is drawn from a Gaussian we write $x \sim N(\underline{x}, \Sigma)$. We also write

$$\text{Var}(x) = \Sigma \in \mathbb{R}^{N \times N}$$

for the variance of x.

Let $x = (y, z)$ for $y \in \mathbb{R}^{N_1}$, $z \in \mathbb{R}^{N_2}$ and $N_1 + N_2 = N$ a partition of the state. One has

$$\Sigma = \text{Var}(x) = \begin{pmatrix} \text{Var}(y) & \text{Var}(y,z) \\ \text{Var}(z,y) & \text{Var}(z) \end{pmatrix}$$

where the $\text{Cov}(y, z) \in \mathbb{R}^{N_1, N_2}$ is the covariance between two variables y and z. One has $\text{Cov}(y, z) = \text{Cov}(z, y)^\top$

The marginal $y \sim N(y)$, or the marginalisation of $x \sim N_{(\underline{x},\Sigma)}(x)$ with respect to z, is the distribution of the partial variable y averaged over all possible values the variable z could take. The marginal y is distributed as

$$y \sim N_{(\underline{y},\text{Var}(y))}(y)$$

which is the top left $\mathbb{R}^{N_1 \times N_1}$ block of $\Sigma = \text{Var}(x)$. Thus, the normal distribution parameters of the marginal y, or the marginalisation of x with respect to z, is obtained by partitioning the mean and variance of x.

2 Fusing two maps

The geometric robot pose $P(t) \in \mathbf{SE}(2)$ and the landmark points $\overline{p}_i \in \mathbb{R}^2$ are represented with respect to a reference frame $\{0\}$ that is implicit in the coordinate formulation. This reference frame, however, has no physical ground truth in the SLAM problem. Indeed, if $(P, \overline{p}_1, \ldots, \overline{p}_n)$ are consistent with your measurements, then for any rigid-body transformation $A \in \mathbf{SE}(2)$ one has that $(AP(t), A\overline{p}_1, \ldots, A\overline{p}_n)$ are also consistent with the measurements. That is translating and rotating the robot and the landmark points by the same transformation is consistent with all the observations. This property is termed an *invariance* of the SLAM problem, and the transformations $A \in \mathbf{SE}(2)$ are known as *gauge transformation*.

[3]Here we use the unusual notation $\underline{x}$ for mean since we reserve the overbar notation for homogeneous coordinates.

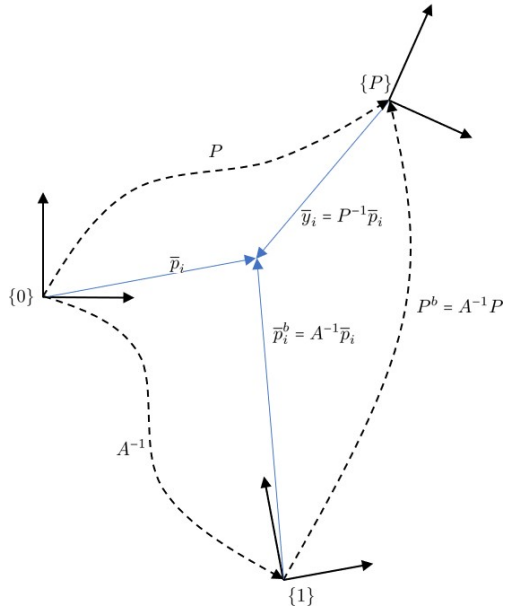

Figure 3: A gauge transform is visualised as a transformation of the reference frame that changes coordinates but doesn't move the robot or landmark points. The rigid body transformation $A : \{1\} \mapsto \{0\}$ and maps $\overline{p}_i^b \mapsto A\overline{p}_i^b = \overline{p}_i$ and $P^b \mapsto AP^b = P$.

Although gauge transforms do not change the relative information contained in a SLAM configuration estimate, they do change the coordinates of a configuration. Thus, for two configurations $(P, \overline{p}_1, \ldots, \overline{p}_n)$ and $(AP(t), A\overline{p}_1, \ldots, A\overline{p}_n)$ that represent the same physical configuration of robot and landmark points, $P \neq AP$ and $\overline{p}_i \neq A\overline{p}_i$. A useful visualisation of this difference is two consider two different coordinate representations of the same physical configuration of robot and landmarks. The different coordinates can be related to each other by applying a gauge transformation to the reference frame by A^{-1} as shown in Figure 3.

This relationship is key to understand when fusing maps from two different robots. When two maps are produced by different robots, the two reference frames implicit in the coordinates used by the robots will be different Fig. 4. The coordinates of the map points estimated by the SLAM algorithm will be different. To combine the data from both robots one needs to *register* the two maps so that they share the same reference frame. Registering the maps refers to computing the relative gauge transform between the reference frames and transforming the coordinates of both maps into a single shared reference frame. In Figure 4, the reference frame from robot a was chosen without loss of generality as common frame, and the gauge transformation A computed that

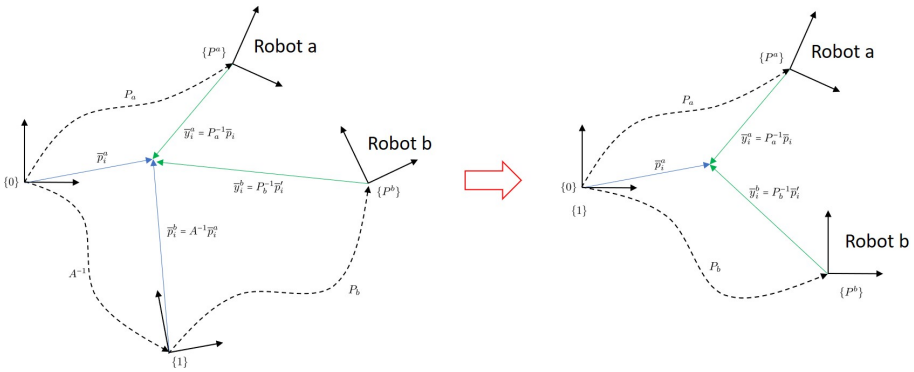

Figure 4: A figure demonstrating map registration for two robot SLAM. The maps on the left are of the same physical points but derived from SLAM algorithms running on different robots with different measurements. The coordinates of the SLAM solution differ by a rigid-transformation A that is visualised as a gauge transformation A^{-1} of the reference frame $\{0\} \to \{1\}$. The right hand side shows the situation after the rigid body transformation $A \in \mathbf{SE}(2)$ is applied to the reference $\{1\}$, as the SLAM configuration $(P_b, \overline{p}_i^b)$. In the registered coordinates the map points correspond. The robot poses are different since the two robots were not collocated.

transforms $\{1\}$ to $\{0\}$. This gauge transform is applied to the SLAM solution $(P^b, \overline{p}_i^b) \mapsto (AP^b, A\overline{p}_i^b)$. In the new registered coordinates then $\overline{p}_i^a = A\overline{p}_i^b$ at least up to noise in the estimates. The robot poses $AP^b \neq P^a$ of course since the two robots were not co-located.

2.1 Registering two maps without noise

Let us begin with the case where there is no noise in either map. That is consider two maps $\{\overline{p}_1^a, \ldots, \overline{p}_n^a\}$ and $\{\overline{p}_1^b, \ldots, \overline{p}_n^b\}$ and ask the question: Find $A \in \mathbf{SE}(2)$ such that

$$A\overline{p}_i^b = \overline{p}_i^a, \text{ for } i = 1, \ldots, n. \tag{2}$$

This problem is linear in the unknown $A \in \mathbf{SE}(2)$, although the nonlinearity of the Lie-group $\mathbf{SE}(2)$ does complicate matters.

Expressing (2) as

$$R_A p_i^b + x_A - p_i^a = 0 \tag{3}$$

then it follows that $x_A = p_i^a - R_A p_i^b$ for all $i = 1, \ldots, n$. Since we know that there will be noise even though at the moment we are not modelling it, we will also

average over all landmarks

$$x_A = \frac{1}{n}\sum_{j=1}^{n}(p_j^a - R_A p_j^b) = \frac{1}{n}\sum_{j=1}^{n} p_j^a - R_A \frac{1}{n}\sum_{j=1}^{n} p_j^b = p_{\text{av}}^a - R_A p_{\text{av}}^b \tag{4}$$

where

$$p_{\text{av}} := \frac{1}{n}\sum_{j=1}^{n} p_j$$

since we have used the overbar notation for homogeneous coordinates.

Substituting (4) into (3) one obtains

$$R_A(p_i^b - p_{\text{av}}^b) = (p_i^a - p_{\text{av}}^a), \qquad \text{for } i = 1, \ldots, n. \tag{5}$$

Define data matrices

$$M_a = \begin{pmatrix}(p_1^a - p_{\text{av}}^a) & \cdots & (p_n^a - p_{\text{av}}^a)\end{pmatrix} \in \mathbb{R}^{2\times n} \tag{6}$$

$$M_b = \begin{pmatrix}(p_1^b - p_{\text{av}}^b) & \cdots & (p_n^b - p_{\text{av}}^b)\end{pmatrix} \in \mathbb{R}^{2\times n} \tag{7}$$

then

$$R_A M_b = M_a \tag{8}$$

As long as $\text{rank}(M_b) = 2$ then this equation has a unique solution

$$R_A = M_a M_b^\top (M_b^\top M_b)^{-1} \tag{9}$$

If there is no noise in the data, then this matrix will be orthogonal $R_A \in \mathbf{SO}(2)$.

In practice, it is always a good idea to make the resulting matrix orthogonal. Take the SVD of the data

$$M_a M_b^\top (M_b^\top M_b)^{-1} = U^\top \Lambda V$$

for $U, V \in \mathbf{SO}(2)$ and Λ diagonal. If the data is ideal then $\Lambda = I_2$ is the identity matrix. If not then make it so, that is, set $R_A = U^\top V$. Of course you should check that Λ is close to the identity matrix. If Λ is note close to identity, this the two maps do not represent the same physical landmark points and this flags an error.

Algorithm 1: Registration of ideal maps.

Choose e_{thresh} for error criteria ;
Compute x_A (4) ;
Compute M_a and M_b from (6) and (7) ;
Compute the SVD $U^\top \Lambda V = M_a M_b^\top (M_b^\top M_b)^{-1}$;
if $\|\Lambda - I_2\|^2 > e_{\text{thresh}}$ **then**
    Error: "maps are not consistent" ;
    return
end
Set $R_A = U^\top V$;
Result: (R_A, x_A)

2.2 Registration of a noisy map to known data

In §2.1 we discussed registering two maps assuming ideal data. A real robot never generates a perfect map. Indeed, the standard SLAM algorithms generate both an estimate and a covariance, or estimate of uncertainty for the map. Indeed, these algorithms are fundamentally stochastic in nature, and the correct interpretation is that they generate an information state for a landmark variable. That is the estimate of the landmark variable is the Gaussian probability distribution with mean (the estimate) and variance (the estimate uncertainty). In order to register noisy data to a ground truth the ideal equations in §2.1 do not hold and a stochastic approach is appropriate.

Formulating the maximum likelihood cost: Consider a known landmark configuration $\{p_1^a, \dots, p_n^a\}$ and a set of noisy data generated by a SLAM algorithm $\{\hat{p}_1^b, \dots, \hat{p}_n^b\}$

$$\hat{p}_i^b \sim N(p_i^b, \Sigma_i^b).$$

where $N(p_i^b, \Sigma_i^b)$ is the Gaussian with mean[4] p_i^b and (positive definite) variance $\Sigma_i^b \in \mathbb{R}^{2\times 2}$. This is typically written as an explicit *generative noise model*

$$\hat{p}_i^b = p_i^b + \nu_i^b, \quad \nu_i^b \sim N(0, \Sigma_i^b). \tag{10}$$

The underlying problem is the same: Find $A \in \mathbf{SE}(2)$ such that

$$A\overline{p}_i^b \approx \overline{p}_i^a.$$

however now we do not have p_i^b directly and we need to consider some sort of approximate solution. To begin with we must define what sort of approximation is appropriate. The most common approach is to use maximum likelihood.

Applying A^{-1} to p_i^a is equivalent to applying A to p_i^b. Thus, from (10) one can write

$$R_A^\top p_i^a - R_A^\top x_A - \hat{p}_i^b = -\nu_i^b \sim N(0, \Sigma_i^b)$$

That is $R_A^\top p_i^a - R_A^\top x_A - \hat{p}_i^b \sim N(0, \Sigma_i^b)$. Note that the left hand side depends on unknown parameters R_A and x_A. The *likelihood* of a data sample, in this case the evaluation of $R_A^\top p_i^a - R_A^\top x_A - \hat{p}_i^b$ is the evaluation of the probability distribution $N(0, \Sigma_i^b)$. The *maximum likelihood* solution is obtained by choosing the parameters R_A and x_A to maximize the likelihood of the data!

Now, for a 2 dimensional Gaussian distribution

$$N_{(0,\Sigma_i^b)}(\zeta) = \frac{1}{2\pi \det(\Sigma_i^b)} \exp\left(-\zeta^\top (\Sigma_i^b)^{-1} \zeta\right) \tag{11}$$

where $\zeta \in \mathbb{R}^2$ is just a placeholder variable here. Maximising (11) is difficult due to the exponential non-linearity in the Gaussian distribution function. However, the log function is monotonic increasing to maximising $\log N_{(0,\Sigma_i^b)}(\nu_i^b)$ is the

[4]We will assume that the SLAM algorithm is unbiased, that is that the mean of the Gaussian from which the estimate is drawn is the true map.

same as maximising $N_{(0,\Sigma_i^b)}(\nu_i^b)$. Moreover, multiplying by -1 makes the problem a minimization, which will match classical least squares methods. Thus, we consider minimizing $-\log N_{(0,\Sigma_i^b)}(\nu_i^b)$ over parameters R_A and x_A. That is

$$\begin{aligned}(\hat{R}_A, \hat{x}_A) &= \underset{(\hat{R},x)}{\operatorname{argmin}} \left((\nu_i^b)^\top (\Sigma_i^b)^{-1} (\nu_i^b) + \log(2\pi \det(\Sigma_i^b)) \right) \\ &= \underset{(R_A,x_A)}{\operatorname{argmin}} (\nu_i^b)^\top (\Sigma_i^b)^{-1} (\nu_i^b)\end{aligned}$$

where $\nu_i^b = R_A^\top p_i^a - R_A^\top x_A - \hat{p}_i^b$. Note that the term $\log(2\pi \det(\Sigma_i^b))$ can be discarded from the optimisation since it does not depend on the parameters $(\hat{R}, x)$. Define the scaled least squares norm to be

$$\|\zeta\|^2_{\Sigma_i^b} := \zeta^\top (\Sigma_i^b)^{-1} \zeta$$

and note that this problem can be written

$$\hat{R}_A, \hat{x}_A = \underset{(R_A,x_A)}{\operatorname{argmin}} \|R_A^\top p_i^a - R_A^\top x_A - \hat{p}_i^b\|^2_{\Sigma_i^b}.$$

This is a least squares problem with respect to a norm that is scaled by the covariance Σ_i^b.

The above discussion leads to a nice formulation for a single data point, in this case a single map estimate of a landmark. In fact, generalising to multiple landmarks is straightforward. The output of a SLAM system is a vector state that combines all the variables of the system, typically

$$\xi = (x, y, \theta, p_1^x, p_1^y, \ldots, p_n^x, p_n^y) \in \mathbb{R}^{3+2n}$$

The covariance estimate is one large matrix $\Sigma \in \mathbb{R}^{(3+2n)\times(3+2n)}$. That is the state $\xi \sim N(\hat{\xi}, \Sigma)$ is a multi-variate Gaussian in $3+2n$ dimensions. It is important to maintain the full state structure since the estimates of the landmarks and robot state are highly correlated. The map registration problem depends only on the landmark variables of the state and the robot pose variables can be *marginalised* out. For a Gaussian distribution, the marginal obtained by integrating out for the robot state is just the map part of the state $m = (p_1, \ldots, p_n) \in \mathbb{R}^{2n}$ corresponding to the n landmarks along with the $2n \times 2n$ partition of Σ^m corresponding to the map. The only tricky part is acting on the measurements with R_A and x_A as this must be done landmark by landmark. This can be vectorised by forming the following Kroneker products

$$I_n \otimes R_A = \begin{pmatrix} R_A & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_A \end{pmatrix}, \quad \mathbf{1}_n \otimes x_A = \begin{pmatrix} x_A \\ \vdots \\ x_A \end{pmatrix}$$

where I_n is the $n \times n$ identity and $\mathbf{1}_n$ is the n-vector with ones in each entry. Then maximum likelihood solution can be written as a function of the full map coordinates as a nonlinear least squares problem

$$(\hat{R}_A, \hat{x}_A) = \underset{(R_A,x_A)}{\operatorname{argmin}} \|(I_n \otimes R_A)^\top \mathbf{m}_a - (I_n \otimes R_A)^\top (\mathbf{1}_n \otimes x_A) - \hat{\mathbf{m}}_b\|^2_{\Sigma_b}. \tag{12}$$

where $\mathbf{m}_a = (p_1^a, \ldots, p_n^a)$ and $\hat{\mathbf{m}}_b = (\hat{p}_1^b, \ldots, \hat{p}_n^b)$.

Solving the least squares problem: The next question is how to solve (12). This is no longer a simple algebraic solution of a linear system of equations. The least squares problem, when weighted by a general Σ_b matrix is quadratic in the variables R_A and x_A and in addition, $R_A \in \mathbf{SO}(2)$ is a rotation matrix. This problem must be solved iteratively by an optimisation algorithm. The standard tool used is a *Levenberg–Marquardt algorithm*, this is a modification of the Gauss–Newton algorithm[5] modified specifically for cost functions of the nonlinear least squares form (12). The Levenberg-Marquardt algorithm is based on linearising the argument of the nonlinear least squares cost around a point, and then solving the resulting quadratic least squares problem in the linearised variable.

The first step is to locally parameterise the rotation matrix R_A. Let[6]

$$\mathfrak{e} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Then the matrix exponential of $\mathfrak{e}$ (skew symmetric) is a rotation matrix $\exp(\theta\mathfrak{e}) \in \mathbf{SO}(2)$. Indeed,

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} = \exp(\theta\mathfrak{e})$$

We only need to parameterize R_A locally around a previous iterate of the optimization algorithm $R_A(j)$ so we write

$$R_A = R_A(j)\exp(\delta\mathfrak{e})$$

for $\delta \in \mathbb{R}$ a small angle. There is still the nonlinear dependence on δ through the exponential function. Thus, finally we linearize around $\delta = 0$ to get

$$R_A \approx R_A(j)(I_2 + \delta\mathfrak{e}). \tag{13}$$

Note that right-hand side is not actually an orthogonal matrix, however, it is this linearisation that is required to compute the Levenberg-Marquardt update. Similarly, we write a local linear approximation for x_A around an estimate $x_A(j)$ by

$$x_A = x_A(j) + \epsilon. \tag{14}$$

This local model is already linear and does not require linearisation.

The solution to the nonlinear least squares problem (12) can now be written locally around an estimate $(R_A(j), x_A(j))$ as

$$(\hat{R}_A, \hat{x}_A) = (R_A(j)\exp(\mathfrak{e}\delta_j), x_A(j) + \epsilon_j) \tag{15}$$

[5]It also has a trust region modification to improve its basin of attraction that will not be discussed in these notes.

[6]The matrix $\mathfrak{e}$ is the generator for the matrix representation of the Lie-algebra $\mathfrak{so}(2)$.

where

$$(\delta_j,\epsilon_j) = \underset{(\delta,\epsilon)}{\operatorname{argmin}} \|(I_n \otimes R_A(j))^\top \mathbf{m}_a + (I_n \otimes R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) - \hat{\mathbf{m}}_b - \delta(I_n \otimes \mathfrak{e}R_A(j))^\top \mathbf{m}_a - \delta(I_n \otimes \mathfrak{e}R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) + \mathbf{1}_n \otimes R_A^{\otimes}(j)^\top \epsilon\|^2_{\Sigma_b}. \tag{16}$$

This equation is obtained by substituting (13) and (14) into (12) and cancelling the quadratic $\epsilon\delta$ term. Although equation (16) looks horrible it is in fact very simple. Define data structures

$$a(j) = (I_n \otimes R_A(j))^\top \mathbf{m}_a + (I_n \otimes R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) - \hat{\mathbf{m}}_b \in \mathbb{R}^{2n} \tag{17}$$

$$b(j) = (I_n \otimes \mathfrak{e}R_A(j))^\top \mathbf{m}_a + (I_n \otimes \mathfrak{e}R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) \in \mathbb{R}^{2n} \tag{18}$$

$$C(j) = \mathbf{1}_n \otimes R_A(j)^\top \in \mathbb{R}^{2n\times 2} \tag{19}$$

Then the nonlinear least squares problem at step k of the iteration can be written

$$(\delta_j, \epsilon_j) = \underset{(\delta,\epsilon)}{\operatorname{argmin}} \|a(j) - \delta b(j) + C(j)\epsilon\|^2_{\Sigma_b}. \tag{20}$$

The solution of this is just least squares in the two linear variables δ and ϵ with respect to the scaled norm $\|\cdot\|^2_{\Sigma_b}$.

Algorithm 2: Levenberg Marquardt for map registration of noisy data to known landmarks

Fix a required accuracy a_{thresh} ;
Choose $(R_A(1), x_A(1))$;
while $\|a(j)\|^2_{\Sigma_b} > a^2_{thresh}$ **do**
  Compute $a(j)$, $b(j)$ and $c(j)$ from (17), (18), (19) ;
  Solve (20) for (ϵ_j, δ_j). ;
  Compute $(R_A(k+1), x_A(k+1)) = (R_A(j)\exp(\mathfrak{e}\delta_j), x_A(j) + \epsilon_j)$;
end
Set $(\hat{R}_A, \hat{x}_A) = (R_A(j), x_A(j))$;
Result: $(\hat{R}_A, \hat{x}_A)$

The Levenberg-Marquardt algorithm solves the (20) recursively. Each iteration finds the exact optimum of the linearised problem and then subsequent linearisation is undertaken closer to the true optimum and contains information to improve the approximation. The iteration is terminated when the cost of the estimated solution

$$\|a(j)\|^2_{\Sigma_b} = \|(I_n \otimes R_A(j))^\top \mathbf{m}_a + (I_n \otimes R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) - \hat{\mathbf{m}}_b\|^2_{\Sigma_b}$$

is small, less than a fixed threshold a^2_{thresh}.

Of course, like any locally convergent algorithm, the initial guess $(R_A(1), x_A(1))$ needs to be pretty good before you start the algorithm, or it will diverge. A good choice for $(R_A(1), x_A(1))$ is to solve the ideal case, using the ideal case

algorithm 1 then use this as a starting point. The ideal case will always compute something that is not too bad, although you may need to set the error threshold fairly high to start with and look at deleting outliers if the maps are not consistent.

2.3 Registering and fusing two noisy maps

> There is data you can trust and data you should distrust but 'true data' is an oxymoron.[7]

The reality in robotics is that the concept of ground truth is nonsensical. All estimates are noisy in some sense, even if they are what you may have thought was ground truth. In the case of fusing two maps generated by separate robots, both maps come with explicit estimates of their uncertainty and the problem is one of both registering and fusing the stochastic information.

Formulating the maximum likelihood cost: Consider two noisy maps generated by robots a) and b). That is $\hat{m}_a = (\hat{p}_1^a, \dots, \hat{p}_n^a)$ and $\hat{m}_b = (\hat{p}_1^b, \dots, \hat{p}_n^b)$

$$\hat{m}_a \sim N(\mathbf{m}_a, \Sigma_a), \quad \hat{m}_b \sim N(\mathbf{m}_b, \Sigma_b).$$

Here we jump straight to representing the map as $\mathbb{R}^{2n}$ objects with correlated variance. The underlying problem is the same: Find $A \in \mathbf{SE}(2)$ such that

$$(I_n \otimes R_A)\mathbf{m}_b + \mathbf{1}_n \otimes x_A \approx \mathbf{m}_a$$

using the Kroneker notation for the vector structure of the full map. The ideal map coordinates $\mathbf{m}_a$ and $\mathbf{m}_b$ are not available, and an exact solution is impossible. We must formulate a criteria to define which approximation for the map registration parameters (R_A, x_A) are appropriate, but also find an approximate solution for the true map $\hat{\mathbf{m}}_\star$ under some appropriate criteria.

Since there is not ground truth, we must start by choosing a reference. Choose robot a) as the reference system. That is, choose the reference in which we will fuse the data to be the arbitrary reference associated with robot a) SLAM solution. Which is to say the "true" map is $\mathbf{m}_\star = \mathbf{m}_a$.

It follows that $\hat{\mathbf{m}}_a \sim N(\mathbf{m}_\star, \Sigma_a)$ and we can consider a coupled set of equations[8]

$$\begin{aligned}(I_n \otimes R_A)\mathbf{m}_b + \mathbf{1}_n \otimes x_A &\approx \mathbf{m}_\star \\ \mathbf{m}_a &= \mathbf{m}_\star\end{aligned}$$

This may appear to be sleight of hand, however, the point is that we now treat $\mathbf{m}_\star$ as a parameter. In particular, applying the inverse and rearranging on has

$$(I_n \otimes R_A)^\top \mathbf{m}_\star - (I_n \otimes R_A)^\top (\mathbf{1}_n \otimes x_A) - \hat{\mathbf{m}}_b \sim N(0, \Sigma_b) \tag{21}$$

$$\mathbf{m}_\star - \hat{\mathbf{m}}_a \sim N(0, \Sigma_a). \tag{22}$$

[7] Mahony 2020 😀.

[8] This trick is equivalent to applying total least squares.

Note particularly the trick to get the estimate $\hat{\mathbf{m}}_a$ and $\hat{\mathbf{m}}_b$ as independent terms here so that the Gaussian distributions generated by the SLAM maps can be used directly. The consequence of this trick is that the parameters get combined in a nonlinear manner $(I_n \otimes R_A)^\top \mathbf{m}_\star$.

The resulting nonlinear least squares problem is obtained by taking the log-likelihood of (21) and (22) to get

$$(\hat{R}_A, \hat{x}_A, \hat{\mathbf{m}}_\star) = \underset{(R_A, x_A, \mathbf{m}_\star)}{\operatorname{argmin}} \big(\|(I_n \otimes R_A)^\top \mathbf{m}_\star - (I_n \otimes R_A)^\top (\mathbf{1}_n \otimes x_A) - \hat{\mathbf{m}}_b\|^2_{\Sigma_b} + \|\mathbf{m}_\star - \hat{\mathbf{m}}_a\|^2_{\Sigma_a} \big) \quad (23)$$

Solving for the data fusion problem: Solving for the optimum using the Levenberg-Marquardt involves the same construction as we undertook for the early noisy registration process.

Recall the linearisations (13) and (14) around the point $(R_A(j), x_A(j))$. The additional parameter is $\mathbf{m}_\star$. We can write

$$\mathbf{m}_\star = \mathbf{m}_\star(j) + \mu$$

for $\mu \in \mathbb{R}^{2n}$ small.

Define data structures

$$a(j) = (I_n \otimes R_A(j))^\top \mathbf{m}_\star(j) + (I_n \otimes R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) - \hat{\mathbf{m}}_b \in \mathbb{R}^{2n} \quad (24)$$

$$b(j) = (I_n \otimes \mathfrak{e} R_A(j))^\top \mathbf{m}_\star(j) + (I_n \otimes \mathfrak{e} R_A(j))^\top (\mathbf{1}_n \otimes x_A(j)) \in \mathbb{R}^{2n} \quad (25)$$

$$C(j) = \mathbf{1}_n \otimes R_A(j)^\top \in \mathbb{R}^{2n \times 2} \quad (26)$$

$$D(j) = (I_n \otimes R_A(j))^\top \in \mathbb{R}^{2n \times 2n} \quad (27)$$

and note that the linearised least squares can now be written locally around an estimate $(R_A(j), x_A(j), \mathbf{m}_\star(j))$ as

$$(\hat{R}_A, \hat{x}_A, \hat{\mathbf{m}}_\star) = (R_A(j) \exp(\mathfrak{e}\delta_j), x_A(j) + \epsilon_j, \mathbf{m}_\star(j) + \mu_j) \quad (28)$$

where

$$(\delta_j, \epsilon_j, \mu_j) = \underset{(\delta, \epsilon, \mu)}{\operatorname{argmin}} \|a(j) - \delta b(j) + C(j)\epsilon + D(j)\mu\|^2_{\Sigma_b} + \|\mathbf{m}_\star(j) + \mu - \hat{\mathbf{m}}_a\|^2_{\Sigma_a}. \quad (29)$$

This is a straightforward linear least squares problem and can be solved with standard matrix techniques.

Algorithm 3: Levenberg Marquardt for map fusion

Fix a required accuracy a_{thresh} ;
Choose $(R_A(1), x_A(1), \mathbf{m}_\star(1))$;
while $\|a(j)\|^2_{\Sigma_b} > a^2_{thresh}$ **do**
  Compute $a(j)$, $b(j)$, $C(j)$ and $D(j)$ from (24), (25), (26), (27) ;
  Solve (29) for $(\epsilon_{k+1}, \delta_{k+1}, \mu_{k+1})$;
  Compute $(R_A(k+1), x_A(k+1), \mathbf{m}_\star(k+1)) =$
   $(R_A(j)\exp(\mathfrak{e}\delta_j), x_A(j) + \epsilon_j, \mathbf{m}_\star(j) + \mu_j)$;
end
Set $(\hat{R}_A, \hat{x}_A, \hat{\mathbf{m}}_\star) = (R_A(j), x_A(j), \mathbf{m}_\star(j))$;
Result: $(\hat{R}_A, \hat{x}_A, \hat{m}_\star)$

The output from this process will both estimate the maximum likelihood registration parameters (R_A, x_A) as well as the maximum likelihood map estimates $\mathbf{m}_\star$. These parameters are estimated with respect to the reference frame for robot a).

3 Estimating position of an animal from bearing measurements

In this section, we assume that all the data has been registered into a single reference frame. Thus, the question of locating an animal in the terrain is one of fusing a collection of separate noisy measurements of the animals position into a single "best" estimate of the animal position. The problem is slightly more complex due to the fact that we only have bearing measurements of the animals position and we will have to triangulate their position from noisy measurements. Figure 5 shows the measurement process.

3.1 Ideal triangulation: computing target position from a collection of bearings.

Consider a collection of $k = 1, \ldots, N$ measurements of the bearing to a target (in this case an animal) taken from a moving robot. Each measurement consists of two parts, the robot pose $\xi_k = (x_k, y_k, \theta_k)$ and the relative bearing from the robot to the animal α_k as shown in Fig. 5. Recall that we write the geometric pose P_k (1) corresponding to the robot state and that one has $x_{P_k} = (x_k, y_k)$ and

$$R_{P_k} = \begin{pmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{pmatrix}$$

For the moment we will assume that these measurements are ideal, or noise free.

Consider the bearing of the target written as a unit vector $\eta \in \mathrm{S}^2$ from the

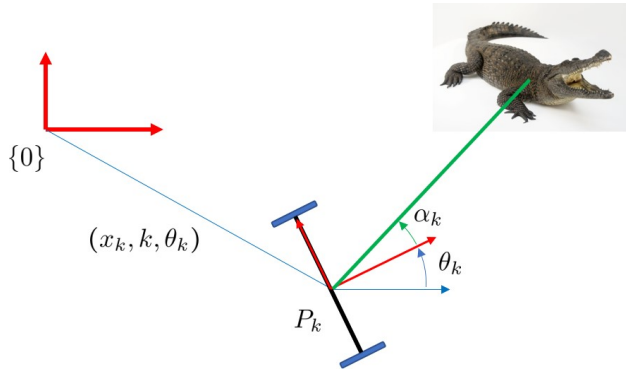

Figure 5: The various states representing the kth measurement bearing measurement of a target (crocodile) as seen from a robot. The robot state is (x_k, y_k, θ_k) and the crocodile bearing is α_k.

robot in the direction of the target. Then

$$\eta(\alpha_k) = \begin{pmatrix} \cos(\alpha_k) \\ \sin(\alpha_k) \end{pmatrix} \in \{P\} \tag{30}$$

in the body fixed frame. Let $z \in \mathbb{R}^2$ denote the location of the target in the reference frame. We can also write the bearing $\gamma \in \mathrm{S}^2$ in terms of the state of the robot and the target position

$$\gamma(z, \xi_k) = \frac{R_{P_k}^\top (z - x_{P_k})}{\|z - x_{P_k}\|} \tag{31}$$

Equating (30) and (32) as $\eta(\alpha_k) = \gamma(z, \xi_k)$ generates a model that can be used to estimate for z

$$\begin{pmatrix} \cos(\alpha_k) \\ \sin(\alpha_k) \end{pmatrix} = \frac{R_{P_k}^\top (z - x_{P_k})}{\|z - x_{P_k}\|} \tag{32}$$

Define an orthogonal bearing

$$\eta_\perp(\alpha_k) = \begin{pmatrix} -\sin(\alpha_k) \\ \cos(\alpha_k) \end{pmatrix}$$

and note that $\eta_\perp(\alpha_k)^\top \eta(\alpha_k) = 0$; in particular. Now multiply (32) by $\eta_\perp(\alpha_k)^\top$ and one has

$$\eta_\perp(\alpha_k)^\top \eta(\alpha_k) = 0 = \eta_\perp(\alpha_k)^\top \gamma(z, \xi_k) = \frac{\eta_\perp(\alpha_k)^\top R_{P_k}^\top (z - x_{P_k})}{\|z - x_{P_k}\|}$$

and assuming $z \neq x_{P_k}$ then $\eta_\perp(\alpha_k)^\top R_{P_k}^\top (z - x_{P_k}) = 0$. Define data

$$a_k = \eta_\perp(\alpha_k)^\top R_{P_k}^\top \in \mathbb{R}^{1\times 2} \tag{33}$$

$$b_k = \eta_\perp(\alpha_k)^\top R_{P_k}^\top x_{P_k} \in \mathbb{R} \tag{34}$$

and note that

$$a_k z = b_k$$

For a single k this equation by itself is insufficient to resolve the target location z. One needs at least two non-collinear measurements in order to solve this set of linear equations. Of course, more measurements is better, as long as they are consistent. In general, we will use all the measurements by forming a data matrices

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} \in \mathbb{R}^{N\times 2}, \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix} \in \mathbb{R}^{N\times 1} \tag{35}$$

and solving the linear equations

$$Az = b. \tag{36}$$

In the ideal case this stack of equations is consistent. In the real world of course, the data a_k and b_k will be noisy and the rows of $Az = b$ will all yield slightly different solutions. In this case, we need to find an approximate solution. There is a stochastically principled way to do this that we will discuss in the next section. However, an OK starting position is to a use a simple 'least squares' solution for the linear system (36)

$$z = (A^\top A)^{-1} A^\top b. \tag{37}$$

The resulting estimate z is the location of the target expressed in the reference frame as computed from N bearing measurements and poses.

3.2 Triangulation in the presence of noise

The linear least squares solution (37) allows for noise, but makes the assumption that: i) all the noise is in the b_k, and ii) the noise is independent and identically distributed. Although this allows the algorithm to be applied to any data, even if it is not consistent, this will often lead to poor estimation. The big issue is assumption i), that all the noise is in b_k. In fact, in the real data it is clear that much of the uncertainty in the task lies in the robot pose estimate (which contributes to A) and not in the estimation of bearing (which contributes to b). Solving the general case can be framed as a nonlinear least squares problem and can solved using the Levenberg-Marquardt method.

Consider the robot state $\xi_k = (x_k, y_k, \theta_k)$ associated with a particular measurement of bearing. This is obtained as a part of a SLAM solution at a particular point in time $(\hat{\xi}_k, \hat{\mathbf{m}}_k)$ taken when the bearing was measured. The SLAM

solution yields also a variance estimate $\Sigma_k \in \mathbb{R}^{(3+2n)\times(3+2n)}$ for the state. Only the state associated with the robot pose is required and we take the marginal ξ_k

$$\xi_k \sim N_{(\hat{\xi}_k, \Sigma_{\xi_k})}(\xi_k)$$

where Σ_{ξ_k} is the upper left 3×3 block of Σ_k. To make this explicit in the sequel we write these relationships in the form generative noise models

$$\xi_k = \hat{\xi}_k + \begin{pmatrix} \epsilon_k \\ \delta_k \end{pmatrix}, \quad (\epsilon_k, \delta_k) \sim N(0, \Sigma_{\xi_k}) \tag{38}$$

We also need a stochastic model for the bearing α_k. This is measurement that comes from the deep network. Although the classification problem underlying the deep network was trained using the logicial loss to try and interpret classification output as a probability, you will see that the bearing is only loosely related to this stochastic interpretation. One of the tasks of the workshop will be to generate an estimate of uncertainty in the bearing measurement form the output of the neural network, an important step in modern robotics where deep networks are used as a tool. We will write

$$\alpha_k = \hat{\alpha}_k + \beta_k, \quad \beta_k \sim N(0, \sigma^2_{\beta_k}) \tag{39}$$

where $\sigma^2_{\beta_k}$ is the estimated variance of α_k, based on whatever method that you believe.

We combine all the noise into a single variable

$$\mu_k = \begin{pmatrix} \epsilon_k \\ \delta_k \\ \beta_k \end{pmatrix} \sim N(0, \Sigma_{\mu_k}) \tag{40}$$

with

$$\Sigma_{\mu_k} = \begin{pmatrix} \Sigma_{\xi_k} & 0 \\ 0 & \sigma^2_{\beta_k} \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

noting that α_k is independent of ξ_k which gives the covariance a block diagonal structure.

Recall the model for the bearing (30) and note that

$$R_{P_k}\eta(\alpha_k) = \begin{pmatrix} \cos(\alpha_k + \theta_k) \\ \sin(\alpha_k + \theta_k) \end{pmatrix} = \eta(\alpha_k + \theta_k)$$

and thus recalling (32) then

$$R_{P_k}\gamma(z, \xi_k) = R_{P_k}\frac{R^\top_{P_k}(z - x_{P_k})}{\|z - x_{P_k}\|} = R_{P_k}\eta(\alpha_k) = \eta(\alpha_k + \theta_k) \tag{41}$$

The goal is to use (41) to determine z from multiple measurements $k = 1, \ldots, N$ taking into account the noise characteristics of $(\xi_k, \alpha_k) = (x_k, y_k, \theta_k, \alpha_k)$. To obtain a generative noise model from (41) then we substitute the generative

noise models for the random variables with the small noise terms and linearise. That is substituting for (38) and (39) one obtains

$$\frac{z - \hat{x}_{P_k} - \epsilon_k}{\|z - \hat{x}_{P_k} - \epsilon_k\|} = \eta(\hat{\alpha}_k + \hat{\theta}_k + \delta_k + \beta_k)$$

and linearising around $(\delta_k, \epsilon_k, \beta_k) = (0, 0, 0)$ one obtains

$$\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \left(I_2 - \hat{R}_{P_k}\hat{\gamma}_k\hat{\gamma}_k^\top\hat{R}_{P_k}\right)\epsilon_k \approx \hat{\eta}_k + (\hat{\eta}_k)_\perp(\delta_k + \beta_k)$$

where $\hat{\eta}_k = \eta(\hat{\alpha}_k + \hat{\theta}_k)$. Thus,

$$\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \hat{\eta}_k \approx -\left(I_2 - \hat{R}_{P_k}\hat{\gamma}_k\hat{\gamma}_k^\top\hat{R}_{P_k}\right)\epsilon_k + (\hat{\eta}_k)_\perp(\delta_k + \beta_k)$$

where the left hand side is the model and the right hand side contains the noise dependence. Define a data structure

$$C_k = \left(\left(I_2 - \hat{R}_{P_k}\hat{\gamma}_k\hat{\gamma}_k^\top\hat{R}_{P_k}\right) \quad (\hat{\eta}_k)_\perp \quad (\hat{\eta}_k)_\perp\right) \in \mathbb{R}^{2\times 4} \tag{42}$$

then

$$\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \hat{\eta}_k \approx C_k\mu_k$$

where μ_k is the combined Gaussian noise vector (40). This linearisation is undertaken to expose the noise as an additive Gaussian term in a straightforward manner. From this expression we can write

$$\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \hat{\eta}_k \sim N(0, C_k^\top\Sigma_{\mu_k}C_k).$$

The associated maximum likelihood nonlinear least squares cost is

$$\hat{z} = \underset{z}{\operatorname{argmin}} \|\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \hat{\eta}_k\|_{C_k^\top\Sigma_{\mu_k}C_k}. \tag{43}$$

This cost function is written for a single data instance k. The full nonlinear least squares cost will be the sum over all data

$$\hat{z} = \underset{z}{\operatorname{argmin}} \sum_{k=1}^{N} \|\hat{R}_{P_k}\gamma_k(z, \hat{\xi}_{P_k}) - \hat{\eta}_k\|_{C_k^\top\Sigma_{\mu_k}C_k}. \tag{44}$$

Remark 3.1 *The data term C_k depends on the parameter z and a careful analysis would involve modelling its dependence through a suitable linearisation. However, this additional complexity is unlikely to lead to significant benefit in the optimisation compared to the advantages of properly account for the position noise. Thus, for the sake of simplicity of formula, we will ignore this dependency and simply solve for z based on the dependency on the left hand side of the equation.*

The least squares problem (44) is still nonlienar due to the z nonlinearity in γ. We define a local model for z around a point $z(j) \in \mathbb{R}^2$

$$z = z(j) + \zeta$$

where $\zeta \in \mathbb{R}^2$ is a small perturbation and $z(j)$ will be the iterate of the Levenberg-Marquardt algorithms. To apply Levenberg-Marquardt to find the best estimate $\hat{z}$ from the data then we linearise the argument of (44) in z around the point $z(j)$

$$\hat{R}_{P_k}\gamma(z, \hat{\xi}_{P_k}) - \hat{\eta}_k \approx \\ \hat{R}_{P_k}\gamma(z(j), \hat{\xi}_{P_k}) - \hat{\eta}_k + \left(I_2 - \hat{R}_{P_k}\gamma(z(j), \hat{\xi}_{P_k})\gamma(z(j), \hat{\xi}_{P_k})^\top \hat{R}_{P_k}^\top\right)\zeta.$$

Define data structure

$$A_{jk} = \left(I_2 - \hat{R}_{P_k}\gamma(z(j), \hat{\xi}_{P_k})\gamma(z(j), \hat{\xi}_{P_k})^\top \hat{R}_{P_k}^\top\right) \in \mathbb{R}^{2\times 2} \tag{45}$$

$$b_{jk} = \hat{R}_{P_k}\gamma(z, \hat{\xi}_{P_k}) - \hat{\eta}_k \in \mathbb{R}^1. \tag{46}$$

Then the nonlinear least squares cost (44) can be written as a linear least squares problem

$$\hat{z} = z(j) + \zeta_j$$

$$\zeta_j = \underset{\zeta}{\operatorname{argmin}} \sum_{k=1}^{N} \|A_{jk}\zeta + b_{jk}\|_{C_k^\top \Sigma_{\mu_k} C_k}. \tag{47}$$

This is straightforward to solve for ζ and provides the update step for the Levenberg-Marquardt.

Algorithm 4: Levenberg Marquardt for noisy triangulation

Fix a required accuracy a_{thresh} ;
Choose $z(1)$;
while $\sum_{k=1}^{N} \|b_{jk}\|_{C_k^\top \Sigma_{\mu_k} C_k} \geq a_{thresh}$ **do**
  Compute C_k, A_{jk} and b_{jk} from (42), (45) and (46) ;
  Solve (47) for ζ_j ;
  Compute $z(j+1) = z(j) + \zeta_j$;
end
Set $\hat{z} = z(j)$;
Result: $\hat{z}$

The estimate $\hat{z}$ is the triangulation point expressed in the reference frame of the problem formulation.