MAANI GHAFFARI

LECTURE NOTES FOR MOBILE ROBOTICS

UNIVERSITY OF MICHIGAN – ANN ARBOR
NA 568

UNIVERSITY OF MICHIGAN – ANN ARBOR
NA 568

CURLY.ENGIN.UMICH.EDU

Online, March 2023

Contents

List of Figures

List of Tables

Introduction

To be written ... not so long intro ... to explain why we did what we did ...

[**Maani: Pardon the mess! These notes are still under construction into a coherent booklet for the course.**]

1
Preliminaries

1.1 Probability Theory and Statistics

Let X be a random variable that maps the sample space Ω (set of all possible outcomes) to the state space $\mathcal{X}$. Let $p(X = x) \geq 0$ be the probability of the random variable X taking a specific value x. If X is a discrete random variable, then

$$\sum_{x \in \mathcal{X}} p(X = x) = 1, \tag{1.1}$$

and for the continuous form we can write

$$\int_{x \in \mathcal{X}} p(X = x)dx = 1. \tag{1.2}$$

For the sake of simplicity, it is common to use $p(x)$ instead of $p(X = x)$ and sometimes refer to x as the random variable itself. Let Y be another random variable, the joint distribution of X and Y is $p(x, y) = p(X = x \text{ and } Y = y)$ and if X and Y are independent

$$p(x, y) = p(x)p(y). \tag{1.3}$$

The conditional probability of X given another random variable Y is defined as

$$p(x|y) = \frac{p(x, y)}{p(y)} \qquad p(y) > 0. \tag{1.4}$$

Given the joint distribution of X and Y, the marginalization rule states that the marginal distribution of X can be computed by summing (integration) over Y. The law of total probability is its variant which uses the conditional probability definition and can be written as

$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y) = \sum_{y \in \mathcal{Y}} p(x|y)p(y), \tag{1.5}$$

and for continuous random variables is

$$p(x) = \int_{y \in \mathcal{Y}} p(x, y)dy = \int_{y \in \mathcal{Y}} p(x|y)p(y)dy. \tag{1.6}$$

Given three random variables X, Y, and Z, Bayes rule relates the prior probability distribution, $p(x|z)$, and the likelihood function, $p(y|x,z)$, as follows.

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)}. \tag{1.7}$$

The term $p(x|y,z)$ is called the posterior probability distribution over X.

$$p(\text{hypothesis}|\text{data}) = \frac{p(\text{data}|\text{hypothesis})p(\text{hypothesis})}{p(\text{data})},$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Evidence (Marginal Likelihood)}}.$$

The expected value of the random variable X is

$$\mathbb{E}[X] = \int_\Omega X dp = \int_{\mathcal{X}} x p(x) dx \tag{1.8}$$

and if X is discrete

$$\mathbb{E}[X] = \sum_{\mathcal{X}} x p(x) \tag{1.9}$$

The expectation operator is linear which follows from linearity of integration and has the following properties:

- $\mathbb{E}[a] = a$
- $\mathbb{E}[aX] = a\mathbb{E}[X]$
- $\mathbb{E}[a + X] = a + \mathbb{E}[X]$
- $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$,

where X and Y are two arbitrary random variables and a and b are constants.

The variance of X can be calculated as

$$\mathbb{V}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2. \tag{1.10}$$

The covariance of a random vector $X = x$ can be calculated as

$$\text{Cov}[X] = \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\mathsf{T}] = \mathbb{E}[XX^\mathsf{T}] - \mathbb{E}[X]\mathbb{E}[X]^\mathsf{T}. \tag{1.11}$$

A covariance matrix such as Σ is symmetric, i.e., $\Sigma = \Sigma^\mathsf{T}$, and positive semi-definite, i.e., $x^\mathsf{T}\Sigma x \geq 0$ and all eigenvalues are nonnegative. A symmetric positive definite matrix ($x^\mathsf{T}\Sigma x > 0$) has positive eigenvalues and a unique Cholesky decomposition, i.e., $\Sigma = LL^\mathsf{T}$, where L is a lower triangular matrix.

The correlation coefficient is defined as

$$\rho_{XY} = \frac{\text{Cov}[XY]}{\sqrt{\mathbb{V}[X]\mathbb{V}[Y]}} \qquad |\rho_{XY}| \leq 1, \tag{1.12}$$

where the bound follows from the Cauchy-Schwarz inequality which asserts (assuming $\mathbb{E}[X^2]$ and $\mathbb{E}[Y^2]$ are finite)

$$|\mathbb{E}[XY]|^2 \leq \mathbb{E}[X^2]\mathbb{E}[Y^2]. \tag{1.13}$$

Remark 1.1. *In general,* $\mathbb{E}[XY] \neq \mathbb{E}[X]\mathbb{E}[Y]$*, unless* X *and* Y *are uncorrelated. Now it is clear that if* X *and* Y *are independent, i.e.,* $X \perp Y$*, then* $\mathrm{Cov}[XY] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] = 0$ *and* X *and* Y *are uncorrelated (*$\rho_{XY} = 0$*). However, uncorrelated random variables are not necessarily independent (e.g.,* Y *is a non-constant function of* X*).*

Remark 1.2. *If a random vector has a multivariate normal distribution then any two or more of its components that are uncorrelated are independent. However, it is not true that two random variables that are (separately, marginally) normally distributed and uncorrelated are independent* [1].

[1] https://en.wikipedia.org/wiki/Multivariate_normal_distribution

The univariate (one-dimensional) *Gaussian (or normal) distribution* with mean μ (location) and variance σ^2 (scale) has the following Probability Density Function (PDF).

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \tag{1.14}$$

We often write $x \sim \mathcal{N}(\mu, \sigma^2)$ or $\mathcal{N}(x; \mu, \sigma^2)$ to imply that x follows a Gaussian distribution with mean $\mu = \mathbb{E}[x]$ and variance $\sigma^2 = \mathbb{V}[x]$.

The *multivariate* Gaussian distribution of an n-dimensional random vector $x \sim \mathcal{N}(\mu, \Sigma)$, with mean $\mu = \mathbb{E}[x] \in \mathbb{R}^n$ and covariance

$$\Sigma = \mathrm{Cov}[x] = \mathbb{E}[(x-\mu)(x-\mu)^\mathsf{T}],$$

can be written as follows.

$$p(x) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)\right). \tag{1.15}$$

Lemma 1.1 (Marginalization and conditioning of normal distribution)**.** *Let* x *and* y *be jointly Gaussian random vectors*

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^\mathsf{T} & B \end{bmatrix}\right), \tag{1.16}$$

then the marginal distribution of x *is*

$$x \sim \mathcal{N}(\mu_x, A), \tag{1.17}$$

and the conditional distribution of x *given* y *is*

$$x|y \sim \mathcal{N}(\mu_x + CB^{-1}(y-\mu_y), A - CB^{-1}C^\mathsf{T}). \tag{1.18}$$

Proposition 1.2 (Affine transformation of a multivariate normal distribution)**.** *Suppose* $x \sim \mathcal{N}(\mu, \Sigma)$ *and* $y = Ax + b$*. Then* $y \sim \mathcal{N}(A\mu + b, A\Sigma A^\mathsf{T})$*.*

Proof. Exercise. □

The canonical parametrization of a multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ can be identified by the *information (or precision) matrix* $\Lambda = \Sigma^{-1}$, and the *information vector* $\eta = \Sigma^{-1}\mu$. Then we can write $p(x) = \mathcal{N}(\mu, \Sigma) = \mathcal{N}^{-1}(\eta, \Lambda)$.

Example 1.1 (Bayes rule)**.** *A diagnostic test has a probability* 0.95 *of giving a positive result when applied to a person suffering from a certain disease, and a probability* 0.10 *of giving a (false) positive when applied to a non-sufferer. It is estimated that* 0.5% *of the population are sufferers. Suppose that the test is now administered to a person about whom we have no relevant information relating to the disease (apart from the fact that he/she comes from this population). Calculate the following probabilities:*

1. *that the test result will be positive;*

2. *that, given a positive result, the person is a sufferer;*

3. *that, given a negative result, the person is a non-sufferer;*

4. *that the person will be misclassified.*

Let us first define the following random variables: T (Test positive), S (Sufferer), M (Misclassified). From given information we have $p(T|S) = 0.95$, $p(T|\neg S) = 0.1$, *and* $P(S) = 0.005$. *Then*

$$p(T) = \sum_S p(T|S)p(S) = p(T|S)p(S) + p(T|\neg S)p(\neg S)$$
$$= 0.95 \times 0.005 + 0.1 \times (1 - 0.005) = 0.10425$$

$$p(S|T) = \frac{p(T|S)p(S)}{p(T)} = (0.95 \times 0.005)/0.10425 = 0.04556$$

$$p(\neg S|\neg T) = \frac{p(\neg T|\neg S)p(\neg S)}{p(\neg T)} = (1 - 0.1) \times (1 - 0.005)/(1 - 0.10425)$$
$$= 0.99972$$

$$p(M) = p(T, \neg S) + p(\neg T, S) = p(T|\neg S)p(\neg S) + p(\neg T|S)p(S)$$
$$= 0.1 \times (1 - 0.005) + (1 - 0.95) \times 0.005 = 0.09975$$

Example 1.2 (Visualizing multivariate Gaussian)**.** *Let* $x = \text{vec}(x_1, x_2)$ *and* $x \sim \mathcal{N}(\mu, \Sigma)$ *where*

$$\mu = \begin{bmatrix} 0.0 \\ 0.5 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.8 & 0.3 \\ 0.3 & 1.0 \end{bmatrix}$$

Figure 1.1 shows the PDF plot of $\mathcal{N}(\mu, \Sigma)$ *as well as marginal and conditional distributions. See* `mvn_plots.m` *for more details and reproducing the plots.*

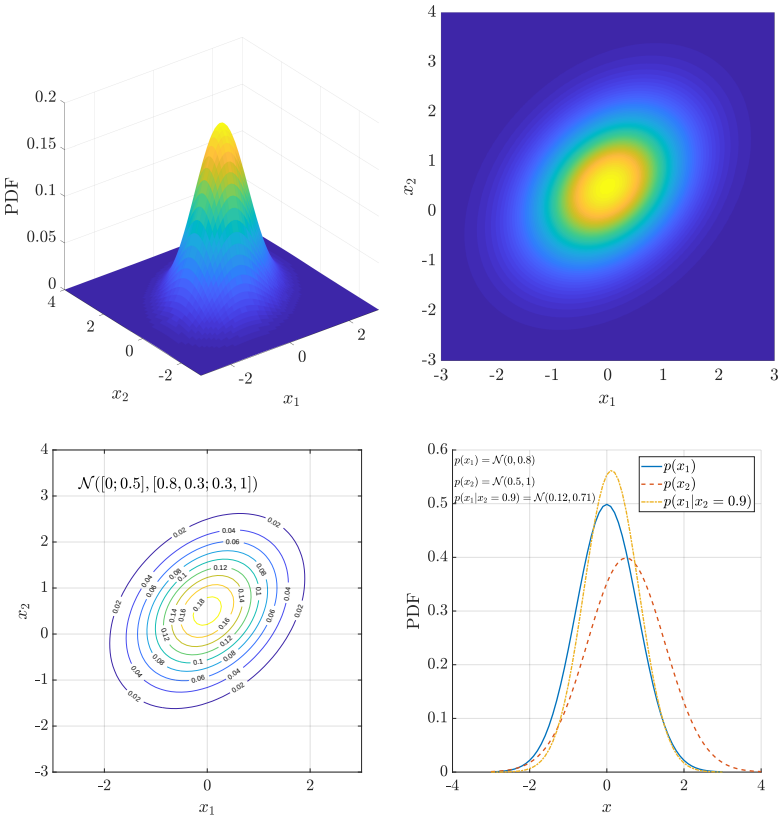

Figure 1.1: From top left, the plots show two-dimensional PDF, top view of the first plot, the contour plot of the PDF, and the marginals and the conditional distribution of $p(x_1|x_2 = 0.9)$.

1.2 Sampling from Gaussian Distributions

Suppose we wish to draw samples from $Y \sim \mathcal{N}(\mu, \Sigma)$. If the covariance matrix of Y is not degenerate (is positive definite), we have $\Sigma = LL^\mathsf{T}$ where L is a lower triangular matrix computed using Cholesky decomposition. Now, let $X \sim \mathcal{N}(0, I_n)$ be a vector of standard normal random variables where n is the dimension (length) of Y. Define $Z := LX + \mu$. Using Proposition 1.2, we have $\mathbb{E}[Z] = \mu$ and $\mathrm{Cov}[Z] = LL^\mathsf{T}$. Therefore, using samples from $\mathcal{N}(0, 1)$, we are able to draw samples from $\mathcal{N}(\mu, \Sigma)$.

1.3 Sample Mean and Covariance

Sometimes we do not know the distribution of data, but instead we have access to samples or observations. Let $X = x$ be a random vector and $x_1, \ldots, x_n$ be n independent samples (realization) of X. The sample or empirical mean, $\overline{x}$, and (unbiased) covariance, $\overline{\Sigma}$, can be computed as follows.

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1.19}$$

$$\overline{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^\mathsf{T} \tag{1.20}$$

Note that the sample mean is a random variable.

1.4 Chi-Square Distribution, Mahalanobis Distance, and Confidence Ellipsoid

Let $X \sim \mathcal{N}(\mu, \Sigma)$ be an n-dimensional Gaussian random vector. The scalar random variable, q, defined by the quadratic form

$$q = (x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu), \tag{1.21}$$

is the sum of the squares of n independent zero-mean, unity-variance Gaussian random variables. We say q has a chi-square distribution with n Degrees Of Freedom (DOF) [2], i.e., $q \sim \chi^2_n$. It can be shown that $\mathbb{E}[q] = n$ and $\mathbb{V}[q] = 2n$. If $q_1 \sim \chi^2_{n_1}$ and $q_2 \sim \chi^2_{n_2}$, then $q_3 = q_1 + q_2 \sim \chi^2_{n_1+n_2}$. Chi-square goodness-of-fit is a statistical test that determines if an observation sample comes from a specified probability distribution. In particular, we would like to test the null hypothesis that the data in x comes from a normal distribution such as $\mathcal{N}(\mu, \Sigma)$. This test is useful for data association. The distance $\sqrt{(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)}$ is known as Mahalanobis distance.

[2] In statistics, the number of degrees of freedom is the number of values in the final calculation of a statistic that are free to vary.

Remark 1.3. *The weighted sum of independent identically distributed (i.i.d.) chi-square random variables does not follow a chi-square distribution. For more details, see Bar-Shalom et al. (2001, page 60).*

Suppose we are a give a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$, and we would like to illustrate the confidence region that the mentioned distribution covers with a certain probability. Geometrically, an equation such as $(x-b)^\mathsf{T}A(x-b) = 1$, where A is positive definite and $x, b \in \mathbb{R}^n$, corresponds to an ellipsoid in $\mathbb{R}^n$ centered at b. Since A is positive definite, we have $A = LL^\mathsf{T}$. It can be shown that L corresponds to a linear transformation that rotates and scale a sphere to the desired ellipsoid.

Now it is clear that $q = (x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)$ also corresponds to an ellipsoid. The chi-square value, q, for a desired significance level (p-value) can be found using the pre-calculated chi-square table. Finally, to map a point from a unit sphere, x_s, to the desired ellipsoid, x_e, we can use the following formula:

$$x_e = \sqrt{q}Lx_s + \mu, \tag{1.22}$$

where L is the Cholesky factor of the covariance matrix, i.e., $\Sigma = LL^\mathsf{T}$. Figure 1.2 shows two examples of the 95% confidence ellipsoid where samples are drawn from known normal distributions.

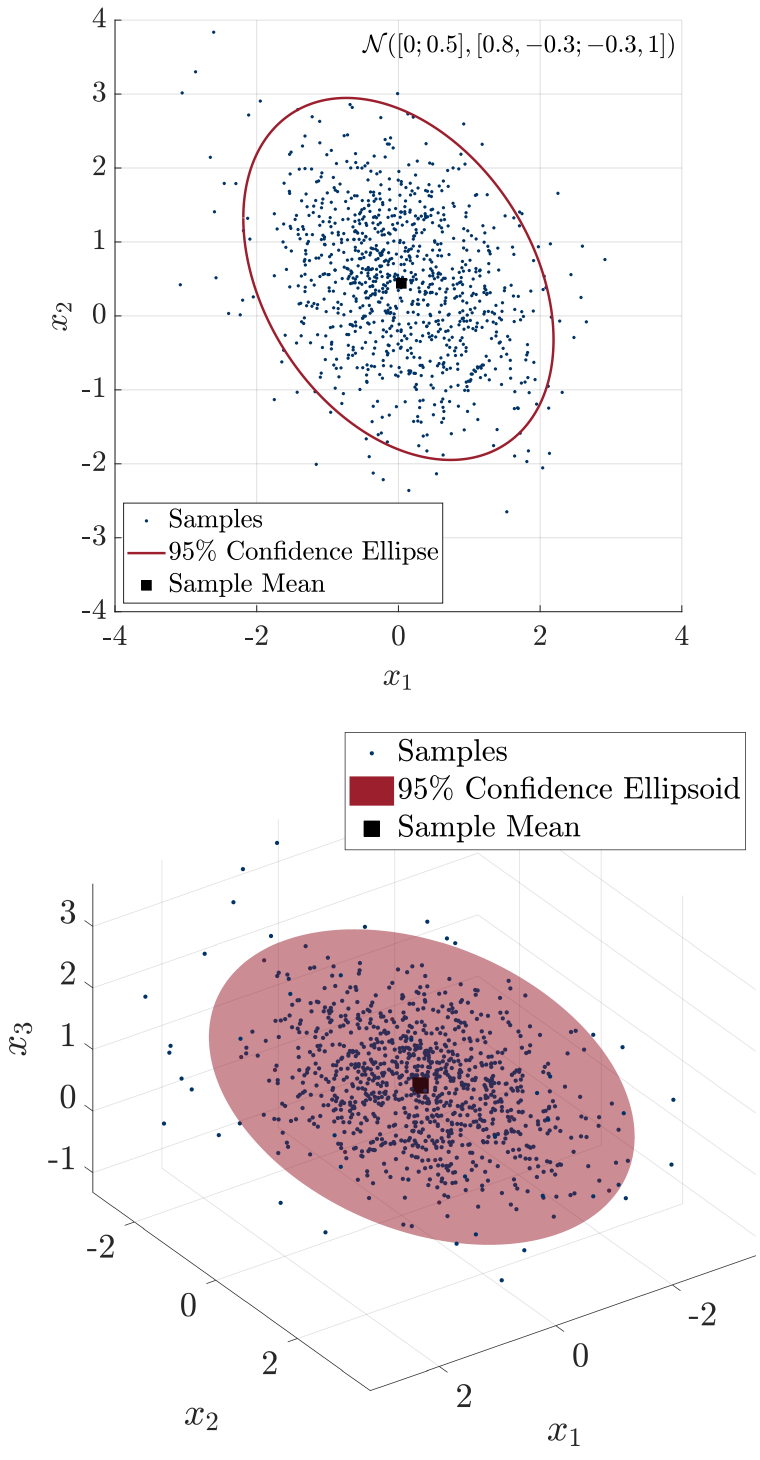

Figure 1.2: Illustrative examples of drawing confidence ellipsoids. See `confidence_ellipsoid_plots.m` for details and reproducing the plots.

1.5 Estimation

A static estimation problem involve estimation of the system parameters. State estimation is a dynamic estimation problem, that is the state variables of the system evolve over time (Bar-Shalom et al., 2001).

Definition 1.1 (Parameter)**.** *A quantity that is assumed to be time invariant or its time variation is slow compared to the state variables of a system.*

The PDF of the measurements, z, conditioned on the parameter, x, is called the *likelihood function* of the parameter

$$l(x) \triangleq p(z|x). \tag{1.23}$$

The likelihood measures that how likely a parameter value is given the obtained observations (evidence from the data).

1.5.1 Maximum Likelihood (ML) Estimator

The maximum likelihood method maximizes the likelihood function, leading to the ML estimator

$$x^\star(z) = \arg\max\ l(x) = \arg\max_x\ p(z|x), \tag{1.24}$$

where $x^\star(z)$ is a random variable as it is a function of random observations, z. The ML estimate is the solution of the *likelihood equation* (necessary condition)

$$\frac{\mathrm{d}l(x)}{\mathrm{d}x} = \frac{\mathrm{d}p(z|x)}{\mathrm{d}x} = 0. \tag{1.25}$$

1.5.2 Maximum A Posteriori (MAP) Estimator

In the Bayesian framework, one can place a prior distribution over the parameter. Then, the MAP estimator maximizes the posterior PDF

$$x^\star(z) = \arg\max_x\ p(x|z) = \arg\max_x\ p(z|x)p(x), \tag{1.26}$$

were the last equality is true because the normalization constant in the Bayes' formula is independent of x.

Remark 1.4. *Since* log *is a monotonic function, it is often the case that we use the logarithm of the likelihood or posterior for better scaling.*

1.5.3 Least Squares (LS) Estimator

A common estimation procedure for nonrandom parameters is the LS method. Let $h_k(x)$ be a nonlinear measurement model and $z_k = h_k(x) + v$

the scalar measurements. The LS estimator of x is the solution of the following problem known as the nonlinear LS.

$$x_k^\star = \arg\min_x \sum_{k=1}^{t} [z_k - h_k(x)]^2. \tag{1.27}$$

If $h_k(x)$ is linear in x, (1.27) becomes the linear LS problem. Furthermore, there is no assumption on the distribution of the noise term, v, in the LS problem. If we assume $v \sim \mathcal{N}(0, \sigma)$, then the LS estimator coincides with the ML estimator.

1.5.4 Minimum Mean Square Error (MMSE) Estimator

For random parameters, the MMSE estimator is

$$x^\star(z) = \arg\min_{\hat{x}} \mathbb{E}[(\hat{x} - x)^2 | z], \tag{1.28}$$

where $\hat{x}$ is an estimator of x. The MMSE estimator's solution corresponds to the conditional mean of x, that is

$$x^\star(z) = \mathbb{E}[x|z] = \int_{-\infty}^{\infty} x p(x|z) dx, \tag{1.29}$$

and can be obtained by

$$\frac{\partial \mathbb{E}[(\hat{x} - x)^2 | z]}{\partial \hat{x}} = \mathbb{E}[2(\hat{x} - x)|z] = 2(\hat{x} - \mathbb{E}[x|z]) = 0.$$

Remark 1.5. *If $p(x|z)$ is Gaussian, then the MMSE estimator (the conditional mean) coincides with the MAP estimator since the mode and mean of the Gaussian distribution are the same.*

Example 1.3. *The MMSE estimate - the conditional mean - of a Gaussian random vector in terms of another Gaussian random vector (the measurement).*

1.5.5 Central Limit Theorem

If the random variables X_i are independent, under general conditions the distribution of

$$Y_n = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{1.30}$$

is approximately Gaussian, of mean $\frac{1}{n}\sum_{i=1}^n \mu_i$ and variance $\frac{1}{n^2}\sum_{i=1}^n \sigma_i^2$, and μ_i and σ_i^2 are the mean and variance of X_i. As $n \to \infty$, the approximation becomes more accurate (Anderson and Moore, 1979).

The Central Limit Theorem (CLT) has a very important role in characterizing many real-world sources of uncertainty: It is used as the justification/excuse to make the omnipresent Gaussian assumption. For example, the thermal noise in electronic devices, as the sum of many "small contributions," is indeed close to Gaussian (Bar-Shalom et al., 2001).

1.5.6 Filtering, Smoothing, and Prediction

This part is a summary from Anderson and Moore (1979, Chapter 2). In general and a broad sense, *filtering* may refer to extraction of information about the internal state of a system from noisy measurements. Technically, filtering refer to a specific type of information processing (inference). In particular, in the latter definition, filtering means recovery of the state at time t, using measurements up to time t.

Example 1.4. *An example of the application of filtering in everyday life is in radio reception. Here the signal of interest is the voice signal. This signal is used to modulate a high frequency carrier that is transmitted to a radio receiver. The received signal is inevitably corrupted by noise, and so, when demodulated, it is filtered to recover as well as possible the original signal.*

Smoothing is another form of information processing that differs from filtering. In smoothing, information about the state need not become available at time t, and measurements derived later than time t can be used in obtaining information about the state. The trade-off here is that there must be a delay in producing the information about the state, as compared with the filtering case, but we can use more measurement data than in the filtering case in producing the information about the state. Since in smoothing we can also use measurements after time t, one should expect the smoothing process to be more accurate in some sense than the filtering process.

Example 1.5. *An example of smoothing is provided by the way the human brain tackles the problem of reading hastily written handwriting. Each word is tackled sequentially, and when word is reached that is particularly difficult to interpret, several words after the difficult word, as well as those before it, may be used to attempt to deduce the word.*

Prediction is the forecasting side of information processing. Here we intend to obtain information about the state at some time after t, while we may use measurements up to time t.

Example 1.6. *When attempting to catch a ball, we have to predict the future trajectory of the ball in order to position a catching hand correctly. This task becomes more difficult the more the ball is subject to random disturbances such as wind gusts. Generally, any prediction task becomes more difficult as the environment becomes noisier.*

1.6 Solving System of Linear Equations

For an accessible background on linear algebra and solving systems of linear equations, see the University of Michigan ROB 101 course materials (rob).

1.6.1 Cholesky Decomposition

The Cholesky decomposition of a symmetric, positive definite matrix A decomposes A into a product of a lower triangular matrix L and its transpose

$$LL^\mathsf{T} = A \tag{1.31}$$

where L is called the Cholesky factor. The Cholesky decomposition is useful for solving linear systems with symmetric, positive definite coefficient matrix A. To solve $Ax = b$ for x, first solve the triangular system $Ly = b$ by forward substitution and then the triangular system $L^\mathsf{T}x = y$ by back substitution. Using the backslash operator, we write the solution as $x = L^\mathsf{T}\backslash(L\backslash b)$, where the notation $A\backslash b$ is the vector x which solves $Ax = b$. Both the forward and backward substitution steps require $n^2/2$ operations, when A is of size $n \times n$. The computation of the Cholesky factor L takes time $n^3/6$, so it is the method of choice when it can be applied [3].

[3] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press, 2006, vol. 1

1.6.2 QR Decomposition

Consider the case of overdetermined set of linear equations in which there are more equations than unknowns [4]. We wish to approximately solve $Ax = b$ where now $A \in \mathbb{R}^{m\times n}$ is skinny, i.e., $m > n$. The QR decomposition or factorization finds $A = QR$ where $Q \in \mathbb{R}^{m\times n}$ is an orthogonal matrix, i.e., $Q^\mathsf{T}Q = I_n$, and $R \in \mathbb{R}^{n\times n}$ is an upper triangular and invertible matrix.

[4] S. Boyd, "Lecture notes for EE263," *Introduction to Linear Dynamical Systems*, 2008

To approximately solve $Ax = b$, find $A = QR$ and substitute it into the pseudo-inverse of A, i.e., $(A^\mathsf{T}A)^{-1}A^\mathsf{T}$, resulting in $\hat{x} = R\backslash Q^\mathsf{T}b$.

1.7 Information Theory

Entropy is a measure of the uncertainty of a random variable (Cover and Thomas, 1991). The entropy $H(X)$ of a discrete random variable X is defined as

$$H(X) = \mathbb{E}_{p(x)}[\log \frac{1}{p(x)}] = -\sum_{\mathcal{X}} p(x) \log p(x). \tag{1.32}$$

The joint entropy $H(X, Y)$ of discrete random variables X and Y with a joint distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(x, y). \tag{1.33}$$

The chain rule implies that

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y), \tag{1.34}$$

where $H(X|Y)$ is the conditional entropy and is defined as

$$H(Y|X) = -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x, y) \log p(y|x) \tag{1.35}$$

Theorem 1.3 (Chain rule for entropy)**.** *Let $X_1, X_2, ..., X_n$ be drawn according to the joint probability distribution $p(x_1, x_2, ..., x_n)$. Then*

$$H(X_1, X_2, ..., X_n) = \sum_{i=1}^{n} H(X_i|X_{i-1}, ..., X_1) \tag{1.36}$$

Proof. Please refer to [5] for the proof. □

[5] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 1991

The relative entropy or Kullback–Leibler Divergence (KLD) is a measure of distance between two distributions $p(x)$ and $q(x)$. It is defined as

$$D(p||q) = \mathbb{E}_{p(x)}[\log \frac{p(x)}{q(x)}]. \tag{1.37}$$

Theorem 1.4 (Information inequality)**.** *Let X be a discrete random variable. Let $p(x)$ and $q(x)$ be two probability mass functions. Then*

$$D(p||q) \geq 0, \tag{1.38}$$

with equality if and only if $p(x) = q(x)$ for all x.

Proof. Please refer to [6] for the proof. □

[6] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 1991

The mutual information $I(X;Y)$ is the reduction in the uncertainty of one random variable due to the knowledge of the other. The mutual information is non-negative and can be written as

$$I(X;Y) = D(p(x,y)||p(x)p(y)) = \mathbb{E}_{p(x,y)}[\log \frac{p(x,y)}{p(x)p(y)}], \tag{1.39}$$

$$I(X;Y) = H(X) - H(X|Y). \tag{1.40}$$

Corollary 1.5 (Nonnegativity of mutual information)**.** *For any two random variables X and Y,*

$$I(X;Y) \geq 0, \tag{1.41}$$

with equality if and only if X and Y are independent.

Proof. $I(X;Y) = D(p(x,y)||p(x)p(y)) \geq 0$, with equality if and only if $p(x,y) = p(x)p(y)$. □

Some immediate consequences of the provided definitions are as follows.

Lemma 1.6. *For any discrete random variable X, we have $H(X) \geq 0$.*

Proof. $0 \leq p(x) \leq 1$ implies that $\log \frac{1}{p(x)} \geq 0$. □

Theorem 1.7 (Conditioning reduces entropy)**.** *For any two random variables X and Y,*

$$H(X|Y) \leq H(X), \tag{1.42}$$

with equality if and only if X and Y are independent.

Proof. $0 \leq I(X;Y) = H(X) - H(X|Y)$. □

We now define the equivalent of the functions mentioned above for probability density functions.

Definition 1.2 (Differential entropy)**.** *Let X be a continuous random variable whose support set is $\mathcal{S}$. Let $p(x)$ be the probability density function for X. The differential entropy $h(X)$ of X is defined as*

$$h(X) = -\int_{\mathcal{S}} p(x) \log p(x) dx. \tag{1.43}$$

Remark 1.6. *The differential entropy of a set of continuous random variables that have a joint distribution is also defined using (1.43).*

Definition 1.3 (Conditional differential entropy)**.** *Let X and Y be continuous random variables that have a joint probability density function $p(x,y)$. The conditional differential entropy $h(X|Y)$ is defined as*

$$h(X|Y) = -\int p(x,y) \log p(x|y) dx dy. \tag{1.44}$$

Definition 1.4 (Relative entropy (KLD))**.** *The relative entropy (KLD) between two probability density functions p and q is defined as*

$$D(p||q) = \int p \log \frac{p}{q}. \tag{1.45}$$

Definition 1.5 (Mutual information)**.** *The mutual information $I(X;Y)$ between two continuous random variables X and Y with joint probability density function $p(x,y)$ is defined as*

$$I(X;Y) = \int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dx dy. \tag{1.46}$$

1.8 Further Readings

Here I list some of the good references for background knowledge. Of course, there are many other notable books and resources available. For probability and statistics see Papoulis and Pillai (2002). For optimal filtering including Kalman filtering, linear systems, and estimation see Anderson and Moore (1979). For a classic textbook on estimation with a focus on target tracking and navigation that also covers an introduction to probability theory see Bar-Shalom et al. (2001).

Problems

1.1 Uncorrelated but not independent. Let $X \sim \mathcal{N}(0,1)$ and $Y = X^2$. Show that X and Y are not independent, but are uncorrelated. Hint: $\mathbb{E}[X] = \mathbb{E}[X^3] = 0$.

1.2 Show that uncorrelated Gaussian random vectors are independent. Hint: In (1.16), if x and y are uncorrelated, then $C = 0$; show $p(x, y) = p(x)p(y)$.

1.3 Derivation of canonical parametrization. Using direct calculation, show that the canonical parametrization of a multivariate normal distribution has the following form and find the constant γ.

$$\mathcal{N}^{-1}(x; \eta, \Lambda) = \gamma \exp(-\frac{1}{2} x^\mathsf{T} \Lambda x + x^\mathsf{T} \eta) \tag{1.47}$$

1.4 Prove Proposition 1.2.

1.5 ML estimate of n Gaussian samples. Find the ML estimator of a sample of n i.i.d. normal random variables (univariate).

1.6 MAP estimate of n Gaussian samples. Find the MAP estimator of a sample of n i.i.d. normal random variables (univariate). Assume the prior is also Gaussian with mean μ_0 and variance σ_0.

1.7 Polynomial regression. Formulate the polynomial regression problem of order n using the LS estimator. Write a function that takes data and the order of the polynomial, and outputs the coefficients. Compare your results with MATLAB's built-in polyfit. What can you say about the goodness of fit? What happens when the order of the polynomial is exactly equal to the number of data points?

2
Uncertainty Propagation

We learned that if a random vector x follows a multivariate Gaussian distribution $x \sim \mathcal{N}(\mu, \Sigma)$ and we are given an affine transformation such as $y = Ax + b$, then $y \sim \mathcal{N}(A\mu + b, A\Sigma A^\mathsf{T})$.

The central question is how to map a probability distribution through a nonlinear function. In general, transformations can be nonlinear in problems we encounter. As such, even if the initial distribution is Gaussian, the transformed variables will become non-Gaussian. We often lack exact solutions for such nonlinear non-Gaussian propagation problems.

Three common methods in practice are based on (Thrun et al., 2005)

1. linearization via Taylor expansion that leads to the Extended Kalman Filter (Chapter 6);
2. and unscented transform (deterministic sampling) that leads to the Unscented Kalman Filter (Chapter 6).
3. Monte Carlo methods (random sampling) [1] that leads to Sequential Monte Carlo methods or Particle Filters (Chapter 8).

[1] https://en.wikipedia.org/wiki/Monte_Carlo_integration

2.1 Linearization

Linearization of a continuously differentiable (and analytic [2]) function $f : \mathbb{R}^n \to \mathbb{R}^m$ around point a is

[2] https://en.wikipedia.org/wiki/Analytic_function

$$
\begin{aligned}
f(x) &\approx f(a) + \frac{\partial f}{\partial x}\Big|_{x=a}(x - a) = (f(a) - \frac{\partial f}{\partial x}\Big|_{x=a} a) + \frac{\partial f}{\partial x}\Big|_{x=a} x \\
&=: x_0 + Fx.
\end{aligned}
$$

This is an affine map, and we know how to propagate a Gaussian distribution through an affine map.

In the context of Kalman filtering, this method leads to the Extended Kalman Filter (EKF) algorithm. We will also use linearization for solving nonlinear optimization problems. In general, engineering methods heavily rely on local linearizations of nonlinear problems as a convenient way to obtain a solution.

2.2 Unscented Transform: Deterministic Sampling

The unscented transform uses deterministic samples to propagate a Gaussian distribution through a nonlinear map. It consists of the following steps.

1. Compute a set of sigma points (samples) $\mathcal{X}$;
2. Each sigma point has a weight w;
3. Transform the point through the nonlinear function $g(x)$;
4. Compute a Gaussian distribution from weighted points using weighted sample mean and covariance.

The first sigma point is the mean, and the other $2n$ sigma points are generated using the columns of the scaled Cholesky factor of the covariance.

$$\begin{aligned} x_0 &= \mu, \\ x_i &= \mu + \ell'_i \quad i = 1, \dots, n, \\ x_i &= \mu - \ell'_{i-n} \quad i = n+1, \dots, 2n. \end{aligned} \tag{2.1}$$

ℓ'_i is the i-th column of L' where $L' = \sqrt{(n+\kappa)}L$ and $\Sigma = LL^\mathsf{T}$ can be computed using the Cholesky decomposition. n is the dimension of the state and κ is a user-definable parameter.

Given a nonlinear map $g : \mathbb{R}^n \to \mathbb{R}^m$, we compute a Gaussian distribution using the weights and transformed points.

$$\mu' = \sum_{i=0}^{2n} w_i g(x_i), \quad \Sigma' = \sum_{i=0}^{2n} w_i (g(x_i) - \mu')(g(x_i) - \mu')^\mathsf{T}. \tag{2.2}$$

$$w_0 = \frac{\kappa}{n+\kappa}, \quad w_i = \frac{1}{2(n+\kappa)} \quad i = 1, \dots, 2n. \tag{2.3}$$

The user-defined parameter, κ, can be tuned to adjust the weight for a particular transformation. For instance $\kappa = 2$; see Barfoot (2017, Ch. 4.2.7.).

Remark 2.1. *If the noise is additive, we simply add the noise covariance to the propagated sample covariance. If the noise is multiplicative, we augment the state with noise by adding zeros of appropriate dimension to the mean and constructing a block-diagonal covariance matrix of the state and noise covariances. Note that in the latter case the dimension of the augmented state is increased to the sum of the dimensions of the state and noise vectors; hence, more samples need to be drawn. See Barfoot (2017, Ch. 4.2.9.).*

Example 2.1 (Unscented Transform)**.** *Transform a Gaussian distribution from polar to Cartesian coordinates.*

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix}, \qquad \begin{bmatrix} r \\ \theta \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1.5 \\ \pi/6 \end{bmatrix}, \begin{bmatrix} 0.1^2 & -0.09^2 \\ -0.09^2 & 0.6^2 \end{bmatrix} \right)$$

Figure 2.1 shows the result before and after applying the unscented transform to the given distribution. See `unscented_transform_example.m` *(or Python version) for code.*

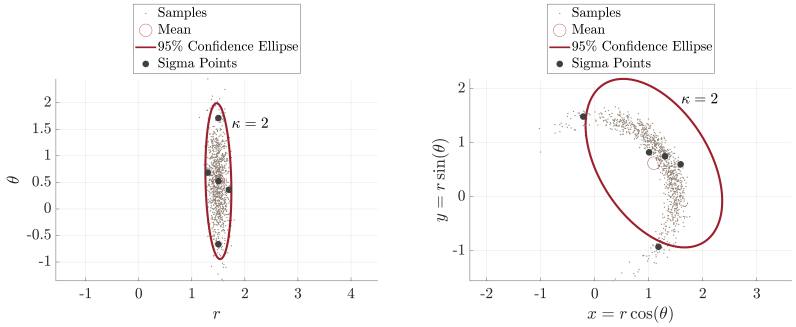

Figure 2.1: Left: Given distribution in the polar coordinates. Right: Transformed distribution using the unscented transform.

2.3 *Monte Carlo: Random Sampling*

See Chapter 8 for related materials to this course. Here the idea is to draw random samples from the distribution and evaluate the function to obtain random samples in the range (output). This way, we have constructed a histogram which is a discrete approximation of the distribution we wanted to map.

The problem is that, in many cases, direct sampling is difficult. Specifically, we can evaluate the probability density function, but we cannot necessarily draw samples from it. This problem leads to algorithms known as Markov Chain Monte Carlo (MCMC) methods (Andrieu et al., 2003; Brooks et al., 2011) [3].

[3] For a few examples of MCMC using a banana distribution in MATLAB and Python, see `https://github.com/UMich-CURLY-teaching/UMich-ROB-530-public/tree/main/code-examples`

Problems

2.1 Explain how a probability distribution can be propagated through a nonlinear function. Elaborate on the methods and reason why they work.

2.2 A range finder sensor provides noisy measurements $z \sim \mathcal{N}(\mu_z, \sigma_z^2)$. A corrected model for calibrating the sensor is $y = az + b$. Derive the distribution over y. Is the noisy model for y an approximation or exact? Why?

2.3 A projection model from the 3D Cartesian space, $\mathbb{R}^3$, to a 2D image is given by

$$p = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix} \in \mathbb{R}^2, \quad \text{and} \quad l = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3$$

where f_x and f_y are focal lengths in x and y directions, respectively, and c_x and c_y are the coordinates of optical center of the image in pixels.

Suppose a 3D landmark $l \in \mathbb{R}^3$ is observed via a noisy sensor, i.e., $l \sim \mathcal{N}(\mu_l, \Sigma_l)$. Derive an analytical model for the mean and covariance of p. Show work and provide necessary equations.

3
Rigid Body Motion

A rigid motion of an object is a motion that preserves the distance between points. The study of robot kinematics, dynamics, and control has at its heart the study of the motion of rigid objects (Murray et al., 1994). We define a rigid body as a collection of particles such that the distance between any two particles remains fixed, regardless of any motions of the body or forces exerted on the body.

A rigid motion of an object is a continuous movement of the particles in the object such that the distance between any two particles remains fixed at all times. The net movement of a rigid body from one location to another via a rigid motion is called rigid displacement. In general, a rigid displacement may consist of both *translation* and *rotation* of the object.

3.1 Rotational Motion in $\mathbb{R}^3$

The three-dimensional rotational motion describes the relative orientation between a coordinate frame attached to the body and a fixed or inertial coordinate frame. All coordinate frames will be right-handed unless stated otherwise.

Let A be the *inertial frame* and B the *body frame*, and $x_{ab}, y_{ab}, z_{ab} \in \mathbb{R}^3$ be the coordinates of the principal axes of B relative to A. Define the 3×3 matrix $R_{ab} = \begin{bmatrix} x_{ab} & y_{ab} & z_{ab} \end{bmatrix}$. Then R_{ab} is a *rotation matrix*. It is easy to check that $R_{ab}R_{ab}^{\mathsf{T}} = R_{ab}^{\mathsf{T}}R_{ab} = I$ because the columns of R_{ab} are mutually orthonormal. It also follows that using the scalar triple product, we have

$$\det R_{ab} = x_{ab} \cdot (y_{ab} \times z_{ab}) = x_{ab} \cdot x_{ab} = 1.$$

The familiar rotation matrices about each axis are:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix},$$

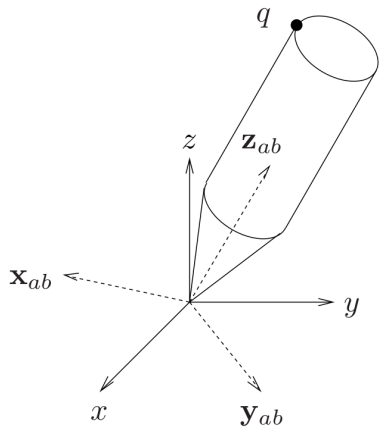

Figure 3.1: Coordinate frame definition for describing rotational motion in $\mathbb{R}^3$.

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix},$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Furthermore, we can combine them using

$$R_{zyx}(\theta, \beta, \alpha) = R_z(\theta) R_y(\beta) R_x(\alpha).$$

Remark 3.1 (Gimbal lock)**.** [1] *When* $\beta = \frac{\pi}{2}$,

$$R_{zyx}(\theta, \frac{\pi}{2}, \alpha) = \begin{bmatrix} 0 & 0 & 1 \\ \sin(\theta+\alpha) & \cos(\theta+\alpha) & 0 \\ -\cos(\theta+\alpha) & \sin(\theta+\alpha) & 0 \end{bmatrix},$$

and θ *and* α *correspond to the same rotation. This is a topological constraint and shows the need for better representations, such as rotation matrices.*

[1] https://en.wikipedia.org/wiki/Gimbal_lock

Remark 3.2 (Rotation topological constraint)**.** *For a rotation by an angle* $\theta = \pi$ *about some axis* e [2]*, both* $R_e(\pi)$ *and* $R_{-e}(\pi)$ *correspond to the same rotation.*

[2] https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation

3.1.1 Properties of Rotation Matrices

We can summarize the properties of a rotation matrix as follows.

- Let $R \in \mathbb{R}^{3\times3}$ be a rotation matrix.
- Then $RR^\mathsf{T} = R^\mathsf{T}R = I$; hence, $R^{-1} = R^\mathsf{T}$. Recall the inverse of a matrix if it exists, it is unique.
- We have $\det R = 1$ (i.e., rotation is a volume-preserving transformation). Note that $\det RR^\mathsf{T} = \det R \det R^\mathsf{T} = (\det R)^2 = \det I = 1$; thus, $\det R = \pm 1$. But as we showed earlier, using the scalar triple product, the right-handed coordinate frame leads to $\det R = +1$.

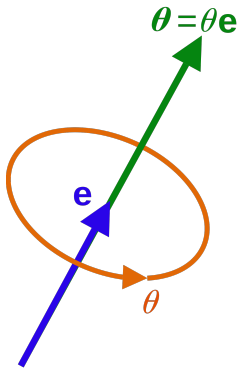

Figure 3.2: Rotation by an angle θ about the axis e.

3.1.2 Group of 3D Rotation Matrices

We can create a set of all possible 3D rotation matrices which we call SO(3) and define as follows.

$$\mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times3} : RR^\mathsf{T} = I \text{ and } \det R = 1\}.$$

We refer to SO(3) as the rotation group of $\mathbb{R}^3$. Every configuration of a rigid body that is free to rotate relative to a fixed frame can be identified with a unique $R \in \mathrm{SO}(3)$.

Problem 3.1. *Verify that* SO(3) *is a group under the matrix multiplication operation. Hint: write down group* [3] *properties and verify then.*

[3] https://en.wikipedia.org/wiki/Group_(mathematics)#Definition

3.1.3 Composition Rule for Rotations

Rotation matrices can be combined using matrix multiplication. Let R_{bc} be the orientation of a frame C relative to another frame B, and R_{ab} be the orientation of the frame B relative to another frame A. Then $R_{ac} = R_{ab}R_{bc}$ is the orientation of frame C relative to A.

Rotations about the same axis commute. We can verify this by setting $R_1 = R_z(\theta_1)$ and $R_2 = R_z(\theta_2)$. Then $R_1R_2 = R_z(\theta_1 + \theta_2) = R_2R_1$. This is similar to rotation in the 2D plane. In general, the order of composition matters because matrix multiplication is noncommutative. For arbitrary R_1 and R_2, $R_1 + R_2$ is not a valid rotation matrix. Mathematically, SO(3) is not closed under the addition.

3.1.4 Action of SO(3) on $\mathbb{R}^3$

Considered R_{ac} as a map from $\mathbb{R}^3$ to $\mathbb{R}^3$, i.e., $R_{ac} : \mathbb{R}^3 \to \mathbb{R}^3$. This map rotates the coordinates of a point from frame C to frame A. We call this the action of SO(3) on $\mathbb{R}^3$.

3.2 Rigid Motion in $\mathbb{R}^3$

In general, rigid motions consist of rotation and translation. We describe the position and orientation of a coordinate frame B attached to the body relative to an inertial frame A as $g_{ab} = (p_{ab}, R_{ab})$ where $p_{ab} \in \mathbb{R}^3$ and $R_{ab} \in \text{SO(3)}$.

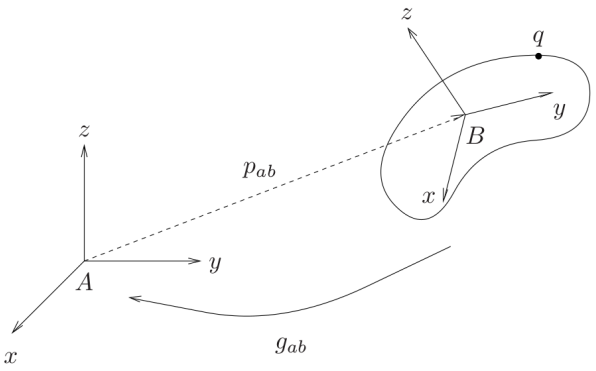

Figure 3.3: Coordinate frame definition for describing rotational motion in $\mathbb{R}^3$.

3.2.1 Group of 3D Rigid Body Transformations

The special Euclidean group is the group of rigid body transformations defined as

$$\text{SE(3)} = \{(p, R) : p \in \mathbb{R}^3 \text{ and } R \in \text{SO(3)}\}.$$

Action of $g \in \text{SE(3)}$ on $\mathbb{R}^3$ is $g(q) = p + Rq$ for $q \in \mathbb{R}^3$.

3.2.2 Homogeneous Representation

The transformation of points and vectors by rigid transformations has a simple representation in terms of matrices and vectors in $\mathbb{R}^4$. We append 1 to the coordinates of a point to yield a vector in $\mathbb{R}^4$, i.e.,

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & 1 \end{bmatrix}^\mathsf{T}.$$

These are called the homogeneous coordinates of the point q. Vectors, which are the difference of points, then have the form

$$v = \begin{bmatrix} v_1 & v_2 & v_3 & 0 \end{bmatrix}^\mathsf{T}.$$

When working with the homogeneous representation, the following rules hold.

1. Sums and differences of vectors are vectors;
2. The sum of a vector and a point is a point;
3. The difference between two points is a vector;
4. The sum of two points is meaningless.

The 4×4 matrix H is called the homogeneous representation of $g \in \mathrm{SE}(3)$. If $g = (p, R) \in \mathrm{SE}(3)$, then

- $H = \begin{bmatrix} R & p \\ 0_{1\times 3} & 1 \end{bmatrix}$;
- $H_1 H_2 = \begin{bmatrix} R_1 & p_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & p_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & p_1 + R_1 p_2 \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(3)$;
- $H^{-1} = \begin{bmatrix} R^\mathsf{T} & -R^\mathsf{T} p \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(3)$;
- $HH^{-1} = H^{-1}H = I_4 \in \mathrm{SE}(3)$.

3.3 Further Readings

For more readings on the topic of rigid body motion see Murray et al. (1994, Chapter 2) [4] and Barfoot (2017, Chapter 6) [5].

[4] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994

[5] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017

Problems

3.1 If $R_1, R_2 \in \mathrm{SO}(3)$, why we cannot combine them using $R_1 + R_2$?

3.2 If $R_1, \ldots, R_n \in \mathrm{SO}(3)$, prove that $R_1 \cdot R_2 \cdot \ldots \cdot R_n \in \mathrm{SO}(3)$.

4
Matrix Lie Groups for Robotics

In this chapter, we look into more general mathematical objects called matrix Lie groups (Chirikjian, 2011; Tapp, 2016; Hall, 2015) that make rotation and rigid body transformation groups particular instances. Lie groups are the best tools for studying the continuous symmetry of geometric spaces, such as invariance and equivariance. Not surprisingly, because Lie groups provide accurate geometric models, they have extensive applications in robotics, computer vision, and artificial intelligence.

Let us first define the group formally. Then we study the general linear group as the largest matrix group and gradually specialize it into other groups, including SO(3) and SE(3).

Definition 4.1 (Group)**.** *A group is a nonempty set $\mathcal{G}$ together with a binary group operation $\cdot$, e.g., $g \cdot h$ where $g, h \in \mathcal{G}$, that satisfies the following properties:*

1. ***Closure:*** *if $g, h \in \mathcal{G}$ then also $g \cdot h \in \mathcal{G}$;*
2. ***Associativity:*** *for all $g, h, l \in \mathcal{G}$, $(g \cdot h) \cdot l = g \cdot (h \cdot l)$;*
3. ***Identity:*** *there exist a unique identity element $e \in \mathcal{G}$ such that $e \cdot g = g \cdot e = g$ for all $g \in \mathcal{G}$;* [1]
4. ***Inverse:*** *if $g \in \mathcal{G}$ there exists an element $g^{-1} \in \mathcal{G}$ such that $g^{-1} \cdot g = g \cdot g^{-1} = e$.*

[1] For matrix groups, we also show the identity element using the identity matrix as I.

Example 4.1. *Show that $(\mathbb{R}, +)$ (the set of real numbers under the addition) and $(\mathbb{R}\backslash\{0\}, \cdot)$ (the set of real numbers excluding 0 under the scalar multiplication) are groups.*

4.1 Matrix Groups

In general, we can work with the set of all m by n matrices with entries in $\mathbb{R}$ denoted $\mathrm{M}_{m,n}(\mathbb{R})$. A matrix group is a group of invertible matrices. This is a simple and purely algebraic definition.

For matrix groups, the group binary operation is the ordinary matrix multiplication. Recall that matrix multiplication is not commutative, i.e., $AB \neq BA$; therefore, matrix groups are generally noncommutative.

Definition 4.2 (General Linear Groups)**.** *The* general linear group *over* $\mathbb{R}$ *is:*

$$\mathrm{GL}_n(\mathbb{R}) = \{A \in \mathrm{M}_n(\mathbb{R}) : \det(A) \neq 0\}.$$

The n-dimensional *affine* [2] *group* over $\mathbb{R}$ is

[2] https://en.wikipedia.org/wiki/Affine_transformation

$$\mathrm{Aff}_n(\mathbb{R}) = \left\{ \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} : A \in \mathrm{GL}_n(\mathbb{R}),\ t \in \mathbb{R}^n \right\}.$$

If we identify $x \in \mathbb{R}^n$ with $\begin{bmatrix} x \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}$, then as a consequence of the formula

$$\begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} Ax + t \\ 1 \end{bmatrix}$$

we obtain an action of $\mathrm{Aff}_n(\mathbb{R})$ on $\mathbb{R}^n$. The vector space $\mathbb{R}^n$ itself can be viewed as the *translation subgroup* of $\mathrm{Aff}_n(\mathbb{R})$,

$$\mathrm{Trans}_n(\mathbb{R}) = \left\{ \begin{bmatrix} I_n & t \\ 0 & 1 \end{bmatrix} : t \in \mathbb{R}^n \right\} \subseteq \mathrm{Aff}_n(\mathbb{R}),$$

and this is a closed subgroup.

Definition 4.3 (Orthogonal Groups)**.** *The* orthogonal group *over* $\mathbb{R}$ *is denoted* $\mathrm{O}(n)$ *and defined as:*

$$\mathrm{O}(n) = \{A \in \mathrm{GL}_n(\mathbb{R}) : A \cdot A^{\mathsf{T}} = I_n\},$$

where "$\cdot$" denotes the standard matrix multiplication as the group operation and is dropped hereafter, i.e., AA^{T}.

Looking closer into the orthogonal group, we see that $\det(AA^{\mathsf{T}}) = \det(A)^2 = \det(I_n) = 1$; therefore, $\det(A) = \pm 1$. Thus we have $\mathrm{O}(n) = \mathrm{O}(n)^+ \cup \mathrm{O}(n)^-$ where

$$\mathrm{O}(n)^+ = \{A \in \mathrm{O}(n) : \det(A) = 1\},$$

$$\mathrm{O}(n)^- = \{A \in \mathrm{O}(n) : \det(A) = -1\}.$$

Notice that $\mathrm{O}(n)^+ \cap \mathrm{O}(n)^- = \varnothing$, so $\mathrm{O}(n)$ is the *disjoint union* of the subsets $\mathrm{O}(n)^+$ and $\mathrm{O}(n)^-$. The important subgroup $\mathrm{SO}(n) = \mathrm{O}(n)^+ \leq \mathrm{O}(n)$ is the $n \times n$ *special orthogonal group.*

4.1.1 Isometry Groups

One of the main reasons for the study of the orthogonal groups $\mathrm{O}(n)$ and $\mathrm{SO}(n)$ is their relationships with *isometries* [3], where an isometry of $\mathbb{R}^n$ is a distance-preserving bijection $f : \mathbb{R}^n \to \mathbb{R}^n$, i.e.,

$$\|f(x) - f(y)\| = \|x - y\| \qquad x, y \in \mathbb{R}^n$$

If such an isometry fixes the origin, 0, then it is a *linear transformation*, often referred to as *linear isometry*, and so with respect with the standard basis it corresponds to a matrix $A \in \mathrm{GL}_n(\mathbb{R})$.

Remark 4.1. *The special orthogonal group* $\mathrm{SO}(n)$ *is the simultaneous rotation of n perpendicular planes!*

Elements of $\mathrm{SO}(n)$ often called *direct isometries* or *rotations*, while elements of $\mathrm{O}(n)^-$ are sometimes called *indirect isometries* [4]. We define the full *isometry group* of $\mathbb{R}^n$ as

$$\mathrm{Isom}_n(\mathbb{R}) = \{f : \mathbb{R}^n \to \mathbb{R}^n : f \text{ is an isometry}\},$$

which contains the subgroup of translations. In fact, $\mathrm{Isom}_n(\mathbb{R}) \subseteq \mathrm{Aff}_n(\mathbb{R})$ and is actually a closed subgroup, hence is a matrix subgroup,

$$\mathrm{Isom}_n(\mathbb{R}) = \left\{ \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} : A \in \mathrm{O}(n),\ t \in \mathbb{R}^n \right\}.$$

The *special Euclidean group* is the isometry group that requires A to be a valid right-handed rotation matrix:

$$\mathrm{SE}(n) = \left\{ \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} : A \in \mathrm{SO}(n),\ t \in \mathbb{R}^n \right\}.$$

This is the group of valid rigid body transformations of $\mathbb{R}^n$.

4.1.2 Symmetry Groups

The *symmetry group* of a subset $\mathcal{X} \subset \mathbb{R}^n$ is the group of all isometries of $\mathbb{R}^n$ that carry $\mathcal{X}$ onto itself.

Definition 4.4. $\mathrm{Symm}(\mathcal{X}) = \{f \in \mathrm{Isom}(\mathbb{R}^n) : f(\mathcal{X}) = \mathcal{X}\}$.

The statement "$f(\mathcal{X}) = \mathcal{X}$" means that each point of $\mathcal{X}$ is sent by f to a (possibly different) point of $\mathcal{X}$. For example, consider the sphere $\mathcal{S}^{n-1} \subset \mathbb{R}^n$:

$$\mathcal{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\| = r\}.$$

The symmetry group of $\mathcal{S}^{n-1}$ equals the group of isometries of $\mathbb{R}^n$ with no translational component, which is isomorphic to the orthogonal group:

$$\mathrm{Symm}(\mathcal{S}^{n-1}) = \left\{ \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} : A \in \mathrm{O}(n),\ t = 0 \right\} \cong \mathrm{O}(n).$$

[3] In the previous chapter, we defined this as a rigid (distance-preserving) motion of an object.

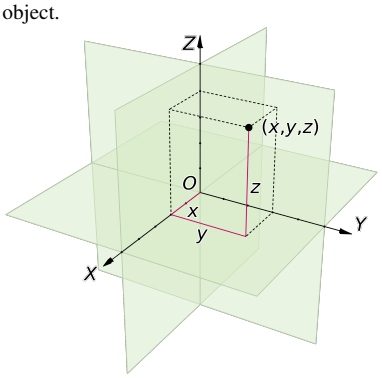

Figure 4.1: For example, SO(3) is the rotation group of $\mathbb{R}^3$ and defines the simultaneous rotation of three perpendicular planes which construct the three-dimensional (3D) Euclidean space.

[4] The elements of $\mathrm{O}(n)^-$ corresponds to reflection maps.

The following definition of *proper* and *improper* symmetries of a set $\mathcal{X}$ is a general principle.

Definition 4.5. $\mathrm{Symm}(\mathcal{X}) = \mathrm{Symm}^{+}(\mathcal{X}) \cup \mathrm{Symm}^{-}(\mathcal{X})$*, where the sets*

$$\mathrm{Symm}(\mathcal{X})^{\pm} = \left\{ \begin{bmatrix} A & t \\ 0 & 1 \end{bmatrix} \in \mathrm{Symm}(\mathcal{X}) : \det(A) = \pm 1 \right\}$$

are called the proper *and* improper *symmetry group of* $\mathcal{X}$.

4.2 Manifolds and Tangent Spaces

Manifolds are higher-dimensional analogues of smooth curves and surfaces (Tu, 2011). A matrix group is an algebraic object. However, it can also be seen as a geometric object since it is a subset of a Euclidean space:

$$\mathcal{G} \subset \mathrm{GL}_n(\mathbb{R}) \subset \mathrm{M}_n(\mathbb{R}) \cong \mathbb{R}^{n^2}.$$

In addition, looking at a matrix group as a subset of a Euclidean space means we can discuss its tangent space.

To study the geometry of a manifold, we need the notion of a tangent space. Let $\gamma(t)$ be some curve in some manifold M, then its derivative $\frac{d}{dt}\gamma(t) = \gamma'(t)$ is a *tangent vector*. If $x \in M$ is a point in the manifold, then the space of all possible tangent vectors is called the *tangent space* and is denoted by T_xM. It is important to point out that T_xM is a vector space and

$$\dim T_xM = \dim M.$$

This means the dimension of a manifold is defined by the dimension of its tangent space as a vector space. For example, if

$$M = \{(x, y, z) \in \mathbb{R}^3 : z = f(x, y)\} \subset \mathbb{R}^3$$

is a surface, then it defines a 2D manifold because any tangent plane to the surface can define a 2D vector space identified as $\mathbb{R}^2$.

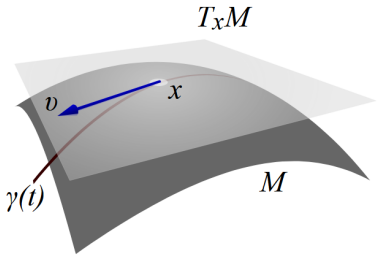

Figure 4.2: Illustration of manifold M and the tangent vector at point x.

Definition 4.6 (Tangent space)**.** *Let* $\mathcal{G} \subset \mathbb{R}^m$ *be a subset, and let* $g \in \mathcal{G}$*. The* tangent space *to* $\mathcal{G}$ *at* g *is:*

$$T_g\mathcal{G} = \left\{ \gamma'(0) : \gamma : (-\epsilon, \epsilon) \to \mathcal{G} \text{ is differentiable with } \gamma(0) = g \right\}.$$

$T_g\mathcal{G}$ means the set of initial velocity vectors of differentiable paths through g in $\mathcal{G}$. The term *differentiable* means that, when we consider γ as a path in $\mathbb{R}^m$, the m components of γ are differentiable functions from $(-\epsilon, \epsilon)$ to $\mathbb{R}$.

4.3 Lie Algebras

We are now ready to discuss the tangent spaces of matrix groups. Since all groups contain at least the identity element, we give a special name to the tangent space at the identity.

Definition 4.7 (Lie algebra)**.** *The* Lie algebra *of a matrix group $\mathcal{G} \subset \mathrm{GL}_n(\mathbb{R})$ is the tangent space to $\mathcal{G}$ at the identity e. It is denoted $\mathfrak{g} = \mathfrak{g}(\mathcal{G}) = T_e\mathcal{G}$.*

In particular, $\mathfrak{g}$ is a subspace of the Euclidean space $\mathrm{M}_n(\mathbb{R})$, which shows that matrix groups are "nice" sets.

Definition 4.8. *The* dimension *of a matrix group $\mathcal{G}$ means the dimension of its Lie algebra.*

Let us consider the Lie algebra of $\mathrm{GL}_n(\mathbb{R})$, denoted $\mathfrak{gl}_n(\mathbb{R})$.

Proposition 4.1. $\mathfrak{gl}_n(\mathbb{R}) = \mathrm{M}_n(\mathbb{R})$. *In particular,* $\dim \mathrm{GL}_n(\mathbb{R}) = n^2$.

Proof. Let $A \in \mathrm{M}_n(\mathbb{R})$. The path $\gamma(t) = I + t.A$ in $\mathrm{M}_n(\mathbb{R})$ satisfies $\gamma(0) = I$ and $\gamma'(0) = A$. Also, γ restricted to sufficiently small interval $(-\epsilon, \epsilon)$ lies in $\mathrm{GL}_n(\mathbb{R})$. To justify this, notice $\det(\gamma(0)) = 1$. Since the determinant function is continuous, $\det(\gamma(t))$ is close to 1 (and is, therefore, non-zero) for t close to 0. This demonstrates that $A \in \mathfrak{gl}_n(\mathbb{R})$. □

The set $\mathfrak{so}(n) = \{A \in \mathrm{M}_n(\mathbb{R}) : A + A^\mathsf{T} = 0\}$ is the Lie algebra of $\mathrm{SO}(n)$ and contains *skew-symmetric* matrices [5]. To see this, consider the path $\gamma(t) \in \mathrm{SO}(n)$ that satisfies $\gamma(0) = I$ and $\gamma'(0) = A$. For every matrix in $\mathrm{SO}(n)$, we have $\gamma(t) \cdot \gamma(t)^\mathsf{T} = I$. Using the product rule to differentiate both sides of

$$\gamma(t) \cdot \gamma(t)^\mathsf{T} = I$$

gives $\gamma'(0) + \gamma'^\mathsf{T}(0) = A + A^\mathsf{T} = 0$, so A must be skew-symmetric.

[5] https://en.wikipedia.org/wiki/Skew-symmetric_matrix

Corollary 4.2. $\dim \mathrm{SO}(n) = \frac{n(n-1)}{2}$.

Proof. Skew-symmetric matrices have zeros on the diagonal, arbitrary real numbers above, and entries below determined by those above, so $\dim \mathfrak{so}(n) = \frac{n(n-1)}{2}$. □

Previously, we assumed the path $\gamma(t)$ goes through the group identity I, i.e., $\gamma(0) = I$. The natural question is that, if $\gamma(0) = g$ where $g \in \mathcal{G}$, what is $T_g\mathcal{G}$ (tangent space at g) and its relationship with $\mathfrak{g} = T_e\mathcal{G}$?

The existence of the group multiplication map $\phi : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ [6] allows one to define left translation $\ell_g : \mathcal{G} \to \mathcal{G}$ for any point $g \in \mathcal{G}$. This map is a diffeomorphism with inverse $\ell_{g^{-1}}$. Left translation ℓ_g can take the identity element I to the element g, i.e., $g \cdot I \mapsto g$, and induces an isomorphism of tangent spaces (Tu, 2011) [7]

$$(\ell_g)_* : \mathfrak{g} \to T_g\mathcal{G}.$$

[6] The matrix multiplication via group elements which gives another group element.

[7] Its differential gives rise to an isomorphism of tangent spaces $(\ell_g)_* : \mathfrak{g} \xrightarrow{\sim} T_g\mathcal{G}$. The notation $(\cdot)_*$ denotes the pushforward map.

For a general matrix group $\mathcal{G}$, we have

$$T_g\mathcal{G} = g \cdot \mathfrak{g}.$$

This means if we can describe the tangent space $\mathfrak{g}$ at the identity, then $(\ell_g)_*\mathfrak{g} = g \cdot \mathfrak{g}$ will describe the tangent space $T_g\mathcal{G}$ at any point $g \in \mathcal{G}$.

Remark 4.2. *Note that for $\gamma(t) \in \mathcal{G}$, $\gamma(0) = I$ and $\gamma'(0) = A \in \mathfrak{g}$, via a left translation for a constant $g \in \mathcal{G}$ we have $g \cdot \gamma(t)$. Differentiating this gives $g \cdot \gamma'(t)\Big|_{t=0} = g \cdot A =: \dot{g} \in T_g\mathcal{G}$. Hence, we arrive at the following first-order differential equation known as the* reconstruction equation.

$$\boxed{\dot{g} = g \cdot A \quad or \quad g^{-1}\dot{g} = A \in \mathfrak{g}}. \tag{4.1}$$

Remark 4.3. *A similar argument can be made for the right translation $r_g : \mathcal{G} \to \mathcal{G}$ for any point $g \in \mathcal{G}$ to get $\gamma'(t)\Big|_{t=0} \cdot g = A \cdot g =: \dot{g} \in T_g\mathcal{G}$.*

$$\boxed{\dot{g} = A \cdot g \quad or \quad \dot{g}g^{-1} = A \in \mathfrak{g}}. \tag{4.2}$$

Example 4.2. *Let γ be a curve in $\mathrm{SO}(n)$ such that $\gamma(0) = g$. Then we know that $\dot{\gamma}(0) \in T_g\mathrm{SO}(n)$.*

$$T_g\mathrm{SO}(n) = \left\{gA : A^\mathsf{T} = -A\right\} = g \cdot \mathfrak{so}(n).$$

4.4 The "Best" Path in a Matrix Group

We now ask what is the "best" choice of path $\gamma(t) \in \mathcal{G}$ in matrix groups? With the reconstruction equation (4.1) or (4.2) in hand, we can look into the solution of these differential equations.

4.4.1 Matrix Exponentiation; Series in $\mathrm{M}_n(\mathbb{R})$

When the power series of the function $f(x) = \exp(x)$ is applied to a matrix $A \in \mathrm{M}_n(\mathbb{R})$, the result is called *matrix exponentiation* [8]:

[8] https://en.wikipedia.org/wiki/Matrix_exponential

$$\exp(A) = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4 + \cdots.$$

This series converges for all $A \in \mathrm{M}_n(\mathbb{R})$.

4.4.2 The "Most Natural" Path in $\mathrm{GL}_n(\mathbb{R})$

If $A \in \mathfrak{gl}_n(\mathbb{R})$, we would like to find the "most natural" path $\gamma(t)$ in $\mathrm{GL}_n(\mathbb{R})$ with $\gamma(0) = I$ and $\gamma'(0) = A$.

Proposition 4.3. *Let $A \in \mathfrak{gl}_n(\mathbb{R})$. The path $\gamma : \mathbb{R} \to \mathrm{M}_n(\mathbb{R})$ defined as $\gamma(t) = \exp(tA)$ is differentiable, and $\gamma'(t) = A \cdot \gamma(t)$ or $\gamma'(t) = \gamma(t) \cdot A$.*

Proof. Each of the n^2 entries of

$$\gamma(t) = I + tA + \frac{1}{2}t^2A^2 + \frac{1}{6}t^3A^3 + \cdots$$

is a power series in t, which, from familiar real calculus, can be term-wise differentiated, giving:

$$\gamma'(t) = 0 + A + tA^2 + \frac{1}{2}t^2A^3 + \cdots.$$

This equals $\gamma(t) \cdot A$ or $A \cdot \gamma(t)$ depending on whether you factor an A out on the left or right. □

4.4.3 *Solution of the Reconstruction Equation*

The general solution can also be obtained [9]. Starting from (4.1) and differentiating both sides, with a constant $A \in \mathfrak{g}$, we can show by induction the following result.

[9] https://math.stackexchange.com/questions/2683070/how-do-you-solve-dotx-ux/2683265

$$\begin{aligned}
\gamma'(t) &= \gamma(t)A, \quad \gamma(0) = g \in \mathcal{G}, \\
\gamma''(t) &= \gamma'(t)A = \gamma(t)A^2, \\
&\vdots \\
\gamma^{(k)}(t) &= \gamma(t)A^k, \\
\gamma^{(k+1)}(t) &= \gamma(t)A^{k+1}.
\end{aligned} \tag{4.3}$$

Since (4.3) is true for all integers $\gamma^{(n)}(t) = \gamma(t)A^n, n > 0$, we can write the power series of $\gamma(t)$ about $t = 0$ as follows.

$$\gamma(t) = \sum_{n=0}^{\infty} \frac{1}{n!}\gamma^{(n)}(0)t^n = \sum_{n=0}^{\infty} \frac{1}{n!}\gamma(0)A^n t^n = g\exp(tA).$$

Therefore, we conclude that the best path in a matrix group going through $\gamma(0) = g \in \mathcal{G}$ with an initial velocity $A \in \mathfrak{g}$ is obtained via the matrix exponential $\gamma(t) = g\exp(tA)$ [10], which is particularly nice to work with in practice. Note that $\exp(\cdot) : \mathfrak{g} \to \mathcal{G}$.

[10] Or $\gamma(t) = \exp(tA)g$ if we start from (4.2).

4.5 *Body vs. Spatial Velocities*

In $\gamma(t) = g\exp(tA)$, the velocity A is considered to be in the *body frame*, whereas in $\gamma(t) = \exp(tA)g$ the velocity A is in the *spatial frame* (relative to a fixed (inertial) coordinate frame). The relation between the body and spatial velocities motivates us to study the adjoint action in the following.

For convenience, we define the following isomorphism

$$(\cdot)^\wedge : \mathbb{R}^n \to \mathfrak{g}, \tag{4.4}$$

that maps an element in the vector space $\mathbb{R}^n$ to the tangent space of the matrix Lie group at the identity. We also define the inverse of $(\cdot)^\wedge$ map as

$$(\cdot)^\vee : \mathfrak{g} \to \mathbb{R}^n. \tag{4.5}$$

Then, for any $\phi \in \mathbb{R}^n$, we can define the Lie exponential map as [11]

$$\exp(\cdot) : \mathbb{R}^n \to \mathcal{G}, \;\; \exp(\phi) = \exp_{\mathrm{m}}(\phi^\wedge), \tag{4.6}$$

where $\exp_{\mathrm{m}}(\cdot)$ is the exponential of square matrices. We also define the Lie logarithmic map as the inverse of the Lie exponential map [12]

$$\log(\cdot) : \mathcal{G} \to \mathbb{R}^n. \tag{4.7}$$

We will use an example via SO(3) to demonstrate the above claim. We learned that SO(3) is the rotation group of $\mathbb{R}^3$. Its Lie algebra

$$\mathfrak{so}(3) = \{A \in \mathrm{M}_3(\mathbb{R}) : A = -A^\mathsf{T}\}$$

is the space of 3×3 skew-symmetric matrices. These three-dimensional vectors describe the angular velocities [13], i.e., the rotation rate about three axes. In particular, let $\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \in \mathbb{R}^3$ be a vector of angular velocity defined using the $\mathbb{R}^3$ standard basis [14] as $\omega = \omega_1 e_1 + \omega_2 e_2 + \omega_3 e_3$. Using the wedge notation, we have

$$\omega^\wedge = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} =: S \in \mathfrak{so}(3).$$

Obviously, $S^\vee = (\omega^\wedge)^\vee = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$. We can also write

$$\omega^\wedge = \omega_1 G_1 + \omega_2 G_2 + \omega_3 G_3,$$

where $G_1 = e_1^\wedge$, $G_2 = e_2^\wedge$, and $G_3 = e_3^\wedge$. The matrices $G1, G2, G3$ give a basis for vector space $\mathfrak{so}(3)$ and called generators.

Consider frames A and B in Figure 3.1. Let us rename A to S to represent the spatial frame. The coordinates of a point in frame B, p_b, is related to its coordinates in frame S, p_s, using the action of relative rotation R_{sb}, as

$$p_s = R_{sb} p_b.$$

We can also define a vector using the difference of two points as $v_b := p_b - q_b$ and show that

$$v_s = R_{sb} v_b = R_{sb} p_b - R_{sb} q_b = p_s - q_s.$$

Therefore, by inspection, we observe that a vector in the body frame can be mapped to the spatial (inertial or fixed) frame using left multiplication by the orientation of the frame B relative to frame S, i.e., R_{sb}. With this observation, we can relate the angular velocity in the body frame B, ω_b [15], to the angular velocity in the spatial frame S, ω_s [16], via

[11] This is an overloaded notation because previously, for simplicity, we denoted the matrix exponential using $\exp(\cdot)$. In general, this should not be a source of confusion because the matrix exponential cannot be evaluated with a vector as its input (syntax error). As we will see, these notations will make our future derivations easier.

[12] Similarly, we can define the log map of square matrices as $\log_m(\cdot) : \mathcal{G} \to \mathfrak{g}$.

[13] https://en.wikipedia.org/wiki/Angular_velocity

[14] https://en.wikipedia.org/wiki/Standard_basis

[15] This is the angular velocity of the body frame B with respect to the spatial frame S as seen in the body frame.

[16] This is the angular velocity of the body frame B with respect to the spatial frame S as seen in the spatial frame.

$$\omega_s = R_{sb}\omega_b \quad \text{or} \quad \omega_b = R_{sb}^{\mathsf{T}}\omega_s \tag{4.8}$$

We now recall 4.1,

$$\dot{R} = R\omega_b^{\wedge},$$

and consider ω_b to be in the body frame as the rotation is multiplied in the left. The consistency with 4.2 can be proved as follows.

$$\dot{R} = \omega_s^{\wedge}R = RR^{\mathsf{T}}\omega_s^{\wedge}R = R\left(R^{\mathsf{T}}\omega_s^{\wedge}R\right) = R\left(R^{\mathsf{T}}\omega_s\right)^{\wedge} = R\omega_b^{\wedge}. \tag{4.9}$$

In (4.9), we used the following lemma.

Lemma 4.4. *For any $R \in \mathrm{SO}(3)$ and $\omega^{\wedge} \in \mathfrak{so}(3)$, we have*

$$R\omega^{\wedge}R^{\mathsf{T}} = (R\omega)^{\wedge}.$$

Proof. Let $a, b \in \mathbb{R}^3$ and recall that $a \times b = a^{\wedge}b$ which can be verified by direct calculations of both sides. Then

$$\begin{aligned}
(R\omega)^{\wedge}a &= R\omega \times a = R\omega \times RR^{\mathsf{T}}a = R\left(\omega \times R^{\mathsf{T}}a\right) \\
&= R\left(\omega^{\wedge}R^{\mathsf{T}}a\right) = R\omega^{\wedge}R^{\mathsf{T}}a, \\
\implies (R\omega)^{\wedge} &= R\omega^{\wedge}R^{\mathsf{T}}.
\end{aligned}$$

□

In the next section, we will generalize this result to all matrix Lie groups.

4.6 Conjugation, Adjoint, and the Lie Bracket

Let $\mathcal{G}$ be a matrix Lie group with Lie algebra $\mathfrak{g}$. For all $g \in \mathcal{G}$, the *conjugation map* $C_g : \mathcal{G} \to \mathcal{G}$, defined as

$$C_g(a) = gag^{-1}, \tag{4.10}$$

is a smooth isomorphism. When working with matrix groups, the conjugacy [17] relation is called matrix similarity [18]. Similar matrices represent the same linear map under different bases. The conjugation map or matrix similarity is also known as the change of basis formula as we will derive it next.

[17] https://en.wikipedia.org/wiki/Conjugacy_class

[18] https://en.wikipedia.org/wiki/Matrix_similarity

Consider two points $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$ represented in the standard basis $e_1, \cdots, e_n$ and an alternative representation of the same points denoted x' and y' in another basis $e_1', \cdots, e_n'$ such that $e_i = P^{-1}e_i'$ for $i = 1, \cdots, n$, i.e., $x = P^{-1}x'$ and $y = P^{-1}y'$. Let A be a linear map such that $y = Ax$ and T another linear map as $y' = Tx'$. Then A and T are similar via the following change of basis.

$$\begin{aligned}
y &= Ax \\
P^{-1}y' &= AP^{-1}x' \\
y' &= PAP^{-1}x' = Tx, \\
\implies T &= PAP^{-1}.
\end{aligned}$$

The derivative of the conjugation map in (4.10) at the identity $\mathrm{d}(\mathrm{C}_g)_I : \mathfrak{g} \to \mathfrak{g}$ is a vector space isomorphism, which we denote as Ad_g (adjoint):

$$\mathrm{Ad}_g = \mathrm{d}(\mathrm{C}_g)_I.$$

To derive a simple formula for $\mathrm{Ad}_g(B)$, notice that any $B \in \mathfrak{g}$ can be represented as $B = b'(0)$, where $b(t)$ is a differentiable path in $\mathcal{G}$ with $b(0) = I$. The product rule gives:

$$\mathrm{Ad}_g(B) = \mathrm{d}(\mathrm{C}_g)_I(B) = \frac{\mathrm{d}}{\mathrm{d}t}\bigg|_{t=0} gb(t)g^{-1} = gBg^{-1}.$$

So we learn that

$$\mathrm{Ad}_g(B) = gBg^{-1}. \tag{4.11}$$

Note that (4.11) enables us to perform a change of basis of velocities defined in the Lie algebra whereas (4.10) is used for the change of basis of a matrix transformation as a group element (e.g., a rotation or pose).

Example 4.3 (Adjoint of SO(3) and SE(3))**.** *We can derive the adjoint map for each Lie group. We're after a matrix transformation called adjoint that takes a group element and maps an element of the Lie algebra to another element of the Lie algebra. The definition of the adjoint is given as*

$$\mathrm{Ad}_g(\xi^\wedge) = g\xi^\wedge g^{-1},$$

where $\xi^\wedge \in \mathfrak{g}$ and $g \in \mathcal{G}$. This is written in group form and is not giving us a matrix representation directly, so we will derive it for each group.

- SO(3) *Case:*
  Let $\omega^\wedge \in \mathfrak{so}(3)$ and $R \in \mathrm{SO}(3)$. From the definition of the adjoint for SO(3) *we have*

  $$\mathrm{Ad}_R(\omega^\wedge) = R\omega^\wedge R^{-1} = R\omega^\wedge R^\mathsf{T} = (R\omega)^\wedge.$$

  Since we seek a matrix that acts on a vector (ω) the left most term must be of this form $(\mathrm{Ad}_R\omega)^\wedge$. Notice that we're solving for matrix Ad_R and $\mathrm{Ad}_R(\cdot)$ is a function in the group form. So we learn that

  $$(\mathrm{Ad}_R\omega)^\wedge = (R\omega)^\wedge$$
  $$\implies \mathrm{Ad}_R = R.$$

- SE(3) *Case:*
  The elements of the group and its Lie algebra are, respectively,

  $$X = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(3) \quad \textit{and} \quad \xi^\wedge = \begin{bmatrix} \omega^\wedge & v \\ 0 & 0 \end{bmatrix}_{4\times 4} \in \mathfrak{se}(3).$$

  $$(\mathrm{Ad}_X\xi)^\wedge = X\xi^\wedge X^{-1} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega^\wedge & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R^\mathsf{T} & -R^\mathsf{T}p \\ 0 & 1 \end{bmatrix},$$

$$(\mathrm{Ad}_X\xi)^\wedge = \begin{bmatrix} R\omega^\wedge R^\mathsf{T} & -R\omega^\wedge R^\mathsf{T} p + Rv \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} (R\omega)^\wedge & -(R\omega)^\wedge p + Rv \\ 0 & 0 \end{bmatrix},$$

$$(\mathrm{Ad}_X\xi)^\wedge = \begin{bmatrix} (R\omega)^\wedge & p^\wedge R\omega + Rv \\ 0 & 0 \end{bmatrix}.$$

We can choose to stack the linear velocity $v \in \mathbb{R}^3$ and angular velocity $\omega \in \mathbb{R}^3$ in the following orders.

1. $\xi = \mathrm{vec}(\omega, v) \in \mathbb{R}^6$*;*

$$\mathrm{Ad}_X\xi = \begin{bmatrix} R & 0 \\ p^\wedge R & R \end{bmatrix} \begin{bmatrix} \omega \\ v \end{bmatrix},$$

and

$$\mathrm{Ad}_X = \begin{bmatrix} R & 0 \\ p^\wedge R & R \end{bmatrix}.$$

2. $\xi = \mathrm{vec}(v, \omega) \in \mathbb{R}^6$*;*

$$\mathrm{Ad}_X\xi = \begin{bmatrix} R & p^\wedge R \\ 0 & R \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

and

$$\mathrm{Ad}_X = \begin{bmatrix} R & p^\wedge R \\ 0 & R \end{bmatrix}.$$

Mathematical notation $(\cdot)^\wedge$ is an overloaded notation as it is different for example for SO(3) *and* SE(3)*. For* SO(3)*, we have $^\wedge : \mathbb{R}^3 \to \mathfrak{so}(3)$. This means if $\omega \in \mathbb{R}^3$, then $\omega^\wedge \in \mathfrak{so}(3)$ (skew-symmetric matrix). Similarly, for* SE(3)*, we have $(\cdot)^\wedge : \mathbb{R}^6 \to \mathfrak{se}(3)$. This means if $\xi \in \mathbb{R}^6$, then $\xi^\wedge \in \mathfrak{se}(3)$. The inverse operation is defined using $(\cdot)^\vee$ (read vee). That is $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ and $(\cdot)^\vee : \mathfrak{se}(3) \to \mathbb{R}^6$. Then we have $(\omega^\wedge)^\vee = \omega$.*

If we take the derivative one more time when g is itself close to I leads to the Lie bracket definition.

Definition 4.9 (Lie bracket)**.** *The Lie bracket of two vectors A and B in $\mathfrak{g}$ is:*

$$[A, B] = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} \mathrm{Ad}_{a(t)}(B) = AB - BA \in \mathfrak{g}, \tag{4.12}$$

where $a(t)$ is any differentiable path in $\mathcal{G}$ with $a(0) = I$ and $a'(0) = A$. The alternative notation is

$$\mathrm{ad}_A(B) = [A, B].$$

The Lie bracket has the following properties. For all $A, A_1, A_2, B, B_1, B_2, C \in \mathfrak{g}$ and $\lambda_1, \lambda_2 \in \mathbb{R}$,

1. Bilinearity: $[\lambda_1 A_1 + \lambda_2 A_2, B] = \lambda_1[A_1, B] + \lambda_2[A_2, B]$.

2. Bilinearity: $[A, \lambda_1 B_1 + \lambda_2 B_2] = \lambda_1 [A, B_1] + \lambda_2 [A, B_2]$.

3. Anticommutativity: $[A, B] = -[B, A]$.

4. Jacobi identity: $[[A, B], C] + [[B, C], A] + [[C, A], B] = 0$.

Example 4.4 (Lie Bracket on $\mathfrak{so}(3)$ and Cross Product)**.** [19] *The following example shows the equivalency of the cross product and Lie bracket calculations. Note that the Lie bracket is more general and works for all Lie algebras.*

[19] See `so3_cross_example.m` for numerical examples and details.

$$G_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \; G_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \; G_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$[G_1, G_2] = G_3, \; [G_2, G_3] = G_1, \; [G_3, G_1] = G_2$$

$$e_1 \times e_2 = e_3, \; e_2 \times e_3 = e_1, \; e_3 \times e_1 = e_2$$

4.6.1 Baker-Campbell-Hausdorff (BCH) Series

For $X, Y, Z \in \mathfrak{g}$ with sufficiently small norm, the equation

$$\exp(X)\exp(Y) = \exp(Z)$$

has a power series solution for Z in terms of repeated Lie bracket of X and Y. The beginning of the series is:

$$Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] + \frac{1}{12}[Y, [Y, X]] + \cdots.$$

The existence of such a series means that the group operation is completely determined by the Lie bracket operation; the product of $\exp(X)$ and $\exp(Y)$ can be expressed purely in terms of repeated Lie brackets of X and Y.

One important consequence of the Baker-Campbell-Hausdorff series is the correspondence between Lie algebras and matrix groups.

Theorem 4.5 (The Lie correspondences theorem (Tapp, 2016))**.** *There is a natural one-to-one correspondences between sub-algebras of* $\mathfrak{gl}_n(\mathbb{R})$ *and path-connected subgroups of* $\mathrm{GL}_n(\mathbb{R})$.

4.7 Useful Lie Groups and Identities in Robotics

Some of the frequently used Lie groups in the literature are as follows.

- Group of 3D rotation matrices, SO(3); it can model rotations without any singularities or ambiguities.
- Group of direct spatial isometries (3D Rigid Body Transformations), SE(3).

- Group of K direct isometries, $\mathrm{SE}_K(3)$; for example, it is used for modeling IMU sensors and robot pose plus landmarks and/or contact points.
- Group of 3D similarity transformations, Sim(3); it is more general than SE(3) and includes a scale factor and used in monocular vision where the scale is not known.

4.7.1 Group of 3D Rotation Matrices, SO(3)

Let $\phi \in \mathbb{R}^3$ be a vector that can be identified with an element of the Lie algebra, $\phi^\wedge \in \mathfrak{so}(3)$. The corresponding rotation matrix, $R \in \mathrm{SO}(3)$, can be computed using the group's exponential map:

$$R = \exp(\phi) = I + \left(\frac{\sin\theta}{\theta}\right)\phi^\wedge + \left(\frac{1-\cos\theta}{\theta^2}\right)(\phi^\wedge)^2 \quad \in \mathrm{SO}(3),$$

where $\theta := \|\phi\|$. The inverse operation maps the rotation matrix back to a 3×3 skew-symmetric matrix.

$$\phi^\wedge = \log(R) = \frac{\theta}{2\sin\theta}(R - R^\mathsf{T}) \quad \in \mathfrak{so}(3),$$

where $\theta := \cos^{-1}\left(\frac{\mathrm{tr}(R)-1}{2}\right)$

The matrix representation for the adjoint for SO(3) is simply the rotation matrix itself.

$$\mathrm{Ad}_R = R$$

The left Jacobian of SO(3) and its inverse can be calculated using:

$$J_l(\phi) = I + \left(\frac{1-\cos\theta}{\theta^2}\right)\phi^\wedge + \left(\frac{\theta-\sin\theta}{\theta^3}\right)(\phi^\wedge)^2,$$
$$J_l^{-1}(\phi) = I - \frac{1}{2}\phi^\wedge + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)(\phi^\wedge)^2,$$

where $\theta := \|\phi\|$.

Example 4.5. *The rotation matrices we want are in the 3D special orthogonal group,* SO(3)*. Its Lie algebra is* $\mathfrak{so}(3)$ *that includes skew-symmetric matrices. More specifically, consider the angular velocity of* $\phi = \begin{bmatrix}\phi_x \\ \phi_y \\ \phi_z\end{bmatrix} \in \mathbb{R}^3$.

We can pick a basis for $\mathfrak{so}(3)$ *(called generators) as*

$$G_z = \begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, G_y = \begin{bmatrix}0 & 0 & 1\\ 0 & 0 & 0\\ -1 & 0 & 0\end{bmatrix}, G_x = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -1\\ 0 & 1 & 0\end{bmatrix}.$$

So an element of $\mathfrak{so}(3)$ *can be easily constructed using*

$$\phi^\wedge = \phi_x G_x + \phi_y G_y + \phi_z G_z = \begin{bmatrix}0 & -\phi_z & \phi_y\\ \phi_z & 0 & -\phi_x\\ -\phi_y & \phi_x & 0\end{bmatrix}.$$

Now using the exponential map (Lie exponential or matrix exponential), we can already compute the corresponding rotation matrix as

$$\begin{aligned} R = \exp(\phi) &= I + \phi^\wedge + \frac{1}{2!}(\phi^\wedge)^2 + \frac{1}{3!}(\phi^\wedge)^3 + \frac{1}{4!}(\phi^\wedge)^4 + \cdots \\ &= I + \left(\frac{\sin\theta}{\theta}\right)\phi^\wedge + \left(\frac{1-\cos\theta}{\theta^2}\right)(\phi^\wedge)^2, \end{aligned}$$

where $\theta = \|\phi\| = \sqrt{\phi_x^2 + \phi_y^2 + \phi_z^2}$.

Numerical Example:

$$\phi = \begin{bmatrix} -0.3034 \\ 0.2939 \\ -0.7873 \end{bmatrix} \text{ and } \phi^\wedge = \begin{bmatrix} 0 & 0.7873 & 0.2939 \\ -0.7873 & 0 & 0.3034 \\ -0.2939 & -0.3034 & 0 \end{bmatrix} \text{ and}$$

$$\theta = \sqrt{(-0.3034)^2 + 0.2939^2 + (-0.7873)^2} = 0.8934.$$

Using `expm` *function in MATLAB* [20] *we get*

[20] https://www.mathworks.com/help/matlab/ref/expm.html

$$\begin{aligned} R &= \texttt{expm}\left(\begin{bmatrix} 0 & 0.7873 & 0.2939 \\ -0.7873 & 0 & 0.3034 \\ -0.2939 & -0.3034 & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} 0.6698 & 0.6450 & 0.3680 \\ -0.7284 & 0.6671 & 0.1564 \\ -0.1446 & -0.3728 & 0.9166 \end{bmatrix}. \end{aligned}$$

Using the closed-form formula, we get the same result

$$\begin{aligned} R = &\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \left(\frac{\sin(0.8934)}{0.8934}\right)\begin{bmatrix} 0 & 0.7873 & 0.2939 \\ -0.7873 & 0 & 0.3034 \\ -0.2939 & -0.3034 & 0 \end{bmatrix} \\ &+ \left(\frac{1-\cos(0.8934)}{0.8934^2}\right)\begin{bmatrix} 0 & 0.7873 & 0.2939 \\ -0.7873 & 0 & 0.3034 \\ -0.2939 & -0.3034 & 0 \end{bmatrix}^2 \\ = &\begin{bmatrix} 0.6698 & 0.6450 & 0.3680 \\ -0.7284 & 0.6671 & 0.1564 \\ -0.1446 & -0.3728 & 0.9166 \end{bmatrix}. \end{aligned}$$

4.7.2 *Group of Direct Spatial Isometries,* SE(3)

Let $\xi \in \mathbb{R}^6$ be a vector that can be identified with an element of the Lie algebra, $\xi^\wedge \in \mathfrak{se}(3)$. The corresponding pose, $H \in \mathrm{SE}(3)$, can be computed using the exponential map:

$$\xi^\wedge = \begin{bmatrix} \phi \\ \rho \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ 0_{1\times 3} & 0 \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} R & p \\ 0_{1\times 3} & 1 \end{bmatrix}$$

$$H = \exp(\xi) = \begin{bmatrix} \exp(\phi^\wedge) & J_l(\phi)\rho \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathrm{SE}(3),$$

where $\exp(\phi)$ and $J_l(\phi)$ are the exponential map and the left Jacobian of SO(3). The logarithm maps the pose back to the tangent space.

$$\xi^\wedge = \log(H) = \begin{bmatrix} \log(R) & J_l^{-1}(\log(R))\, p \\ 0_{1\times 3} & 0 \end{bmatrix} \in \mathfrak{se}(3)$$

The matrix representation for the adjoint of SE(3) is given by

$$\mathrm{Ad}_H = \begin{bmatrix} R & 0_{3\times 3} \\ p^\wedge R & R \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

The left Jacobian of SE(3) and its inverse are [21]

$$J_l(\xi) = \begin{bmatrix} J_l(\phi) & 0_{3\times 3} \\ Q(\phi,\rho) & J_l(\phi) \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

$$J_l^{-1}(\xi) = \begin{bmatrix} J_l^{-1}(\phi) & 0_{3\times 3} \\ -J_l^{-1}(\phi)Q(\phi,\rho)J_l^{-1}(\phi) & J_l^{-1}(\phi) \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

$$\begin{aligned} Q(\phi,\rho) :=\ & \frac{1}{2}\rho^\wedge + \frac{\theta - \sin\theta}{\theta^3}(\phi^\wedge\rho^\wedge + \rho^\wedge\phi^\wedge + \phi^\wedge\rho^\wedge\phi^\wedge) \\ & - \frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4}(\phi^\wedge\phi^\wedge\rho^\wedge + \rho^\wedge\phi^\wedge\phi^\wedge - 3\phi^\wedge\rho^\wedge\phi^\wedge) \\ & - \frac{1}{2}\left(\frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4} - 3\frac{\theta - \sin\theta - \frac{\theta^3}{6}}{\theta^5}\right)(\phi^\wedge\rho^\wedge\phi^\wedge\phi^\wedge + \phi^\wedge\phi^\wedge\rho^\wedge\phi^\wedge). \end{aligned}$$

[21] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014

4.7.3 *Group of K Direct Spatial Isometries,* $\mathrm{SE}_K(3)$

Let $\xi \in \mathbb{R}^{3(K+1)}$ be a vector identified with an element of the Lie algebra, $\xi^\wedge \in \mathfrak{se}_K(3)$. The corresponding group element, $X \in \mathrm{SE}_K(3)$, is computed using the exponential map:

$$\xi^\wedge = \begin{bmatrix} \phi \\ \rho_1 \\ \vdots \\ \rho_K \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho_1 & \cdots & \rho_K \\ 0_{1\times 3} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 3} & 0 & \cdots & 0 \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} R & p_1 & \cdots & p_K \\ 0_{1\times 3} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 3} & 0 & \cdots & 1 \end{bmatrix}$$

$$X = \exp(\xi) = \begin{bmatrix} \exp(\phi^\wedge) & J_l(\phi)\rho_1 & \cdots & J_l(\phi)\rho_K \\ 0_{1\times 3} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 3} & 0 & \cdots & 1 \end{bmatrix} \in \mathrm{SE}_K(3),$$

where $\exp(\phi^\wedge)$ and $J_l(\phi)$ are the exponential map and the left Jacobian of SO(3).

$$\begin{aligned}\xi^\wedge &= \log(H) \\ &= \begin{bmatrix} \log(R) & J_l^{-1}(\log(R))\, p_1 & \cdots & J_l^{-1}(\log(R))\, p_K \\ 0_{1\times 3} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0_{1\times 3} & 0 & \cdots & 0 \end{bmatrix} \in \mathfrak{se}_K(3).\end{aligned}$$

The matrix representation for the adjoint of $\mathrm{SE}_K(3)$ is given by

$$\mathrm{Ad}_X = \begin{bmatrix} R & 0_{3\times 3} & \cdots & 0_{3\times 3} \\ p_1^\wedge R & R & \cdots & 0_{3\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ p_K^\wedge R & 0_{3\times 3} & \cdots & R \end{bmatrix} \in \mathbb{R}^{3(K+1)\times 3(K+1)}$$

The left Jacobian of $\mathrm{SE}_K(3)$ is

$$J_l(\xi) = \begin{bmatrix} J_l(\phi) & 0_{3\times 3} & \cdots & 0_{3\times 3} \\ Q(\phi,\rho_1) & J_l(\phi) & \cdots & 0_{3\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ Q(\phi,\rho_K) & 0_{3\times 3} & \cdots & J_l(\phi) \end{bmatrix} \in \mathbb{R}^{3(K+1)\times 3(K+1)}$$

4.7.4 First-Order Approximation using BCH

The adjoint representation of a Lie group is a linear map that captures the non-commutative structure of the group. The following properties from adjoint representation and the BCH formula for the first-order approximation are useful.

$$\begin{aligned} X\exp(\xi)X^{-1} &= \exp\left((\mathrm{Ad}_X\xi)^\wedge\right) \\ X\exp(\xi) &= \exp\left((\mathrm{Ad}_X\xi)^\wedge\right) X \\ \exp(\xi)X &= X\exp\left((\mathrm{Ad}_{X^{-1}}\xi)^\wedge\right). \end{aligned} \tag{4.13}$$

In the above equations $X \in \mathcal{G}$ and $\xi^\wedge \in \mathfrak{g}$.

The BCH formula can be used to compound two matrix exponentials. If both terms are small, by keeping the first two terms and ignoring the higher-order terms, we have

$$\mathrm{BCH}(\xi_1^\wedge, \xi_2^\wedge) = \xi_1^\wedge + \xi_2^\wedge + \mathrm{HOT},$$

$$\exp(\xi_1)\exp(\xi_2) \approx \exp(\xi_1 + \xi_2).$$

When both terms are not small and assuming ξ is small, by keeping the linear terms in ξ, we have [22]:

$$\log\left(\exp(r)\exp(\xi)\right)^\vee \approx r + J_r^{-1}(r)\xi,$$

[22] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017

$$\log\left(\exp(\xi)\exp(r)\right)^{\vee} \approx r + J_l^{-1}(r)\xi,$$

where J_r and J_l are the right and left Jacobians of the Lie group $\mathcal{G}$, respectively. The left and right Jacobians are related through the adjoint map,

$$J_r(\xi) = \mathrm{Ad}_{\exp(-\xi)} J_l(\xi).$$

4.8 Jacobians

Suppose we can parameterize a transformation $g \in \mathcal{G}$ using a vector of local coordinates $q \in \mathbb{R}^n$ such that $g(q) : \mathbb{R}^n \to \mathcal{G}$. The Jacobian J is a matrix that relates the rate of change $\frac{d}{dt}q = \dot{q}$ to $\xi^{\wedge} \in \mathfrak{g}$, i.e.,

$$\xi = J\dot{q}.$$

From the reconstruction equations, we can consider the velocity ξ in the body frame as in (4.1) or the spatial frame as in (4.2), which are related via the adjoint map $\xi_s = \mathrm{Ad}_g \xi_b$. Then we obtain the following definitions for the left and right Jacobians [23].

$$\begin{aligned} \left(g^{-1}\dot{g}\right)^{\vee} &= \xi_b = J_r(q)\dot{q}, \\ \left(\dot{g}g^{-1}\right)^{\vee} &= \xi_s = J_l(q)\dot{q}. \end{aligned} \tag{4.14}$$

The particular choice of the group exponential coordinates leads to the group Jacobians presented earlier.

[23] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011

Problems

4.1 Prove (4.12); For all $A, B \in \mathfrak{g}$, $[A, B] = AB - BA$.

4.2 Write down the reconstruction equations for SE(3) using matrices and extract terms for spatial and body velocities.

4.3 Derive the reconstruction equation for SO(n) by directly differentiating the group constraint $RR^{\mathsf{T}} = I$ and $R^{\mathsf{T}}R = I$.

4.4 Repeat Example 4.4 for $\mathfrak{se}(3)$ generators and establish the isomorphism with $\mathbb{R}^6$ standard basis.

4.5 Based on the context of Sec. 4.5, prove that for any $g \in \mathrm{G}$ and $\xi \in \mathfrak{g}$, we have the following.

$$\dot{g} = g\xi_b = \mathrm{Ad}_g \xi_b g.$$

4.6 Using ZYX Euler angles as local coordinates for $R \in \mathrm{SO}(3)$ derive the left and right Jacobians.

4.7 The *special linear group* over $\mathbb{R}$ is:

$$\mathrm{SL}(n) = \mathrm{SL}_n(\mathbb{R}) = \{A \in \mathrm{GL}_n(\mathbb{R}) : \det(A) = 1\}.$$

Prove that its Lie algebra $\mathfrak{sl}(n)$ elements are the traceless matrices. What is $\dim \mathrm{SL}(n)$?

5
Robot Motion

In this chapter, we will take a closer look into a robot motion described by SE(2) and SE(3) groups. We consider the model fully actuated and do not consider the nonholonomic constraints [1]. Of course, these constraints are important for robot control and accurate description of the robot motion; however, it is often the case that simple kinematic models work well in practice for state estimation problems. One reason is that the accurate model of the robot can be complex, with many unknown or unmodeled parts that can lead to lower performance.

[1] These are velocity constraints that cannot be integrated (Bloch, 2015).

5.1 Velocity Motion Model

Consider a robot as a rigid body that operates in 2D or 3D space. Such a robot naturally operates in SE(2) or SE(3). An element of the Lie algebra represents the velocity in the body frame and can be measured by sensors attached to the robot.

The kinematic equation of motion is described by the curve $X(t) \in$ SE(2) or SE(3) as

$$\frac{d}{dt} X_t = \dot{X}_t = X_t u_t^\wedge, \quad u_t^\wedge \in \mathfrak{g}.$$

The control input $u_t = \text{vec}(\omega_t, v_t) \in \mathbb{R}^3$ or $\mathbb{R}^6$ is a vector of angular velocity, ω_t, and linear velocity, v_t [2]. Define the deterministic dynamics as $f_{u_t}(X_t) := X_t u_t^\wedge$. As such, the reconstruction equation is a simple kinematic model that provides a process model to predict the motion of a rigid body.

[2] The notation $\text{vec}(x_1, \cdots, x_n)$ means $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$.

The integration of the reconstruction equation [3] also makes it convenient to predict rotation and translation simultaneously. To see the separate terms for the rotation and translation, we can write

[3] See Ch. 4.4.3.

$$\dot{X}_t = \begin{bmatrix} \dot{R} & \dot{p} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega^\wedge & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R\omega^\wedge & Rv \\ 0 & 0 \end{bmatrix}.$$

Suppose we integrate this process model for a fixed sampling time $\Delta t = t_{k+1} - t_k$ by assuming a zero-order hold [4] on the input. Define $u_k := u_{t_k} \cdot \Delta t$. Then we

[4] https://en.wikipedia.org/wiki/Zero-order_hold

have the following discrete-time motion model.

$$X_{k+1} = X_k \exp(u_k) =: X_k U_k.$$

5.2 Noisy Process and Uncertainty Propagation

Suppose a discrete-time process model on matrix Lie group $\mathcal{G}$ where the state at any two successive times-steps k and $k+1$ is related using input such as $U_k \in \mathcal{G}$. The deterministic process model is as follows.

$$X_{k+1} = f_{u_k}(X_k) := X_k U_k.$$

Substituting in the noisy process model, we have

$$X_{k+1} = f_{u_k}(X_k) \exp(w_k),$$

where $w_k \sim \mathcal{N}(0, \Sigma_{w_k})$.

Notice that the noise is defined in the Lie algebra and mapped to the nonlinear error using the Lie exponential map (matrix exponential). $w_k \in \mathbb{R}^n$ is defined in the Euclidean vector space that is isomorphic to the Lie algebra, where n is the dimension of the Lie algebra.

Remark 5.1. *The continuous-time noisy process can be obtained by corrupting the input $\tilde{u}_t = u_t + w_t$ using $w_t \sim \mathcal{GP}(0, \Sigma_{w_t}\delta(t-t'))$, where $\mathcal{GP}$ denotes a Gaussian process, Σ_{w_t} is power spectral density matrix, and $\delta(t-t')$ denotes the Dirac delta function* [5]. *Then it follows that*

$$\dot{X}_t = X_t(\tilde{u}_t - w_t)^\wedge = X_t\tilde{u}^\wedge - Xw_t^\wedge = f_{u_t}(X) - Xw_t^\wedge.$$

We use the *left-invariant error* [6] to track the covariance of the spatial error as seen in the body-fixed frame:

$$\eta = \exp(\xi) = X^{-1}\bar{X}.$$

Then, the state X is of the following form where $\bar{X}$ is the mean or estimated value of the true state:

$$X = \bar{X}\exp(-\xi).$$

We substitute $X = \bar{X}\exp(-\xi)$ in the noisy process model as follows.

$$\bar{X}_{k+1}\exp(-\xi_{k+1}) = \bar{X}_k \exp(-\xi_k) U_k \exp(w_k),$$

and using the adjoint in (4.13) to shift all noise terms to the right, we get

$$\bar{X}_{k+1}\exp(-\xi_{k+1}) = \bar{X}_k U_k \exp\left((-\mathrm{Ad}_{U_k^{-1}}\xi_k)\right)\exp(w_k).$$

This previous equation shows the relation between noise terms at timesteps k and $k+1$. Thus,

$$\exp(-\xi_{k+1}) = \exp\left((-\mathrm{Ad}_{U_k^{-1}}\xi_k)\right)\exp(w_k).$$

[5] This is the continuous time definition of the noise density. In discrete time we denote the noise distribution as Gaussian. See (Barfoot, 2017, Chapter 2.3).

[6] See (7.3) for the formal definition of the left and right-invariant errors.

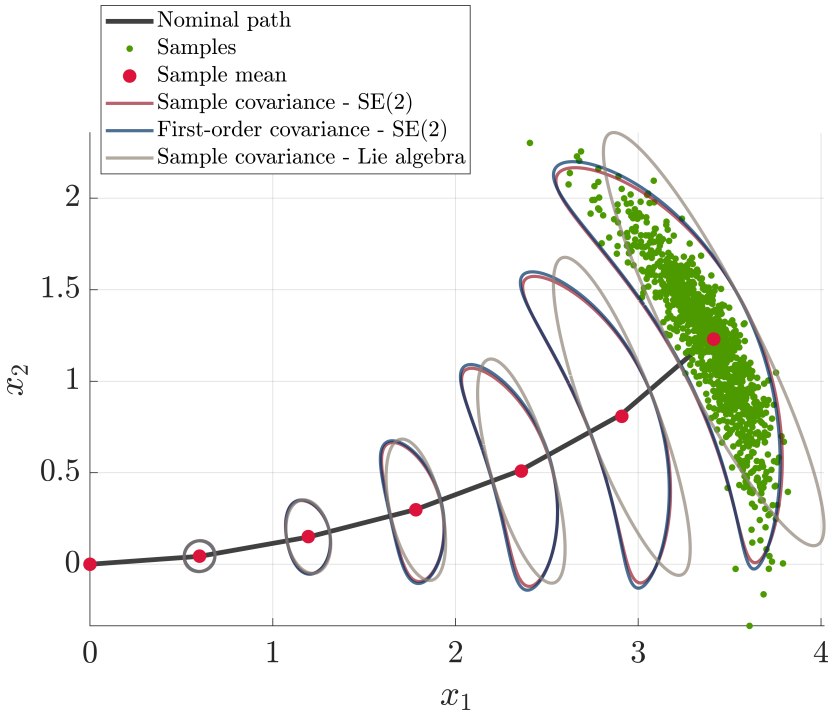

Figure 5.1: Uncertainty propagation on SE(2). See `odometry_propagation_se2.m` (or Python version) for code.

Since all noise terms are small, after applying the BCH formula and keeping the first two terms, we arrive at the approximate error dynamics:

$$\xi_{k+1} = \mathrm{Ad}_{U_k^{-1}} \xi_k - w_k.$$

From here, one can easily show that

$$\Sigma_{k+1} = \mathrm{Ad}_{U_k^{-1}} \Sigma_k \mathrm{Ad}_{U_k^{-1}}^{\mathsf{T}} + \Sigma_{w_k}.$$

Example 5.1 (Uncertainty propagation on SE(2))**.** *In this example, we model a velocity motion model and uncertainty propagation on* SE(2)*. The process model is* $X_{k+1} = X_k U_k \exp(w_k)$ *where* X_k*,* U_k *are both in* SE(2) *and* $w_k \sim \mathcal{N}(0, Q_k)$ *and defined in the Lie algebra* $\mathfrak{se}(2)$*. We use Monte Carlo methods to propagate samples over a path and compute the sample mean and covariance on* SE(2)*. Note that the sample mean and covariance are computed using an iterative algorithm* [7] *different from the usual Euclidean sample statistics. The covariance in Lie algebra is flat, as expected, but it is nonlinear when mapped to the manifold using the exponential map. The results are shown in Figure 5.1.*

[7] A. W. Long, K. C. Wolfe, M. J. Mashner, and G. S. Chirikjian, "The banana distribution is Gaussian: A localization study with exponential coordinates," *Proceedings of the Robotics: Science and Systems Conference*, 2012

Example 5.2 (Uncertainty propagation on SE(3))**.** *In this example, we model a velocity motion model and uncertainty propagation on* SE(3)*. The process model is* $X_{k+1} = X_k \exp(u_k + w_k)$ *where* $X_k \in$ SE(3)*, and* u_k *and*

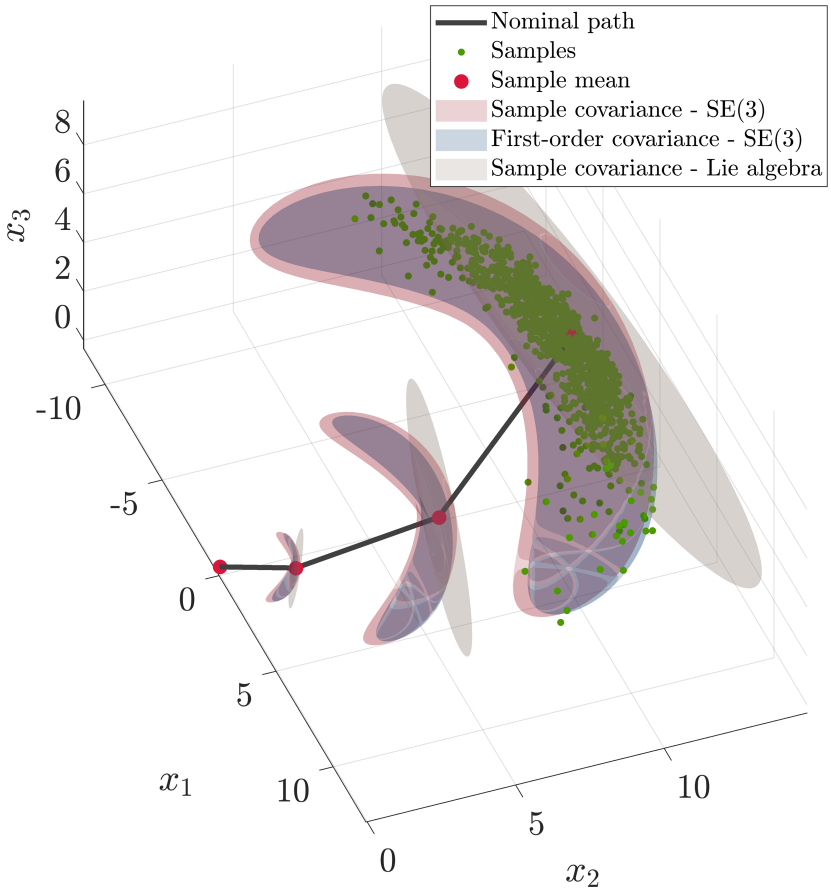

Figure 5.2: Uncertainty propagation on SE(3). See `odometry_propagation_se3.m` (or Python version) for code.

$w_k \sim \mathcal{N}(0, Q_k)$ are defined in the Lie algebra $\mathfrak{se}(3)$. Unlike the previous case, we perturb the input via additive white Gaussian noise in the Lie algebra before taking the exponential map. In practice, the noise source is determined by the sensing model and the problem at hand. We can use the method in Ch. 4.7.4 to separate the noise term as

$$X_{k+1} = X_k \exp(u_k + w_k) \approx X_k \exp(u_k) \exp(J_r(u_k) w_k).$$

Then the covariance propagation can be done via

$$\Sigma_{k+1} = \mathrm{Ad}_{U_k^{-1}} \Sigma_k \mathrm{Ad}_{U_k^{-1}}^{\mathsf{T}} + J_r(u_k) Q_k J_r^{\mathsf{T}}(u_k),$$

where $U_k = \exp(u_k)$. Assuming a small input, we set $J_r(u_k) \approx I$ in the following implementation.

We use Monte Carlo methods to propagate samples over a path and compute the sample mean and covariance on SE(3)*. Similarly, the sample mean and covariance are computed using an iterative algorithm different from the usual Euclidean sample statistics. The covariance in Lie algebra is flat, as expected, but it is nonlinear when mapped to the manifold using the exponential map. The results are shown in Figure 5.2.*

5.3 *Inertial Measurement Unit*

Inertial Measurement Units (IMUs) are ubiquitous and are available in most modern robotic systems. An example is shown in Figure 5.3. The IMU measurements, angular velocity $\tilde{\omega}_t$ and linear acceleration $\tilde{a}_t$ in the body frame. They are modeled as

Figure 5.3: Vectornav VN-100 Inertial Measurement Unit.

$$\tilde{\omega}_t = \omega_t + w_t^g, \quad w_t^g \sim \mathcal{GP}\left(0_{3,1}, \Sigma^g \, \delta(t - t')\right),$$

and

$$\tilde{a}_t = a_t + w_t^a, \quad w_t^a \sim \mathcal{GP}\left(0_{3,1}, \Sigma^a \, \delta(t - t')\right),$$

where $\mathcal{GP}$ denotes a Gaussian process, Σ^g and Σ^a are power spectral density matrices, and $\delta(t - t')$ denotes the Dirac delta function [8].

The IMU dynamics can be written as [9]

$$\begin{aligned} \dot{R}_t &= R_t(\tilde{\omega}_t - w_t^g)^\wedge \\ \dot{v}_t &= R_t(\tilde{a}_t - w_t^a) + g \\ \dot{p}_t &= v_t, \end{aligned} \tag{5.1}$$

where g is the gravity vector. We can also model the state and input as

$$X_t = \begin{bmatrix} R_t & v_t & p_t \\ 0_{1\times 3} & 1 & 0 \\ 0_{1\times 3} & 0 & 1 \end{bmatrix} \in \mathrm{SE}_2(3), \quad u_t = \mathrm{vec}(\tilde{\omega}_t, \tilde{a}_t),$$

and write the IMU process model in matrix form.

$$\begin{aligned} \dot{X} &= \begin{bmatrix} R_t\tilde{\omega}_t^\wedge & R_t\tilde{a}_t + g & v_t \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} R_t & v_t & p_t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_t^{g\wedge} & w_t^a & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= f_{u_t}(X) - X w_t^\wedge, \quad w_t := \mathrm{vec}(w_t^g, w_t^a, 0) \in \mathbb{R}^9. \end{aligned} \tag{5.2}$$

Assuming a zero-order hold on the incoming IMU measurements between times t_k and t_{k+1}, for the discretized [10] system we have (Hartley et al., 2020)

$$\begin{aligned} R_{k+1} &= R_k \, \Gamma_0(\bar{\omega}_k \Delta t) = R_k \exp(\bar{\omega}_k \Delta t), \\ v_{k+1} &= v_k + R_k \Gamma_1(\bar{\omega}_k \Delta t)\bar{a}_k \Delta t + g\Delta t, \\ p_{k+1} &= p_k + v_k \Delta t + R_k \Gamma_2(\bar{\omega}_k \Delta t)\bar{a}_k \Delta t^2 + \frac{1}{2} g \Delta t^2, \end{aligned} \tag{5.3}$$

[8] This is the continuous time definition of the noise density. In discrete-time we denote the noise distribution as Gaussian. See (Barfoot, 2017, Chapter 2.3).

[9] As we will learn in Chapter 7, this model in deterministic form satisfies the group affine property.

[10] https://en.wikipedia.org/wiki/Discretization

where $\bar{\omega}_t := \tilde{\omega}_t - \bar{b}_t^g$ and $\bar{a}_t := \tilde{a}_t - \bar{b}_t^a$ are the "bias-corrected" inputs. These discrete dynamics are an exact integration of the continuous-time system under the assumption that the IMU measurements are constant over $\Delta t := t_{k+1} - t_k$. $\Gamma_0(\phi)$ is simply the exponential map of SO(3). $\Gamma_1(\phi)$ is also known as the left Jacobian of SO(3) (Hartley et al., 2020).

$$\Gamma_0(\phi) = I + \frac{\sin(\|\phi\|)}{\|\phi\|}(\phi^\wedge) + \frac{1-\cos(\|\phi\|)}{\|\phi\|^2}(\phi^\wedge)^2 \tag{5.4}$$

$$\Gamma_1(\phi) = I + \frac{1-\cos(\|\phi\|)}{\|\phi\|^2}(\phi^\wedge) + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}(\phi^\wedge)^2 \tag{5.5}$$

$$\Gamma_2(\phi) = \frac{1}{2}I + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}(\phi^\wedge) + \frac{\|\phi\|^2 + 2\cos(\|\phi\|) - 2}{2\|\phi\|^4}(\phi^\wedge)^2 \tag{5.6}$$

$$\Gamma_m(\phi) := \left(\sum_{n=0}^{\infty} \frac{1}{(n+m)!}(\phi^\wedge)^n \right) \tag{5.7}$$

Problems

5.1 Repeat Examples 5.1 and 5.2 using right-invariant error

$$\eta = \exp(\xi) = \bar{X}X^{-1}.$$

5.2 Replace the process model in Example 5.2 with the IMU model and perform the Monte Carlo simulation for uncertainty propagation.

6
Kalman Filtering

6.1 Discrete-Time Gauss-Markov Process

The Markov property states that "the future is independent of the past if the present is known." A stochastic process that has this property is called a Markov process. The state of a dynamic system driven by white noise is a Markov process.

A discrete-time random process (random sequence), w_k, is called white noise if:

$$\mathbb{E}[w_k w_j^\mathsf{T}] = Q_k \delta_{kj},$$

where the Kronecker δ_{kj} is

$$\delta_{kj} = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases}.$$

The state of a dynamic system excited by white noise,

$$x_{k+1} = f(x_k, w_k),$$

is a discrete-time Markov process or Markov sequence. The state of a linear dynamic system excited by white Gaussian noise,

$$x_{k+1} = F x_k + w_k,$$

is called a Gauss-Markov process. Assuming the initial condition is Gaussian, because of linearity x_k is Gaussian and because of the whiteness of the process noise it is Markov.

6.2 Kalman Filter Assumptions

The state, x_k, evolves according to a known linear dynamic equation with known inputs, u_k, an additive process noise, w_k, which is a zero-mean white (uncorrelated) process with known covariance Q_k;

$$x_k^- = F_k x_{k-1} + G_k u_k + w_k. \tag{6.1}$$

The measurement model is a known linear function of the state with an additive measurement noise, v_k, which is a zero-mean white (uncorrelated) process with known covariance R_k;

$$z_k = H_k x_k^- + v_k. \tag{6.2}$$

Initial state is assumed to be a random variable with known mean (initial estimate) and known covariance (initial uncertainty). Initial state and noises are all mutually uncorrelated.

In summary:

- Known initial state x_0 (with possibly given prior information z_0): mean $\mathbb{E}[x_0|z_0] = \hat{x}_0$ and covariance $\mathrm{Cov}[x_0|z_0] = P_0$.
- Process and measurement noise sequences are white with known covariances:

$$\mathbb{E}[w_k] = 0,\ \mathbb{E}[w_k w_j^\mathsf{T}] = Q_k \delta_{kj}, \quad \text{and} \quad \mathbb{E}[w_k] = 0,\ \mathbb{E}[v_k v_j^\mathsf{T}] = R_k \delta_{kj}.$$

- All the above are uncorrelated.

6.3 Probabilistic Derivation for the Multivariate Gaussian Case

In the prediction step, we use the joint distribution of the state at time steps k and $k-1$, $p(x_k, x_{k-1}|u_{1:k}, z_{1:k-1})$, and marginalize x_{k-1} as follows:

$$\underbrace{\begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}}_{y_k} = \underbrace{\begin{bmatrix} F_k \\ I \end{bmatrix}}_{A_k} x_{k-1} + \underbrace{\begin{bmatrix} G_k \\ 0 \end{bmatrix}}_{B_k} u_k + \underbrace{\begin{bmatrix} I \\ 0 \end{bmatrix}}_{W_k} w_k,$$

$$y_k = A_k x_{k-1} + B_k u_k + W_k w_k.$$

We compute the mean and covariance as:

$$\mathbb{E}[y_k] = \mathbb{E}[A_k x_{k-1} + B_k u_k + W_k w_k] = A_k \mu_{k-1} + B_k u_k = \begin{bmatrix} F_k \mu_{k-1} + G_k u_k \\ \mu_{k-1} \end{bmatrix},$$

$$\begin{aligned} \mathrm{Cov}[y_k] &= \mathbb{E}[(y_k - \mathbb{E}[y_k])(y_k - \mathbb{E}[y_k])^\mathsf{T}] \\ &= \mathbb{E}[(A_k(x_{k-1} - \mu_{k-1}) + W_k w_k)(A_k(x_{k-1} - \mu_{k-1}) + W_k w_k)^\mathsf{T}] \\ &= A_k \Sigma_{k-1} A_k^\mathsf{T} + W_k Q_k W_k^\mathsf{T} = \begin{bmatrix} F_k \Sigma_{k-1} F_k^\mathsf{T} + Q_k & F_k \Sigma_{k-1} \\ \Sigma_{k-1} F_k^\mathsf{T} & \Sigma_{k-1} \end{bmatrix}. \end{aligned}$$

Using the marginalization property of jointly Gaussian random vectors we have:

$$\mathbb{E}[x_k] =: \mu_k^- = F_k \mu_{k-1} + G_k u_k \tag{6.3}$$

$$\text{Cov}[x_k] =: \Sigma_k^- = F_k \Sigma_{k-1} F_k^\mathsf{T} + Q_k \tag{6.4}$$

In the correction step, we form the joint distribution $p(x_k, z_k | u_{1:k}, z_{1:t-1})$ and then condition on z_k.

$$\underbrace{\begin{bmatrix} x_k \\ z_k \end{bmatrix}}_{s_k} = \underbrace{\begin{bmatrix} I \\ H_k \end{bmatrix}}_{C_k} x_k + \underbrace{\begin{bmatrix} 0 \\ I \end{bmatrix}}_{V_k} v_k,$$

$$s_k = C_k x_k + V_k v_k.$$

$$\mathbb{E}[s_k] = \mathbb{E}[C_k x_k + V_k v_k] = C_k \mu_k^- = \begin{bmatrix} \mu_k^- \\ H_k \mu_k^- \end{bmatrix},$$

$$\begin{aligned} \text{Cov}[s_k] &= \mathbb{E}[(s_k - \mathbb{E}[s_k])(s_k - \mathbb{E}[s_k])^\mathsf{T}] \\ &= C_k \Sigma_k^- C_k^\mathsf{T} + V_k R_k V_k^\mathsf{T} = \begin{bmatrix} \Sigma_k^- & \Sigma_k^- H_k^\mathsf{T} \\ H_k \Sigma_k^- & H_k \Sigma_k^- H_k^\mathsf{T} + R_k \end{bmatrix}. \end{aligned}$$

Using the conditioning property of jointly Gaussian random vectors we have:

$$\mathbb{E}[x_k | z_k] =: \mu_k = \mu_k^- + \Sigma_k^- H_k^\mathsf{T} (H_k \Sigma_k^- H_k^\mathsf{T} + R_k)^{-1} (z_k - H_k \mu_k^-), \tag{6.5}$$

$$\begin{aligned} \text{Cov}[x_k | z_k] =: \Sigma_k &= \Sigma_k^- - \Sigma_k^- H_k^\mathsf{T} (H_k \Sigma_k^- H_k^\mathsf{T} + R_k)^{-1} H_k \Sigma_k^- \\ &= (I - \Sigma_k^- H_k^\mathsf{T} (H_k \Sigma_k^- H_k^\mathsf{T} + R_k)^{-1} H_k) \Sigma_k^-. \end{aligned} \tag{6.6}$$

6.4 Kalman Filter Algorithm

Algorithm 1: `Kalman-filter`

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k;

1: $\mu_k^- \leftarrow F_k \mu_{k-1} + G_k u_k$ ▷ predicted mean
2: $\Sigma_k^- \leftarrow F_k \Sigma_{k-1} F_k^\mathsf{T} + Q_k$ ▷ predicted covariance
3: $\nu_k \leftarrow z_k - H_k \mu_k^-$ ▷ innovation
4: $S_k \leftarrow H_k \Sigma_k^- H_k^\mathsf{T} + R_k$ ▷ innovation covariance
5: $K_k \leftarrow \Sigma_k^- H_k^\mathsf{T} S_k^{-1}$ ▷ filter gain
6: $\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean
7: $\Sigma_k \leftarrow (I - K_k H_k) \Sigma_k^-$ ▷ corrected covariance
8: $//\Sigma_k \leftarrow (I - K_k H_k) \Sigma_k^- (I - K_k H_k)^\mathsf{T} + K_k R_k K_k^\mathsf{T}$ ▷ numerically stable form
9: **return** μ_k, Σ_k

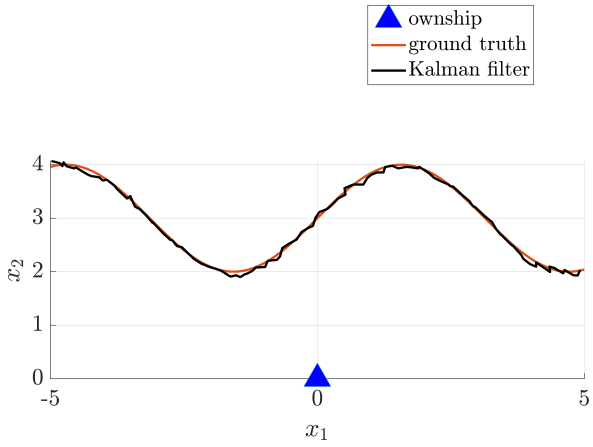

Figure 6.1: 2D target tracking using a Kalman filter.

6.5 *Overview and Limitations of Kalman Filter*

Under the Gaussian assumption for the initial state (or initial state error) and all the noises entering into the system, the Kalman filter is the optimal Minimum Mean Square Error (MMSE) state estimator. In other words, for a linear system with Gaussian noise, the Kalman filter equations are the Best Linear Unbiased Estimator (BLUE); this means they are performing right at the Cramér-Rao lower bound (Barfoot, 2017). If these random variables are not Gaussian and one has only their first two moments, then the Kalman filter algorithm is the best linear state estimator in the sense of MMSE (Bar-Shalom et al., 2001, page 207).

In practice, we come across many interesting problems that violate our assumptions. These are include:

- nonlinear motion (process) and measurement models;
- unknown control inputs or mode changes;
- data association uncertainty;
- auto-correlated or cross-correlated noise sequences.

Example 6.1 (Kalman Filter Target Tracking)**.** *A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to noisy measurements that directly observe the target 2D coordinates at any time step.*

$$F_k = I_2,\ G_k = 0_{2\times 2},\ H_k = I_2,\ Q_k = 0.001\ I_2,\ R_k = 0.05^2\ I_2$$

We estimate the target position using a Kalman filter as shown in Figure 6.1. See `kf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

6.6 Nonlinear Dynamic Systems

We consider nonlinear systems. In the deterministic case, the system is described using the nonlinear process model, $f(u_k, x_{k-1})$, and measurement model, $h(x_k)$, as

$$\begin{aligned} x_k &= f(u_k, x_{k-1}), \\ z_k &= h(x_k). \end{aligned} \tag{6.7}$$

In general, we may consider the nonlinear dynamic system excited by multiplicative noise.

$$\begin{aligned} x_k &= f(u_k, x_{k-1}, w_k), \\ z_k &= h(x_k, v_k). \end{aligned} \tag{6.8}$$

Then the additive noise case can be considered as a special case of (**??**).

$$\begin{aligned} x_k &= f(u_k, x_{k-1}) + w_k, \\ z_k &= h(x_k) + v_k. \end{aligned} \tag{6.9}$$

6.7 EKF: Linearization via Taylor Expansion

We modify the linear Kalman filter to use the linearized models around the current operating point at every step. The following algorithm summarizes the EKF algorithm where $F_k = \frac{\partial f}{\partial x}\Big|_{x=\mu_{k-1}}$, $W_k = \frac{\partial f}{\partial w}\Big|_{x=\mu_{k-1}}$, $H_k = \frac{\partial h}{\partial x}\Big|_{x=\mu_k^-}$, and $V_k = \frac{\partial h}{\partial v}\Big|_{x=\mu_k^-}$. Note that the linearized model must be re-evaluated at each iteration.

Algorithm 2: `Extended-Kalman-filter`

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k;

1: $\mu_k^- \leftarrow f(u_k, \mu_{k-1})$ ▷ predicted mean
2: $\Sigma_k^- \leftarrow F_k \Sigma_{k-1} F_k^\mathsf{T} + W_k Q_k W_k^\mathsf{T}$ ▷ predicted covariance
3: $\nu_k \leftarrow z_k - h(\mu_k^-)$ ▷ innovation
4: $S_k \leftarrow H_k \Sigma_k^- H_k^\mathsf{T} + V_k R_k V_k^\mathsf{T}$ ▷ innovation covariance
5: $K_k \leftarrow \Sigma_k^- H_k^\mathsf{T} S_k^{-1}$ ▷ filter gain
6: $\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean
7: $\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-$ ▷ corrected covariance
8: $//\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^- (I - K_k H_k)^\mathsf{T} + K_k R_k K_k^\mathsf{T}$ ▷ numerically stable form
9: **return** μ_k, Σ_k

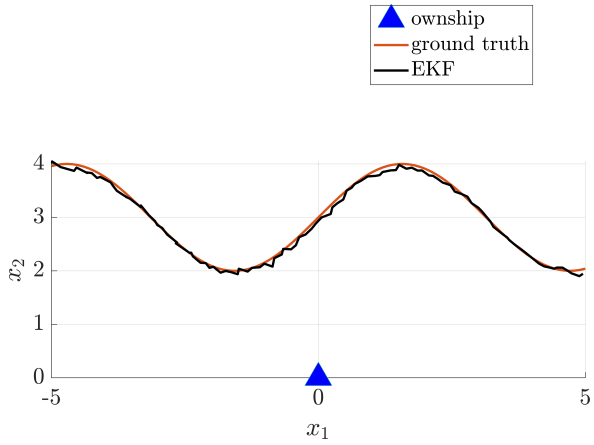

Figure 6.2: 2D target tracking using an EKF.

6.7.1 EKF Summary

- Highly efficient; polynomial time in measurement dimensionality n_z and state dimensionality n_x.
- Not optimal.
- Can diverge if nonlinearities are large.
- Can work well in practice for many problems despite violating all the underlying assumptions.

Example 6.2 (EKF Target Tracking)**.** *A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.*

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k^{1^2} + x_k^{2^2}} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k$$

$$F_k = I_2,\ G_k = 0_2,\ Q_k = 0.001\ I_2,\ R_k = \text{diag}(0.05^2, 0.01^2)$$

$$H_k = \begin{bmatrix} \frac{x_k^1}{\sqrt{x_k^{1^2}+x_k^{2^2}}} & \frac{x_k^2}{\sqrt{x_k^{1^2}+x_k^{2^2}}} \\ \frac{x_k^2}{x_k^{1^2}+x_k^{2^2}} & \frac{-x_k^1}{x_k^{1^2}+x_k^{2^2}} \end{bmatrix}$$

We estimate the target position using an EKF as shown in Figure 6.2. See `ekf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

6.8 UKF: Filtering via Unscented Transform

Algorithm 3 shows the implementation of UKF using the unscented transform for uncertainty propagation.

Algorithm 3: `Unscented-Kalman-filter`

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k;

$\mathcal{X}_{k-1} \leftarrow$ compute the set of $2n+1$ sigma points using μ_{k-1} and Σ_{k-1}

$w^- \leftarrow$ compute the set of $2n+1$ weights

$\mu_k^- = \sum_{i=0}^{2n} w_i^- f(u_k, x_{k-1,i})$ ▷ predicted mean

$\Sigma_k^- \leftarrow \sum_{i=0}^{2n} w_i^- (f(u_k, x_{k-1,i}) - \mu_k^-)(f(u_k, x_{k-1,i}) - \mu_k^-)^\mathsf{T} + Q_k$ ▷ predicted covariance

$\mathcal{X}_k^- \leftarrow$ compute the set of $2n+1$ sigma points using μ_k^- and Σ_k^-

$w \leftarrow$ compute the set of $2n+1$ weights

$z_k^- = \sum_{i=0}^{2n} w_i h(x_{k,i}^-)$ ▷ predicted measurement

$\nu_k \leftarrow z_k - z_k^-$ ▷ innovation

$S_k \leftarrow \sum_{i=0}^{2n} w_i (h(x_{k,i}^-) - z_k^-)(h(x_{k,i}^-) - z_k^-)^\mathsf{T} + R_k$ ▷ innovation covariance

$\Sigma_k^{xz} \leftarrow \sum_{i=0}^{2n} w_k^{[i]} (x_{k,i}^- - \mu_k^-)(h(x_{k,i}^-) - z_k^-)^\mathsf{T}$ ▷ state and measurement cross covariance

$K_k \leftarrow \Sigma_k^{xz} S_k^{-1}$ ▷ filter gain

$\mu_k \leftarrow \mu_k^- + K_k \nu_k$ ▷ corrected mean

$\Sigma_k \leftarrow \Sigma_k^- - K_k S_k K_k^\mathsf{T}$ ▷ corrected covariance

return μ_k, Σ_k

Example 6.3 (UKF Target Tracking)**.** *A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.*

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k^{1\,2} + x_k^{2\,2}} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k$$

$$F_k = I_2,\ G_k = 0_2,\ Q_k = 0.001\ I_2,\ R_k = \text{diag}(0.05^2, 0.01^2)$$

We estimate the target position using a UKF as shown in Figure 6.3. See `ukf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

6.9 UKF vs. EKF

In the following, we summarize the similarities and differences between the two approaches.

- Same results as EKF for linear models;
- Better approximation than EKF for nonlinear models;
- Differences often "somewhat small";
- No Jacobians needed for the UKF;
- UKF has the same complexity class;

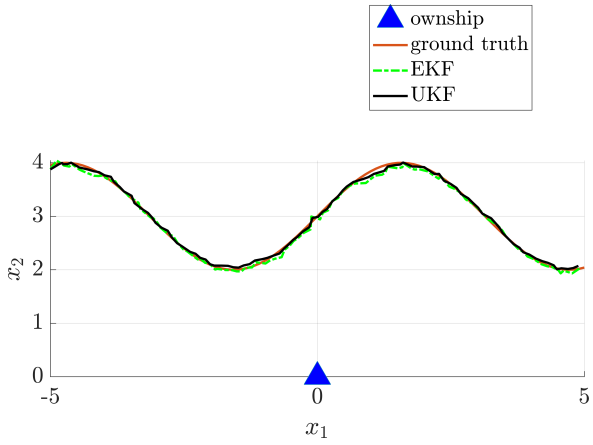

Figure 6.3: 2D target tracking using a UKF.

- UKF is slightly slower than the EKF.

For in-depth reading of the discussed topics, see Chapters 3 and 4 of Barfoot (2017).

6.10 *Error-State Kalman Filter*

For EKF, a much better formulation called Error-State Kalman Filter (ESKF) exists (Trawny and Roumeliotis, 2005; Solà, 2017). Because the filter estimates the covariance of the state, it is better to derive the error dynamics by centering the error about the true state and to evaluate the Jacobians at the current estimate. This way, the linearization is about the identity as opposed to the current state estimate (which can be far from the true state), leading to reduced or negligible higher-order effects of the error. In the next chapter, we study the Invariant EKF, an error-state EKF on matrix Lie groups, with interesting theoretical results. The Invariant EKF is a generalization of ESKF with formal convergence proof (Barrau and Bonnabel, 2017).

Problems

6.1 Describe the Kalman filter estimation cycle and intuitively explain the meaning of each step in connection with Bayes' filter.

6.2 Show that the matrix differential equation $\frac{d}{dt}P_t = A_t P_t + P_t A_t + Q_t$ is a continuous-time model for $P_{k+1} = \Phi_k P_k \Phi_k^\mathsf{T} + Q_k$ where $\Phi_k = \exp(A_t \Delta t)$ for some sampling time $\Delta t = t_{k+1} - t_k$.

6.3 Prove that the Kalman filter gain $K_k = \Sigma_k^- H_k^\mathsf{T} S_k^{-1}$ is the optimal gain in the sense of minimum mean-square error estimation.

6.4 A target is moving in a 1D plane. The ownship position is known and fixed at the origin, thus the target's position at time step k is x_k. We

have no information on the target's trajectory and thus we model the motion as a random walk with variance of 4 m^2. Based on previous measurements, we have an initial guess of the target position as $\mu_0 = 20$ m with variance $\sigma_0^2 = 9$ m^2. We then receive measurement from each of the next two time steps $z_1 = 22$ m and $z_2 = 23$ m. Each with variance 1 m^2. Use a Kalman filter to estimate the state of the target at time step 2 (μ_2 and σ_2^2).

7
Invariant Kalman Filtering

Matrix Lie groups (Chirikjian, 2011; Hall, 2015; Barfoot, 2017) provide natural (exponential) coordinates that exploits symmetries of the space (Long et al., 2012; Barfoot and Furgale, 2014; Forster et al., 2016; Mangelson et al., 2020; Mahony and Trumpf, 2021; Brossard et al., 2022). State estimation is the problem of determining a robot's position, orientation, and velocity that are vital for robot control (Barfoot, 2017). An interesting class of state estimators that can be run at high frequency, e.g., 2 kHz, are based on Invariant Extended Kalman Filter (InEKF) (Barrau, 2015; Barrau and Bonnabel, 2017, 2018). The theory of invariant observer design is based on the estimation error being invariant under the action of a matrix Lie group. The fundamental result is that by correct parametrization of the error variable, a wide range of nonlinear problems can lead to (log) linear error dynamics (Bonnabel et al., 2009; Barrau, 2015; Barrau and Bonnabel, 2017).

Consider a deterministic Linear Time-Invariant (LTI) process model

$$\dot{x} = Ax + Bu. \tag{7.1}$$

Let $\bar{x}$ be an estimate of x, i.e.,

$$\dot{\bar{x}} = A\bar{x} + Bu.$$

Define the error $e := x - \bar{x}$. Then

$$\dot{e} = \dot{x} - \dot{\bar{x}} = A(x - \bar{x}) = Ae$$

is an autonomous differential equation. Given an initial condition $e(0) = e_0$, we can solve for the error at any time [1]

$$e(t) = \exp(At)e_0.$$

Error propagation is independent of the system trajectory (state estimate).

[1] See https://web.mit.edu/2.14/www/Handouts/StateSpaceResponse.pdf for more explanation on the time-domain solution of LIT systems.

7.1 A Motivating Example: 3D Orientation Propagation

Suppose we wish to estimate the 3D orientation of a rigid body given angular velocity measurements in the body frame [2],

[2] The notation $\text{vec}(x_1, \cdots, x_n)$ means $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$.

$$\omega_t := \text{vec}(\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3.$$

This type of measurement can be easily obtained from a gyroscope. If we let $q_t := \text{vec}(q_x, q_y, q_z)$ be a vector of Euler angles using the $R = R_z R_y R_x$ convention, then the orientation dynamics can be expressed as

$$\frac{d}{dt}\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} 1 & \sin(q_x)\tan(q_y) & \cos(q_x)\tan(q_y) \\ 0 & \cos(q_x) & -\sin(q_x) \\ 0 & \sin(q_x)\sec(q_y) & \cos(q_x)\sec(q_y) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}.$$

Remark 7.1. $\omega^b = R^\mathsf{T}\dot{R} = R^\mathsf{T}\frac{dR}{dq}\cdot\frac{dq}{dt} = R^\mathsf{T}\sum_{i,j,k=1}^{3}\frac{dRij}{dq_k}\cdot\frac{dq_k}{dt} =: E^{-1}(q)\dot{q}$. *Then we have* $\dot{q} = E(q)\omega^b$. *In this case,* R_{ij} *is a* 3×3 *matrix of all zeros except for the* ij*-th entry from* R [3].

[3] To generate this matrix, see https://github.com/RossHartley/angles

Let $\delta q_t := q_t - \bar{q}_t \in \mathbb{R}^3$ be the error between the true and estimated Euler angles. The error dynamics can be written as a nonlinear function of the error variable, the inputs, and the state

$$\frac{d}{dt}\delta q_t = g(\delta q_t, \omega_t, q_t).$$

To propagate the covariance in an EKF, we need to linearize the error dynamics at the current state estimate, $q_t = \bar{q}_t$ (i.e. zero error). This leads to a linear error dynamics of the form:

$$\begin{aligned}\frac{d}{dt}\delta q_t &\approx \begin{bmatrix} 0 & (\omega_z\bar{c}_x + \omega_y\bar{s}_x)/\bar{c}_y^2 & \bar{t}_y(\omega_y\bar{c}_x - \omega_z\bar{s}_x) \\ 0 & 0 & \omega_z\bar{c}_x + \omega_y\bar{s}_x \\ 0 & (\bar{s}_y(\omega_z\bar{c}_x + \omega_y\bar{s}_x))/\bar{c}_y^2 & (\omega_y\bar{c}_x - \omega_z\bar{s}_x)/\bar{c}_y \end{bmatrix}\delta q_t \\ &=: A(\omega_t, \bar{q}_t)\delta q_t,\end{aligned}$$

where $\bar{c}_x$, $\bar{s}_x$, and $\bar{t}_x$ are shorthand for $\cos(\bar{q}_x)$, $\sin(\bar{q}_x)$, and $\tan(\bar{q}_x)$.

$$\frac{d}{dt}\delta q_t := A(\omega_t, \bar{q}_t)\delta q_t,$$

The linear dynamics matrix, $A(\omega_t, \bar{q}_t)$, clearly depends on the estimated angles. Therefore, bad estimates will affect the accuracy of the linearization and ultimately the performance and consistency of the filter.

7.2 Process Dynamics on Lie Groups

A process dynamics evolving on the Lie group, for state $X_t \in \mathcal{G}$, is

$$\frac{d}{dt}X_t = f_{u_t}(X_t). \tag{7.2}$$

$\bar{X}_t$ denotes an estimate of the state. The state estimation error is defined using right or left multiplication of X_t^{-1}.

Definition 7.1 (Left and Right Invariant Error)**.** *The right- and left-invariant errors between two trajectories X_t and $\bar{X}_t$ are:*

$$\begin{aligned} \eta_t^r &= \bar{X}_t X_t^{-1} = (\bar{X}_t L)(X_t L)^{-1} \quad \textit{(Right-Invariant)} \\ \eta_t^l &= X_t^{-1}\bar{X}_t = (L\bar{X}_t)^{-1}(LX_t), \quad \textit{(Left-Invariant)} \end{aligned} \tag{7.3}$$

where $L \in \mathcal{G}$ is an arbitrary element of the group.

Theorem 7.1 (Autonomous Error Dynamics)**.** [4] *A system is group affine if the dynamics, $f_{u_t}(\cdot)$, satisfies:*

$$f_{u_t}(X_1X_2) = f_{u_t}(X_1)X_2 + X_1 f_{u_t}(X_2) - X_1 f_{u_t}(I)X_2 \tag{7.4}$$

for all $t > 0$ and $X_1, X_2 \in \mathcal{G}$. Furthermore, if this condition is satisfied, the right- and left-invariant error dynamics are trajectory independent and satisfy:

$$\begin{aligned} \frac{d}{dt}\eta_t^r &= g_{u_t}(\eta_t^r) \quad \textit{where} \quad g_{u_t}(\eta^r) = f_{u_t}(\eta^r) - \eta^r f_{u_t}(I), \\ \frac{d}{dt}\eta_t^l &= g_{u_t}(\eta_t^l) \quad \textit{where} \quad g_{u_t}(\eta^l) = f_{u_t}(\eta^l) - f_{u_t}(I)\eta^l. \end{aligned} \tag{7.5}$$

Define A_t to be a $\dim\mathfrak{g} \times \dim\mathfrak{g}$ matrix satisfying

$$g_{u_t}(\exp(\xi)) := (A_t\xi)^\wedge + \mathcal{O}(\|\xi\|^2).$$

For all $t \geq 0$, let $\xi_t \in \mathbb{R}^{\dim\mathfrak{g}}$ be the solution of the linear differential equation $\frac{d}{dt}\xi_t = A_t\xi_t$.

Theorem 7.2 (Log-Linear Property of the Error)**.** [5] *Consider the right-invariant error, η_t, between two trajectories (possibly far apart). For arbitrary initial error $\xi_0 \in \mathbb{R}^{\dim\mathfrak{g}}$, if $\eta_0 = \exp(\xi_0)$, then for all $t \geq 0$,*

$$\eta_t = \exp(\xi_t);$$

that is, the nonlinear estimation error η_t can be exactly recovered from the time-varying linear differential equation.

Example 7.1 (3D Orientation Propagation)**.** [6] *Suppose we are interested in estimating the 3D orientation of a rigid body given angular velocity measurements in the body frame, $\omega_t := \text{vec}(\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$. This type of measurement can be easily obtained from a gyroscope. Using a rotation matrix, $R_t \in \text{SO}(3)$. The dynamics becomes*

$$\frac{d}{dt}R_t = R_t\omega_t^\wedge.$$

If we define the error between the true and estimated orientation as

$$\eta_t := R_t^\mathsf{T}\bar{R}_t \in \text{SO}(3),$$

[4] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017

[5] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017

[6] The good way!

then the (left-invariant) error dynamics becomes

$$\begin{aligned}\frac{d}{dt}\eta_t &= R_t^\mathsf{T}\frac{d}{dt}\bar{R}_t + \frac{d}{dt}R_t^\mathsf{T}\bar{R}_t = R_t^\mathsf{T}\bar{R}_t\omega_t^\wedge + \left(R_t\omega_t^\wedge\right)^\mathsf{T}\bar{R}_t \\ &= R_t^\mathsf{T}\bar{R}_t\omega_t^\wedge - \omega_t^\wedge R_t^\mathsf{T}\bar{R}_t \\ &= \eta_t\omega_t^\wedge - \omega_t^\wedge\eta_t \\ &= g(\eta_t, \omega_t).\end{aligned}$$

Using this particular choice of state and error variable yields an autonomous error dynamics function (independent of the state).

Now let $\eta_t := \exp(\xi_t)$*. Using the first-order approximation for the exponential map,* $\exp(\zeta_t) \approx I + \xi_t^\wedge$*, we have* [7]

[7] For all $a, b \in \mathbb{R}^3$, we have $a^\wedge b^\wedge - b^\wedge a^\wedge = [a^\wedge, b^\wedge] = (a \times b)^\wedge$.

$$\begin{aligned}\frac{d}{dt}(\exp(\xi_t)) &= \exp(\xi_t)\omega_t^\wedge - \omega_t^\wedge\exp(\xi_t) \\ \frac{d}{dt}\left(I + \xi_t^\wedge\right) &\approx \left(I + \xi_t^\wedge\right)\omega_t^\wedge - \omega_t^\wedge\left(I + \xi_t^\wedge\right) \\ \frac{d}{dt}\xi_t^\wedge &= \xi_t^\wedge\omega_t^\wedge - \omega_t^\wedge\xi_t^\wedge = (-\omega_t^\wedge\xi_t)^\wedge \\ \frac{d}{dt}\xi_t &= -\omega_t^\wedge\xi_t.\end{aligned}$$

In the previous example, we arrived at a log-linear error dynamics (the linearized error dynamics in the Lie algebra) that is independent of the state variables. This is an attractive results as it enables error propagation similar to that of the linear system in (7.1). We now prove that this linearization is in fact exact, which is a surprising result.

Proof. Let $\eta_0 = \exp(\xi_0)$ be the initial left invariant error. We can show that $\eta_t = R_t^\mathsf{T}\eta_0 R_t$ is the solution to the error dynamics via differentiation.

$$\frac{d}{dt}\eta_t = R_t^\mathsf{T}\eta_0 R_t\omega_t^\wedge - \omega_t^\wedge R_t^\mathsf{T}\eta_0 R_t = \eta_t\omega_t^\wedge - \omega_t^\wedge\eta_t,$$

and

$$\begin{aligned}\eta_t &= R_t^\mathsf{T}\eta_0 R_t \\ \exp(\xi_t) &= R_t^\mathsf{T}\exp(\xi_0)R_t \overset{\text{Adjoint}}{=} \exp(R_t^\mathsf{T}\xi_0) \\ \xi_t &= R_t^\mathsf{T}\xi_0\end{aligned}$$

By differentiating, we get the log-linear error dynamics.

$$\frac{d}{dt}\xi_t = -\omega_t^\wedge R_t^\mathsf{T}\xi_0 = -\omega_t^\wedge\xi_t$$

□

Remark 7.2. *In the previous proof, we used the definition of the adjoin and the property of matrix exponential* [8] *In particular,* $\exp(XYX^{-1}) = X\exp(Y)X^{-1}$ *for any invertible matrix* X*. To prove this we can expand the*

[8] https://en.wikipedia.org/wiki/Matrix_exponential#Properties.

power series of the matrix $\exp(\cdot)$ *function and use the fact that* $(XYX^{-1})^n = XY^nX^{-1}$. *This last property can be proved using induction as follows.*

$n = 1, \quad (XYX^{-1})^1 = XY^nX^{-1}$
$n = 2, \quad (XYX^{-1})^2 = (XYX^{-1})(XYX^{-1}) = XY^2X^{-1}$
$n = 3, \quad (XYX^{-1})^3 = (XYX^{-1})^2(XYX^{-1}) = XY^2X^{-1}(XYX^{-1}) = XY^3X^{-1}$
$n = k, \quad (XYX^{-1})^k = (XYX^{-1})^{k-1}(XYX^{-1}) = XY^{k-1}X^{-1}(XYX^{-1}) = XY^kX^{-1}$
$n = k+1, \quad (XYX^{-1})^{k+1} = (XYX^{-1})^k(XYX^{-1}) = XY^kX^{-1}(XYX^{-1}) = XY^{k+1}X^{-1}.$

7.3 Associated Noisy System

A noisy process dynamics evolving on the Lie group take the following form.

$$\frac{d}{dt}X_t = f_{u_t}(X_t) + X_t w_t^\wedge,$$

where $w_t^\wedge \in \mathfrak{g}$ is a continuous white noise whose covariance matrix is denoted by Q_t. Then the equivalent noisy error dynamics can be shown to be

$$\begin{aligned}\frac{d}{dt}\eta_t^r &= g_{u_t}(\eta_t^r) - (\bar{X}_t w_t^\wedge \bar{X}_t^{-1})\eta_t^r = g_{u_t}(\eta_t^r) - (\mathrm{Ad}_{\bar{X}_t} w_t)^\wedge \eta_t^r, \\ \frac{d}{dt}\eta_t^l &= g_{u_t}(\eta_t^l) - w_t^\wedge \eta_t^l. \end{aligned} \tag{7.6}$$

7.4 Invariant Observation Model

During the correction step of a Kalman filter, the error is updated using incoming sensor measurements. If observations take a particular form, then the linearized observation model and the innovation will also be autonomous. This happens when the measurement, Y_{t_k}, can be written as either

$$\begin{aligned} Y_{t_k} &= X_{t_k} b + V_{t_k} \quad \text{(Left-Invariant Observation)} \quad \text{or} \\ Y_{t_k} &= X_{t_k}^{-1} b + V_{t_k} \quad \text{(Right-Invariant Observation).} \end{aligned} \tag{7.7}$$

b is a constant vector and V_{t_k} is a vector of Gaussian noise.

7.5 Left-Invariant Extended Kalman Filter

We now derive the Left-Invariant Extended Kalman Filter (LI-EKF) following the Kalman filtering theory in the previous chapters. In particular, the filter tracks the state mean and covariance as parameters and corrects the error via a "linear" update rule. As we will see, the innovation is in the Lie algebra, and the update rule is similar to how the reconstruction equation is integrated.

7.5.1 Propagation

In the propagation (or prediction) step, the mean can be directly computed using the process model. However, for the covariance propagation, we use the left-invariant log-linear error dynamics. As such, the state's mean evolves on the group while the covariance is tracked in the Lie algebra.

$$\frac{d}{dt}\bar{X}_t = f_{u_t}(\bar{X}_t), \quad t_{k-1} \leq t < t_k$$

$$\frac{d}{dt}\eta_t^l = g_{u_t}(\eta_t^l) - w_t^\wedge \eta_t^l \implies \frac{d}{dt}\xi_t^l = A_t^l \xi_t^l - w_t$$

$$\frac{d}{dt}P_t^l = A_t^l P_t^l + P_t^l A_t^{l\mathsf{T}} + Q_t$$

7.5.2 Update

In the update (or correction) step, we use $\exp(\xi) \approx I + \xi^\wedge$ and neglect the higher order terms to derive the linear update error equation as follows.

$$\bar{X}_{t_k}^+ = \bar{X}_{t_k} \exp\left(L_{t_k}\left(\bar{X}_{t_k}^{-1} Y_{t_k} - b\right)\right)$$

$$X_{t_k}^{-1}\bar{X}_{t_k}^+ = X_{t_k}^{-1}\bar{X}_{t_k} \exp\left(L_{t_k}\left(\bar{X}_{t_k}^{-1}(X_{t_k} b + V_{t_k}) - b\right)\right)$$

$$\eta_{t_k}^{l+} = \eta_{t_k}^l \exp\left(L_{t_k}\left((\eta_{t_k}^l)^{-1} b - b + \bar{X}_{t_k}^{-1} V_t\right)\right)$$

$$I + \xi_{t_k}^{l+\wedge} = (I + \xi_{t_k}^{l\ \wedge})\left(I + \left(L_{t_k}\left((I - \xi_{t_k}^{l\ \wedge})b - b + \bar{X}_{t_k}^{-1} V_t\right)\right)^\wedge\right)$$

$$\xi_{t_k}^{l+\wedge} = \xi_{t_k}^{l\ \wedge} + \left(L_{t_k}\left((I - \xi_{t_k}^{l\ \wedge})b - b + \bar{X}_{t_k}^{-1} V_t\right)\right)^\wedge$$

$$\xi_{t_k}^{l+} = \xi_{t_k}^l + L_{t_k}\left(-\xi_{t_k}^{l\ \wedge} b + \bar{X}_{t_k}^{-1} V_t\right)$$

To arrange all terms according to the vector form of ξ, define the measurement Jacobian, H, such that

$$\boxed{H\xi = \xi^\wedge b}.$$

Then it follows that

$$\bar{X}_{t_k}^+ = \bar{X}_{t_k} \exp\left(L_{t_k}\left(\bar{X}_{t_k}^{-1} Y_{t_k} - b\right)\right)$$

$$\xi_{t_k}^{l+} = \xi_{t_k}^l - L_{t_k} H \xi_{t_k}^l + L_{t_k} \bar{X}_{t_k}^{-1} V_t$$

$$\xi_{t_k}^{l+} = (I - L_{t_k} H)\xi_{t_k}^l + L_{t_k} \bar{X}_{t_k}^{-1} V_t$$

$$P_{t_k}^{l+} = (I - L_{t_k} H) P_{t_k}^l (I - L_{t_k} H)^\mathsf{T} + L_{t_k} \bar{N}_k L_{t_k}^\mathsf{T}$$

where

$$\bar{N}_k := \bar{X}_{t_k}^{-1} \mathrm{Cov}[V_k] \bar{X}_{t_k}^{-\mathsf{T}}.$$

7.5.3 LI-EKF Result

To summarize, we have the following two steps (as usual for a Kalman filter) for an LI-EKF.

1. Propagation:

$$\frac{d}{dt}\bar{X}_t = f_{u_t}(\bar{X}_t), \quad t_{k-1} \leq t < t_k \tag{7.8}$$

$$\frac{d}{dt}P_t^l = A_t^l P_t^l + P_t^l A_t^{l\mathsf{T}} + Q_t \tag{7.9}$$

2. Update:

$$\bar{X}_{t_k}^+ = \bar{X}_{t_k} \exp\left(L_{t_k}\left(\bar{X}_{t_k}^{-1} Y_{t_k} - b\right)\right) \tag{7.10}$$

$$P_{t_k}^{l+} = (I - L_{t_k}H)P_{t_k}^l(I - L_{t_k}H)^\mathsf{T} + L_{t_k}\bar{N}_k L_{t_k}^\mathsf{T} \tag{7.11}$$

where

$$L_{t_k} = P_{t_k}^l H^\mathsf{T} S^{-1}, \quad S = HP_{t_k}^l H^\mathsf{T} + \bar{N}_k.$$

Given these equations, once we know A_t^l and H matrices, we can implement the LI-EKF.

Example 7.2 (IMU-GPS Left-Invariant EKF)**.** *A robot is equipped with an IMU and Global Positioning System (GPS) sensors. We use a Left-Invariant EKF to estimate its pose* (R_k, p_k) *and velocity,* v_k*, in the world frame. The state is modeled using* $\mathrm{SE}_2(3)$ *such that*

$$X_k = \begin{bmatrix} R_k & v_k & p_k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathrm{SE}_2(3).$$

For the prediction step, we use the discretized IMU model, and we update the predictions using GPS measurements.

The (world-centric) left-invariant error dynamics matrix only depends on the IMU inputs that are assumed to be constant between times t_k *and* t_{k+1}*.*

$$A_t^l = \begin{bmatrix} -\tilde{\omega}_t^\wedge & 0 & 0 \\ -\tilde{a}_t^\wedge & -\tilde{\omega}_t^\wedge & 0 \\ 0 & I & -\tilde{\omega}_t^\wedge \end{bmatrix}.$$

The state transition matrix can be simply computed using the matrix exponential.

$$\Phi^l(t_{k+1}, t_k) = \exp(A_t^l \Delta t).$$

This state transition matrix also has an analytical solution of the form:

$$\Phi^l(t_{k+1}, t_k) = \begin{bmatrix} \Phi_{11}^l & 0 & 0 \\ \Phi_{21}^l & \Phi_{22}^l & 0 \\ \Phi_{31}^l & \Phi_{32}^l & \Phi_{33}^l \end{bmatrix}.$$

The individual terms are

$$\begin{aligned}
\Phi^l_{11} &= \Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t) \\
\Phi^l_{21} &= -\Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t)(\Gamma_1(\bar{\omega}_t \Delta t)\bar{a}_t)^\wedge \Delta t \\
\Phi^l_{31} &= -\Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t)(\Gamma_2(\bar{\omega}_t \Delta t)\bar{a}_t)^\wedge \Delta t^2 \\
\Phi^l_{22} &= \Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t) \\
\Phi^l_{32} &= \Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t)\Delta t \\
\Phi^l_{33} &= \Gamma_0^\mathsf{T}(\bar{\omega}_t \Delta t)
\end{aligned}$$

The GPS measurement model corresponds to the left-invariant observation form:

$$Y_k = \bar{X}_k b + V_k$$

$$\begin{bmatrix} y_k \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{R}_k & \bar{v}_k & \bar{p}_k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} v_k \\ 0 \\ 0 \end{bmatrix}$$

Using the Left-Invariant EKF equations, we proceed to find H.

$$\begin{aligned}
H\xi^r_k = \xi^{r\wedge}_k b &= \begin{bmatrix} \xi^{\omega\wedge}_k & \xi^v_k & \xi^p_k \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \xi^p \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & I \\ 0_{1,3} & 0_{1,3} & 0_{1,3} \\ 0_{1,3} & 0_{1,3} & 0_{1,3} \end{bmatrix} \begin{bmatrix} \xi^\omega_k \\ \xi^v_k \\ \xi^p_k \end{bmatrix},
\end{aligned}$$

and in its reduced form, is

$$H = \begin{bmatrix} 0 & 0 & I \end{bmatrix}.$$

7.6 Right-Invariant Extended Kalman Filter

Next, we derive the Right-Invariant Extended Kalman Filter (RI-EKF) similarly.

7.6.1 Propagation

In the propagation step, the mean can be directly computed using the process model. However, for the covariance propagation, we use the right-invariant log-linear error dynamics. As such, the state's mean evolves on the group while the covariance is tracked in the Lie algebra.

$$\frac{d}{dt}\bar{X}_t = f_{u_t}(\bar{X}_t), \quad t_{k-1} \leq t < t_k$$

$$\frac{d}{dt}\eta^r_t = g_{u_t}(\eta^r_t) - (\mathrm{Ad}_{\bar{X}_t} w_t)^\wedge \eta^r_t \implies \frac{d}{dt}\xi^r_t = A^r_t \xi^r_t - \mathrm{Ad}_{\bar{X}_t} w_t$$

$$\frac{d}{dt}P^r_t = A^r_t P^r_t + P^r_t A^{r\mathsf{T}}_t + \mathrm{Ad}_{\bar{X}_t} Q_t \mathrm{Ad}^\mathsf{T}_{\bar{X}_t}$$

7.6.2 Update

In the update step, we use $\exp(\xi) \approx I + \xi^\wedge$ and neglect the higher order terms to derive the linear update error equation as follows.

$$\bar{X}^+_{t_k} = \exp\left(L_{t_k}\left(\bar{X}_{t_k} Y_{t_k} - b\right)\right)\bar{X}_{t_k}$$

$$\eta^{r+}_{t_k} = \exp\left(L_{t_k}\left(\eta^r_{t_k} b - b + \bar{X}_{t_k} V_t\right)\right)\eta^r_{t_k}$$

$$I + \xi^{r+\wedge}_{t_k} = \left(I + \left(L_{t_k}\left((I + \xi^{r\,\wedge}_{t_k})b - b + \bar{X}_{t_k} V_t\right)\right)^\wedge\right)(I + \xi^{r\,\wedge}_{t_k})$$

$$\xi^{r+\wedge}_{t_k} = \xi^{r\,\wedge}_{t_k} + \left(L_{t_k}\left((I + \xi^{r\,\wedge}_{t_k})b - b + \bar{X}_{t_k} V_t\right)\right)^\wedge$$

$$\xi^{r+}_{t_k} = \xi^r_{t_k} + L_{t_k}\left(\xi^{r\,\wedge}_{t_k} b + \bar{X}_{t_k} V_t\right)$$

To arrange all terms according to the vector form of ξ, define the measurement Jacobian, H, such that

$$\boxed{H\xi = -\xi^\wedge b}.$$

Then it follows that

$$\bar{X}^+_{t_k} = \exp\left(L_{t_k}\left(\bar{X}_{t_k} Y_{t_k} - b\right)\right)\bar{X}_{t_k}$$

$$\xi^{r+}_{t_k} = \xi^r_{t_k} - L_{t_k} H \xi^r_{t_k} + L_{t_k}\bar{X}_{t_k} V_t$$

$$\xi^{r+}_{t_k} = (I - L_{t_k} H)\xi^r_{t_k} + L_{t_k}\bar{X}_{t_k} V_t$$

$$P^{r+}_{t_k} = (I - L_{t_k} H)P^r_{t_k}(I - L_{t_k} H)^\mathsf{T} + L_{t_k}\bar{N}_k L^\mathsf{T}_{t_k}$$

where

$$\bar{N}_k := \bar{X}_{t_k}\mathrm{Cov}[V_k]\bar{X}^\mathsf{T}_{t_k}.$$

7.6.3 RI-EKF Result

To summarize, we have the following two steps for an RI-EKF.

1. Propagation:

$$\frac{d}{dt}\bar{X}_t = f_{u_t}(\bar{X}_t), \quad t_{k-1} \le t < t_k \tag{7.12}$$

$$\frac{d}{dt}P^r_t = A^r_t P^r_t + P^r_t A^{r\,\mathsf{T}}_t + \mathrm{Ad}_{\bar{X}_t} Q_t \mathrm{Ad}^\mathsf{T}_{\bar{X}_t} \tag{7.13}$$

2. Update:

$$\bar{X}^+_{t_k} = \exp\left(L_{t_k}\left(\bar{X}_{t_k} Y_{t_k} - b\right)\right)\bar{X}_{t_k} \tag{7.14}$$

$$P^{r+}_{t_k} = (I - L_{t_k} H)P^r_{t_k}(I - L_{t_k} H)^\mathsf{T} + L_{t_k}\bar{N}_k L^\mathsf{T}_{t_k} \tag{7.15}$$

where

$$L_{t_k} = P^r_{t_k} H^\mathsf{T} S^{-1}, \quad S = H P^r_{t_k} H^\mathsf{T} + \bar{N}_k.$$

Given these equations, once we know A^r_t and H matrices, we can implement the RI-EKF.

Example 7.3 (Right-Invariant EKF Localization)**.** *A robot operates in a known 2D environment with point landmarks. The robot has 3 DOF, and the state space is* SE(2)*. The sensor provides the relative 2D position of nearby landmarks. We use the velocity motion model and landmarks measurement model within a Right-Invariant EKF to localize the robot.*

The robot pose at any time-step is $X_k = \begin{bmatrix} R_k & p_k \\ 0 & 1 \end{bmatrix} \in \mathrm{SE}(2)$*. For the prediction step, we discretize the velocity motion model:*

$$\bar{X}_{k+1} = \bar{X}_k \exp(u_k^\wedge) \quad \textit{motion model}$$

$$\Phi = \exp(A_t^r \Delta t) = I \quad \textit{transition matrix}$$

$$P_{k+1} = \Phi P_k \Phi^\mathsf{T} + \mathrm{Ad}_{\bar{X}_k} Q_d \mathrm{Ad}_{\bar{X}_k}^\mathsf{T} \quad \textit{covariance propagation}$$
$$P_{k+1} = P_k + \mathrm{Ad}_{\bar{X}_k} Q_d \mathrm{Ad}_{\bar{X}_k}^\mathsf{T}$$

$$Q_d \approx \Phi Q_t \Phi^\mathsf{T} \Delta t = Q_t \Delta t \quad \textit{discrete noise covariance}$$

The global map of landmarks, $m \in \mathbb{R}^2$*, is given. The relative landmark measurement model corresponds to the right-invariant observation form:*

$$Y_k = \bar{X}_k^{-1} b + V_k$$

$$\begin{bmatrix} y_k^1 \\ y_k^2 \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{R}_k^\mathsf{T} & -\bar{R}_k^\mathsf{T} p_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} m \\ 1 \end{bmatrix} + \begin{bmatrix} v_k \\ 0 \end{bmatrix}$$

Using the Right-Invariant EKF equations, we proceed to find H*.*

$$H\xi_k^r = -\xi_k^{r\wedge} b = -\begin{bmatrix} 0 & -\xi_k^\omega & \xi_k^{v_1} \\ \xi_k^\omega & 0 & \xi_k^{v_2} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} m^1 \\ m^2 \\ 1 \end{bmatrix}$$
$$= -\begin{bmatrix} -\xi_k^\omega m^2 + \xi_k^{v_1} \\ \xi_k^\omega m^1 + \xi_k^{v_2} \\ 0 \end{bmatrix} = \begin{bmatrix} m^2 & -1 & 0 \\ -m^1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_k^\omega \\ \xi_k^{v_1} \\ \xi_k^{v_2} \end{bmatrix}$$

$$H = \begin{bmatrix} m^2 & -1 & 0 \\ -m^1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

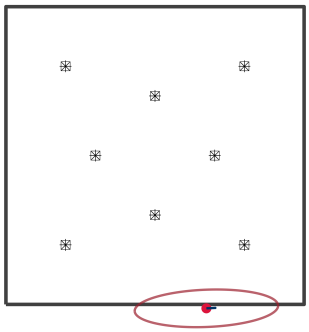

Figure 7.1: Known 2D map with point landmarks. See `riekf_localization_se2.m` for details and code.

Remark 7.3. *Notice that since the map is known,* H *is constant. The last row corresponds to the homogeneous coordinates and can be removed during the implementation.*

Now let's look into the observability of the linearized system. Using $\Phi = I$

and $H = \begin{bmatrix} m^2 & -1 & 0 \\ -m^1 & 0 & -1 \end{bmatrix}$*, the discrete-time observability matrix is*

$$\mathcal{O} = \begin{bmatrix} H \\ H\Phi \\ H\Phi^2 \\ \vdots \end{bmatrix} = \begin{bmatrix} m^2 & -1 & 0 \\ -m^1 & 0 & -1 \\ m^2 & -1 & 0 \\ -m^1 & 0 & -1 \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

The first column is a linear combination of the second and third columns and, therefore, the first dimension (orientation) is unobservable. Using only one landmark the robot heading angle is not observable. How many landmarks make the robot pose fully observable?

To resolve the observability problem, we use two landmarks, $m_1, m_2 \in \mathbb{R}^2$*, in each correction step. The stacked right-invariant observation model becomes:*

$$\begin{bmatrix} Y_{1,k} \\ Y_{2,k} \end{bmatrix} = \begin{bmatrix} \bar{X}_k^{-1} & 0 \\ 0 & \bar{X}_k^{-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} V_k \\ V_k \end{bmatrix},$$

and the stacked measurement Jacobian, H*, becomes*

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} = \begin{bmatrix} m_1^2 & -1 & 0 \\ -m_1^1 & 0 & -1 \\ m_2^2 & -1 & 0 \\ -m_2^1 & 0 & -1 \end{bmatrix}$$

Remark 7.4. *It is easy to verify that the observability Gramian has rank* 3 *which makes the pose observable.*

7.7 Switching Between Left and Right-Invariant Errors

We can switch between the left and right error forms through the use of the adjoint map.

$$\begin{aligned} \eta_t^r &= \bar{X}_t X_t^{-1} = \bar{X}_t \eta_t^l \bar{X}_t^{-1} \\ \exp(\xi_t^r) &= \bar{X}_t \exp(\xi_t^l) \bar{X}_t^{-1} = \exp(\mathrm{Ad}_{\bar{X}_t} \xi_t^l) \\ \xi_t^r &= \mathrm{Ad}_{\bar{X}_t} \xi_t^l \end{aligned}$$

Remark 7.5. *This transformation is exact, which means that we can easily switch between the covariance of the left and right invariant errors using*

$$P_t^r = \mathrm{Ad}_{\bar{X}_t} P_t^l \mathrm{Ad}_{\bar{X}_t}^{\mathsf{T}}.$$

7.8 Summary

- Use it if the process dynamics naturally evolves on a Lie group;

- Works well in practice despite the fact by the addition of noise and calibration parameters the theoretical result is lost;
- Excellent consistency and no spurious correlation among state and parameters.
- Highly efficient and suitable for high-frequency state estimation tasks.

7.9 Further Readings

Body velocity measurements provide a generic correction model that can work on any robotic platform. However, accurate body velocity measurement is not readily available. Specifically, the filer requires the ground-referenced body velocity (Teng et al., 2021; Potokar et al., 2021).

The robot's nonholonomic constraints (i.e., velocity constraints that cannot be integrated) can provide pseudo observations (Dissanayake et al., 2001) that can significantly improve performance. However, these constraints are assumptions and detached from the robot's behavior. Learning such constraints provides a way to use sensory inputs instead of assumptions (Brossard et al., 2019, 2020; Lin et al., 2022).

Moreover, the nonholonomic constraints are violated when the robot drifts. Slip detection and friction estimation are challenging and necessary tasks for robot state estimation (Yu et al., 2022).

Problems

7.1 Derive the deterministic right- and left-invariant error dynamics in Theorem 7.1 by direct calculation. *Hint: Define the invariant error and take its derivative with respect to time.*

7.2 Prove Theorem 7.1.

7.3 Derive the noisy error dynamics in (7.6). *Hint:* $\bar{X}_t$ *is a realization of the state at time* t *and is not random; therefore,* $\frac{d}{dt}\bar{X}_t = f_{u_t}(\bar{X}_t)$. *On the other hand,* X *is a random variable and we have* $\frac{d}{dt}X_t = f_{u_t}(X_t) + X_t w_t^\wedge$.

7.4 Show that the reconstruction equations in (4.1), $\dot{X} = X\xi^\wedge$, and (4.2), $\dot{X} = \xi^\wedge X$, satisfy the group affine property.

7.5 Derive the right- and left-invariant error dynamics and their linearized log-linear forms in the Lie algebra for the reconstruction equations in (4.1) and (4.2).

7.6 Perform the observability analysis of the IMU-GPS example. If there are unobservable state variables, how to resolve the observability problem?

7.7 Repeat Example 7.3 in a 3D map where the robot has 6 DOF, and the state space is SE(3). Draw a conclusion and discuss each step in detail.

7.8 Suppose you are using a right-invariant EKF to perform robot localization in 3D and the ground truth is given in coordinates using x, y, z for position and roll, pitch, yaw for the orientation with respect to x, y, and z axes, respectively. The Right-Invariant EKF error and covariance are in the Lie algebra of SE(3). Derive a model to map the error and its covariance from $\mathfrak{se}(3)$ to $\mathbb{R}^6$ (x, y, z, roll, pitch, yaw). Show work and provide necessary equations for mapping the error and covariance from the Lie algebra to the Cartesian space.

8
Particle Filtering

8.1 Nonlinear Dynamic Systems and Uncertainty Propagation

We consider nonlinear systems. In the deterministic case, the system is described using the nonlinear process model, $f(u_k, x_{k-1})$, and measurement model, $h(x_k)$, as

$$\begin{aligned} x_k &= f(u_k, x_{k-1}), \\ z_k &= h(x_k). \end{aligned} \tag{8.1}$$

In addition, we consider the nonlinear dynamic system in (6.8) excited by noise, w_k and v_k, as

$$\begin{aligned} x_k &= f(u_k, x_{k-1}, w_k), \\ z_k &= h(x_k, v_k). \end{aligned}$$

In this chapter, we explore a class of methods based on random sampling known as Monte Carlo methods. In particular, we study sequential Monte Carlo methods (Particle Filters).

We will use the delta function in the following sections. The Dirac delta function, for $x \in \mathbb{R}$, is defined by the properties

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) \mathrm{d}x = 1.$$

For any smooth function f and $a \in \mathbb{R}$, we have:

$$\int_{-\infty}^{\infty} \delta(x - a) f(x) = f(a),$$

which is a Lebesgue integral with respect to the measure δ (thought as a point mass). This can be generalized to $\mathbb{R}^n$ or any set with a similar idea to define δ measure or unit mass concentrated at a point.

8.2 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods (Doucet et al., 2001) are a set of simulation-based methods for computing posterior distributions. In this framework, observations arrive sequentially in time, and we wish to perform online inference. The posterior distribution is updated as data become available (recursive Bayesian estimation/learning). SMC methods are used when dealing with non-Gaussian, high-dimensionality, and nonlinearity where often obtaining an analytical solution is not possible. In addition, SMC methods can be used for inferring both filtering and smoothing posterior distributions.

8.2.1 Perfect Monte Carlo Sampling

Suppose we can simulate n independent and identically distributed (i.i.d.) random samples (particles), $\{x^i_{0:k}\}^n_{i=1}$ according to $p(x_{0:k} \mid z_{1:k})$. An empirical estimate of this distribution is given by

$$p(x_{0:k} \mid z_{1:k}) \approx p_n(x_{0:k} - x^{1:n}_{0:k} \mid z_{1:k}) = \frac{1}{n}\sum_{i=1}^{n} \delta(x_{0:k} - x^i_{0:k}).$$

We note that each delta term places infinite density at the corresponding random sample. The integration of this empirical density results in n, hence, normalization by $\frac{1}{n}$. Then, the following integral can be computed:

$$\mathrm{I}_n(f) = \int f(x_{0:k}) p_n(x_{0:k} - x^{1:n}_{0:k} \mid z_{1:k}) = \frac{1}{n}\sum_{i=1}^{n} f(x^i_{0:k}).$$

If the posterior variance is bounded, i.e., $\sigma^2_f := \mathbb{E}_{p(x_{0:k}|z_{1:k})}[f^2(x_{0:k})] - \mathrm{I}^2_n(f) < \infty$, then $\mathbb{V}[\mathrm{I}_n(f)] = \frac{\sigma^2_f}{n}$. From the law of large numbers, $n \to \infty$, almost surely, $\mathrm{I}_n(f) \to \mathrm{I}(f)$. Therefore, This estimate is unbiased. Moreover, if $\sigma^2_f < \infty$, then a central limit theorem holds; that is $n \to \infty$, $\sqrt{n}(\mathrm{I}_n(f) - \mathrm{I}(f))$ converges in distribution to $\mathcal{N}(0, \sigma^2_f)$.

This result is important because we can easily estimate any quantity $\mathrm{I}(f)$ while the rate of convergence is independent of the integrand dimension. In particular, any deterministic numerical integration method has a rate of convergence that decreases as the dimension of the integrand increases. However, in practice, it is usually impossible to sample efficiently from the posterior distribution $p(x_{0:k} \mid z_{1:k})$. In the next section, we use the idea of importance sampling to resolve the problem that we do not know the posterior distribution, or we cannot sample from it.

8.2.2 Importance Sampling

We introduce an *importance sampling distribution* (also called *proposal distribution*), $\pi(x_{0:k} \mid z_{1:k})$. We also assume the support of $\pi(x_{0:k} \mid z_{1:k})$

includes the support of $p(x_{0:k} \mid z_{1:k})$. We get

$$\mathrm{I}(f) = \frac{\int f(x_{0:k}) w(x_{0:k}) \pi(x_{0:k} \mid z_{1:k}) \mathrm{d}x_{0:k}}{\int w(x_{0:k}) \pi(x_{0:k} \mid z_{1:k}) \mathrm{d}x_{0:k}} = \frac{\int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}}{\int p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k}} = \int f(x_{0:k}) p(x_{0:k}|z_{1:k}) \mathrm{d}x_{0:k},$$

where $w(x_{0:k})$ is known importance weight:

$$w(x_{0:k}) = \frac{p(x_{0:k} \mid z_{1:k})}{\pi(x_{0:k} \mid z_{1:k})}$$

Next, by drawing n i.i.d. particles according to $\pi(x_{0:k}|z_{1:k})$ (we know π and we can draw samples from it),

$$\hat{\mathrm{I}}_n(f) = \frac{\frac{1}{n}\sum_{i=1}^n f(x_{0:k}^i) w(x_{0:k}^i)}{\frac{1}{n}\sum_{i=1}^n w(x_{0:k}^i)} = \sum_{i=1}^n f(x_{0:k}^i) \tilde{w}_k^i,$$

where the *normalized importance weights* are given by

$$\tilde{w}_k^i = \frac{w(x_{0:k}^i)}{\sum_{i=1}^n w(x_{0:k}^i)}.$$

However, this is not adequate for recursive estimation.

8.3 Sequential Importance Sampling (SIS)

Let us factor the importance sampling distribution as follows.

$$\pi(x_{0:k}|z_{1:k}) = \pi(x_{0:k-1}|z_{1:k-1}) \pi(x_k|x_{0:k-1}, z_{1:k}) = \pi(x_0) \prod_{j=1}^k \pi(x_j|x_{0:j-1}, z_{1:j}).$$

Then

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i \frac{p(z_k|x_k^i) p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{0:k-1}^i, z_{1:k})} \quad \left(w(x_{0:k}) = \frac{p(x_{0:k}|z_{1:k})}{\pi(x_{0:k}|z_{1:k})} \right).$$

Remark 8.1. *Recall that* $p(x_{0:k}^i|z_{1:k}) = p(x_{0:k-1}^i|z_{1:k-1}) \frac{p(z_k|x_k^i) p(x_k^i|x_{k-1}^i)}{p(z_k|z_{1:k-1})}$, *where we used the Markov assumption and causality of the models in the Bayes' rule.*

In filtering, we often set $\pi(x_k|x_{0:k-1}, z_{1:k}) = \pi(x_k|x_{k-1}, z_k)$ so that the importance sampling distribution only depends on x_{k-1} and z_k.

8.3.1 SIS Particle Filter

At this point, we have successfully derived the SIS particle filter algorithm.

Algorithm 4: `sis-particle-filter`

Require: particles $\mathcal{X}_{k-1} = \{x^i_{k-1}, w^i_{k-1}\}^n_{i=1}$, measurement z_k;
1: $\mathcal{X}_k \leftarrow \varnothing$
2: **for** each $x^i_{k-1} \in \mathcal{X}_{k-1}$ **do**
3: $x^i_k \sim \pi(x^i_k | x^i_{k-1}, z_k)$ ▷ sample from proposal distribution
4: $w^i_k \leftarrow w^i_k \frac{p(z_k|x^i_k)p(x^i_k|x^i_{k-1})}{\pi(x^i_k|x^i_{k-1},z_k)}$ ▷ update importance weights
5: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x^i_k, w^i_k\}$ ▷ add i-th weighted sample to the new set
6: **end for**
7: **return** $\mathcal{X}_k$

Unfortunately, the `sis-particle-filter` algorithm suffers from the *degeneracy problem*. As time increases, the distribution of the of the weights, $\tilde{w}^i_k$ becomes more and more skewed, in practice, reducing to one particle with non-zero weight after a few iterations. We can define the following measure of degeneracy using the effective sample size as

$$n_{\text{eff}} = \frac{1}{\sum_{i=1}^{n} (\tilde{w}^i_k)^2} \quad 1 < n_{\text{eff}} < n.$$

At the two extreme cases we have: 1) all particles have the same weights (uniform): $\forall i \in \{1 : n\}, \tilde{w}^i_k = \frac{1}{n} \Longrightarrow n_{\text{eff}} = n$; 2) the entire distribution mass is placed in one particle (singular): $\forall i \in \{1 : j-1, j+1 : n\}, \tilde{w}^i_k = 0$ and $\tilde{w}^j_k = 1 \Longrightarrow n_{\text{eff}} = 1$.

8.3.2 Resampling and Generic Particle Filter Algorithm

The *resampling* idea was introduced to fix the degeneracy problem. Resampling eliminates particles with low weights and multiplies particles with high weights. Although the resampling step reduces the effect of degeneracy, it introduces a new problem known as *sample impoverishment*. It limits the parallel implementation of the algorithm since all particles must be combined. In addition, the particles with high weights are selected many times, leading to the loss of diversity, i.e., loss of alternative hypotheses. This is because eventually, most particles are initiated from a few initial parent particles.

SIS particle filter combined with the resampling idea leads to the Sample Importance Resampling (SIR) algorithm. A generic version of the SIR particle filter is shown in Algorithm 5.

Algorithm 5: generic-particle-filter

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, measurement z_k, resampling threshold n_t (e.g., $n/3$);

$\mathcal{X}_k \leftarrow \varnothing$
for each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
  draw $x_k^i \sim \pi(x_k^i | x_{k-1}^i, z_k)$ ▷ sample from proposal distribution
  $w_k^i \leftarrow \tilde{w}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\pi(x_k^i|x_{k-1}^i, z_k)}$ ▷ update importance weights
end for
$w_\mathsf{total} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
$\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_\mathsf{total}\}_{i=1}^n$ ▷ add weighted samples to the new set
$n_\mathsf{eff} \leftarrow 1/\sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size
if $n_\mathsf{eff} < n_\mathsf{t}$ **then**
  $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
end if
return $\mathcal{X}_k$

The low-variance resampling is a simple and effective algorithm to use in practice. Its algorithmic implementation is shown in Algorithm 6.

Algorithm 6: low-variance-resampling

Require: particles $\mathcal{X}_k = \{x_k^i, \tilde{w}_k^i\}_{i=1}^n$;

$w_\mathsf{c} \leftarrow$ compute the vector of cumulative sum of the weights using $\{\tilde{w}_k^i\}_{i=1}^n$ ▷ w_c is the Cumulative Distribution Function (CDF)
$r \leftarrow \mathtt{rand}(0, n^{-1})$ ▷ draw a uniform random number between 0 and n^{-1}
$j \leftarrow 1$ ▷ dummy index to climb the CDF and select particles
for all $i \in \{1 : n\}$ **do**
  $u \leftarrow r + (i-1)n^{-1}$ ▷ move along the CDF
  while $u > w_\mathsf{c}^j$ **do**
    $j \leftarrow j + 1$
  end while
  $x_k^i \leftarrow x_k^j$ ▷ replicate the survived particle
  $\tilde{w}_k^i \leftarrow n^{-1}$ ▷ set the weight to n^{-1} (uniform distribution)
end for
return $\mathcal{X}_k$

8.3.3 SIR Particle Filter Algorithm in Robotics

A special case is when the importance sampling distribution is chosen to be the prior distribution, or in robotics the robot motion model $p(x_k|x_{k-1}, u_k)$. Note that u_k is deterministic, and the motion model does not depend on the

measurement z_k. Then the weights can be computed using

$$\tilde{w}_k^i \propto \tilde{w}_{k-1}^i p(z_k|x_k^i).$$

Algorithm 7: `sir-particle-filter`

Require: particles $\mathcal{X}_{k-1} = \{x_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, action u_k, measurement z_k, resampling threshold n_t (e.g., $n/3$);

1: $\mathcal{X}_k \leftarrow \varnothing$
2: **for** each $x_{k-1}^i \in \mathcal{X}_{k-1}$ **do**
3: draw $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$ ▷ sample from motion model
4: $w_k^i \leftarrow \tilde{w}_{k-1}^i p(z_k|x_k^i)$ ▷ update importance weights
5: **end for**
6: $w_\mathsf{total} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
7: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{x_k^i, w_k^i / w_\mathsf{total}\}_{i=1}^n$ ▷ add weighted samples to the new set
8: $n_\mathsf{eff} \leftarrow 1/\sum_{i=1}^n (\tilde{w}_k^i)^2$ ▷ compute effective sample size
9: **if** $n_\mathsf{eff} < n_\mathsf{t}$ **then**
10: $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles with higher weights
11: **end if**
12: **return** $\mathcal{X}_k$

Example 8.1 (PF Target Tracking)**.** *A target is moving in a 2D plane. The ownship position is known and fixed at the origin. We have access to relative noisy range and bearing measurements of the target position at any time step.*

$$x_k = f(u_k, x_{k-1}) + w_k = x_{k-1} + w_k,$$

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k^{1\,2} + x_k^{2\,2}} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k,$$

$$Q_k = 0.1\ I_2,\ R_k = \text{diag}(0.05^2, 0.01^2).$$

We estimate the target position using a Particle Filter (PF) as shown in Figure 8.1. See `pf_single_target.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

Example 8.2 (PF Target Tracking, Constant Velocity Motion Model)**.** *There is no knowledge of the target motion, but this time, we assume a constant velocity random walk motion model and estimate the target velocity along with the position.*

$$x_k = f(u_k, x_{k-1}) + w_k = F_k x_{k-1} + w_k,$$

$$F_k = \begin{bmatrix} I & \Delta t I \\ 0 & I \end{bmatrix} \qquad \Delta t : \textit{sampling time},$$

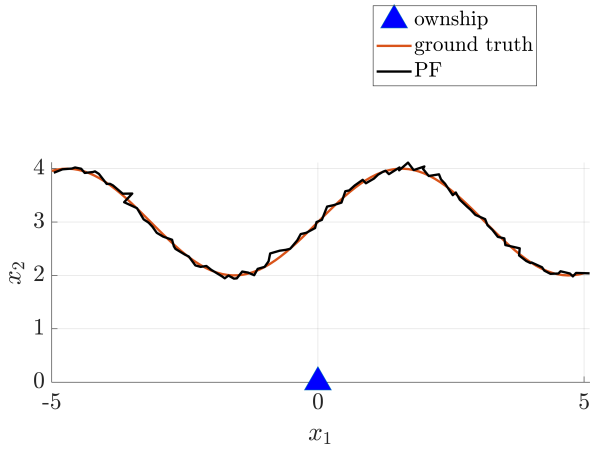

Figure 8.1: 2D target tracking using a PF.

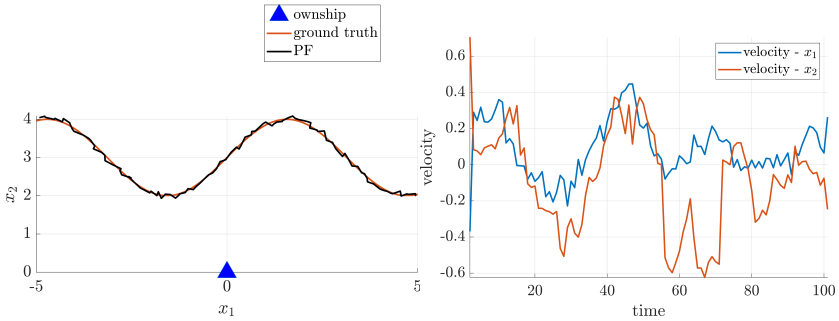

Figure 8.2: 2D target tracking with a constant velocity motion model using a PF.

$$z_k = h(x_k) + v_k = \begin{bmatrix} \sqrt{x_k^{1^2} + x_k^{2^2}} \\ \text{atan2}(x_k^1, x_k^2) \end{bmatrix} + v_k,$$

$$Q_k = \text{diag}(0.1^2, 0.1^2, 0.01^2, 0.01^2), \; R_k = \text{diag}(0.05^2, 0.01^2).$$

We estimate the target position and velocity using a PF as shown in Figure 8.2. See `pf_single_target_cv.m` *(or Python version) for code. Try to change the parameters and study the filter's behavior.*

8.4 Summary

SMC methods can solve complex nonlinear, non-Gaussian online estimation problems. For example, dealing with global uncertainty in robot localization and solving the "kidnapped robot" problem (see Thrun et al. (2005, Chapters 7 and 8)). The algorithms are applicable to a very large class of models and are often straightforward to implement. The price to pay for this simplicity is inefficiency in some application domains. For further reading, see also Barfoot (2017, Chapter 4).

8.5 Further Readings

Problems

8.1

8.2

8.3

8.4

8.5

8.6

8.7

9
Localization

See Chapters 7 and 8 of Probabilistic Robotics [1]. For summary slides, see the course repository [2]. The Kalman filtering approaches can be updated via the Invariant EKF and generally filtering on Lie groups. The particle filtering approach can also use Lie group methods by properly defining the state, noise, and measurement error.

Ignoring problem-specific setups, landmark-based Kalman filtering is currently out of favor as a sole solution because of its linearization error and vulnerability to outliers in data association if the landmark signatures are not known a priori. Although the invariant EKF addresses the linearization error for the group affine systems, the data association challenge remains unsolved [3]. At this point, the problem is approached via factor graphs, similar to graph SLAM formulations, and solved using robust optimization techniques. If a map is given, particle filtering is still an attractive solution and widely used.

Another interesting localization method enabled by offline training of deep neural networks is called image-based localization (Kendall and Cipolla, 2016; Patel et al., 2018; Song et al., 2022). This method does not use a sequence of relative motion and relative noisy measurements to estimate the global pose of the robot or a camera. Instead, a database of images with pose labels is created, and a deep neural network is trained to learn features that can associate input images with global poses. This problem is similar to SLAM systems' place recognition [4] or relocalization modules (Gálvez-López and Tardós, 2012). As such, it can be combined with graph SLAM or particle filtering as a global source of information.

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005, vol. 1

[2] https://github.com/UMich-CURLY-teaching/UMich-ROB-530-public/blob/main/slides/11_Localization.pdf

[3] This is why a proprioceptive Invariant EKF (Hartley et al., 2020; Lin et al., 2022; Yu et al., 2022) is attractive for dead reckoning. The odometry solution is independent of perceptual data and can boost other perception tasks such as localization.

[4] Place recognition does not necessarily include pose estimation, and it can be limited to detection.

10

Mapping

11
Unconstrained Optimization and Smoothing

12
Sensor Registration

Sensor registration is defined as finding the transformation between two sensors with overlapping measurements or tracking a moving sensor using its sequential measurements. This problem frequently arises in engineering domains such as point cloud registration (Chen and Medioni, 1991; Censi, 2008; Segal et al., 2009a; Stoyanov et al., 2012; Servos and Waslander, 2017; Parkison et al., 2018; Zaganidis et al., 2018; Parkison et al., 2019a), visual odometry and SLAM (Nistér et al., 2006; Scaramuzza and Fraundorfer, 2011; Fraundorfer and Scaramuzza, 2012; Strasdat, 2012; Kerl et al., 2013; Engel et al., 2015; Ghaffari et al., 2019; Rosinol et al., 2019), satellite image registration (Kim and Im, 2003; Bentoutou et al., 2005), and photogrammetry (Mikhail et al., 2001). In terms of applications, the above-mentioned problems are fundamental to many computer vision (Hartley and Zisserman, 2003; Szeliski, 2010) and robotics (Thrun et al., 2005; Barfoot, 2017) algorithms.

12.1 Point Cloud Registration

[Maani: To be added ...]

12.2 RGBD Visual Odometry

RGB-D sensors are essential for acquiring rich spatial data that combines color and depth information, enabling robotics, computer vision, and augmented reality applications. These sensors can be categorized into active and passive systems. Active systems, like Time-of-Flight and structured light cameras, project light onto the scene and measure the reflected light to compute depth. They offer reliable depth data even in low light but can be affected by limited range and interference. Passive systems, such as stereo cameras, rely on natural illumination and estimate depth through the triangulation of corresponding features from multiple viewpoints. They require sufficient texture and lighting, making them less suitable for poorly lit or textureless environments.

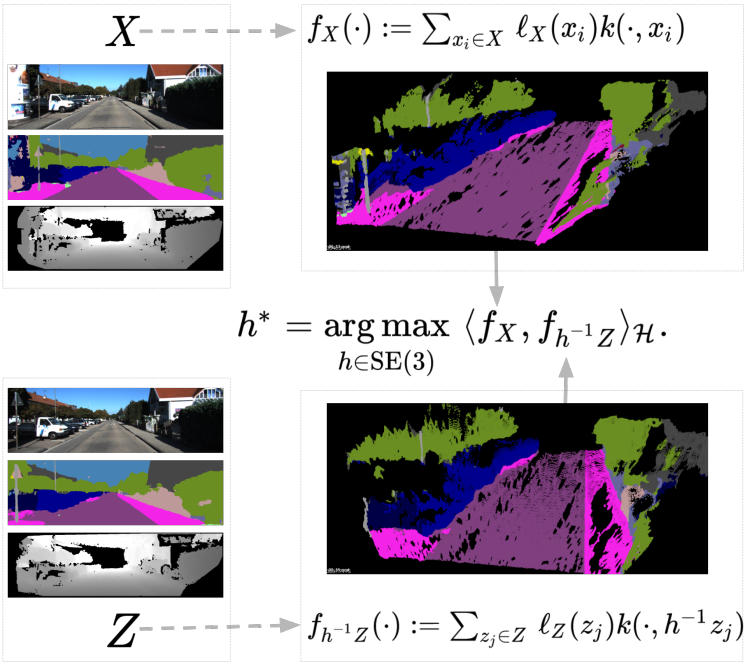

Figure 12.1: Point clouds X and Z are represented by two continuous functions f_X, f_Z in an RKHS. Each point x_i has its own semantic labels, $\ell_X(x_i)$, encoded in the corresponding function representation via a tensor product representation (Zhang et al., 2021). The registration is formulated as maximizing the inner product between two point cloud functions.

Acquiring accurate and reliable depth data from RGB-D sensors presents challenges, including sensor noise, incomplete depth measurements, and misalignment between color and depth images. Preprocessing techniques like filtering, inpainting, and registration exist to enhance data quality and improve subsequent registration and visual odometry performance. Understanding the characteristics and limitations of different RGB-D sensors and their data acquisition process is crucial for designing robust and efficient algorithms that effectively exploit the information provided by these sensors.

[**Maani: To be added ...**]

12.3 RKHS Registration for Spatial-Semantic Perception

Point clouds obtained by modern sensors such as RGB-D cameras, stereo cameras, and LIDARs contain up to $300,000$ points per scan at $10 - 60\,\mathrm{Hz}$ and rich color and intensity (reflectivity of a material sensed by an active light beam) measurements besides the geometric information. In addition, *deep learning* (LeCun et al., 2015) can provide semantic attributes of the scene as measurements (Long et al., 2015; Chen et al., 2017; Zhu et al., 2019).

Illustrated in Figure 12.1, the following formulation provides a general framework for lifting semantically labeled point clouds into a function space to solve a registration problem (Ghaffari et al., 2019; Clark et al., 2021; Zhang et al., 2021). Consider two (finite) collections of points, $X = \{x_i\}$, $Z = \{z_j\} \subset \mathbb{R}^3$. We want to determine which element $h \in \mathrm{SE}(3)$, aligns the two point clouds X and $hZ = \{hz_j\}$ the "best." To assist with this, we

will assume that each point contains information described by a point in an inner product space, $(\mathcal{I}, \langle\cdot,\cdot\rangle_{\mathcal{I}})$. To this end, we will introduce two labeling functions, $\ell_X : X \to \mathcal{I}$ and $\ell_Z : Z \to \mathcal{I}$. To measure their alignment, we turn the point clouds, X and Z, into functions $f_X, f_Z : \mathbb{R}^3 \to \mathcal{I}$ that live in some RKHS, $(\mathcal{H}, \langle\cdot,\cdot\rangle_{\mathcal{H}})$. The action, $\mathrm{SE}(3) \curvearrowright \mathbb{R}^3$ induces an action $\mathrm{SE}(3) \curvearrowright \mathcal{H}$ by $h.f(x) := f(h^{-1}x)$. Inspired by this observation, we will set $h.f_Z := f_{h^{-1}Z}$.

Problem 12.1. *The problem of aligning the point clouds can now be rephrased as maximizing the scalar products of f_X and $h.f_Z$, i.e., we want to solve*

$$\underset{h\in\mathrm{SE}(3)}{\arg\max}\ F(h), \quad F(h) := \langle f_X, f_{h^{-1}Z}\rangle_{\mathcal{H}}. \tag{12.1}$$

12.3.1 Constructing The Functions

For the kernel of our RKHS, $\mathcal{H}$, we first choose the squared exponential kernel $k : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$:

$$k(x,z) = \sigma^2 \exp\left(\frac{-\|x-z\|_3^2}{2\ell^2}\right), \tag{12.2}$$

for some fixed real parameters (hyperparameters) σ and ℓ (the *length-scale*), and $\|\cdot\|_3$ is the standard Euclidean norm on $\mathbb{R}^3$. This allows us to turn the point clouds to functions via $f_X(\cdot) := \sum_{x_i\in X} \ell_X(x_i)k(\cdot, x_i)$ and $f_{h^{-1}Z}(\cdot) := \sum_{z_j\in Z} \ell_Z(z_j)k(\cdot, h^{-1}z_j)$. Here $\ell_X(x_i)$ encodes the semantic information, for example, LIDAR intensity and image pixel color. $k(\cdot, x_i)$ encodes the geometric information. We can now obtain the inner product of f_X and f_Z as

$$\langle f_X, f_{h^{-1}Z}\rangle_{\mathcal{H}} := \sum_{x_i\in X, z_j\in Z} \langle \ell_X(x_i), \ell_Z(z_j)\rangle_{\mathcal{I}} \cdot k(x_i, h^{-1}z_j). \tag{12.3}$$

We use the kernel trick (Murphy, 2012) to substitute the inner products in (12.3) with the semantic kernel as

$$\langle f_X, f_{h^{-1}Z}\rangle_{\mathcal{H}} = \sum_{x_i\in X, z_j\in Z} k_c(\ell_X(x_i), \ell_Z(z_j)) \cdot k(x_i, h^{-1}z_j).$$

We choose k_c to be the squared exponential kernel with real hyperparameters σ_c and ℓ_c that are set independently.

12.3.2 Feature Embedding via Tensor Product Representation

We now extend the feature space to a hierarchical distributed representation to incorporate the full geometric and hierarchical semantic relationship between the two point clouds. Let $(V_1, V_2, \dots)$ be different inner product spaces describing different types of non-geometric features of a point, such as color, intensity, and semantics. Their tensor product, $V_1 \otimes V_2 \otimes \dots$,

is also an inner product space. For any $x \in X, z \in Z$ with features $\ell_X(x) = (u_1, u_2, \dots)$ and $\ell_Z(z) = (v_1, v_2, \dots)$, with $u_1, v_1 \in V_1$, $u_2, v_2 \in V_2, \dots$, we have

$$\begin{aligned}\langle \ell_X(x), \ell_Z(z) \rangle_{\mathcal{I}} &= \langle u_1 \otimes u_2 \otimes \dots, v_1 \otimes v_2 \otimes \dots \rangle \\ &= \langle u_1, v_1 \rangle \cdot \langle u_2, v_2 \rangle \cdot \dots . \end{aligned} \tag{12.4}$$

By substituting (12.4) into (12.3), we obtain

$$\langle f_X, f_{h^{-1}Z} \rangle_{\mathcal{H}} = \sum_{x_i \in X, z_j \in Z} \langle u_{1i}, v_{1j} \rangle \cdot \langle u_{2i}, v_{2j} \rangle \dots k(x_i, h^{-1} z_j).$$

After applying the kernel trick, we arrive at

$$\begin{aligned}\langle f_X, f_{h^{-1}Z} \rangle_{\mathcal{H}} &= \sum_{x_i \in X, z_j \in Z} k(x_i, h^{-1} z_j) \cdot \prod_k k_{V_k}(u_{ki}, v_{kj}) \\ &:= \sum_{x_i \in X, z_j \in Z} k(x_i, h^{-1} z_j) \cdot c_{ij}. \end{aligned} \tag{12.5}$$

Each c_{ij} does not depend on the relative transformation. It is worth noting that, when choosing the squared exponential kernel and when the input point clouds have only geometric information, c_{ij} will be identity, and (12.5) has the same formulation as Kernel Correlation (Tsin and Kanade, 2004).

12.3.3 Equivariance Property

If instead of working with the inverse of the transformation acting on the function basis, we work with the function input, then the equivariance property becomes evident. Let $\mathcal{C}(\mathbb{R}^3)$ be the set of point clouds on $\mathbb{R}^3$ and $\mathcal{H}$ be the RKHS. Let $f : \mathcal{C}(\mathbb{R}^3) \to \mathcal{H}$ be our map which assigns a function to a point cloud. Consider the space of smooth functions on $\mathbb{R}^3$, $C^\infty(\mathbb{R}^3)$, and let the group $\mathcal{G}$ act on $\mathbb{R}^3$. The action lifts to an action on $C^\infty(\mathbb{R}^3)$ via $g.f(x) = f(g^{-1}x)$, $g \in \mathcal{G}$. This inverse is needed to make the action a *group* action:

$$(hg).f(x) = h.f(g^{-1}x) = f(g^{-1}h^{-1}x) = f((hg)^{-1}x), \quad h, g \in \mathcal{G}.$$

Now let Z be a point cloud and f_Z be its associated function. If $\mathcal{G}$ acts on $\mathbb{R}^3$ via isometries, then $k(gx, gz) = k(x, z)$ and we have

$$g.f_Z(x) = f_Z(g^{-1}x) = \sum_j \ell_Z(z_j) \cdot k(g^{-1}x, z_j) = \sum_j \ell_j \cdot k(x, g z_j) = f_{gZ}(x).$$

12.3.4 Experimental Results

We present the point cloud registration experiments on real-world outdoor and indoor datasets: KITTI (Geiger et al., 2012) odometry and TUM RGB-D data set (Sturm et al., 2012), with the following setup: All experiments are performed in a frame-to-frame manner without skipping images. The first

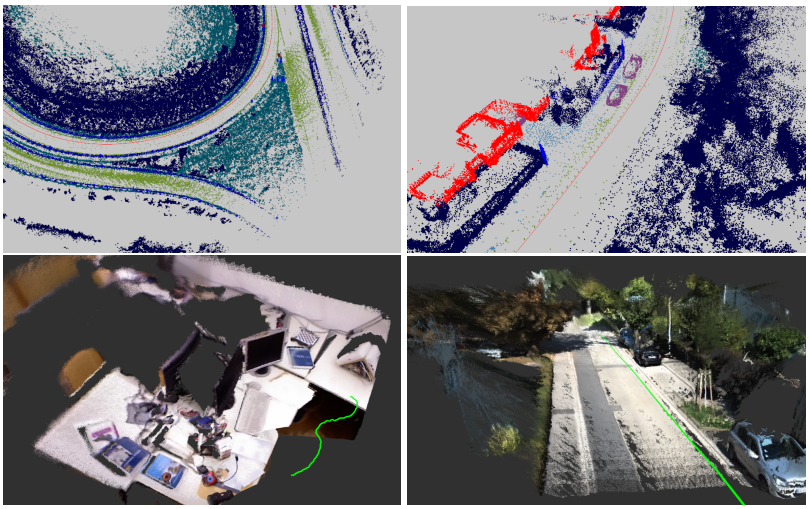

Figure 12.2: Stacked semantic and color point clouds based on frame-to-frame registration results using KITTI (Geiger et al., 2012) LiDAR, TUM RGB-D (Sturm et al., 2012) and KITTI Stereo sensors.

	t (%)	**r** (°/m)
Geometric Registration (Proposed)	4.55	0.0236
GICP (Segal et al., 2009b)	11.23	0.0452
3D-NDT (Magnusson et al., 2007)	8.50	0.0396
Robust-ICP (Zhang et al., 2022)	11.02	0.0256
Color Registration (Proposed)	3.69	0.0159
MC-ICP (Servos and Waslander, 2014)	14.10	0.0488
Semantic Registration (Proposed)	**3.64**	**0.0155**

Table 12.1: Results of the proposed frame-to-frame method using the KITTI (Geiger et al., 2012) stereo odometry benchmark averaged over Sequence 00-10. The table lists the average drift in translation, as a percentage (%), and rotation, in degrees per meter(°/m). The drifts are calculated for all possible subsequences of 100, 200...., 800 meters.

frame's transformation is initialized with identity, and all later frames start with the previous frames' results. The same hyperparameter values, such as the lengthscale of the kernels in (12.2), are used for the proposed registration methods within one data set.

All the baselines except Robust-ICP (Zhang et al., 2022) use all the pixels without downsampling because they do not provide an optimal point selection scheme. Fast-Robust-ICP and the proposed methods select a subset of pixels via OpenCV's FAST (Rosten and Drummond, 2006) feature detector to reduce the frame-wise running time.

The qualitative and quantitative results on KITTI Stereo is provided in Figure 12.2, Figure 12.3, and Table 12.1, respectively. The baselines are GICP (Segal et al., 2009b), Multichannel-ICP (Servos and Waslander, 2014), 3D-NDT (Magnusson et al., 2007), and Robust-ICP (Zhang et al., 2022). GICP and NDT are compared with our geometric registration method (*Geometric CVO*, i.e., $\ell_X(x_i) = \ell_Z(z_j) = 1$). Multichannel-ICP competes with our color-assisted registration method (*Color CVO*). GICP and 3D-NDT implementation are from PCL (Rusu and Cousins, 2011). The Robust-ICP implementation is from its open-source Github repository. The Multichannel-ICP

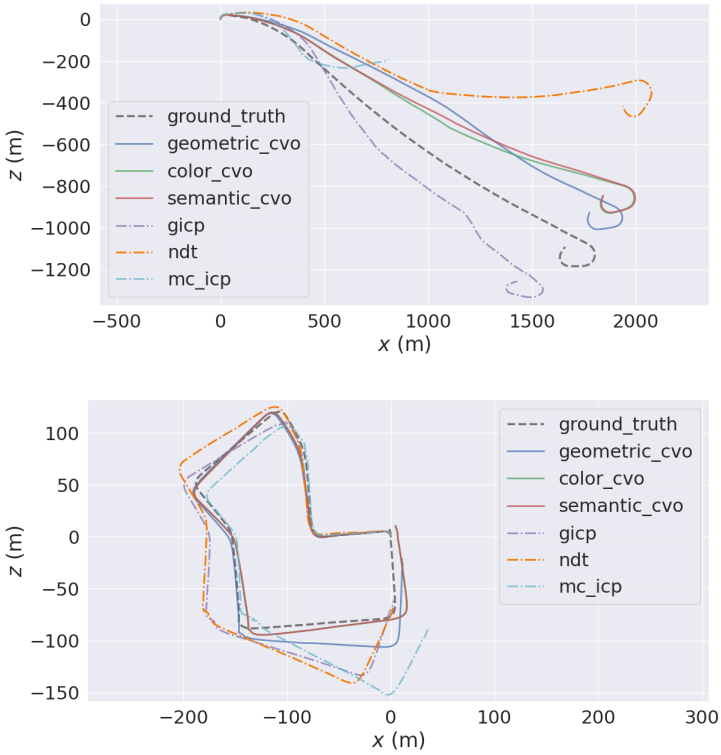

Figure 12.3: An illustration of the proposed registration methods on KITTI Stereo (Geiger et al., 2012) sequence 01 (left) and 07 (right) versus the baselines. The black dashed trajectory is the ground truth. The dot-dashed trajectories are the baselines. Plotted with EVO (Grupp, 2017).

implementation is from (Parkison et al., 2019b). The semantic predictions of the images come from Nvidia's pre-trained neural network (Zhu et al., 2019), which was trained on 200 labeled images on KITTI. The depth values of the stereo images are generated with ELAS (Geiger et al., 2010). All the baselines and the proposed methods remove the first 100 rows of image pixels that mainly include sky pixels, as well as points that are more than 55 meters away. Averaged over sequence 00 to 10, our geometric method has a lower translational error (4.55%) comparing to the GICP (11.23%), NDT (8.50%), and Robust-ICP (11.02%). Our color version has a lower average translational drift (3.69%) than Multichannel-ICP (14.10%). If we add semantic information, the error is further reduced (3.64%). In addition, excluding the image I/O and point cloud generation operations, the proposed implementations take, on average, 1.4 sec per frame on GPU when registering less than 15k downsampled points. Fast-Robust-ICP also takes downsampled point clouds and takes 0.3 sec per frame on CPU. GICP, NDT, and Multichannel-ICP on CPU use full point clouds (150k-350k points), and take 6.3 sec, 6.6 sec, and 57 sec per frame, respectively.

The qualitative and quantitative results on TUM RGB-D is provided in Figure 12.2 and Table 12.2, respectively. We evaluated our method on the `fr1` sequences, which are recorded in an office environment, and `fr3` sequences, which contain image sequences in structured/nostructured and

	fr1		fr3 structure v.s texture	
	t (m/sec)	**r** (deg/sec)	**t** (m/sec)	**r** (deg/sec)
Geometric Registration (Proposed)	0.0730	**2.3805**	0.0794	2.8536
GICP (Segal et al., 2009b)	0.4034	15.8838	0.2116	5.2979
3D-NDT (Magnusson et al., 2007)	0.2290	14.0311	0.2487	6.9860
Robust-ICP (Zhang et al., 2022)	0.1487	6.6911	0.2091	5.4168
Color Registration (Proposed)	**0.0545**	2.4333	**0.0754**	2.6651
DVO (Kerl et al., 2013)	0.0623	2.6943	0.1386	4.9843
Color ICP (Park et al., 2017)	0.1353	5.8985	0.0820	**2.2041**

Table 12.2: The RMSE of Relative Pose Error (RPE) averaged over TUM RGB-D (Sturm et al., 2012) `fr1` and `fr3` structure v.s. texture sequences. The **t** columns show the RMSE of the translational drift in m/sec and the **r** columns show the RMSE of the rotational error in deg/sec. The RMSE is averaged over all sequences.

texture/notextured environments. We use the same baselines for geometric registration as KITTI. We compare Color CVO with Dense Visual Odometry (DVO) (Kerl et al., 2013) and Color ICP (Park et al., 2017). We reproduced DVO results with the code from (Pizenberg, 2019). The Color ICP implementation is taken from Open3D (Zhou et al., 2018). From Table 12.2, the proposed geometric registration outperforms the geometric baselines and achieves a similar performance to DVO and Color ICP. Moreover, with color information, the average error of the proposed registration decreases.

12.3.5 Discussions and Limitations

Results in Section 12.3.4 demonstrate that embedding features like color and semantics in function representations provide finer data associations. Specifically, in (12.5), the extra appearance information c_{ij} encodes the similarity in color or semantics between the two associated points. It eliminates pairwise associations whose color or semantic appearances do not agree. Moreover, each point $x_i \in X$ is matched to multiple points $z_j \in Z$. The proposed color registration significantly improves over geometric-only methods in both KITTI Stereo and TUM RGB-D datasets.

One limitation of the proposed method is the computational complexity introduced by the double sum in (12.5). However, the double sum is sparse because a point $x_i \in X$ is far away from most of the points $z_j \in Z$, either in the spatial (geometry) space or one of the feature (semantic) spaces. But this similarity still has to be calculated with the help of GPU implementations or K-nearest-neighbor search (Blanco and Rai, 2014). In practice, an efficient point selection mechanism like FAST (Rosten and Drummond, 2006) corner selector or DSO's (Engel et al., 2017) image gradient-based pixel selector can reduce the computation time. Alternatively, representation learning can be a way to reduce the number of input points while providing richer features.

12.4 Further Readings

Problems

12.1

12.2

12.3

12.4

12.5

12.6

12.7

13
Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics and autonomous systems, where the objective is to concurrently build a map of the environment and estimate the position and orientation of the robot or sensor platform. SLAM plays a crucial role in various applications, including search and rescue, home automation, and industrial automation.

13.1 Objectives

- Understanding SLAM as:
  - Dynamic Bayes Network
  - Factor Graphs
  - Least squares
- Maximum a-Posteriori (MAP) solution.
- Enabling you to read papers and understand literature based on these ideas.

13.2 Notation and Weighted Norm

We define $\|r\|_{\Sigma}^2 := r^\top \Sigma^{-1} r$. Σ (covariance) is symmetric and PSD. $\Sigma = LL^\top$, L is the lower triangular Cholesky factor of Σ.

$$\Sigma^{-1} = \left(LL^\top\right)^{-1} = L^{-\top} L^{-1}.$$

Remark 13.1. $(AB)^{-1} = B^{-1}A^{-1}$ *and* $A^{-\top} = \left(A^\top\right)^{-1} = \left(A^{-1}\right)^\top$.

We can convert the weighted norm to the Euclidean norm as follows.

$$\|r\|_L^2 = r^\top \Sigma^{-1} r = r^\top L^{-\top} L^{-1} r = \left(L^{-1} r\right)^\top \left(L^{-1} r\right) = \left\| L^{-1} r \right\|^2, \quad (13.1)$$

where the Euclidean norm is given by

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}, \quad x \in \mathbb{R}^n.$$

$$\begin{aligned}
x &= x_{0:k} = \{x_0, x_1, \ldots, x_k\} && \text{State variables} \\
z &= z_{1:k} = \{z_1, z_2, \ldots, z_k\} && \text{Measurements} \\
p(x,z) &= p\left(x_{0:k}, z_{1:k-1}, z_k\right) && \text{Joint distribution} \\
&= p(z_k \mid \underbrace{x_{0:k}, z_{1:k-1}}_{\text{Markov assumption}}) p\left(x_{0:k}, z_{1:k-1}\right) \\
&= p\left(z_k \mid x_k\right) p\left(x_{0:k-1}, x_k, z_{1:k-1}\right) \\
&= p\left(z_k \mid x_k\right) p(x_k \mid \underbrace{x_{0:k-1}, z_{1:k-1}}_{\text{Markov assumption}}) \underbrace{p(x_{0:k-1}, z_{1:k-1})}_{\text{recursive term}} \\
&= p\left(z_k \mid x_k\right) p\left(x_k \mid x_{k-1}\right) p\left(x_{0:k-1}, z_{1:k-1}\right) \\
&= \boxed{p\left(x_0\right) \prod_{i=1}^{k} p\left(x_i \mid x_{i-1}\right) p\left(z_i \mid x_i\right) = p(x,z)}
\end{aligned}$$

Factorization; prior, motion, and measurement models.

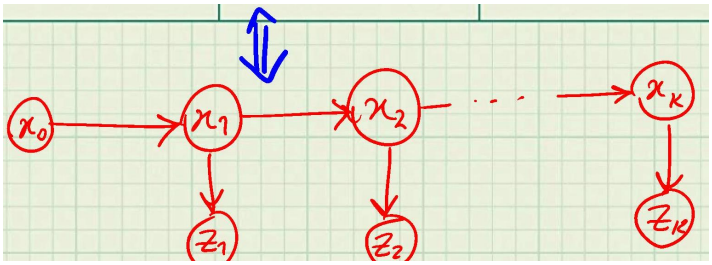

Figure 13.1: Dynamic Bayes Network.

Example 13.1. *In Figure 13.1,*

$p\left(x_0\right)$ *is prior;*
$p\left(x_1 \mid x_0\right)$ *is the motion model;*
$p\left(z_1 \mid x_1\right)$ *is the measurement mode;*
$p\left(x_2 \mid x_1\right)$ *is the motion model.*

Assumption 13.1. *Gaussian noise model.*

$$\begin{cases} x_i = f\left(x_{i-1}, u_i\right) + w_i, \quad w_i \sim \mathcal{N}\left(0, \Sigma_{w_i}\right) \\ p\left(x_i \mid x_{i-1}, u_i\right) \propto \exp\left(-\frac{1}{2} \left\| f\left(x_{i-1}, u_i\right) - x_i \right\|_{\Sigma_{w_i}}^2\right) \end{cases} \tag{13.2}$$

$$\begin{cases} z_i = h\left(x_i\right) + v_i, \quad v_i \sim \mathcal{N}\left(0, \Sigma_{v_i}\right) \\ p\left(z_i \mid x_i\right) \propto \exp\left(-\frac{1}{2} \left\| h\left(x_i\right) - z_i \right\|_{\Sigma_{v_i}}^2\right) \end{cases} \tag{13.3}$$

13.3 Factor Graphs

$$p(x,z) \propto \phi_0\left(x_0\right) \psi\left(x_0, x_1\right) \psi\left(x_1, x_2\right) \alpha\left(x_1, z_1\right) \alpha\left(x_2, z_2\right). \tag{13.4}$$

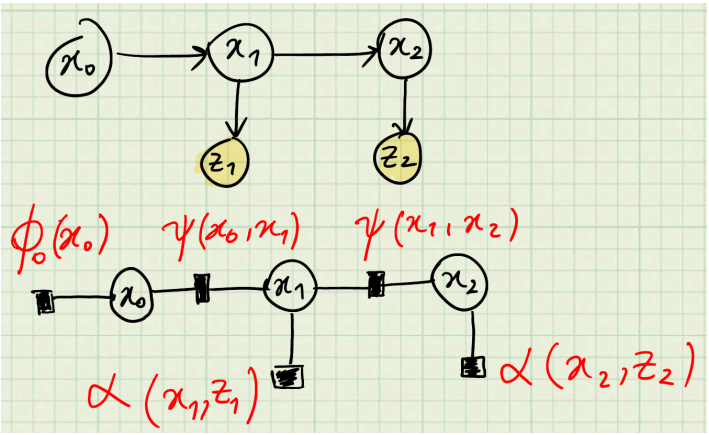

Figure 13.2: Factor graphs and dynamic Bayes network equivalency.

Remark 13.2. *Relationship between the dynamic Bayes network and factor graph.*

$$\begin{aligned}
&\phi_0(x_0) \propto p(x_0) \\
&\psi(x_{i-1}, x_i) \propto p(x_i \mid x_{i-1}) \\
&\alpha(x_i, z_i) \propto p(z_i \mid x_i).
\end{aligned}$$

- Graphical modeling is an easy way to describe conditional independence and write the joint distribution.
- Factor graphs are particularly convenient as the joint distribution is the product of factors.
- Each factor is a prior, motion, or measurement model.

13.4 MAP Estimation

$$\begin{aligned}
x^* &= \arg\max_x p(x \mid z) = \arg\max_x p(x, z) \\
&= \arg\min_x -\log p(x, z)
\end{aligned} \tag{13.5}$$

Remark 13.3. $p(x \mid z) = \frac{p(x,z)}{p(z)}$.

The factor graph formulation and MAP estimation provide a general sensor fusion framework.

$$p(x, z) = p(x_0) \prod_{i=1}^{M} p(x_i \mid x_{i-1}, u_i) \prod_{j=1}^{K} p(z_j \mid x_{ji}).$$

$$\begin{aligned}
x^* &= \arg\min_x -\log p(x, z) \\
&= \arg\min_x \frac{1}{2} \|\bar{x}_0 - x_0\|^2_{\Sigma_0} + \frac{1}{2} \|f(x_{i-1}, u_i) - x_i\|^2_{\Sigma_{w_i}} + \frac{1}{2} \|h(x_{ji}) - z_j,\|^2_{\Sigma_{v_j}}.
\end{aligned}$$

Where the prior is

$$p\left(x_0\right) \propto \exp\left(-\frac{1}{2}\left\|\bar{x}_0 - x_0\right\|_{\Sigma_0}^2\right)$$

and

$$\begin{aligned}
-\log p\left(x_0\right) &\propto \frac{1}{2}\left\|\bar{x}_0 - x_0\right\|_{\Sigma_0}^2, \\
-\log p\left(x_i \mid x_{i-1}, u_i\right) &\propto \frac{1}{2}\left\|f\left(x_{i-1}, u_i\right) - x_i\right\|_{\Sigma_{w_i}}^2, \\
-\log p\left(z_j \mid x_{ji}\right) &\propto \frac{1}{2}\left\|h\left(x_{ji}\right) - z_j\right\|_{\Sigma_{v_j}}^2.
\end{aligned}$$

Remark 13.4. *$\bar{x}_0$ is the mean prior. In many problems, we can set $\bar{x}_0 = 0$ as the origin.*

Bibliography

R. Zhang, T.-Y. Lin, C. E. Lin, S. A. Parkison, W. Clark, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "A new framework for registration of semantic point clouds from stereo and RGB-D cameras," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2021, pp. 12 214–12 221.

A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 573–580.

M. Grupp, "EVO: Python package for the evaluation of odometry and SLAM." https://github.com/MichaelGrupp/evo, 2017.

Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

B. D. Anderson and J. B. Moore, *Optimal filtering*. Englewood Cliffs, 1979.

"ROB 101: Computational linear algebra," https://robotics.umich.edu/academic-program/course-offerings/rob101-fall-2021/, accessed: 2022-01-01.

C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT press, 2006, vol. 1.

S. Boyd, "Lecture notes for EE263," *Introduction to Linear Dynamical Systems*, 2008.

T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 1991.

A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. McGraw-Hill Education, 2002.

S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005, vol. 1.

T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, vol. 50, no. 1, pp. 5–43, 2003.

S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011.

K. Tapp, *Matrix groups for undergraduates*. American Mathematical Soc., 2016, vol. 79.

B. Hall, *Lie groups, Lie algebras, and representations: an elementary introduction*. Springer, 2015, vol. 222.

L. W. Tu, *An Introduction to Manifolds*. Springer, 2011.

T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

A. M. Bloch, *Nonholonomic Mechanics and Control*, P. S. Krishnaprasad and R. M. Murray, Eds. Springer, New York, NY, 2015.

A. W. Long, K. C. Wolfe, M. J. Mashner, and G. S. Chirikjian, "The banana distribution is Gaussian: A localization study with exponential coordinates," *Proceedings of the Robotics: Science and Systems Conference*, 2012.

R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," http://mars.cs.umn.edu/tr/reports/Trawny05b.pdf, Dept. of Computer Science & Engineering University of Minnesota, Minneapolis, MN 55455, Tech. Rep. 2005–002, 2005.

J. Solà, "Quaternion kinematics for the error-state Kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017.

C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

J. G. Mangelson, M. Ghaffari, R. Vasudevan, and R. M. Eustice, "Characterizing the uncertainty of jointly distributed poses in the Lie algebra," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1371–1388, 2020.

R. Mahony and J. Trumpf, "Equivariant filter design for kinematic systems on lie groups," *IFAC-PapersOnLine*, vol. 54, no. 9, pp. 253–260, 2021.

M. Brossard, A. Barrau, P. Chauchat, and S. Bonnabel, "Associating uncertainty to extended poses for on Lie group IMU preintegration with rotating earth," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 998–1015, 2022.

A. Barrau, "Non-linear state error based extended Kalman filters with applications to navigation," Ph.D. dissertation, Mines Paristech, 2015.

A. Barrau and S. Bonnabel, "Invariant Kalman filtering," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

S. Bonnabel, P. Martin, and P. Rouchon, "Non-linear symmetry-preserving observers on Lie groups," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1709–1713, 2009.

S. Teng, M. W. Mueller, and K. Sreenath, "Legged robot state estimation in slippery environments using invariant extended kalman filter with velocity update," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2021, pp. 3104–3110.

E. R. Potokar, K. Norman, and J. G. Mangelson, "Invariant extended kalman filtering for underwater navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5792–5799, 2021.

G. Dissanayake, S. Sukkarieh, E. Nebot, and H. Durrant-Whyte, "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications," *IEEE transactions on robotics and automation*, vol. 17, no. 5, pp. 731–747, 2001.

M. Brossard, A. Barrau, and S. Bonnabel, "RINS-W: Robust inertial navigation system on wheels," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2019, pp. 2068–2075.

——, "AI-IMU dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

T.-Y. Lin, R. Zhang, J. Yu, and M. Ghaffari, "Legged robot state estimation using invariant Kalman filtering and learned contact events," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 1057–1066.

X. Yu, S. Teng, T. Chakhachiro, W. Tong, T. Li, T.-Y. Lin, S. Koehler, M. Ahumada, J. M. Walls, and M. Ghaffari, "Fully proprioceptive slip-velocity-aware state estimation for mobile robots via invariant Kalman filtering and disturbance observer," *arXiv preprint arXiv:2209.15140*, 2022.

A. Doucet, N. De Freitas, N. J. Gordon *et al.*, *Sequential Monte Carlo methods in practice*. Springer, 2001, vol. 1, no. 2.

A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 4762–4769.

M. Patel, B. Emery, and Y.-Y. Chen, "ContextualNet: Exploiting contextual information using LSTMs to improve image-based localization," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 5890–5896.

J. Song, M. Patel, and M. Ghaffari, "Fusing convolutional neural network and geometric constraint for image-based indoor localization," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1674–1681, 2022.

D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, pp. 1188–1197, 2012.

Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 2724–2729.

A. Censi, "An ICP variant using a point-to-line metric," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 19–25.

A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proceedings of the Robotics: Science and Systems Conference*, Seattle, USA, June 2009.

T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *International Journal of Robotics Research*, vol. 31, no. 12, pp. 1377–1393, 2012.

J. Servos and S. L. Waslander, "Multi-Channel Generalized-ICP: A robust framework for multi-channel scan registration," *Robotics and Autonomous Systems*, vol. 87, pp. 247–257, 2017.

S. A. Parkison, L. Gan, M. Ghaffari Jadidi, and R. M. Eustice, "Semantic iterative closest point through expectation-maximization," in *Proceedings of the British Machine Vision Conference*, Newcastle, UK, September 2018, pp. 1–17.

A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak, "Integrating deep semantic segmentation into 3-d point cloud registration," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2942–2949, 2018.

S. A. Parkison, M. Ghaffari, L. Gan, R. Zhang, A. K. Ushani, and R. M. Eustice, "Boosting shape registration algorithms via reproducing kernel Hilbert space regularizers," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4563–4570, 2019.

D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

D. Scaramuzza and F. Fraundorfer, "Visual odometry," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

H. Strasdat, "Local accuracy and global consistency for efficient visual SLAM," Ph.D. dissertation, Imperial College London, 2012.

C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 1935–1942.

M. Ghaffari, W. Clark, A. Bloch, R. M. Eustice, and J. W. Grizzle, "Continuous direct sparse visual odometry from RGB-D images," in *Proceedings of the Robotics: Science and Systems Conference*, Freiburg, Germany, June 2019.

A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," *arXiv preprint arXiv:1910.02490*, 2019.

T. Kim and Y.-J. Im, "Automatic satellite image registration by combination of matching and random sample consensus," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 1111–1117, 2003.

Y. Bentoutou, N. Taleb, K. Kpalma, and J. Ronsin, "An automatic image registration for applications in remote sensing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 9, pp. 2127–2137, 2005.

E. M. Mikhail, J. S. Bethel, and J. C. McGlone, *Introduction to modern photogrammetry*. Wiley, 2001.

R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8856–8865.

W. Clark, M. Ghaffari, and A. Bloch, "Nonparametric continuous sensor registration," *Journal of Machine Learning Research*, vol. 22, no. 271, pp. 1–50, 2021.

K. P. Murphy, *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *European conference on computer vision*. Springer, 2004, pp. 558–569.

J. Zhang, Y. Yao, and B. Deng, "Fast and robust iterative closest point," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 07, pp. 3450–3466, 2022.

E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the European Conference on Computer Vision*. Springer, 2006, pp. 430–443.

A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proceedings of the Robotics: Science and Systems Conference*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

J. Servos and S. L. Waslander, "Multi channel generalized-ICP," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2014, pp. 3644–3649.

R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9-13 2011.

S. A. Parkison, M. Ghaffari, L. Gan, R. Zhang, A. K. Ushani, and R. M. Eustice, "Boosting shape registration algorithms via reproducing kernel Hilbert space regularizers," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4563–4570, 2019.

A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proceedings of the Asian Conference on Computer Vision*, 2010.

J. Park, Q.-Y. Zhou, and V. Koltun, "Colored point cloud registration revisited," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 143–152.

M. Pizenberg, "DVO (without ROS dependency)," https://github.com/mpizenberg/dvo/tree/76f65f0c9b438675997f595471d39863901556a9, 2019.

Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees," https://github.com/jlblancoc/nanoflann, 2014.

J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.