Contents

1 Notations **4**

2 Statistics **6**

2.1 Mean 6

2.2 Variance 6

2.3 Standard Deviation 7

2.4 Standard Error 7

2.5 Pearson's Chi-Squared Test 7

2.5.1 Goodness of Fit 8

2.5.2 Problems with Pearson's Chi-Squared Test [1] 9

3 Linear Algebra **12**

3.1 Trace 12

3.2 Rank 12

3.3 Condition Number 12

4 Probability Theory **15**

4.1 Bayes Theorem 15

5 Computer Vision **16**

5.1 Fundamentals 16

5.1.1 Point on Line 16

5.1.2 Intersection of Lines 16

5.1.3 Plane 16

5.1.4 Fundamental Matrix 17

5.1.5 Essential Matrix 17

5.2 Pinhole Camera Model 17

5.3 Radial Tangential Distortion 18

5.3.1 Radial Tangential Point Jacobian 19

5.3.2 Radial Tangential Parameter Jacobian 19

5.4 Equi-distant Distortion 20

5.4.1 Equi-distant Point Jacobian 20

5.4.2 Equi-distant Parameter Jacobian 21

5.5 Linear Triangulation 21

5.6 Bundle Adjustment 22

5.7 Illumination Invariant Transform 25

6 Feature Detection and Correspondance **26**

6.1 FAST 26

6.2 ORB Feature Descriptor and Matching 27

6.3 KLT Feature Tracker 28

7 Rotations **31**

7.1 Euler Angles 31

7.2 Quaternions 31

7.2.1 Main Quaternion Properties 32

7.2.2 Quaternion from Two Vectors 34

7.2.3 Quaternion to Rotation Matrix 35

7.2.4 Rotation Matrix to Quaternion 36

7.3 Differential Calculus 36

8 Optimization 38

8.1 Linear Least Squares 38

8.2 Non-linear Least Squares 39

8.3 Gauge 40

8.3.1 Gauge fixation 40

8.3.2 Gauge prior 41

8.3.3 Free gauge 41

9 Calibration 42

9.1 Jacobian w.r.t Sensor Pose, $\mathbf{T}_{\text{WS}}$ 43

9.2 Jacobian w.r.t Sensor-Camera Extrinsics, $\mathbf{T}_{\text{SC}}$ 43

9.3 Jacobian w.r.t Fiducial Pose, $\mathbf{T}_{\text{WF}}$ 44

10 State Estimation 46

10.1 Bayes Filter 46

10.2 2D Range Measurement Model 46

10.3 2D Relative Bearing Measurement Model 49

11 Inertial Measurement Unit (IMU) 51

11.1 IMU kinematics 51

Chapter 1

Notations

A large part of robotics is about developing machines that perceives and interact with the environment. For that robots use sensors to collect and process data, and knowing what the data describes is of utmost importance. Imagine obtaining the position of the robot but not knowing what that position is with respect to. Missing data descriptions such as what a position vector is expressing, what is it with respect to and more causes many hours of painful trail and error to extract that information.

In the following section the notation used throughout this document will be described, and it follows closely of that of Paul Furgale's [2]. The aim is to mitigate the ambiguity that arises when describing robot poses, sensor data and more.

A vector expressed in the world frame, $\mathcal{F}_{\text{W}}$, is written as ${}_{\text{W}}\mathbf{r}$. Or more precisely if the vector describes the position of the camera frame, $\mathcal{F}_{\text{C}}$, expressed in $\mathcal{F}_{\text{W}}$, the vector can be written as ${}_{\text{W}}\mathbf{r}_{\text{WC}}$ with W and C as start and end points. The left hand subscripts indicates the coordinate system the vector is expressed in, while the right-hand subscripts indicate the start and end points. For brevity if the vector has the same start point as the frame to which it is expressed in, the same vector can be written as ${}_{\text{W}}\mathbf{r}_{\text{C}}$. Similarly a transformation of a point from $\mathcal{F}_{\text{C}}$ to $\mathcal{F}_{\text{W}}$ can be represented by a homogeneous transform matrix, $\mathbf{T}_{\text{WC}}$, where its rotation matrix component is written as $\mathbf{C}_{\text{WC}}$ and the translation component written as ${}_{\text{W}}\mathbf{r}_{\text{WC}}$. A rotation matrix that is parametrized by quaternion $\mathbf{q}_{\text{WC}}$ is written as $\mathbf{C}\{\mathbf{q}_{\text{WC}}\}$.

Position:	${}_{\mathrm{W}}\mathbf{r}_{\mathrm{WB}}$
Velocity:	${}_{\mathrm{W}}\mathbf{v}_{\mathrm{WB}}$
Acceleration:	${}_{\mathrm{W}}\mathbf{a}_{\mathrm{WB}}$
Angular velocity:	${}_{\mathrm{W}}\boldsymbol{\omega}_{\mathrm{WB}}$
Rotation:	$\mathbf{C}_{\mathrm{WB}}$
Transform:	$\mathbf{T}_{\mathrm{WB}}$
Point:	${}_{\mathrm{W}}\mathbf{p}$

Chapter 2

Statistics

2.1 Mean

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.1}$$

2.2 Variance

Variance is a measure of spread, it is the average squared distance between measurements and the mean.

The Population variance is defined as,

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}. \tag{2.2}$$

The Sample variance is defined as,

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}. \tag{2.3}$$

Where N is the population size and n is the sample size.

In plain english, the population variance, standard deviation, etc, is when the whole population data is available. Sample variance, standard deviation, etc, on the other hand implies the availble data is only a subset of the population.

The problem with sample data is that estimates without applying the $n-1$ correction will often underestimate the true value. This is particularly true if n is small. By using $n-1$ instead of n as the divsor corrects the estimate by making the result slightly larger.

The degrees of freedom is another perspective of why $n-1$ should be use when estimating from sample data. In statistics the degrees of freedom describe the number variables that could affect the response or output of a model. (need more explaining)

2.3 Standard Deviation

The standard deviation σ is defined as the square root of the variance σ^2. It is also a measure of spread in the data. However standard deviation is often more intuitive, because instead of the spread in terms of distance from the *mean squared*, the standard deviation is simply the distance from the mean (not *mean squared*).

$$\sigma = \sqrt{\sigma^2} \tag{2.4}$$

2.4 Standard Error

Standard error of the regression, or standard error of the estimate is the average distance between observed values and the fitted model. This metric conveys objectively how wrong the regression model is on average using the units of the response variable. A small standard error indicates observations are closer to the fitted model. Conceptually it is similar to the standard deviation, the difference is standard deviation measures te average distance of the observed alues from the mean.

The standard error (S) is defined as:

$$S = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{df}} \tag{2.5}$$

2.5 Pearson's Chi-Squared Test

Pearson's chi-squared test χ^2 is a statistical test applied to sets of *categorial data* to quantify the likelihood that the observed difference between measured and predicted arose by chance. The test is used to assess three types of comparisons:

- **Goodness of fit**: checks whether the observed frequency distribution matches the theoretical distribution.
- **Homogeneity**: compares the distribution of counts for two or more groups using the same categorical variable.
- **Independence**: checks whether the unparied observations on two variables, expressed in a contingency table, are independent of each other.

2.5.1 Goodness of Fit

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = N \sum_{i=1}^{n} \frac{(O_i/N - p_i)^2}{p_i} \tag{2.6}$$

where χ^2 is Pearson's cumulative test statistic, O_i is the number of observations of type i, N is the total number of observations, $E_i = Np_i$ is the expected (theoretical) count of type i, and n is the number of cells in the table.

Once the chi-squared test statistic is calculated, the p-value is obtained by comparing the value of the statistic to a chi-squared distribution. The number of degrees of freedom is equal to the number of cells n, minus the reduction in degrees of freedom, p.

Example: Fairness of dice

A 6-sided die is thrown 60 times. The number of times it lands with 1, 2, 3, 4, 5 and 6 face up is 5, 8, 9, 8, 20 and 20, respectively. Is the die biased, according to the Pearson's chi-squared test at a significance level of 95%, and, or 99%?

In this example $n = 6$ as there are 6 possible outcomes, 1 to 6. The null hypothesis is that the die is unbaised, hence each dice number is expected to occur the same number of times, in this case, $60/n = 10$. The outcomes can be tabulated as follows:

The number of degrees of freedom is $n - 1 = 5$. The Upper tail critical values of chi-square distribution gives a critical value of 11.070 at 95% significance level: As the chi-squared statistic of 13.4 exceeds this critical value, the null hypothesis is rejected and conclude that the die is biased at 95% significance level. At 99% significance level, the critical value is 15.086. As the chi-squared statistic does not exceed it, the null hypothesis is not rejected and thus conclude that there is insufficient evidence to show that the die is biased at 99% significance level.

i	O_i	E_i	$O_i - E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
1	5	10	-5	25	2.5
2	8	10	-2	4	0.4
3	9	10	-1	1	0.1
4	8	10	-2	4	0.4
5	10	10	0	0	0
6	20	10	10	100	10
				Sum:	13.4

Degrees of Freedom	Probability less than the critical value				
	0.90	**0.95**	**0.975**	**0.99**	**0.999**
5	9.236	11.070	12.833	15.086	20.515

2.5.2 Problems with Pearson's Chi-Squared Test [1]

- The degrees of freedom can only be estimated for *linear models*. It is non-trivial or near impossible to estimate the degrees of freedom for a *non-linear model*.
- The value of chi-squared itself is subject to noise in the data, as such the value is uncertain.

Knowing the degrees of freedom of the model in question is required for chi-squared test. For N data points and P parameters, a naive guess is that the number of degrees of freedom is $N - P$. This, however, as will be demonstrated is not always the case.

The chi-squared test statistic, χ^2, for continuous data is defined as,

$$\chi^2 = \sum_{n=1}^{N} \left(\frac{(y_n - f(x_n, \theta))}{\sigma_n} \right)^2 \tag{2.7}$$

which is equivalent to maximizing the liklihood function. For a linear model, the chi-squared statistic, χ^2 can be written in matrix form as,

$$\chi^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T \boldsymbol{\Sigma}^{-1} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}). \tag{2.8}$$

Rearranging in terms of the model parameters, $\boldsymbol{\theta}$, gives,

$$\boldsymbol{\theta} = (\mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{\Sigma}^{-1} \mathbf{y}. \tag{2.9}$$

Finally, the prediction, $\hat{\mathbf{y}}$, of the measurements, $\mathbf{y}$, can be obtained by multiplying the design matrix, $\mathbf{X}$, with the model parameters, $\boldsymbol{\theta}$, and further simplified to give us,

$$\begin{aligned}
\hat{\mathbf{y}} &= \mathbf{X}\boldsymbol{\theta} && (2.10)\\
&= \mathbf{X}(\mathbf{X}^T\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{\Sigma}^{-1}\mathbf{y} && (2.11)\\
&= \mathbf{H}\mathbf{y} && (2.12)
\end{aligned}$$

where $\mathbf{H}$ is an $N \times N$ matrix, sometimes called the "hat matrix" because it translates the measurement data, $\mathbf{y}$, into a model prediction, $\hat{\mathbf{y}}$. The number of *effecitve* model parameters, P_{eff}, is then given by the trace of $\mathbf{H}$,

$$P_{\text{eff}} = \text{tr}(\mathbf{H}) = \sum_{n=1}^{N} H_{nn} = \text{rank}(\mathbf{X}). \qquad (2.13)$$

which also equals the rank of the design matrix $\mathbf{X}$. And so, $P_{\text{eff}} \leq P$, where the equality holds if and only if the design matrix $\mathbf{X}$ has full rank. Consequently, for linear models the number of degrees of freedom is,

$$K = N - P_{\text{eff}} \geq N - P \qquad (2.14)$$

For nonlinear models, the degrees of freedom is not as straight forward.

Example 1

Let us consider a nonlinear model with three free parameters, A, B, and C,

$$f(x) = A\cos(Bx + C). \qquad (2.15)$$

If we are given a set of N measurement (x_n, y_n, σ_n) such that no two data points have identical x_n, then the model $f(x)$ is capable of fitting any such data set perfectly. The way this works is by increasing the "frequency" B such that $f(x)$ can change on arbitrarily short scales. As $f(x)$ provides a perfect fit in this case, χ^2 is equal to zero for all possible noise realizations of the data. Evidently, this three-parameter model has infinite flexibility (if there are no priors) and $K = N - P$ is a poor estimate of the number of degrees of freedom, which actually is $K = 0$.

Example 2

To build upon the first example, three additional model parameters, D, E and F are added,

$$f(x) = A\cos(Bx + C) + D\cos(Ex + F). \qquad (2.16)$$

If the fit parameter D becomes small such that $|D| \leq |A|$, the second component cannot influence the fit anymore and the two model parameters E and F are "lost". In simple words: This model may change its flexibility during the fitting procedure.

Hence, for nonlinear models, K may not even be constant. Of course, these two examples do not verify the claim that always $K \neq N - P$ for nonlinear models. However, acting as counter-examples, they clearly falsify the claim that $K = N - P$ is always true for nonlinear models.

Chapter 3

Linear Algebra

3.1 Trace

$$\mathrm{tr}(\mathbf{A}) = \sum_i \mathbf{A}_{ii} \tag{3.1}$$

3.2 Rank

The rank $\rho(\mathbf{A})$ of a matrix $\mathbf{A}$ of n rows and m columns is defined as **the number of independent rows or columns**.

- Full rank (non-singular): $\rho(\mathbf{A}) = \min(n, m)$
- Not full rank (singular): $\rho(\mathbf{A}) < \min(n, m)$

3.3 Condition Number

There are different condition numbers in the following the condition number for the problem $\mathbf{Ax} = \mathbf{b}$ and matrix inversion are discussed. In general, the condition number, $\kappa(\cdot)$, for a matrix, $\mathbf{A}$, or computational task such as $\mathbf{Ax} = \mathbf{b}$ measures how sensitive the output is to perturbations in the input data and to round off errors. If the condition number is large, even a small error in $\mathbf{x}$ would cause a large error in $\mathbf{x}$. On the other hand, if the condition number is small then the error in $\mathbf{x}$ will not be much bigger than the error in $\mathbf{b}$.

$$\kappa(\mathbf{A}) \approx 1 \quad \text{well-Conditioned} \tag{3.2}$$

$$\kappa(\mathbf{A}) > 1 \quad \text{ill-Conditioned} \tag{3.3}$$

The condition number is defined more precisely to be the maximum ratio of the relative error in $\mathbf{x}$ to the relative error in $\mathbf{b}$.

Let $\mathbf{e}$ be the error in $\mathbf{b}$. Assuming that $\mathbf{A}$ is a nonsingular matrix, the error in the solution $\mathbf{A}^{-1}\mathbf{b}$ is $\mathbf{A}^{-1}\mathbf{e}$. The ratio of the relative error in the solution to the relative error in $\mathbf{b}$ is

$$\frac{\|\mathbf{A}^{-1}\mathbf{e}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} \bigg/ \frac{\|\mathbf{e}\|}{\|\mathbf{b}\|} \tag{3.4}$$

which can be rewritten as,

$$\left(\frac{\|\mathbf{A}^{-1}\mathbf{e}\|}{\|\mathbf{e}\|}\right) \cdot \left(\frac{\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|}\right) \tag{3.5}$$

The maximum value (for nonzero $\mathbf{b}$ and $\mathbf{e}$) is then seen to be the product of the two operator norms as follows:

$$\max_{\mathbf{e},\mathbf{b}\neq 0}\left\{\left(\frac{\|\mathbf{A}^{-1}\mathbf{e}\|}{\|\mathbf{e}\|} \cdot \frac{\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|}\right)\right\} \tag{3.6}$$

$$= \max_{\mathbf{e},\mathbf{b}\neq 0}\left\{\left(\frac{\|\mathbf{A}^{-1}\mathbf{e}\|}{\|\mathbf{e}\|}\right)\right\} \cdot \max_{\mathbf{e},\mathbf{b}\neq 0}\left\{\left(\frac{\|\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|}\right)\right\} \tag{3.7}$$

$$= \max_{\mathbf{e},\mathbf{b}\neq 0}\left\{\left(\frac{\|\mathbf{A}^{-1}\mathbf{e}\|}{\|\mathbf{e}\|}\right)\right\} \cdot \max_{\mathbf{e},\mathbf{b}\neq 0}\left\{\left(\frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}\right)\right\} \tag{3.8}$$

$$= \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\| \tag{3.9}$$

The same definition is used for any matrix norm, i.e. one that satisfies

$$\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\| \geq \|\mathbf{A}^{-1} \cdot \mathbf{A}\| = 1. \tag{3.10}$$

When the condition number is exactly one (which can only happen if $\mathbf{A}$ is a scalar multiple of a linear isometry), then a solution algorithm can find (in principle, meaning if the algorithm introduces no errors of its own) an approximation of the solution whose precision is no worse than that of the data.

However, it does not mean that the algorithm will converge rapidly to this solution, just that it won't diverge arbitrarily because of inaccuracy on the source data (backward error), provided that the forward error introduced by the algorithm does not diverge as well because of accumulating intermediate rounding errors.

The condition number may also be infinite, but this implies that the problem is ill-posed (does not possess a unique, well-defined solution for each choice of data – that is, the matrix is not invertible), and no algorithm can be expected to reliably find a solution.

The definition of the condition number depends on the choice of norm, as can be illustrated by two examples.

Chapter 4

Probability Theory

4.1 Bayes Theorem

$$p(\mathbf{x},\mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) \tag{4.1}$$

$$\underbrace{p(\mathbf{x}|\mathbf{y})}_{\text{Posterior}} = \frac{\overbrace{p(\mathbf{x}|\mathbf{y})}^{\text{Likelihood}} \quad \overbrace{p(\mathbf{y})}^{\text{Prior}}}{\underbrace{p(\mathbf{y})}_{\text{Evidence}}} \tag{4.2}$$

Chapter 5

Computer Vision

5.1 Fundamentals

5.1.1 Point on Line

$$\mathbf{x}^T\mathbf{l} = 0 \tag{5.1}$$

5.1.2 Intersection of Lines

$$\mathbf{l}(\mathbf{l} \times \mathbf{l}')\mathbf{l}'(\mathbf{l} \times \mathbf{l}') = 0\mathbf{l}^T\mathbf{x} = \mathbf{l}^{T'}\mathbf{x} = 0 \tag{5.2}$$

$$x = \mathbf{l} \times \mathbf{l}' \tag{5.3}$$

5.1.3 Plane

- A plane can be defined by the join between three points, or the join between a line and a point in general
- Two planes intersecting a unique line
- Three planes intersecting a unique point

Three Points Define a Plane

Suppose you have three points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, and are incident with a plane, π then each point satisfies

$$\pi^T \mathbf{X}_i = 0. \tag{5.4}$$

By stacking each point as a matrix

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \pi = 0 \tag{5.5}$$

Since three points in general rare linearly independent, it follows that the $3x4$ matrix compsed of the points $\mathbf{X}_i$ as rows has rank 3.

5.1.4 Fundamental Matrix

5.1.5 Essential Matrix

5.2 Pinhole Camera Model

The pinhole camera model describes how 3D scene points are projected onto the 2D image plane of an ideal pinhole camera. The model makes the assumption that light rays emitted from an object in the scene pass through the pinhole of the camera, and projected onto the image plane. A 3D point $\mathbf{X}_C = (X, Y, Z)$ expressed in the camera frame, $\mathcal{F}_C$, projected on to a camera's 2D image plane in homogeneous coordinates $\mathbf{x}_C = (u, v, 1)$ can be written as

$$u = \frac{X f_x}{Z} \quad v = \frac{Y f_y}{Z} \tag{5.6}$$

where f_x and f_y denote the focal length in the x and y-axis. Or, in matrix form

$$\mathbf{x}_C = \mathbf{K}\mathbf{X}_C \tag{5.7}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_x & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{5.8}$$

where $\mathbf{K}$ represents the intrinsic matrix, c_x and c_y represents the principal point offset in the x and y direction.

In practice, the pinhole camera model only serves as an approximation to modern cameras. The assumptions made in the model are often violated with factors such as large camera apertures (pinhole size), distortion effects in camera lenses, and other factors. That is why the pinhole camera model is often used in combination with a distortion model in the hope of minimizing projection errors from 3D to 2D.

5.3 Radial Tangential Distortion

Lens distortion generally exist in all camera lenses, therefore it is vital we model the distortions observed. The most common distortion model is the radial-tangential (or simply as radtan) distortion model. The two main distortion components, as the name suggests, are the radial and tangential distortion.

The radial distortion occurs due to the shape of the lens. Light passing through the center undergoes no refraction. Light passing through the edges of the lens, on the other hand, undergoes through severe bending causing the radial distortion. The effects of a positive and negative radial distortion can be seen in Fig **??**.

The tangential distortion is due to camera sensor misalignment during the manufacturing process. It occurs when the camera sensor is not in parallel with the lens. The cause of a tangential distortion can be seen in Fig **??**.

The combined radial-tangential distortion is modelled using a polynomial approximation with parameters k_1, k_2 and p_1, p_2 respectively. To apply the distortion the observed 3D point (X, Y, Z) is first projected, distorted, and finally scaled and offset in the image plane (u, v). The radial-tangential distortion model in combination with the pinhole camera model is given in Eq. (**??**) as

$$\begin{aligned}
x &= X/Z \\
y &= Y/Z \\
r^2 &= x^2 + y^2 \\
\\
x' &= x \cdot (1 + (k_1 r^2) + (k_2 r^4)) \\
y' &= y \cdot (1 + (k_1 r^2) + (k_2 r^4)) \\
x'' &= x' + (2p_1 xy + p_2(r^2 + 2x^2)) \\
y'' &= y' + (p_1(r^2 + 2y^2) + 2p_2 xy)
\end{aligned} \tag{5.9}$$

5.3.1 Radial Tangential Point Jacobian

$$\frac{\partial \mathbf{d}_{\text{radtan}}}{\partial_{\text{C}}\mathbf{p}} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$$

$$\begin{aligned}
r^2 &= x^2 + y^2 \\
\\
J_{11} &= k_1 r^2 + k_2 r^4 + 2p_1 y + 6p_2 x + x(2k_1 x + 4k_2 x r^2) + 1 \\
J_{12} &= 2x p_1 + 2y p_2 + y(2k_1 x + 4k_2 x r^2) \\
J_{21} &= 2x p_1 + 2y p_2 + y(2k_1 x + 4k_2 x r^2) \\
J_{22} &= k_1 r^2 + k_2 r^4 + 6p_1 y + 2p_2 x + y(2k_1 y + 4k_2 y r^2) + 1
\end{aligned} \tag{5.10}$$

5.3.2 Radial Tangential Parameter Jacobian

$$\frac{\partial \mathbf{d}_{\text{radtan}}}{\partial \mathbf{d}_{\text{params}}} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \end{bmatrix}$$

$$\begin{aligned}
r^2 &= x^2 + y^2 \\
\\
J_{11} &= x r^2 \\
J_{12} &= x r^4 \\
J_{13} &= 2xy \\
J_{14} &= 3x^2 + y^2 \\
\\
J_{21} &= y r^2 \\
J_{22} &= y r^4 \\
J_{23} &= x^2 + 3y^2 \\
J_{24} &= 2xy
\end{aligned} \tag{5.11}$$

5.4 Equi-distant Distortion

$$
\begin{aligned}
r &= \sqrt{x^2 + y^2} \\
\theta &= \arctan{(r)} \\
\theta_d &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \\
x' &= (\theta_d / r) \cdot x \\
y' &= (\theta_d / r) \cdot y
\end{aligned}
\tag{5.12}
$$

5.4.1 Equi-distant Point Jacobian

$$
\frac{\partial \mathbf{d}_{\text{equi}}}{\partial_{\text{C}} \mathbf{p}} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}
$$

$$
\begin{aligned}
\theta &= \arctan(r) \\
\theta_d &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \\
\theta_d' &= 1 + 3k_1\theta^2 + 5k_2\theta^4 + 7k_3\theta^6 + 9k_4\theta^8 \\
\theta_r &= 1 / (r^2 + 1) \\
s &= \theta_d / r \\
s_r &= \theta_d' \theta_r / r - \theta_d / r^2 \\
r_x &= 1 / r x \\
r_y &= 1 / r y \\
\\
J_{11} &= s + x s_r r_x \\
J_{12} &= x s_r r_y \\
J_{21} &= y s_r r_x \\
J_{22} &= s + y s_r r_y
\end{aligned}
\tag{5.13}
$$

5.4.2 Equi-distant Parameter Jacobian

$$\frac{\partial \mathbf{d}_{\text{equi}}}{\partial \mathbf{d}_{\text{params}}} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \end{bmatrix}$$

$$\theta = \arctan(r)$$

$$\begin{aligned} J_{11} &= x\theta^3/r \\ J_{12} &= x\theta^5/r \\ J_{13} &= x\theta^7/r \\ J_{14} &= x\theta^9/r \\ \\ J_{21} &= y\theta^3/r \\ J_{22} &= y\theta^5/r \\ J_{23} &= y\theta^7/r \\ J_{24} &= y\theta^9/r \end{aligned} \tag{5.14}$$

5.5 Linear Triangulation

There are various methods for triangulating a 3D point obeserved from at least two camera views. The linear triangulation method [3] is frequently used. This method assumes a pair of homogeneous pixel measurements $\mathbf{z}$ and $\mathbf{z}' \in \mathbb{R}^3$ that observes the same 3D point, $\mathbf{X} \in \mathbb{R}^4$, in homogeneous coordinates from two different camera frames. The homogeneous projection from 3D to 2D with a known camera matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$ for each measurement is given as,

$$\begin{aligned} \mathbf{z} &= \mathbf{P}\mathbf{X} \\ \mathbf{z}' &= \mathbf{P}'\mathbf{X}. \end{aligned} \tag{5.15}$$

These equations can be combined to form a system of equations of the form $\mathbf{A}\mathbf{x} = 0$. To eliminate the homogeneous scale factor we apply a cross product to give three equations for each image point, for example $\mathbf{z} \times (\mathbf{P}\mathbf{X}) = \mathbf{0}$ writing this out gives

$$\begin{aligned} x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) &= 0 \\ y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) &= 0 \\ x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) &= 0 \end{aligned} \tag{5.16}$$

where $\mathbf{p}^{iT}$ is the i^{th} row of $\mathbf{P}$.

From Eq. (5.16), an equation of the form $\mathbf{A}\mathbf{x} = \mathbf{0}$ for each image point can be formed, where $\mathbf{x}$ represents the unknown homogeneous feature location to be estimated, and $\mathbf{A}$ is given as

$$\mathbf{A} = \begin{bmatrix} x(\mathbf{p}^{3T}) - (\mathbf{p}^{1T}) \\ y(\mathbf{p}^{3T}) - (\mathbf{p}^{2T}) \\ x'(\mathbf{p}'^{3T}) - (\mathbf{p}'^{1T}) \\ y'(\mathbf{p}'^{3T}) - (\mathbf{p}'^{2T}) \end{bmatrix} \tag{5.17}$$

giving a total of four equations in four homogeneous unknowns. Solving for $\mathbf{A}$ using SVD allows us to estimate the initial feature location.

In an ideal world, the position of 3D points can be solved as a system of equations using the linear triangulation method. In reality, however, errors are present in the camera poses and pixel measurements. The pixel measurements observing the same 3D point are generally noisy. In addition, the camera models and distortion models used often do not model the camera projection or distortion observed perfectly. Therefore an iterative method can be used to further refine the feature position. This problem is generally formulated as a non-linear least square problem and can be solved by numerical methods, such as the Gauss-Newton algorithm.

5.6 Bundle Adjustment

Let $\mathbf{z} \in \mathbb{R}^2$ be the image measurement and $\mathbf{h}(\cdot) \in \mathbb{R}^2$ be the projection function that produces an image projection $\tilde{\mathbf{z}}$. The reprojection error $\mathbf{e}$ is defined as the euclidean distance between $\mathbf{z}$ and $\tilde{\mathbf{z}}$.

$$\mathbf{e} = \mathbf{z} - \tilde{\mathbf{z}} \tag{5.18}$$

Our aim given image measurement $\mathbf{z}$ is to find the image projection $\tilde{\mathbf{z}}$ that minimizes the reprojection $\mathbf{e}$. The image projection $\tilde{\mathbf{z}}$ in pixels can be represented in homogeneous coordinates with u, v, w as

$$\begin{aligned} {}_{\text{C}}\mathbf{p} &= \mathbf{T}_{\text{CW}} \;\; {}_{\text{W}}\mathbf{p} && (5.19) \\ &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} && (5.20) \end{aligned}$$

$$\tilde{\mathbf{z}} = \begin{bmatrix} x/z \\ y/z \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{5.21}$$

where u, v, w is computed by projecting a landmark position ${}_{\text{W}}\mathbf{p}$ in the world frame to the camera's image plane with projection matrix $\mathbf{P}$. The projection matrix, $\mathbf{P}$, can be decomposed into the camera intrinsics matrix, $\mathbf{K}$, the camera rotation, $\mathbf{C}_{\mathrm{WC}}$, and camera position, ${}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}$, expressed in the world frame. The orientation of the camera in (5.21) is represented using a rotation matrix. To reduce the optimization parameters the rotation matrix can be parameterized by a quaternion by using the following formula,

By parameterzing the rotation matrix with a quaternion, the optimization parameters for the camera's orientation is reduced from 9 to 4.

Our objective is to optimize for the camera rotation $\mathbf{C}_{\mathrm{WC}}$, camera position ${}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}$ and 3D landmark position ${}_{\mathrm{W}}\mathbf{p}$ in order to minimize the cost function,

$$\underset{\mathbf{C}_{\mathrm{WC}},\, {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}},\, {}_{\mathrm{W}}\mathbf{p}}{\operatorname{argmin}} \quad \left\| \mathbf{z} - \mathbf{h}(\mathbf{C}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{p}) \right\|^2. \tag{5.22}$$

The cost function above assumes only a single measurement, if there are N measurements corresponding to N unique landmarks the cost function can be rewritten as a maximum likelihood estimation problem as,

$$\underset{\mathbf{C}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{p}}{\operatorname{argmin}} \sum_{j=1}^{N} \left\| \mathbf{z}_j - \mathbf{h}({\mathbf{C}_{\mathrm{WC}}}^j, {{}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}}^j, {}_{\mathrm{W}}\mathbf{p}) \right\|^2 \tag{5.23}$$

under the assumption that the observed landmark, ${}_{\mathrm{W}}\mathbf{p}$, measured in the image plane, z, are corrupted by a **zero-mean Gaussian noise**.

For the general case of M images taken at different camera poses the cost function can be further extended to,

$$\min_{\mathbf{C}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}, {}_{\mathrm{W}}\mathbf{p}} \sum_{i=1}^{M} \sum_{j=1}^{N} \left\| \mathbf{z}_{i,j} - \mathbf{h}({\mathbf{C}_{\mathrm{WC}}}^i, {{}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}}^i, {}_{\mathrm{W}}\mathbf{p}) \right\|^2 \tag{5.24}$$

The optimization process begins by setting the first image camera pose as world origin, and subsequent $\mathbf{C}_{\mathrm{WC}i}$ and ${}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}i}$ will be relative to the first camera pose.

Jacobians

The Jacobian for the optimization problem for a **single measurement** has the form:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{e}}{\partial \delta \boldsymbol{\alpha}} & \dfrac{\partial \mathbf{e}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}} & \dfrac{\partial \mathbf{e}}{\partial {}_{\mathrm{W}}\mathbf{p}} \end{bmatrix} \tag{5.25}$$

If there are two measurements the Jacobian is stacked with the following pattern:

$$\mathbf{J} = \begin{bmatrix} \text{Pose}_{2\times 7} & \mathbf{0}_{2\times 7} & \text{3D Point}_{2\times 3} \\ \mathbf{0}_{2\times 7} & \text{Pose}_{2\times 7} & \text{3D Point}_{2\times 3} \end{bmatrix} \tag{5.26}$$

$$\mathbf{e} = \mathbf{z} - \mathbf{h}(\mathbf{T}_{\mathrm{WC}}^{-1} \ {}_{\mathrm{W}}\mathbf{p}) \tag{5.27}$$

$$\mathbf{h}(\mathbf{T}_{\mathrm{WC}}^{-1} \ {}_{\mathrm{W}}\mathbf{p}) = \mathbf{k}(\mathbf{p}(\mathbf{T}_{\mathrm{WC}}^{-1} \ {}_{\mathrm{W}}\mathbf{p})) \tag{5.28}$$

$$\begin{aligned} \mathbf{p}_{\mathrm{C}} &= \mathbf{T}_{\mathrm{WC}}^{-1} \ {}_{\mathrm{W}}\mathbf{p} \qquad (5.29) \\ &= \mathbf{C}_{\mathrm{WC}}^{-1}({}_{\mathrm{W}}\mathbf{p} - {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}) \qquad (5.30) \end{aligned}$$

$$\frac{\partial \mathbf{e}}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{k}} \frac{\partial \mathbf{k}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \mathbf{p}_{\mathrm{C}}} \tag{5.31}$$

$$\begin{aligned} \frac{\partial \mathbf{e}}{\partial \mathbf{h}} &= -\mathbf{1}_{2\times2} \qquad (5.32) \\ \frac{\partial \mathbf{h}}{\partial \mathbf{k}} &= \mathbf{1}_{2\times2} \qquad (5.33) \\ \frac{\partial \mathbf{k}}{\partial \mathbf{p}} &= \begin{bmatrix} f_x & 0 \\ 0 & fy \end{bmatrix} \qquad (5.34) \\ \frac{\partial \mathbf{p}}{\partial \mathbf{p}_{\mathrm{C}}} &= \begin{bmatrix} 1/z & 0 & -x/z^2 \\ 0 & 1/z & -y/z^2 \end{bmatrix} \qquad (5.35) \end{aligned}$$

$$\begin{aligned} \frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial \delta \boldsymbol{\alpha}} &= \mathbf{C}_{\mathrm{WC}}^{T} \lfloor {}_{\mathrm{W}}\mathbf{p} - {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}} \ \times \rfloor \qquad (5.36) \\ \frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WC}}} &= -\mathbf{C}_{\mathrm{WC}}^{T} \qquad (5.37) \\ \frac{\partial \mathbf{p}_{\mathrm{C}}}{\partial {}_{\mathrm{W}}\mathbf{p}} &= \mathbf{C}_{\mathrm{WC}}^{T} \qquad (5.38) \end{aligned}$$

5.7 Illumination Invariant Transform

Robust fast AprilTag detection emerged from experience with the standard black and white AprilTag during outdoor experiments, where the detection becomes unreliable in certain lighting conditions. In particular, detection fails when strong shadows cover tag features fully or partially. The cause of failure is due to how the detection process relies on image gradients to detect the edges and lines of the tag in order to extract the relative tag pose. Depending on the time of day or weather conditions, this can have a significant impact on reliable AprilTag detection. This sensitivity to illumination was addressed by using the illumination invariant transform by Maddern et al. (2014) [4].

The illumination invariant transform takes three input channels from the image, and returns a single illumination adjusted channel, I, as follows,

$$I = \log(R_2) - \alpha\log(R_1) - (1-\alpha)\log(R_3) \tag{5.39}$$

where R_1, R_2, R_3 are sensor responses (or image channels) corresponding to peak sensitivities at ordered wavelengths $\lambda_1 < \lambda_2 < \lambda_3$, and α is determined by Eq. (5.40).

$$\begin{aligned} \frac{1}{\lambda_2} &= \frac{\alpha}{\lambda_1} + \frac{(1-\alpha)}{\lambda_3} \\ \alpha &= \frac{\lambda_1(\lambda_2 - \lambda_3)}{\lambda_2(\lambda_1 - \lambda_3)} \end{aligned} \tag{5.40}$$

This transform, however, has a non-intuitive effect on black and white targets, as the three channels tend to be equally over and under exposed in RGB images. As a result, the transform leads to similar values for white and black pixels, eliminating the ability for the AprilTag library to detect edges. To resolve this issue, we designed a new AprilTag so that the single channel image produced by using Eq. (5.39) produces a grey scale like image that is robust to shadows and changes in illumination. Examining Eq. (5.39), it can be observed the resulting pixel intensities are maximized when the camera observes green (R_2) and minimized when viewing a mixture of red and blue, (R_1 and R_3 respectively). The proposed illumination invariant AprilTag shown in Fig. **??** is created by replacing the white and black portions of a typical AprilTag with green and magenta. This modification was tested under various lighting conditions. Fig. **??** shows the tag's appearance after performing the illumination invariant transform, creating a single channel image that replaces the typical single channel, grey scale image that is typically used by the AprilTag library. The images shown in Fig. **??** are taken using a PointGrey Chameleon3 (CM3-U3-28S4C-CS) with a Sony ICX818 image sensor. The corresponding values of $\lambda_1, \lambda_2, \lambda_3$ and α are 480 nm, 510 nm, 640 nm and 0.56 respectively as noted in the sensor data sheets.

Chapter 6

Feature Detection and Correspondance

In this section we explain how feature points are detected and matched between different camera frames. The common feature detection and matching pipeline for localization and mapping algorithms is:

1. Detect regions of interests (image feature) in the image
2. Extract image feature information using descriptors
3. Match extracted descriptors

6.1 FAST

Feature detection in computer vision is a process of gathering scene information and deciding locally whether an image feature exists. The resulting subset of image features in the image domain can in turn be used for localization and mapping algorithms to estimate the camera pose.

For our requirements corners was the chosen image feature. The most widely used corner detector is the FAST feature detector [5]. The advantage of using FAST includes its speed and high detection rate. It operates by inspecting a gray-scale image and applying a Bresenham circle or patch of configurable radius (radius of 3 for a 16 pixel circle in Fig 6.1), where each pixel value on the circle is labeled clockwise. If a set of N contiguous pixels in the circle are all brighter than the intensity of the center candidate pixel p plus a threshold value t, or are all darker compared to p minus a threshold value t, then p is considered a corner.

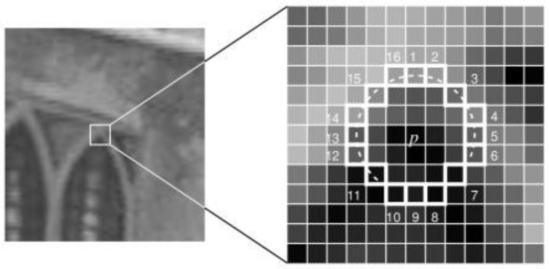
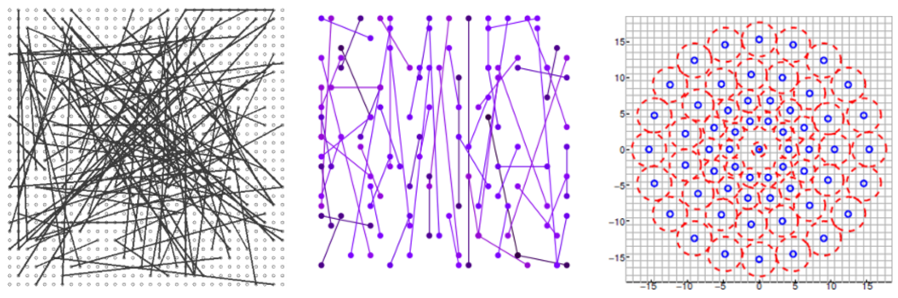


Figure 6.1: FAST Corner Detection [5]

6.2 ORB Feature Descriptor and Matching

To correspond image features detected in two different image frames a feature descriptor is used. Feature descriptors are a way to describe the image feature observed for matching. There are a number of feature descriptors that extract patch information in order to create a robust and repeatable match. Feature descriptors such as SIFT [6], SURF [7], are histogram of gradients (HOG) based patch descriptors. These HOG descriptors are invariant to small rotations and lighting variations, they are however, relatively expensive to compute. The computationally expensive components are its calculation of the image gradient and large descriptor dimension. While both descriptors provide quality information of image features, the aforementioned computational factors impact the matching speed significantly.

Binary descriptors such as BRIEF [8], ORB [9] and BRISK [10] have been proposed to speed up the feature descriptor and matching process. The performance boost in binary descriptors comes in the form of using a binary sampling pattern around each image feature previously detected (see Fig 6.2), and outputting a binary vector, instead of computing image gradients and outputting a floating point vector. Each binary descriptor uses its own unique sampling pattern, and outputs a binary string to be used for matching. The matching process is cheaper compared to the HOG based descriptors, because instead of comparing two floating point vectors, comparing binary descriptors is performed by computing the Hamming distance using a XOR or bit count operation, which can be performed extremely quickly on modern CPUs [11].



(a) BRIEF Descriptor [11] (b) ORB Descriptor [9] (c) BRISK Descriptor [10]

Figure 6.2: Binary Descriptors

6.3 KLT Feature Tracker

In the previous section, we showed how ORB descriptors and matching methods are used to correspond and match features across multiple camera frames. This process of corresponding the the same features across multiple camera frames is called feature tracking. There are many different feature tracking pipelines. Matching feature descriptors such as ORB has shown to have better temporal tracking accuracy compared to KLT-based methods [12]. However, in the work of [13], descriptor-based feature trackers were found to require more computational resources with limited gains in tracking accuracy. Making it less attractive for real-time operation. In the following we will briefly describe the KLT feature tracker, but first an understanding of optical flow is required.

Optical Flow

Optical flow estimates the velocity of each image feature in successive images of a scene. It makes the following explicit assumptions:

- Pixel intensity does not change between consecutive frames
- Displacement of features is small
- Features are within the same local neighbourhood

Let us consider a pixel, p, in the first frame which has an intensity, $I(x, y, t)$, where it is a function of the pixel location, x and y, and time, t. If we apply the aforementioned assumptions, we can say that the intensity of said pixel in the first frame to the second does not change. Additionally, if there was a small displacement, dx and dy, and small time difference, dt, between images this can be written in mathematical form as,

$$I(x, y, t) = I(x + dx, y + dy, t + dt). \tag{6.1}$$

This is known as the brightness constancy equation. To obtain the image gradient and velocity of the pixel, we can use Taylor series approximation of right-hand side of Eq. (6.1) to get,

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \text{H.O.T}, \tag{6.2}$$

removing common terms and dividing by dt we get,

$$I_x v_x + I_y v_y + I_t = 0 \tag{6.3}$$

or,

$$I_x v_x + I_y v_y = -I_t \tag{6.4}$$

where:

$$I_x = \frac{\partial I}{\partial x}; \quad I_y = \frac{\partial I}{\partial y}$$
$$v_x = \frac{dx}{dt}; \quad v_y = \frac{dy}{dt}.$$

The image gradients along the x and y directions are I_x and I_y, where I_t is the image gradient along time, finally, v_x and v_y are the pixel velocity in x and y directions, which is unknown. The problem with Eq. 6.4 is that it provides a single constraint with two degrees of freedom, and as such requires at least one additional constraint to identify a solution.

The Lucas-Kanade method solves the aperture problem by introducing additional conditions. This method assumes all pixels within a window centered around a pixel p will have similar motion, and that the window size is configurable. For example, a window size of 3×3 around the pixel p, the 9 points within the window should have a similar motion. Using Eq. 6.4, the intensity inside the window must therefore satisfy,

$$\begin{aligned}
I_x(p_1)v_x(p_1) + I_y(p_1)v_y &= -I_t(p_1) \\
I_x(p_1)v_x(p_2) + I_y(p_2)v_y &= -I_t(p_2) \\
&\vdots \\
I_x(p_1)v_x(p_n) + I_y(p_n)v_y &= -I_t(p_n)
\end{aligned}$$

where $p_1, p_2, \ldots, p_n$ are the pixels in the window. This can be re-written in matrix form $\mathbf{Ax} = \mathbf{b}$ as,

$$\mathbf{A} = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix}. \tag{6.5}$$

The linear system of equations of Eq. 6.5 is over-determined, therefore there is no exact solution. To address this issue, a least squares method can be used to approximate the solution by applying the ordinary least squares. For the system $\mathbf{Ax} = \mathbf{b}$, the least squares formula is obtained by minimizing the following,

$$\underset{\mathbf{x}}{\operatorname{argmin}} \ ||\mathbf{Ax} - \mathbf{b}||, \tag{6.6}$$

the solution of which can be obtained by using *normal equations*,

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b} \tag{6.7}$$

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{6.8}$$

Rewriting Eq 6.5 in the form of Eq. 6.8 we get,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_i)^2 & \sum_i I_x(p_i)I_y(p_i) \\ \sum_i I_x(p_i)I_y(p_i) & \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_i)I_t(p_i) \\ -\sum_i I_y(p_i)I_t(p_i) \end{bmatrix} \tag{6.9}$$

which is finally used to obtain the optical flow of pixel p.

KLT Feature Tracker

The Lucas-Kanade method recovers feature pixel velocities from consecutive camera frames. The issue with the Lucas-Kanade method is that it assumes the features detected have small displacements between consecutive camera frames. Therefore to track features that have a large motion the Kanade-Lucas-Tomasi (KLT) feature tracker [**?**] uses reduced-scale versions of the input images in order to track features over multiple camera frames. The general steps of the KLT feature tracker is as follows,

1. Detect corners in the first camera frame
2. For each corners, compute the motion between consecutive camera frames using a pyramidal implementation of the Lucas-Kanade method
3. Match motion vectors between consecutive camera frames to track corners
4. Detect new corners if number of tracks currently tracking is too low
5. Repeat steps 2 to 4

Chapter 7

Rotations

7.1 Euler Angles

Z-Y-X rotation sequence:

$$\mathbf{C}_{zyx} = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) \\ s(\psi)c(\theta) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{7.1}$$

7.2 Quaternions

A quaternion, $\mathbf{q} \in \mathbb{R}^4$, generally has the following form

$$\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}, \tag{7.2}$$

where $\{q_w, q_x, q_y, q_z\} \in \mathbb{R}$ and $\{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ are the imaginary numbers satisfying

$$\begin{gathered} \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \\ \mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \;\; \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \;\; \mathbf{ki} = -\mathbf{ik} = \mathbf{j} \end{gathered} \tag{7.3}$$

corresponding to the Hamiltonian convention. The quaternion can be written as a 4 element vector consisting of a *real* (*scalar*) part, q_w, and *imaginary* (*vector*) part $\mathbf{q}_v$ as,

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{7.4}$$

There are other quaternion conventions, for example, the JPL convention. A more detailed discussion between Hamiltonian and JPL quaternion convention is discussed in [14].

7.2.1 Main Quaternion Properties

Sum

Let $\mathbf{p}$ and $\mathbf{q}$ be two quaternions, the sum of both quaternions is,

$$\mathbf{p} \pm \mathbf{q} = \begin{bmatrix} p_w \\ \mathbf{p}_v \end{bmatrix} \pm \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} p_w \pm q_w \\ \mathbf{p}_v \pm \mathbf{q}_v \end{bmatrix} . \tag{7.5}$$

The sum between two quaternions $\mathbf{p}$ and $\mathbf{q}$ is **commutative** and **associative**.

$$\mathbf{p} + \mathbf{q} = \mathbf{q} + \mathbf{p} \tag{7.6}$$

$$\mathbf{p} + (\mathbf{q} + \mathbf{r}) = (\mathbf{p} + \mathbf{q}) + \mathbf{r} \tag{7.7}$$

Product

The quaternion multiplication (or product) of two quaternions $\mathbf{p}$ and $\mathbf{q}$, denoted by $\otimes$ is defined as

$$\begin{aligned}
\mathbf{p} \otimes \mathbf{q} &= (p_w + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k})(q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}) && (7.8) \\
&= \begin{matrix} (p_w q_w - p_x q_x - p_y q_y - p_z q_z) & \\ (p_w q_x + p_x q_w + p_y q_z - p_z q_y) & \mathbf{i} \\ (p_w q_y - p_y q_w + p_z q_x + p_x q_z) & \mathbf{j} \\ (p_w q_z + p_z q_w - p_x q_y + p_y q_x) & \mathbf{k} \end{matrix} && (7.9) \\
&= \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + q_x p_w + p_y q_z - p_z q_y \\ p_w q_y - p_y q_w + p_z q_x + p_x q_z \\ p_w q_z + p_z q_w - p_x q_y + p_y q_x \end{bmatrix} && (7.10) \\
&= \begin{bmatrix} p_w q_w - \mathbf{p}_v^T \mathbf{q}_v \\ p_w \mathbf{q}_v + q_w \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{bmatrix} . && (7.11)
\end{aligned}$$

The quaternion product is **not commutative** in the general case[1],

$$\mathbf{p} \otimes \mathbf{q} \neq \mathbf{q} \otimes \mathbf{p} \ . \tag{7.12}$$

[1]There are exceptions to the general non-commutative rule, where either $\mathbf{p}$ or $\mathbf{q}$ is real such that $\mathbf{p}_v \times \mathbf{q}_v = 0$, or when both $\mathbf{p}_v$ and $\mathbf{q}_v$ are parallel, $\mathbf{p}_v || \mathbf{q}_v$. Only in these cirmcumstances is the quaternion product commutative.

The quaternion product is however **associative**,

$$\mathbf{p} \otimes (\mathbf{q} \otimes \mathbf{r}) = (\mathbf{p} \otimes \mathbf{q}) \otimes \mathbf{r} \tag{7.13}$$

and **distributive over the sum**

$$\mathbf{p} \otimes (\mathbf{q} + \mathbf{r}) = \mathbf{p} \otimes \mathbf{q} + \mathbf{p} \otimes \mathbf{r} \quad \text{and} \quad (\mathbf{p} \otimes \mathbf{q}) + \mathbf{r} = \mathbf{p} \otimes \mathbf{r} + \mathbf{q} \otimes \mathbf{r} \tag{7.14}$$

The quaternion product can alternatively be expressed in matrix form as

$$\mathbf{p} \otimes \mathbf{q} = [\mathbf{p}]_L \mathbf{q} \quad \text{and} \quad \mathbf{p} \otimes \mathbf{q} = [\mathbf{q}]_R \mathbf{p} \ , \tag{7.15}$$

where $[\mathbf{p}]_L$ and $[\mathbf{q}]_R$ are the left and right quaternion-product matrices which are derived from (7.10),

$$[\mathbf{p}]_L = \begin{bmatrix} p_w & -p_x & -p_y & -p_z \\ p_x & p_w & -p_z & p_y \\ p_y & p_z & p_w & -p_x \\ p_z & -p_y & p_x & p_w \end{bmatrix}, \quad \text{and} \quad [\mathbf{q}]_R = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & q_z & -q_y \\ q_y & -q_z & q_w & q_x \\ q_z & q_y & -q_x & q_w \end{bmatrix}, \tag{7.16}$$

or inspecting (7.11) a compact form can be derived as,

$$[\mathbf{p}]_L = \begin{bmatrix} 0 & -\mathbf{p}_v^T \\ \mathbf{p}_w \mathbf{I}_{3\times 3} + \mathbf{p}_v & \mathbf{p}_w \mathbf{I}_{3\times 3} - \lfloor \mathbf{p}_v \ \times \rfloor \end{bmatrix} \tag{7.17}$$

and

$$[\mathbf{q}]_R = \begin{bmatrix} 0 & -\mathbf{q}_v^T \\ \mathbf{q}_w \mathbf{I}_{3\times 3} + \mathbf{q}_v & \mathbf{q}_w \mathbf{I}_{3\times 3} - \lfloor \mathbf{q}_v \ \times \rfloor \end{bmatrix}, \tag{7.18}$$

where $\lfloor \bullet \ \times \rfloor$ is the skew operator that produces a matrix cross product matrix, and is defined as

$$\lfloor \mathbf{v} \ \times \rfloor = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \quad \mathbf{v} \in \mathbb{R}^3 \tag{7.19}$$

Conjugate

The conjugate operator for quaternion, $(\bullet)^*$, is defined as

$$\mathbf{q}^* = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{bmatrix}. \tag{7.20}$$

This has the properties

$$\mathbf{q} \otimes \mathbf{q}^{-1} = \mathbf{q}^{-1} \otimes \mathbf{q} = q_w^2 + q_x^2 + q_y^2 + q_z^2 = \begin{bmatrix} q_w^2 + q_x^2 + q_y^2 + q_z^2 \\ \mathbf{0} \end{bmatrix}, \tag{7.21}$$

and

$$(\mathbf{p} \otimes \mathbf{q})^* = \mathbf{q}^* \otimes \mathbf{p}^*. \tag{7.22}$$

Norm

The norm of a quaternion is defined by

$$\begin{aligned} \|\mathbf{q}\| &= \sqrt{\mathbf{q} \otimes \mathbf{q}^*} && (7.23) \\ &= \sqrt{\mathbf{q}^* \otimes \mathbf{q}} && (7.24) \\ &= \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \quad \in \mathbb{R}, && (7.25) \end{aligned}$$

and has the property

$$\|\mathbf{p} \otimes \mathbf{q}\| = \|\mathbf{q} \otimes \mathbf{p}\| = \|\mathbf{p}\|\|\mathbf{q}\| \tag{7.26}$$

7.2.2 Quaternion from Two Vectors

Using the properties of the cross and dot product

$$\begin{aligned} \mathbf{u} \cdot \mathbf{v} &= \|\mathbf{u}\|\|\mathbf{v}\| \cos\theta && (7.27) \\ \|\mathbf{u} \times \mathbf{v}\| &= \|\mathbf{u}\|\|\mathbf{v}\|\|\sin\theta\|, && (7.28) \end{aligned}$$

the axis angle, $\boldsymbol{\theta} \in \mathbb{R}^3$, can be obtained from $\mathbf{u}$ and $\mathbf{v}$ with

$$\begin{aligned} \boldsymbol{\theta} &= \theta\mathbf{e} && (7.29) \\ \theta &= \cos^{-1}\left(\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right) \quad , \quad \theta \in \mathbb{R} && (7.30) \\ \mathbf{e} &= \frac{\mathbf{u} \times \mathbf{v}}{\|\mathbf{u} \times \mathbf{v}\|} \quad , \quad \mathbf{e} \in \mathbb{R}^3 && (7.31) \end{aligned}$$

where $\mathbf{e}$ is the unit vector that defines the rotation axis and θ is the rotation angle about $\mathbf{e}$. Once the axis angle, $\boldsymbol{\theta}$, is obtained a quaternion can be formed

$$\mathbf{q} = \cos\frac{\theta}{2} + \mathbf{i}\sin\frac{\theta}{2}e_x + \mathbf{j}\sin\frac{\theta}{2}e_y + \mathbf{k}\sin\frac{\theta}{2}e_z \quad . \tag{7.32}$$

Example: Attitude from gravity and accelerometer vectors

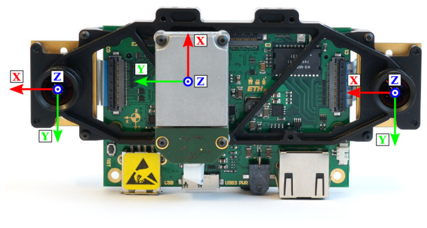
In robotics knowing the attitude of the system is often required. An Inertial Measurement Unit (IMU) is commonly used to obtain this information. Using the method described previously, a gravity vector along with an accelerometer measurement vector can be used to obtain an attitude in form of a quaternion.

Let $\mathbf{g} \in \mathbb{R}^3$ be the gravity vector, and $\mathbf{a}_m \in \mathbb{R}^3$ be the accelerometer measurement from an IMU. With the two vectors $\mathbf{g}$ and $\mathbf{a}_m$ a quaternion $\mathbf{q}_{\text{WS}}$ expressing the rotation of the IMU sensor frame, $\mathcal{F}_{\text{S}}$, with respect to the world frame, $\mathcal{F}_{\mathrm{W}}$, can be calculated given that values for $\mathbf{g}$ and $\mathbf{a}_m$ are known. For example let

$$\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T \tag{7.33}$$

$$\mathbf{a}_m = \begin{bmatrix} 9.2681 & -0.310816 & -3.14984 \end{bmatrix}^T, \tag{7.34}$$

taken from the first measurement of the `imu_april` calibration sequence of the EuRoC MAV dataset.



(a) VI-Sensor Coordinate Frames

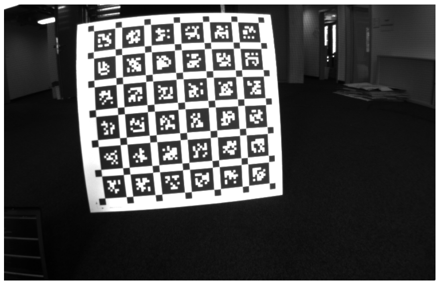

(b) Image frame

Figure 7.1: EuRoC MAV Dataset - `imu_april` Sequence

Before calculating the axis-angle, however, it should be noted that when an accelerometer is at rest the measurement reading in the z-axis is positive instead of negative. The reason is accelerometers measures acceleration by measuring the displacement of a proof mass that is suspended with springs. For example, if gravity is ignored and the accelerometer moves upwards, the proof mass will be displaced towards the bottom of the accelerometer. This is interpreted as an acceleration in the upwards direction, and so when the accelerometer is at rest on a flat surface, gravity pulls on the proof mass yeilding a positive measurement in the upwards direction. To resolve this issue the gravity vector is negated, and so $\mathbf{u} = -\mathbf{g}$ and $\mathbf{v} = \mathbf{a}_m$. Using (7.30) and (7.31) the axis-angle, $\boldsymbol{\theta}$, is thereby

$$\theta = 1.8982 \tag{7.35}$$

$$\mathbf{e} = \begin{bmatrix} 0.03352 & 0.99944 & 0.00000 \end{bmatrix}^T \tag{7.36}$$

Finally the quaternion, $\mathbf{q}_{\mathrm{WS}}$, can be calculated using (7.32) resulting in

$$\mathbf{q}_{\mathrm{WS}} = \begin{bmatrix} 0.58240 & 0.02725 & 0.81245 & 0.00000 \end{bmatrix}^T . \tag{7.37}$$

7.2.3 Quaternion to Rotation Matrix

$$\mathbf{R}\{\mathbf{q}\} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_y - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{7.38}$$

7.2.4 Rotation Matrix to Quaternion

$$q_w = \frac{\sqrt{1 + m_{11} + m_{22} + m_{33}}}{2} \tag{7.39}$$

$$q_x = \frac{m_{32} - m_{23}}{4q_w} \tag{7.40}$$

$$q_y = \frac{m_{13} - m_{31}}{4q_w} \tag{7.41}$$

$$q_z = \frac{m_{21} - m_{02}}{4q_w} \tag{7.42}$$

Note, while the equations seems straight forward in practice, however,the trace of the rotation matrix need to be checked inorder to guarantee correctness.

7.3 Differential Calculus

Lie Group $SO(3)$

- Not a vector space
- Has no addition operator
- Has no subtraction operator

State estimation frameworks rely on small differences and gradients in order to correct the state estimate. Orientations unlike translation and velocity do not have an addition operator, as such it is more involving to update or find the gradients of orientations. Forunately, since orientations are a special orhogonal group $SO(3)$ as well as a Lie group, an exponential map exists that relates to its Lie algebra allowing orientations to be perturbed and its gradients calculated.

Elements in Lie algebra are abstract vectors and not suitable for actual computations. A basis $\mathbf{B} = [\vec{\boldsymbol{\varphi}}_1 \;\; \vec{\boldsymbol{\varphi}}_2 \;\; \vec{\boldsymbol{\varphi}}_3]$ can be used to extend the map to $\mathbb{R}^3$.

The definition of an exponential map $\exp : \mathbb{R}^3 \mapsto SO(3)$ of a coordinate tuple $\boldsymbol{\varphi}(\varphi_1, \varphi_2, \varphi_3) \in \mathbb{R}^3$ is defined by

$$\exp(\boldsymbol{\varphi}) := Exp(\vec{\boldsymbol{\varphi}}_1 \varphi_1, \vec{\boldsymbol{\varphi}}_2 \varphi_2, \vec{\boldsymbol{\varphi}}_3 \varphi_3) \tag{7.43}$$

$\forall t, s \in \mathbb{R}$, $\forall \vec{\boldsymbol{\varphi}} \in$

$$Exp((t+s)\vec{\boldsymbol{\varphi}}) = Exp(t\vec{\boldsymbol{\varphi}}) \circ Exp(s\vec{\boldsymbol{\varphi}}) \tag{7.44}$$

$$\begin{aligned}
\boxplus :& SO(3) \times \mathbb{R}^3 \to SO(3), && (7.45)\\
& \Phi, \boldsymbol{\varphi} \mapsto \exp(\boldsymbol{\varphi}) \circ \Phi, \\
\boxminus :& SO(3) \times SO(3) \to \mathbb{R}^3, && (7.46)\\
& \Phi_1, \Phi_2 \mapsto \log(\Phi_1 \circ \Phi_2^{-1})
\end{aligned}$$

Similar to regular addition and subtraction, both operators have the following identities,

$$\Phi \boxplus \mathbf{0} = \Phi \tag{7.47}$$

$$(\Phi \boxplus \boldsymbol{\varphi}) \boxminus \Phi = \boldsymbol{\varphi} \tag{7.48}$$

$$\Phi_1 \boxplus (\Phi_2 \boxminus \Phi_1) = \Phi_2 \tag{7.49}$$

Chapter 8

Optimization

8.1 Linear Least Squares

Linear problems generally have the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{8.1}$$

If $\mathbf{A}$ is skinny (number of rows is larger than number of columns) the problem is over constrained and there is no *unique* solution. Instead, the problem can be solved by minizming the squared error between $\mathbf{A}\mathbf{x}$ and $\mathbf{b}$. The linear least squares problem is then defined as,

$$\min_{\mathbf{x}} ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 \ , \tag{8.2}$$

where the goal is to find an *approximate* solution.

The local minima can be found when the derivative of the squared error is zero. First the squared error is expanded to give:

$$(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{8.3}$$

$$(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b}) \tag{8.4}$$

then by differentiating the expanded squared error with respect to $\mathbf{x}$, setting the derivative to zero, and rearranging the equation with respect to $\mathbf{x}$ gives the following

$$2\mathbf{x}^T\mathbf{A}^T\mathbf{A} - 2\mathbf{b}^T\mathbf{A} = 0 \tag{8.5}$$
$$\mathbf{x}^T\mathbf{A}^T\mathbf{A} = \mathbf{b}^T\mathbf{A} \tag{8.6}$$
$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{8.7}$$
$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \tag{8.8}$$
$$\mathbf{x} = \mathbf{A}^\dagger\mathbf{b}\ , \tag{8.9}$$

where $\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is known as the pseudo inverse $\mathbf{A}^\dagger$.

8.2 Non-linear Least Squares

A non-linear least squares is an optimization problem that minimizes the sum of squares of the residual functions, and has the form

$$\min_{\mathbf{x}} \ \frac{1}{2}\mathbf{e}(\mathbf{x})^T\mathbf{W}\,\mathbf{e}(\mathbf{x}) \tag{8.10}$$

where the error function, $\mathbf{e}(\cdot)$, depends on the optimization parameter, $\mathbf{x} \in \mathbb{R}^n$. The error function, $\mathbf{e}(\cdot)$, has a form of

$$\mathbf{e}_i = \mathbf{z} - \mathbf{h}(\mathbf{x}) \tag{8.11}$$

is defined as the difference between the measured value, $\mathbf{z}$, and the estimated value calculated using the measurement function, $\mathbf{h}(\cdot)$.

A local minima for the problem is found when the gradient of the cost, $\mathbf{C}$, is zero

$$\frac{\partial\mathbf{C}}{\partial\mathbf{x}} = \frac{\partial\mathbf{C}}{\partial\mathbf{e}}\frac{\partial\mathbf{e}}{\partial\mathbf{x}} \tag{8.12}$$
$$= \mathbf{e}(\mathbf{x})^T\mathbf{W}\frac{\partial\mathbf{e}}{\partial\mathbf{x}} \tag{8.13}$$
$$= \mathbf{e}(\mathbf{x})^T\mathbf{W}\mathbf{E}(\mathbf{x}) \tag{8.14}$$

linearizing $\mathbf{e}(\mathbf{x})$ with the first-order Taylor series, $\mathbf{e}(\mathbf{x}) \approx \mathbf{e}(\bar{\mathbf{x}}) + \mathbf{E}(\bar{\mathbf{x}})\Delta\mathbf{x}$, gives,

$$\frac{\partial\mathbf{C}}{\partial\mathbf{x}} = (\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{E}(\bar{\mathbf{x}})\Delta\mathbf{x})^T\mathbf{W}\mathbf{E}(\mathbf{x}) = 0 \tag{8.15}$$
$$\mathbf{e}(\bar{\mathbf{x}})^T\mathbf{W}\mathbf{E}(\bar{\mathbf{x}}) + \Delta\mathbf{x}^T\mathbf{E}(\bar{\mathbf{x}})^T\mathbf{W}\mathbf{E}(\bar{\mathbf{x}}) = 0 \tag{8.16}$$
$$\mathbf{E}(\bar{\mathbf{x}})^T\mathbf{W}\,\mathbf{e}(\bar{\mathbf{x}}) + \mathbf{E}(\bar{\mathbf{x}})^T\mathbf{W}\mathbf{E}(\bar{\mathbf{x}})\Delta\mathbf{x} = 0 \tag{8.17}$$
$$\underbrace{\mathbf{E}(\bar{\mathbf{x}})^T\mathbf{W}\mathbf{E}(\bar{\mathbf{x}})}_{\mathbf{A}}\ \underbrace{\Delta\mathbf{x}}_{\mathbf{x}} = \underbrace{-\mathbf{E}(\bar{\mathbf{x}})^T\mathbf{W}\,\mathbf{e}(\bar{\mathbf{x}})}_{\mathbf{b}} \tag{8.18}$$

solve the normal equations for $\Delta\mathbf{x}$ and update $\mathbf{x}$ using,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x} \tag{8.19}$$

8.3 Gauge

Gauge theory is borrowed from physics.

Accurate structure from motion or vision based state estimation is hard. One hurdle is addressing the accuracy quantitatively. There are two main problems that arise:

- **Inherent Physical Indeterminancy**: cause by loss of information while projecting 3D objects onto a 2D image plane.
- **Overparameterized Problem**: e.g. a shape model that can be parameterized by a vector, each representing the absolute position and orientation of the object could itself be indeterminant.

It is well known that a vision only bundle adjustment has 7 unobserable degrees-of-freedom (DoF), while for a VI-system, the global position and global yaw is not observable, a total of four unobservable DoFs. These unobservable DoFs (a.k.a gauge freedoms) have to be handled properly.

There are three main approaches to address the unobservability in a VI-system. They are *gauge fixation*, *gauge prior*, *free gauge.*

8.3.1 Gauge fixation

Gauge fixation method works by decreasing the number of optimization parameters to where there are no unobservable states left for the opitmization problem to optimize. This is to ensure the Hessian is well conditioned and invertable. This approach enforces hard constraints to the solution.

The standard method to update orientation variables such as a rotation, $\mathbf{C}$, during the iterations of a non-linear least squares solver is to use local coordinates, where at the k-th iteration, the update is

$$\mathbf{C}^{k+1} = \text{Exp}(\delta\boldsymbol{\phi}^{k})\mathbf{C}^{k}. \tag{8.20}$$

Setting the z component of $\boldsymbol{\phi}^{k}$ to 0 allows fixating the yaw with respect to $\mathbf{C}^{k}$. However, concatenating several such updates over K-iterations,

$$\mathbf{C}^{K} = \prod_{k=0}^{K-1} \text{Exp}(\delta\boldsymbol{\phi}^{k}), \tag{8.21}$$

does not fixate the yaw with respect to the initial rotation $\mathbf{C}^0$, and therefore, this parameterization cannot be used to fix the yaw-value of $\mathbf{C}^K$ to that of the initial value $\mathbf{C}^0$.

Although pose fixation or prior can be applied to any camera pose, it is common practice to fixate the first camera.

$$\mathbf{r}_0 = \mathbf{r}_0^0,\ \ \Delta\boldsymbol{\phi}_{0z} \doteq \mathbf{e}_z^T \boldsymbol{\phi}_0 = 0\,, \tag{8.22}$$

where $\mathbf{r}_0^0$ is the initial position of the first camera. Which is equivalent to setting the corresponding columns of the Jacobian of the residual vector to zero, namely $\mathbf{J}_{\mathbf{r}_0} = 0$, $\mathbf{J}_{\Delta\phi_{0z}} = 0$. Thus, for rotations of the other camera poses, the standard iterative update Eq. (8.20) is used, and, for the first camera rotation, $\mathbf{C}_0$, a more convenient parameterization is used. Instead of directly using $\mathbf{C}_0$, a left-multiplicative increment is used.

$$\mathbf{C}_0 = \mathrm{Exp}(\Delta\boldsymbol{\phi}_0)\mathbf{C}_0^0\,, \tag{8.23}$$

where the rotation vector $\Delta\boldsymbol{\phi}_0$ is initialized to zero and updated.

8.3.2 Gauge prior

Gauge prior augments the objective function with an additional penalty to favor a solution that satisfies certain constraints in a soft manner.

$$\|\mathbf{e}_0^{\mathbf{r}}\|^2_{\Sigma_0^{\mathbf{r}}}\,, \quad \text{where} \quad \mathbf{e}_0^{\mathbf{r}}(\boldsymbol{\theta}) \doteq (\mathbf{r}_0 - \mathbf{r}_0^0,\ \Delta\phi_{0z}) \tag{8.24}$$

8.3.3 Free gauge

Free gauge is the most general, lets the optimization parameters evolve freely. In order to deal with the singularity with the Hessian, the pseudo inverse is used or some preconditioning method inorder to make the Hessian well-conditioned and invertible.

Chapter 9

Calibration

$$\underset{\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}}}{\operatorname{argmin}} \|\mathbf{r}(\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}})\|^2 \tag{9.1}$$

Where

$$\mathbf{r}(\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}}) = \mathbf{z} - \mathbf{h}(\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}}) \tag{9.2}$$

In ceres-solver the expression $\|\mathbf{r}(\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}})\|^2$ is known as a residual block. As such the optimization jacobian matrix is with respect to the residual.

$$\hat{\mathbf{z}} = \mathbf{h}(\mathbf{T}_{\mathrm{WS}}, \mathbf{T}_{\mathrm{SC}}, \mathbf{T}_{\mathrm{WF}}) \tag{9.3}$$

$$\mathbf{e} = \mathbf{z} - \hat{\mathbf{z}} \tag{9.4}$$

$$\hat{\mathbf{z}} = \begin{bmatrix} {}_{\mathrm{C}}X / {}_{\mathrm{C}}Z \\ {}_{\mathrm{C}}Y / {}_{\mathrm{C}}Z \end{bmatrix} \tag{9.5}$$

$${}_{\mathrm{C}}\mathbf{p} = \begin{bmatrix} {}_{\mathrm{C}}X \\ {}_{\mathrm{C}}Y \\ {}_{\mathrm{C}}Z \end{bmatrix} \tag{9.6}$$

$$\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} = \begin{bmatrix} 1/{}_{\mathrm{C}}Z & 0 & -{}_{\mathrm{C}}X/{}_{\mathrm{C}}Z^2 \\ 0 & 1/{}_{\mathrm{C}}Z & -{}_{\mathrm{C}}Y/{}_{\mathrm{C}}Z^2 \end{bmatrix} \tag{9.7}$$

$${}_{\mathrm{C}}\mathbf{p} = \mathbf{T}_{\mathrm{SC}}{}^{-1} \; \mathbf{T}_{\mathrm{WS}}{}^{-1} \; \mathbf{T}_{\mathrm{WF}} \; {}_{\mathrm{F}}\mathbf{p} \tag{9.8}$$

9.1 Jacobian w.r.t Sensor Pose, $\mathbf{T}_{\mathrm{WS}}$

$$
\begin{aligned}
{}_{\mathrm{W}}\mathbf{p} &= \mathbf{T}_{\mathrm{WS}} \; {}_{\mathrm{S}}\mathbf{p} && (9.9)\\
&= \mathbf{C}_{\mathrm{WS}} \; {}_{\mathrm{S}}\mathbf{p} + {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WS}} && (9.10)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{p}} &= \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} && (9.11)\\
\frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} &= \mathbf{C}_{\mathrm{CW}} && (9.12)
\end{aligned}
$$

$$
\frac{\partial \mathbf{h}}{\partial \mathbf{T}_{\mathrm{WS}}} = \begin{bmatrix} \dfrac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} & \dfrac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WS}}} \end{bmatrix} \qquad (9.13)
$$

$$
\begin{aligned}
\frac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} &= \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} && (9.14)\\
\frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} &= -\lfloor \mathbf{C}\{\delta \boldsymbol{\theta}\} \; {}_{\mathrm{S}}\mathbf{p} \; \times \rfloor && (9.15)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WS}}} &= \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WS}}} && (9.16)\\
\frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WS}}} &= \mathbf{I} && (9.17)
\end{aligned}
$$

9.2 Jacobian w.r.t Sensor-Camera Extrinsics, $\mathbf{T}_{\mathrm{SC}}$

$$
\begin{aligned}
{}_{\mathrm{S}}\mathbf{p} &= \mathbf{T}_{\mathrm{SC}} \; {}_{\mathrm{C}}\mathbf{p} && (9.18)\\
&= \mathbf{C}_{\mathrm{SC}} \; {}_{\mathrm{C}}\mathbf{p} + {}_{\mathrm{S}}\mathbf{r}_{\mathrm{SC}} && (9.19)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{S}}\mathbf{p}} &= \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \; \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{p}} && (9.20)\\
\frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{p}} &= \mathbf{C}_{\mathrm{CS}} && (9.21)
\end{aligned}
$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{T}_{\mathrm{SC}}} = \begin{bmatrix} \frac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} & \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{S}}\mathbf{r}_{\mathrm{SC}}} \end{bmatrix} \tag{9.22}$$

$$\frac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} = \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{p}} \frac{\partial {}_{\mathrm{S}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} \tag{9.23}$$

$$\frac{\partial {}_{\mathrm{S}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} = -\lfloor \mathbf{C}\{\delta \boldsymbol{\theta}\} \; {}_{\mathrm{S}}\mathbf{p} \; \times \rfloor \tag{9.24}$$

$$\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{S}}\mathbf{r}_{\mathrm{SC}}} = \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{p}} \frac{\partial {}_{\mathrm{S}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{r}_{\mathrm{SC}}} \tag{9.25}$$

$$\frac{\partial {}_{\mathrm{S}}\mathbf{p}}{\partial {}_{\mathrm{S}}\mathbf{r}_{\mathrm{SC}}} = \mathbf{I} \tag{9.26}$$

$$\tag{9.27}$$

9.3 Jacobian w.r.t Fiducial Pose, $\mathbf{T}_{\mathbf{WF}}$

$$\begin{aligned} {}_{\mathrm{W}}\mathbf{p} &= \mathbf{T}_{\mathrm{WF}} \; {}_{\mathrm{F}}\mathbf{p} \\ &= \mathbf{C}_{\mathrm{WF}} \; {}_{\mathrm{F}}\mathbf{p} + {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WF}} \end{aligned} \tag{9.28}$$

$$\frac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{p}} = \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{C}}\mathbf{p}} \frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} \tag{9.29}$$

$$\frac{\partial {}_{\mathrm{C}}\mathbf{p}}{\partial {}_{\mathrm{W}}\mathbf{p}} = \mathbf{C}_{\mathrm{CW}} \tag{9.30}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{T}_{\mathrm{WF}}} = \begin{bmatrix} \frac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} & \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{r}_{\mathrm{WF}}} \end{bmatrix} \tag{9.31}$$

$$\frac{\partial \mathbf{h}}{\partial \delta \boldsymbol{\theta}} = \frac{\partial \mathbf{h}}{\partial {}_{\mathrm{W}}\mathbf{p}} \frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} \tag{9.32}$$

$$\frac{\partial {}_{\mathrm{W}}\mathbf{p}}{\partial \delta \boldsymbol{\theta}} = -\lfloor \mathbf{C}\{\delta \boldsymbol{\theta}\} \; {}_{\mathrm{F}}\mathbf{p} \; \times \rfloor \tag{9.33}$$

$$\frac{\partial \mathbf{h}}{\partial_{\mathrm{W}}\mathbf{r}_{\mathrm{WF}}} = \frac{\partial \mathbf{h}}{\partial_{\mathrm{W}}\mathbf{p}} \; \frac{\partial_{\mathrm{W}}\mathbf{p}}{\partial_{\mathrm{W}}\mathbf{r}_{\mathrm{WF}}} \tag{9.34}$$

$$\frac{\partial_{\mathrm{W}}\mathbf{p}}{\partial_{\mathrm{W}}\mathbf{r}_{\mathrm{WF}}} = \mathbf{I} \tag{9.35}$$

Chapter 10

State Estimation

10.1 Bayes Filter

$$\text{bel}(\mathbf{x}_0) = p(\mathbf{x}_0) \tag{10.1}$$

$$\begin{aligned}
\text{bel}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) && (10.2)\\
&= \frac{p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\ \ p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})} && (10.3)
\end{aligned}$$

$$\begin{aligned}
\overline{\text{bel}}(\mathbf{x}_t) &= p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) && (10.4)\\
&= \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\ \ p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})\ \ d\mathbf{x}_{t-1} && (10.5)\\
&= \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{1:t})\ \ \text{bel}(\mathbf{x}_{t-1})\ \ d\mathbf{x}_{t-1} && (10.6)
\end{aligned}$$

10.2 2D Range Measurement Model

$$\boxed{d = \sqrt{(m_x - x)^2 + (m_y - y)^2}} \tag{10.7}$$

$$\boxed{\frac{\partial d}{\partial x} = \frac{-(m_x - x)}{\sqrt{(m_x - x)^2 + (m_y - y)^2}}} \tag{10.8}$$

$$\boxed{\frac{\partial d}{\partial y} = \frac{-(m_y - y)}{\sqrt{(m_x - x)^2 + (m_y - y)^2}}} \tag{10.9}$$

Let $u = (m_x - x)^2 + (m_y - y)^2$,

$$\frac{\partial d}{\partial x} = \frac{\partial d}{\partial u}\frac{\partial u}{\partial x} \qquad \frac{\partial d}{\partial y} = \frac{\partial d}{\partial u}\frac{\partial u}{\partial y} \tag{10.10}$$

$$\frac{\partial d}{\partial u} = \sqrt{u} = \frac{1}{2\sqrt{u}} \tag{10.11}$$

$$\begin{aligned}
\frac{\partial u}{\partial x} &= (m_x - x)^2 && (10.12)\\
&= (m_x - x)(m_x - x) && (10.13)\\
&= m_x^2 - m_x x - x m_x + x^2 && (10.14)\\
&= -2m_x + 2x && (10.15)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial u}{\partial y} &= (m_y - y)^2 && (10.16)\\
&= (m_y - y)(m_y - y) && (10.17)\\
&= m_y^2 - m_y y - y m_y + y^2 && (10.18)\\
&= -2m_y + 2y && (10.19)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial d}{\partial x} &= \frac{\partial d}{\partial u}\frac{\partial u}{\partial x} && (10.20)\\
&= \left(\frac{1}{2\sqrt{u}}\right)(-2m_x + 2x) && (10.21)\\
&= \frac{-(m_x - x)}{\sqrt{u}} && (10.22)\\
&= \frac{-(m_x - x)}{\sqrt{(m_x - x)^2 + (m_y - y)^2}} && (10.23)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial d}{\partial y} &= \frac{\partial d}{\partial u}\frac{\partial u}{\partial y} && (10.24)\\
&= \left(-\frac{1}{2\sqrt{u}}\right)(-2m_y + 2y) && (10.25)\\
&= \frac{-(m_y - y)}{\sqrt{u}} && (10.26)\\
&= \frac{-(m_y - y)}{\sqrt{(m_x - x)^2 + (m_y - y)^2}} && (10.27)
\end{aligned}$$

10.3 2D Relative Bearing Measurement Model

$$\boxed{\theta_b = \tan^{-1}\left(\frac{m_y - y}{m_x - x}\right) - \theta_w} \tag{10.28}$$

$$\boxed{\frac{\partial \theta_b}{\partial x} = \frac{(m_y - y)}{(m_x - x)^2 + (m_y - y)^2}} \tag{10.29}$$

$$\boxed{\frac{\partial \theta_b}{\partial y} = \frac{-(m_x - x)}{(m_x - x)^2 + (m_y - y)^2}} \tag{10.30}$$

Let $u = \dfrac{m_y - y}{m_x - x}$,

$$\begin{aligned}
\frac{\partial \theta_b}{\partial u} &= \tan^{-1}(u) - \theta_w \\
&= \frac{1}{1 + u^2} && (10.31)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial u}{\partial dx} &= \frac{m_y - y}{m_x - x} = \frac{m_y - y}{dx} \\
&= (m_y - y)dx^{-1} \\
&= -(m_y - y)dx^{-2} && (10.32) \\
&= \frac{-(m_y - y)}{dx^2} && (10.33)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial u}{\partial dy} &= \frac{m_y - y}{m_x - x} = \frac{dy}{m_x - x} \\
&= \frac{1}{mx - x} && (10.34)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial dx}{\partial x} &= m_x - x = -1 && (10.35) \\
\frac{\partial dy}{\partial y} &= m_y - y = -1 && (10.36)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\theta_b}{\partial x} &= \frac{\partial\theta_b}{\partial u}\frac{\partial u}{\partial dx}\frac{\partial dx}{\partial x} \\
&= \left(\frac{1}{1+u^2}\right)\left(\frac{-(m_y-y)}{dx^2}\right)(-1) \\
&= \frac{1}{\left(1+\frac{(m_y-y)^2}{(m_x-x)^2}\right)}\frac{(m_y-y)}{(m_x-x)^2} \\
&= \frac{(m_y-y)}{(m_x-x)^2+(m_y-y)^2} \qquad (10.37)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\theta_b}{\partial y} &= \frac{\partial\theta_b}{\partial u}\frac{\partial u}{\partial dy}\frac{\partial dy}{\partial x} \\
&= \left(\frac{1}{1+u^2}\right)\left(\frac{1}{m_x-x}\right)(-1) \\
&= \frac{1}{\left(1+\frac{(m_y-y)^2}{(m_x-x)^2}\right)}\frac{-1}{(m_x-x)} \\
&= \frac{-(m_x-x)}{(m_x-x)^2+(m_y-y)^2} \qquad (10.38)
\end{aligned}$$

Chapter 11

Inertial Measurement Unit (IMU)

11.1 IMU kinematics

$$
\begin{aligned}
{}_{\mathrm{W}}\dot{\mathbf{r}}_{\mathrm{WS}} &= {}_{\mathrm{W}}\mathbf{v}_{\mathrm{WS}} && (11.1)\\
\dot{\mathbf{q}}_{\mathrm{WS}} &= \frac{1}{2}\boldsymbol{\Omega}({}_{\mathrm{S}}\tilde{\boldsymbol{\omega}}_{\mathrm{WS}}, \mathbf{w}_g, \mathbf{b}_g)\mathbf{q}_{\mathrm{WS}} && (11.2)\\
{}_{\mathrm{W}}\dot{\mathbf{v}}_{\mathrm{WS}} &= \mathbf{C}_{\mathrm{WS}}({}_{\mathrm{S}}\tilde{\mathbf{a}}_{\mathrm{WS}} + \mathbf{w}_a - \mathbf{b}_a) + \mathbf{g}_{\mathrm{W}} && (11.3)\\
\dot{\mathbf{b}}_g &= \mathbf{w}_{\mathbf{b}_g} && (11.4)\\
\dot{\mathbf{b}}_a &= -\frac{1}{\tau}\mathbf{b}_a + \mathbf{w}_{\mathbf{b}_g} && (11.5)
\end{aligned}
$$

The matrix $\boldsymbol{\Omega}$ is formed from the estimated angular rate ${}_{\mathrm{S}}\boldsymbol{\omega}_{\mathrm{WS}} = {}_{\mathrm{S}}\tilde{\boldsymbol{\omega}}_{\mathrm{WS}} + \mathbf{w}_g - \mathbf{b}_g$

$$
\mathbf{F}_c = \begin{bmatrix}
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{C}_{\mathrm{WS}} & \mathbf{0}_{3\times3}\\
\mathbf{0}_{3\times3} & \lfloor \mathbf{C}_{\mathrm{WS}}({}_{\mathrm{S}}\tilde{\mathbf{a}}_{\mathrm{WS}} - \mathbf{b}_a)\ \times \rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}_{\mathrm{WS}}\\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3}\\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\frac{1}{\tau}\mathbf{1}_3
\end{bmatrix} \qquad (11.6)
$$

Bibliography

[1] R. Andrae, T. Schulze-Hartung, and P. Melchior, "Dos and don'ts of reduced chi-squared," *arXiv preprint arXiv:1012.3754*, 2010.

[2] P. Furgale, "Representing Robot Pose: The good, the bad, and the ugly." `http://paulfurgale.info/news/2014/6/9/representing-robot-pose-the-good-the-bad-and-the-ugly`, 2014. Accessed: 2018-11-15.

[3] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[4] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," in *Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong, China), May 2014.

[5] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision (ECCV)*, pp. 430–443, Springer, 2006.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1150–1157, 1999.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision (ECCV)*, pp. 404–417, Springer, 2006.

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European Conference on Computer Vision (ECCV)*, pp. 778–792, Springer, 2010.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2564–2571, 2011.

[10] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2548–2555, 2011.

[11] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 34, no. 7, pp. 1281–1298, 2012.

[12] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 165–172, 2017.

[13] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 2, pp. 965–972, 2018.

[14] J. Solà, "Quaternion kinematics for the error-state Kalman filter," Nov. 2017.