3D Algebra for Vision Systems in Robotics

Joan Solà

May 15, 2006

Foreword

Every one entering the robotics community will have to achieve knowledge on a series of basic techniques that are of common use in the field. Some of them are so common that no one mentions them any more. For me, everything related to 3D geometry fitted well into this case. 3D rotations, translations, frame transformations, quaternions, Euler angles... every single topic is by itself not very difficult to understand, specially if one has a good intuition about volumes, about solids in the 3D world.

This understanding and this intuition were of incalculable value to me, but they became useless when, after some weeks, I realized that I still understood everything, but I was no longer able to retain the expressions that permit to play with it, the precise algebra that lives behind.

There is a whole lot of examples. How was the definition of the quaternion from the rotation vector? What was the expression of the rotation matrix given the Euler angles? Worse than that is the never-sufficiently-answered question: my rotation matrix, is it world-to-body or body-to-world? Should I use it directly to express a vector of frame $\mathcal{A}$ in frame $\mathcal{B}$, or should I use its transposed form? Or these ones: my Euler angles, under which of the dozens of conventions are they expressed? Which is the direction of the three x, y and z axes of the frame corresponding to the null rotation? And still (and if you are anglophone you will find this one amazing): which one of the three Euler angles is *yaw*?

One can for sure find extensive literature on these subjects. The only problem might be to get all answers within a unique document, so as to be sure that all definitions assume exactly the same conventions and, also very gratifying, that they use the same terminology and notation.

Something that is more difficult to find in the literature is the treatment of uncertainty in all this algebra. Many areas in robotics are incorporating

probabilistic approaches to guarantee consistent representations and computations in the presence of noise. Every source of information must specify its uncertainty and every transformation of this information must therefore deal with it.

Taking Gaussian probability densities and linearizing the transformations is how the most popular Extended Kalman Filter (EKF) deals with uncertainty. EKF, and many other techniques too, needs the computation of Jacobians, a task that is normally left to symbolic calculators like MAPPLE or MATLAB (the latter is actually using MAPPLE libraries), which normally return huge and cryptic expressions.

This paper you are handling is an attempt to solve all this issues. It started as a bunch of personal notes to fix and to store, in a unique place, every discovery, every expression that was *one hundred percent* sure, in order to avoid worrying about obscure details before using it again.

I put special attention in trying to enlighten the way to manipulate Jacobians by treating the non linear functions as compositions of more elementary transformations. This way, precise expressions of all the necessary Jacobians are systematically developed.

The document grew with time as my work evolved, and now it includes an important part on Vision and on 3D Kinematics, as well as all the algebra for a variety of Bearing-Only/Vision-based SLAM algorithms. At this time I'm working on the observability of moving objects, something that will bring me to vision-based SLAMMOT (SLAM with Moving Objects Tracking).

I hope it will be useful to you.

Joan Solà

Contents

Chapter 1

Differentiation of matrices and vectors

Note: *This section is extracted from a document provided by M. Courdesses, LAAS-CNRS, except that transposed expressions of those proposed in the document are also given.*

1.1 Differentiation with respect to a scalar

We define differentiation of a vector or matrix with respect to a scalar in the following way:

$$\mathbf{z} = \mathbf{z}(t), \qquad \frac{d\mathbf{z}}{dt} = \begin{bmatrix} \frac{dz_1}{dt} & \frac{dz_2}{dt} & \cdots & \frac{dz_n}{dt} \end{bmatrix}^\top \tag{1.1}$$

$$\mathbf{F} = \mathbf{F}(t), \qquad \frac{d\mathbf{F}}{dt} = \begin{bmatrix} \frac{df_{11}}{dt} & \frac{df_{12}}{dt} & \cdots & \frac{df_{1n}}{dt} \\ \frac{df_{21}}{dt} & \frac{df_{22}}{dt} & \cdots & \frac{df_{2n}}{dt} \\ \vdots & \vdots & & \vdots \\ \frac{df_{m1}}{dt} & \frac{df_{m2}}{dt} & \cdots & \frac{df_{mn}}{dt} \end{bmatrix} \tag{1.2}$$

1.2 Differentiation with respect to a vector

Differentiation of scalars with respect to vectors are defined as follows:

$$f = f(\mathbf{x}) = scalar \qquad \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{df}{dx_1} & \frac{df}{dx_2} & \cdots & \frac{df}{dx_n} \end{bmatrix}^\top \tag{1.3}$$

This is often called a gradient and written $\nabla_{\mathbf{x}} f = \frac{df}{d\mathbf{x}}$. We can also differentiate f with respect to the transposed vector:

$$f = f(\mathbf{x}) = scalar \qquad \frac{df}{d\mathbf{x}^\top} = \begin{bmatrix} \frac{df}{dx_1} & \frac{df}{dx_2} & \cdots & \frac{df}{dx_n} \end{bmatrix} \tag{1.4}$$

For vector functions we have:

$$\mathbf{z} = \mathbf{z}(\mathbf{x}) \qquad \frac{d\mathbf{z}^\top}{d\mathbf{x}} = \begin{bmatrix} \frac{dz_1}{dx_1} & \frac{dz_2}{dx_1} & \cdots & \frac{dz_m}{dx_1} \\ \frac{dz_1}{dx_2} & \frac{dz_2}{dx_2} & \cdots & \frac{dz_m}{dx_2} \\ \vdots & \vdots & & \vdots \\ \frac{dz_1}{dx_n} & \frac{dz_2}{dx_n} & \cdots & \frac{dz_m}{dx_n} \end{bmatrix} \tag{1.5}$$

This is called the Jacobian matrix and written $J_{\mathbf{x}}\mathbf{z}(\mathbf{x})$, although we will use its transposed form and the notation $\mathbf{Z}_{\mathbf{x}} \equiv \frac{d\mathbf{z}}{d\mathbf{x}^\top}$:

$$\mathbf{z} = \mathbf{z}(\mathbf{x}) \qquad \frac{d\mathbf{z}}{d\mathbf{x}^\top} = \begin{bmatrix} \frac{dz_1}{dx_1} & \frac{dz_1}{dx_2} & \cdots & \frac{dz_1}{dx_n} \\ \frac{dz_2}{dx_1} & \frac{dz_2}{dx_2} & \cdots & \frac{dz_2}{dx_n} \\ \vdots & \vdots & & \vdots \\ \frac{dz_m}{dx_1} & \frac{dz_m}{dx_2} & \cdots & \frac{dz_m}{dx_n} \end{bmatrix} \tag{1.6}$$

The derivative of a matrix with respect to a vector is here defined as a special partitionned matrix with the following properties

$$\begin{aligned} \mathbf{F} &= \mathbf{F}(\mathbf{x}) \\ \frac{d\mathbf{F}}{d\mathbf{x}} &= \begin{bmatrix} \frac{d\mathbf{F}^\top}{dx_1} & \vdots & \frac{d\mathbf{F}^\top}{dx_2} & \vdots & \cdots & \vdots & \frac{d\mathbf{F}^\top}{dx_n} \end{bmatrix}^\top \\ \frac{d\mathbf{F}}{d\mathbf{x}^\top} &= \begin{bmatrix} \frac{d\mathbf{F}}{dx_1} & \vdots & \frac{d\mathbf{F}}{dx_2} & \vdots & \cdots & \vdots & \frac{d\mathbf{F}}{dx_n} \end{bmatrix} \end{aligned} \tag{1.7}$$

It will be convenient to use a concept of expanded matrices in what is to follow. We define

$$[\mathbf{I}]_{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{bmatrix} \qquad [\mathbf{A}]_{\mathbf{I}} = \begin{bmatrix} a_{11}\mathbf{I} & a_{12}\mathbf{I} & \cdots & a_{1n}\mathbf{I} \\ a_{21}\mathbf{I} & a_{22}\mathbf{I} & \cdots & a_{2n}\mathbf{I} \\ \vdots & & & \vdots \\ a_{m1}\mathbf{I} & a_{m2}\mathbf{I} & \cdots & a_{mn}\mathbf{I} \end{bmatrix} \tag{1.8}$$

which is a Kronecker product type construction valid regardless of the dimension of $\mathbf{A}$.

1.3 Operations involving partial derivatives

As it has been shown, working with the transposed forms leads to a natural extension of the derivation and partial-derivation rules for composed functions. Here the original form is not transcripted and it is substitued by its transposed counterpart.

Consider the functions $f = f(\mathbf{y}, \mathbf{x}, t)$, $\mathbf{y} = \mathbf{y}(\mathbf{x}, t)$ and $\mathbf{x} = \mathbf{x}(t)$. We define the following operations:

$$\begin{aligned} \frac{df}{d\mathbf{x}^\top} &= \frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial f}{\partial \mathbf{x}^\top} \\ \frac{df}{dt} &= \left[\frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial f}{\partial \mathbf{x}^\top}\right]\frac{d\mathbf{x}}{dt} + \frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial t} + \frac{\partial f}{\partial t} \end{aligned} \tag{1.9}$$

Similar operations on the vector functions

$$\mathbf{z} = \mathbf{z}(\mathbf{y}, \mathbf{x}, t) \qquad \mathbf{y} = \mathbf{y}(\mathbf{x}, t) \qquad \mathbf{x} = \mathbf{x}(t)$$

are defined by

$$\begin{aligned} \frac{d\mathbf{z}}{d\mathbf{x}^\top} &= \frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top} \\ \frac{d\mathbf{z}}{dt} &= \left[\frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top}\right]\frac{d\mathbf{x}}{dt} + \frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial t} + \frac{\partial \mathbf{z}}{\partial t} \end{aligned} \tag{1.10}$$

We note that it is often possible to avoid operating on matrices if we re-define them in terms of vectors. In certain special cases, slight modifications of the last section are desirable for specific forms. Also, Kronecker products or the direct matrix product may be used to advantage.

1.4 Linear forms

If, for instance,

$$\mathbf{z} = \mathbf{A}\mathbf{y}, \qquad \mathbf{A} = \mathbf{A}(\mathbf{x}, t), \qquad \mathbf{y} = \mathbf{y}(\mathbf{x}, t), \qquad \mathbf{x} = \mathbf{x}(t)$$

then

$$\frac{d\mathbf{z}}{d\mathbf{x}^\top} = \mathbf{A}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{A}}{\partial \mathbf{x}^\top}[\mathbf{I}]_{\mathbf{y}} \tag{1.11}$$

and

$$\frac{d\mathbf{z}}{dt} = \mathbf{A}\left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}\frac{d\mathbf{x}}{dt} + \frac{\partial \mathbf{y}}{\partial t}\right] + \left\{\frac{\partial \mathbf{A}}{\partial \mathbf{x}^\top}\left[\frac{d\mathbf{x}}{dt}\right]_{\mathbf{I}} + \frac{\partial \mathbf{A}}{\partial t}\right\}\mathbf{y} \tag{1.12}$$

Chapter 2

Non linear function composition for random variables

2.1 Gaussian random variables

A random n-dimensional variable $\mathbf{x}$ is called Gaussian if its probability distribution function (*pdf*) can be written as

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{X}|}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathbf{X}^{-1}(\mathbf{x}-\hat{\mathbf{x}})\Big). \tag{2.1}$$

This *pdf* is fully specified by providing the vector $\hat{\mathbf{x}}$ and the matrix $\mathbf{X}$, its first- and second-order expectations respectively. In effect, from the definition of the expectation operator

$$E\left[f(\mathbf{x})\right] \triangleq \int_\Omega f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{2.2}$$

where Ω is the domain where $\mathbf{x}$ lives and which, being a linear operator, satisfies the following properties

$$\begin{aligned} E\left[k \cdot f(\mathbf{x})\right] &= k \cdot E\left[f(\mathbf{x})\right] \\ E\left[f(\mathbf{x}) + g(\mathbf{x})\right] &= E\left[f(\mathbf{x})\right] + E\left[g(\mathbf{x})\right] \end{aligned} \tag{2.3}$$

we have

$$\begin{aligned} \hat{\mathbf{x}} &= E\left[\mathbf{x}\right] \\ \mathbf{X} &= E\left[(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^\top\right] \end{aligned} \tag{2.4}$$

and we call $\hat{\mathbf{x}}$ the mean of $\mathbf{x}$ and $\mathbf{X}$ its covariances matrix. It is then common to specify a Gaussian variable with the notations

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} - \hat{\mathbf{x}}; \mathbf{X}) \qquad p(\mathbf{x}) = \Gamma(\mathbf{x} - \hat{\mathbf{x}}; \mathbf{X})$$

but in this document we will simply use

$$\mathbf{x} \sim \mathcal{N}\{\hat{\mathbf{x}}; \mathbf{X}\}$$

that clearly reads: *"$\mathbf{x}$ is Gaussian with mean $\hat{\mathbf{x}}$ and covariance $\mathbf{X}$"*.

2.2 One variable functions

We have the Gaussian random variable

$$\mathbf{x} \sim \mathcal{N}\{\hat{\mathbf{x}}, \mathbf{X}\}$$

and the non linear functions

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \qquad \mathbf{z} = \mathbf{g}(\mathbf{y})$$

We aim to approximate the *pdf* of the output variables $\mathbf{y}$ and $\mathbf{z}$ with Gaussian densities, and to precise their means and covariances matrices. For that, we linearize the functions around the means of the independent variables making use of the Jacobians

$$\mathbf{F} = \left.\frac{d\mathbf{f}}{d\mathbf{x}^\top}\right|_{\hat{\mathbf{x}}} \qquad \mathbf{G} = \left.\frac{d\mathbf{g}}{d\mathbf{y}^\top}\right|_{\hat{\mathbf{y}}} \tag{2.5}$$

and write the first order Taylor expansions of $\mathbf{y}$ and $\mathbf{z}$ (see (1.6) and (1.10))

$$\begin{aligned} \mathbf{y} &= \mathbf{f}(\hat{\mathbf{x}}) + \mathbf{F} \cdot (\mathbf{x} - \hat{\mathbf{x}}) + \cdots \\ \mathbf{z} &= \mathbf{g}(\hat{\mathbf{y}}) + \mathbf{G} \cdot (\mathbf{y} - \hat{\mathbf{y}}) + \cdots \end{aligned} \tag{2.6}$$

We apply then the definition of the expectations (2.2) and write

$$\begin{aligned} \hat{\mathbf{y}} &\triangleq E[\mathbf{y}] = E[\mathbf{f}(\hat{\mathbf{x}}) + \mathbf{F} \cdot (\mathbf{x} - \hat{\mathbf{x}}) + \cdots] \\ \hat{\mathbf{z}} &\triangleq E[\mathbf{z}] = E[\mathbf{g}(\hat{\mathbf{y}}) + \mathbf{G} \cdot (\mathbf{y} - \hat{\mathbf{y}}) + \cdots] \end{aligned} \tag{2.7}$$

which using (2.3) leads to the means of the (approximately Gaussian) output variables

$$\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{x}}) \qquad \hat{\mathbf{z}} = \mathbf{g}(\hat{\mathbf{y}}) = \mathbf{g}(\mathbf{f}(\hat{\mathbf{x}})) \tag{2.8}$$

and also

$$\begin{aligned}\mathbf{Y} &\triangleq E\left[(\mathbf{y}-\hat{\mathbf{y}})(\mathbf{y}-\hat{\mathbf{y}})^\top\right] \\ &\approx E\left[\left\{\mathbf{f}(\hat{\mathbf{x}})+\mathbf{F}(\mathbf{x}-\hat{\mathbf{x}})-\mathbf{f}(\hat{\mathbf{x}})\right\}\left\{\mathbf{f}(\hat{\mathbf{x}})^\top+(\mathbf{x}-\hat{\mathbf{x}})^\top\mathbf{F}^\top-\mathbf{f}(\hat{\mathbf{x}})^\top\right\}\right] \\ &= E\left[\mathbf{F}(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^\top\mathbf{F}^\top\right] \\ &= \mathbf{F}\cdot E\left[(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{x}-\hat{\mathbf{x}})^\top\right]\cdot\mathbf{F}^\top \end{aligned} \tag{2.9}$$

$$\begin{aligned}\mathbf{Z} &\triangleq E\left[(\mathbf{z}-\hat{\mathbf{z}})(\mathbf{z}-\hat{\mathbf{z}})^\top\right] \\ &\approx \mathbf{G}\cdot E\left[(\mathbf{y}-\hat{\mathbf{y}})(\mathbf{y}-\hat{\mathbf{y}})^\top\right]\cdot\mathbf{G}^\top\end{aligned}$$

which permits to obtain the covariances matrices:

$$\mathbf{Y}=\mathbf{F}\cdot\mathbf{X}\cdot\mathbf{F}^\top \qquad \mathbf{Z}=\mathbf{G}\cdot\mathbf{Y}\cdot\mathbf{G}^\top=\mathbf{G}\cdot\mathbf{F}\cdot\mathbf{X}\cdot\mathbf{F}^\top\cdot\mathbf{G}^\top. \tag{2.10}$$

These approximations are valid as long as the truncated Taylor series are valid. This translates to the following condition: the Jacobians computed at the means must be valid inside all the region defined by the covariances matrix. This region is commonly defined by the 2- or 3-sigma bound of the original Gaussian.

2.3 Functions of two independent variables

If we add now the new variables

$$\mathbf{u}\sim\mathcal{N}\{\hat{\mathbf{u}},\mathbf{U}\} \qquad \mathbf{v}\sim\mathcal{N}\{\hat{\mathbf{v}},\mathbf{V}\}$$

and we redefine the non linear functions

$$\mathbf{y}=\mathbf{f}(\mathbf{x},\mathbf{u}) \qquad \mathbf{z}=\mathbf{g}(\mathbf{y},\mathbf{v})$$

then the Taylor linearizations are (see (1.6) and (1.10)):

$$\begin{aligned}\mathbf{y} &= \mathbf{f}(\hat{\mathbf{x}},\hat{\mathbf{u}})+\mathbf{F}_{\mathbf{x}}(\mathbf{x}-\hat{\mathbf{x}})+\mathbf{F}_{\mathbf{u}}(\mathbf{u}-\hat{\mathbf{u}})+\cdots \\ \mathbf{z} &= \mathbf{g}(\hat{\mathbf{y}},\hat{\mathbf{v}})+\mathbf{G}_{\mathbf{y}}(\mathbf{y}-\hat{\mathbf{y}})+\mathbf{G}_{\mathbf{v}}(\mathbf{v}-\hat{\mathbf{v}})+\cdots\end{aligned} \tag{2.11}$$

with the Jacobians

$$\mathbf{F}_{\mathbf{x}}=\left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}^\top}\right|_{\hat{\mathbf{x}},\hat{\mathbf{u}}} \qquad \mathbf{F}_{\mathbf{u}}=\left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}^\top}\right|_{\hat{\mathbf{x}},\hat{\mathbf{u}}} \qquad \mathbf{G}_{\mathbf{y}}=\left.\frac{\partial\mathbf{g}}{\partial\mathbf{y}^\top}\right|_{\hat{\mathbf{y}},\hat{\mathbf{v}}} \qquad \mathbf{G}_{\mathbf{v}}=\left.\frac{\partial\mathbf{g}}{\partial\mathbf{v}^\top}\right|_{\hat{\mathbf{y}},\hat{\mathbf{v}}} \tag{2.12}$$

We get the means:

$$\hat{\mathbf{y}} = \mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{u}}) \qquad \hat{\mathbf{z}} = \mathbf{g}(\hat{\mathbf{y}}, \hat{\mathbf{v}}) = \mathbf{g}(\mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{u}}), \hat{\mathbf{v}}) \tag{2.13}$$

and the covariances of the composed vectors:

$$\begin{aligned} \mathbf{Y} &= \mathbf{F_x} \mathbf{X} \mathbf{F_x}^\top + \mathbf{F_u} \mathbf{U} \mathbf{F_u}^\top \\ \mathbf{Z} &= \mathbf{G_y} \mathbf{Y} \mathbf{G_y}^\top + \mathbf{G_v} \mathbf{V} \mathbf{G_v}^\top \\ &= \mathbf{G_y} \mathbf{F_x} \mathbf{X} \mathbf{F_x}^\top \mathbf{G_y}^\top + \mathbf{G_y} \mathbf{F_u} \mathbf{U} \mathbf{F_u}^\top \mathbf{G_y}^\top + \mathbf{G_v} \mathbf{V} \mathbf{G_v}^\top. \end{aligned} \tag{2.14}$$

2.4 Functions of two cross-correlated variables

If we consider eventual cross-correlations between the different variables defined by

$$\begin{aligned} \mathbf{C_{xu}} &= E(\mathbf{x}\mathbf{u}^\top) \qquad \mathbf{C_{xv}} = E(\mathbf{x}\mathbf{v}^\top) \qquad \mathbf{C_{uv}} = E(\mathbf{u}\mathbf{v}^\top) \\ \mathbf{C_{yu}} &= E(\mathbf{y}\mathbf{u}^\top) \qquad \mathbf{C_{yv}} = E(\mathbf{y}\mathbf{v}^\top) \qquad \mathbf{C_{yx}} = E(\mathbf{y}\mathbf{x}^\top) \end{aligned}$$

with

$$\mathbf{C_{ab}} = \mathbf{C_{ba}}^\top \quad \forall \mathbf{a}, \mathbf{b}$$

then the following approximations hold

$$\begin{aligned} \mathbf{Y} &= \mathbf{F_x} \mathbf{X} \mathbf{F_x}^\top + \mathbf{F_x} \mathbf{C_{xu}} \mathbf{F_u}^\top + \mathbf{F_u} \mathbf{C_{ux}} \mathbf{F_x}^\top + \mathbf{F_u} \mathbf{U} \mathbf{F_u}^\top \\ \mathbf{C_{yx}} &= \mathbf{F_x} \mathbf{X} + \mathbf{F_u} \mathbf{C_{ux}} \\ \mathbf{C_{yu}} &= \mathbf{F_x} \mathbf{C_{xu}} + \mathbf{F_u} \mathbf{U} \\ \mathbf{C_{yv}} &= \mathbf{F_x} \mathbf{C_{xv}} + \mathbf{F_u} \mathbf{C_{uv}} \\ \mathbf{Z} &= \mathbf{G_y} \mathbf{Y} \mathbf{G_y}^\top + \mathbf{G_y} \mathbf{C_{yv}} \mathbf{G_v}^\top + \mathbf{G_v} \mathbf{C_{vy}} \mathbf{G_y}^\top + \mathbf{G_v} \mathbf{V} \mathbf{G_v}^\top. \end{aligned} \tag{2.15}$$

Chapter 3

Reference frame transformations

Given the reference frames $\mathcal{F}$ and $\mathcal{W}$ and a point $\mathbf{p}$, we note $\mathbf{p}^{\mathcal{W}}$ the point $\mathbf{p}$ in $\mathcal{W}$ and $\mathbf{p}^{\mathcal{F}}$ the point $\mathbf{p}$ in $\mathcal{F}$. We define the functions $toFrame()$ and $fromFrame()$ as

$$\mathbf{p}^{\mathcal{F}} \triangleq toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}}) \qquad \mathbf{p}^{\mathcal{W}} \triangleq fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})$$

with the following meaning: $toFrame()$ expresses the $\mathcal{W}$-referenced point $\mathbf{p}^{\mathcal{W}}$ (the $\mathcal{W}$-point) in $\mathcal{F}$ frame. $fromFrame()$ expresses the $\mathcal{F}$-referenced point $\mathbf{p}^{\mathcal{F}}$ (the $\mathcal{F}$-point) in $\mathcal{W}$ frame.

3.1 3D implementation

The functions $toFrame()$ and $fromFrame()$ above can be implemented in a number of ways. Here the rotation-translation and the homogeneous matrices are shown.

3.1.1 The Rotation matrix and Translation vector

Given the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$, that express the attitude of a reference frame (name it $\mathcal{F}$ for *Frame*) with respect to another one (name it $\mathcal{W}$ for *World*), we have

$$\begin{aligned} p^{\mathcal{W}} &= \mathbf{R}p^{\mathcal{F}} + \mathbf{t} &&= fromFrame(\mathcal{F}, p^{\mathcal{F}}) \\ p^{\mathcal{F}} &= \mathbf{R}^{\top}p^{\mathcal{W}} - \mathbf{R}^{\top}\mathbf{t} = &&\quad toFrame(\mathcal{F}, p^{\mathcal{W}}) \end{aligned} \tag{3.1}$$

so we can express $\mathcal{F}^{\mathcal{W}} : \{\mathbf{R}, \mathbf{t}\}$ to mean either the $\mathcal{F}$ frame expressed in $\mathcal{W}$ frame or the transformation between them.

The rotation matrix may also be specified in a number of ways. We put its realization in function of the Euler angles, the quaternion and the rotation vector:

Rotation matrix from Euler angles

Given the Euler angles $\mathbf{e} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top$ corresponding to the *roll*, *pitch* and *yaw* angles of the attitude of a reference frame, we have the rotation matrix:

$$\mathbf{R}(\mathbf{e}) = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{3.2}$$

Several conventions exist (with no real consensus) on the axes of rotation and the order in which these rotations are performed. The convention adopted here corresponds to a first rotation along the z axis (yaw), a second rotation along the rotated y' axis (pitch) and a third one along the doubly rotated x'' axis (roll). All rotations follow the so called *right hand rule*, eg. a positive rotation is clockwise (closed right hand fingers) when seen from the origin to the positive direction of the rotation axis (extended right hand thumb).

Euler angles are normally defined for aircrafts, where *pitch* is defined as positive when the vehicle nose raises. It is therefore a definition related to a reference frame of the FRD type (x-Front, y-Right, z-Down). However, it can be extended to any other frame by defining the *roll*, *pitch* and *yaw* angles as being the angles rotated by the frame around each one of its positive axis (always following the right hand rule) in the order $[z, y', x'']$. For example, in an FLU frame (x-Front, y-Left, z-Up), which is more common for terrestrial vehicles, positive pitch means that the nose of the vehicle is sinking because of a positive rotation around the y-axis.

This is a minimal representation for rotations. It has the disadvantage of not being continuous, so when rotations evolve one has to take care to bring all angles to their convenient ranges:

$$\phi \in \begin{bmatrix}-\pi & \pi\end{bmatrix} \qquad \theta \in \begin{bmatrix}-\pi/2 & \pi/2\end{bmatrix} \qquad \psi \in \begin{bmatrix}0 & 2\pi\end{bmatrix}$$

It has the advantage of giving quite compact Jacobians. Averaging is only possible for very small differences which involve no discontinuities. It is a fairly used representation for orientations.

Rotation matrix from quaternion

Given the quaternion $\mathbf{q} = \begin{bmatrix} a & b & c & d \end{bmatrix}^\top$ satisfying $\|\mathbf{q}\| = 1$, the rotation matrix is

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2 \cdot (bc + ad) & 2 \cdot (bd + ac) \\ 2 \cdot (bc - ad) & a^2 - b^2 + c^2 - d^2 & 2 \cdot (cd + ab) \\ 2 \cdot (bd - ac) & 2 \cdot (cd - ab) & a^2 - b^2 - c^2 + d^2 \end{bmatrix} \tag{3.3}$$

The opposite rotation is obtained with the conjugate quaternion $\mathbf{q}^* = \begin{bmatrix} a & -b & -c & -d \end{bmatrix}^\top$. It is easy to show that $\mathbf{R}(\mathbf{q}^*) = \mathbf{R}^\top(\mathbf{q})$.

This is a non minimal representation for rotations, where the negated quaternion represents exactly the same rotation as the original one, that is $\mathbf{R}(-\mathbf{q}) = \mathbf{R}(\mathbf{q})$. Normalization to a unity vector have to be performed at each modification of the quaternion. Averaging is possible (how? I think, by doing the average and re-normalizing) and Jacobians of functions involving the rotation matrix are fairly compact.

Rotation matrix from rotation vector

Given the rotation vector $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^\top$ where the vector direction is the axe of rotation and the norm is the rotated angle, the associated rotation matrix is computed following Rodrigues' formula:

$$\begin{aligned} \theta &= \|\mathbf{v}\| \\ \mathbf{u} &= \frac{\mathbf{v}}{\|\mathbf{v}\|} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^\top \\ \mathbf{\Omega} &= \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \\ \mathbf{R}(\mathbf{v}) &= \mathbf{I} + \sin\theta \cdot \mathbf{\Omega} + (1 - \cos\theta) \cdot \mathbf{\Omega}^2 \end{aligned} \tag{3.4}$$

This is a minimal representation for rotations. It has the advantage of being continuous. It has the disadvantages of being non-bounded or periodical, and that Jacobian expressions are huge. Averaging is not possible because of the periodical behavior.

The rotation vector is related with the quaternion by the definition

$$\mathbf{q} \triangleq \begin{bmatrix} \cos(\theta/2) \\ u_x \sin(\theta/2) \\ u_y \sin(\theta/2) \\ u_z \sin(\theta/2) \end{bmatrix}. \tag{3.5}$$

3.1.2 The Homogeneous matrix

The homogeneous matrix $\mathbf{H}_{\mathcal{F}}^{\mathcal{W}}$ that transforms an homogeneous point $\begin{bmatrix}\mathbf{p}^\top & 1\end{bmatrix}^\top$ from frame $\mathcal{F}$ to frame $\mathcal{W}$ is

$$\mathbf{H}_{\mathcal{F}}^{\mathcal{W}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

and the inverse transformation is done with $\mathbf{H}_{\mathcal{W}}^{\mathcal{F}}$:

$$\mathbf{H}_{\mathcal{W}}^{\mathcal{F}} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{3.7}$$

so we have $toFrame()$ and $fromFrame()$ implemented as

$$\begin{aligned} \begin{bmatrix} \mathbf{p}^{\mathcal{W}} \\ 1 \end{bmatrix} = \mathbf{H}_{\mathcal{F}}^{\mathcal{W}} \cdot \begin{bmatrix} \mathbf{p}^{\mathcal{F}} \\ 1 \end{bmatrix} &\iff \mathbf{p}^{\mathcal{W}} = fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}}) \\ \begin{bmatrix} \mathbf{p}^{\mathcal{F}} \\ 1 \end{bmatrix} = \mathbf{H}_{\mathcal{W}}^{\mathcal{F}} \cdot \begin{bmatrix} \mathbf{p}^{\mathcal{W}} \\ 1 \end{bmatrix} &\iff \mathbf{p}^{\mathcal{F}} = \quad toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}}) \end{aligned} \tag{3.8}$$

3.2 Manipulating different rotation representations in a unique project

We have seen four different representations for rotations so far: the **Euler angles**, the **quaternion**, the **rotation vector** and the **rotation matrix**. All forms have their pros and cons and, except for the rotation vector that I rarely use, are all used within my projects.

The rotation matrix $\mathbf{R}$ is used to perform the rotations to the points and vectors in 3D space. The reason is that the algebra is straightforward and linear, specially if one wants to use the Homogeneous matrix for both rotation and translation.

The quaternion $\mathbf{q}$ is used to store orientation informations in the state vectors. The time evolution equations based on quaternions are continuous and continuously derivable, something that will be very important for later automatic manipulations such as filtering (which is all I do actually).

The Euler angles $\mathbf{e}$ are easy to visualize and to be understood by a human user, and are often used to provide input and output human interface. The pose of a camera in a robot is easily specified this way. They are also useful in odometry as the relatively small changes in vehicle orientation are easily expressed in Euler angles. And in inertial measurement units (IMU), the set of three orthogonal gyrometers will directly provide the three Euler angular rates in robot frame.

In order to facilitate all possible passages between these three representations, six conversion functions must be defined:

1. **Rotation matrix to Euler angles conversion.** Given the rotation matrix
$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.9}$$
the three Euler angles $\mathbf{e} = [\phi\ \theta\ \psi]^\top$ corresponding to the same rotation are given by
$$\begin{aligned} \phi &= \arctan(r_{32}/r_{33}) \\ \theta &= \arcsin(-r_{31}) \\ \psi &= \arctan(r_{21}/r_{11}) \end{aligned} \tag{3.10}$$
We can name this function R2e($\bullet$) and write
$$\mathbf{e} = \text{R2e}(\mathbf{R}).$$

2. **Euler angles to rotation matrix conversion.** This conversion is already specified in (3.2) as $\mathbf{R}(\mathbf{e})$. We can name this function e2R($\bullet$) and write
$$\mathbf{R} = \text{e2R}(\mathbf{e}).$$

3. **Rotation matrix to quaternion conversion.** Given the rotation matrix (3.9), the quaternion $\mathbf{q} = [a\ b\ c\ d]^\top$ corresponding to the same rotation is given by the algorithm in Table 3.1. At first sight it may appear a long algorithm, but it is simply a test for eventual singularities

Table 3.1: Rotation matrix to quaternion conversion algorithm

$$T = 1 + trace(\mathbf{R})$$

`if` $(T > 10^{-8})$

$$\begin{aligned} S &= 2\sqrt{T} \\ a &= S/4 \\ b &= (r_{32} - r_{23})/S \\ c &= (r_{13} - r_{31})/S \\ d &= (r_{21} - r_{12})/S \end{aligned}$$

`else`

`if` $(r_{11} > r_{22})$ `and` $(r_{11} > r_{33})$

$$\begin{aligned} S &= 2\sqrt{1 + r_{11} - r_{22} - r_{33}} \\ a &= (r_{23} - r_{32})/S \\ b &= -S/4 \\ c &= (r_{21} - r_{12})/S \\ d &= (r_{13} - r_{31})/S \end{aligned}$$

`elseif` $(r_{22} > r_{33})$

$$\begin{aligned} S &= 2\sqrt{1 - r_{11} + r_{22} - r_{33}} \\ a &= (r_{31} - r_{13})/S \\ b &= (r_{21} - r_{12})/S \\ c &= -S/4 \\ d &= (r_{32} - r_{23})/S \end{aligned}$$

`else`

$$\begin{aligned} S &= 2\sqrt{1 - r_{11} - r_{22} + r_{33}} \\ a &= (r_{12} - r_{21})/S \\ b &= (r_{13} - r_{31})/S \\ c &= (r_{32} - r_{23})/S \\ d &= -S/4 \end{aligned}$$

`end`

`end`

that results in the selection of a particular (one over four) small group of equations.

We can name this function R2q($\bullet$) and write

$$\mathbf{q} = \text{R2q}(\mathbf{R}).$$

4. **Quaternion to rotation matrix conversion.** This conversion is already specified in (3.3) as $\mathbf{R}(\mathbf{q})$. We can name this function q2R($\bullet$) and write

$$\mathbf{R} = \text{q2R}(\mathbf{q}).$$

5. **Quaternion to Euler angles conversion.** Given the quaternion $\mathbf{q} = [a\ b\ c\ d]^\top$, the Euler angles $\mathbf{e} = [\phi\ \theta\ \psi]^\top$ corresponding to the same rotation are given by

$$\begin{aligned} \phi &= \arctan\left(\frac{2cd + 2ab}{a^2 - b^2 - c^2 + d^2}\right) \\ \theta &= \arcsin\left(-2bd + 2ac\right) \\ \psi &= \arctan\left(\frac{2bc + 2ad}{a^2 + b^2 - c^2 - d^2}\right) \end{aligned} \tag{3.11}$$

We can name this function q2e($\bullet$) and write

$$\mathbf{e} = \text{q2e}(\mathbf{q}).$$

6. **Euler angles to quaternion conversion.** This conversion is much more complicated if one wants to express it in a closed form. We propose to convert the Euler angles to the rotation matrix and then reconvert the result into the quaternion with the expressions above. We can name this function e2q($\bullet$) and write

$$\mathbf{q} = \text{e2q}(\mathbf{e}) = \text{R2q}\big(\text{e2R}(\mathbf{e})\big).$$

3.3 Computation of Jacobians

The Jacobians of the functions $toFrame()$ and $fromFrame()$ with respect to its arguments are defined as follows:

$$\begin{aligned}
\mathbf{TF}_{\mathcal{F}}(\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{W}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}})}{\partial \mathcal{F}^{\top}}\right|_{\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{W}}} \\
\mathbf{TF}_{\mathbf{p}}(\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{W}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}})}{\partial \mathbf{p}^{\mathcal{W}\top}}\right|_{\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{W}}} \\
\mathbf{FF}_{\mathcal{F}}(\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})}{\partial \mathcal{F}^{\top}}\right|_{\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{F}}} \\
\mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})}{\partial \mathbf{p}^{\top}}\right|_{\hat{\mathcal{F}}, \hat{\mathbf{p}}^{\mathcal{F}}}
\end{aligned} \tag{3.12}$$

3.3.1 Transformations using Euler angles

We give the Jacobians of functions $toFrame()$ and $fromFrame()$ when orientations are specified in Euler angles. A generic frame $\mathcal{F}$ is expressed as a 6-dimension vector:

$$\mathcal{F} = \begin{bmatrix} \mathbf{t} \\ \mathbf{e} \end{bmatrix} = \begin{bmatrix} t_x & t_y & t_z & \phi & \theta & \psi \end{bmatrix}^{\top} \tag{3.13}$$

so Jacobians will have the form:

$$\mathbf{FF}_{\mathcal{F}} = \begin{bmatrix} \mathbf{FF}_{\mathbf{t}} & \mathbf{FF}_{\mathbf{e}} \end{bmatrix} \qquad \mathbf{TF}_{\mathcal{F}} = \begin{bmatrix} \mathbf{TF}_{\mathbf{t}} & \mathbf{TF}_{\mathbf{e}} \end{bmatrix} \tag{3.14}$$

Jacobians of $fromFrame()$ are:

$$\begin{aligned}
\mathbf{FF}_{\mathbf{t}}(\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})}{\partial \mathbf{t}^{\top}}\right|_{\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{I} \\
\mathbf{FF}_{\mathbf{e}}(\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})}{\partial \mathbf{e}^{\top}}\right|_{\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}} = \left.\frac{\partial(\mathbf{R}(\mathbf{e}) \cdot \mathbf{p}^{\mathcal{F}})}{\partial \mathbf{e}^{\top}}\right|_{\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}} \\
\mathbf{FF}_{\mathbf{p}}(\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F}, \mathbf{p}^{\mathcal{F}})}{\partial \mathbf{p}^{\mathcal{F}\top}}\right|_{\hat{\mathbf{t}}, \hat{\mathbf{e}}, \hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{R}(\mathbf{e})
\end{aligned} \tag{3.15}$$

where, using (1.11), $\mathbf{F_e}$ can be further developed as

$$\mathbf{FF_e}(\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}^{\mathcal{F}}) = \left[\tfrac{\partial \mathbf{R}}{\partial \phi}\mathbf{p}^{\mathcal{F}} \quad \tfrac{\partial \mathbf{R}}{\partial \theta}\mathbf{p}^{\mathcal{F}} \quad \tfrac{\partial \mathbf{R}}{\partial \psi}\mathbf{p}^{\mathcal{F}}\right]_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}^{\mathcal{F}}} \tag{3.16}$$

Jacobians of $toFrame()$ are:

$$\begin{aligned}
\mathbf{TF_t}(\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{t}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}} = -\mathbf{R}^\top(\mathbf{e}) \\
\mathbf{TF_e}(\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{e}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}} = \left.\frac{\partial(\mathbf{R}^\top(\mathbf{e})\cdot(\mathbf{p}-\mathbf{t}))}{\partial \mathbf{e}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}} \\
\mathbf{TF_p}(\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{p}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}} = \mathbf{R}^\top(\mathbf{e})
\end{aligned} \tag{3.17}$$

where $\mathbf{TF_e}$ can be further developed as

$$\mathbf{TF_e}(\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}) = \left[\tfrac{\partial \mathbf{R}^\top}{\partial \phi}(\mathbf{p}-\mathbf{t}) \quad \tfrac{\partial \mathbf{R}^\top}{\partial \theta}(\mathbf{p}-\mathbf{t}) \quad \tfrac{\partial \mathbf{R}^\top}{\partial \psi}(\mathbf{p}-\mathbf{t})\right]_{\hat{\mathbf{t}},\hat{\mathbf{e}},\hat{\mathbf{p}}} \tag{3.18}$$

3.3.2 Transformations using quaternions

We give the Jacobians of functions $toFrame()$ and $fromFrame()$ when orientations are specified in quaternions. A generic frame $\mathcal{F}$ is expressed as a 7-dimension vector:

$$\mathcal{F} = \begin{bmatrix}\mathbf{t}\\ \mathbf{q}\end{bmatrix} = \begin{bmatrix} t_x & t_y & t_z & a & b & c & d\end{bmatrix}^\top \tag{3.19}$$

so Jacobians will have the form:

$$\mathbf{FF}_{\mathcal{F}} = \begin{bmatrix}\mathbf{FF_t} & \mathbf{FF_q}\end{bmatrix} \qquad \mathbf{TF}_{\mathcal{F}} = \begin{bmatrix}\mathbf{TF_t} & \mathbf{TF_q}\end{bmatrix} \tag{3.20}$$

Jacobians of $fromFrame()$ are:

$$\begin{aligned}
\mathbf{FF_t}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{t}^{\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{I} \\
\mathbf{FF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{q}^{\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \left.\frac{\partial(\mathbf{R}(\mathbf{q})\cdot\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{q}^{\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} \\
\mathbf{FF_p}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\ fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{p}^{\mathcal{F}\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{R}(\mathbf{q})
\end{aligned} \tag{3.21}$$

where, using (1.11), $\mathbf{FF_q}$ can be further developed as

$$\mathbf{FF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) = \left[\tfrac{\partial \mathbf{R}}{\partial a}\mathbf{p}^{\mathcal{F}} \quad \tfrac{\partial \mathbf{R}}{\partial b}\mathbf{p}^{\mathcal{F}} \quad \tfrac{\partial \mathbf{R}}{\partial c}\mathbf{p}^{\mathcal{F}} \quad \tfrac{\partial \mathbf{R}}{\partial d}\mathbf{p}^{\mathcal{F}}\right]_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} \tag{3.22}$$

The quaternion representation allows for interesting expressions for the above Jacobian. We give for this case the complete forms to show the compacity of the results. Specific development of (3.22) using the quaternion rotation matrix (3.3) leads to the following procedure: build the matrix

$$\mathbf{\Pi} = \begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix}_{\hat{\mathbf{q}}} \tag{3.23}$$

then the vector

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = 2\mathbf{\Pi}\mathbf{p}^{\mathcal{F}} \tag{3.24}$$

and the Jacobian is finally

$$\mathbf{FF_q} = \begin{bmatrix} s_2 & -s_1 & s_4 & -s_3 \\ s_3 & -s_4 & -s_1 & s_2 \\ s_4 & s_3 & -s_2 & -s_1 \end{bmatrix} \tag{3.25}$$

Jacobians of $toFrame()$ are:

$$\begin{aligned}
\mathbf{TF_t}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{t}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}} = -\mathbf{R}^\top(\mathbf{q}) \\
\mathbf{TF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}} = \left.\frac{\partial(\mathbf{R}^\top(\mathbf{q})\cdot(\mathbf{p}-\mathbf{t}))}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}} \\
\mathbf{TF_p}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}) &\triangleq \left.\frac{\partial\ toFrame(\mathcal{F},\mathbf{p})}{\partial \mathbf{p}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}} = \mathbf{R}^\top(\mathbf{q})
\end{aligned} \tag{3.26}$$

where $\mathbf{TF_q}$ can be further developed as

$$\mathbf{TF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}) = \left[\frac{\partial \mathbf{R}^\top}{\partial a}(\mathbf{p}-\mathbf{t}) \quad \frac{\partial \mathbf{R}^\top}{\partial b}(\mathbf{p}-\mathbf{t}) \quad \frac{\partial \mathbf{R}^\top}{\partial c}(\mathbf{p}-\mathbf{t}) \quad \frac{\partial \mathbf{R}^\top}{\partial d}(\mathbf{p}-\mathbf{t})\right]_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}} \tag{3.27}$$

To obtain the closed forms build the matrix

$$\mathbf{\Pi}^* = \begin{bmatrix} b & c & d \\ a & d & -c \\ -d & a & b \\ c & -b & a \end{bmatrix}_{\hat{\mathbf{q}}} \tag{3.28}$$

then the vector

$$\mathbf{s}^* = \begin{bmatrix} s_1^* \\ s_2^* \\ s_3^* \\ s_4^* \end{bmatrix} = 2\mathbf{\Pi}^*(\mathbf{p}-\mathbf{t}) \tag{3.29}$$

and the Jacobian is finally

$$\mathbf{TF_q} = \begin{bmatrix} s_2^* & s_1^* & -s_4^* & s_3^* \\ s_3^* & s_4^* & s_1^* & -s_2^* \\ s_4^* & -s_3^* & s_2^* & s_1^* \end{bmatrix} \tag{3.30}$$

As a summary of all the presented Jacobians, for $fromFrame()$ we have:

$$\begin{aligned} \mathbf{\Pi} &= \begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix}_{\hat{\mathbf{q}}} \\ \mathbf{s} &= \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = 2\mathbf{\Pi}\mathbf{p}^{\mathcal{F}} \end{aligned} \tag{3.31}$$

and

$$\begin{aligned} \mathbf{FF_t} &= \mathbf{I} \\ \mathbf{FF_q}(\hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{F}}) &= \begin{bmatrix} s_2 & -s_1 & s_4 & -s_3 \\ s_3 & -s_4 & -s_1 & s_2 \\ s_4 & s_3 & -s_2 & -s_1 \end{bmatrix} \\ \mathbf{FF_p}(\hat{\mathbf{q}}) &= \mathbf{R} \end{aligned} \tag{3.32}$$

For $toFrame()$ we have:

$$\begin{aligned} \mathbf{\Pi}^* &= \begin{bmatrix} b & c & d \\ a & d & -c \\ -d & a & b \\ c & -b & a \end{bmatrix}_{\hat{\mathbf{q}}} \\ \mathbf{s}^* &= \begin{bmatrix} s_1^* \\ s_2^* \\ s_3^* \\ s_4^* \end{bmatrix} = 2\mathbf{\Pi}^*(\mathbf{p} - \mathbf{t}) \end{aligned} \tag{3.33}$$

and

$$\begin{aligned} \mathbf{TF_t}(\hat{\mathbf{q}}) &= -\mathbf{R}^\top \\ \mathbf{TF_q}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}) &= \begin{bmatrix} s_2^* & s_1^* & -s_4^* & s_3^* \\ s_3^* & s_4^* & s_1^* & -s_2^* \\ s_4^* & -s_3^* & s_2^* & s_1^* \end{bmatrix} \\ \mathbf{TF_p}(\hat{\mathbf{q}}) &= \mathbf{R}^\top \end{aligned} \tag{3.34}$$

These expressions can now be easily and efficiently coded into the functions $fromFrameJac(\hat{\mathcal{F}}, \hat{\mathbf{p}^{\mathcal{F}}})$ and $toFrameJac(\hat{\mathcal{F}}, \hat{\mathbf{p}})$.

Chapter 4

Landmark observations from a robot with a camera

4.1 The pin hole camera

Given a camera with calibration parameters $\mathbf{c} = [u_0 \ v_0 \ \alpha_u \ \alpha_v]^\top$ and given the reference frames

- $\mathcal{C}$: Camera frame: RDF (x-Right, y-Down, z-Front)
- $\mathcal{I}$: Image frame: RD (u-Right, v-Down)

which we denote $\mathcal{C} : \{RDF; RD\}$, the intrinsic matrix is defined:

$$\mathbf{K} \triangleq \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

and the projection matrix:

$$\mathbf{T}_i \triangleq \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.2}$$

4.1.1 The forward or projection function

A 3D-point $\mathbf{p}^{\mathcal{C}} = \begin{bmatrix} x^{\mathcal{C}} & y^{\mathcal{C}} & z^{\mathcal{C}} \end{bmatrix}^\top$ in $\mathcal{C}$ frame is projected onto the pixel $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^\top$ by the projective transformation

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \mathbf{K} \cdot \mathbf{p}^{\mathcal{C}} = \begin{bmatrix} u_0\ z^{\mathcal{C}} + \alpha_u\ x^{\mathcal{C}} \\ v_0\ z^{\mathcal{C}} + \alpha_v\ y^{\mathcal{C}} \\ z^{\mathcal{C}} \end{bmatrix} \tag{4.3}$$

giving the pin-hole camera equations

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = h^{\mathcal{C}}(\mathbf{p}^{\mathcal{C}}, \mathbf{c}) = \begin{bmatrix} u_0 + \alpha_u \frac{x^{\mathcal{C}}}{z^{\mathcal{C}}} \\ v_0 + \alpha_v \frac{y^{\mathcal{C}}}{z^{\mathcal{C}}} \end{bmatrix} \tag{4.4}$$

The measured pixel is

$$\mathbf{y} = \mathbf{u} + \upsilon \tag{4.5}$$

where υ is a white Gaussian noise $\upsilon \sim \mathcal{N}\{\mathbf{0}; \mathbf{R}\}$, which leads to the following characterisation of $\mathbf{u}$:

$$\mathbf{u} \sim \mathcal{N}\{\mathbf{y}; \mathbf{R}\}. \tag{4.6}$$

The Jacobian expression of the pin-hole equation is straightforward:

$$\mathbf{H}^{\mathcal{C}}_{\mathbf{p}} = \left. \frac{\partial h^{\mathcal{C}}(\mathbf{p}^{\mathcal{C}}, \mathbf{c})}{\partial \mathbf{p}^{\mathcal{C}\top}} \right|_{\hat{\mathbf{p}}^{\mathcal{C}}, \mathbf{c}} = \begin{bmatrix} \frac{\alpha_u}{z^{\mathcal{C}}} & 0 & -\frac{\alpha_u x^{\mathcal{C}}}{(z^{\mathcal{C}})^2} \\ 0 & \frac{\alpha_v}{z^{\mathcal{C}}} & -\frac{\alpha_v y^{\mathcal{C}}}{(z^{\mathcal{C}})^2} \end{bmatrix} \tag{4.7}$$

In cases where camera auto-calibration is desired, one may include $\mathbf{c}$ into the state vector of the map. The Jacobian with respect to these calibration parameters is then

$$\mathbf{H}^{\mathcal{C}}_{\mathbf{c}} = \left. \frac{\partial h^{\mathcal{C}}(\mathbf{p}^{\mathcal{C}}, \mathbf{c})}{\partial \mathbf{c}^{\top}} \right|_{\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}}} = \begin{bmatrix} 1 & 0 & \frac{x^{\mathcal{C}}}{z^{\mathcal{C}}} & 0 \\ 0 & 1 & 0 & \frac{y^{\mathcal{C}}}{z^{\mathcal{C}}} \end{bmatrix}_{\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}}} \tag{4.8}$$

These equations can also be easily coded into $pinHoleJac(\hat{\mathbf{p}^{\mathcal{C}}}, \hat{\mathbf{c}})$.

4.1.2 The inverse or retro-projection function

A pixel $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^\top$ defines a line in the $\mathcal{C}$-referenced $3D$ space which can be parametrized by the depth s with the expression:

$$\mathbf{p}^{\mathcal{C}} = g^{\mathcal{C}}(\mathbf{u}, s, \mathbf{c}) = s \begin{bmatrix} \frac{u-u_0}{\alpha_u} \\ \frac{v-v_0}{\alpha_v} \\ 1 \end{bmatrix}. \tag{4.9}$$

This is called the retro-projection function of the camera. Its Jacobians are:

$$\begin{aligned} \mathbf{G}^{\mathcal{C}}_{\mathbf{u}} &= \left.\frac{\partial g^{\mathcal{C}}}{\partial \mathbf{u}^\top}\right|_{\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = s \begin{bmatrix} \frac{1}{\alpha_u} & 0 \\ 0 & \frac{1}{\alpha_v} \\ 0 & 0 \end{bmatrix} \\ \mathbf{G}^{\mathcal{C}}_{s} &= \left.\frac{\partial g^{\mathcal{C}}}{\partial s}\right|_{\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \begin{bmatrix} \frac{u-u_0}{\alpha_u} \\ \frac{v-v_0}{\alpha_v} \\ 1 \end{bmatrix} \\ \mathbf{G}^{\mathcal{C}}_{\mathbf{c}} &= \left.\frac{\partial g^{\mathcal{C}}}{\partial \mathbf{c}^\top}\right|_{\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = s \begin{bmatrix} -\frac{1}{\alpha_u} & 0 & -\frac{u-u_0}{\alpha_u^2} & 0 \\ 0 & -\frac{1}{\alpha_v} & 0 & -\frac{v-v_0}{\alpha_v^2} \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{4.10}$$

where from (4.6) we have $\hat{\mathbf{u}} = \mathbf{y}$.

4.2 Composition of frame transformations and observation functions

A robot $\mathcal{R}$ uses its own camera $\mathcal{C}$ to take a picture of a fixed landmark $\mathbf{p}$. Robot pose in the world $\mathcal{R}^{\mathcal{W}}$, camera pose in the robot $\mathcal{C}^{\mathcal{R}}$ and landmark position in the world $\mathbf{p}^{\mathcal{W}}$ may be included in a fully correlated map. Frame super-indexes for variables in the map are omitted and so we identify $\mathcal{R}^{\mathcal{W}} \equiv \mathcal{R}$, $\mathcal{C}^{\mathcal{R}} \equiv \mathcal{C}$ and $\mathbf{p}^{\mathcal{W}} \equiv \mathbf{p}$. We can have the expressions of the landmark position in any frame:

$$\mathbf{p}^{\mathcal{C}} = toFrame(\mathcal{C}, \mathbf{p}^{\mathcal{R}}) \qquad \mathbf{p}^{\mathcal{R}} = toFrame(\mathcal{R}, \mathbf{p}). \tag{4.11}$$

4.2.1 The observation function

The observation function expressed in $\mathcal{C}$-frame is the simple pin-hole camera projection function (see 4.1). Generically we say that

$$\mathbf{u} = h^{\mathcal{C}}(\mathbf{p}^{\mathcal{C}}, \mathbf{c}).$$

We can get the observation as a function of the variables in the map:

$$\begin{aligned}\mathbf{u} &= h^{\mathcal{C}}(toFrame(\mathcal{C}, \mathbf{p}^{\mathcal{R}}), \mathbf{c}) \\ &= h^{\mathcal{C}}(toFrame(\mathcal{C}, toFrame(\mathcal{R}, \mathbf{p})), \mathbf{c}) \\ &\triangleq h(\mathcal{R}, \mathcal{C}, \mathbf{p}, \mathbf{c})\end{aligned} \tag{4.12}$$

Using (1.10), we can compute the Jacobians of $h()$ with respect to all of its arguments:

$$\begin{aligned}\mathbf{H}_{\mathcal{R}} &= \left.\frac{\partial h}{\partial \mathcal{R}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{p}},\hat{\mathbf{c}}} = \left.\frac{\partial h^{\mathcal{C}}}{\partial \mathbf{p}^{\mathcal{C}\top}}\right|_{\hat{\mathbf{p}}^{\mathcal{C}},\hat{\mathbf{c}}} \left.\frac{\partial \mathbf{p}^{\mathcal{C}}}{\partial \mathbf{p}^{\mathcal{R}\top}}\right|_{\hat{\mathcal{C}},\hat{\mathbf{p}}^{\mathcal{R}}} \left.\frac{\partial \mathbf{p}^{\mathcal{R}}}{\partial \mathcal{R}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathbf{p}}} \\ \mathbf{H}_{\mathcal{C}} &= \left.\frac{\partial h}{\partial \mathcal{C}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{p}},\hat{\mathbf{c}}} = \left.\frac{\partial h^{\mathcal{C}}}{\partial \mathbf{p}^{\mathcal{C}\top}}\right|_{\hat{\mathbf{p}}^{\mathcal{C}},\hat{\mathbf{c}}} \left.\frac{\partial \mathbf{p}^{\mathcal{C}}}{\partial \mathcal{C}^{\top}}\right|_{\hat{\mathcal{C}},\hat{\mathbf{p}}^{\mathcal{R}}} \\ \mathbf{H}_{\mathbf{p}} &= \left.\frac{\partial h}{\partial \mathbf{p}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{p}},\hat{\mathbf{c}}} = \left.\frac{\partial h^{\mathcal{C}}}{\partial \mathbf{p}^{\mathcal{C}\top}}\right|_{\hat{\mathbf{p}}^{\mathcal{C}},\hat{\mathbf{c}}} \left.\frac{\partial \mathbf{p}^{\mathcal{C}}}{\partial \mathbf{p}^{\mathcal{R}\top}}\right|_{\hat{\mathcal{C}},\hat{\mathbf{p}}^{\mathcal{R}}} \left.\frac{\partial \mathbf{p}^{\mathcal{R}}}{\partial \mathbf{p}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathbf{p}}} \\ \mathbf{H}_{\mathbf{c}} &= \left.\frac{\partial h}{\partial \mathbf{c}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{p}},\hat{\mathbf{c}}} = \left.\frac{\partial h^{\mathcal{C}}}{\partial \mathbf{c}^{\top}}\right|_{\hat{\mathbf{p}}^{\mathcal{C}},\hat{\mathbf{c}}}\end{aligned} \tag{4.13}$$

where

$$\hat{\mathbf{p}}^{\mathcal{C}} = toFrame(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{R}}) \qquad \hat{\mathbf{p}}^{\mathcal{R}} = toFrame(\hat{\mathcal{R}}, \hat{\mathbf{p}}).$$

These expressions can be rewritten as

$$\begin{aligned}\mathbf{H}_{\mathcal{R}} &= \mathbf{H}^{\mathcal{C}}_{\mathbf{p}}(\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}}) \cdot \mathbf{TF}_{\mathbf{p}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{TF}_{\mathcal{F}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}) \\ \mathbf{H}_{\mathcal{C}} &= \mathbf{H}^{\mathcal{C}}_{\mathbf{p}}(\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}}) \cdot \mathbf{TF}_{\mathcal{F}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{R}}) \\ \mathbf{H}_{\mathbf{p}} &= \mathbf{H}^{\mathcal{C}}_{\mathbf{p}}(\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}}) \cdot \mathbf{TF}_{\mathbf{p}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{TF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}) \\ \mathbf{H}_{\mathbf{c}} &= \mathbf{H}^{\mathcal{C}}_{\mathbf{c}}(\hat{\mathbf{p}}^{\mathcal{C}}, \hat{\mathbf{c}})\end{aligned} \tag{4.14}$$

4.2.2 The inverse observation function

A pixel $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^\top$ defines a line in the $\mathcal{W}$-referenced $3D$ space which can be parametrized by the depth s with the expression:

$$\begin{aligned} \mathbf{p} &= fromFrame(\mathcal{R}, \mathbf{p}^{\mathcal{R}}) \\ &= fromFrame(\mathcal{R}, fromFrame(\mathcal{C}, \mathbf{p}^{\mathcal{C}})) \\ &= fromFrame(\mathcal{R}, fromFrame(\mathcal{C}, g^{\mathcal{C}}(\mathbf{u}, s, \mathbf{c}))) \\ &\triangleq g(\mathcal{R}, \mathcal{C}, \mathbf{u}, s, \mathbf{c}). \end{aligned} \tag{4.15}$$

Its Jacobians are:

$$\begin{aligned} \mathbf{G}_{\mathcal{R}} &= \left.\frac{\partial g}{\partial \mathcal{R}^\top}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \mathbf{FF}_{\mathcal{F}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}^{\mathcal{R}}) \\ \mathbf{G}_{\mathcal{C}} &= \left.\frac{\partial g}{\partial \mathcal{C}^\top}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{FF}_{\mathcal{F}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{C}}) \\ \mathbf{G}_{\mathbf{u}} &= \left.\frac{\partial g}{\partial \mathbf{u}^\top}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{C}}) \cdot \mathbf{G}_{\mathbf{u}}^{\mathcal{C}}(\hat{\mathbf{u}}, \hat{s}, \hat{\mathbf{c}}) \\ \mathbf{G}_{s} &= \left.\frac{\partial g}{\partial s}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{C}}) \cdot \mathbf{G}_{s}^{\mathcal{C}}(\hat{\mathbf{u}}, \hat{s}, \hat{\mathbf{c}}) \\ \mathbf{G}_{\mathbf{c}} &= \left.\frac{\partial g}{\partial \mathbf{c}^\top}\right|_{\hat{\mathcal{R}},\hat{\mathcal{C}},\hat{\mathbf{u}},\hat{s},\hat{\mathbf{c}}} = \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\mathbf{p}}^{\mathcal{R}}) \cdot \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{C}}) \cdot \mathbf{G}_{\mathbf{c}}^{\mathcal{C}}(\hat{\mathbf{u}}, \hat{s}, \hat{\mathbf{c}}) \end{aligned} \tag{4.16}$$

where

$$\hat{\mathbf{p}}^{\mathcal{C}} = g^{\mathcal{C}}(\hat{\mathbf{u}}, \hat{s}, \hat{\mathbf{c}}) \qquad \hat{\mathbf{p}}^{\mathcal{R}} = fromFrame(\hat{\mathcal{C}}, \hat{\mathbf{p}}^{\mathcal{C}}) \tag{4.17}$$

and $\hat{\mathbf{u}} = \mathbf{y}$.

Chapter 5

Robot kinematics

As the robot moves, its pose $\mathcal{R}$ evolves following a particular uncertain model. Generically we say that[1]

$$\mathcal{R}^+ = f(\mathcal{R}, \mathbf{v})$$

where $\mathbf{v} \sim \{\hat{\mathbf{v}}; \mathbf{V}\}$ is a noisy vector of controls or odometry measurements.

The form of the evolution function f depends on the chosen representation for $\mathcal{R}$ –Euler angles or quaternion–, which has already been discussed, and on the format of the odometry data $\mathbf{v}$. it has already been pointed that the quaternion representation offers interesting properties. This chapter treats only the algebra for modeling robot kinematics when orientations are specified in quaternions, *ie.* $\mathcal{R}^\top = [\mathbf{x}^\top \ \mathbf{q}^\top]$.

5.1 Odometry models

The algebra for two different odometry formats is presented below.

[1]The notation A^+ means *the updated value of* A, for any A.

5.1.1 Position and Euler increments in robot frame

Odometry vector is in the form

$$\mathbf{v} = \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{e} \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix}$$

where $\delta\mathbf{x}$ is the increment in the robot position and $\delta\mathbf{e}$ is the increment in the robot orientation expressed in Euler angles, all in $\mathcal{R}$ frame. The evolution function is

$$\begin{aligned} \mathbf{x}^+ &= \mathbf{x} + \mathbf{R}(\mathbf{q}) \cdot \delta\mathbf{x} = fromFrame(\mathcal{R}, \delta\mathbf{x}) \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2}\boldsymbol{\Omega}(\delta\mathbf{e}) \cdot \mathbf{q} \end{aligned} \tag{5.1}$$

where $\mathbf{R}(\mathbf{q})$ is the rotation matrix corresponding to the orientation $\mathbf{q}$ and $\boldsymbol{\Omega}(\delta\mathbf{e})$ is the skew symmetric matrix

$$\boldsymbol{\Omega}(\delta\mathbf{e}) = \begin{bmatrix} 0 & -\delta\phi & -\delta\theta & -\delta\psi \\ \delta\phi & 0 & \delta\psi & -\delta\theta \\ \delta\theta & -\delta\psi & 0 & \delta\phi \\ \delta\psi & \delta\theta & -\delta\phi & 0 \end{bmatrix} \tag{5.2}$$

The associated Jacobians are:

$$\begin{aligned} \mathbf{X}^+_{\mathbf{x}} &= \left.\frac{\partial \mathbf{x}^+}{\partial \mathbf{x}^\top}\right|_{\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}, \hat{\delta\mathbf{e}}} = \mathbf{FF}_{\mathbf{t}}(\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}) = \mathbf{I} \\ \mathbf{X}^+_{\mathbf{q}} &= \left.\frac{\partial \mathbf{x}^+}{\partial \mathbf{q}^\top}\right|_{\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}, \hat{\delta\mathbf{e}}} = \mathbf{FF}_{\mathbf{q}}(\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}) \\ \mathbf{X}^+_{\delta\mathbf{x}} &= \left.\frac{\partial \mathbf{x}^+}{\partial \delta\mathbf{x}^\top}\right|_{\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}, \hat{\delta\mathbf{e}}} = \mathbf{FF}_{\mathbf{p}}(\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}) = \mathbf{R}(\hat{\mathbf{q}}) \\ \mathbf{X}^+_{\delta\mathbf{e}} &= \left.\frac{\partial \mathbf{x}^+}{\partial \delta\mathbf{e}^\top}\right|_{\hat{\mathcal{R}}, \hat{\delta\mathbf{x}}, \hat{\delta\mathbf{e}}} = \mathbf{0} \end{aligned} \tag{5.3}$$

and

$$\begin{aligned}
\mathbf{Q}_{\mathbf{x}}^{+} &= \left.\frac{\partial \mathbf{q}^{+}}{\partial \mathbf{x}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\delta\mathbf{x}},\hat{\delta\mathbf{e}}} = \mathbf{0} \\
\mathbf{Q}_{\mathbf{q}}^{+} &= \left.\frac{\partial \mathbf{q}^{+}}{\partial \mathbf{q}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\delta\mathbf{x}},\hat{\delta\mathbf{e}}} = \mathbf{I} + \frac{1}{2}\mathbf{\Omega}(\hat{\delta\mathbf{e}}) \\
\mathbf{Q}_{\delta\mathbf{x}}^{+} &= \left.\frac{\partial \mathbf{q}^{+}}{\partial \delta\mathbf{x}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\delta\mathbf{x}},\hat{\delta\mathbf{e}}} = \mathbf{0} \\
\mathbf{Q}_{\delta\mathbf{e}}^{+} &= \left.\frac{\partial \mathbf{q}^{+}}{\partial \delta\mathbf{e}^{\top}}\right|_{\hat{\mathcal{R}},\hat{\delta\mathbf{x}},\hat{\delta\mathbf{e}}} = \frac{1}{2}\mathbf{\Pi}(\hat{\mathbf{q}})
\end{aligned} \tag{5.4}$$

where $\mathbf{\Pi}$ is the already defined matrix

$$\mathbf{\Pi}(\mathbf{q}) = \begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix}.$$

If we are interested in the full Jacobians of $\mathcal{R}^{+} = f(\mathcal{R}, \mathbf{v})$ we just need to build them from the previous results:

$$\mathbf{F}_{\mathcal{R}} = \begin{bmatrix} \mathbf{I} & \mathbf{F}\mathbf{F}_{\mathbf{q}} \\ \mathbf{0} & \mathbf{I} + \frac{1}{2}\mathbf{\Omega} \end{bmatrix} \qquad \mathbf{F}_{\mathbf{v}} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{\Pi} \end{bmatrix}. \tag{5.5}$$

5.1.2 Forward motion and Euler increments in robot frame

For nonholonomic robots this is an interesting representation because it contains only a forward motion and the orientation increments, leading to an odometry vector of only four components. For a reference frame with the x axis looking forward we have

$$\mathbf{v} = \begin{bmatrix} \delta x \\ \delta\mathbf{e} \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix}$$

where δx is the forward motion and $\delta\mathbf{e}$ is the increment in the robot orientation expressed in Euler angles, all in $\mathcal{R}$ frame. The evolution function

is

$$\begin{aligned}\mathbf{x}^+ &= \mathbf{x} + \mathbf{R}(\mathbf{q} + \frac{1}{4}\mathbf{\Omega}(\delta\mathbf{e}) \cdot \mathbf{q}) \cdot \begin{bmatrix} \delta x \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2}\mathbf{\Omega}(\delta\mathbf{e}) \cdot \mathbf{q}\end{aligned} \tag{5.6}$$

where $\mathbf{R}(\mathbf{q} + \frac{1}{4}\mathbf{\Omega}(\delta\mathbf{e}) \cdot \mathbf{q})$ is the rotation matrix corresponding to the robot orientation $\mathbf{q}$ plus one half of the orientation increment $\delta\mathbf{e}$. This corresponds to uniformly distributing the full rotation $\delta\mathbf{e}$ over the full displacement δx.

Jacobians are quite more complex than in the previous case and are not given.

5.2 Dynamic models

We propose the kinematics functions for more complete dynamic models, specially for constant-velocity models or constant-acceleration models.

5.2.1 Constant velocity dynamic model in world frame

The evolution model for the robot state vector $\mathcal{R}^\top = [\mathbf{x}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \omega^\top]$ is written as

$$\begin{aligned}\mathbf{x}^+ &= \mathbf{x} + T_s \cdot \mathbf{v} \\ \mathbf{v}^+ &= \mathbf{v} + \upsilon_{\mathbf{v}} \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2}T_s\mathbf{\Omega}(\omega) \cdot \mathbf{q} \\ \omega^+ &= \omega + \upsilon_\omega\end{aligned} \tag{5.7}$$

where T_s is the sampling time, $\mathbf{x}$ and $\mathbf{q}$ are the position and the orientation, $\mathbf{v}$ and ω are the linear and the angular velocities, and $\upsilon_{\mathbf{v}}$ and υ_ω are independent white gaussian noises with covariances matrices $\mathbf{U}_{\mathbf{v}}$ and $\mathbf{U}_\omega$. All is expressed in world frame except angular velocity ω which is in robot frame.

The full Jacobian is easily obtained from the results of the previous sections

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & T_s\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} + \frac{1}{2}T_s\mathbf{\Omega}(\omega) & \frac{1}{2}T_s\mathbf{\Pi}(\mathbf{q}) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5.8}$$

where $\mathbf{\Omega}(\omega)$ and $\mathbf{\Pi}(\mathbf{q})$ are already defined.

5.2.2 Constant velocity dynamic model in robot frame

The evolution model for the robot state vector $\mathcal{R}^\top = [\mathbf{x}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \omega^\top]$ is written as

$$\begin{aligned} \mathbf{x}^+ &= \mathbf{x} + T_s \mathbf{R}(\mathbf{q}) \cdot \mathbf{v} \\ \mathbf{v}^+ &= \mathbf{v} + \upsilon_{\mathbf{v}} \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2} T_s \boldsymbol{\Omega}(\omega) \cdot \mathbf{q} \\ \omega^+ &= \omega + \upsilon_\omega \end{aligned} \tag{5.9}$$

where $\mathbf{x}$ and $\mathbf{q}$ are position and orientation, $\mathbf{v}$ and ω are the linear and angular velocities respectively, and $\upsilon_{\mathbf{v}}$ and υ_ω are independent white gaussian noises with covariances matrices $\mathbf{U}_{\mathbf{v}}$ and $\mathbf{U}_\omega$. Position and orientation are expressed in world frame while linear and angular velocities are in robot frame.

The full Jacobian is easily obtained from the results of the previous sections

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & T_s \mathbf{R}(\mathbf{q}) & \mathbf{F}\mathbf{F}_{\mathbf{q}}(T_s \mathbf{v}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} + \frac{1}{2} T_s \boldsymbol{\Omega}(\omega) & \frac{1}{2} \boldsymbol{\Pi}(\mathbf{q}) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{5.10}$$

where $\mathbf{R}(\mathbf{q})$, $\mathbf{F}\mathbf{F}_{\mathbf{q}}(T_s \mathbf{v})$, $\boldsymbol{\Omega}(\omega)$ and $\boldsymbol{\Pi}(\mathbf{q})$ are already defined.

Chapter 6

VSLAM: Vision based Simultaneous Localization And Mapping

With the algebra presented in all the previous sections we are in position of writing all the necessary equations for a 3D Bearing-Only SLAM system based on vision, local linearizations and Gaussian distributions:

- EKF-VSLAM, where the *Extended Kalman Filter* is directly applied. A good *a priori* estimation of the distance to the landmarks is required;
- GSF-VSLAM, where the *Gaussian Sum Filter* is used to deal with the multi-hypothesis representing the *a priori* estimate of the distance to the landmarks;
- FIS-VSLAM, where the *Federated Information Sharing* technique is used to boost the unsatisfactory performances of GSF; and
- BU-VSLAM, where EKF is used and a *Batch Update* is performed for a delayed initialization of landmarks.

6.1 EKF-VSLAM

6.1.1 Map representation

A Gaussian random vector including robot state and landmark positions will be our map:

$$X = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{q} \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{bmatrix} \tag{6.1}$$

where $\mathcal{R}^\top = \begin{bmatrix} \mathbf{x}^\top & \mathbf{q}^\top \end{bmatrix}$ is the robot state vector containing position and orientation and $\mathcal{M}^\top = \begin{bmatrix} \mathbf{p}_1^\top & \cdots & \mathbf{p}_n^\top \end{bmatrix}$ is the set of landmark positions. This map being Gaussian, its *pdf* is fully specified by providing its mean and its covariances matrix

$$\hat{X} = \begin{bmatrix} \hat{\mathcal{R}} \\ \hat{\mathcal{M}} \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{M}} \\ \mathbf{P}_{\mathcal{M}\mathcal{R}} & \mathbf{P}_{\mathcal{M}\mathcal{M}} \end{bmatrix} \tag{6.2}$$

where $\mathbf{P_{ab}} \triangleq E\big[(\mathbf{a} - \hat{\mathbf{a}})(\mathbf{b} - \hat{\mathbf{b}})^\top\big]$ for any $\mathbf{a}$ and $\mathbf{b}$. For example:

- $\mathbf{P}_{\mathcal{R}\mathcal{M}}$ is the cross-variance between $\mathcal{R}$ and all landmarks positions $\mathcal{M}$;
- $\mathbf{P}_{\mathcal{R}i}$ is the cross-variance between $\mathcal{R}$ and landmark $\mathbf{p}_i$;
- $\mathbf{P}_{ii}$ is the covariance of landmark $\mathbf{p}_i$;
- $\mathbf{P}_{iX}$ is the row-band of $\mathbf{P}$ corresponding to landmark $\mathbf{p}_i$; and
- $\mathbf{P}_{X\mathcal{R}}$ is the column-band of $\mathbf{P}$ corresponding to robot pose $\mathcal{R}$.

The objective of a SLAM system is to keep this *pdf* up to date when any of the following situations occur:

1. The robot moves;
2. The robot observes a landmark already existing in the map; and
3. The robot observes a landmark that is not in the map and decides to incorporate it.

The necessary algebra for all three cases is detailed below.

6.1.2 Robot motion

Robot motion is described by the function

$$\mathcal{R}^+ = f(\mathcal{R}, \mathbf{v}) \tag{6.3}$$

which has already been detailed in section 5.1.1. Map updates correspond to the EKF prediction equations:

$$\begin{aligned}
\hat{\mathcal{R}}^+ &= f(\hat{\mathcal{R}}, \hat{\mathbf{v}}) \\
\hat{\mathcal{M}}^+ &= \hat{\mathcal{M}} \\
\mathbf{P}^+_{\mathcal{R}\mathcal{R}} &= \mathbf{F}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{R}}\mathbf{F}_{\mathcal{R}}^\top + \mathbf{F}_{\mathbf{v}}\mathbf{V}\mathbf{F}_{\mathbf{v}}^\top \\
\mathbf{P}^+_{\mathcal{R}\mathcal{M}} &= \mathbf{F}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{M}} \\
\mathbf{P}^+_{\mathcal{M}\mathcal{M}} &= \mathbf{P}_{\mathcal{M}\mathcal{M}}
\end{aligned} \tag{6.4}$$

where $\mathbf{F}_{\mathcal{R}}$ and $\mathbf{F}_{\mathbf{v}}$ are the Jacobians of f defined also in section 5.1.1.

6.1.3 Landmark observations

Observation of landmark $\mathbf{p}_i$ is defined by the function

$$\begin{aligned}
\mathbf{u}_i &= h(\mathcal{R}, \mathcal{C}, \mathbf{p}_i, \mathbf{c}) \\
\mathbf{y}_i &= \mathbf{u}_i + \upsilon
\end{aligned} \tag{6.5}$$

which has already been detailed in section 4.2.1. Map updates correspond to the EKF correction equations:

$$\begin{aligned}
\mathbf{Z}_i &= \mathbf{H}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{H}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}i}\mathbf{H}_i^\top + \mathbf{H}_i\mathbf{P}_{i\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{H}_i\mathbf{P}_{ii}\mathbf{H}_i^\top + \mathbf{R} \\
\mathbf{K}_i &= (\mathbf{P}_{X\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{P}_{Xi}\mathbf{H}_i^\top) \cdot \mathbf{Z}_i^{-1} \\
\hat{X}^+ &= \hat{X} + \mathbf{K}_i \cdot (\mathbf{y}_i - h(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_i, \mathbf{c})) \\
\mathbf{P}^+ &= \mathbf{P} - \mathbf{K}_i\mathbf{Z}_i\mathbf{K}_i^\top
\end{aligned} \tag{6.6}$$

where $\mathbf{H}_i = \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_i, \mathbf{c})$.

6.1.4 Landmark initializations

The observation of a new landmark $\mathbf{p}_p$ is defined by the same function as before:

$$\begin{aligned}
\mathbf{u}_p &= h(\mathcal{R}, \mathcal{C}, \mathbf{p}_p, \mathbf{c}) \\
\mathbf{y}_p &= \mathbf{u}_p + \upsilon
\end{aligned} \tag{6.7}$$

In the initialization procedure we are interested in finding the *pdf* of the augmented map

$$X^+ = \begin{bmatrix} X \\ \mathbf{p}_p \end{bmatrix} \tag{6.8}$$

conditioned to observation $\mathbf{y}_p$. We need for that the inverse observation function

$$\mathbf{p}_p = g(\mathcal{R}, \mathcal{C}, \mathbf{u}_p, s, \mathbf{c}) \tag{6.9}$$

and its Jacobians $\mathbf{G}_\mathcal{R}$, $\mathbf{G}_\mathbf{u}$ and $\mathbf{G}_s$, all defined in section 4.2.2.

For EKF to work, it is necessary to have a good Gaussian estimate of the unmeasured range s, in the form $s \sim \mathcal{N}\{\hat{s}; \sigma_s^2\}$, *good* meaning its standard deviation being sufficiently small (it is widely accepted that the worst case should be $\sigma_s \leq 0.3\hat{s}$). Assuming this estimate is available, we can write

$$\begin{aligned} \hat{\mathbf{p}}_p &= g(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, \hat{s}, \mathbf{c}) \\ \mathbf{P}_{pp} &= \mathbf{G}_\mathcal{R} \mathbf{P}_{\mathcal{R}\mathcal{R}} \mathbf{G}_\mathcal{R}^\top + \mathbf{G}_\mathbf{u} \mathbf{R} \mathbf{G}_\mathbf{u}^\top + \mathbf{G}_s \sigma_s^2 \mathbf{G}_s^\top \\ \mathbf{P}_{pX} &= \mathbf{G}_\mathcal{R} \mathbf{P}_{\mathcal{R}X} \end{aligned} \tag{6.10}$$

The updated map is then the result of augmenting the previous map as

$$\hat{X}^+ = \begin{bmatrix} \hat{X} \\ \hat{\mathbf{p}}_p \end{bmatrix} \quad \mathbf{P}^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}_{pX}^\top \\ \mathbf{P}_{pX} & \mathbf{P}_{pp} \end{bmatrix}. \tag{6.11}$$

6.2 GSF-VSLAM

GSF-VSLAM tries to solve the limitations of EKF-VSLAM in landmark initialization when the *a priori* of s is not small enough –which is normally the case. The different situations are treated here in a different order to easy up their comprehension:

1. The robot observes a new landmark and decides to incorporate it in the map;
2. The robot observes a landmark already existing in the map; and
3. The robot moves.

6.2.1 Landmark initialization

A multi-hypothesis system based on a sum of Gaussian maps is generated at each new *landmark initialization*. Each Gaussian map holds an hypothesis for the landmark position, and only one among them is the good one. If landmark initialization occurs at a time when multiple hypothesis already exist, a whole set of hypothesis is generated for *each one* of the existing maps. One can say that every landmark initialization multiplies the number of maps by the number of terms in the Gaussian sum. For this reason, landmark initialization is discouraged unless the current number of hypothesis is very small –for example, only one.

As the range s cannot be characterised by a single Gaussian, the best option is to approximate its *pdf* with a sum of Gaussians. To minimise the number of terms while keeping EKF linearization constraints, the terms of this sum must follow a geometric series:

$$p(s) = \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(s - \beta^{j-1} s_1, (\beta^{j-1}\sigma_1)^2) \tag{6.12}$$

with

$$\begin{aligned} 0.15 \leq \alpha \leq 0.3 \\ 1.5 \leq \beta \leq 3 \\ s_1 = (1-\alpha)^{-1} \cdot s_{min} \\ \sigma_1 = \alpha \cdot s_1 \\ N_g = 1 + \text{ceil}\left[\log_\beta \left(\frac{1-\alpha}{1+\alpha} \cdot \frac{s_{max}}{s_{min}}\right)\right] \end{aligned} \tag{6.13}$$

where $\text{ceil}(x)$ is the next integer to x.

The *pdf* of the augmented map $X^+ = [X^\top \ \mathbf{p}_p^\top]^\top$ conditioned to observation $\mathbf{y}_p$ becomes then a weighted sum of Gaussian maps

$$p(X^+|\mathbf{y}_p) = \sum_{j=1}^{N_g} \Lambda_j \cdot \mathcal{N}(X^+ - \hat{X}_j; \mathbf{P}_j) \tag{6.14}$$

where each weighted Gaussian map X_j is characterised by three parameters

$$X_j \sim \mathcal{N}\{\Lambda_j; \hat{X}_j; \mathbf{P}_j\}. \tag{6.15}$$

To compute each map hypothesis we need to perform several operations. First compute the hypothesis-dependent mean and standard deviation for the range s

$$\begin{aligned} s_j &= \beta^{(j-1)} s_1 \\ \sigma_j &= \beta^{(j-1)} \sigma_1 \end{aligned} \tag{6.16}$$

then the Jacobians of the inverse observation function $\mathbf{p}_p = g(\mathcal{R}, \mathcal{C}, \mathbf{u}_p, s, \mathbf{c})$ defined in section 4.2.2

$$\begin{aligned} \mathbf{G}^j_{\mathcal{R}} &= \mathbf{G}_{\mathcal{R}}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{G}^j_{\mathbf{u}} &= \mathbf{G}_{\mathbf{u}}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{G}^j_{s} &= \mathbf{G}_{s}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \end{aligned} \tag{6.17}$$

then the statistics of the hypothetic landmark

$$\begin{aligned} \hat{\mathbf{x}}^j_p &= g(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{P}^j_{pX} &= \mathbf{G}^j_{\mathcal{R}} \mathbf{P}_{\mathcal{R}X} \\ \mathbf{P}^j_{pp} &= \mathbf{G}^j_{\mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{R}} \mathbf{G}^{j\top}_{\mathcal{R}} + \mathbf{G}^j_{\mathbf{u}} \mathbf{R} \mathbf{G}^{j\top}_{\mathbf{u}} + \mathbf{G}^j_{s} \sigma_j^2 \mathbf{G}^{j\top}_{s} \end{aligned} \tag{6.18}$$

and finally construct each new map in the sum (6.14) with

$$\Lambda_j = 1/N_g \qquad \hat{X}_j = \begin{bmatrix} \hat{X} \\ \hat{\mathbf{x}}^j_p \end{bmatrix} \quad \mathbf{P}_j = \begin{bmatrix} \mathbf{P} & \mathbf{P}^{j\top}_{pX} \\ \mathbf{P}^j_{pX} & \mathbf{P}^j_{pp} \end{bmatrix} \tag{6.19}$$

6.2.2 Landmark observations

Landmark observations are processed in three steps: First, likelihoods are calculated to update the weight of each map. Second, unlikely maps are pruned. Third, EKF-VSLAM corrections are applied to each one of the survivors.

Likelihood computation

To compute the likelihood λ^i_j of an hypothetic map X_j given an observation $\mathbf{y}_i$ we need to perform several operations. First compute the Jacobians

$$\begin{aligned} \mathbf{H}^j_{\mathcal{R}} &= \mathbf{H}_{\mathcal{R}}(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}^j_i, \mathbf{c}) \\ \mathbf{H}^j_{i} &= \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}^j_i, \mathbf{c}) \end{aligned} \tag{6.20}$$

then the innovation and its covariances matrix

$$\begin{aligned}\mathbf{z}_i^j &= \mathbf{y}_i - h(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}_i^j, \mathbf{c}) \\ \mathbf{Z}_i^j &= \mathbf{H}_{\mathcal{R}}^j \mathbf{P}_{\mathcal{R}\mathcal{R}}^j \mathbf{H}_{\mathcal{R}}^{j\top} + \mathbf{H}_{\mathcal{R}}^j \mathbf{P}_{\mathcal{R}i}^j \mathbf{H}_i^{j\top} + \mathbf{H}_i^j \mathbf{P}_{i\mathcal{R}}^j \mathbf{H}_{\mathcal{R}}^{j\top} + \mathbf{H}_i^j \mathbf{P}_{ii}^j \mathbf{H}_i^{j\top} + \mathbf{R}\end{aligned} \tag{6.21}$$

The likelihood is finally

$$\lambda_j^i = \mathcal{N}(\mathbf{z}_i^j; \mathbf{Z}_i^j) = \frac{1}{2\pi\sqrt{|\mathbf{Z}_i^j|}} \exp\Big\{ -\frac{1}{2} \mathbf{z}_i^{j\top} (\mathbf{Z}_i^j)^{-1} \mathbf{z}_i^j \Big\}. \tag{6.22}$$

The likelihood λ_j of the hypothesis X_j given all the observations $\{\mathbf{y}_i, 1 \leq i \leq N\}$ at time t is then the product of the likelihoods given each one of these observations

$$\lambda_j = \prod_{i=1}^{N} \lambda_j^i \tag{6.23}$$

which are normalised so that they sum up to unity

$$\lambda_j^+ = \frac{\lambda_j}{\sum_k \lambda_k} \tag{6.24}$$

Finally we update the weight of each hypothesis with this likelihood, leading to what we call the Aggregated Likelihood (AL)

$$\Lambda_j^+ = \Lambda_j \cdot \lambda_j \tag{6.25}$$

We will need to store the Jacobians $\mathbf{H}_{\mathcal{R}}^j$ and $\mathbf{H}_i^j$ and the innovation covariances matrix $\mathbf{Z}_i^j$ for later use.

Maps pruning

This weight or Aggregated Likelihood is, at the end of the day, a measure of probability. It is the probability of the hypothesis X_j being *true* given all the accumulated observations. It is natural to define a pruning criteria in the following way:

An hypothetic map X_j is deleted or pruned if its Aggregated Likelihood Λ_j drops below a predefined threshold:

$$\Lambda_j < \tau/N \Rightarrow \text{prune } X_j \tag{6.26}$$

where N is the actual number of maps and $0.0001 \leq \tau \leq 0.01$, typically $\tau = 0.001$.

Maps corrections

Subsisting maps are corrected using the standard EKF-VSLAM procedures. For each map, first we need to recuperate the previously stored Jacobians and the covariances matrix of the innovation

$$\begin{aligned}
\mathbf{H}^j_{\mathcal{R}} &= \mathbf{H}_{\mathcal{R}}(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}^j_i, \mathbf{c}) \\
\mathbf{H}^j_i &= \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}^j_i, \mathbf{c}) \\
\mathbf{Z}^j_i &= \mathbf{H}^j_{\mathcal{R}}\mathbf{P}^j_{\mathcal{R}\mathcal{R}}\mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{H}^j_{\mathcal{R}}\mathbf{P}^j_{\mathcal{R}i}\mathbf{H}^{j\top}_i + \mathbf{H}^j_i\mathbf{P}^j_{i\mathcal{R}}\mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{H}^j_i\mathbf{P}^j_{ii}\mathbf{H}^{j\top}_i + \mathbf{R}
\end{aligned} \tag{6.27}$$

and then apply the correction equations to get the updated maps

$$\begin{aligned}
\mathbf{K}^j_i &= (\mathbf{P}^j_{X\mathcal{R}}\mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{P}^j_{Xi}\mathbf{H}^{j\top}_i) \cdot \mathbf{Z}^{j-1}_i \\
\hat{X}^+_j &= \hat{X}_j + \mathbf{K}^j_i \cdot (\mathbf{y}_i - h(\hat{\mathcal{R}}_j, \mathcal{C}, \hat{\mathbf{p}}^j_i, \mathbf{c})) \\
\mathbf{P}^+_j &= \mathbf{P}_j - \mathbf{K}^j_i\mathbf{Z}^j_i\mathbf{K}^{j\top}_i
\end{aligned} \tag{6.28}$$

6.2.3 Robot motion

Robot motion gets essentially the same treatment as in EKF-VSLAM except that multiple maps need to be updated. For each hypothetic map X_j, first compute the Jacobians

$$\begin{aligned}
\mathbf{F}^j_{\mathcal{R}} &= \mathbf{F}_{\mathcal{R}}(\hat{\mathcal{R}}_j, \hat{\mathbf{v}}) \\
\mathbf{F}^j_{\mathbf{v}} &= \mathbf{F}_{\mathbf{v}}(\hat{\mathcal{R}}_j, \hat{\mathbf{v}})
\end{aligned} \tag{6.29}$$

then apply EKF prediction equations

$$\begin{aligned}
\hat{\mathcal{R}}^+_j &= f(\hat{\mathcal{R}}_j, \hat{\mathbf{v}}) \\
\hat{\mathcal{M}}^+_j &= \hat{\mathcal{M}}_j \\
\mathbf{P}^{j+}_{\mathcal{R}\mathcal{R}} &= \mathbf{F}^j_{\mathcal{R}}\mathbf{P}^j_{\mathcal{R}\mathcal{R}}\mathbf{F}^{j\top}_{\mathcal{R}} + \mathbf{F}^j_{\mathbf{v}}\mathbf{V}\mathbf{F}^{j\top}_{\mathbf{v}} \\
\mathbf{P}^{j+}_{\mathcal{R}\mathcal{M}} &= \mathbf{F}^j_{\mathcal{R}}\mathbf{P}^j_{\mathcal{R}\mathcal{M}} \\
\mathbf{P}^{j+}_{\mathcal{M}\mathcal{M}} &= \mathbf{P}^j_{\mathcal{M}\mathcal{M}}
\end{aligned} \tag{6.30}$$

6.3 FIS-VSLAM

FIS-VSLAM is an approximated method to overcome the limitations of GSF-VSLAM in landmark initialization. As it has been pointed, GSF-VSLAM

multiplies the number of maps each time a new landmark is to be initialized. This is because of the multi-Gaussian characterization of the *pdf* of the range s, which naturally leads to a multi-Gaussian characterization for the whole map.

In FIS-VSLAM, the same geometric sum of Gaussians is used to characterize the range s. But this multi-Gaussian is interpreted as the existence of different hypothetic landmarks along the ray in the direction to the perceived landmark, and so they can all be included in a single Gaussian map. The set of all hypothesis for a particular landmark is called a *ray*. Subsequent observations will try to detect and prune this ray's wrong members, while keeping the rest of them and the map up to date.

It comes out that we have to distinguish two types of landmarks in our map:

- Landmarks which are represented by a single Gaussian
- Landmarks which are represented by a ray, the set of Gaussians.

As expected, FIS-VSLAM has the objective of keeping the map up to date when the following situations occur:

1. The robot observes a new landmark and decides to incorporate it in the map;
2. The robot observes a landmark already existing in the map in the form of a single Gaussian;
3. The robot observes a landmark already existing in the map in the form of a ray; and
4. The robot moves.

6.3.1 Landmark initialization

The idea is simply to initialize as many landmarks in the map as hypothesis (or terms in the Gaussian sum) are in the characterization of s, for which we take the geometric sum of Gaussians introduced in the previous section. The result is a single Gaussian map that has grown in an additive way, avoiding the multiplicative effect of GSF-VSLAM, and therefore allowing simultaneous initializations of multiple landmarks.

To compute the set of hypothesis or *ray* corresponding to a landmark we need to perform several operations. First compute the hypothesis-dependent mean and standard deviation for the range s

$$\begin{aligned} s_j &= \beta^{(j-1)} s_1 \\ \sigma_j &= \beta^{(j-1)} \sigma_1 \end{aligned} \tag{6.31}$$

then the Jacobians of the inverse observation function $\mathbf{p}_p = g(\mathcal{R}, \mathcal{C}, \mathbf{u}_p, s, \mathbf{c})$ defined in section 4.2.2

$$\begin{aligned} \mathbf{G}^j_{\mathcal{R}} &= \mathbf{G}_{\mathcal{R}}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{G}^j_{\mathbf{u}} &= \mathbf{G}_{\mathbf{u}}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{G}^j_{s} &= \mathbf{G}_{s}(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \end{aligned} \tag{6.32}$$

then the statistics of the hypothetic landmark

$$\begin{aligned} \hat{\mathbf{x}}^j_p &= g(\hat{\mathcal{R}}, \mathcal{C}, \mathbf{y}_p, s_j, \mathbf{c}) \\ \mathbf{P}^j_{pX} &= \mathbf{G}^j_{\mathcal{R}} \mathbf{P}_{\mathcal{R}X} \\ \mathbf{P}^j_{pp} &= \mathbf{G}^j_{\mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{R}} \mathbf{G}^{j\top}_{\mathcal{R}} + \mathbf{G}^j_{\mathbf{u}} \mathbf{R} \mathbf{G}^{j\top}_{\mathbf{u}} + \mathbf{G}^j_{s} \sigma_j^2 \mathbf{G}^{j\top}_{s} \end{aligned} \tag{6.33}$$

and finally augment the current map with it

$$\hat{X}^+ = \begin{bmatrix} \hat{X} \\ \hat{\mathbf{x}}^j_p \end{bmatrix} \quad \mathbf{P}^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}^{j\top}_{pX} \\ \mathbf{P}^j_{pX} & \mathbf{P}^j_{pp} \end{bmatrix} \tag{6.34}$$

At the same time, initialize the AL of the hypothesis j

$$\Lambda_j = 1/N_g \tag{6.35}$$

Repeating this procedure for all hypothesis we end up with the following expression for the map after landmark initialization:

$$\hat{X}^+ = \begin{bmatrix} \hat{X} \\ \hat{\mathbf{x}}^1_p \\ \vdots \\ \hat{\mathbf{x}}^{N_g}_p \end{bmatrix} \quad \mathbf{P}^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}^{1\top}_{pX} & \cdots & \mathbf{P}^{N_g\top}_{pX} \\ \mathbf{P}^1_{pX} & \mathbf{P}^1_{pp} & & \\ \vdots & & \ddots & \\ \mathbf{P}^{N_g}_{pX} & & & \mathbf{P}^{N_g}_{pp} \end{bmatrix} \quad \Lambda_p = \begin{bmatrix} 1/N_g \\ \vdots \\ 1/N_g \end{bmatrix} \tag{6.36}$$

6.3.2 Gaussian landmarks observations

For the landmarks $\mathbf{p}$ that are in the map under the form of a single gaussian, we just have to perform EKF-VSLAM corrections:

$$\begin{aligned}
\mathbf{Z}_p &= \mathbf{H}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{H}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}p}\mathbf{H}_p^\top + \mathbf{H}_p\mathbf{P}_{p\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{H}_p\mathbf{P}_{pp}\mathbf{H}_p^\top + \mathbf{R} \\
\mathbf{K}_p &= (\mathbf{P}_{X\mathcal{R}}\mathbf{H}_{\mathcal{R}}^\top + \mathbf{P}_{Xp}\mathbf{H}_p^\top) \cdot \mathbf{Z}_p^{-1} \\
\mathbf{P}^+ &= \mathbf{P} - \mathbf{K}_p\mathbf{Z}_p\mathbf{K}_p^\top \\
\hat{X}^+ &= \hat{X} + \mathbf{K}_p \cdot (\mathbf{y}_p - h(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}, \mathbf{c}))
\end{aligned} \tag{6.37}$$

where $\mathbf{H}_p = \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}, \mathbf{c})$.

6.3.3 Ray landmarks observations

For the landmarks $\mathbf{p}$ in the map that are in the form of a ray –the sum of Gaussians, and as in the GSF-VSLAM, observations are treated in a three-step procedure: likelihood computation, pruning and correction.

Likelihood computation

Likelihood computation is lighter than in GSF-VSLAM because all information is in a unique map. We just need to compute the likelihood λ_j of the landmark hypothesis $\mathbf{p}_j$ being *true* given the current observation $\{\mathbf{y}_p; \mathbf{R}\}$, and the Aggregated Likelihood Λ_j. We use the Jacobians

$$\begin{aligned}
\mathbf{H}_{\mathcal{R}}^j &= \mathbf{H}_{\mathcal{R}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c}) \\
\mathbf{H}_p^j &= \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c})
\end{aligned} \tag{6.38}$$

to compute the innovation's mean and covariances matrix

$$\begin{aligned}
\mathbf{z}_j &= \mathbf{y}_p - h(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c}) \\
\mathbf{Z}_j &= \mathbf{H}_{\mathcal{R}}^j\mathbf{P}_{\mathcal{R}\mathcal{R}}^j\mathbf{H}_{\mathcal{R}}^{j\top} + \mathbf{H}_{\mathcal{R}}^j\mathbf{P}_{\mathcal{R}p}^j\mathbf{H}_p^{j\top} + \mathbf{H}_p^j\mathbf{P}_{p\mathcal{R}}^j\mathbf{H}_{\mathcal{R}}^{j\top} + \mathbf{H}_p^j\mathbf{P}_{pp}^j\mathbf{H}_p^{j\top} + \mathbf{R}
\end{aligned} \tag{6.39}$$

with which we get the likelihood

$$\lambda_j = \mathcal{N}(\mathbf{z}_j; \mathbf{Z}_j) = \frac{1}{2\pi\sqrt{|\mathbf{Z}_j|}} \exp\Big\{ -\frac{1}{2}\mathbf{z}_j^\top (\mathbf{Z}_j)^{-1}\mathbf{z}_j \Big\} \tag{6.40}$$

which is normalised so that the sum for all hypothesis is one

$$\lambda_j^+ = \frac{\lambda_j}{\sum_k \lambda_k} \tag{6.41}$$

and the Aggregated Likelihood

$$\Lambda_j^+ = \Lambda_j \cdot \lambda_j \tag{6.42}$$

which is also normalised

$$\Lambda_j^+ = \frac{\Lambda_j}{\sum_k \Lambda_k} \tag{6.43}$$

We will need to store the Jacobians $\mathbf{H}_{\mathcal{R}}^j$ and $\mathbf{H}_p^j$ and the likelihoods λ_j and Λ_j for later use.

Ray members pruning

The Aggregated Likelihood Λ_j is a measure of probability. It is the probability of $\mathbf{p}_j$ (the hypothesis j for landmark $\mathbf{p}$) being *true* given all the observations of this landmark. It is natural to define a pruning criteria in the way we did before.

An hypothetic landmark $\mathbf{p}_j$ is deleted or pruned if its Aggregated Likelihood Λ_j drops below a predefined threshold:

$$\Lambda_j < \tau/N \Rightarrow \text{prune } \mathbf{p}_j \tag{6.44}$$

where N is the actual number of hypothesis for $\mathbf{p}$ and $0.0001 \leq \tau \leq 0.01$, typically $\tau = 0.001$.

Map corrections

Corrections are performed following the Federated Information Sharing technique. First compute the federated coefficient for each hypothesis, whith a choice for likelihood or aggregated likelihood

$$\rho_j = \frac{\lambda_j}{\sum \lambda_j} \qquad or \qquad \rho_j = \frac{\Lambda_j}{\sum \Lambda_j} \tag{6.45}$$

then affect the noise covariances matrix $\mathbf{R}$ with it

$$\mathbf{R}_j = \mathbf{R}/\rho_j \tag{6.46}$$

then perform EKF corrections for all hypothesis $\mathbf{p}_j$. For that, recuperate the previously stored Jacobians

$$\begin{aligned} \mathbf{H}^j_{\mathcal{R}} &= \mathbf{H}_{\mathcal{R}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c}) \\ \mathbf{H}^j_{p} &= \mathbf{H}_{\mathbf{p}}(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c}) \end{aligned} \tag{6.47}$$

recompute the covariances matrix of the innovation

$$\mathbf{Z}_j = \mathbf{H}^j_{\mathcal{R}} \mathbf{P}^j_{\mathcal{R}\mathcal{R}} \mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{H}^j_{\mathcal{R}} \mathbf{P}^j_{\mathcal{R}p} \mathbf{H}^{j\top}_{p} + \mathbf{H}^j_{p} \mathbf{P}^j_{p\mathcal{R}} \mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{H}^j_{p} \mathbf{P}^j_{pp} \mathbf{H}^{j\top}_{p} + \mathbf{R}_j \tag{6.48}$$

and then apply the correction equations

$$\begin{aligned} \mathbf{K}_j &= (\mathbf{P}^j_{X\mathcal{R}} \mathbf{H}^{j\top}_{\mathcal{R}} + \mathbf{P}^j_{Xi} \mathbf{H}^{j\top}_{i}) \cdot \mathbf{Z}^{j-1}_{i} \\ \hat{X}^{+} &= \hat{X} + \mathbf{K}_j \cdot (\mathbf{y}_p - h(\hat{\mathcal{R}}, \mathcal{C}, \hat{\mathbf{p}}_j, \mathbf{c})) \\ \mathbf{P}^{+} &= \mathbf{P} - \mathbf{K}_j \mathbf{Z}_j \mathbf{K}^{\top}_{j} \end{aligned} \tag{6.49}$$

6.3.4 Robot motion

Robot motion is described by the function

$$\mathcal{R}^{+} = f(\mathcal{R}, \mathbf{v}) \tag{6.50}$$

which has already been detailed in section 5.1.1. Map updates correspond to the EKF prediction equations:

$$\begin{aligned} \hat{\mathcal{R}}^{+} &= f(\hat{\mathcal{R}}, \hat{\mathbf{v}}) \\ \hat{\mathcal{M}}^{+} &= \hat{\mathcal{M}} \\ \mathbf{P}^{+}_{\mathcal{R}\mathcal{R}} &= \mathbf{F}_{\mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{R}} \mathbf{F}^{\top}_{\mathcal{R}} + \mathbf{F}_{\mathbf{v}} \mathbf{V} \mathbf{F}^{\top}_{\mathbf{v}} \\ \mathbf{P}^{+}_{\mathcal{R}\mathcal{M}} &= \mathbf{F}_{\mathcal{R}} \mathbf{P}_{\mathcal{R}\mathcal{M}} \\ \mathbf{P}^{+}_{\mathcal{M}\mathcal{M}} &= \mathbf{P}_{\mathcal{M}\mathcal{M}} \end{aligned} \tag{6.51}$$

where $\mathbf{F}_{\mathcal{R}}$ and $\mathbf{F}_{\mathbf{v}}$ are the Jacobians of f defined also in section 5.1.1.

6.4 BU-VSLAM

Batch Update VSLAM is the only delayed method presented here. Initialization is deferred until enough evidence on the landmark distance has been gained, so it is performed in the classical EKF-VSLAM way.

Chapter 7

Observability of moving objects from vision equipped moving platforms

7.1 Introduction

As we have seen so far, SLAM systems are capable of estimating both vehicle trajectory and world structure. It is now worth mentioning that one of the strongest hypothesis that has to be made to put SLAM systems into work is the fact that world landmarks are static. This hypothesis provides two important advantages: on one hand, landmark position estimates can be progressively refined; on the other hand, they can be used as external references for robot localization.

In this chapter we are going to break this hypothesis. We are going to introduce some moving objects in the perceived world, while still trying to solve the SLAM problem and, additionally, recover each moving object's dynamics.

In order to keep SLAM solvable, we will keep a significant amount of fixed landmarks. For the subsystem {*vehicle; fixed landmarks*} it is clear that the problem remains the same. For the subsystem {*vehicle; moving landmarks*}, new ideas will have to be developed and demonstrated in order to decide on problem solubility and to imagine the possible resolution methods.

This chapter is concerned about problem solubility which is strongly connected to system observability. The analysis of this observability will be per-

formed for different system configurations including sensing (mono or stereo vision systems), moving object's nature (single points, rigid objects or non rigid objects) and moving object's trajectories (constant speed, rotations, random motions).

7.2 General methodology

Roughly speaking, we want to know whether we can recover the 3D information of the perceived scenes from the information acquired by the cameras. This 3D information includes static world structure, camera motion, and moving objects structure and motion. If the whole 3D information is not observable, we want to know which parts are missing and under which conditions. There may be singular situations of non observability that could be avoided or treated as special cases.

Camera motion will be resumed to a series of n camera poses. The observability analysis will have to discover if the characteristics of this poses series (aligned poses, constant speed, planar or full 3D motions, etc) have an impact on the reconstructed scene.

Object motion will be defined by a dynamic model in state space. The observability analysis will have to discover if there are particular motions that are not observable.

Object structure will define the way object's points are arranged. The observability analysis will have to discover which is the minimum number of points to track per object and if they have to follow (or avoid) particular arrangements.

A full state vector will be constructed with all the data to estimate. An observation vector with all observed data. The nonlinear functions relating the observations to the state will have to be written, and their Jacobians calculated. A rank analysis of these Jacobians will tell us about the feasibility to locally invert the observation functions, and hence to recover the state from the observations.

7.3 Sensor and object properties

7.3.1 Sensor platforms

Single camera

We will consider a single camera at n consecutive positions

$$\begin{bmatrix}\mathcal{C}_1 & \cdots & \mathcal{C}_n\end{bmatrix} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_n\end{bmatrix}. \tag{7.1}$$

As a bearings-only sensor, camera orientation is not relevant for the observability analysis as long as considered objects and landmarks lie inside its field of view.

As usual, the standard Pin Hole camera model will be taken, with its calibration parameters defined as $\mathbf{c} = \begin{bmatrix}u_0 & v_0 & \alpha_u & \alpha_v\end{bmatrix}$. Without loss of generality, these parameters can be taken as the normalized calibration parameters given by $\mathbf{c} = \begin{bmatrix}0 & 0 & 1 & 1\end{bmatrix}$.

Stereo head

A stereo head of base b can be built with two identical cameras as the one above. They both look forward and are arranged at the left- and right- hand sides of the head's reference frame $\mathcal{H} : \{RDF\}$. In $\mathcal{H}$ frame, left hand camera is at $x = -b/2$ while right hand camera is at $x = b/2$. The head's orientation modifies the optical center's position of both cameras, and therefore we need to provide it in the world frame $\mathcal{W}$.

A series of n consecutive head poses is then written as

$$\begin{bmatrix}\mathcal{H}_1 & \cdots & \mathcal{H}_n\end{bmatrix} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_n \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n\end{bmatrix}. \tag{7.2}$$

At any given head pose $\mathcal{H}_i$, individual camera poses in frame $\mathcal{W}$ can be easily computed. Here the expression for the left hand camera is given:

$$\mathcal{C}_{li} = \begin{bmatrix} fromFrame\big(\mathcal{H}_i, \begin{bmatrix}-b/2 & 0 & 0\end{bmatrix}^\top\big) \\ \mathbf{q}_i \end{bmatrix}. \tag{7.3}$$

7.3.2 Object models

Single point object

The model of a single moving point is simply

$$\mathcal{O} = \begin{bmatrix}\mathbf{x}_o\end{bmatrix}. \tag{7.4}$$

Rigid object

A rigid object is here defined as a set of points that are rigidly linked together, defining a fixed structure. This structure $\mathcal{S}$ is a characteristic of the object and can be specified by all object's points coordinates in a certain object frame $\mathcal{O}$. We have:

$$\begin{aligned} \mathcal{S} &= \begin{bmatrix} \mathbf{o}_1 & \cdots & \mathbf{o}_n \end{bmatrix} \\ \mathcal{O} &= \begin{bmatrix} \mathbf{x}_o \\ \mathbf{q}_o \end{bmatrix}. \end{aligned} \tag{7.5}$$

A moving rigid object keeps the fixed structure but its frame varies with time giving $\mathcal{O} = \mathcal{O}(t)$.

Any object's point $\mathbf{o}_i$ can be expressed in the world frame $\mathcal{W}$ with the usual frame transformation function

$$\mathbf{o}_i^{\mathcal{W}} = fromFrame(\mathcal{O}, \mathbf{o}_i). \tag{7.6}$$

It can also be expressed in a camera frame $\mathcal{C}$ with

$$\mathbf{o}_i^{\mathcal{C}} = toFrame(\mathcal{C}, fromFrame(\mathcal{O}, \mathbf{o}_i)). \tag{7.7}$$

Non rigid object

We distinguish two kinds of non rigidity: chained rigid objects (skeletons or articulated artifacts) and soft objects.

For chained rigid objects, the problem of object detection and tracking becomes a set of problems of detecting and tracking individual rigid objects, with the addition of several constraints representing the joints. These constraints may add some observability to the whole system, but are not considered in this study. Hence the problem can be decoupled into a set of single rigid object problems.

For soft objects, the links between the points of the structure are no longer rigid. This represents a loss of an important constraint and hence an important loss in observability. Soft objects are not considered in this study.

7.3.3 Object motions

As only rigid objects are being considered, motions involve the trajectory of the object's reference frame in the form

$$\mathcal{O} = \mathcal{O}(t) = \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{q}_o(t) \end{bmatrix}. \tag{7.8}$$

$\mathcal{O}(t)$ represents the time-variant component of the object's state vector, the time-invariant part corresponding to the structure $\mathcal{S}$. It follows the random evolution equation

$$\mathcal{O}(t + \Delta t) = f(\mathcal{O}(t), \mathbf{v}(t)) \tag{7.9}$$

where $\mathbf{v}(t)$ is a random variable corresponding to all non modeled effects.

Constant linear and angular speeds

The evolution in time will generally be modeled with a constant speed model both for position and orientation, giving the state vector

$$\mathcal{O}(t) = \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{v}_o(t) \\ \mathbf{q}_o(t) \\ \omega_o(t) \end{bmatrix}. \tag{7.10}$$

Pure translation

A pure translation is characterized by a null rotation or constant orientation. This translates to

$$\mathcal{O}(t) = \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{v}_o(t) \\ \mathbf{q}_o \\ \mathbf{0} \end{bmatrix}. \tag{7.11}$$

As the object's structure is expressed in an arbitrary object's reference frame, one can choose for it $\mathbf{q}_o \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\top$, the null rotation, and omit it. This way an object following a pure translation can be defined by its (fixed) structure and its (time-varying) position in space:

$$\begin{aligned} \mathcal{S} &= \begin{bmatrix} \mathbf{o}_1 & \cdots & \mathbf{o}_n \end{bmatrix} \\ \mathcal{O}(t) &= \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{v}_o(t) \end{bmatrix}. \end{aligned} \tag{7.12}$$

Random motion

Constant speed motion models produce more or less smooth trajectories depending on the amount of noise introduced in the system. If the noise contribution to the dynamics is too important with respect to the model itself, the motion approaches the state of *random motion*, in which the object's position at any given time is weakly correlated to its past. In these cases predictions of the object positions will be poor and object tracking will become an endless task of detections and re-detections.

At the same time, the preponderance of the noisy term results in a severe relaxation of the constraints imposed by the motion model, lowering the system's observability.

Stop'n'Go motion

For the reasons exposed above, in cases where a single Gaussian noise is not representative of the motion's randomness, instead of increasing its covariance to accommodate all possible cases, it will be preferable to find other non Gaussian models.

For example, the motion of a pedestrian can be modeled with a constant speed model in the 2D plane. If we consider it cannot stop abruptly, a single, relatively small Gaussian will do. A larger Gaussian could be used to accommodate for sudden stops and goes, but that would reduce the motion's smoothness and eliminate the difference between the states *go* and *stop*.

The better suited Stop'n'Go model is a two hypotheses model with associated transition probabilities. If you know some Fuzzy logics, these kind of linguistic rules will sound familiar to you:

At any given time t:	if $\mathcal{O}$ is moving,	it probably keeps moving, but may eventually stop.
	if $\mathcal{O}$ is stopped,	it probably stays, but may eventually go.

At the satisfaction of any of these rules, different Gaussian noises will be assigned to $\mathbf{v}(t)$ and incorporated to the system. These rules are summarized in Table 7.1 below. We call P_S the probability of stopping while in motion and P_G the probability of starting moving. We add a Gaussaian noise to the system with covariance $\mathbf{Q}_M$ each time the pedestrian keeps moving; $\mathbf{Q}_S$

when it stops; and $\mathbf{Q}_G$ when it starts moving. No noise is added when it stays steady. Fuzzily speaking, P_S and P_G are small, $\mathbf{Q}_S$ is small, $\mathbf{Q}_M$ is quite small and $\mathbf{Q}_G$ is relatively large.

Table 7.1: Transition table for the Stop'n'Go motion model

current state	next state	probability	noise covariance
GO	GO	$1 - P_S$	$\mathbf{Q}_M$
GO	STOP	P_S	$\mathbf{Q}_S$
STOP	GO	P_G	$\mathbf{Q}_G$
STOP	STOP	$1 - P_G$	$\mathbf{0}$