mathmatical models from visual inertial slam

张志勇*

北京小鸟看看科技有限公司

May 12, 2017

1 abstract

本文将简单的介绍下visual inertial slam 的概况和使用到得算法。算法主要涉及到IMU的预积分理论和流形的理论。使用camera + IMU的方案来做SLAM/Odometry，一般被称作Visual-Inertial Odometry (VIO)或者Visual-Inertial Navigation System (VINS)，其实只有odometry还不能说是一个slam系统。按照David Scaramuzza的分类方法，首先分成filter-based 和 optimization based 的两大类。Stefan Leutenegger则则把visual inertial slam 分成tightly-coupled 和loosely-coupled，另外他还给了另外一分类方法”batch optimization and recursive filtering”。个人的理解就是按照是否将IMU数据加入到优化框架中为原则进行分类。

`state of the art visual inertial slam systems`

- manifold preintegration.
- keyframe-based visual-inertial slam using nonlinear opt。
- visual-inertial monocular slam with map reuse.
- A multi-state constraint Kalman filter for vision-aided inertial navigation.
- Direct Visual Inertial Odometry with stereo camera(IMU+LSD-SLAM).
- Robust Visual Inertial Odometry(有代码).
- Vision Based Navigation for Micro Helicopters (有代码).

2 short intro

loosely-coupled systems independently estimate the pose by a vision only algorithm and fuse IMU measurements only in a separate estimation step, limiting computational complexity. Tightly-coupled approaches in contrast include both the measurements from the IMU and the camera into a common problem where all states are jointly estimated, thus considering all correlations amongst them. the later appraoch promize high accuracy, leaving aside computation demands.

*email: hh_maizi@163.com

`loosely-coupled systems`

- large-scale visual odometry for rough terrain.
- factor graph based incremental smoothing in inertialo navigation systems.
- fast 3d pose estimation with out-of-sequence measurements.
- real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown enviroments.

`tightly-coupled systems`

okvis combines visual and inertial terms in a fully probabilistic manner.At the system level, we developed both the hardware and the algorithms for accurate real-time SLAM, including robust keypoint matching, bootstrapping and outlier rejection using inertial cues.

viorb visual-inertial monocular slam with map reuse.

msckf A multi-state constraint Kalman filter for vision-aided inertial navigation.

`filtering based`

- robust visual inertial odometry using a direct ekf-based approach.
- robust direct visual inertial odometry via entropy-based relative pose estimation.
- high-precision consistent ekf-based visual-inertial odometry.
- semi-direct ekf-based monocular visual-inertial odometry.

`optimization based`

- okvis
- IMU preintegration on manifold for efficient visual-inertial map estimation.
- visual-inertial navigation, mapping and localization: a scalable real-time approach.

3 okvis and vi-orb

3.1 vi-orb

个人感觉正如其论文题目”visual-inertial monocular slam with map reuse”, visual inertial orbslam的靓点应该在map reuse部分, 在IMU图优化方面只是借鉴了预积分和okvis的算法。visual inertial orb slam 的优化流程:

3.2 okvis

这个算法基本上可以说是imu加入到优化框架中较早的”state of the art”算法，而且开源。

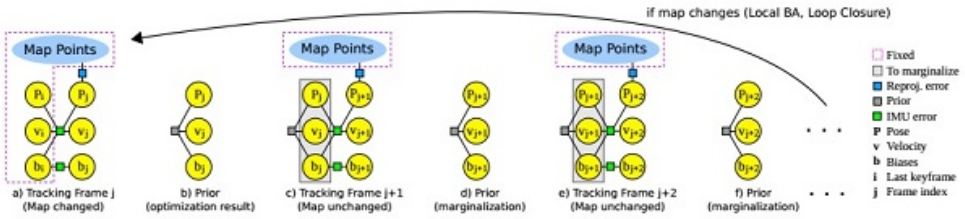

Fig. 2. Evolution of the optimization in the Tracking thread. (a) We start optimizing the frame j linked by an IMU constraint to last keyframe i. (b) The result of the optimization (estimation and Hessian matrix) serves as prior for next optimization. (c) When tracking next frame $j+1$, both frames j and $j+1$ are jointly optimized, being linked by an IMU constraint, and having frame j the prior from previous optimization. (d) At the end of the optimization, the frame j is marginalized out and the result serves as prior for following optimization. (e-f) This process is repeated until there is a map update from the Local Mapping or Loop Closing thread. In such case the optimization links the current frame to last keyframe discarding the prior, which is not valid after the map change.

Figure 1: visual inertial orbslam

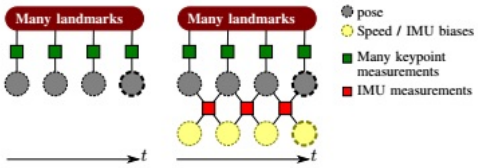

Fig. 3. Graphs of the state variables and measurements involved in the visual SLAM problem (left) versus visual-inertial SLAM (right): incorporating inertial measurements introduces temporal constraints, and necessitates a state augmentation by the robot speed as well as IMU biases.

Figure 2: okvis state graph

3.2.1 *okvis graph*

okvis 的图优化模型：

3.2.2 *initial state for marginalization*

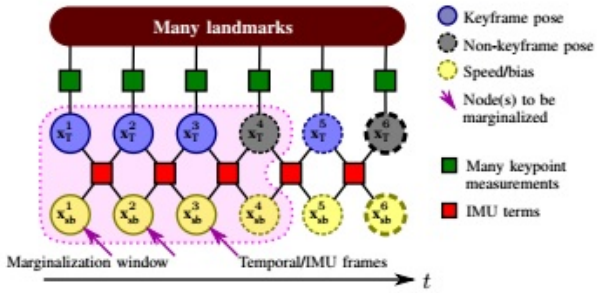

Figure 3: initial state for marginalization

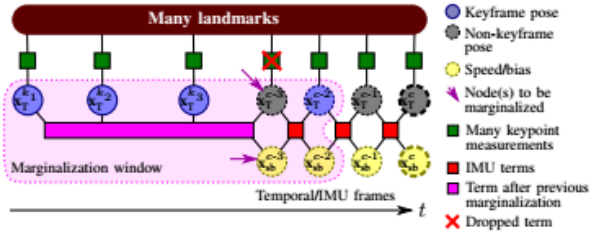

Figure 4: non-key-frame marginalization

3.2.3 *marginalization non key frame*

3.2.4 *marginalization key frame*

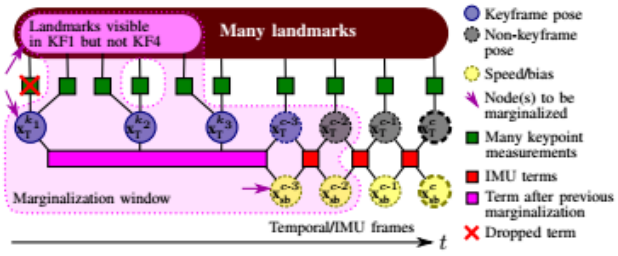

Figure 5: marginalization key-frame

4 math models from visual inertial slam

4.1 Notations

this is todo work.

4.2 Preliminaries

Rieman manifold The skew symmetric matrices:

$$\omega^{\wedge} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} 0, -\omega_3, \omega_2 \\ \omega_3, 0, -\omega_1 \\ -\omega_2, \omega_1, 0 \end{bmatrix} \qquad \in so(3) \tag{1}$$

wedge operator converts an vector to its corresponding matrix, and the vee operator convert the skew symmetric matrix to an vector. if we have $S = \omega^{\wedge}$, then

$$S^{\vee} = \omega \tag{2}$$

The exponential map(at identity): 指数映射是skew symmetric matrix 到 rotation matrix 的映射.

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

a first order exponential map:

$$exp(\phi^\wedge) \approx I + \phi^\wedge \tag{4}$$

logarithm map at the identity:

$$\log(R) = \frac{\varphi \cdot (R - R^T)}{2\sin(\varphi)} \qquad with\varphi = \arccos(\frac{tr(R)-1}{2}) \tag{5}$$

the vectorized versions of the exponential and logarithm map: 向量与SO(3)的映射.

$$Exp : R^3 \to SO(3); \Phi \rightarrowtail exp(\Phi^{wedge}) \tag{6a}$$
$$Log : SO(3) \to R^3; R \rightarrowtail log(R)^\vee \tag{6b}$$

Exp 的一阶估计:

$$Exp(\Phi + \delta\Phi) \approx Exp(\Phi)Exp(J_r(\Phi)\delta\Phi) \tag{7}$$

Log 的一阶估计:

$$Log(Exp(\Phi)Exp(\delta\Phi)) \approx \Phi + J_r^{-1}(\Phi)\delta\Phi \tag{8}$$

another useful property of exponential map:

$$RExp(\Phi)R^T = exp(R\Phi^\wedge R^T) = Exp(R\Phi) \tag{9a}$$
$$\Leftrightarrow Exp(\Phi)R = RExp(R^T\Phi) \tag{9b}$$

Uncertainty description in SO(3)

$$\tilde{R} = RExp(\epsilon), \quad \epsilon \sim \mathcal{N}(0, \Sigma) \tag{10}$$

the cost function:

$$\mathcal{L}(R) = \frac{1}{2}||Log(R^{-1}\tilde{R})||^2_\Sigma + const = \frac{1}{2}||Log(\tilde{R}^{-1})R||^2_\Sigma + const \tag{11}$$

this is geodesic uncertainty Σ^{-1}.

Gauss-Newton on Manifold move this paragraph to the supplementes.

4.3 MAP

Notations

model description the state:

$$x_i \doteq \left\{R_i, p_i, v_i, b_i\right\}. \tag{12}$$

Let $\mathcal{K}$ denote the set of all keyframes up to time k, the state of all keyframes:

$$\mathcal{X}_k \doteq \{x_i\}_{i\in\mathcal{K}_k} \tag{13}$$

the measurements:

$$\mathcal{Z} \doteq \{\mathcal{C}_i, \mathcal{I}_{ij}\}_{(i,j)\in\mathcal{K}_k} \tag{14}$$

factor graphs and map estimation: the psoterior probability of the variables $\mathcal{X}_k$:

$$p(\mathcal{X}_k|\mathcal{Z}_k) \propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k) = p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_k} p(\mathcal{C}_i, \mathcal{I}_{ij}|\mathcal{X}_k) \tag{15a}$$

$$= p(\mathcal{X}_0) \prod_{(i,j)\in\mathcal{K}_k} p(\mathcal{I}_{ij}|x_i, x_j) \prod_{i\in\mathcal{K}_k} \prod_{l\in\mathcal{C}_i} p(z_{il}|x_i) \tag{15b}$$

finally the map estimate:

$$\mathcal{X}_k^* \doteq argmin_{\mathcal{X}_k} - log_e p(\mathcal{X}_k|\mathcal{Z}_k) \tag{16}$$

展开后得到，

$$argmin_{\mathcal{X}_k} ||r_0||^2_{\Sigma_0} + \Sigma_{(i,j)\in\mathcal{K}_k} ||r_{\mathcal{I}_{ij}}||^2_{\Sigma_{ij}} + ||r_{\mathcal{C}_{il}}||^2_{\Sigma_{\mathcal{C}}} \tag{17}$$

4.4 IMU Model and Motion Integration

4.4.1 IMU Model

$$_B\tilde{\omega}_{WB}(t) =_B \omega_{WB}(t) + b^g(t) + \eta^g(t) \tag{18}$$

$$_B\tilde{a}(t) = R^T_{WB}(t)(_W a(t) -_w g) + b^a(t) + \eta^a(t) \tag{19}$$

4.4.2 Kinematic Model

$$\dot{R_{WB}} = R_{WB} \quad _B\omega^{\wedge}_{WB}, \quad _W\dot{v} =_w a, \quad _w\dot{p} =_W v, \tag{20}$$

the discrete form assuming $_W a$ and $_B\omega_{WB}$ remains constant in the time interval $[t, t+\Delta t]$

$$R_{WB}(t+\Delta t) = R_{WB}(t)Exp(_B\omega_{WB}\Delta t) \tag{21}$$

$$_W v(t+\Delta t) =_W v(t) +_w a(t)\Delta t \tag{22}$$

$$_W p(t+\Delta t) =_W p(t) +_W v(t)\Delta t + \frac{1}{2}{}_W a(t)\Delta t^2 \tag{23}$$

all in all rewrite equations above, we get,

$$R(t+\Delta t) = R(t)Exp((\tilde{\omega}(t) - b^g(t) - \eta^g(t))\Delta t) \tag{24}$$

$$v(t+\Delta t) = v(t) + g(t)\Delta t + R(t)(\tilde{a}(t) - b^a(t) - \eta^a(t))\Delta t \tag{25}$$

$$p(t+\Delta t) = p(t) + v(t)\Delta t + +\frac{1}{2}g\Delta t^2 + \frac{1}{2}R(t)(\tilde{a}(t) - b^a(t) - \eta^a(t))\Delta t^2 \tag{26}$$

4.5 IMU pre-integration

IMU integration for all Δt between two consecutive frames at times i and j

$$R_j = R_i \prod_{k=i}^{j-1} Exp((\tilde{\omega} - b_k^g - \eta_k^{gd})\Delta t) \tag{27a}$$

$$v_j = v_i + g\Delta_{ij} + \Sigma_{k=i}^{j-1} R_k(\tilde{a}_k - b_k^a - \eta_k^{ad})\Delta t \tag{27b}$$

$$p_j = p_i + \Sigma_{k=i}^{j-1}[v_k\Delta + \frac{1}{2}g\Delta t^2 + \frac{1}{2}R^k(\tilde{a}_k - b_k^a - \eta_k^{ad})\Delta t^2] \tag{27c}$$

relative motion increments

$$\Delta R_{ij} \doteq R_i^T R_j = \prod_{k=i}^{j-1} Exp((\tilde{\omega} - b_k^g - \eta_k^{gd})\Delta t) \tag{28a}$$

$$\Delta v_{ij} \doteq R_i^T(v_j - v_i - g\Delta t_{ij}) = \Sigma_{k=i}^{j-1}\Delta R_{ik}(\tilde{a}_k - b_k^a - \eta_k^{ad})\Delta t \tag{28b}$$

$$\Delta p_{ij} \doteq R_i^T(p_j - p_i - v_i\Delta t_{ij} - \frac{1}{2}\Sigma_{k=i}^{j-1} g\Delta t^2) \tag{28c}$$

$$= \Sigma_{k=i}^{j-1}[\Delta v_{ik}\Delta t + \frac{1}{2}\Delta R_{ik}(\tilde{a}_k - b_k^a - \eta_k^{ad})\Delta t^2] \tag{28d}$$

4.5.1 Preintegrated IMU Measurements

$$\Delta R_{ij} \simeq \prod_{k=i}^{j-1}[Exp((\tilde{\omega}_k - b_i^g)\Delta t)Exp(-J_r^k\eta_k^{gd}\Delta t)] \tag{29a}$$

$$= \Delta\tilde{R}_{ij}\prod_{k=i}^{j-1} Exp(-\Delta\tilde{R}_{k+1j}^T J_r^k\eta_k^{gd}\Delta t) \tag{29b}$$

$$\doteq \Delta\tilde{R}_{ij}Exp(-\Delta\phi_{ij}) \tag{29c}$$

... 类似可得到 Δv_{ij} 和 Δp_{ij}.
接下来就是 “preintegrated measurement model” 。

$$\Delta\tilde{R}_{ij} = R_i^T R_j Exp(\delta\phi_{ij}) \tag{30a}$$

$$\Delta\tilde{v}_{ij} = R_i^T(v_j - v_i - g\Delta t_{ij}) + \delta v_{ij} \tag{30b}$$

$$\Delta p_{ij} = R_i^T(p_j - p_i - v_i\Delta t_{ij} - \frac{1}{2}g\Delta t_{ij}^2) + \delta p_{ij} \tag{30c}$$

4.5.2 *Noise Propagation*

最终可以证明ϕ_{ij})， δv_{ij} 和 δp_{ij} 是 η_k^{gd} 的线性函数，服从高斯分布。

4.5.3 *Incorporating Bias Updates, given a bias update:* $b \leftarrow \bar{b} + \delta b$

$$\Delta\tilde{R}_{ij}(b_i^g) \simeq \Delta\tilde{R}_{ij}(\bar{b}^g)Exp(\frac{\partial\Delta\bar{R}_{ij}}{\partial b^g}\delta b_i^g) \tag{31a}$$

$$\Delta\tilde{v}_{ij}(b_i^g, b_i^a) \simeq \Delta\tilde{v}_{ij}(\bar{b}_i^g, \bar{b}_i^a) + \frac{\partial\Delta\bar{v}_{ij}}{\partial b^g}\delta b_i^g + \frac{\partial\Delta\bar{v}_{ij}}{\partial b^a}\delta b_i^a \tag{31b}$$

$$\Delta\tilde{p}_{ij}(b_i^g, b_i^a) \simeq \Delta\tilde{p}_{ij}(\bar{b}_i^g, \bar{b}_i^a) + \frac{\partial\Delta\bar{p}_{ij}}{\partial b_i^g}\delta b_i^g + \frac{\partial\Delta\bar{p}_{ij}}{\partial b^a}\delta b_i^a \tag{31c}$$

$$\tag{31d}$$

4.5.4 *Bias Model*

Bias are slowly time-varying quantities, Hence model them with a ” Browian Motion” . For more info, see IMU Model

$$\dot{b}^g(t) = \eta^{bg}, \quad \dot{b}^a(t) = \eta ba \tag{32}$$

`okvis` uses a different bias model for acc(bounded random walk).

$$\dot{b}_g(t) = w_{bg}, \quad \dot{b}_a(t) = -\frac{1}{\tau}b_a + w_{ba} \tag{33}$$

the discrete form.

$$b_j^g = b_i^g + \eta^{bgd}, \quad b_j^a = b_i^a + \eta^{bad}, \tag{34}$$

covariance of discrete bias.

$$\Sigma^{bgd} \doteq \Delta t_{ij} Cov(\eta^{bg}), \quad \Sigma^{bad} \doteq \Delta t_{ij} Cov(\eta^{ba}), \tag{35}$$

factor graph of bias model.

$$||r_{b_{ij}}||^2 \doteq ||b_j^g - b_i^g||^2_{\Sigma^{bgd}} + ||b_j^a - b_i^a||^2_{\Sigma^{bad}} \tag{36}$$

4.6 Linearizing expression for nonlinear optimization

`linearizing rotation matrix`

`transformation matrix`

`linearizing homogeneous points`

`examples on stereo camera`

5 Problems still working on

- right jacobian and left jacobian
-

6 supplementaries

6.1 高斯噪声与随机游走

6.1.1 Bias Model

Bias are slowly time-varying quantities, Hence model them with a ” Browian Motion” . For more info, see IMU Model

$$\dot{b}^g(t) = \eta^{bg}, \quad \dot{b}^a(t) = \eta ba \tag{37}$$

`okvis` uses a different bias model for acc(bounded random walk).

$$\dot{b}_g(t) = w_{bg}, \quad \dot{b}_a(t) = -\frac{1}{\tau} b_a + w_{ba} \tag{38}$$

6.2 IMU的工作原理与机制

6.3 gtsam 与isam

6.4 matrix calculus

6.5 manifold and linearization

Chapter 3

State Parameterization for Visual Odometry

This chapter will discuss the choice of parameterization for state variables common to estimation problems in robotics. In the VO algorithm described in Chapter 2, our goal is to estimate the motion of a vehicle in three-dimensional space using a set of sparse stereo keypoints tracked through an image sequence. To perform this estimation, we must choose a set of parameters to represent the position and orientation of the vehicle, as well as the positions of the landmarks in the scene. The most common solution methods used for VO are based on batch nonlinear least squares—an iterative numerical optimization technique that requires successive linearization of the individual error terms in the cost function.

The purpose of this chapter is to derive linearization strategies for several state-parameter classes of interest in robotics and, for each derivation, to provide some notation and algebraic tools for manipulating expressions involving these linearized quantities. We provide derivations for the following state-parameter classes:

1. *rotation matrices:* 3×3 matrices that represent elements of the group $SO(3)$,
2. *transformation matrices:* 4×4 matrices that represent elements of the group $SE(3)$ and can be used to represent coordinate frame transformations compactly, and
3. *homogeneous points:* unit-length 4×1 columns that represent Euclidean points encoded in homogeneous coordinates.

While many of the identities derived below may be found scattered throughout other

textbooks and papers, the contribution of this chapter is to collect the results together in one place, and show how they may be derived using a first-principles Taylor-series expansion approach. This avoids having to resort to Lie algebras, matrix exponentials, tensors, and other tools that can be very useful, but are unnecessary if our goal is simply to linearize error terms for nonlinear optimization.

We begin with a brief sketch of the Gauss-Newton solution to the nonlinear least-squares problem, and then derive linearization strategies for rotation matrices, transformation matrices, and homogeneous points. To demonstrate the utility of our approach, we provide two examples that are used in later chapters: (i) an example of linearizing a stereo camera error term, and (ii) an example of forming and linearizing a prior information term on a 3×3 rotation matrix.

3.1 A Brief Sketch of Batch Nonlinear Least-Squares

Many state estimation tasks in robotics reduce to the problem of finding the state parameter vector, $\mathbf{x}$, that minimizes a scalar squared-error function, $J(\cdot)$, of the form

$$J(\mathbf{x}) := \frac{1}{2} \sum_{n=1}^{N} \mathbf{e}_n(\mathbf{x})^T \mathbf{W}_n \mathbf{e}_n(\mathbf{x}), \tag{3.1}$$

where $\mathbf{e}_n(\cdot)$ is one of N individual error terms weighted by the matrix $\mathbf{W}_n$. For stereo VO, the error terms are based on an observation model of the form,

$$\mathbf{z}_n = \mathbf{g}_n(\mathbf{x}) + \mathbf{v}_n, \tag{3.2}$$

where $\mathbf{z}_n$ is an individual observation, $\mathbf{g}_n(\mathbf{x})$ is the (possibly nonlinear) observation model—our model of how some subset of state parameters produced the observation—and $\mathbf{v}_n$ is a random variable representing observation noise. When we assume that $\mathbf{v}_n$ is independent, zero-mean, and Gaussian,

$$\mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_n), \quad E[\mathbf{v}_m \mathbf{v}_n] = \mathbf{0}, \tag{3.3}$$

setting

$$\underbrace{\mathbf{e}_n(\mathbf{x})}_{\text{error}} := \underbrace{\mathbf{z}_n}_{\text{observation}} - \underbrace{\mathbf{g}_n(\mathbf{x})}_{\substack{\text{predicted} \\ \text{observation}}} \quad \text{and} \quad \mathbf{W}_n := E\left[\mathbf{e}_n \mathbf{e}_n^T\right]^{-1} = \mathbf{R}_n^{-1} \tag{3.4}$$

makes

$$\mathbf{x}^\star = \underset{\mathbf{x}}{\operatorname{argmin}} \left(J \left(\mathbf{x} \right) \right) \tag{3.5}$$

equivalent to the maximum-likelihood estimate of the state given the measurements (Jazwinski, 1970, p. 156). Defining

$$\mathbf{e}(\mathbf{x}) := \begin{bmatrix} \mathbf{e}_1(\mathbf{x}) \\ \vdots \\ \mathbf{e}_N(\mathbf{x}) \end{bmatrix}, \quad \mathbf{R} := \operatorname{diag} \left\{ \mathbf{R}_1, \ldots, \mathbf{R}_N \right\}, \tag{3.6}$$

we may express (3.1) in matrix form,

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{e}(\mathbf{x})^T \mathbf{R}^{-1} \mathbf{e}(\mathbf{x}). \tag{3.7}$$

If $\mathbf{e}(\cdot)$ is a linear function of $\mathbf{x}$, $J(\cdot)$ is exactly quadratic in $\mathbf{x}$ and we may find its minimum by setting $\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}}^T$ to zero and solving the resulting system of equations. When $\mathbf{e}(\cdot)$ is a nonlinear function, the minimum of $J(\cdot)$ must be found iteratively, using a nonlinear optimization technique (Nocedal and Wright, 2006). In robotics, gradient-based optimization techniques, such as Gauss-Newton or Conjugate Gradient, are commonly used. In this thesis, we have used Gauss-Newton and so a basic sketch of the algorithm is provided here.

Starting with an initial guess for the state, $\bar{\mathbf{x}}$—arrived at using a lower-fidelity linear method or through a solution using only a portion of the data available—we make the approximation that

$$\mathbf{x} = \bar{\mathbf{x}} + \delta \mathbf{x}, \tag{3.8}$$

for some small update step, $\delta \mathbf{x}$. We then substitute (3.8) into (3.7) and use a first-order Taylor-series approximation to linearize $\mathbf{e}(\cdot)$ about $\bar{\mathbf{x}}$. The result approximates $J(\cdot)$ as quadratic in $\delta \mathbf{x}$,

$$J(\bar{\mathbf{x}} + \delta \mathbf{x}) = \frac{1}{2} \mathbf{e}(\bar{\mathbf{x}} + \delta \mathbf{x})^T \mathbf{R}^{-1} \mathbf{e}(\bar{\mathbf{x}} + \delta \mathbf{x}) \tag{3.9a}$$

$$\approx \frac{1}{2} \left(\mathbf{e}(\bar{\mathbf{x}}) + \left. \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} \delta \mathbf{x} \right)^T \mathbf{R}^{-1} \left(\mathbf{e}(\bar{\mathbf{x}}) + \left. \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}} \delta \mathbf{x} \right), \tag{3.9b}$$

which we write as

$$J(\delta \mathbf{x}) = \frac{1}{2} \mathbf{e}(\delta \mathbf{x})^T \mathbf{R}^{-1} \mathbf{e}(\delta \mathbf{x}), \tag{3.10}$$

where

$$\bar{\mathbf{e}} := \mathbf{e}(\bar{\mathbf{x}}), \quad \mathbf{E} := \left. \frac{\partial \mathbf{e}(\mathbf{x})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}}, \quad \mathbf{e}(\delta \mathbf{x}) := \bar{\mathbf{e}} + \mathbf{E} \delta \mathbf{x}. \tag{3.11}$$

Now we may find the minimum of (3.10),

$$\delta\mathbf{x}^\star = \underset{\delta\mathbf{x}}{\operatorname{argmin}} \left(J\left(\delta\mathbf{x}\right)\right), \tag{3.12}$$

by expanding $\frac{\partial J(\delta\mathbf{x})}{\partial \delta\mathbf{x}}^T$,

$$\frac{\partial J(\delta\mathbf{x})}{\partial \delta\mathbf{x}}^T = \left(\frac{\partial J(\delta\mathbf{x})}{\partial \mathbf{e}(\delta\mathbf{x})}\frac{\partial \mathbf{e}(\delta\mathbf{x})}{\partial \delta\mathbf{x}}\right)^T \tag{3.13a}$$

$$= \frac{\partial \mathbf{e}(\delta\mathbf{x})}{\partial \delta\mathbf{x}}^T \frac{\partial J(\delta\mathbf{x})}{\partial \mathbf{e}(\delta\mathbf{x})}^T \tag{3.13b}$$

$$= \mathbf{E}^T\mathbf{R}^{-1}\mathbf{e}(\delta\mathbf{x}) \tag{3.13c}$$

$$= \mathbf{E}^T\mathbf{R}^{-1}\left(\bar{\mathbf{e}} + \mathbf{E}\delta\mathbf{x}\right), \tag{3.13d}$$

setting it to zero, and solving the resulting linear system of equations for $\delta\mathbf{x}^\star$,

$$\mathbf{E}^T\mathbf{R}^{-1}\left(\bar{\mathbf{e}} + \mathbf{E}\delta\mathbf{x}^\star\right) = 0 \tag{3.14a}$$

$$\mathbf{E}^T\mathbf{R}^{-1}\mathbf{E}\delta\mathbf{x}^\star = -\mathbf{E}^T\mathbf{R}^{-1}\bar{\mathbf{e}} \tag{3.14b}$$

$$\delta\mathbf{x}^\star = -\left(\mathbf{E}^T\mathbf{R}^{-1}\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{R}^{-1}\bar{\mathbf{e}}. \tag{3.14c}$$

Generally you would not compute the inverse in (3.14c), but rather solve the linear system in (3.14b)[1]. The optimal update is then applied to our current guess,

$$\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \delta\mathbf{x}^\star, \tag{3.15}$$

to result in a new (hopefully better) estimate of the state parameters. This process is iterated until $J(\mathbf{x})$ converges to a minimum[2]. The resulting state estimate, $\mathbf{x}^\star$, has covariance (Bell and Cathey, 1993)

$$\mathbf{P} := \left(\mathbf{E}^T\mathbf{R}^{-1}\mathbf{E}\right)^{-1}, \tag{3.16}$$

which we may think of as

$$\mathbf{x} = \mathbf{x}^\star + \delta\mathbf{x}, \quad \delta\mathbf{x} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}\right). \tag{3.17}$$

One assumption we made in the derivation above was that there were no constraints on the values that $\mathbf{x}$ could assume. This presents challenges for many practical estimation

[1] Often the specific sparsity of the system of equations is exploited to speed up this step as in Hartley and Zisserman (2004, Appendix 6). For general sparse solution techniques, please see Davis (2006).

[2] This is the barest sketch of batch nonlinear optimization. For more details on this and other related algorithms, please see Nocedal and Wright (2006) and the excellent Appendix 6 of Hartley and Zisserman (2004).

problems in robotics where a particular choice of parameters to represent a state variable may have singularities or constraints. For example, the set of rotations constitutes a *non-commutative group*, called $SO(3)$. Regardless of the choice of representation (e.g., rotation matrix, unit-length quaternion, Euler angles), a rotation has exactly three degrees of freedom. All rotational representations involving exactly three parameters have singularities (Stuelpnagel, 1964) and all representations having more than three parameters have constraints. The question of how best to parameterize and handle rotations in state estimation is by no means new. There are many rotational parameterizations available, each with its unique advantages and disadvantages (Shuster, 1993). In spacecraft attitude and robotics estimation, the 4×1 unit-length quaternion (a.k.a., Euler-Rodrigues symmetric parameters), the standard 3×3 rotation matrix, and Euler angles are all common (Crassidis et al., 2007).

In this chapter, we derive linearization strategies for a number of common state-parameter classes: (i) 3×3 rotation matrices that represent elements of the group $SO(3)$, (ii) 4×4 transformation matrices that represent elements of the group $SE(3)$ and can be used to represent coordinate frame transformations compactly, and (iii) unit-length 4×1 columns that represent Euclidean points encoded in homogeneous coordinates. The linearization strategies presented have a number of nice properties: they are *minimal*, in the sense that the update parameterization has the same number of degrees of freedom as the underlying state variable, they are *constraint sensitive* in that the equation used to update the state variable preserves constraints on the state, and they are *unconstrained* in that, as long as the update parameters are small, there are no restrictions on the values they may take. Because of these properties, the approaches can be used in unconstrained optimization.

3.2 Linearizing Expressions Involving Rotation Matrices

In this section we derive a method for linearizing expressions involving rotation matrices[3]. Our approach is a simple first-principles Taylor approximation. To begin, we require the establishment of two identities. Euler's theorem allows us to write a rotation matrix, $\mathbf{C}$,

[3]The contents of this section were derived in collaboration with James R. Forbes and Timothy D. Barfoot and originally appeared in Barfoot et al. (2011a).

in terms of a rotation about a unit-length axis, $\mathbf{a}$, through an angle, φ (Hughes, 1986),

$$\mathbf{C}(\mathbf{a},\varphi) = \cos\varphi\,\mathbf{1} + (1-\cos\varphi)\mathbf{a}\mathbf{a}^T - \sin\varphi\,\mathbf{a}^\times, \tag{3.18}$$

where $\mathbf{1}$ is the identity matrix. We may now take the partial derivative of $\mathbf{C}(\mathbf{a},\varphi)$ with respect to the angle, φ:

$$\begin{aligned}
\frac{\partial\mathbf{C}(\mathbf{a},\varphi)}{\partial\varphi} &= -\sin\varphi\,\mathbf{1} + \sin\varphi\,\mathbf{a}\mathbf{a}^T - \cos\varphi\,\mathbf{a}^\times && \text{(3.19a)}\\
&= \sin\varphi\,\underbrace{\left(-\mathbf{1}+\mathbf{a}\mathbf{a}^T\right)}_{\mathbf{a}^\times\mathbf{a}^\times} - \cos\varphi\,\mathbf{a}^\times && \text{(3.19b)}\\
&= -\cos\varphi\,\mathbf{a}^\times - (1-\cos\varphi)\,\underbrace{\mathbf{a}^\times\mathbf{a}}_{\mathbf{0}}\,\mathbf{a}^T + \sin\varphi\,\mathbf{a}^\times\mathbf{a}^\times && \text{(3.19c)}\\
&= -\mathbf{a}^\times\underbrace{\left(\cos\varphi\,\mathbf{1} + (1-\cos\varphi)\mathbf{a}\mathbf{a}^T - \sin\varphi\,\mathbf{a}^\times\right)}_{\mathbf{C}(\mathbf{a},\varphi)} && \text{(3.19d)}
\end{aligned}$$

Thus, our first key identity is

$$\frac{\partial\mathbf{C}(\mathbf{a},\varphi)}{\partial\varphi} \equiv -\mathbf{a}^\times\mathbf{C}(\mathbf{a},\varphi), \tag{3.20}$$

where

$$\begin{bmatrix}x\\y\\z\end{bmatrix}^\times := \begin{bmatrix}0 & -z & y\\ z & 0 & -x\\ -y & x & 0\end{bmatrix} \tag{3.21}$$

defines the usual 3×3 skew-symmetric matrix, which may be used to implement the cross product for 3×1 columns (Hughes, 1986). An immediate application of this is that for any principal-axis rotation, $\mathbf{C}_\psi(\theta)$, about principal axis ψ and through angle θ, we have

$$\frac{\partial\mathbf{C}_\psi(\theta)}{\partial\theta} = -\mathbf{1}_\psi^\times\mathbf{C}_\psi(\theta), \tag{3.22}$$

where $\mathbf{1}_\psi$ is column ψ of the 3×3 identity matrix. Let us now consider an α-β-γ Euler sequence (with $\alpha\neq\beta$ and $\beta\neq\gamma$),

$$\mathbf{C}(\boldsymbol{\theta}) := \mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{C}_\alpha(\theta_1), \tag{3.23}$$

where $\boldsymbol{\theta} := [\theta_1\ \theta_2\ \theta_3]^T$. Furthermore, select an arbitrary constant 3×1 column, $\mathbf{v}$. Applying (3.22), we have

$$\begin{aligned}
\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\theta_3} &= -\mathbf{1}_\gamma^\times\mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{C}_\alpha(\theta_1)\mathbf{v} = \left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)^\times\mathbf{1}_\gamma, && \text{(3.24a)}\\
\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\theta_2} &= -\mathbf{C}_\gamma(\theta_3)\mathbf{1}_\beta^\times\mathbf{C}_\beta(\theta_2)\mathbf{C}_\alpha(\theta_1)\mathbf{v} = \left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)^\times\mathbf{C}_\gamma(\theta_3)\mathbf{1}_\beta, && \text{(3.24b)}\\
\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\theta_1} &= -\mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{1}_\alpha^\times\mathbf{C}_\alpha(\theta_1)\mathbf{v} = \left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)^\times\mathbf{C}_\gamma(\theta_3)\mathbf{C}_\beta(\theta_2)\mathbf{1}_\alpha, && \text{(3.24c)}
\end{aligned}$$

where we have made use of the two general identities,

$$\mathbf{r}^{\times}\mathbf{s} \equiv -\mathbf{s}^{\times}\mathbf{r}, \tag{3.25a}$$

$$(\mathbf{C}\mathbf{s})^{\times} \equiv \mathbf{C}\mathbf{s}^{\times}\mathbf{C}^{T}, \tag{3.25b}$$

for any 3×1 columns $\mathbf{r}$, $\mathbf{s}$ and any 3×3 rotation matrix, $\mathbf{C}$ (Hughes, 1986). Combining the results in (3.24) we have

$$\begin{aligned}\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\boldsymbol{\theta}} &= \begin{bmatrix}\frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial\theta_1} & \frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial\theta_2} & \frac{\partial(\mathbf{C}(\boldsymbol{\theta})\mathbf{v})}{\partial\theta_3}\end{bmatrix} \\ &= (\mathbf{C}(\boldsymbol{\theta})\mathbf{v})^{\times}\underbrace{\begin{bmatrix}\mathbf{C}_{\gamma}(\theta_3)\mathbf{C}_{\beta}(\theta_2)\mathbf{1}_{\alpha} & \mathbf{C}_{\gamma}(\theta_3)\mathbf{1}_{\beta} & \mathbf{1}_{\gamma}\end{bmatrix}}_{\mathbf{S}(\boldsymbol{\theta})},\end{aligned} \tag{3.26}$$

and thus our second key identity is

$$\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\boldsymbol{\theta}} \equiv \Big(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\Big)^{\times}\mathbf{S}(\boldsymbol{\theta}), \tag{3.27}$$

which we note is true regardless of the choice of Euler sequence. The matrix, $\mathbf{S}(\boldsymbol{\theta})$, is the usual matrix relating angular velocity to Euler-angle rates (Hughes, 1986).

Having established identities (3.20) and (3.27), we now return to first principles and consider carefully how to linearize a rotation. If we have a function, $\mathbf{f}(\mathbf{x})$, of some variable, $\mathbf{x}$, then perturbing $\mathbf{x}$ slightly from its nominal value, $\overline{\mathbf{x}}$, by an amount $\delta\mathbf{x}$ will result in a change in the function. We can express this in terms of a Taylor-series expansion of $\mathbf{f}$ about $\overline{\mathbf{x}}$:

$$\mathbf{f}(\overline{\mathbf{x}} + \delta\mathbf{x}) = \mathbf{f}(\overline{\mathbf{x}}) + \left.\frac{\partial\mathbf{f}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\overline{\mathbf{x}}}\delta\mathbf{x} + (\text{higher order terms}) \tag{3.28}$$

This presupposes that $\delta\mathbf{x}$ is not constrained in any way. The trouble with carrying out the same process with rotations is that most of the representations involve constraints and thus are not easily perturbed (without enforcing the constraint). The notable exceptions are the three-parameter representations, the most common of which are the Euler angle sequences. These contain exactly three parameters and thus each can be varied independently. For this reason, we choose to use Euler angles in our perturbation of functions involving rotations.

Consider perturbing $\mathbf{C}(\boldsymbol{\theta})\mathbf{v}$ with respect to Euler angles $\boldsymbol{\theta}$, where $\mathbf{v}$ is an arbitrary constant 3×1 column. Letting $\overline{\boldsymbol{\theta}} := \begin{bmatrix}\bar{\theta}_1 & \bar{\theta}_2 & \bar{\theta}_3\end{bmatrix}^T$ and $\delta\boldsymbol{\theta} := \begin{bmatrix}\delta\theta_1 & \delta\theta_2 & \delta\theta_3\end{bmatrix}^T$, then

applying a first-order Taylor-series approximation we have

$$\begin{aligned}
\mathbf{C}(\overline{\boldsymbol{\theta}}+\delta\boldsymbol{\theta})\mathbf{v} &\approx \mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v} + \left.\frac{\partial\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)}{\partial\boldsymbol{\theta}}\right|_{\overline{\boldsymbol{\theta}}}\delta\boldsymbol{\theta} && (3.29a)\\
&= \mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v} + \left.\left(\left(\mathbf{C}(\boldsymbol{\theta})\mathbf{v}\right)^{\times}\mathbf{S}(\boldsymbol{\theta})\right)\right|_{\overline{\boldsymbol{\theta}}}\delta\boldsymbol{\theta} && (3.29b)\\
&= \mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v} + \left(\mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v}\right)^{\times}\mathbf{S}(\overline{\boldsymbol{\theta}})\,\delta\boldsymbol{\theta} && (3.29c)\\
&= \mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v} - \left(\mathbf{S}(\overline{\boldsymbol{\theta}})\,\delta\boldsymbol{\theta}\right)^{\times}\left(\mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v}\right) && (3.29d)\\
&= \left(\mathbf{1} - \left(\mathbf{S}(\overline{\boldsymbol{\theta}})\,\delta\boldsymbol{\theta}\right)^{\times}\right)\mathbf{C}(\overline{\boldsymbol{\theta}})\mathbf{v}, && (3.29e)
\end{aligned}$$

where we have used (3.27) to get to the second line. Observing that $\mathbf{v}$ is arbitrary, we can drop it from both sides and write

$$\mathbf{C}(\overline{\boldsymbol{\theta}}+\delta\boldsymbol{\theta}) \approx \underbrace{\left(\mathbf{1} - \left(\mathbf{S}(\overline{\boldsymbol{\theta}})\,\delta\boldsymbol{\theta}\right)^{\times}\right)}_{\text{infinitesimal rot.}}\mathbf{C}(\overline{\boldsymbol{\theta}}), \tag{3.30}$$

which we see is the product of an infinitesimal rotation matrix (Hughes, 1986) and the unperturbed rotation matrix, $\mathbf{C}(\overline{\boldsymbol{\theta}})$. It is worth noting that we did not assume the perturbation is of this multiplicative form, but rather showed that it is a consequence of the linearization procedure. Notationally, it is simpler to write

$$\mathbf{C}(\overline{\boldsymbol{\theta}}+\delta\boldsymbol{\theta}) \approx \left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\mathbf{C}(\overline{\boldsymbol{\theta}}), \tag{3.31}$$

with

$$\delta\boldsymbol{\phi} := \mathbf{S}(\overline{\boldsymbol{\theta}})\,\delta\boldsymbol{\theta}. \tag{3.32}$$

Equation (3.31) is revealing as it tells us how to perturb a rotation matrix when it appears inside any function. This may be done either in terms of perturbations to the Euler angles, $\delta\boldsymbol{\theta}$, or directly through the *rotation vector*, $\delta\boldsymbol{\phi}$.

Rotation matrices fundamentally have three degrees of freedom but are represented by nine parameters. There are therefore six constraints, which may be written as a single matrix orthogonality constraint: $\mathbf{C}\mathbf{C}^{T} = \mathbf{1}$. Suppose this constraint holds for $\mathbf{C}(\overline{\boldsymbol{\theta}})$. Then for the perturbed rotation matrix according to (3.31) we have

$$\begin{aligned}
\mathbf{C}(\overline{\boldsymbol{\theta}}+\delta\boldsymbol{\theta})\mathbf{C}(\overline{\boldsymbol{\theta}}+\delta\boldsymbol{\theta})^{T} &= \left(\left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\mathbf{C}(\overline{\boldsymbol{\theta}})\right)\left(\left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\mathbf{C}(\overline{\boldsymbol{\theta}})\right)^{T}\\
&= \mathbf{1} - \delta\boldsymbol{\phi}^{\times}\delta\boldsymbol{\phi}^{\times}, && (3.33)
\end{aligned}$$

which we see is correct to first order in $\delta\boldsymbol{\phi}$. For this reason, this approach to linearization may be thought of as constraint-sensitive.

Working in the other direction, suppose we have a perturbation in the form of a rotation vector, $\delta\boldsymbol{\phi}$, and we wish to apply this to a prior value of the rotation, $\mathbf{C}(\overline{\boldsymbol{\theta}})$. In terms of Euler angles, we would like to carry out the update

$$\boldsymbol{\theta} = \overline{\boldsymbol{\theta}} + \mathbf{S}(\overline{\boldsymbol{\theta}})^{-1}\,\delta\boldsymbol{\phi}. \tag{3.34}$$

However, we would prefer not to use the Euler angles, because $\mathbf{S}(\overline{\boldsymbol{\theta}})^{-1}$ does not exist precisely at the associated singularities. Instead, we would like to simply store and update the rotation as a rotation matrix. The updated rotation matrix, corresponding to the updated Euler angle sequence above, is given by

$$\begin{aligned}
\mathbf{C}(\boldsymbol{\theta}) &= \mathbf{C}\Big(\overline{\boldsymbol{\theta}} + \mathbf{S}(\overline{\boldsymbol{\theta}})^{-1}\,\delta\boldsymbol{\phi}\Big) && \text{(3.35a)}\\
&\approx \Big(\mathbf{1} - \big(\underbrace{\mathbf{S}(\overline{\boldsymbol{\theta}})\mathbf{S}(\overline{\boldsymbol{\theta}})^{-1}}_{\mathbf{1}}\,\delta\boldsymbol{\phi}\big)^{\times}\Big)\mathbf{C}(\overline{\boldsymbol{\theta}}) && \text{(3.35b)}\\
&\approx \left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\mathbf{C}(\overline{\boldsymbol{\theta}}), && \text{(3.35c)}
\end{aligned}$$

where we have used (3.31), our linearized rotation matrix expression. We then make the observation that setting $\overline{\boldsymbol{\theta}} = \mathbf{0}$ in this last expression reveals

$$\begin{aligned}
\mathbf{C}(\delta\boldsymbol{\phi}) &= \mathbf{C}\Big(\mathbf{0} + \underbrace{\mathbf{S}(\mathbf{0})^{-1}}_{\mathbf{1}}\,\delta\boldsymbol{\phi}\Big) && \text{(3.36a)}\\
&\approx \left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\underbrace{\mathbf{C}(\mathbf{0})}_{\mathbf{1}} && \text{(3.36b)}\\
&\approx \left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right). && \text{(3.36c)}
\end{aligned}$$

Using $\delta\boldsymbol{\phi}$ as an Euler angle sequence to construct a rotation matrix, $\mathbf{C}(\delta\boldsymbol{\phi})$, is somewhat unsettling (since $\delta\boldsymbol{\phi}$ are not Euler angles), but in the neighbourhood of $\overline{\boldsymbol{\theta}} = \mathbf{0}$, $\delta\boldsymbol{\phi} \approx \delta\boldsymbol{\theta}$, so this is reasonable. In fact, *any* Euler sequence could be used to compute $\mathbf{C}(\delta\boldsymbol{\phi})$, as they all result in the same linearized expression. Substituting (3.36c) into (3.35c), we arrive at an expression for our rotation matrix update,

$$\mathbf{C}(\boldsymbol{\theta}) = \mathbf{C}(\delta\boldsymbol{\phi})\,\mathbf{C}(\overline{\boldsymbol{\theta}}), \tag{3.37}$$

where we have dropped the approximation symbol due to the fact that the rotation matrix constraint, $\mathbf{C}(\boldsymbol{\theta})\mathbf{C}(\boldsymbol{\theta})^T = \mathbf{1}$, is satisfied. This update approach allows us to store and update the rotation as a rotation matrix, thereby avoiding singularities and the need to restore the constraint afterwards (i.e., constraint restoration is built in).

The relationship between the rotation vector, $\delta\boldsymbol{\phi}$, and the Euler angle perturbation, $\delta\boldsymbol{\theta}$, expressed in (3.32) is algebraically equivalent to the well-known relationship between rotational velocity, $\boldsymbol{\omega}$, and Euler angle rates (Hughes, 1986):

$$\boldsymbol{\omega} = \mathbf{S}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{3.38}$$

In essence, may think of (3.32) as (3.38) multiplied through by an infinitesimal time increment, dt, such that $\delta\boldsymbol{\phi} = \boldsymbol{\omega}dt$ and $\delta\boldsymbol{\theta} = \dot{\boldsymbol{\theta}}dt$. This insight allows us to perturb a rotation matrix when it appears in any function by using (3.31) in terms of a rotation vector, $\delta\boldsymbol{\phi}$, or by using the well-known $\mathbf{S}(\boldsymbol{\theta})$ matrix formulae to relate small changes in *any* minimal rotation parameterization, $\delta\boldsymbol{\theta}$, to small changes in $\delta\boldsymbol{\phi}$. An extensive list of formulae for $\mathbf{S}(\boldsymbol{\theta})$ covering many popular rotation parameterizations is available in Hughes (1986), Table 2.3 on pages 30 and 31.

The results from this section are fundamental to our approach to estimating rotation matrices that show up commonly in estimation problems in robotics. We use these results below (i) to derive similar expressions for 4×4 transformation matrices in Section 3.3, and (ii) in the example of applying a probabilistic prior term to a rotation matrix in Section 3.5.2.

3.3 Linearizing Expressions Involving Transformation Matrices

In this section we derive a similar linearized perturbation expression for 4×4 transformation matrices and develop notation and identities useful for manipulating expressions containing these quantities. Some of the identities in this section may be found in Murray et al. (1994) where they are derived using Lie algebras and applied to problems concerning the dynamics of robotic manipulators. However, they provide no handling of points at infinity and so further notation and concepts are borrowed from Hartley and Zisserman (2004), and Faugeras and Luong (2001). Here we present the material in the context of state estimation and in notation consistent with Section 3.2. We begin by introducing homogeneous coordinates and transformation matrices as a compact method of representing coordinate-frame transformations.

The projective space, $\mathbb{P}^3$, is the set of equivalence classes of vectors in $\mathbb{R}^4 - \{\mathbf{0}\}$, under the equivalence relationship $\boldsymbol{v} \equiv s\boldsymbol{v}$, for a nonzero scalar, s, and a 4×1 column, $\boldsymbol{v}$

(Hartley and Zisserman, 2004; Faugeras and Luong, 2001). Informally, $\mathbb{P}^3$ can be thought of as the set of lines through the origin of $\mathbb{R}^4$; each element of $\mathbb{P}^3$ is an infinite subset of $\mathbb{R}^4$, and given any point in $\mathbb{R}^4 - \{\mathbf{0}\}$, it maps to a specific element in $\mathbb{P}^3$.

Given the coordinates of a point in three-dimensional Euclidean space, $\mathbf{v} =: \begin{bmatrix} x & y & z \end{bmatrix}^T \in \mathbb{R}^3$, any vector $\boldsymbol{v} =: \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}^T \in \mathbb{P}^3$ which satisfies $\boldsymbol{v} = s \begin{bmatrix} \mathbf{v}^T & 1 \end{bmatrix}^T$, for some real, nonzero scalar, s, is considered the *homogeneous representation*[4] of $\mathbf{v}$. Hence, we define a pair of functions for moving between homogeneous (bold italic symbol) and nonhomogeneous (bold symbol) coordinates,

$$\boldsymbol{v} = \boldsymbol{h}(\mathbf{v}) := \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \qquad \mathbf{v} = \mathbf{h}(\boldsymbol{v}) := \frac{1}{v_4} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \tag{3.39}$$

Both $\mathbf{v}$ and $\boldsymbol{v}$ encode the coordinates of the same point expressed in some coordinate frame. When $v_4 = 0$, the conversion back to $\mathbb{R}^3$ is not possible, as the corresponding point in $\mathbb{R}^3$ is infinitely far away from the coordinate frame origin. However, there is no singularity when keeping these points in homogeneous coordinates. The Jacobian matrices for $\boldsymbol{h}(\cdot)$ and $\mathbf{h}(\cdot)$ (needed later when linearizing error terms for batch estimation) are

$$\frac{\partial \boldsymbol{h}(\mathbf{v})}{\partial \mathbf{v}} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0}^T \end{bmatrix}, \tag{3.40a}$$

$$\frac{\partial \mathbf{h}(\boldsymbol{v})}{\partial \boldsymbol{v}} = \frac{1}{v_4} \begin{bmatrix} \mathbf{1} & -\mathbf{h}(\boldsymbol{v}) \end{bmatrix}. \tag{3.40b}$$

[4] According to Kline (1972), what we now know as homogeneous coordinates were first proposed by Augustus Ferdinand Möbius in his work entitled *Der barycentrische Calcul*, published in 1827. Möbius parameterized a point on a plane, $\mathbf{p} = \begin{bmatrix} x & y \end{bmatrix}^T$, by considering a fixed triangle within the plane and determining the masses, m_1, m_2 and m_3, that must be placed at the triangle vertices to make $\mathbf{p}$ the triangle's center of gravity. Using this system, the coordinates $\boldsymbol{p} := \begin{bmatrix} m_1 & m_2 & m_3 \end{bmatrix}^T$ are not unique as scaling the three masses equally does not change the point location. When the equation of a curve is written in this coordinate system, it becomes homogeneous in m_1, m_2 and m_3—every term in the equation has the same degree. Take for example the equation of a circle centered at $\begin{bmatrix} a & b \end{bmatrix}^T$ with radius r:

$$(x-a)^2 + (y-b)^2 = r^2$$

Written in homogeneous coordinates with $x = m_1/m_3$ and $y = m_2/m_3$, the equation becomes

$$(m_1 - m_3 a)^2 + (m_2 - m_3 b)^2 = m_3^2 r^2,$$

where every term is now quadratic in the homogeneous coordinates. Similarly, the equation of a parabola, $y = x^2$, becomes $m_2 m_3 = m_1^2$.

In homogeneous coordinates, coordinate-frame transformations may be applied to points using a 4×4 transformation matrix, $\boldsymbol{T}$,

$$\boldsymbol{T}_{1,0} = \begin{bmatrix} \mathbf{C}_{1,0} & \boldsymbol{\rho}_1^{0,1} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \boldsymbol{p}_1^{j,1} = \boldsymbol{T}_{1,0}\boldsymbol{p}_0^{j,0}, \quad \boldsymbol{T}_{1,0}^{-1} = \begin{bmatrix} \mathbf{C}_{1,0}^T & -\mathbf{C}_{1,0}^T\boldsymbol{\rho}_1^{0,1} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \boldsymbol{p}_1^{j1} = \boldsymbol{T}_{0,1}^{-1}\boldsymbol{p}_0^{j,0}, \tag{3.41}$$

where $\boldsymbol{p}_0^{j,0}$ encodes the coordinates of a vector from the origin of $\underrightarrow{\mathcal{F}}_0$ to a point j (represented by the superscript $j,0$), and expressed in $\underrightarrow{\mathcal{F}}_0$ (represented by the subscript 0), $\mathbf{C}_{1,0}$ is the rotation matrix that takes vectors from $\underrightarrow{\mathcal{F}}_0$ to $\underrightarrow{\mathcal{F}}_1$, and $\boldsymbol{T}_{1,0}$ is the transformation matrix that takes points from $\underrightarrow{\mathcal{F}}_0$ to $\underrightarrow{\mathcal{F}}_1$. The full complement of subscripts and superscripts is provided in (3.41) for reference. For ease of notation, in the remainder of section we will drop all subscripts and superscripts. Following an approach similar to the one used in Section 3.2, we define a column of parameters,

$$\mathbf{x} := \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix}, \tag{3.42}$$

and write the transformation matrix, $\boldsymbol{T}$, as

$$\boldsymbol{T}(\mathbf{x}) = \begin{bmatrix} \mathbf{C}(\boldsymbol{\theta}) & \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.43}$$

Using the machinery developed in Section 3.2, we perturb $\mathbf{x}$ about the nominal value,

$$\overline{\mathbf{x}} := \begin{bmatrix} \overline{\boldsymbol{\rho}} \\ \overline{\boldsymbol{\theta}} \end{bmatrix}, \tag{3.44}$$

by a perturbation

$$\delta\mathbf{x} := \begin{bmatrix} \delta\boldsymbol{\rho} \\ \delta\boldsymbol{\theta} \end{bmatrix}, \tag{3.45}$$

to get

$$\boldsymbol{T}(\overline{\mathbf{x}} + \delta\mathbf{x}) \approx \begin{bmatrix} (\mathbf{1} - (\mathbf{S}(\overline{\boldsymbol{\theta}})\delta\boldsymbol{\theta})^\times)\mathbf{C}(\overline{\boldsymbol{\theta}}) & \overline{\boldsymbol{\rho}} + \delta\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{3.46}$$

which we rearrange into the form of a multiplicative update,

$$\boldsymbol{T}(\overline{\mathbf{x}} + \delta\mathbf{x}) \approx \begin{bmatrix} \mathbf{1} - (\mathbf{S}(\overline{\boldsymbol{\theta}})\delta\boldsymbol{\theta})^\times & \delta\boldsymbol{\rho} + (\mathbf{S}(\overline{\boldsymbol{\theta}})\delta\boldsymbol{\theta})^\times\overline{\boldsymbol{\rho}} \\ \mathbf{0}^T & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{C}(\overline{\boldsymbol{\theta}}) & \overline{\boldsymbol{\rho}} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\boldsymbol{T}(\overline{\mathbf{x}})}. \tag{3.47}$$

This expression may be simplified by substituting in $\delta\boldsymbol{\phi} = \mathbf{S}(\overline{\boldsymbol{\theta}})\delta\boldsymbol{\theta}$ and defining

$$\delta\boldsymbol{\varrho} := \delta\boldsymbol{\rho} + \delta\boldsymbol{\phi}^{\times}\overline{\boldsymbol{\rho}}, \tag{3.48}$$

to get

$$\boldsymbol{T}(\overline{\mathbf{x}} + \delta\mathbf{x}) \approx \begin{bmatrix} (\mathbf{1} - \delta\boldsymbol{\phi}^{\times}) & \delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{T}(\overline{\mathbf{x}}), \tag{3.49a}$$

$$= \left(\mathbf{1} - \begin{bmatrix} \delta\boldsymbol{\phi}^{\times} & -\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \right) \boldsymbol{T}(\overline{\mathbf{x}}). \tag{3.49b}$$

We further simplify the notation by defining the terms

$$\overline{\boldsymbol{T}} := \boldsymbol{T}(\overline{\mathbf{x}}), \quad \delta\mathbf{t} := \begin{bmatrix} \delta\boldsymbol{\varrho} \\ \delta\boldsymbol{\phi} \end{bmatrix}, \tag{3.50}$$

and the operator $(\cdot)^{\boxplus}$,

$$\begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}^{\boxplus} := \begin{bmatrix} \mathbf{s}^{\times} & -\mathbf{r} \\ \mathbf{0}^T & 0 \end{bmatrix}, \tag{3.51}$$

for 3×1 columns $\mathbf{r}$ and $\mathbf{s}$. This allows us to write

$$\boldsymbol{T}(\overline{\mathbf{x}} + \delta\mathbf{x}) \approx \left(\mathbf{1} - \delta\mathbf{t}^{\boxplus}\right) \overline{\boldsymbol{T}}, \tag{3.52}$$

which may be compared to our rotation matrix result, (3.31). A similar derivation gives

$$\boldsymbol{T}(\overline{\mathbf{x}} + \delta\mathbf{x})^{-1} \approx \overline{\boldsymbol{T}}^{-1} \left(\mathbf{1} + \delta\mathbf{t}^{\boxplus}\right). \tag{3.53}$$

Working in the other direction, suppose we have a perturbation in the form of $\delta\mathbf{t}$, and we wish to apply this to a prior value of the transformation, $\boldsymbol{T}(\overline{\mathbf{x}})$. Rearranging (3.48) to get

$$\delta\boldsymbol{\rho} = \delta\boldsymbol{\varrho} + \overline{\boldsymbol{\rho}}^{\times}\delta\boldsymbol{\phi}, \tag{3.54}$$

we may write the update to $\mathbf{x}$ as

$$\mathbf{x} = \overline{\mathbf{x}} + \begin{bmatrix} \mathbf{1} & \overline{\boldsymbol{\rho}}^{\times} \\ \mathbf{0} & \mathbf{S}(\overline{\boldsymbol{\theta}})^{-1} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\varrho} \\ \delta\boldsymbol{\phi} \end{bmatrix}. \tag{3.55}$$

However, we have again run into the situation that we would prefer not to use the Euler angles, because $\mathbf{S}(\overline{\boldsymbol{\theta}})^{-1}$ does not exist precisely at the associated singularities. Instead, we would like to simply store and update the transformation as a transformation matrix.

Substituting our rotation matrix result, (3.37), into (3.49a), results in an equation for the transformation matrix update step,

$$\boldsymbol{T}(\mathbf{x}) = \begin{bmatrix} \mathbf{C}(\delta\boldsymbol{\phi}) & \delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{T}(\bar{\mathbf{x}}), \tag{3.56a}$$

$$= \boldsymbol{T}(\delta\mathbf{t})\boldsymbol{T}(\bar{\mathbf{x}}), \tag{3.56b}$$

where we have dropped the approximation symbol due to the fact that the rotation matrix constraint, $\mathbf{C}(\boldsymbol{\theta})\mathbf{C}(\boldsymbol{\theta})^T = \mathbf{1}$, is satisfied and the resulting expression on the right-hand side is a valid transformation matrix. This update approach allows us to store and update the transformation as a transformation matrix, thereby avoiding singularities and the need to restore the constraint afterwards.

When dealing with infinitesimal transformation matrices, the $(\cdot)^{\boxplus}$ operator takes on a role similar to that played by the skew-symmetric operator, $(\cdot)^{\times}$, when dealing with infinitesimal rotation matrices. Now we examine how the perturbation, $\delta\mathbf{t}$, affects the transformation of a point, $\mathbf{p}$, represented in homogeneous coordinates by $\boldsymbol{p}$,

$$\boldsymbol{p} := \begin{bmatrix} \mathbf{u} \\ s \end{bmatrix}, \quad \boldsymbol{T}\boldsymbol{p} = \begin{bmatrix} \mathbf{C}\mathbf{u} + s\boldsymbol{\rho} \\ s \end{bmatrix}, \tag{3.57}$$

where s is a nonzero scalar and $\mathbf{u} = s\mathbf{p}$, so that $\mathbf{p} = \mathbf{h}(\boldsymbol{p})$. Applying the perturbation, (3.52), gives us

$$\boldsymbol{T}\boldsymbol{p} \approx \left(\mathbf{1} - \delta\mathbf{t}^{\boxplus}\right)\overline{\boldsymbol{T}}\boldsymbol{p} \tag{3.58a}$$

$$= \overline{\boldsymbol{T}}\boldsymbol{p} - \delta\mathbf{t}^{\boxplus}\overline{\boldsymbol{T}}\boldsymbol{p}, \tag{3.58b}$$

with

$$-\delta\mathbf{t}^{\boxplus}\overline{\boldsymbol{T}}\boldsymbol{p} = \begin{bmatrix} s\delta\boldsymbol{\varrho} - \delta\boldsymbol{\phi}^{\times}(\overline{\mathbf{C}}\mathbf{u} + s\overline{\boldsymbol{\rho}}) \\ 0 \end{bmatrix} \tag{3.59a}$$

$$= \begin{bmatrix} s\delta\boldsymbol{\varrho} + (\overline{\mathbf{C}}\mathbf{u} + s\overline{\boldsymbol{\rho}})^{\times}\delta\boldsymbol{\phi} \\ 0 \end{bmatrix} \tag{3.59b}$$

$$= \underbrace{\begin{bmatrix} s\mathbf{1} & (\overline{\mathbf{C}}\mathbf{u} + s\overline{\boldsymbol{\rho}})^{\times} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}}_{=:(\overline{\boldsymbol{T}}\boldsymbol{p})^{\boxminus}} \underbrace{\begin{bmatrix} \delta\boldsymbol{\varrho} \\ \delta\boldsymbol{\phi} \end{bmatrix}}_{\delta\mathbf{t}}, \tag{3.59c}$$

where we have defined the operator, $(\cdot)^{\boxminus}$, to be

$$\begin{bmatrix} \mathbf{s} \\ t \end{bmatrix}^{\boxminus} = \begin{bmatrix} t\mathbf{1} & \mathbf{s}^{\times} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}, \tag{3.60}$$

for any 3×1 column $\mathbf{s}$ and scalar t. This demonstrates a useful identity,

$$-\mathbf{c}^{\boxplus}\boldsymbol{v} \equiv \boldsymbol{v}^{\boxminus}\mathbf{c}, \tag{3.61}$$

for any 4×1 column $\boldsymbol{v}$ and 6×1 column $\mathbf{c}$. Using this identity we may write a first-order approximation of how a perturbation, $\delta\mathbf{t}$, produces small changes in the transformed point:

$$\boldsymbol{T}\boldsymbol{p} \approx (\mathbf{1} - \delta\mathbf{t}^{\boxplus})\overline{\boldsymbol{T}}\boldsymbol{p} \tag{3.62a}$$

$$= \overline{\boldsymbol{T}}\boldsymbol{p} + (\overline{\boldsymbol{T}}\boldsymbol{p})^{\boxminus}\delta\mathbf{t} \tag{3.62b}$$

Similar results hold for perturbations involving $\boldsymbol{T}^{-1}$:

$$\overline{\boldsymbol{T}}^{-1}\boldsymbol{p} \approx \overline{\boldsymbol{T}}^{-1}(\mathbf{1} + \delta\mathbf{t}^{\boxplus})\boldsymbol{p} \tag{3.63a}$$

$$= \overline{\boldsymbol{T}}^{-1}\boldsymbol{p} - \overline{\boldsymbol{T}}^{-1}\boldsymbol{p}^{\boxminus}\delta\mathbf{t} \tag{3.63b}$$

Finally, we derive some other useful identities for manipulating expressions involving transformation matrices. First we see that we can push a transformation matrix onto the other side of a perturbation:

$$\boldsymbol{T}\delta\mathbf{t}^{\boxplus} = \begin{bmatrix} \mathbf{C} & \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\phi}^{\times} & -\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.64a}$$

$$= \begin{bmatrix} \mathbf{C}\delta\boldsymbol{\phi}^{\times} & -\mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{3.64b}$$

$$= \begin{bmatrix} \mathbf{C}\delta\boldsymbol{\phi}^{\times} & -\mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \underbrace{\boldsymbol{T}^{-1}\boldsymbol{T}}_{\mathbf{1}} \tag{3.64c}$$

$$= \begin{bmatrix} \mathbf{C}\delta\boldsymbol{\phi}^{\times} & -\mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}^T & -\mathbf{C}^T\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \boldsymbol{T} \tag{3.64d}$$

$$= \begin{bmatrix} \mathbf{C}\delta\boldsymbol{\phi}^{\times}\mathbf{C}^T & -\mathbf{C}\delta\boldsymbol{\phi}^{\times}\mathbf{C}^T\boldsymbol{\rho} - \mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \boldsymbol{T} \tag{3.64e}$$

$$= \begin{bmatrix} (\mathbf{C}\delta\boldsymbol{\phi})^{\times} & -(\mathbf{C}\delta\boldsymbol{\phi})^{\times}\boldsymbol{\rho} - \mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \boldsymbol{T} \tag{3.64f}$$

$$= \begin{bmatrix} (\mathbf{C}\delta\boldsymbol{\phi})^{\times} & \boldsymbol{\rho}^{\times}\mathbf{C}\delta\boldsymbol{\phi} - \mathbf{C}\delta\boldsymbol{\varrho} \\ \mathbf{0}^T & 0 \end{bmatrix} \boldsymbol{T} \tag{3.64g}$$

$$= \left(\underbrace{\begin{bmatrix} \mathbf{C} & -\boldsymbol{\rho}^{\times}\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}}_{=:\boldsymbol{T}^{\boxtimes}} \begin{bmatrix} \delta\boldsymbol{\varrho} \\ \delta\boldsymbol{\phi} \end{bmatrix} \right)^{\boxplus} \boldsymbol{T} \tag{3.64h}$$

$$= \left(\boldsymbol{T}^{\boxtimes}\delta\mathbf{t}\right)^{\boxplus}\boldsymbol{T} \tag{3.64i}$$

Stating this result, we have the identity

$$\boldsymbol{T}\mathbf{c}^{\boxplus} \equiv \left(\boldsymbol{T}^{\boxtimes}\mathbf{c}\right)^{\boxplus}\boldsymbol{T}, \tag{3.65}$$

which holds for any transformation matrix $\boldsymbol{T}$ and 6×1 column $\mathbf{c}$. This identity may alternately be written as

$$\boldsymbol{T}\mathbf{c}^{\boxplus}\boldsymbol{T}^{-1} \equiv \left(\boldsymbol{T}^{\boxtimes}\mathbf{c}\right)^{\boxplus}, \tag{3.66}$$

for comparison with the identity, $\mathbf{C}\mathbf{s}^{\times}\mathbf{C}^{T} \equiv (\mathbf{C}\mathbf{s})^{\times}$, which is valid for any rotation matrix $\mathbf{C}$ and 3×1 column $\mathbf{s}$.

The operator, $(\cdot)^{\boxtimes}$, defined as

$$\begin{bmatrix} \mathbf{C} & \boldsymbol{\rho} \\ \mathbf{0}^{T} & 1 \end{bmatrix}^{\boxtimes} := \begin{bmatrix} \mathbf{C} & -\boldsymbol{\rho}^{\times}\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}. \tag{3.67}$$

produces an invertible matrix with the property

$$\boldsymbol{T}^{-\boxtimes} := \left(\boldsymbol{T}^{\boxtimes}\right)^{-1} = \left(\boldsymbol{T}^{-1}\right)^{\boxtimes}. \tag{3.68}$$

This allows us to write (3.65) as

$$\delta\mathbf{t}^{\boxplus}\boldsymbol{T} \equiv \boldsymbol{T}\left(\boldsymbol{T}^{-\boxtimes}\delta\mathbf{t}\right)^{\boxplus}. \tag{3.69}$$

In a bit of manipulation similar to (3.64), we can derive another useful identity:

$$\boldsymbol{T}\boldsymbol{p}^{\boxminus} = \begin{bmatrix} \mathbf{C} & \boldsymbol{\rho} \\ \mathbf{0}^{T} & 1 \end{bmatrix}\begin{bmatrix} \mathbf{u} \\ s \end{bmatrix}^{\boxminus} \tag{3.70a}$$

$$= \begin{bmatrix} \mathbf{C} & \boldsymbol{\rho} \\ \mathbf{0}^{T} & 1 \end{bmatrix}\begin{bmatrix} s\mathbf{1} & \mathbf{u}^{\times} \\ \mathbf{0}^{T} & \mathbf{0}^{T} \end{bmatrix} \tag{3.70b}$$

$$= \begin{bmatrix} s\mathbf{C} & \mathbf{C}\mathbf{u}^{\times} \\ \mathbf{0}^{T} & \mathbf{0}^{T} \end{bmatrix} \tag{3.70c}$$

$$= \begin{bmatrix} s\mathbf{C} & (\mathbf{C}\mathbf{u})^{\times}\mathbf{C} \\ \mathbf{0}^{T} & \mathbf{0}^{T} \end{bmatrix} \tag{3.70d}$$

$$= \begin{bmatrix} s\mathbf{1} & (\mathbf{C}\mathbf{u})^{\times} + s\boldsymbol{\rho}^{\times} \\ \mathbf{0}^{T} & \mathbf{0}^{T} \end{bmatrix}\begin{bmatrix} \mathbf{C} & -\boldsymbol{\rho}^{\times}\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \tag{3.70e}$$

$$= \begin{bmatrix} s\mathbf{1} & (\mathbf{C}\mathbf{u} + s\boldsymbol{\rho})^{\times} \\ \mathbf{0}^{T} & \mathbf{0}^{T} \end{bmatrix}\begin{bmatrix} \mathbf{C} & -\boldsymbol{\rho}^{\times}\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \tag{3.70f}$$

$$= \begin{bmatrix} \mathbf{C}\mathbf{u} + s\boldsymbol{\rho} \\ s \end{bmatrix}^{\boxminus} \begin{bmatrix} \mathbf{C} & -\boldsymbol{\rho}^{\times}\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \tag{3.70g}$$

$$= (\boldsymbol{T}\boldsymbol{p})^{\boxminus} \boldsymbol{T}^{\boxtimes} \tag{3.70h}$$

This identity,

$$\boldsymbol{T}\boldsymbol{p}^{\boxminus} \equiv (\boldsymbol{T}\boldsymbol{p})^{\boxminus} \boldsymbol{T}^{\boxtimes}, \tag{3.71}$$

may also be written as

$$\boldsymbol{T}\boldsymbol{p}^{\boxminus}\boldsymbol{T}^{-\boxtimes} \equiv (\boldsymbol{T}\boldsymbol{p})^{\boxminus}, \tag{3.72}$$

which is again similar to $\mathbf{C}\mathbf{s}^{\times}\mathbf{C}^{T} \equiv (\mathbf{C}\mathbf{s})^{\times}$ and $\boldsymbol{T}\mathbf{c}^{\boxplus}\boldsymbol{T}^{-1} \equiv (\boldsymbol{T}\mathbf{c})^{\boxplus}$.

The results derived in this section are used to derive the estimators used in Sections 5 and 6. We provide a worked example of how to linearize a stereo camera error term in Section 3.5.1.

3.4 Linearizing Expressions Involving Homogeneous Points

One of the main benefits of using transformation matrices to represent coordinate-frame transformations is that it allows us to use homogeneous coordinates to represent points. When estimating a distant point location in Euclidean coordinates, a Gauss-Newton estimator will often attempt to push the Euclidean point out towards infinity. This causes numerical issues in the linear system of equations, (3.14b), and can cause the estimator to diverge. Homogeneous coordinates have the benefit of representing both near and distant landmarks with no singularities or scaling issues (Triggs et al., 2000). Equation, (3.62b), gives some great intuition about this. If we define the components of $\boldsymbol{p}$ to be $\boldsymbol{p} =: \begin{bmatrix} \mathbf{u}^T & s \end{bmatrix}^T$, we may restate (3.62b) as

$$\boldsymbol{T}\boldsymbol{p} \approx \overline{\boldsymbol{T}}\boldsymbol{p} + (\overline{\boldsymbol{T}}\boldsymbol{p})^{\boxminus}\delta\mathbf{t} \tag{3.73a}$$

$$= \begin{bmatrix} \overline{\mathbf{C}}\mathbf{u} + s\boldsymbol{\rho} \\ s \end{bmatrix} + \begin{bmatrix} s\mathbf{1} & (\overline{\mathbf{C}}\mathbf{u} + s\boldsymbol{\rho})^{\times} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\varrho} \\ \delta\boldsymbol{\phi} \end{bmatrix}. \tag{3.73b}$$

As the Euclidean point, $\mathbf{p}$, represented in homogeneous coordinates, $\boldsymbol{p}$, moves away from the coordinate frame origin, s approaches zero. In the limit, we have

$$\lim_{s\to 0}\left(\overline{\boldsymbol{T}}\boldsymbol{p}+(\overline{\boldsymbol{T}}\boldsymbol{p})^{\boxminus}\delta\mathbf{t}\right)\approx\lim_{s\to 0}\left(\begin{bmatrix}\overline{\mathbf{C}}\mathbf{u}+s\boldsymbol{\rho}\\ s\end{bmatrix}+\begin{bmatrix}s\mathbf{1} & (\overline{\mathbf{C}}\mathbf{u}+s\boldsymbol{\rho})^{\times}\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\varrho}\\ \delta\boldsymbol{\phi}\end{bmatrix}\right) \tag{3.74a}$$

$$=\begin{bmatrix}\overline{\mathbf{C}}\mathbf{u}\\ 0\end{bmatrix}+\begin{bmatrix}(\overline{\mathbf{C}}\mathbf{u})^{\times}\\ \mathbf{0}^T\end{bmatrix}\delta\boldsymbol{\phi}. \tag{3.74b}$$

This is a mathematical statement of what we suspect by intuition—distant landmarks only provide information about a camera's orientation, $\delta\boldsymbol{\phi}$, not its position; the homogeneous representation automatically encapsulates the different information that can be discerned from near and distant points.

Therefore, when estimating landmark locations, we would like to use a parameterization for landmarks that allows $s \to 0$. This is also advocated by Triggs et al. (2000). However, a landmark stored in homogeneous coordinates has four parameters representing three fundamental degrees of freedom. Unlike the rotation matrix case, the homogeneous representation is not subject to a constraint. Rather, it has an extra degree of freedom. For example, when estimating motion using the error term derived in Section 3.5.1, multiplying the homogeneous coordinates by a nonzero scalar will result in no change in the objective function. Unconstrained degrees of freedom like this are disastrous for the Gauss-Newton algorithm as they result in an infinite number of possible solutions to the update-step equation, (3.14b), which can cause an implementation of the algorithm to fail.

Hartley and Zisserman (2004) provide a possible solution based on a minimal parameterization of the unit sphere in $\mathbb{R}^4$. They let $\boldsymbol{\vartheta}$ be a 3×1 column of point parameters. Using $\varphi := \|\boldsymbol{\vartheta}\|$ and $\mathbf{a} := \boldsymbol{\vartheta}/\varphi$, they define a map from the parameters, $\boldsymbol{\vartheta}$, onto a homogeneous point[5]

$$\boldsymbol{p}(\boldsymbol{\vartheta}) := \begin{bmatrix}\sin\frac{\varphi}{2}\mathbf{a}\\ \cos\frac{\varphi}{2}\end{bmatrix}. \tag{3.75}$$

[5]This representation has no singularity at $\boldsymbol{\vartheta} = \mathbf{0}$ as $\lim_{\varphi\to 0}\frac{1}{\varphi}\sin\left(\frac{\varphi}{2}\right) = \frac{1}{2}$. Grassia (1998) provides a method of computing this term that is accurate to machine precision. Let ϵ be the smallest increment represented by your floating point type, then

$$\frac{1}{\varphi}\sin\left(\tfrac{\varphi}{2}\right) = \begin{cases}\frac{1}{2}+\frac{\varphi^2}{48} & \text{if } \varphi \le \sqrt[4]{\epsilon},\\ \frac{1}{\varphi}\sin\left(\frac{\varphi}{2}\right) & \text{otherwise.}\end{cases}$$

Essentially they are trading the degree of freedom inherent in the homogeneous representation for the constraint $\boldsymbol{p}(\boldsymbol{\vartheta})^T\boldsymbol{p}(\boldsymbol{\vartheta}) = 1$. This constraint has a hidden benefit in that it keeps the entries of $\boldsymbol{p}$ finite as $\mathbf{p}$ approaches infinity (Triggs et al., 2000). This parameterization may still represent all points, but it has a singularity in that every $\boldsymbol{\vartheta}$ with $\|\boldsymbol{\vartheta}\| = 2\pi$ maps to the same point, $\begin{bmatrix}0 & 0 & 0 & -1\end{bmatrix}^T$. This is similar to parameterizing a rotation using Euler angles; it is minimal but has a singularity.

Our goal here is similar to the previous sections: we would like to derive a linearization method for homogeneous points that allows us to store points in homogeneous coordinates as unit-length 4×1 columns, but with some minimal (3×1) linearized perturbation that, after solution, may be turned into an update step that preserves the unit-length constraint. This goal leads us to a generalization of the linearization strategy used in the sections above.

Given a state variable, $\mathbf{x}$, that has constraints that must be satisfied, we may follow this general strategy for linearization:

1. Choose an update equation, which we will write $\mathbf{x} \leftarrow \mathbf{x} \oplus \delta\mathbf{x}$, where $\oplus$ represents some (possibly nonlinear) update equation with the following properties:

   - it is *minimal*: the update parameter column, $\delta\mathbf{x}$, is a $D \times 1$ column where D corresponds to the number of underlying degrees of freedom in the state, $\mathbf{x}$,
   - it is *constraint sensitive*: after the update is applied, the new value of $\mathbf{x}$ still satisfies any constraints, and
   - it is *unconstrained*: there are no restrictions on the values $\delta\mathbf{x}$ can take and when it is small, it is far away from any singularities in the update equation.

2. Linearize the update equation about the operating point $\delta\mathbf{x} = \mathbf{0}$. This will usually result in greatly simplified, closed-form Jacobian matrices.

3. Wherever $\mathbf{x}$ appears in an error term, substitute in the linearized update equation. Continue to linearize the expression using Taylor-series expansions of nonlinear functions.

4. Use the linearized error terms in the Gauss-Newton algorithm, solving for the optimal update step, $\delta\mathbf{x}^\star$. Because the update equation does not impose constraints on $\delta\mathbf{x}$, there is no need to enforce constraints in the optimization algorithm.

5. Update the state, $\mathbf{x}$, using $\delta\mathbf{x}^\star$ in the full nonlinear update equation chosen in Step 1. Because the update equation was chosen to be constraint-sensitive, the new value of $\mathbf{x}$ should still satisfy any constraints.

The difficult part of this general strategy for linearization is Step 1, finding a suitable update equation that satisfies our three conditions. In the case of homogeneous coordinates, quaternion algebra gives us precisely the tools we need in this regard[6].

In what is to follow, a *quaternion* will be a 4×1 column that may be written as

$$\mathbf{q} := \begin{bmatrix} \boldsymbol{\epsilon} \\ \eta \end{bmatrix}, \tag{3.76}$$

where $\boldsymbol{\epsilon}$ is a 3×1 and η is a scalar. The quaternion *left-hand compound* operator, $+$, and the *right-hand compound* operator, $\oplus$, will be defined as

$$\mathbf{q}^+ := \begin{bmatrix} \eta\mathbf{1} - \boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon} \\ -\boldsymbol{\epsilon}^T & \eta \end{bmatrix} \quad \text{and} \quad \mathbf{q}^\oplus := \begin{bmatrix} \eta\mathbf{1} + \boldsymbol{\epsilon}^\times & \boldsymbol{\epsilon} \\ -\boldsymbol{\epsilon}^T & \eta \end{bmatrix}. \tag{3.77}$$

Under these definitions, the *multiplication* of quaternions, $\mathbf{q}$ and $\mathbf{r}$, which is typically written as $\mathbf{q} \otimes \mathbf{r}$ (Shuster, 1993), may be written equivalently as either

$$\mathbf{q}^+\mathbf{r} \quad \text{or} \quad \mathbf{r}^\oplus\mathbf{q}, \tag{3.78}$$

which are both products of a 4×4 matrix with a 4×1 column. The *conjugate* operator for quaternions, -1, will be defined by

$$\mathbf{q}^{-1} := \begin{bmatrix} -\boldsymbol{\epsilon} \\ \eta \end{bmatrix}. \tag{3.79}$$

The set of quaternions forms a *non-commutative group* under both the $+$ and $\oplus$ operations (Shuster, 1993). The *identity element* of this group, $\boldsymbol{\iota} := \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$, is such that

$$\boldsymbol{\iota}^+ = \boldsymbol{\iota}^\oplus = \mathbf{1}, \tag{3.80}$$

where $\mathbf{1}$ is the 4×4 identity matrix. None of the preceding definitions require the quaternions to be of unit length. However, given two unit-length quaternions, $\mathbf{q}$ and $\mathbf{r}$,

$$\mathbf{q}^T\mathbf{q} = 1, \quad \mathbf{r}^T\mathbf{r} = 1, \tag{3.81}$$

[6] We use the notational conventions of Barfoot et al. (2011b). Please see that reference for a review of quaternion algebra, relevant identities, and historical perspective.

both the $+$ and $\oplus$ operators preserve the unit length:

$$\left(\mathbf{q}^{+}\mathbf{r}\right)^{T}\left(\mathbf{q}^{+}\mathbf{r}\right)=1,\quad \left(\mathbf{q}^{\oplus}\mathbf{r}\right)^{T}\left(\mathbf{q}^{\oplus}\mathbf{r}\right)=1 \tag{3.82}$$

Consequently, we may use (3.77) and (3.75) to define a minimal, constraint-sensitive, update equation for homogeneous points,

$$\overline{\boldsymbol{p}} \leftarrow \boldsymbol{p}(\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}}, \tag{3.83}$$

where $\overline{\boldsymbol{p}}$ is the 4×1, unit-length homogeneous point being updated, and $\boldsymbol{\vartheta}$ is a 3×1 column of update parameters. Next we linearize the update equation around the operating point $\overline{\boldsymbol{\vartheta}}=\mathbf{0}$,

$$\begin{aligned}\boldsymbol{p}(\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}} &= \boldsymbol{p}(\overline{\boldsymbol{\vartheta}}+\delta\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}} && (3.84a)\\ &= \boldsymbol{p}(\mathbf{0}+\delta\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}} && (3.84b)\\ &\approx \left(\boldsymbol{p}(\mathbf{0})+\left.\frac{\partial\boldsymbol{p}(\boldsymbol{\vartheta})}{\partial\boldsymbol{\vartheta}}\right|_{\boldsymbol{\vartheta}=\mathbf{0}}\delta\boldsymbol{\vartheta}\right)^{+}\overline{\boldsymbol{p}}. && (3.84c)\end{aligned}$$

Here we make the substitutions,

$$\boldsymbol{p}(\mathbf{0})=\begin{bmatrix}0\\0\\0\\1\end{bmatrix}=\boldsymbol{\iota},\quad \boldsymbol{V}:=\left.\frac{\partial\boldsymbol{p}(\boldsymbol{\vartheta})}{\partial\boldsymbol{\vartheta}}\right|_{\boldsymbol{\vartheta}=\mathbf{0}}=\begin{bmatrix}\frac{1}{2}&0&0\\0&\frac{1}{2}&0\\0&0&\frac{1}{2}\\0&0&0\end{bmatrix}, \tag{3.85}$$

to get

$$\begin{aligned}\boldsymbol{p}(\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}} &\approx (\boldsymbol{\iota}+\boldsymbol{V}\delta\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}} && (3.86a)\\ &= \overline{\boldsymbol{p}}^{\oplus}(\boldsymbol{\iota}+\boldsymbol{V}\delta\boldsymbol{\vartheta}) && (3.86b)\\ &= \overline{\boldsymbol{p}}+\overline{\boldsymbol{p}}^{\oplus}\boldsymbol{V}\delta\boldsymbol{\vartheta}, && (3.86c)\end{aligned}$$

three equivalent forms of the linearized update equation. Now, to linearize an error term involving a homogeneous point, $\boldsymbol{p}$, we may substitute in any of the linearized forms on the right-hand side of (3.86). When Gauss-Newton returns the optimal update step, $\delta\boldsymbol{\vartheta}^{\star}$, we may update the current value of the point, $\overline{\boldsymbol{p}}$ using (3.83). The updated value will still be of unit length. This is the parameterization used for points in Chapter 6, and so we present a full derivation of the linearized error term for a stereo camera model observing homogeneous points in the examples section below.

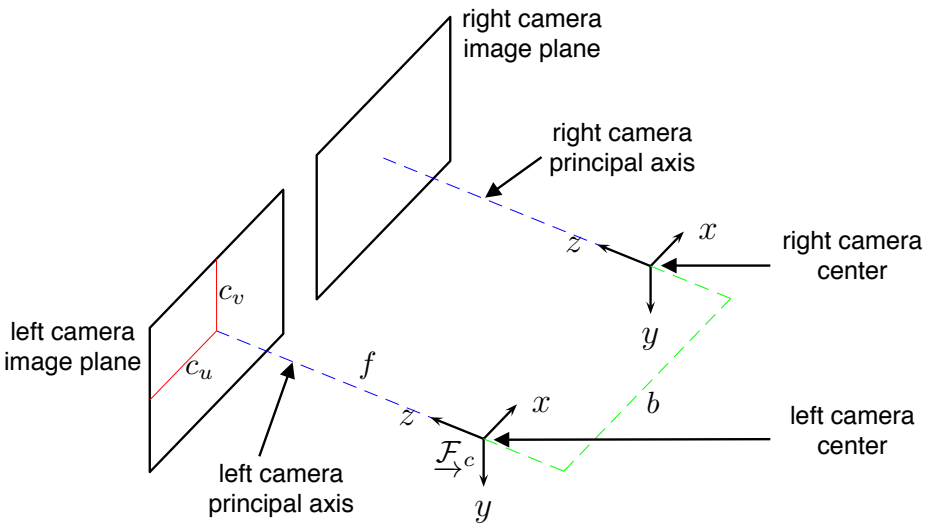

Figure 3.1: The stereo camera model for an idealized pair of cameras with equal focal lengths, parallel optical axes, and aligned image planes.

3.5 Examples

In this section we develop two worked examples using the linearization strategies outlined in this chapter. Section 3.5.1 uses the linearized transformation matrix expressions derived in Section 3.3 to develop a linearized error term for a stereo camera model, and Section 3.5.2 shows how to handle prior information terms on rotation matrices.

3.5.1 Stereo Camera Transformation Matrix Example

This section develops an example using our linearized transformation matrix expression to construct a linearized error term for a stereo camera for use in the Gauss-Newton algorithm. Before using stereo images for estimation, it is common to correct the images for lens distortion and rectify them so that they appear to have come from an idealized pair of cameras with equal focal lengths, parallel optical axes, and aligned image planes. This idealized camera model is shown in Figure 3.1. More details of this preprocessing step and sample images are given in Chapter 2. In this section we assume this preprocessing step has been used, and derive the observation model and linearized error term for an idealized stereo camera observing a point landmark.

Without loss of generality we will assume that the origin of the camera frame, $\underrightarrow{\mathcal{F}}_c$, is placed at the left camera's center. The z-axis points out of the lens, and the x- and

y-axes are aligned with horizontal and vertical pixels, respectively. The idealized stereo camera model has the following parameters:

- c_u, c_v : The horizontal and vertical coordinates of the camera's principal point in pixels (from the top left of the image)
- f_u, f_v : The horizontal and vertical focal length in pixels (these values correspond to the physical focal length, f, (mm) divided by the number of pixels per mm in either the horizontal (f_u) or vertical (f_v) direction (Hartley and Zisserman, 2004, p. 156))
- b : The camera *baseline*: the distance between the two centers of projection in meters

Let $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ be the position of a landmark expressed in $\underrightarrow{\mathcal{F}}_c$. The nonlinear stereo observation model, $\mathbf{g}(\cdot)$, projects $\mathbf{p}$ into the rectified images of the stereo camera:

$$\mathbf{y} = \mathbf{g}(\mathbf{p}) := \frac{1}{z} \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & bf_u \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \tag{3.87}$$

Under this model, the projection, $\mathbf{y}$, is expressed in *disparity coordinates* (Demirdjian and Darrell, 2002). A landmark projection $\mathbf{y}$ has components,

$$\mathbf{y} =: \begin{bmatrix} u \\ v \\ d \end{bmatrix}, \tag{3.88}$$

where u and v are respectively the horizontal and vertical pixel coordinates in the left image, and d is the *disparity*—the difference between the left and right horizontal pixel locations. These equations become linear when expressed in homogeneous coordinates[7],

$$\boldsymbol{y} = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & bf_u \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{=: \boldsymbol{M}} \boldsymbol{p}, \tag{3.89}$$

[7]Recall from Section 3.3 that we are using bold italic symbols to represent columns in homogeneous coordinates and plain bold symbols for quantities not in homogeneous coordinates. In this case, $\boldsymbol{p}$ and $\mathbf{p}$ represent the same quantities, as do $\boldsymbol{y}$ and $\mathbf{y}$, with (3.39) giving us $\mathbf{p} = \mathbf{h}(\boldsymbol{p})$ and $\mathbf{y} = \mathbf{h}(\boldsymbol{y})$.

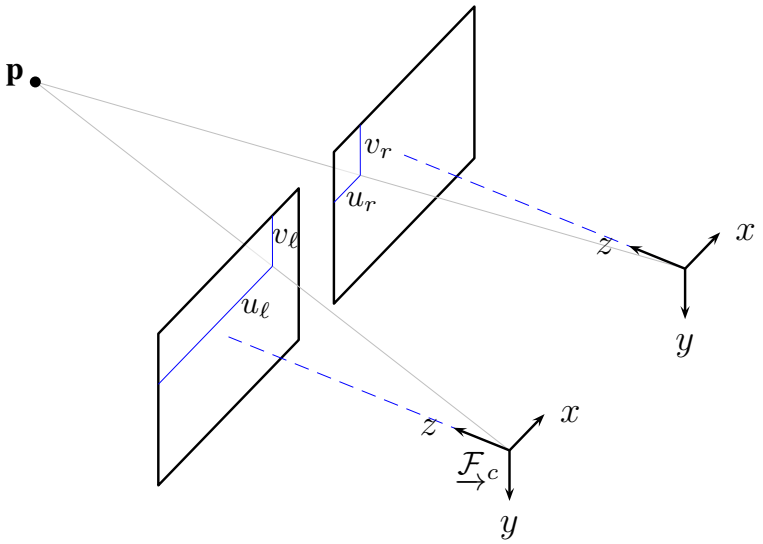

Figure 3.2: An example of a stereo camera measurement of a point landmark. In the idealized stereo camera model, $u_\ell \geq u_r$ and $v_\ell = v_r$.

and the nonlinear form is recovered using (3.39) to get

$$\mathbf{y} = \mathbf{h}\left(\boldsymbol{M}\boldsymbol{p}\right), \tag{3.90}$$

where $\boldsymbol{M}$ is the invertible *stereo projection matrix*. Because $\boldsymbol{M}$ is invertible, we also have

$$\boldsymbol{p} = \boldsymbol{M}^{-1}\boldsymbol{y}, \tag{3.91}$$

which may be used to triangulate points seen in a stereo image. If stereo measurements are made separately in the left and right images, we may define a linear transformation, $\mathbf{U}$, from disparity coordinates, to the predicted left and right pixel measurements,

$$\begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}}_{=:\mathbf{U}} \begin{bmatrix} u \\ v \\ d \end{bmatrix}, \tag{3.92}$$

where u_ℓ and v_ℓ are respectively the horizontal and vertical pixel coordinates of the measurement in the left image and u_r and v_r are the right-image coordinates, as shown

in Figure 3.2. To get back to disparity coordinates, we use the linear transformation

$$\begin{bmatrix} u \\ v \\ d \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & -1 & 0 \end{bmatrix}}_{=:\mathbf{D}} \begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix}. \tag{3.93}$$

Using this camera model, a typical error term for a stereo camera observing landmark j at time k would be

$$\mathbf{e}_{k,j} := \mathbf{y}_{k,j} - \mathbf{h}(\boldsymbol{M}\boldsymbol{T}_{c_k,m}\boldsymbol{p}_m^{j,m}), \tag{3.94}$$

where $\underrightarrow{\mathcal{F}}_{c_k}$ is the camera frame at time k, $\underrightarrow{\mathcal{F}}_m$ is some map frame, $\boldsymbol{p}_m^{j,m}$ is the homogeneous representation of landmark j expressed in the map frame, $\mathbf{y}_{k,j}$ is the 3×1 observation expressed in disparity coordinates, and $\boldsymbol{T}_{c_k,m}$ is the transformation matrix that represents the pose of the camera at time k (i.e., it transforms points from the map frame, to the camera frame at time k). The system described in Chapter 5 only produces an estimate of $\boldsymbol{T}_{c_k,m}$, but the follow-on work in Chapter 6 also estimates the landmark locations. For completeness we will derive the full linearized error term with respect to both the pose variables and the landmark variables. For clarity, we will drop all subscripts.

Let $\delta\mathbf{t}$ represent a 6×1 perturbation of $\boldsymbol{T}$ about its nominal value, $\overline{\boldsymbol{T}}$, and let $\delta\boldsymbol{\vartheta}$ represent a 3×1 perturbation to $\boldsymbol{p}$ about the nominal value, $\overline{\boldsymbol{p}}$. Applying both perturbations gives us

$$\mathbf{e} = \mathbf{y} - \mathbf{h}(\boldsymbol{M}\boldsymbol{T}\boldsymbol{p}), \tag{3.95a}$$

$$\approx \mathbf{y} - \mathbf{h}\left(\boldsymbol{M}\left(\mathbf{1} - \delta\mathbf{t}^{\boxplus}\right)\overline{\boldsymbol{T}}(\boldsymbol{\iota} + \boldsymbol{V}\delta\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}}\right), \tag{3.95b}$$

where we have used (3.52) to perturb $\boldsymbol{T}$, and (3.86) to perturb $\boldsymbol{p}$. Multiplying through, we drop the products of small terms (consistent with a first-order approximation):

$$\mathbf{e} \approx \mathbf{y} - \mathbf{h}\left(\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}} - \boldsymbol{M}\delta\mathbf{t}^{\boxplus}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}} + \boldsymbol{M}\overline{\boldsymbol{T}}(\boldsymbol{V}\delta\boldsymbol{\vartheta})^{+}\overline{\boldsymbol{p}}\right) \tag{3.96}$$

At this point, we may apply some of the identities from this chapter to get

$$\mathbf{e} \approx \mathbf{y} - \mathbf{h}\left(\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}} + \boldsymbol{M}\left(\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}\right)^{\boxminus}\delta\mathbf{t} + \boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}^{\oplus}\boldsymbol{V}\delta\boldsymbol{\vartheta}\right) \tag{3.97}$$

where we have applied (3.61) and (3.78). Finally, letting

$$\mathbf{H} := \left.\frac{\partial\mathbf{h}(\boldsymbol{v})}{\partial\boldsymbol{v}}\right|_{\boldsymbol{v}=\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}} \tag{3.98}$$

from (3.40b), the linearized error term becomes

$$\mathbf{e} \approx \mathbf{y} - \mathbf{h}\left(\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}\right) - \mathbf{H}\boldsymbol{M}\left(\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}\right)^{\boxminus}\delta\mathbf{t} - \mathbf{H}\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}^{\oplus}\boldsymbol{V}\delta\boldsymbol{\vartheta}. \tag{3.99}$$

Defining

$$\bar{\mathbf{e}} := \mathbf{y} - \mathbf{h}\left(\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}\right), \quad \mathbf{E} := \begin{bmatrix} -\mathbf{H}\boldsymbol{M}\left(\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}\right)^{\boxminus} & -\mathbf{H}\boldsymbol{M}\overline{\boldsymbol{T}}\overline{\boldsymbol{p}}^{\oplus}\boldsymbol{V} \end{bmatrix}, \quad \delta\mathbf{x} := \begin{bmatrix} \delta\mathbf{t} \\ \delta\boldsymbol{\vartheta} \end{bmatrix}, \tag{3.100}$$

we may rewrite (3.99) as

$$\mathbf{e}(\delta\mathbf{t}, \delta\boldsymbol{\vartheta}) = \bar{\mathbf{e}} + \mathbf{E}\delta\mathbf{x}, \tag{3.101}$$

which is of the form required by Gauss-Newton in (3.11). Returning to our initial problem, (3.94), one iteration of the Gauss-Newton algorithm would proceed as follows:

1. Begin with initial pose estimates, $\overline{\boldsymbol{T}}_{c_k,m}$, $k = 1 \ldots K$, and an estimate of the position of each point, $\overline{\boldsymbol{p}}_m^{j,m}$, $j = 1 \ldots M$.
2. For each stereo keypoint measurement, compute $\bar{\mathbf{e}}_{k,j}$ and $\mathbf{E}_{k,j}$ from $\mathbf{y}_{k,j}$, $\overline{\boldsymbol{T}}_{c_k,m}$, and $\overline{\boldsymbol{p}}_m^{j,m}$. If an individual pose is held fixed in the optimization, discard the term associated with $\delta\mathbf{t}_{c_k,m}$.
3. Build and solve equation (3.14b) for $\delta\mathbf{t}^{\star}_{c_k,m}$ and $\delta\boldsymbol{\vartheta}^{\star}_j$, the optimal updates to the initial estimates.
4. Update each $\overline{\boldsymbol{T}}_{c_k,m}$ using $\delta\mathbf{t}^{\star}_{c_k,m}$ according to the procedure in (3.56).
5. Update each $\overline{\boldsymbol{p}}_m^{j,m}$ using $\delta\boldsymbol{\vartheta}^{\star}_j$ according to the procedure in (3.83).
6. Check for convergence. If not converged, return to step 2.

The linearized error term derived in this section is used to solve for rover poses in Chapter 5, and to solve for rover poses and landmark locations in Chapter 6. The update steps are singularity free and the landmark parameterization handles both near and distant landmarks.

3.5.2 Rotation Matrix Priors

In this section we derive a method for applying a probabilistic prior to a rotation matrix using the results from Section 3.2. If the uncertainty is small, we may approximate a Gaussian prior on the value of a rotation matrix, $\mathbf{C}$, in the form,

$$\delta\boldsymbol{\varphi} \sim \mathcal{N}\left(\mathbf{0}, \hat{\mathbf{P}}\right), \quad \mathbf{C} = \left(\mathbf{1} - \delta\boldsymbol{\varphi}^{\times}\right)\hat{\mathbf{C}}, \tag{3.102}$$

where $\hat{\mathbf{C}}$ is the mean rotation matrix, and $\delta\boldsymbol{\varphi}$ is a rotation vector representing uncertainty about the mean. The questions we address in this example are (i) given a guess for $\mathbf{C}$, how can we compute an error term—a 3×1 column that represents the "distance" that $\mathbf{C}$ is from the mean rotation, $\hat{\mathbf{C}}$— and (ii) how does this error change with respect to small changes in $\mathbf{C}$ under the linearization strategy derived in Section 3.2? We answer these questions in order, first deriving the error term, and then linearizing the error term.

The Error Term

Given a guess for $\mathbf{C}$, we would like to form an error term based on our prior belief from (3.102). To understand our error term derivation for rotation matrices, it is useful to understand the method of applying a prior to a typical state variable (i.e., not a rotation). So, given a state variable $\mathbf{x}$, a Gaussian prior belief may be written as

$$\delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{P}}), \quad \mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}, \tag{3.103}$$

where $\hat{\mathbf{x}}$ is the mean and $\delta\mathbf{x}$ represents our uncertainty about the mean. Given a guess for $\mathbf{x}$, we may form a prior error term by defining

$$\mathbf{e}(\mathbf{x}) := \delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}, \tag{3.104}$$

which gives us

$$E\left[\mathbf{e}(\mathbf{x})\right] = E\left[\delta\mathbf{x}\right] = \mathbf{0}, \quad E\left[\mathbf{e}(\mathbf{x})\mathbf{e}(\mathbf{x})^T\right] = E\left[\delta\mathbf{x}\delta\mathbf{x}^T\right] = \hat{\mathbf{P}}. \tag{3.105}$$

This error term becomes part of our maximum-likelihood cost function as

$$J_{\text{prior}}(\mathbf{x}) := \frac{1}{2}\mathbf{e}(\mathbf{x})^T\hat{\mathbf{P}}^{-1}\mathbf{e}(\mathbf{x}). \tag{3.106}$$

Following a similar strategy for rotation state variables, we define $\mathbf{e}(\mathbf{C}) := \delta\boldsymbol{\varphi}$. Substituting this into (3.102) we get

$$\left(\mathbf{1} - \mathbf{e}(\mathbf{C})^\times\right)\hat{\mathbf{C}} \approx \mathbf{C} \tag{3.107a}$$

$$\left(\mathbf{1} - \mathbf{e}(\mathbf{C})^\times\right) = \mathbf{C}\hat{\mathbf{C}}^T. \tag{3.107b}$$

Assuming that $\mathbf{C}\hat{\mathbf{C}}^T$ is small[8], it may be approximated by a rotation vector $\boldsymbol{\psi}$ as,

$$\mathbf{C}\hat{\mathbf{C}}^T \approx \left(\mathbf{1} - \boldsymbol{\psi}^\times\right). \tag{3.108}$$

[8]Indeed, all of the linear approximations that we are using to handle rotation matrices require the quantities to be small.

Substituting (3.108) into (3.107b) gives us

$$\left(\mathbf{1} - \mathbf{e}(\mathbf{C})^{\times}\right) \approx \left(\mathbf{1} - \boldsymbol{\psi}^{\times}\right), \tag{3.109}$$

which implies

$$\mathbf{e}(\mathbf{C}) = \boldsymbol{\psi}. \tag{3.110}$$

Unlike the standard case derived in (3.103)–(3.106), the computation of the error, $\boldsymbol{\psi}$, is a nonlinear operation that takes a rotation matrix (that we hope is near identity) and computes a 3×1 column. The choice of how to perform this conversion determines the form of the error, and subsequently the form of the linearized error term derived below.

We choose to use the inverse of our rotation-matrix update equation. In this update equation, (3.37), we use a rotation vector to build a rotation matrix by using the entries of the vector as Euler angles. As we showed in Section 3.2, this approximation holds (to first order) as long as the rotation vector is small. The inverse of this process, which we will write as $\boldsymbol{\psi} \leftarrow \mathbf{C}\hat{\mathbf{C}}^T$, involves extracting a 3×1 column of Euler angles, $\boldsymbol{\psi}$, from the rotation matrix $\mathbf{C}\hat{\mathbf{C}}^T$. This is a nonlinear operation that must be linearized in order to use this error term in Gauss-Newton.

The Linearized Error Term

To produce the error term, (3.110), we have used the inverse of our rotation-matrix update equation, which extracts Euler angles, $\boldsymbol{\psi}$, from the rotation matrix, $\mathbf{C}\hat{\mathbf{C}}^T$. For a Gauss-Newton iteration, we start with a guess, $\overline{\mathbf{C}}$, and make the approximation, (3.31), that

$$\mathbf{C} \approx \left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\overline{\mathbf{C}}, \tag{3.111}$$

for some small update step, $\delta\boldsymbol{\phi}$. Substituting this into our error function, (3.110), we would like to derive an expression, linear in $\delta\boldsymbol{\phi}$, that describes how small changes in $\delta\boldsymbol{\phi}$ become small changes in $\mathbf{e}(\cdot)$. In the notation of Section 3.1, this would be:

$$\mathbf{e}\left(\left(\mathbf{1} - \delta\boldsymbol{\phi}^{\times}\right)\overline{\mathbf{C}}\right) \approx \mathbf{e}\left(\overline{\mathbf{C}}\right) + \mathbf{E}\delta\boldsymbol{\phi} \tag{3.112}$$

In fact, we know the answer to this question from (3.34), which describes how a perturbation in the form of a rotation vector results in small changes in the Euler angles representing a rotation (to first order). From (3.34), we have

$$\mathbf{e}(\delta\boldsymbol{\phi}) = \overline{\boldsymbol{\psi}} + \mathbf{S}\left(\overline{\boldsymbol{\psi}}\right)^{-1}\delta\boldsymbol{\phi}, \tag{3.113}$$

where $\overline{\boldsymbol{\psi}} \leftarrow \overline{\mathbf{C}}\hat{\mathbf{C}}^T$. Hence, the linearized error term in our objective function becomes

$$J_{\text{prior}}(\delta\boldsymbol{\phi}) := \frac{1}{2}\mathbf{e}(\delta\boldsymbol{\phi})^T\hat{\mathbf{P}}^{-1}\mathbf{e}(\delta\boldsymbol{\phi}), \tag{3.114}$$

which is of the form required by Gauss-Newton in (3.11). A prior error term of this form is used to smooth out the localization estimates in Section 5.2.3.

3.6 Conclusion

This chapter has presented a first-principles approach to linearizing expressions involving rotations represented by 3×3 rotation matrices, coordinate-frame transformations represented by 4×4 transformation matrices, and Euclidean points represented as unit-length 4×1 columns. The linearization approach was demonstrated through two examples: (i) linearizing a stereo-camera-model error term, and (ii) forming and linearizing a prior information term on a rotation matrix. Without listing contributions stemming from the collaborative work in Barfoot et al. (2011b), we believe the contributions of this chapter are:

1. A first-principles derivation of the multiplicative constraint-sensitive perturbations of a 4×4 transformation matrix and a unit-length 4×1 homogeneous point given by (3.52) and (3.86) respectively. These may be used to linearize any expression involving a transformation matrix, or homogeneous point.

2. Expressions for updating transformation matrices and unit-length homogeneous points with a constraint-sensitive perturbation are provided in (3.56), and (3.83) respectively. These updates avoid the need to restore constraints afterwards.

3. Development of a number of identities for manipulating expressions involving linearized transformation matrices. These identities are given in (3.61), (3.65), (3.66), (3.69), (3.71), and (3.72).

4. Demonstration of linearizing a stereo-camera-model error term involving a transformation matrix and homogeneous point landmark. The resulting linearized error term, given by (3.99) is used for the VO estimates in subsequent chapters.

5. Demonstration of how to build and linearize an error term representing a Gaussian prior on a rotation matrix. The linearized error term is given in (3.114).

In all cases, the parameterizations we propose are *minimal*, in the sense that the update parameterization has the same number of degrees of freedom as the underlying state variable, they are *constraint sensitive* in that the equation used to update the state variable preserves constraints on the state, and they are *unconstrained* in that, as long as the update parameters are small, there are no restrictions on the values they may take. This allows us to use unconstrained optimization methods without fear of hitting singularities in our underlying representations.

7 reference

http://blog.csdn.net/mulinb/article/details/53421864