

A Quick Reference to Geometric Transforms in Robotics

Dr. Kaustubh Pathak
k.pathak@jacobs-university.de
Jacobs University Bremen, Germany

September 26, 2014

Contents

1	Introduction	2
1.1	Notation	2
1.2	Coordinate Frames	2
1.2.1	Robot-Fixed Map3D Frame	2
1.2.2	Principal Axes of a Frame	3
1.2.3	X3D/VRML Frame	3
2	Representing Rotation	4
2.1	Semantics	4
2.2	Rotation Matrix Properties	5
2.2.1	Axis Angle to Rotation Matrix	5
2.2.2	Chaining or Composition of Rotations	5
2.2.3	Roll, Pitch, and Yaw Angles	5
2.2.4	Eigenspace	6
2.3	Quaternions	6
2.3.1	Compositions	7
3	Transforms and Scan-Matching Update	8
3.1	Notation and Coding Conventions	8
3.2	Transforms	8
3.2.1	Transform Operations	9
3.2.2	Definition of Global Robot Pose	10
3.3	Scan-Matching with Sensor Translational and Rotational Offset	10
4	Propagating Transform Uncertainty	11
4.1	Notation	11
4.2	Compounding Covariances	11
4.2.1	Using Roll, Pitch, and Yaw	12
4.2.2	Using Quaternions	12
4.3	Inverse Transform Covariance Update Using Quaternions	13

Chapter 1

Introduction

This document gives an overview of the conventions used for representing a mobile robot pose (rotation and translation), and how these poses along with their uncertainties are compounded or propagated. When different conventions are prevalent in the literature, the differences are explained. The document also serves as a standards specification for our software for this topic.

1.1 Notation

Vectors are represented in bold small-letters (\mathbf{m}), and unit vectors with an additional carat ($\hat{\mathbf{n}}$). Matrices are in bold capitals (\mathbf{M}). Quaternions are represented as \mathbf{q} and the quaternion product as \diamond . The left subscript/superscript notation is the same as in the popular textbook by J. J. Craig [1]. The left superscript always shows the frame with respect to which a quantity is specified/resolved. Other frames-specific notations are summarized later in Table 3.1.

1.2 Coordinate Frames

For a mobile robot, its body-fixed reference frame/coordinate-system/referential at time-instant k is denoted as $\mathcal{F}_{R[k]}$. The global (world) reference frame will be denoted as \mathcal{F}_G . This notation shows that the global/world frame is fixed and time-independent.

1.2.1 Robot-Fixed Map3D Frame

The Map3D coordinate-frame $\mathcal{F}_{R[k]}$ is attached to the robot at any time-instant k , and is shown in Fig. 1.1. If the time is irrelevant in the context, the time-index will be dropped and the frame denoted as \mathcal{F}_R .

The origin $\mathcal{O}_{R[k]}$ of the frame $\mathcal{F}_{R[k]}$ is on the ground in the middle of the robot between the wheels, $\hat{\mathbf{x}}$ -axis is to the front, $\hat{\mathbf{z}}$ -axis to the top, and $\hat{\mathbf{y}}$ -axis to the left, to give a right-handed coordinate system. For any given sensor, the center is denoted as $\mathcal{O}_{S[k]}$, and the sensor axes are usually parallel to that of the robot-frame. The sensor offset ${}^R_{S[k]}\mathbf{t}$ is $\overrightarrow{\mathcal{O}_{R[k]}\mathcal{O}_{S[k]}}$ resolved in $\mathcal{F}_{R[k]}$ and is noted in the configuration file.

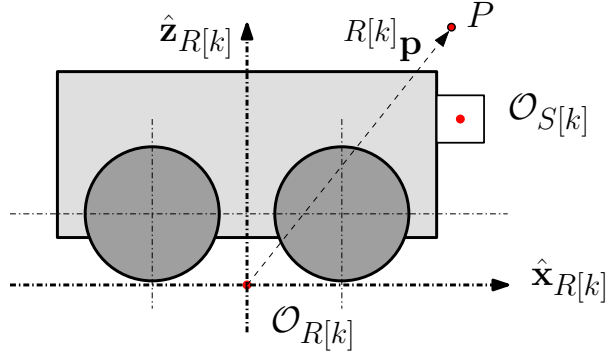


Figure 1.1: The Map3D frame $\mathcal{F}_{R[k]}$ shown from the side view of the robot.

1.2.2 Principal Axes of a Frame

Further we define the following principal unit-vectors for any frame \mathcal{F}_R . Time indices can be added to the frames, if necessary.

$${}^R\hat{\mathbf{x}}_R \equiv \hat{\mathbf{e}}_1 \equiv \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad {}^R\hat{\mathbf{y}}_R \equiv \hat{\mathbf{e}}_2 \equiv \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad {}^R\hat{\mathbf{z}}_R \equiv \hat{\mathbf{e}}_3 \equiv \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \quad (1.1)$$

If $\hat{\mathbf{x}}_R$ is resolved in another frame \mathcal{F}_H , it will be denoted as ${}^H\hat{\mathbf{x}}_R$; in this case, its components will generally be different from 0 and 1.

1.2.3 X3D/VRML Frame

This coordinate-system view, usually representing \mathcal{F}_G , is shown in Fig. 1.2– so if you store the Map3D coordinates as X3D, you see the top view, when you open the scene in a viewer. For performance reasons, it is better to store distances in meters rather than millimeters in X3D.

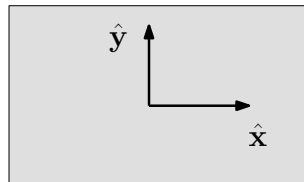


Figure 1.2: The default X3D coordinate-frame representing \mathcal{F}_G on the screen.

Chapter 2

Representing Rotation

2.1 Semantics

A rotation is usually used in two different semantics

1. The frame (coordinate system) remains fixed but the scene rotates. This will be referred to as the fixed-axes **FiAx** semantics. Since only one frame is involved the rotation-matrix is denoted simply as \mathbf{R} without any additional sub/super-scripts. The coordinates of a point \mathbf{p} are changed to \mathbf{p}' , which are related by

$$\mathbf{p}' = \mathbf{R}\mathbf{p}. \quad (2.1)$$

2. The physical scene remains fixed, but the observation frame moves. This will be referred to as the rotated-axes **RoAx** semantics. There are always two frames involved: 1) a *start* reference frame \mathcal{F}_S before the rotation, and 2) an *end* frame \mathcal{F}_E after the rotation. The rotation matrix of \mathcal{F}_E *with respect to* \mathcal{F}_S is now denoted as

$${}^S_E\mathbf{R} \triangleq \begin{pmatrix} S\hat{\mathbf{x}}_E & S\hat{\mathbf{y}}_E & S\hat{\mathbf{z}}_E \end{pmatrix}, \quad (2.2)$$

where, $S\hat{\mathbf{x}}_E$ represents the column vector comprising of the components of the x-axis of \mathcal{F}_E resolved in \mathcal{F}_S , etc.

Assuming no translation, the coordinates of the same physical object in the two frames are related by

$${}^S\mathbf{p} = {}^S_E\mathbf{R} {}^E\mathbf{p} \quad (2.3)$$

Note that in (2.3), the left hand side (LHS) are the “before” coordinates, whereas in (2.1) the LHS has the “after” coordinates. The backward definition in (2.3), however, now allows us the following convenience: the rotation matrix which rotates a scene within a fixed frame is the *same* rotation matrix which rotates the frame while keeping the scene fixed. Thus, we do not need to maintain two different definitions of the rotation matrix!

2.2 Rotation Matrix Properties

2.2.1 Axis Angle to Rotation Matrix

If the rotation given by (2.3) is of angle θ (counter-clockwise positive using right-hand-rule) about axis ${}^S\hat{\mathbf{a}} \equiv [a_x, a_y, a_z]^\top$, then the rotation matrix is given by the Rodrigues' formula

$${}^S\mathbf{R} \equiv \mathbf{R}({}^S\hat{\mathbf{a}}, \theta) = (\cos \theta)\mathbf{I}_3 + (\sin \theta)[{}^S\hat{\mathbf{a}} \times] + (1 - \cos \theta){}^S\hat{\mathbf{a}}{}^S\hat{\mathbf{a}}^\top, \quad (2.4)$$

$$[{}^S\hat{\mathbf{a}} \times] \triangleq \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \quad (2.5)$$

2.2.2 Chaining or Composition of Rotations

Body-fixed Chaining

Analogous to (2.3) we can write the rotation sequence $\mathcal{F}_S \rightarrow \mathcal{F}_E \rightarrow \mathcal{F}_F$ as

$${}^S\mathbf{p} = {}^S\mathbf{R} {}^E\mathbf{p}, \quad {}^E\mathbf{p} = {}^E\mathbf{R} {}^F\mathbf{p}, \quad {}^S\mathbf{p} = {}^S\mathbf{R} {}^F\mathbf{p}. \quad (2.6)$$

Substituting the second in first and comparing with the third, we get the chaining rule

$${}^S\mathbf{R} = {}^S\mathbf{R} {}^E\mathbf{R} \quad (2.7)$$

Space-fixed Chaining

If, in the sequence $\mathcal{F}_S \rightarrow \mathcal{F}_E \rightarrow \mathcal{F}_F$, the rotation $\mathcal{F}_E \rightarrow \mathcal{F}_F$ was given as being about axis ${}^S\hat{\mathbf{a}}$ of angle θ , then

$${}^E\mathbf{R} = {}^S\mathbf{R}^\top \mathbf{R}({}^S\hat{\mathbf{a}}, \theta) {}^S\mathbf{R}, \quad \therefore {}^F\mathbf{R} = \mathbf{R}({}^S\hat{\mathbf{a}}, \theta) {}^E\mathbf{R} \quad (2.8)$$

2.2.3 Roll, Pitch, and Yaw Angles

As an example, roll-pitch-yaw angles can be visualized as body-fixed rotations in the order from the initial frame \mathcal{F}_G :

1. Yaw θ_w about ${}^G\hat{\mathbf{z}}_G \equiv \hat{\mathbf{e}}_3$. After this rotation, we have frame \mathcal{F}_W .
2. Pitch θ_p about ${}^W\hat{\mathbf{y}}_W \equiv \hat{\mathbf{e}}_2$. After this rotation, we have frame \mathcal{F}_P .
3. Roll θ_r about ${}^P\hat{\mathbf{x}}_P \equiv \hat{\mathbf{e}}_1$. After this rotation, we have frame \mathcal{F}_R .

The resultant rotation is

$$\mathbf{R}(\theta_r, \theta_p, \theta_w) = \mathbf{R}(\hat{\mathbf{e}}_3, \theta_w)\mathbf{R}(\hat{\mathbf{e}}_2, \theta_p)\mathbf{R}(\hat{\mathbf{e}}_1, \theta_r) \quad (2.9)$$

Some authors describe the above as a *space-fixed* axes rotation sequence: roll θ_r about ${}^G\hat{\mathbf{x}}_G$ resulting in frame \mathcal{F}_R , followed by pitch θ_p about ${}^G\hat{\mathbf{y}}_G$ resulting in frame \mathcal{F}_P , followed by yaw θ_w about ${}^G\hat{\mathbf{z}}_G$. This interpretation is, however, quite a bit less intuitive and although entirely equivalent due to the results from space-fixed chaining, is impossible to visualize. The reader may verify the equivalence using Eqs. (2.8).

2.2.4 Eigenspace

A rotation matrix is orthonormal, belongs to the special orthogonal group $SO(3)$, and thus has determinant unity. It has one real eigenvalue of unity with the corresponding eigenvector being the rotation axis. The other two eigenvalues are complex conjugates $\cos \theta \pm i \sin \theta$, where θ is the rotation angle about the axis.

2.3 Quaternions

A quaternion can be considered as a 4-tuple which is a generalization of complex numbers. $\check{\mathbf{p}} \equiv p_0 + p_x \mathbf{i} + p_y \mathbf{j} + p_z \mathbf{k}$. The quaternion product is defined as the expansion of $\check{\mathbf{p}} \diamond \check{\mathbf{q}} = (p_0 + p_x \mathbf{i} + p_y \mathbf{j} + p_z \mathbf{k})(q_0 + q_x \mathbf{i} + q_y \mathbf{j} + q_z \mathbf{k})$, using $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. These can be used to derive other relations between the imaginary units, like $\mathbf{ij} = \mathbf{k}$. A quaternion can also be considered simply as a 4-vector.

$$\check{\mathbf{q}} = (q_0 \quad q_x \quad q_y \quad q_z)^\top \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} \quad (2.10)$$

The first component q_0 is called the scalar or real part of the quaternion, whereas the vector \mathbf{q} comprising of its last three components is called the vector or imaginary part of the quaternion. The quaternion multiplication can also be represented by matrix-vector multiplications as follows

$$\check{\mathbf{p}} \diamond \check{\mathbf{q}} \triangleq \Psi(\check{\mathbf{p}})\check{\mathbf{q}} \triangleq \begin{pmatrix} p_0 & -\mathbf{p}^\top \\ \mathbf{p} & p_0 \mathbf{I}_3 + [\mathbf{p} \times] \end{pmatrix} \check{\mathbf{q}}, \quad (2.11a)$$

$$\check{\mathbf{q}} \diamond \check{\mathbf{p}} \triangleq \bar{\Psi}(\check{\mathbf{p}})\check{\mathbf{q}} \triangleq \begin{pmatrix} p_0 & -\mathbf{p}^\top \\ \mathbf{p} & p_0 \mathbf{I}_3 - [\mathbf{p} \times] \end{pmatrix} \check{\mathbf{q}}. \quad (2.11b)$$

The matrix $\Psi(\check{\mathbf{p}})$ is called the quaternion left-product matrix, and $\bar{\Psi}(\check{\mathbf{p}})$ is the right-product matrix.

The quaternion dot product and norm are defined as those of the corresponding 4-vectors. The conjugate of a quaternion $\check{\mathbf{q}}$ is denoted $\check{\mathbf{q}}^*$ and is simply defined as

$$\check{\mathbf{q}}^* \triangleq \begin{pmatrix} q_0 \\ -\mathbf{q} \end{pmatrix} \quad (2.12)$$

Only unit-quaternions with norm unity are used for rotation. To rotate a vector \mathbf{p} , we first “quaternionize” it by creating a purely imaginary quaternion

$$\check{\mathbf{p}} \triangleq \begin{pmatrix} 0 \\ \mathbf{p} \end{pmatrix}. \quad (2.13)$$

Then to rotate it using the unit quaternion $\check{\mathbf{q}}$, one uses

$$\check{\mathbf{p}}' = \check{\mathbf{q}} \diamond \check{\mathbf{p}} \diamond \check{\mathbf{q}}^*, \quad \|\check{\mathbf{q}}\|^2 \equiv q_0^2 + \|\mathbf{q}\|^2 = 1. \quad (2.14)$$

When frames need to be specified, we write, for example,

$${}^S \check{\mathbf{r}} = {}^S \check{\mathbf{q}} \diamond {}^E \check{\mathbf{r}} \diamond {}^S \check{\mathbf{q}}^*. \quad (2.15)$$

The rotation matrix equivalent to this rotation is given by

$$\mathbf{R}(\check{\mathbf{q}}) \equiv (q_0^2 - \mathbf{q}^\top \mathbf{q}) \mathbf{I}_3 + 2\mathbf{q}\mathbf{q}^\top + 2q_0[\mathbf{q} \times]. \quad (2.16)$$

A unit quaternion $\check{\mathbf{q}}(\hat{\mathbf{a}}, \theta)$ which represents a rotation of angle θ (counter-clockwise positive according to the right hand rule) about the axis $\hat{\mathbf{a}}$ is

$$\check{\mathbf{q}}(\hat{\mathbf{a}}, \theta) \equiv \begin{pmatrix} \cos \frac{\theta}{2} \\ \sin \frac{\theta}{2} \hat{\mathbf{a}} \end{pmatrix} \quad (2.17)$$

The reader may verify that the substitution of (2.17) into (2.16) followed by simplification leads us to the Rodrigues formula (2.4).

Quaternions form a singularity-free parameterization of rotations in 3D, i.e. $SO(3)$. Furthermore, $\check{\mathbf{q}}$ and $-\check{\mathbf{q}}$ represent the same rotation, whereas $\check{\mathbf{q}}^*$ represents the opposite rotation.

2.3.1 Compositions

we can write the rotation sequence $\mathcal{F}_S \rightarrow \mathcal{F}_E \rightarrow \mathcal{F}_F$ as

$${}^S\check{\mathbf{p}} = {}^S\check{\mathbf{q}} \diamond {}^E\check{\mathbf{p}} \diamond {}^S\check{\mathbf{q}}^*, \quad {}^E\check{\mathbf{p}} = {}^E\check{\mathbf{q}} \diamond {}^F\check{\mathbf{p}} \diamond {}^E\check{\mathbf{q}}^*, \quad {}^S\check{\mathbf{p}} = {}^S\check{\mathbf{q}} \diamond {}^F\check{\mathbf{p}} \diamond {}^S\check{\mathbf{q}}^*.$$

Substituting the second in first and comparing with the third, we get the chaining rule

$${}^S\check{\mathbf{q}} = {}^S\check{\mathbf{q}} \diamond {}^E\check{\mathbf{q}} \quad (2.18)$$

Combining this result with (2.17) and (2.9), we can derive the mapping from roll, pitch, yaw to the resultant quaternion

$$\check{\mathbf{q}}(\theta_r, \theta_p, \theta_w) = \check{\mathbf{q}}(\hat{\mathbf{e}}_3, \theta_w) \diamond \check{\mathbf{q}}(\hat{\mathbf{e}}_2, \theta_p) \diamond \check{\mathbf{q}}(\hat{\mathbf{e}}_1, \theta_r). \quad (2.19)$$

Chapter 3

Transforms and Scan-Matching Update

3.1 Notation and Coding Conventions

The frames-specific notation is summarized in Table 3.1.

Table 3.1: Notations

Notation	Meaning
$\mathcal{F}_{R[k]}$	Coordinate frame attached to object ‘R’ (usually the robot) at sample time-instant k .
$\mathcal{O}_{R[k]}$	Origin of $\mathcal{F}_{R[k]}$.
${}^R[k]\mathbf{p}$	For any general point P , the position vector $\overrightarrow{\mathcal{O}_{R[k]}P}$ resolved in $\mathcal{F}_{R[k]}$.
${}^H\hat{\mathbf{x}}_R$	The x-axis direction of \mathcal{F}_R resolved in \mathcal{F}_H . Similarly, ${}^H\hat{\mathbf{y}}_R$, ${}^H\hat{\mathbf{z}}_R$ can be defined. Obviously, ${}^R\hat{\mathbf{x}}_R = \hat{\mathbf{e}}_1$. Time indices can be added to the frames, if necessary.
${}^{R[k]}_{S[k']}\mathbf{R}$	The rotation-matrix of $\mathcal{F}_{S[k']}$ with respect to $\mathcal{F}_{R[k]}$. See also (2.2).
${}^R_S\mathbf{t}$	The translation vector $\overrightarrow{\mathcal{O}_R\mathcal{O}_S}$ resolved in \mathcal{F}_R . In [1], this is also written as ${}^R\mathbf{p}_{ORGS}$, which is a bit unwieldy. Time indices can be added to the frames, if necessary.

Coding Convention The rotation matrix ${}^{S[k]}_{S[k+1]}\mathbf{R}$ will be denoted as `R_s_kp1_s_k`, ${}^{R[k]}_{S[j]}\mathbf{R}$ as `R_s_j_r_k`. The translation ${}^{R[k]}_{R[j]}\mathbf{t}$ will be denoted as `t_r_j_r_k`. The position vector ${}^B\mathbf{p}$ will be denoted as `p_b`. These examples show that the reference frame should always come at the end.

3.2 Transforms

As shown in Fig. 3.1, the same point P has different coordinates when observed and resolved in different frames \mathcal{F}_A , \mathcal{F}_B , \mathcal{F}_G .

We define

$${}^G_A\mathbf{t} \triangleq \overrightarrow{\mathcal{O}_G\mathcal{O}_A} \text{ resolved in } \mathcal{F}_G.$$

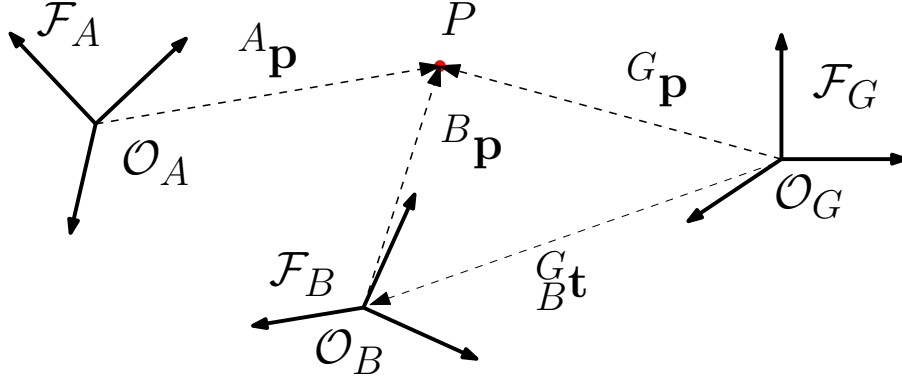


Figure 3.1: The coordinates of the same physical point P viewed from different frames.

Then we have the following relationships,

$${}^G\mathbf{p} = {}^G\mathbf{R} \ {}^A\mathbf{p} + {}^G\mathbf{t} \quad (3.1a)$$

$$\triangleq {}^G\mathbf{T}({}^A\mathbf{p}). \quad (3.1b)$$

The transform or *mapping* ${}^G\mathbf{T}$ can be realized in many ways, for example by using rotation-matrices or quaternions. One popular way is the 4×4 homogenous matrix

$$\begin{pmatrix} {}^G\mathbf{p} \\ 1 \end{pmatrix} \equiv \begin{pmatrix} {}^G\mathbf{R} & {}^G\mathbf{t} \\ \mathbf{0}_{1 \times 3} & 1 \end{pmatrix} \begin{pmatrix} {}^A\mathbf{p} \\ 1 \end{pmatrix} \quad (3.2)$$

In general, we specify a transform as a pair of relative translation and rotation, e.g.

$${}^G\mathbf{T} \equiv \begin{Bmatrix} {}^G\mathbf{t} \\ {}^A\mathbf{R} \\ {}^G\mathbf{R} \end{Bmatrix}, \text{ or,} \quad {}^G\mathbf{T} \equiv \begin{Bmatrix} {}^G\mathbf{t} \\ {}^A\mathbf{R} \\ {}^G\mathbf{q} \end{Bmatrix} \quad (3.3)$$

3.2.1 Transform Operations

Chaining or Composition of or Compounding or Adding Transforms

Transforms can be easily chained as follows

$${}^G\mathbf{T} = {}^G\mathbf{T} \ {}^A\mathbf{T} \quad \equiv \begin{Bmatrix} {}^G\mathbf{R} \ {}^A\mathbf{t} + {}^G\mathbf{t} \\ {}^G\mathbf{R} \ {}^A\mathbf{R} \end{Bmatrix} \quad (3.4)$$

Inverse Transform

$${}^B\mathbf{T} = {}^A\mathbf{T}^{-1} \quad \equiv \begin{Bmatrix} -{}^A\mathbf{R}^T \ {}^A\mathbf{t} \\ {}^A\mathbf{R}^T \end{Bmatrix} \quad (3.5)$$

Relative or Difference Transform

Given ${}^G\mathbf{T}$ and ${}^B\mathbf{T}$, find ${}^B\mathbf{T}$.

$${}^B\mathbf{T} = {}^G\mathbf{T}^{-1} \ {}^A\mathbf{T}. \quad (3.6)$$

3.2.2 Definition of Global Robot Pose

The robot's global pose at sample-time k is given by the transform

$${}^G_R\mathbf{T} \equiv \left\{ \begin{array}{c} {}^G\mathbf{t} \\ {}^G_R\mathbf{R} \end{array} \right\} \quad (3.7)$$

3.3 Scan-Matching with Sensor Translational and Rotational Offset

Problem The robot has a sensor attached on it which observes the scene. The transform of the sensor with respect to the robot ${}^{R[k]}_{S[k]}\mathbf{T}$ is possibly time-varying, perhaps because the sensor is mounted on an arm, which in turn is attached to the robot. Two sensor readings are taken at times k and j . A *scan-matching* algorithm runs on the sensor readings and determines the relative sensor transform ${}^{S[k]}_{S[j]}\mathbf{T}$. What is ${}^{R[k]}_{R[j]}\mathbf{T}$, and how can the global pose of the robot ${}^G_{R[k]}\mathbf{T}$ be updated to ${}^G_{R[j]}\mathbf{T}$?

Solution From the compounding relations, we get the following

$${}^{R[k]}_{R[j]}\mathbf{T} = {}^{R[k]}_{S[k]}\mathbf{T} {}^{S[k]}_{S[j]}\mathbf{T} {}^{R[j]}_{S[j]}\mathbf{T}^{-1} \quad (3.8a)$$

$${}^G_{R[j]}\mathbf{T} = {}^G_{R[k]}\mathbf{T} {}^{R[k]}_{R[j]}\mathbf{T}. \quad (3.8b)$$

All terms on the right hand side are known.

Chapter 4

Propagating Transform Uncertainty

If all we needed were transforms without any uncertainty information, rotation-matrix based transforms would have sufficed. However, most scan-matchers also provide us with an uncertainty estimation in terms of a covariance matrix of the transform. In the following sections, we will show how to perform elementary transform operations of composition and inverse when the transforms also contain a covariance matrix of translation and rotation. It is also shown that the rotation is best parameterized internally as a quaternion for algebraically simpler covariance updates. On the other hand, covariances for rotation parameterization based on roll, pitch, and yaw are easier to understand. Fortunately covariances can be converted across parameterizations using suitable Jacobians.

4.1 Notation

A transform ${}^G\mathbf{T}_A$, when written in terms of roll, pitch, and yaw angles, can be represented as a 6-vector

$${}^G\mathbf{T}_A \equiv \begin{pmatrix} {}^G\mathbf{t}_A \\ {}^G\boldsymbol{\Theta} \end{pmatrix}, \quad \boldsymbol{\Theta} \triangleq (\theta_r \quad \theta_p \quad \theta_w)^\top. \quad (4.1)$$

In this case, the covariance matrix of ${}^G\mathbf{T}_A$ is a 6×6 matrix denoted as ${}^G\mathbf{C}_A$.

A transform ${}^G\mathbf{T}_A$, when written in terms of a unit quaternion, can be represented as a 7-vector

$${}^G\mathbf{T}_A \equiv \begin{pmatrix} {}^G\mathbf{t}_A \\ {}^G\tilde{\mathbf{q}}_A \end{pmatrix} \quad (4.2)$$

In this case, the covariance matrix of ${}^G\mathbf{T}_A$ is a 7×7 matrix denoted again as ${}^G\mathbf{C}_A$. Whether one is using roll, pitch, yaw or quaternions is clear from the context.

4.2 Compounding Covariances

The problem can be stated generally as follows: given the transform ${}^G\mathbf{T}_A$ and its covariance ${}^G\mathbf{C}_A$, and a relative transform ${}^A\mathbf{T}_B$ and its covariance ${}^A\mathbf{C}_B$, what are the compounded transform ${}^G\mathbf{T}_B$ and its covariance ${}^G\mathbf{C}_B$? We have already found ${}^G\mathbf{T}_B$ in Eq. (3.4). We now seek to find its covariance ${}^G\mathbf{C}_B$.

4.2.1 Using Roll, Pitch, and Yaw

We denote the Eq. (2.9), which computes a rotation-matrix from Θ , as a mapping $\hat{g} : \mathbb{R}^3 \mapsto SO(3)$. Therefore, its reverse mapping is $\hat{g}^{-1} : SO(3) \mapsto \mathbb{R}^3$, which computes the angles from the rotation-matrix.

We need to now estimate the updated covariance ${}^G_B\mathbf{C}$. Eq. (3.4) can be written together as a mapping $\mathbf{f} : \mathbb{R}^6 \times \mathbb{R}^6 \mapsto \mathbb{R}^6$ as

$${}^G_B\mathbf{T} = \mathbf{f}({}^G_A\mathbf{T}, {}^A_B\mathbf{T}) = \begin{pmatrix} {}^G_A\mathbf{t} + \hat{g}({}^G_A\Theta) {}^A_B\mathbf{t} \\ \hat{g}^{-1}(\hat{g}({}^G_A\Theta) \hat{g}({}^A_B\Theta)) \end{pmatrix} \quad (4.3)$$

Its 6×12 Jacobian is given by

$$\mathbf{J} = \left(\frac{\partial \mathbf{f}}{\partial {}^G_A\mathbf{T}} \quad \frac{\partial \mathbf{f}}{\partial {}^A_B\mathbf{T}} \right) \quad (4.4)$$

Needless to say, this will be a highly involved nonlinear expression, which can be computed only numerically at each step. Assuming that the random-vectors ${}^G_A\mathbf{T}$ and ${}^A_B\mathbf{T}$ are mutually independent, the covariance update is given by

$${}^G_B\mathbf{C} = \mathbf{J} \begin{pmatrix} {}^G_A\mathbf{C} & \mathbf{0}_{6 \times 6} \\ \mathbf{0}_{6 \times 6} & {}^A_B\mathbf{C} \end{pmatrix} \mathbf{J}^\top. \quad (4.5)$$

4.2.2 Using Quaternions

In this case, the transform is given by a 7-vector as in (4.2). We denote the computation of a rotation matrix from a quaternion using Eq. (2.16) as a mapping $\mathbf{R}(\check{\mathbf{q}}) : \mathbb{R}^4 \mapsto SO(3)$. If this mapping is used to rotate a vector $\mathbf{t}' = \mathbf{R}(\check{\mathbf{q}})\mathbf{t}$, then we can derive the following two Jacobians:

$$\frac{\partial \{\mathbf{R}(\check{\mathbf{q}})\mathbf{t}\}}{\partial \check{\mathbf{q}}} = 2 \begin{bmatrix} q_0\mathbf{t} + \mathbf{q} \times \mathbf{t} & | & -\mathbf{t}\mathbf{q}^\top + (\mathbf{q}^\top\mathbf{t})\mathbf{I}_3 + \mathbf{q}\mathbf{t}^\top - q_0[\mathbf{t} \times] \end{bmatrix} \quad (4.6)$$

$$\frac{\partial \{\mathbf{R}(\check{\mathbf{q}}^*)\mathbf{t}\}}{\partial \check{\mathbf{q}}} = 2 \begin{bmatrix} q_0\mathbf{t} - \mathbf{q} \times \mathbf{t} & | & -\mathbf{t}\mathbf{q}^\top + (\mathbf{q}^\top\mathbf{t})\mathbf{I}_3 + \mathbf{q}\mathbf{t}^\top + q_0[\mathbf{t} \times] \end{bmatrix} \quad (4.7)$$

These expressions will be quite useful in subsequent computations.

The transform update equations (3.4) can be written as

$${}^G_B\mathbf{T} = \mathbf{f}({}^G_A\mathbf{T}, {}^A_B\mathbf{T}) = \begin{pmatrix} {}^G_A\mathbf{t} + \mathbf{R}({}^G_A\check{\mathbf{q}}) {}^A_B\mathbf{t} \\ {}^G_A\check{\mathbf{q}} \diamond {}^A_B\check{\mathbf{q}} \end{pmatrix} \quad (4.8)$$

The Jacobian of ${}^G_B\mathbf{T}$ with respect to its arguments is given by

$$\mathbf{J}_{7 \times 14} \triangleq \begin{pmatrix} \frac{\partial {}^G_B\mathbf{T}}{\partial {}^G_A\mathbf{T}} & \frac{\partial {}^G_B\mathbf{T}}{\partial {}^A_B\mathbf{T}} \end{pmatrix} \quad (4.9)$$

Its individual submatrices are given by

$$\frac{\partial {}^G_B\mathbf{T}}{\partial {}^G_A\mathbf{T}} = \begin{pmatrix} \frac{\partial {}^G_B\mathbf{t}}{\partial {}^G_A\mathbf{t}} & \frac{\partial {}^G_B\mathbf{t}}{\partial {}^G_A\check{\mathbf{q}}} \\ \frac{\partial {}^G_B\check{\mathbf{q}}}{\partial {}^G_A\mathbf{t}} & \frac{\partial {}^G_B\check{\mathbf{q}}}{\partial {}^G_A\check{\mathbf{q}}} \end{pmatrix} = \begin{pmatrix} \mathbf{I}_3 & \frac{\partial \mathbf{R}({}^G_A\check{\mathbf{q}}) {}^A_B\mathbf{t}}{\partial {}^G_A\check{\mathbf{q}}} \\ \mathbf{0}_{4 \times 3} & \overline{\Psi}({}^G_A\check{\mathbf{q}}) \end{pmatrix}, \quad (4.10a)$$

where, definitions in (2.11) and (4.6) should be recalled. Similarly, one can obtain

$$\frac{\partial {}^G_B\mathbf{T}}{\partial {}^A_B\mathbf{T}} = \begin{pmatrix} \frac{\partial {}^G_B\mathbf{t}}{\partial {}^A_B\mathbf{t}} & \frac{\partial {}^G_B\mathbf{t}}{\partial {}^A_B\check{\mathbf{q}}} \\ \frac{\partial {}^G_B\check{\mathbf{q}}}{\partial {}^A_B\mathbf{t}} & \frac{\partial {}^G_B\check{\mathbf{q}}}{\partial {}^A_B\check{\mathbf{q}}} \end{pmatrix} = \begin{pmatrix} \mathbf{R}({}^G_A\check{\mathbf{q}}) & \mathbf{0}_{3 \times 4} \\ \mathbf{0}_{4 \times 3} & \Psi({}^G_A\check{\mathbf{q}}) \end{pmatrix}, \quad (4.10b)$$

Substitution of Eqs. (4.10) in (4.9) gives the required Jacobian. Assuming that the random-vectors ${}^G\mathbf{T}$ and ${}^A\mathbf{T}$ are mutually independent, the covariance update is given by

$${}^G\mathbf{C}_{7\times 7} = \mathbf{J}_{7\times 14} \begin{pmatrix} {}^G\mathbf{C}_{7\times 7} & \mathbf{0}_{7\times 7} \\ \mathbf{0}_{7\times 7} & {}^A\mathbf{C}_{7\times 7} \end{pmatrix} \mathbf{J}^\top. \quad (4.11)$$

Algebra-wise, the quaternion-based update is much easier than the roll-pitch-yaw based update because closed form expressions can be obtained.

4.3 Inverse Transform Covariance Update Using Quaternions

The transform ${}^A\mathbf{T}$ is related to its inverse ${}^B\mathbf{T}$ by (3.5), which can be written using quaternions as

$${}^B\mathbf{T} = \begin{pmatrix} -\mathbf{R}({}^A\check{\mathbf{q}}^*) & {}^A\mathbf{t} \\ {}^A\check{\mathbf{q}}^* & \mathbf{0} \end{pmatrix} \quad (4.12)$$

The conjugate can be computed from a matrix multiplication as

$$\check{\mathbf{q}}^* = \mathcal{J}\check{\mathbf{q}}, \quad \mathcal{J} \triangleq \text{diag}(1 \quad -1 \quad -1 \quad -1) \quad (4.13)$$

We thus get the Jacobian

$$\mathbf{J} \equiv \frac{\partial {}^B\mathbf{T}}{\partial {}^A\mathbf{T}} = \begin{pmatrix} -{}^B\mathbf{R} & -\partial\mathbf{R}({}^A\check{\mathbf{q}}^*)_{\mathbf{B}}\mathbf{t}/\partial {}^A\check{\mathbf{q}} \\ \mathbf{0}_{4\times 3} & \mathcal{J} \end{pmatrix} \quad (4.14)$$

This can be computed using (4.7). Finally,

$${}^B\mathbf{C} = \mathbf{J} {}^A\mathbf{C} \mathbf{J}^\top. \quad (4.15)$$

Bibliography

- [1] J. J. Craig, *Introduction to robotics – Mechanics and control*. Prentice Hall, 2005.