SLAM十四讲笔记

Xin Li

Contents

SLAM十四讲笔记
Contents
Notations
Matrix Transformation
- Coordinates & Basis
- Inner/Outer Product
- Euclidean Transformation
- Overall Transform Matrix
- Rotation Vector & Euler Angles
  - Rotation Vector
  - Euler Angles
- Quaternions
- C++ Eigen 3 Implementation

Notations

Notation	Meaning
a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$(\cdot)^T$	Matrix transpose
$(\cdot)^{-1}$	Matrix inverse
$\mathcal{E}\{\cdot\}$	Expectation
$\|\|\mathbf{a}\|\|$	Euclidean norm of vector $\mathbf{a}$
$\|\|\mathbf{A}\|\|_F$	Frobenius norm of matrix $\mathbf{A}$
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers

Matrix Transformation

Coordinates & Basis

A point in the real Cartesian space $\mathbb{R}^3$ can be described by the basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, as

$$a = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3. \tag{1}$$

$$\mathbf{a} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3. \tag{1}$$

Inner/Outer Product

For two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$, their inner product is defined as

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos \langle \mathbf{a}, \mathbf{b} \rangle \tag{2}$$

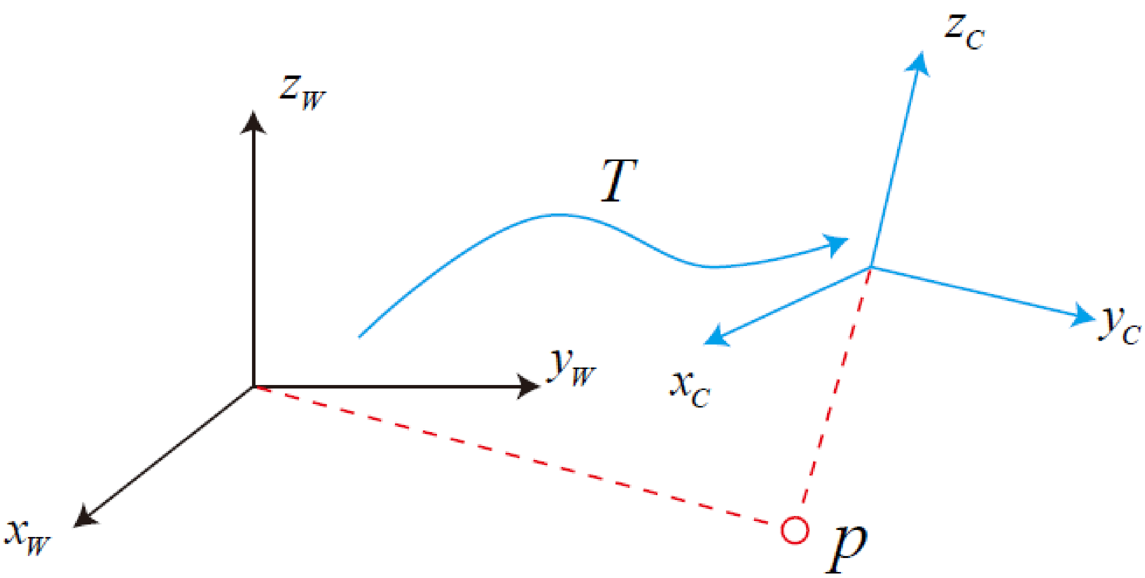
and their outer product is

$$\begin{aligned} \mathbf{a} \times \mathbf{b} &= \begin{bmatrix} i & j & k \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \mathbf{b} \\ &\triangleq \mathbf{a} \wedge \mathbf{b} \end{aligned} \tag{3}$$

which is orthogonal to the vectors $\mathbf{a}$ and $\mathbf{b}$. Here, $\mathbf{a}\wedge$ is a "Skew-symmetric" (or anti-symmetric) matrix.

Euclidean Transformation

Suppose the world coordinates (x_w, y_w, z_w) are stationary while the robot can be indicated by a moving coordinates (x_c, y_c, z_c). Consider a vector $\mathbf{p} \in \mathbb{R}^3$ in the figure below:



We can represent the vector using those two different coordinates. Assume the world coordinates are described by the basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ and the robot coordinates are described by the basis $(\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3)$, the vector $\mathbf{p}$ will not change using the representations from those two bases, i.e.,

$$[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = [\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3] \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix}. \tag{4}$$

Then, multiply the two sides with $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]^T$,

$$\begin{aligned} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} &= [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]^T [\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3] \begin{bmatrix} a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} \\ &\triangleq \mathbf{R}\mathbf{a}'. \end{aligned} \tag{5}$$

where $\mathbf{R}$ denotes the ***rotation matrix***, which contains certain special properties. We can define the set of the rotation matrix as

$$SO(n) = \{\mathbf{R} \in \mathbb{R}^{n\times n} | \mathbf{R}\mathbf{R}^T = \mathbf{I}, \det(\mathbf{R}) = 1\} \tag{6}$$

where $SO(n)$ stands for "***Special Orthogonal Group***". Basically, the rotation matrix can describe the rotation of certain object. We can also perform the inverse operation (the reversed rotation) by $\mathbf{R}^{-1}\mathbf{a}$ (equivalent to $\mathbf{R}^T\mathbf{a}$ since $\mathbf{R}$ is symmetric) to obtain the vector $\mathbf{a}'$.

Moreover, the Euclidean transformation also includes the translation of the robot's location, and thus we also need the translation vector $\mathbf{t} \in \mathbb{R}^3$ to the original expression, i.e.,

$$\mathbf{a}' = \mathbf{R}\mathbf{a} + \mathbf{t}. \tag{7}$$

Overall Transform Matrix

We can represent above Euclidean transformation into a ***homogeneous coordinates*** form, as

$$\begin{bmatrix} \mathbf{a}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix} \triangleq \mathbf{T} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix} \tag{8}$$

which allows us to represent the overall transform in a linear form, and in the rest of the notes, I will implicitly represent the homogeneous coordinates $[\mathbf{a}, 1]^T$ by $\mathbf{a}$ for simplicity. In addition, the set of the transform matrix $\mathbf{T}$ can be defined as

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{t} \in \mathbb{R}^3 \right\} \tag{9}$$

where $SE(3)$ denotes the ***special Euclidean group***.

Rotation Vector & Euler Angles

In $SE(3)$, we employ totally 16 entries to describe the Euclidean transform of the object, and in $SO(3)$ we employ 9 entries to describe the rotation with 3 DoFs, which can be redundant. A more compact representation of the rotation transform can be

- Rotation axis + rotation angle
- Euler angles

which will be introduced as follows.

Rotation Vector

Suppose the rotation axis is given by the vector $\mathbf{n}$, and the rotation angle is θ, the rotation vector can be represented by $\theta\mathbf{n}$, which relates the rotation matrix using the Rodrigues's Formula [1], as

$$\mathbf{R} = \cos\theta\mathbf{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta\mathbf{n}\wedge \tag{10}$$

where $\wedge$ is the operator that transform the vector to the anti-symmetric matrix. From this relation, we can also retrieve the rotation angle by

$$\begin{aligned} \operatorname{tr}(\mathbf{R}) &= \cos\theta \operatorname{tr}(\mathbf{I}) + (1 - \cos\theta)\operatorname{tr}\left(\mathbf{n}\mathbf{n}^T\right) + \sin\theta \operatorname{tr}(\mathbf{n}\wedge) \\ &= 3\cos\theta + (1 - \cos\theta) \\ &= 1 + 2\cos\theta \end{aligned} \tag{11}$$

and therefore we can obtain the rotation angle by

$$\theta = \cos^{-1}\left\{\frac{\operatorname{tr}(\mathbf{R}) - 1}{2}\right\} \tag{12}$$

In addition, if we rotate the rotation axis $\mathbf{n}$ by the matrix $\mathbf{R}$, the result are still $\mathbf{R}\mathbf{n} = \mathbf{n}$. Therefore, $\mathbf{n}$ is the eigenvector of the matrix $\mathbf{R}$ which corresponds to the eigenvalue 1. We can use eigen decomposition to find the rotation axis.

Euler Angles

Quaternions

C++ Eigen 3 Implementation

```
#include<iostream>
#include<Eigen/Eigen>
#include<Eigen/Core>
#include<Eigen/Geometry>
#include<cmath>
/**
 * @brief SLAM 14讲 第二章
 *
 * 复习eigen库有关知识: 向量变换
 *
 * @return int
 */

int main()
{
    //Eigen/Geometry: translate & rotate
    Eigen::Matrix3d rot_mat = Eigen::Matrix3d::Identity();

    std::cout.precision(3);
    std::cout<<"rotation matrix = \n"<<rot_mat<<std::endl;

    return 0;
}
```

1. Refer to https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula ↩