Parameterization of the 3D Rotation *

Nobuyuki Umetani

July 17, 2018

Contents

*I 'm writing this note to keep in mind what I learned when I was MSc student

1 Introduction

2 Notation and Vector Calculus

For arbitrary three dimensional vector $\mathbf{a}$, We denote skew-symmetric matrix (antisymmetric matrix) $\tilde{\mathbf{a}} \in \mathbb{R}^{3\times3}$ such as

$$\tilde{\mathbf{a}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{1}$$

using the code of exterior product we can write as:

$$\mathbf{a} \times \mathbf{b} = \tilde{\mathbf{a}}\mathbf{b} \tag{2}$$

The component of this skew - symmetric matrix can be written as:

$$\tilde{\mathbf{a}} = -\epsilon_{ijk} a_k \mathbf{e}_i \otimes \mathbf{e}_j \tag{3}$$

Here, ϵ is the emph Levi-Civita symbol.

As follows, if we change the order of this skew-symmetric matrix and vector, the sign is changed.

$$\begin{aligned} \tilde{\mathbf{a}}\mathbf{b} &= \begin{Bmatrix} -a_3 b_2 + a_2 b_3 \\ +a_3 b_1 - a_1 b_3 \\ -a_2 b_1 + a_1 b_2 \end{Bmatrix} && (4) \\ &= -\begin{Bmatrix} -b_3 a_2 + b_2 a_3 \\ +b_3 a_1 - b_1 a_3 \\ -b_2 a_1 + b_1 a_2 \end{Bmatrix} && (5) \\ &= -\tilde{\mathbf{b}}\mathbf{a} && (6) \\ \Leftrightarrow \tilde{\mathbf{a}}\mathbf{b} &= -\tilde{\mathbf{b}}\mathbf{a} && (7) \end{aligned}$$

We multiply a vector to the left hand side of a skew-matrix to get this:

$$\mathbf{a}^T \tilde{\mathbf{b}} = -\mathbf{b}^T \tilde{\mathbf{a}} \tag{8}$$

The product of two skew-matrices can be written as;

$$\tilde{\mathbf{a}}\tilde{\mathbf{b}} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} \tag{9}$$

$$= \begin{bmatrix} -a_2b_2 - a_3b_3 & a_2b_1 & a_3b_1 \\ a_1b_2 & -a_1b_1 - a_3b_3 & a_3b_2 \\ a_1b_3 & a_2b_3 & -a_1b_1 - a_2b_2 \end{bmatrix} \tag{10}$$

$$= \begin{bmatrix} a_1b_1 & a_2b_1 & a_3b_1 \\ a_1b_2 & a_2b_2 & a_3b_2 \\ a_1b_3 & a_2b_3 & a_3b_3 \end{bmatrix} - (a_1b_1 + a_2b_2 + a_3b_3) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$= \mathbf{b}\mathbf{a}^T - (\mathbf{a}^T\mathbf{b})I \tag{12}$$

Using the expression below we can do following transformation:

$$\widetilde{\tilde{\mathbf{a}}\mathbf{b}} = \begin{bmatrix} 0 & a_2b_1 - a_1b_2 & +a_3b_1 - a_1b_3 \\ -a_2b_1 + a_1b_2 & 0 & +a_3b_2 - a_2b_3 \\ -a_3b_1 + a_1b_3 & -a_3b_2 + a_2b_3 & 0 \end{bmatrix} \tag{13}$$

$$= \begin{bmatrix} a_1b_1 & a_2b_1 & a_3b_1 \\ a_1b_2 & a_2b_2 & a_3b_2 \\ a_1b_3 & a_2b_3 & a_3b_3 \end{bmatrix} - \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix} \tag{14}$$

$$= \mathbf{b}\mathbf{a}^T - \mathbf{a}\mathbf{b}^T = 2asym(\mathbf{b}^T\mathbf{a}) \tag{15}$$

$$= \{\mathbf{b}\mathbf{a}^T - (\mathbf{a}^T\mathbf{b})I\} - \{\mathbf{a}\mathbf{b}^T - (\mathbf{b}^T\mathbf{a})I\} \tag{16}$$

$$= \tilde{\mathbf{a}}\tilde{\mathbf{b}} - \tilde{\mathbf{b}}\tilde{\mathbf{a}} \tag{17}$$

Using the notation of cross product $\times$, this can be written as:

$$(\mathbf{a} \times \mathbf{b}) \times \mathbf{v} = -\mathbf{v} \times (\mathbf{a} \times \mathbf{b}) \tag{18}$$

$$= \mathbf{v} \times (\mathbf{b} \times \mathbf{a}) \tag{19}$$

$$= \mathbf{b}(\mathbf{v} \cdot \mathbf{a}) - (\mathbf{v} \cdot \mathbf{b})\mathbf{a} \tag{20}$$

$$= \{\mathbf{b}\mathbf{a}^T - \mathbf{a}\mathbf{b}^T\}\mathbf{v} \tag{21}$$

Corresponds to that holds. Let's calculate the process of applying the same outer product multiple times using this relationship.

$$\mathbf{a} \times (\mathbf{a} \times (\mathbf{a} \times \mathbf{v})) = \mathbf{a} \times \{\mathbf{a}(\mathbf{a} \cdot \mathbf{v}) - (\mathbf{a} \cdot \mathbf{a})\mathbf{v}\} \tag{22}$$

$$= (\mathbf{a} \times \mathbf{a})(\mathbf{a} \cdot \mathbf{a}) - (\mathbf{a} \cdot \mathbf{a})(\mathbf{a} \times \mathbf{v}) \tag{23}$$

$$= -|a|^2(\mathbf{a} \times \mathbf{v}) \tag{24}$$

. If you apply the same outer product $\tilde{\mathbf{a}}$ twice, you can see that only $-|a|^2$ is scalar multiplied. Therefore,

$$\tilde{\mathbf{a}}^3 = \tilde{\mathbf{a}}\tilde{\mathbf{a}}\tilde{\mathbf{a}} = -|a|^2\tilde{\mathbf{a}} \tag{25}$$

$$\tilde{\mathbf{a}}^4 = -|a|^2\tilde{\mathbf{a}}^2 \tag{26}$$

We apply this transformation recursively to obtain

$$\begin{cases} \tilde{\mathbf{a}}^{2n-1} &= (-1)^{n-1}|a|^{2(n-1)}\tilde{\mathbf{a}} \\ \tilde{\mathbf{a}}^{2n} &= (-1)^{n-1}|a|^{2(n-1)}\tilde{\mathbf{a}}^2 \end{cases} \tag{27}$$

For an orthogonal matrix (ie rotation) whose determinant is 1 $\mathbf{R}$,

$$(\mathbf{R}\mathbf{e}_1) \times (\mathbf{R}\mathbf{e}_2) = \mathbf{R}(\mathbf{e}_1 \times \mathbf{e}_2) = \mathbf{R}\mathbf{e}_3 \tag{28}$$

Using this relationship, we have

$$\widetilde{\mathbf{R}\mathbf{v}} = \mathbf{R}\tilde{v}\mathbf{R}^T \tag{29}$$

This is not limited to conversion by a general orthogonal matrix V ($V^{-1} = V^T$).

For the transformation ($\det V = -1$) including mirror image transformation, it is understood that the sign is reversed as follows.

$$\widetilde{Vu} = -V\tilde{u}V^T \tag{30}$$

A vector with such property is called *pseudo vector*.

3 Parameterization of Rotation

Three dimensional rotation specifies linear transformation from a three dimensional vector to a three dimensional vector. Therefore, it can be written as a 3x3 matrix. However, due to the orthogonality constraints, these nine components of the matrix can not take arbitrary value independent. We can, the rotation can be represented using three or four parameters.

Parameterization of rotation is important not only for reducing variables but also for interpolation. If you want to find a rotation and an intermediate rotation between a rotation, you can not average the rotation matrix. The average of the components of the two rotation matrices is no longer a rotation matrix. However, since the parameter can always be converted to a rotation matrix, it is possible to average the parameters and obtain an intermediate rotation matrix therefrom.

Typical parameters are given below.

- Euler angle
- Bryant angle
- Cartesian rotation vector
- Rodrigues parameter
- Euler parameter (Quaternion)
- Conformal Rotation Vector (CRV)

Euler Angle and Bryant Angle take a parameterization method of how much an object is rotated along the regular coordinate axes on the object and three coordinate axes on the space, whereas Cartesian rotation vector, Rodrigues parameter and Euler parameter And Conformal rotation vector parameterize the object by rotating it around a certain vector. The former is easy to understand intuitively, but since there are problems such as gimbal lock, in general the latter parameters are used inside the calculation.

Each parameter will be described below.

3.1 Cartesian Rotation Vector

Suppose the following vector $\mathbf{\Psi}$ represents a rotation rotated by θ around the unit vector $\mathbf{n}$.

$$\mathbf{\Psi} = \mathbf{n}\theta \tag{31}$$

The rotation matrix can be written as follows.

rotation matrix

$$\mathbf{R} = \mathbf{I} + \sin\theta\tilde{\mathbf{n}} + (1 - \cos\theta)\tilde{\mathbf{n}}\tilde{\mathbf{n}} \tag{32}$$

3.1.1 Infinitesimal Rotation Approximation

At $|\Psi| = \theta << 1$ it is Infinitesimal rotation. Since $\sin(\theta) \simeq \theta, (1 - \cos(\theta)) \simeq \theta^2/2$ holds at this time, the rotation matrix can be approximated as follows.

$$\begin{aligned}
\mathbf{R} \simeq \mathbf{I} + \theta\tilde{\mathbf{n}} + \frac{\theta^2}{2}\tilde{\mathbf{n}}\tilde{\mathbf{n}} &= \mathbf{I} + \tilde{\mathbf{\Psi}} + \frac{1}{2}\tilde{\mathbf{\Psi}}\tilde{\mathbf{\Psi}} && (33)\\
&= \mathbf{I} + \tilde{\mathbf{\Psi}} - \frac{1}{2}|\Psi|^2\mathbf{I} && (34)\\
&= \left(1 - \frac{1}{2}|\Psi|^2\right)\mathbf{I} + \tilde{\mathbf{\Psi}} && (35)
\end{aligned}$$

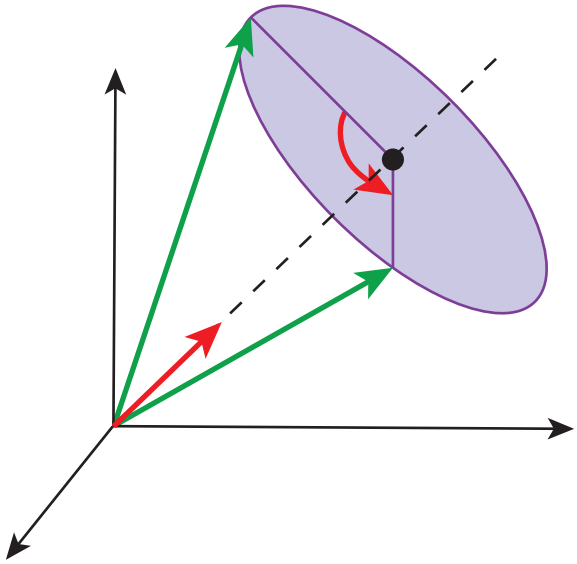

Figure 1: Vector rotation with Cartesian Rotation Vector

This approximation is second-order accuracy, but if we consider $(1 - \cos(\theta)) \simeq 0$ simply considering first-order accuracy, we can approximate as follows.

infinitesimal rotation (1st order approximation)

$$\mathbf{R} \simeq \mathbf{I} + \theta\tilde{\mathbf{n}} = \mathbf{I} + \tilde{\mathbf{\Psi}} \tag{36}$$

3.1.2 Rodrigue's rotation formula

Using $\tilde{\mathbf{a}}\tilde{\mathbf{b}} = \mathbf{a}\mathbf{b}^T - (\mathbf{a}^T\mathbf{b})\mathbf{I}$ for this expression, the rotation matrix can be transformed as follows.

$$\begin{aligned} \mathbf{R} &= \mathbf{I} + \sin\theta\tilde{\mathbf{n}} + (1 - \cos\theta)(\mathbf{n}\mathbf{n}^T - \|\mathbf{n}\|^2\mathbf{I}) && (37)\\ &= \cos\theta\mathbf{I} + \sin\theta\tilde{\mathbf{n}} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T && (38) \end{aligned}$$

This formula is called Rodrigue's rotation formula.

Rodrigue's rotation formula

$$\mathbf{R} = \cos\theta\mathbf{I} + \sin\theta\tilde{\mathbf{n}} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T \tag{39}$$

3.1.3 Rotation Matrix as a Exponential Function

The rotation matrix can also be interpreted by infinitely small sets of small rotations as follows. This can be written from the definition of the operator on the shoulder of the index as follows.

$$\mathbf{R}(\mathbf{\Psi}) = \lim_{n\to\infty}\left\{1 + \frac{1}{n}\tilde{\mathbf{\Psi}}\right\}^n = exp\tilde{\mathbf{\Psi}} \tag{40}$$

Now, if we transform the equation as follows, it turns out that the definition of rotation using this exponential function gives the same rotation matrix. However, in this case, the relational expression of $\tilde{\mathbf{\Psi}}\tilde{\mathbf{\Psi}} = -|\Psi|^2\mathbf{I}$ is used.

$$\begin{aligned}
\mathbf{R}(\mathbf{\Psi}) &= exp\tilde{\mathbf{\Psi}} = I + \frac{1}{1!}\tilde{\mathbf{\Psi}} + \frac{1}{2!}\tilde{\mathbf{\Psi}}^2 + \frac{1}{3!}\tilde{\mathbf{\Psi}}^3 + \cdots && (41)\\
&= I + \left(\frac{1}{1!} - \frac{1}{3!}\theta^2 + \frac{1}{5!}\theta^4 + \cdots + \frac{(-1)^{n-1}}{(2n-1)!}\theta^{2(n-1)} + \cdots\right)\tilde{\mathbf{\Psi}} && (42)\\
&\quad + \left(\frac{1}{2!} - \frac{1}{4!}\theta^2 + \frac{1}{6!}\theta^4 + \cdots + \frac{(-1)^{n-1}}{(2n)!}\theta^{2(n-1)} + \cdots\right)\tilde{\mathbf{\Psi}}^2 && (43)\\
&= I + \left(\frac{1}{1!}\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 + \cdots + \frac{(-1)^{n-1}}{(2n-1)!}\theta^{2n-1} + \cdots\right)\tilde{\mathbf{\Psi}}/\theta && (44)\\
&\quad + \left(\frac{1}{2!}\theta^2 - \frac{1}{4!}\theta^4 + \frac{1}{6!}\theta^6 + \cdots + \frac{(-1)^{n-1}}{(2n)!}\theta^{2n} + \cdots\right)\tilde{\mathbf{\Psi}}^2/\theta^2 && (45)\\
&= \mathbf{I} + \sin\theta\tilde{\mathbf{n}} + (1-\cos\theta)\tilde{\mathbf{n}}\tilde{\mathbf{n}} && (46)
\end{aligned}$$

3.1.4 Example of C ++ code to obtain rotation matrix from Cartesian Rotation Vector

Cartesian Rotation Vector $\mathbf{\Psi}$ gives an example of a C ++ program for obtaining a rotation matrix $\mathbf{R}$.

- mat [i * 3 + j] = R_{ij}
- vec [i] = Ψ_i

```
#include <math.h>

void SetRotMatrix_Cartesian (double mat [], const double vec []) {
  const double sqt = vec [0] * vec [0] + vec [1] * vec [1] + vec [2] * vec [2];
  if (sqt <1.0 e - 20) {// infinitesmal rotation approximation
    mat[0] = 1; mat[1] = -vec[2]; mat[2] = +vec[1];
    mat[3] = +vec[2]; mat[4] = 1; mat[5] = - vec[0];
    mat[6] = -vec[1]; mat[7] = +vec[0]; mat[8] = 1;
    return;
```

```
  }
  const double t = sqrt(sqt);
  const double invt = 1.0 / t;
  const double n [3] = {vec[0]*invt, vec[1]*invt, vec[2]*invt};
  const double c0 = cos(t);
  const double s0 = sin(t);
  mat[0*3+0] = c0 + (1 - c0) * n[0] * n[0];
  mat[0*3+1] = - n[2] * s 0 + (1-c0) * n[0]*n[1];
  mat[0*3+2] = + n[1] * s 0 + (1-c0) * n[0]*n[2];
  Mat[1*3+0] = + n[2] * s 0 + (1-c0) * n[1]*n[0];
  mat[1*3+1] = c0 + (1-c0)*n[1]*n[1];
  mat[1*3+2] = - n[0] * s 0 + (1-c0) * n[1]*n[2];
  mat[2*3+0] = - n[1] * s 0 + (1-c0) * n[2]*n[0];
  mat[2*3+1] = + n[0] * s 0 + (1-c0) * n[2]*n[1];
  mat[2*3+2] = c0 + (1-c0)*n[2]*n[2];
}
```

3.2 Rodrigues Parameters

$$\mathbf{w} = 2\tan(\theta/2)\mathbf{n} = \frac{2\tan(\theta/2)}{\theta}\mathbf{\Psi} \tag{47}$$

Rotation using **w** is as follows.

rotation matrix

$$R = I + \frac{1}{1 + 0.25|\omega|^2}\{\tilde{\omega} + \frac{1}{2}\tilde{\omega}\tilde{\omega}\} \tag{48}$$

As opposed to when using θ, terms such as $\sin(\theta)$ disappear and handling becomes easier.

3.2.1 synthesis rule

$$R(\omega_2)R(\omega_1) = R(\omega_{12}) \tag{49}$$

$$\mathbf{w}_{12} = \frac{\omega_1 + \omega_2 - \frac{1}{2}\omega_1 \times \omega_2}{1 - \frac{1}{4}\omega_1^T\omega_2} \tag{50}$$

3.2.2 C ++ code example to derive rotation matrix from Rodrigues parameter

Cartesian Rotation Vector An example of C ++ program for obtaining **R** rotation matrix from **w** is given below.

- mat [i * 3 + j] = R_{ij}
- vec [i] = w_i

3.3 Euler Parameter (Quaternion)

The Euler parameter is the amount of four variables of the cosine of the half angle of the rotation angle of the Cartesian Rotation Vector and the vector obtained by scaling the Cartesian Rotation Vector by $\sin\frac{\theta}{2}/\theta$ times as follows.

$$e_0 = \cos\frac{\theta}{2}, \quad \mathbf{e} = \mathbf{n}\sin\frac{\theta}{2} \tag{51}$$

This is equal to the quaternion whose e_0 is the real part and $\mathbf{e}$ is the imaginary part. The following relational expression holds between e_0 and $\mathbf{e}$.

$$e_0^2 = 1 - \|\mathbf{e}\|^2 \tag{52}$$

In addition, the following relational expression holds with the Rodrigues parameter.

$$\omega = \frac{2}{e_0}\mathbf{e} \tag{53}$$

The rotation matrix is as follows.

rotation matrix

$$\mathbf{R} = (2e_0^2 - 1)\mathbf{I} + 2\mathbf{e}\mathbf{e}^T + 2e_0\tilde{\mathbf{e}} \tag{54}$$

3.3.1 Deriving Euler Parameter from Rotation Matrix (ver1)

$$\mathrm{tr}\mathbf{R} = 3 \times (2\mathbf{e}_0^2 - 1) + 2|||\mathbf{e}\|^2 = 4e_0^2 - 1 \tag{55}$$

$$e_0 = \frac{1}{2}\sqrt{1 + \mathrm{tr}\mathbf{R}} \tag{56}$$

$$r_{kk} = (2e_0^2 - 1) + 2e_k^2 = \frac{1}{2}\mathrm{tr}\mathbf{R} - \frac{1}{2} + 2e_k^2 \quad \Rightarrow \quad |e_k| = \frac{1}{2}\sqrt{1 + 2r_{kk} - \mathrm{tr}\mathbf{R}} \tag{57}$$

$$vect(\mathbf{R}) = 2e_0\mathbf{e} \tag{58}$$

$$e_k = \frac{1}{2}sign(vect(\mathbf{R})_k)\sqrt{1 + 2r_{kk} - \mathrm{tr}\mathbf{R}} \tag{59}$$

3.3.2 Deriving Euler Parameter from Rotation Matrix (ver 2)

The method that is more accurate than the above method is the following method. Consider the following matrix **S**.

$$S = 4\{e_0, \mathbf{e}\}^T\{e_0, \mathbf{e}\} = 4\begin{bmatrix} e_0^2 & e_0e_1 & e_0e_2 & e_0e_3 \\ e_1e_0 & e_1^2 & e_1e_2 & e_1e_3 \\ e_2e_0 & e_2e_1 & e_2^2 & e_2e_3 \\ e_3e_0 & e_3e_1 & e_3e_2 & e_3^2 \end{bmatrix} \tag{60}$$

The matrix **S** can be created as follows.

$$\mathbf{S} = \begin{bmatrix} 1 + r_{11} + r_{22} + r_{33} & r_{32} - r_{23} & r_{13} - r_{31} & r_{21} - r_{12} \\ r_{32} - r_{23} & 1 + r_{11} - r_{22} - r_{33} & r_{12} + r_{21} & r_{13} + r_{31} \\ r_{13} - r_{31} & r_{21} + r_{12} & 1 - r_{11} + r_{22} - r_{33} & r_{23} + r_{32} \\ r_{21} - r_{12} & r_{13} + r_{31} & r_{23} + r_{32} & 1 - r_{11} - r_{22} + r_{33} \end{bmatrix} \tag{61}$$

At this time, Euler parameters can be obtained as follows.

$$S_{ii} = \max_k\{S_{kk}\} \Rightarrow \begin{cases} e_i = \frac{1}{2}\sqrt{S_{ii}} \\ e_k = \frac{S_{ii}}{4e_i} \end{cases} \tag{62}$$

Example of C ++ code to derive the Euler parameter from rotation matrix

```
void GetCRV_RotMatrix (double eparam [], const double mat []) {
  const double smat [16] = {
    1 + mat [0 * 3 + 0] + mat [1 * 3 + 1] + mat [2 * 3 + 2]
    mat [2 * 3 + 1] -mat [1 * 3 + 2],
    mat [0 * 3 + 2] -mat [2 * 3 + 0],
    mat [1 * 3 + 0] -mat [0 * 3 + 1],
    mat [2 * 3 + 1] -mat [1 * 3 + 2],
    1 + mat [0 * 3 + 0] -mat [1 * 3 + 1] -mat [2 * 3 + 2]
    mat [0 * 3 + 1] + mat [1 * 3 + 0],
    mat [0 * 3 + 2] + mat [2 * 3 + 0],
    mat [0 * 3 + 2] -mat [2 * 3 + 0],
    mat [1 * 3 + 0] + mat [0 * 3 + 1],
    1-mat [0 * 3 + 0] + mat [1 * 3 + 1] -mat [2 * 3 + 2]
    mat [1 * 3 + 2] + mat [2 * 3 + 1],
    mat [1 * 3 + 0] -mat [0 * 3 + 1],
    mat [0 * 3 + 2] + mat [2 * 3 + 0],
    mat [1 * 3 + 2] + mat [2 * 3 + 1],
    1-mat [0 * 3 + 0] -mat [1 * 3 + 1] + mat [2 * 3 + 2]
  };

  unsigned int imax;
  imax = (smat [0 * 4 + 0]> smat [1 * 4 + 1])? 0: 1;
  imax = (smat [imax * 4 + imax]> smat [2 * 4 + 2])? imax: 2;
  imax = (smat [imax * 4 + imax]> smat [3 * 4 + 3])? imax: 3;

  double eparam [4]; // euler param
```

```
  eparam [imax] = 0.5 * sqrt (smat [imax * 4 + imax]);
  for (unsigned int k = 0; k <4; k ++) {
    if (k == imax) continue;
    eparam [k] = smat [imax * 4 + k] * 0.25 / eparam [imax];
  }
```

3.4 Conformal Rotation Vector (CRV)

The Conformal Rotation Vector (CRV)

$$\mathbf{c} = 4\mathbf{n} \tan \frac{\theta}{4} \tag{63}$$

The following relational expression holds between the Euler parameter.

$$c_i = \frac{4e_i}{1 + e_0} \quad (i = 0, 1, 2, 3) \tag{64}$$

$$c_0 = \frac{1}{8}(16 - \|\mathbf{c}\|^2) \tag{65}$$

Using this, the rotation matrix can be written as

rotation matrix

$$\mathbf{R} = \frac{1}{(4 - c_0)^2}\left[(c_0^2 + 8c_0 - 16)\mathbf{I} + 2\mathbf{c}\mathbf{c}^T + 2c_0\tilde{\mathbf{c}}\right] \tag{66}$$

3.5 Bryant Angle

It is Bryant Angle's way to rotate around an orthogonal coordinate system fixed in space.

Bryant Angle (ϕ, ψ, θ) represents the rotation when the coordinate axes $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ of space are rotated by ϕ, ψ, θ in order.

1. $\mathbf{X}$ Rotate by ϕ around the axis
2. $\mathbf{Y}$ Rotate ψ around the axis
3. $\mathbf{Z}$ Rotate θ around the axis

When writing a matrix that rotates θ around the axis $\mathbf{v}$ as $\mathbf{R}(\mathbf{v}, \theta)$, the rotation matrix $\mathbf{R}$ is as follows.

rotation matrix

$$\mathbf{R} = \mathbf{R}(Z, \theta)\mathbf{R}(\mathbf{Y}, \psi)\mathbf{R}(\mathbf{X}, \phi) \tag{67}$$

3.6 Euler Angle

It is the Euler angle approach to rotate around an orthogonal coordinate axis fixed to the object.

3.6.1 Euler Angle Definition

It is assumed that the coordinate axis fixed to the substance coincides with the coordinate axis (X, Y, Z) fixed in the space as the state before rotation as (x, y, z). The Euler angle (ψ, θ, ϕ) is defined as follows

1. Rotate ψ around z. The coordinate axes are rotated to $(X, Y, Z) \rightarrow (x', y', z')$
2. Next, rotate only θ around x'. Rotated to $(x', y', z') \rightarrow (x'', y'', z'')$
3. Next, rotate ϕ around z''. Rotated to $(x'', y'', z'') \rightarrow (x, y, z)$

3.6.2 Rotation Matrix from Euler Angle

In the Euler angle, we obtained the rotation around the rotated axis. Let's consider a matrix $\mathbf{R}(\mathbf{R}_1\mathbf{v}, \theta)$ that rotates around $\mathbf{R}_1\mathbf{v}$ by θ. In this case the following holds.

$$\mathbf{R}(\mathbf{R}_1\mathbf{v}, \theta) = \mathbf{R}_1\mathbf{R}(\mathbf{v}, \theta)\mathbf{R}_1^T \tag{68}$$

Let us derive the rotation matrix of Euler angles using this.

$$\begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} = \mathbf{R}(\mathbf{Z}, \psi) \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{69}$$

$$\begin{Bmatrix} x'' \\ y'' \\ z'' \end{Bmatrix} = \mathbf{R}(\mathbf{x}', \theta) \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \tag{70}$$

$$= \mathbf{R}\left(\mathbf{R}(\mathbf{Z}, \psi)\mathbf{x}, \theta\right) \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \tag{71}$$

$$= \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}\left(\mathbf{X}, \theta\right) \mathbf{R}^T(\mathbf{Z}, \psi) \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \tag{72}$$

$$= \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}\left(\mathbf{X}, \theta\right) \mathbf{R}^T(\mathbf{Z}, \psi)\mathbf{R}(\mathbf{Z}, \psi) \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{73}$$

$$= \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}\left(\mathbf{X}, \theta\right) \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{74}$$

$$= \mathbf{R}_1 \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{75}$$

However, we defined rotation matrix $\mathbf{R}_1$ as follows.

$$\mathbf{R}_1 = \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}\left(\mathbf{X}, \theta\right) \tag{76}$$

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \mathbf{R}(\mathbf{z}'', \phi) \begin{Bmatrix} x'' \\ y'' \\ z'' \end{Bmatrix} \tag{77}$$

$$= \mathbf{R}\left(\mathbf{R}_1\mathbf{Z}, \phi\right) \begin{Bmatrix} x' \\ y' \\ z' \end{Bmatrix} \tag{78}$$

$$= \mathbf{R}_1\mathbf{R}\left(\mathbf{Z}, \phi\right)\mathbf{R}_1^T \begin{Bmatrix} x'' \\ y'' \\ z'' \end{Bmatrix} \tag{79}$$

$$= \mathbf{R}_1\mathbf{R}\left(\mathbf{Z}, \phi\right)\mathbf{R}_1^T\mathbf{R}_1 \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{80}$$

$$= \mathbf{R}_1\mathbf{R}\left(\mathbf{Z}, \phi\right) \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{81}$$

$$= \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}\left(\mathbf{X}, \theta\right)\mathbf{R}\left(\mathbf{Z}, \phi\right) \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix} \tag{82}$$

Therefore, the rotation matrix $\mathbf{R}$ becomes as follows.

rotation matrix

$$\mathbf{R} = \mathbf{R}(\mathbf{Z}, \psi)\mathbf{R}(\mathbf{X}, \theta)\mathbf{R}(\mathbf{Z}, \phi) \tag{83}$$

The rotation matrix $\mathbf{R}(\mathbf{Z}, \theta)$ that rotates by θ around the Z axis

$$\mathbf{R}(\mathbf{Z}, \theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{84}$$

Similarly, the rotation matrix $\mathbf{R}(\mathbf{X}, \theta)$ rotating by θ about the X axis is

$$\mathbf{R}(\mathbf{X}, \theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{85}$$

$$\mathbf{R} = \mathbf{R}(\mathbf{Z},\psi)\mathbf{R}(\mathbf{X},\theta)\mathbf{R}(\mathbf{Z},\phi) \tag{86}$$

$$= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{87}$$

$$= \begin{bmatrix} \cos\psi\cos\phi - \sin\psi\cos\theta\sin\phi & -\cos\psi\sin\phi - \sin\psi\cos\theta\sin\phi & \sin\psi\sin\theta \\ \sin\psi\cos\phi + \cos\psi\cos\theta\sin\phi & -\sin\psi\sin\phi + \cos\psi\cos\theta\cos\phi & -\cos\psi\sin\theta \\ \sin\theta\sin\phi & \sin\theta\cos\phi & \cos\theta \end{bmatrix} \tag{88}$$