Wrocław University of Science and Technology

Faculty of Electronics, Photonics and Microsystems

Field: Electronics and Computer Engineering

Bachelor Thesis

Title:

A comparison of parameterizations of rotation matrices in robotics

Author:
Chiemelie Nzelibe

Supervisor:
prof. dr hab. inż. Ignacy Dulęba

Grade:

Wrocław 2025

Contents

Abstract

In this bachelor thesis various parameterizations of $\mathbb{SO}(3)$ (the special orthogonal group) of rotation matrices are analyzed. The main goal is to compare a straight line motion in a given parameterization with a straight line motion (a geodesic curve) obtained with the axis-angle parameterization. The theory part of the dissertation provided complete formulas to represent rotation matrices with Euler angles, RPY angles and others as well as the axis-angle parameterization. A method to calculate a distance between two paths in $\mathbb{SO}(3)$ is also provided. Some simulations were performed to evaluate how a straight line motion in Euler angles is far from the geodesic curve generated with axis-angle parameterization.

Keywords: robotics, special orthogonal group, rotation matrix, geodesic curve, parameterization, distance between paths

Streszczenie

W niniejszej pracy inżynierskiej analizowane są różne parametryzacje $\mathbb{SO}(3)$ (specjalnej grupy ortogonalnej) macierzy rotacji. Głównym celem jest porównanie ruchu prostoliniowego w danej parametryzacji z ruchem prostoliniowym (krzywą geodezyjną) uzyskanym przy zastosowaniu parametryzacji oś-kąt. Część teoretyczna rozprawy dostarcza kompletnych wzorów do reprezentowania macierzy rotacji kątami Eulera, kątami RPY i innymi parametryzacjami, w tym także parametryzację oś-kąt. Przedstawiono sposób obliczania odległości między dwiema ścieżkami ruchu w $\mathbb{SO}(3)$. Przeprowadzono kilka symulacji, aby ocenić, w jaki sposób ruch prostoliniowy w kątach Eulera jest odległy od krzywej geodezyjnej wygenerowanej przy użyciu parametryzacji oś-kąt.

Słowa kluczowe: robotyka, specjalna grupa ortogonalna, macierz rotacji, krzywa geodezyjna, parametryzacja, odległość między ścieżkami

Acknowledgements

I would like to take this opportunity to thank my thesis supervisor, prof. dr hab. inż. Ignacy Dulęba for accepting me as his student. He always made time for me out of his busy schedule to provide me with valuable guidance, advice and made sure that I clearly understood the task at hand. He always made sure updates were sent frequently and systematically therefore the thesis was able to be done with time to spare. It was a great pleasure working with him on my Bachelor of Science thesis. Finally, I would like to thank my friends and family especially my mom, for their support without the mental support and motivations I wonder how not only the thesis would have gone but as well as all these years in university. Thank you all for the support and encouragement.

Chapter 1

Introduction

In the current times robotics has become a very important part of modern civilization and its significance is likely to increase in the future due to it's many advantages. A motion of a robot is traditionally described by evolution of its position and orientation. In this thesis we will concentrate on analysing its orientation described via parameterizations of the special orthogonal group.

Capturing the complex dynamics of three dimensional rotations has been a challenge in several scientific domains such as robotics. But due to the emergence of mathematical tools such as rotation matrices, understanding and precisely tracking orientations in three dimensional spaces have become feasible and are also necessary for a multitude of applications. In summary rotation matrices is very useful in the realm of robotics as its imperative for precise motion control and manipulation.

1.1 Background

In robotics Euler angles have long been a corner stone in rotational applications/mathematics but has always had its limitations in the case of gimbal lock which then hinders their application in certain robotic aspects/scenarios. Whereas, on the other hand, the axis-angle representation offers a more compact approach, which potentially provides a solution to challenges posed from gimbal lock. The importance of the axis-angle parameterization lies in its ability to define the geodesic curves which is vital for understanding the shortest paths between two orientations in a 3D space. Geodesic paths are important in robotics because they ensure the optimal trajectories for robot systems which likely reduces energy consumption and so much more. In recent years, advancements in robot technologies have underlined the need for a deeper understanding of rotational representations and their impact on real world robot applications.

1.2 Thesis Objective

In this thesis a straight line motion among boundary matrices will be examined for a selected parameterization and compared with the geodesic curve between the matrices obtained from the axis-angle parameterization. Therefore with the use of various calculations this study seeks to find which representation between the two holds the key to unlocking the most efficient and accurate geodesic curve/paths in 3D rotations.

1.3 Thesis Outline

The Chapter 1 provides basic introduction to the thesis topic, background knowledge regarding the research questions as well as main objective of thesis. Chapter 2 delves deeper into complex explanations and function of a rotations matrix and its different parameterization methods (for example Euler angle and also the axis-angle representation) as well as basic insight into the concept of geodesic curve. Chapter 3 then visualises the explanations in Chapter 2 using various complex calculations to further explain and show examples of the parameterization methods and to determine geodesic distance. Chapter 4 is concentrated on presentation of Matlab implementation details. In Chapter 5 some simulation results are collected. Chapter 6 summarizes the thesis which is followed by the bibliography section.

Chapter 2

Rotation Matrices in $\mathbb{SO}(3)$

2.1 A special orthogonal group in three dimensions

A group G is a finite or infinite set of elements together with a binary operation (called the group operation) that together satisfy the four fundamental properties of closure, associativity, the identity property, and the inverse property. [1].

A special orthogonal group is a set of all [2] 3×3 matrices $\boldsymbol{R}$ (real-values or in a symbolic form) with the following features:

1. the operation – a matrix multiplication,
2. the identity element – $\boldsymbol{I}_3$ (the 3×3 identity matrix),
3. the inverse element – $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$.

As a consequence, they have a determinant equal to ± 1. In robotics, we are interested with matrices with determinant $+1$, as they describe rotation matrices in Right-handed coordinate frames (-1 corresponds to the left handed coordinate systems). The $\mathbb{SO}(3)$ is not a commutative group, as generally

$$\boldsymbol{A}\,\boldsymbol{B} \neq \boldsymbol{B}\,\boldsymbol{A} \tag{2.1}$$

for $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{SO}(3)$. One should be aware of this fact and multiply matrices in an appropriate order.

The right handed rule is a convention which is used to determine the direction of some various vectors, it helps to establish a consistent and universally understood reference frame, in the three dimensional space. In this coordinate frame, based on Fig. 2.1, the thumb points in the direction of the positive z axis, index finger points in the positive direction of the x axis which is usually the first coordinate and the middle finger point in the positive direction of the y axis usually the second coordinate.[3] The orthogonality means that column (as well as rows) of any matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ are orthogonal to each other (in fact they are also orthonormal as having length equal to 1)

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} \end{bmatrix}, \tag{2.2}$$

$$\|\boldsymbol{n}\| = \|\boldsymbol{o}\| = \|\boldsymbol{a}\| = 1, \quad \boldsymbol{n} \times \boldsymbol{o} = \boldsymbol{a}. \tag{2.3}$$

In Eq. (2.3) $\|\cdot\|$ stands for an Euclidean distance, while $\times$ denotes a cross product. From Eq. (2.3) one can conclude that there exist three length constraints and three resulting from the cross product. Consequently,

$$\dim \mathbb{SO}(3) = 9 - 3 - 3 = 3, \tag{2.4}$$

and three parameters are enough to describe any rotation matrix.

Sometimes one needs to check if a (3×3) matrix is really a rotation matrix. There are two possibilities:

1. given by Eq. (2.2), (2.3), i.e. split the matrix into columns and check conditions (2.3),
2. check if $\boldsymbol{R}\,\boldsymbol{R}^T = \boldsymbol{I}_3$ and $\det(\boldsymbol{R}) = +1$.

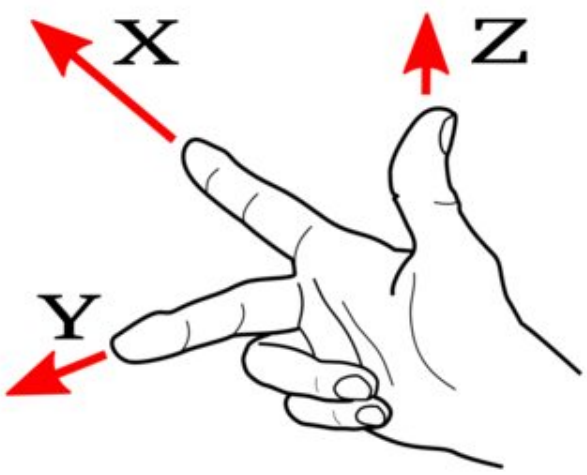


Figure 2.1 Right handed coordinate frame

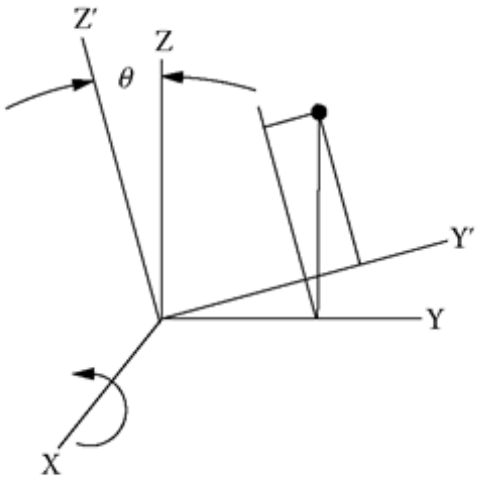
2.2 Elementary Rotations

Elementary rotations can easily be understood as a rotation matrix that explains the rotation at an axis about a specific angle (in 3D, axes are traditionally marked as X, Y, Z). These rotations enables one to precisely define the orientation of the object in 3D space by combining rotations about the different axes in a specific sequence. Below are the important equations which will be used through out this project to determine the orientation of the object :

$$\boldsymbol{R}(X,\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \tag{2.5}$$

$$\boldsymbol{R}(Y,\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \tag{2.6}$$

$$\boldsymbol{R}(Z,\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.7}$$



$$X' = X$$
$$Y' = Y\cos\theta - Z\sin\theta$$
$$Z' = Y\sin\theta + Z\cos\theta$$

In matrix form,

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Figure 2.2 Deriving elementary rotations

To better understand these rotations, Fig. 2.2 illustrates how coordinates are transformed when a rotation around the X axis is performed. Exactly the same can be done to obtain the other rotation matrices around the Y/Z axis.

[4]

2.3 Parameterizations of type ABA and ABC

Three angles (cf. Eq. (2.4)) are enough to describe the orientation of rigid body with respect to a fixed coordinate system. When rotations are performed, the order of the rotation matters because different sequences of rotations will result in a different final result. There are two main orders used to describe these different rotations known as the **ABC** order and the **ABA** order. Using these two orders we can determine exactly twelve different possibilities of order of rotations. These rotations are represented using the different axes. For example, a rotation around the X axes followed by the Y axis followed by the Z axes can be shown by XYZ represented as:

$$\boldsymbol{R} = rot(X, \alpha)(Y, \beta)(Z, \phi). \tag{2.8}$$

From the **ABC** order we have six possible rotations : XYZ, XZY, YZX, YXZ, ZXY,ZYX. The ABA order follows the same principle but we rotate on one axis twice. It is important to note that we cannot rotate around an axis twice without first rotating around a different axes. For example XXY is not plausible as it can be seen as just a rotation on X. A correct rotation would be XYX which rotates on the X axes followed by the Y axes before it rotates on the X axes again.
From the **ABA** order we have six possible rotations: XYX, XZX, YXY, YZY, ZXZ, ZYZ. With the use of the combinations we can get the final matrix for each rotation which enables us to manipulate the objects and orientations in space. We can see the final Matrix gotten from the combinations in the Matlab section.

2.4 Axis-angle parameterization

The axis-angle parameterization is an alternate method for describing any rotation in the 3D space. As seen in figure2.3, it represents a rotation matrix as a rotation of a given angle $\theta \in [0, \pi]$ around a given, three dimensional unit axis $\boldsymbol{v} = (v_x, v_y, v_z)$, $\|\boldsymbol{v}\| = 1$ [5]

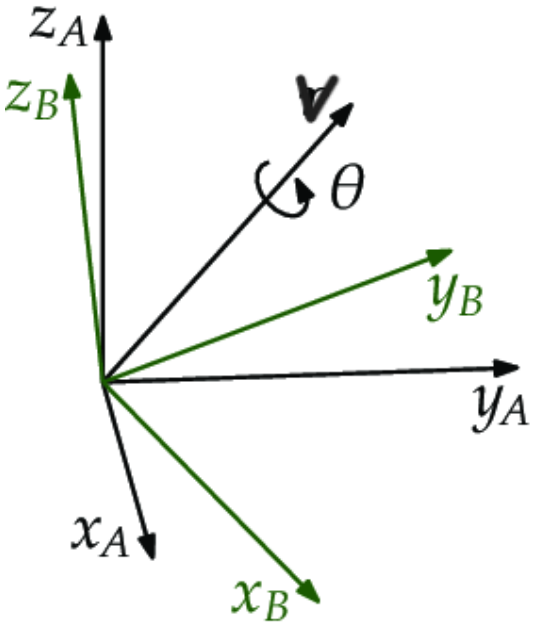


Figure 2.3 Rotation along the unit-vector $\boldsymbol{v}$ by the angle θ

There are four parameters v_x, v_y, v_z, θ. But with the constraint on the length of $\boldsymbol{v}$, only three parameters in the form $(\theta v_x, \theta v_y, \theta v_z)$ are enough to describe the parameterization. Below you can find the equation which expresses an axis angle rotation based on rodrigues formula.

$$\boldsymbol{R}(\boldsymbol{v}, \theta) = cos\theta \cdot \boldsymbol{I}_3 + sin\theta \cdot \hat{\boldsymbol{v}} + (1 - cos\theta) \cdot \boldsymbol{v}\,\boldsymbol{v}^T \tag{2.9}$$

where:

- θ is the angle of rotation,
- $\boldsymbol{v}$ is the unit axis of rotation $\boldsymbol{v} = (v_x, v_y, v_z)$,
- $\boldsymbol{I}_3$ is the (3×3) identity matrix,
- $\boldsymbol{v}\boldsymbol{v}^T$ is the outer product of v with itself, resulting in a 3 by 3 matrix,
- $\hat{\boldsymbol{v}}$ is the skew-symmetric matrix based on $\boldsymbol{v}$

$$\hat{\boldsymbol{v}} = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}. \tag{2.10}$$

A main advantage of the axis-angle representation is that it can represent the geodesic curve (i.e. the shortest trajectory) from the identity rotation matrix $\boldsymbol{I}_3$ to any given

rotation matrix $\boldsymbol{R}$. Let the the pair $\boldsymbol{v}, \theta$ represents the rotation matrix $\boldsymbol{R}^\star$. Thus the curve is given as

$$\boldsymbol{R}(t) = rot(\boldsymbol{v}, t \cdot \theta) \tag{2.11}$$

which starts at $t = 0$ with $\boldsymbol{I}_3$ and ends up for $t = 1$ at $\boldsymbol{R}^\star$.

Obviously, one can also derive a geodesic curve between any two rotation matrices $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ according to the following procedure:

1. rotate the matrices $\boldsymbol{R}_1, \boldsymbol{R}_2$ to get transformed matrix $\boldsymbol{R}_1$ equal to $\boldsymbol{I}_3$ by multiplying both of them from the left using $\boldsymbol{R}_1^T$

$$\boldsymbol{R}_1, \longrightarrow \boldsymbol{R}_1^T \boldsymbol{R}_1 = \boldsymbol{I}_3, \qquad \boldsymbol{R_2} \longrightarrow \boldsymbol{R}_1^T \boldsymbol{R}_2 = \boldsymbol{R}^\star \tag{2.12}$$

2. for $\boldsymbol{R}^\star$ calculate the pair $(\boldsymbol{v}, \theta)$, cf. Subsection 3.3
3. calculate $\boldsymbol{R}(t)$ according to Eq. (2.11),
4. calculate the geodesic curve between $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ as follows

$$\boldsymbol{R}_{12}(t) = \boldsymbol{R}_1 \cdot \boldsymbol{R}(t). \tag{2.13}$$

Chapter 3

Calculations

In this chapter some formulas are presented to get matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ for given parameterizations, and how to determine parameterization data for a given rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$.

3.1 Rotation matrices for a given parameterization

3.1.1 Rotation matrices for all ABC combinations

The elementary matrices used for all the calculations can be seen in (2.5), (2.6), (2.7). Using the formula $\boldsymbol{R}(X,\alpha)\,\boldsymbol{R}(Y,\beta)\,\boldsymbol{R}(Z,\phi)$ as shown in Eq. (2.8) the following rotation matrix is obtained

$$\boldsymbol{R}_{xyz} = \begin{bmatrix} \cos(\beta)\cos(\phi) & -\cos(\beta)\sin(\phi) & \sin(\beta) \\ \cos(\alpha)\sin(\phi)+\cos(\phi)\sin(\alpha)\sin(\beta) & \cos(\alpha)\cos(\phi)-\sin(\alpha)\sin(\beta)\sin(\phi) & -\cos(\beta)\sin(\alpha) \\ \sin(\alpha)\sin(\phi)-\cos(\phi)\cos(\alpha)\sin(\beta) & \cos(\alpha)\sin(\phi)+\cos(\alpha)\sin(\beta)\sin(\phi) & -\cos(\alpha)\cos(\beta) \end{bmatrix} \tag{3.1}$$

XZY:

$$\boldsymbol{R}_{xzy} = \begin{bmatrix} \cos(\beta)\cos(\phi) & -\sin(\beta) & \cos(\beta)\sin(\phi) \\ \sin(\alpha)\sin(\phi)+\cos(\alpha)\cos(\phi)\sin(\beta) & \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\phi)\sin(\beta)-\cos(\phi)\sin(\alpha) \\ \cos(\phi)\sin(\alpha)\sin(\beta)-\cos(\alpha)\sin(\phi) & -\cos(\beta)\sin(\alpha) & \cos(\alpha)\cos(\phi)+\sin(\alpha)\sin(\beta)\sin(\phi) \end{bmatrix} \tag{3.2}$$

ZYX:

$$\boldsymbol{R}_{zyx} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \cos(\alpha)\sin(\beta)\sin(\phi)-\cos(\phi)\sin(\alpha) & \sin(\alpha)\sin(\phi)+\cos(\alpha)\cos(\phi)\sin(\beta) \\ \cos(\beta)\sin(\alpha) & \cos(\alpha)\cos(\phi)+\sin(\alpha)\sin(\beta)\sin(\phi) & \cos(\phi)\sin(\alpha)\sin(\beta)-\cos(\alpha)\sin(\phi) \\ -\sin(\beta) & \cos(\beta)\sin(\phi) & \cos(\beta)\cos(\phi) \end{bmatrix} \tag{3.3}$$

ZXY:

$$\boldsymbol{R}_{zyx} = \begin{bmatrix} \cos(\alpha)\cos(\phi)-\sin(\alpha)\sin(\beta)\sin(\phi) & -\cos(\beta)\sin(\alpha) & \cos(\alpha)\sin(\phi)+\cos(\phi)\sin(\alpha)\sin(\beta) \\ \cos(\phi)\sin(\alpha)+\cos(\alpha)\sin(\beta)\sin(\phi) & \cos(\alpha)\cos(\beta) & \sin(\alpha)\sin(\phi)-\cos(\alpha)\cos(\phi)\sin(\beta) \\ -\cos(\beta)\sin(\phi) & \sin(\beta) & \cos(\beta)\cos(\phi) \end{bmatrix} \tag{3.4}$$

YXZ:

$$\boldsymbol{R}_{yxz} = \begin{bmatrix} \cos(\alpha)\cos(\phi)+\sin(\alpha)\sin(\beta)\sin(\phi) & \cos(\phi)\sin(\alpha)\sin(\beta)-\cos(\alpha)\sin(\phi) & \cos(\beta)\sin(\alpha) \\ \cos(\beta)\sin(\phi) & \cos(\beta)\cos(\phi) & -\sin(\beta) \\ \cos(\alpha)\sin(\beta)\sin(\phi)-\cos(\phi)\sin(\alpha) & \sin(\alpha)\sin(\phi)+\cos(alpha)\cos(\phi)\sin(\beta) & \cos(\alpha)\cos(\beta) \end{bmatrix} \tag{3.5}$$

YZX:

$$\boldsymbol{R}_{yzx} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \sin(\alpha)\sin(\phi) - \cos(\alpha)\cos(\phi)\sin(\beta) & \cos(\phi)\sin(\alpha) + \cos(\alpha)\sin(\beta)\sin(\phi) \\ \sin(\beta) & \cos(\beta)\cos(\phi) & \cos(\beta)\sin(\phi) \\ -\cos(\beta)\sin(\alpha) & \cos(\alpha)\sin(\phi) + \cos(\phi)\sin(\alpha)\sin(\beta) & \cos(\alpha)\cos(\phi) - \sin(\alpha)\sin(\beta)\sin(\phi) \end{bmatrix} \tag{3.6}$$

3.1.2 Rotation matrices for all ABA combinations

ZYZ:

$$\boldsymbol{R}_{zyz} = \begin{bmatrix} \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) & -\cos(\phi)\sin(\alpha) - \cos(\alpha)\cos(\beta)\sin(\phi) & \cos(\alpha)\sin(\beta) \\ \cos(\alpha)\sin(\phi) + \cos(\beta)\cos(\phi)\sin(\alpha) & \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) & \sin(\alpha)\sin(\beta) \\ -\cos(\phi)\sin(\beta) & \sin(\beta)\sin(\phi) & \cos(\beta) \end{bmatrix} \tag{3.7}$$

ZXZ:

$$\boldsymbol{R}_{zxz} = \begin{bmatrix} \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) & -\cos(\alpha)\sin(\phi) - \cos(\beta)\cos(\phi)\sin(\alpha) & \sin(\alpha)\sin(\beta) \\ \cos(\phi)\sin(\alpha) + \cos(\alpha)\cos(\beta)\sin(\phi) & \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) & -\cos(\alpha)\sin(\beta) \\ \sin(\beta)\sin(\phi) & \cos(\phi)\sin(\beta) & \cos(\beta) \end{bmatrix} \tag{3.8}$$

XYX:

$$\boldsymbol{R}_{xyx} = \begin{bmatrix} \cos(\beta) & \sin(\beta)\sin(\phi) & \cos(\phi)\sin(\beta) \\ \sin(\alpha)\sin(\beta) & \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) & -\cos(\alpha)\sin(\phi) - \cos(\beta)\cos(\phi)\sin(\alpha) \\ -\cos(\alpha)\sin(\beta) & \cos(\phi)\sin(\alpha) + \cos(\alpha)\cos(\beta)\sin(\phi) & \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) \end{bmatrix} \tag{3.9}$$

XZX:

$$\boldsymbol{R}_{xzx} = \begin{bmatrix} \cos(\beta) & -\cos(\phi)\sin(\beta) & \sin(\beta)\sin(\phi) \\ \cos(\alpha)\sin(\beta) & \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) & -\cos(\phi)\sin(\alpha) - \cos(\alpha)\cos(\beta)\sin(\phi) \\ \sin(\alpha)\sin(\beta) & \cos(\alpha)\sin(\phi) + \cos(\beta)\cos(\phi)\sin(\alpha) & \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) \end{bmatrix} \tag{3.10}$$

YXY:

$$\boldsymbol{R}_{yxy} = \begin{bmatrix} \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) & \sin(\alpha)\sin(\beta) & \cos(\alpha)\sin(\phi) + \cos(\beta)\cos(\phi)\sin(\alpha) \\ \sin(\beta)\sin(\phi) & \cos(\beta) & -\cos(\phi)\sin(\beta) \\ -\cos(\phi)\sin(\alpha) - \cos(\alpha)\cos(\beta)\sin(\phi) & \cos(\alpha)\sin(\beta) & \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) \end{bmatrix} \tag{3.11}$$

YZY:

$$\boldsymbol{R}_{yzy} = \begin{bmatrix} \cos(\alpha)\cos(\beta)\cos(\phi) - \sin(\alpha)\sin(\phi) & -\cos(\alpha)\sin(\beta) & \cos(\phi)\sin(\alpha) + \cos(\alpha)\cos(\beta)\sin(\phi) \\ \cos(\phi)\sin(\beta) & \cos(\beta) & \sin(\beta)\sin(\phi) \\ -\cos(\alpha)\sin(\phi) - \cos(\beta)\cos(\phi)\sin(\alpha) & \sin(\alpha)\sin(\beta) & \cos(\alpha)\cos(\phi) - \cos(\beta)\sin(\alpha)\sin(\phi) \end{bmatrix} \tag{3.12}$$

3.1.3 Rotation matrices for an axis-angle parameterization

In order to determine the rotation matrix based on axis angle representation given the angle and the vector v, an equation is used based Rodrigues formula which can be seen in (2.9). Expanding this equation shows the rotation matrix of the axis angle representation as follows:

$$\boldsymbol{R} = \boldsymbol{Rot}(\boldsymbol{v}, \theta) = \begin{bmatrix} u_\theta \cdot v_x^2 + cos(\theta) & u_\theta \cdot v_x \cdot v_y - v_z \cdot sin(\theta) & u_\theta \cdot v_x \cdot v_z + v_y \cdot sin(\theta) \\ u_\theta \cdot v_x \cdot v_y + v_z \cdot sin(\theta) & u_\theta \cdot v_y^2 + cos(\theta) & u_\theta \cdot v_y \cdot v_z - v_x \cdot sin(\theta) \\ u_\theta \cdot v_x \cdot v_z - v_y \cdot sin(\theta) & u_\theta \cdot v_y \cdot v_z + v_x \cdot sin(\theta) & u_\theta \cdot v_z^2 + cos(\theta) \end{bmatrix} \tag{3.13}$$

where:

$$u_\theta = 1 - cos(\theta). \tag{3.14}$$

3.2 Determining angles for a given rotation matrix

Given any rotation matrix (3.1)- (3.12) one can determine angles (α, β, ϕ) corresponding to the matrix. The angles will be determined for Euler angles, Roll-Pitch-Yaw angles and axis-angle parameterization. For other parameterizations appropriate formulas are easy to derive following the method applied below.

3.2.1 Euler angles for a given rotation matrix

In robotics the ZYZ parameterization, known as Euler angles, is a frequently used. Therefore the following example will be based of this particular combination. Based on the equation (3.7) which shows the final matrix of the ZYZ Euler angle order, looking at $r_{33} = \cos(\beta)$ value which depends only on one angle, one can conclude that this angle can be calculated fairly easily. We start with boundary values $r_{33} = \pm 1$ which correspond to some special cases known as singularities.

$$r_{33} = cos(\beta) = 1 \quad \Rightarrow \quad \beta = 0. \tag{3.15}$$

When this value of β is substituted into the matrix (3.7) we arrive at

$$\boldsymbol{R} = \begin{bmatrix} cos(\alpha+\phi) & -sin(\alpha+\phi) & 0 \\ sin(\alpha+\phi) & cos(\alpha+\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.16}$$

It appears that it is possible to calculate uniquely only the sum $\alpha + \phi$ of angles based on

$$\alpha + \phi = atan2(r_{21}, r_{23}), \tag{3.17}$$

but not any of them. Consequently, there is an infinite number of admissable triples (α, β, ϕ) for $\beta = 0$. In (3.17) the function $atan2(\sin(\xi), \cos(\xi))$ was used. It gives a unique value of $\xi \in [0, 2\pi)$ when sinus and cosinus of the angle are given.

The same applies for when $r_{33} = -1$ which can be seen below:

$$r_{33} = cos(\beta) = -1 \quad \Rightarrow \quad \beta = \pi. \tag{3.18}$$

When substituted into matrix (3.7) we then get the following:

$$\boldsymbol{R} = \begin{bmatrix} -cos(\alpha-\phi) & -sin(\alpha-\phi) & 0 \\ sin(\alpha-\phi) & cos(\alpha-\phi) & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{3.19}$$

Similar to previous case in (3.17), it appears in this case that it is possible to calculate uniquely only the difference $\alpha - \phi$ of angles based on

$$\alpha - \phi = atan2(r_{21}, r_{23}), \tag{3.20}$$

but not any of them. Which leads to an infinite number of admissible triples (α, β, ϕ) for $\beta = \pi$. In (3.20) the function $atan2(\sin(\xi), \cos(\xi))$ was used. It gives a unique value of $\xi \in [0, 2\pi)$ when sinus and cosinus of the angle are given.

If these singularities are prevented we then have the regular case where $r_{33} \in (-1, 1)$,

and in these situation one can calculate the different angles separately, but its important to take note that in the regular case we then have two values of beta (β) where as:

$$\beta_1 \in (0, \pi), \qquad \beta_2 = 2\pi - \beta_1 \tag{3.21}$$

we have two values for beta due to the periodicity of angles, which is a case where multiple angles represent the same orientation but in this case we set the maximum angle to be 2π, therefore we can only have two angles representing the same orientation. Based on the explanation above, we can then determine the values for two different sets of angles with the following equations:

$$\beta_1 = \arccos(r33), \quad \alpha_1 = atan2(\frac{r_{13}}{sin(\beta_1)}, \frac{r_{23}}{sin(\beta_1)}), \quad \phi_1 = atan2(\frac{r_{31}}{-sin(\beta_1)}, \frac{r_{32}}{sin(\beta_1)}), \tag{3.22}$$

$$\beta_2 = 2\pi - \beta_1, \quad \alpha_2 = atan2(\frac{r_{13}}{sin(\beta_2)}, \frac{r_{23}}{sin(\beta_2)}), \quad \phi_2 = atan2(\frac{r_{31}}{-sin(\beta_2)}, \frac{r_{32}}{sin(\beta_2)}). \tag{3.23}$$

3.2.2 RPY angles for a given rotation matrix

Another parameterization which encountered frequently would be the ZYX (roll-pitch-yaw angles). Like the previous example the angles can also be gotten from the equation in (3.3). From referenced equation it can be observed that -sin(β) in r31 is the only angle so the angle can easily be calculated where:

$$r_{31} = -sin(\beta) = 1 \quad \Rightarrow \quad \beta = -\pi/2. \tag{3.24}$$

When this value of β is substituted into the matrix (3.3) we arrive at

$$\boldsymbol{R} = \begin{bmatrix} 0 & -sin(\alpha - \phi) & cos(\alpha - \phi) \\ 0 & cos(\alpha - \phi) & -sin(\alpha - \phi) \\ 1 & 0 & 0 \end{bmatrix} \tag{3.25}$$

It appears in this case that it is possible to calculate uniquely only the difference $\alpha - \phi$ of angles based on

$$\alpha - \phi = atan2(r_{12}, r_{22}), \tag{3.26}$$

but not any of them. Which leads to an infinite number of admissible triples (α, β, ϕ) for $\beta = -\pi/2$. In (3.26) the function $atan2(\sin(\xi), \cos(\xi))$ was used. It gives a unique value of $\xi \in [0, 2\pi)$ when sinus and cosinus of the angle are given.

The same applies for when r31=-1 which can be seen below:

$$r_{31} = -sin(\beta) = -1 \quad \Rightarrow \quad \beta = \pi/2. \tag{3.27}$$

When substituted into matrix (3.3) we then get the following:

$$\boldsymbol{R} = \begin{bmatrix} 0 & -sin(\alpha + \phi) & cos(\alpha + \phi) \\ 0 & cos(\alpha + \phi) & -sin(\alpha + \phi) \\ -1 & 0 & 0 \end{bmatrix} \tag{3.28}$$

It appears that it is possible to calculate uniquely only the sum $\alpha + \phi$ of angles based on

$$\alpha + \phi = atan2(r_{12}, r_{22}), \tag{3.29}$$

but not any of them. Consequently, there is an infinite number of admissible triples (α, β, ϕ) for $\beta = \pi/2$. In (3.31) the function $atan2(\sin(\xi), \cos(\xi))$ was used. It gives a unique value of $\xi \in [0, 2\pi)$ when sinus and cosinus of the angle are given.
If these singularities are prevented we then have the regular case where $r_{31} \in (-1, 1)$, and in these situation one can calculate the different angles separately, it is also important to note that in the regular case we have two different values for β where:

$$\beta_1 \in (0, \pi/2), \qquad \beta_2 = \pi - \beta_1. \tag{3.30}$$

Below are the equations which explain this particular phenomenon:

$$\beta_1 = \arcsin(-r_{31}), \quad \alpha_1 = atan2(\frac{r_{11}}{sin(\beta 1)}, \frac{r_{21}}{sin(\beta_1)}), \quad \phi_1 = atan2(\frac{r_{32}}{-sin(\beta 1)}, \frac{r_{33}}{sin(\beta_1)}), \tag{3.31}$$

$$\beta_2 = \pi - \beta_1, \quad \alpha_2 = atan2(\frac{r_{11}}{sin(\beta_2)}, \frac{r_{21}}{sin(\beta_2)}), \quad \phi_2 = atan2(\frac{r_{32}}{-sin(\beta_2)}, \frac{r_{33}}{sin(\beta_2)}). \tag{3.32}$$

3.3 The axis-angle pair for a given rotation matrix

As seen in (3.13) we can easily find the rotation matrix given the axis angle. This section will show how to calculate the axis angle given a rotation matrix. Firstly when the rotation matrix is given one can start off by calculating the absolute angle of the rotation matrix which can be seen below:

$$\theta = \cos^{-1} \frac{tr(\boldsymbol{R}) - 1}{2} \tag{3.33}$$

where the trace operator is defined as follows

$$tr(\boldsymbol{R}) = \sum_{i=1}^{3} r_{ii}. \tag{3.34}$$

One can then proceed to calculate the axis (v_x, v_y, v_z) using the following formulas below:

$$v_x = \frac{r_{32} - r_{23}}{2\, sin(\theta)}, \quad v_y = \frac{r_{13} - r_{31}}{2\, sin(\theta)}, \quad v_z = \frac{r_{21} - r_{12}}{2\, sin(\theta)}. \tag{3.35}$$

The above case is the regular case where as $\theta \neq 0$ and $\theta \neq \pi$, In a situation where this is the case we come across the singularities that can be encountered in the axis angle representation. When $\theta = 0$, we assume that the axis arbitrary and from (3.35) we can see that the expression will be undefined. In the case of π :

$$cos(\theta) = -1, \quad \boldsymbol{R} = -\boldsymbol{I}_3 + 2 \cdot \boldsymbol{v}\, \boldsymbol{v}^T \tag{3.36}$$

$$v_x = \pm\sqrt{\frac{r_{11} + 1}{2}} \quad v_y = \pm\sqrt{\frac{r_{22} + 1}{2}} \quad v_z = \pm\sqrt{\frac{r_{33} + 1}{2}}. \tag{3.37}$$

3.4 A distance from a straight line in a selected parameterization from the geodesic curve

This section will show some calculations to be taken in order to find a distance from a straight line in a selected parameterization to the geodesic curve joining two rotation matrices. At the end of the section one should be able to make multiple comparisons of different parameterization methods and come to a conclusion.

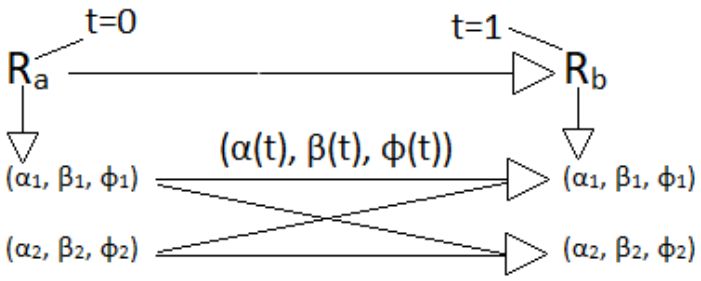


Figure 3.1 Transition from $\boldsymbol{R}_a$ to $\boldsymbol{R}_b$

3.4.1 Generating random matrices

At first we need to generate two matrices belonging to the $\mathbb{SO}(3)$ group being the initial, $\boldsymbol{R}_a$, and the final $\boldsymbol{R}_b$ matrix of the planning. The simplest method is to take any parameterization, generate angles (α, β, γ) randomly and using appropriate formulas for the parameterization (3.1)-(3.12) to get boundary matrices. In practice we take integer values for the angles from the range $\{0, 360°\}$. We also assume that the time it takes to complete the transition between the matrices is one unit of time therefore, the initial matrix $\boldsymbol{R}_a$ has its time set to 0 while the final one $\boldsymbol{R}_b$ has its time set to 1.

3.4.2 Generating the geodesic curve between boundary matrices

The geodesic curve is generated as follows:

- shift the matrices $\boldsymbol{R}_a$, $\boldsymbol{R}_b$, by multiplying both of them from the left by $\boldsymbol{R}_a^T$, to get transformed $\boldsymbol{R}_a$ matrix equal to the identity matrix $\boldsymbol{I}_3$

$$(\boldsymbol{R}_a, \boldsymbol{R}_b) \quad \rightarrow \quad (\boldsymbol{R}_a^T \boldsymbol{R}_a, \boldsymbol{R}_a^T \boldsymbol{R}_b) = (\boldsymbol{I}_3, \boldsymbol{R}_c). \tag{3.38}$$

- calculated the axis-angle $(\boldsymbol{v}, \theta)$ parameterization (3.33), (3.35) for the matrix $\boldsymbol{R}_c$
- calculate the geodesic curve

$$\boldsymbol{R}_{geo}(t) = \boldsymbol{R}_a \cdot \boldsymbol{Rot}(\boldsymbol{v}, t \cdot \theta). \tag{3.39}$$

  using Eq. (3.13)

3.4.3 Generating a straight line in a parameterization

The process is composed of the following the steps (for the sake of simplicity we assume that $\boldsymbol{R}_a$ and $\boldsymbol{R}_b$ are a regular matrices):

1. Select a given parameterization.
2. For the matrix $\boldsymbol{R}_a$ calculate two sets of angles, cf. Section 3.2, $(\alpha_1^a, \beta_1^a, \phi_1^a)$, $(\alpha_2^a, \beta_2^a, \phi_2^a)$.
3. The previous step is repeated for the matrix $\boldsymbol{R}_b$ to get $(\alpha_1^b, \beta_1^b, \phi_1^b)$, $(\alpha_2^b, \beta_2^b, \phi_2^b)$.

We can have four different transitions from one set of angles (corresponding to $\boldsymbol{R}_a$) to the other corresponding to $\boldsymbol{R}_b$, cf. Fig. 3.1. For example one can move from $(\alpha_1^a, \beta_1^a, \phi_1^a)$ to $(\alpha_1^b, \beta_1^b, \phi_1^b)$ or from $(\alpha_1^a, \beta_1^a, \phi_1^a)$ to $(\alpha_2^b, \beta_2^b, \phi_2^b)$.

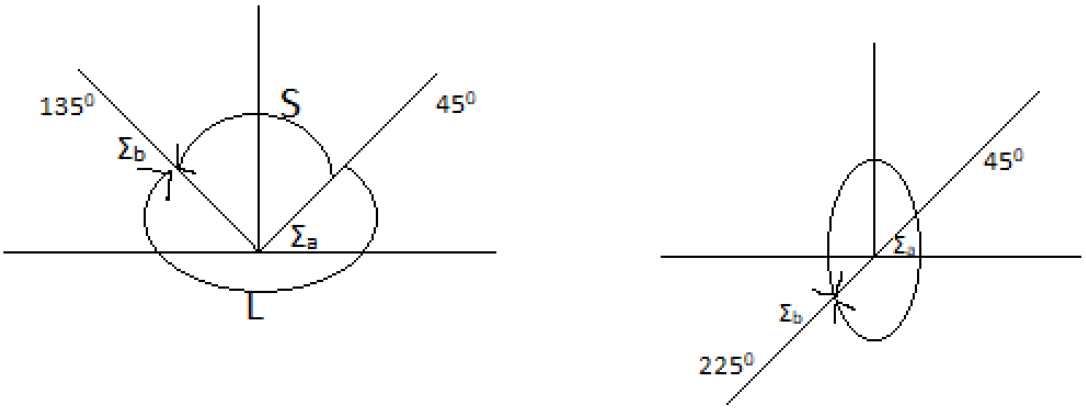


Figure 3.2 Paths from Σ_a to Σ_b

α	β	ϕ
S	S	S
S	S	L
S	L	S
S	L	L
L	S	S
L	S	L
L	L	S
L	L	L

Tabela. 3.1 Combinations of short/long motion among angles

Now, we need to find time evolution of angles $(\alpha(t), \beta(t), \phi(t))$ between boundary sets of angles (the initial marked with the supersript a, and the final one marked with the superscript b). It is important to note that all angles are normalized, that is the angles belong to the range $[0, 2\,\pi]$. As we have three angles, $(\alpha(t), \beta(t), \phi(t))$ a process of getting their time evolution will be split into three parts when each angle is considered separately.

One can observe in Fig. 3.2 that depending on the value of the angles the path taken to reach Σ_b from Σ_a could either be a short (denotes as S) or could be a long path (denotes as L).

Consequently we have four scenarios of motion between two pairs of boundary angles and for each of them eight scenarios for short and long transitions, as given in Table 3.1

Let us derive equations describing shorter/longer motion from initial σ_a to the final σ_b (notice that both angles are normalized, i.e. $\in [0, 2\pi)$). Based on the diagram, we have only one possibility for calculating the equations when $\Sigma_b > \Sigma_a$ which is the case where $|\alpha_b - \alpha_a| \leq \pi$. Then in the case that $|\alpha_b - \alpha_a| \geq \pi$ we need to satisfy this condition as well. These equations can be found below with further explanations.
When $\Sigma_b > \Sigma_a$:

if $|\alpha_b - \alpha_a| \leq \pi$

$$\sigma(t) = \sigma_a + t(|\sigma_b - \sigma_a|) \tag{3.40}$$

$$\sigma(t) = \sigma_a - t(2\pi - |\sigma_b - \sigma_a|) \tag{3.41}$$

As seen in the equations above, when $|\alpha_b - \alpha_a| \leq \pi$, (3.40) then represents the shorter path from $\sigma_a \Rightarrow \sigma_b$ while (3.41) represents the longer path from $\sigma_a \Rightarrow \sigma_b$. What then are

the equations when $|\alpha_b - \alpha_a| \geq \pi$. Firstly note that when this is the case the diagram above changes and the counter clockwise path becomes the longer path while the clockwise path becomes the shorter path, therefore the equations remain the same but due to the longer and shorter paths changing, (3.40) then represents the longer path from $\sigma_a \Rightarrow \sigma_b$ while (3.41) represents the shorter path from $\sigma_a \Rightarrow \sigma_b$. These sets of equations represent when the end point angle $\sigma_b > \sigma_a$ the starting point angle. Therefore, we need to satisfy the condition where the end point angle $\sigma_b < \sigma_a$ the starting point angle. In this case we also have two conditions similar to when $\Sigma b > \Sigma a$. To better visualize when $\Sigma_b < \Sigma_a$, you can assume that the diagram on the left most side of 3.2 has its Σ_a as 135° and the Σ_b has its as 45°, therefore in this case to move from Σa to Σ_b the arrows would be heading in the opposite direction. With that in mind, we can derive the formulas for this.
When $\Sigma_b < \Sigma_a$:
if $|\alpha_b - \alpha_a| \leq \pi$

$$\sigma(t) = \sigma_a - t(|\sigma_b - \sigma_a|) \tag{3.42}$$

$$\sigma(t) = \sigma a + t(2\pi - |\sigma b - \sigma a|) \tag{3.43}$$

From the equations above one can observe that when $|\alpha_b - \alpha_a| \leq \pi$, (3.42) represents the shorter path from $\sigma a_{\Rightarrow} \sigma_b$ and (3.43) represents the longer path therefore when $|\alpha_b - \alpha_a| \geq \pi$ then the equations remain the same but the (3.42) represents the longer path and (3.43) represents the shorter path. With these equations we can find the distances from any of the angles in $\boldsymbol{R}_a$ to the angles in $\boldsymbol{R}_b$. In summary one can obtain two different path for each angle as shown in figure 3.1. where S represents short path final angle an L represents long path final angle.

We can also observe that we can choose to move through the long path for each angle or the short path since eventually it ends in the same position therefore, we can combine the paths from the different angles and end up with multiple paths which we can then compare to see which combination is the shortest path. Since we have two paths for 3 different angles we can obtain $2^3(8)$ different combinations for this and since we have 4 different methods of transitioning, in order to check and compare it all we'll have 32 different comparisons from which we can select the best.
Notice that intermediate angles $\sigma(t)$ in Eq. (3.40)-(3.41),(3.42)-(3.43) should also be normalized. For example, while transferring from 350° to 30° the angle 360° is to be replaced by 0° while 380° by 20°.

3.4.4 Evaluation

Having derived the time evolution of angles $(\alpha(t), \beta(t), \phi(t))$ now we are in a position to introduce a measure between a geodesic curve and a matrices corresponding to a straight line in a given parameterization. The procedure is the following:

- Select an integer N, for example 100,
- discretize the time horizon $t \in [0, 1]$ into N points

$$t_i = \frac{1}{N-1} \quad i = 0, \ldots, N, \tag{3.44}$$

- for each discrete time point t_i calculate angles and corresponding matrix $\boldsymbol{R}(t_i)$

$$(\alpha(t_i), \beta(t_i), \phi(t_i)) \quad \Rightarrow \quad \boldsymbol{R}(t_i). \tag{3.45}$$

- for each discrete time point t_i calculate a matrix from the geodesic curve using Eq. (3.39) to get $\boldsymbol{R}_{geo}(t_i)$
- calculate the distance between matrices at t_i

$$d(t_i) = \|\boldsymbol{R}(t_i) - \boldsymbol{R}_{geo}(t_i)\|, \tag{3.46}$$

  where $\|\cdot\|$ denotes any matrix norm.
- calculate and output the total distance:

$$D = \sum_{i=0}^{N} d(t_i). \tag{3.47}$$

Chapter 4

Matlab Implementation

In this chapter we'll be looking at how the data obtained and calculated in Chapter 2 and Chapter 3 will be implemented in Matlab in order to visualize these comparisons and actually come to a conclusion based on calculated statistical data done in this project and by generalizing as well.

4.1 Select Parameterization Method

4.1.1 type ABA order of describing matrix

```
%euler (z,y,z)
%syms t_a t_b t_p real
t_a=30;t_b=50;t_p=70;
Rz1 = [cosd(t_a), -sind(t_a), 0; sind(t_a), cosd(t_a), 0; 0, 0, 1];
Ry = [cosd(t_b), 0, sind(t_b); 0, 1, 0; -sind(t_b), 0, cosd(t_b)];
Rz = [cosd(t_p), -sind(t_p), 0; sind(t_p), cosd(t_p), 0; 0, 0, 1];
R=Rz1*Ry*Rz
```

Now, in this short snippet we present a code implementing the ZYZ Euler angles parameterization following the format of type ABA as explained in Chapter 2. It will be used for comparisons with the axis-angle parameterization as we go further. The all other parameterizations of type ABA follow the same format with appropriately utilized basic elementary rotations. Using Matlab this can also be simplified as there are built in methods to easily get the Euler ZYZ matrix but we've used this for the sake of actually understanding what is happening to ensure one can first do it without relying on the built in codes. In the code there are some values $t_a/t_b/t_p$ which represent the $\alpha/\beta/\phi$ angles, respectively. It should be noted that the angle data are presented in grads, but sinus/cosinus functions accept data in radians. Therefore *sind/cosd* functions, at first transform their arguments form grads to radians and then apply sin / cos functions.

Below there are some examples following the above code example for determining either ZXZ or RPY matrix:

```
%euler(z,x,z)
Rz1 = [cos(t_a), -sin(t_a), 0; sin(t_a), cos(t_a), 0; 0, 0, 1];
Rx = [1, 0, 0; 0, cos(t_b), -sin(t_b); 0, sin(t_b), cos(t_b)];
Rz = [cos(t_p), -sin(t_p), 0; sin(t_p), cos(t_p), 0; 0, 0, 1];
R=Rz1*Rx*Rz;
```

```
%RPY(z,y,x)
Rx = [1, 0, 0; 0, cos(t_p), -sin(t_p); 0, sin(t_p), cos(t_p)];
Ry = [cos(t_b), 0, sin(t_b); 0, 1, 0; -sin(t_b), 0, cos(t_b)];
Rz = [cos(t_a), -sin(t_a), 0; sin(t_a), cos(t_a), 0; 0, 0, 1];
R=Rz*Ry*Rx;
```

4.1.2 All sets of angles for a given rotation matrix

This section explains the procedure for determining the 2nd set of angles based on some calculations and formulas.

```
RA=[-0.2795,-0.6941,0.6634;0.9237,-0.0058,0.3830;-0.2620,0.7198,0.6428];
%1st set of euler angles used to generate matrix
alpha1A=30;  beta1A=50;  phi1A=70;
%calculated 2nd set of euler angles
beta2A=360-beta1A;
alpha2A=mod(atan2d(0.3830/sind(beta2A), 0.6634/sind(beta2A)),360);
phi2A=mod(atan2d(0.7198/sind(beta2A),-0.2620/(-sind(beta2A))),360);

%Generate 2nd Rotation Matrix from order_ABC.m/ order_ABA.m
RB=[-0.9769,0.1285,0.1710; 0.1955, 0.8608, 0.4698;-0.0868,0.4924,-0.8660];
%1st set of euler anlges used to generate matrix
alpha1B=70;  beta1B=150;  phi1B=80;
%calculated 2nd set of euler angles
beta2B=360-beta1B;
alpha2B=mod(atan2d(0.4698/sind(beta2B), 0.1710/sind(beta2B)),360);
phi2B=mod(atan2d(0.4924/sind(beta2B),-0.0868/(-sind(beta2B))),360);
```

In this section we define the angles which were used as follows from the previous explanation to obtain the rotation matrix. Notice we also defined the starting point rotation matrix R_A and the final rotation matrix R_B. Now that the angles have been defined based on the information which has been explained in (3.23) in Chapter 3 we then use the formulas seen in the code to obtain the 2nd sets of angles which when used to generate a rotation matrix give us the same exact rotation points. In the code above, one can observe that we have used mod(360) which is a function in Matlab which of course calculated the modulus, in simpler terms it helps to ensure our calculated angles stay within the range of ($[0-2\pi$) expressed in radians). In summary we obtained the two sets of angles for both rotation matrices R_A and R_B and with these angles we can begin to run simulations to observe which path is more optimal for transitioning from R_A to R_B as seen in Fig. 3.1.

4.2 Transition from starting to end point rotation matrix

4.2.1 Short and long paths for α, β, ϕ, angles

In this section we will dive deep into the code explaining how both the short and long paths for each angle can be obtained and show some snippets of the result from such code.

```
    N=100;
for  i=0:(N-1)
    t=(1/(N-1))*i;
if  alpha1B>alpha1A
```

```
    if abs(alpha1B-alpha1A)<=180
       s_a(i+1)=mod(alpha1A+t*(abs(alpha1B-alpha1A)),360);
       l_a(i+1)=mod(alpha1A-t*(360-abs(alpha1B-alpha1A)),360);
    elseif abs(alpha1BA-alpha1A)>180
        l_a(i+1)=mod(alpha1A+t*(abs(alpha1B-alpha1A)),360);
       s_a(i+1)=mod(alpha1A-t*(360-abs(alpha1B-alpha1A)),360);
    end
elseif  alpha1B<alpha1A
     if abs(alpha1B-alpha1A)<=180
       s_a(i+1)=mod(alpha1A-t*(abs(alpha1B-alpha1A)),360);
       l_a(i+1)=mod(alpha1A+t*(360-abs(alpha1B-alpha1A)),360);
     elseif abs(alpha1B-alpha1A)>180
        l_a(i+1)=mod(alpha1A-t*(abs(alpha1B-alpha1A)),360);
       s_a(i+1)=mod(alpha1A+t*(360-abs(alpha1B-alpha1A)),360) ;
     end
end

if beta1B>beta1A
    if abs(beta1B-beta1A)<=180
       s_b(i+1)=mod(beta1A+t*(abs(beta1B-beta1A)),360);
       l_b(i+1)=mod(beta1A-t*(360-abs(beta1B-beta1A)),360);
    elseif abs(beta1B-beta1A)>180
        l_b(i+1)=mod(beta1A+t*(abs(beta1B-beta1A)),360);
       s_b(i+1)=mod(beta1A-t*(360-abs(beta1B-beta1A)),360);
    end
elseif  beta1B<beta1A
     if abs(beta1B-beta1A)<=180
       s_b(i+1)=mod(beta1A-t*(abs(beta1B-beta1A)),360);
       l_b(i+1)=mod(beta1A+t*(360-abs(beta1B-beta1A)),360);
     elseif abs(beta1B-beta1A)>180
        l_b(i+1)=mod(beta1A-t*(abs(beta1B-beta1A)),360);
       s_b(i+1)=mod(beta1A+t*(360-abs(beta1B-beta1A)),360);
     end
end

if phi1B>phi1A
    if abs(phi1B-phi1A)<=180
       s_p(i+1)=mod(phi1A+t*(abs(phi1B-phi1A)),360);
       l_p(i+1)=mod(phi1A-t*(360-abs(phi1B-phi1A)),360);
    elseif abs(phi1B-phi1A)>180
        l_p(i+1)=mod(phi1A+t*(abs(phi1B-phi1A)),360);
       s_p(i+1)=mod(phi1A-t*(360-abs(phi1B-phi1A)),360);
    end
elseif  phi1B<phi1A
     if abs(phi1B-phi1A)<=180
       s_p(i+1)=mod(phi1A-t*(abs(phi1B-phi1A)),360);
       l_p(i+1)=mod(phi1A+t*(360-abs(phi1B-phi1A)),360);
     elseif abs(phi1B-phi1A)>180
        l_p(i+1)=mod(phi1A-t*(abs(phi1B-phi1A)),360);
       s_p(i+1)=mod(phi1A+t*(360-abs(phi1B-phi1A)),360);
     end
end
end
```

As stated earlier, this section shows how to obtain the short and long path. Here is a basic rundown of how the code functions. The first thing that can be observed on the code is where N is defined to be 100 which we use to generate 100 evolution angles which will be understood as we move on with the explanation. Now with the for loop the code is ran

100 times and after that we have a time variable(t) set with a formula which will help us to obtain 100 consecutive time points on the time interval $[0, 1]$ and for each time stamp to obtain triples of angles. That is we obtain both the short and long paths(angles) for α, β, ϕ and we can plug them in multiple combination to obtain a rotation matrix which will be explained in next section. The code above is a code which shows how we transition from 1 point to another below are some results that can be obtained as short and long paths.

```
short_alpha(0.00)=30
short_alpha(0.01)=30.404
short_alpha(0.02)=30.8081
short_alpha(0.03)=31.2121
short_alpha(0.04)=31.6162
short_alpha(0.05)=32.0202
short_alpha(0.06)=32.4242
short_alpha(0.07)=32.8283
short_alpha(0.08)=33.2323
short_alpha(0.09)=33.6364
short_alpha(0.10)=34.0404
```

Basic example of how results can be obtained where the angles represent the short path angles with respect to time when going from 0 to 1, in this way we can observe all other angles as well with some extra code such as the code below in order to understand how these angles are going from the starting point to the end point with respect to time:

```
      for i=0:(N-1)
      t=(1/(N-1))*i;
      var=sprintf('short_alpha(%.2f)',t);
     disp([var, '=', num2str(s_a(i+1))]);
 end

% for i=0:(N-1)
%       t=(1/(N-1))*i;
%       var=sprintf('long_alpha(%.2f)',t);
%       disp([var, '=', num2str(l_a(i+1))]);
% end
%
% for i=0:(N-1)
%       t=(1/(N-1))*i;
%       var=sprintf('short_beta(%.2f)',t);
%       disp([var, '=', num2str(s_b(i+1))])
% end

% for i=0:(N-1)
%       t=(1/(N-1))*i;
%       var=sprintf('long_beta(%.2f)',t);
%       disp([var, '=', num2str(l_b(i+1))])
% end
%
% for i=0:(N-1)
%       t=(1/(N-1))*i;
%       var=sprintf('short_phi(%.2f)',t);
%       disp([var, '=', num2str(s_p(i+1))])
% end
%
% for i=0:(N-1)
%       t=(1/(N-1))*i;
%       var=sprintf('long_phi(%.2f)',t);
%       disp([var, '=', num2str(l_p(i+1))])
% end
```

Where the for loop enables us to print all hundred time evolution angles and display them with the time it takes to reach said angle at each point in order to easily visualize how angle moves with time.

4.2.2 Geodesic matrix Using Time Evolution Angles

The code below explains how we can obtain the rotation matrix of the time evolution angles at different time points.

```
    Rti = zeros(3, 3, N);
combo=9;
for i=0:(N-1)
    if(combo==1)
     Rti(:,:,i+1)= [(cosd(s_a(i+1))*cosd(s_b(i+1))*cosd(s_p(i+1))-
     sind(s_a(i+1))*sind(s_p(i+1))),(-cosd(s_p(i+1))*sind(s_a(i+1))-
     cosd(s_a(i+1))*cosd(s_b(i+1))*sind(s_p(i+1))), (cosd(s_a(i+1))*
     sind(s_b(i+1)));cosd(s_a(i+1))*sind(s_p(i+1))+cosd(s_b(i+1))*
     cosd(s_p(i+1))*sind(s_a(i+1)), cosd(s_a(i+1))*cosd(s_p(i+1))-
     cosd(s_b(i+1))*sind(s_a(i+1))*sind(s_p(i+1)),sind(s_a(i+1))*
     sind(s_b(i+1));-cosd(s_p(i+1))*sind(s_b(i+1)), sind(s_b(i+1))*
     sind(s_p(i+1)), cosd(s_b(i+1))];

     elseif(combo==2)
       Rti(:,:,i+1)= [(cosd(s_a(i+1))*cosd(s_b(i+1))*cosd(l_p(i+1))-
       sind(s_a(i+1))*sind(l_p(i+1))), (-cosd(l_p(i+1))*sind(s_a(i+1))-
       cosd(s_a(i+1))*cosd(s_b(i+1))*sind(l_p(i+1))), (cosd(s_a(i+1))*
       sind(s_b(i+1)));cosd(s_a(i+1))*sind(l_p(i+1))+cosd(s_b(i+1))*
       cosd(l_p(i+1))*sind(s_a(i+1)), cosd(s_a(i+1))*cosd(l_p(i+1))-
       cosd(s_b(i+1))*sind(s_a(i+1))*sind(l_p(i+1)), sind(s_a(i+1))*
       sind(s_b(i+1));-cosd(l_p(i+1))*sind(s_b(i+1)), sind(s_b(i+1))*
       sind(l_p(i+1)), cosd(s_b(i+1))];

    elseif(combo==3)
       Rti(:,:,i+1)= [(cosd(s_a(i+1))*cosd(l_b(i+1))*cosd(s_p(i+1))-
       sind(s_a(i+1))*sind(s_p(i+1))), (-cosd(s_p(i+1))*sind(s_a(i+1))-
       cosd(s_a(i+1))*cosd(l_b(i+1))*sind(s_p(i+1))), (cosd(s_a(i+1))*
       sind(l_b(i+1)));cosd(s_a(i+1))*sind(s_p(i+1))+cosd(l_b(i+1))*
       cosd(s_p(i+1))*sind(s_a(i+1)), cosd(s_a(i+1))*cosd(s_p(i+1))-
       cosd(l_b(i+1))*sind(s_a(i+1))*sind(s_p(i+1)), sind(s_a(i+1))*
       sind(l_b(i+1)); -cosd(s_p(i+1))*sind(l_b(i+1)), sind(l_b(i+1))*
       sind(s_p(i+1)), cosd(l_b(i+1))];

    elseif(combo==4)
       Rti(:,:,i+1)= [(cosd(s_a(i+1))*cosd(l_b(i+1))*cosd(l_p(i+1))-
       sind(s_a(i+1))*sind(l_p(i+1))), (-cosd(l_p(i+1))*sind(s_a(i+1))-
       cosd(s_a(i+1))*cosd(l_b(i+1))*sind(l_p(i+1))), (cosd(s_a(i+1))*
       sind(l_b(i+1)));cosd(s_a(i+1))*sind(l_p(i+1))+cosd(l_b(i+1))*
       cosd(l_p(i+1))*sind(s_a(i+1)), cosd(s_a(i+1))*cosd(l_p(i+1))-
       cosd(l_b(i+1))*sind(s_a(i+1))*sind(l_p(i+1)), sind(s_a(i+1))*
       sind(l_b(i+1));-cosd(l_p(i+1))*sind(l_b(i+1)), sind(l_b(i+1))*
       sind(l_p(i+1)), cosd(l_b(i+1))];

    elseif(combo==5)
       Rti(:,:,i+1)= [(cosd(l_a(i+1))*cosd(s_b(i+1))*cosd(s_p(i+1))-
       sind(l_a(i+1))*sind(s_p(i+1))), (-cosd(s_p(i+1))*sind(l_a(i+1))-
       cosd(l_a(i+1))*cosd(s_b(i+1))*sind(s_p(i+1))), (cosd(l_a(i+1))*
       sind(s_b(i+1)));cosd(l_a(i+1))*sind(s_p(i+1))+cosd(s_b(i+1))*
       cosd(s_p(i+1))*sind(l_a(i+1)), cosd(l_a(i+1))*cosd(s_p(i+1))-
       cosd(s_b(i+1))*sind(l_a(i+1))*sind(s_p(i+1)), sind(l_a(i+1))*
```

```
        sind(s_b(i+1));-cosd(s_p(i+1))*sind(s_b(i+1)),  sind(s_b(i+1))*
        sind(s_p(i+1)), cosd(s_b(i+1))];

    elseif(combo==6)
        Rti(:,:,i+1)= [(cosd(l_a(i+1))*cosd(s_b(i+1))*cosd(l_p(i+1))-
        sind(l_a(i+1))*sind(l_p(i+1))), (-cosd(l_p(i+1))*sind(l_a(i+1))-
        cosd(l_a(i+1))*cosd(s_b(i+1))*sind(l_p(i+1))), (cosd(l_a(i+1))*
        sind(s_b(i+1)));cosd(l_a(i+1))*sind(l_p(i+1))+cosd(s_b(i+1))*
        cosd(l_p(i+1))*sind(l_a(i+1)), cosd(l_a(i+1))*cosd(l_p(i+1))-
        cosd(s_b(i+1))*sind(l_a(i+1))*sind(l_p(i+1)), sind(l_a(i+1))*
        sind(s_b(i+1));-cosd(l_p(i+1))*sind(s_b(i+1)),  sind(s_b(i+1))*
        sind(l_p(i+1)), cosd(s_b(i+1))];

    elseif(combo==7)
        Rti(:,:,i+1)= [(cosd(l_a(i+1))*cosd(l_b(i+1))*cosd(s_p(i+1))-
        sind(l_a(i+1))*sind(s_p(i+1))), (-cosd(s_p(i+1))*sind(l_a(i+1))-
        cosd(l_a(i+1))*cosd(l_b(i+1))*sind(s_p(i+1))), (cosd(l_a(i+1))*
        sind(l_b(i+1)));cosd(l_a(i+1))*sind(s_p(i+1))+cosd(l_b(i+1))*
        cosd(s_p(i+1))*sind(l_a(i+1)), cosd(l_a(i+1))*cosd(s_p(i+1))-
        cosd(l_b(i+1))*sind(l_a(i+1))*sind(s_p(i+1)), sind(l_a(i+1))*
        sind(l_b(i+1));-cosd(s_p(i+1))*sind(l_b(i+1)),  sind(l_b(i+1))*
        sind(s_p(i+1)), cosd(l_b(i+1))];

    else
        Rti(:,:,i+1)= [(cosd(l_a(i+1))*cosd(l_b(i+1))*cosd(l_p(i+1))-
        sind(l_a(i+1))*sind(l_p(i+1))), (-cosd(l_p(i+1))*sind(l_a(i+1))-
        cosd(l_a(i+1))*cosd(l_b(i+1))*sind(l_p(i+1))), (cosd(l_a(i+1))*
        sind(l_b(i+1)));cosd(l_a(i+1))*sind(l_p(i+1))+cosd(l_b(i+1))*
        cosd(l_p(i+1))*sind(l_a(i+1)), cosd(l_a(i+1))*cosd(l_p(i+1))-
        cosd(l_b(i+1))*sind(l_a(i+1))*sind(l_p(i+1)), sind(l_a(i+1))*
        sind(l_b(i+1));-cosd(l_p(i+1))*sind(l_b(i+1)),  sind(l_b(i+1))*
        sind(l_p(i+1)), cosd(l_b(i+1))];
    end
end
for i=0:(N-1)
    fprintf('Rti(%d)\n',i+1);
    disp(Rti(:,:,i+1));
end
```

This code visualizes how we can obtain the time evolution angles manually. In this Matlab software there are also some built in functions to easily obtain this but it has been done manually so we can understand the process of obtaining this matrix. Referring back to (type ABA order of describing matrix) once the elementary matrices are multiplied and the ZYZ Euler angle is obtained it can then be used and we can simply substitute the corresponding time evolution angles to obtain the rotation matrix. This code shows 8 different conditions for calculating this rotation matrix, these conditions simply represent the 8 different combinations for all the short and long paths as explained in 3.1. Therefore with this data we can obtain the rotation matrix for all time points for all 8 different combinations and select which transition method among all of them is more efficient. We can observe which transition method is more efficient by comparing it to the geodesic matrix of the axis angle which will be observed in the next section. The for loop at the end of the code helps to visualize these rotation matrices which we can then use to ensure that we transition from $t = 0$ which should be the starting point matrix R_A to $t = 1$ the end point matrix R_B.

```
    Rti(1)
   -0.2795   -0.6941    0.6634
    0.9237   -0.0058    0.3830
   -0.2620    0.7198    0.6428
```

```
Rti(2)
   -0.1715   -0.7342    0.6569
    0.9404    0.0769    0.3314
   -0.2938    0.6746    0.6772

Rti(3)
   -0.0592   -0.7615    0.6454
    0.9454    0.1647    0.2811
   -0.3204    0.6269    0.7102

Rti(4)
    0.0555   -0.7754    0.6290
    0.9382    0.2560    0.2327
   -0.3415    0.5773    0.7417

Rti(5)
    0.1711   -0.7753    0.6080
    0.9183    0.3490    0.1867
   -0.3570    0.5264    0.7717
```

The code snippet above shows the output of some of the rotation matrices when printed using the for loop. All hundred different rotation matrices for each time point can be visualized and used for comparison.

4.3 Geodesic curve matrices corresponding to the axis-angle parameterization

In this section we will explain how simulate the calculation of the axis-angle geodesic distance as explained in 3.3.

```
N=100;
I=RA*RA';
R2=RA'*RB;
Theta=acosd((trace(R2)-1)/2);
V_x= (R2(3,2)-R2(2,3))/(2*sind(Theta));
V_y= (R2(1,3)-R2(3,1))/(2*sind(Theta));
V_z= (R2(2,1)-R2(1,2))/(2*sind(Theta));
V=[V_x,V_y,V_z];
 Rgeo_a=zeros(3,3,N);
for i=0:(N-1)
      t=(1/(N-1))*i;
      theta=(t*Theta);
      u_t=1-cosd(theta);
      Rgeo_a(:,:,i+1)=RA*[(u_t*(V_x)^2) + cosd(theta), (u_t * V_x * V_y)
      - (V_z*sind(theta)), (u_t * V_x * V_z) + (V_y * sind(theta));
          (u_t * V_x * V_y) + (V_z *sind(theta)), (u_t*(V_y)^2) +
          cosd(theta),(u_t * V_y * V_z) - (V_x * sind(theta));
          (u_t * V_x * V_z) - (V_y * sind(theta)), (u_t * V_y * V_z) +
          (V_x * sind(theta)), (u_t*(V_z)^2) + cosd(theta)];
end
for i=0:(N-1)
    fprintf('Rgeo_a(%d)\n',i+1);
    disp(Rgeo_a(:,:,i+1));
end
```

How does this code function? As explained in Section 3.3 we have to multiply both the starting and endpoint matrices by the transpose of the starting matrix where as we then

obtain the identity matrix and a new matrix. Now from the new matrix we obtain the axis angle parameters which will be used to calculate geodesic matrix. As usual there are suitable functions to enable to easily calculate the axis angle parameters but it has been done manually to ensure understanding in calculating axis angle parameters and also confirm that equations derived for for such calculations are correct. With the axis angle parameters obtained we can then obtain the geodesic matrix using formula which was explained in Eq. (3.39). Where the formula helps to obtain the geodesic matrix but with hundred different time points corresponding with the time points used in the Euler parameterization which basically gives the leeway for comparisons. The for loop at the end also helps to visualize data as show in previous section but for axis angle rotation matrix. Therefore with all the data obtained we can now focus on the main aim of the thesis which is comparing the different parameterization methods.

4.3.1 Distance between matrices in a time interval

This section shows how the distance between the matrices of the two different matrices will be obtained.

```
    for i=0:(N-1)
    dist(i+1)=norm(Rti(:,:,i+1)- (Rgeo_a(:,:,i+1)),1);
end
for i=0:(N-1)
    fprintf('dist(%d)\n',i+1);
    disp(dist(i+1));
end
```

As explained in 3.46 to obtain the distance between the matrix we get the difference of the matrices by subtraction and then we use the matrix norm to obtain a single value, in our case we'll be using the norm of type 1 where as there are other types and with this we can obtain the distance between each matrix at the different time points. The for loop helps to visualize so we can see the distance at each point for basic understanding of what is taking place, but in order to get a full conclusion we calculate the total sum of all the distances across the the time points. The code below shows how that can be achieved.

```
    for i=0:(N-1)
    dist(i+1)=norm(Rti(:,:,i+1)- (Rgeo_a(:,:,i+1)),1);
end
for i=0:(N-1)
    fprintf('dist(%d)\n',i+1);
    disp(dist(i+1));
end
```

Now with all the distance summed we can come to conclusion of which parameterization method is more efficient and also which transition method is more efficient withing the Euler parameterization.

Chapter 5

Simulations

To evaluate which which path for a single parameterization is better some simulations were performed. In Table 5.1 it was showed a sample of data for each transition(4 transitions) and in each transition we have data for all different combinations of the Euler angles depending short/long path angle.

	SSS	SSL	SLS	SLL	LSS	LSL	LLS	LLL
1A,1B	17.5	205.5	200.3	218.3405	163.2031	187.5	215.1	142.5
1A,2B	142.9	183.4	183.5	173.0	174.8	147.9	167.9695	168.1
2A,1B	142.9	183.4	183.5	173.0	174.8	147.9	167.9672	168.1
2A,2B	17.5	205.6	200.4	218.3	163.2034	187.5	215.1	142.5

Tabela. 5.1 Total sum of differences between the Euler and the axis-angle parameterizations for exemplary motion between two matrices

All the values in Table 5.1 are a result of evaluation of the difference between matrices generated with the parameterization Euler method and the geodesic curve obtained from the axis-angle parameterization. Ideally, the difference should be zero but it is not the case. In our case, cf. Table 5.1, we have smallest difference when transitioning from 1A to 1B and from 2A to 2B using all short angles(SSS) so based on that we can conclude that this is the most efficient method. It confirms our natural expectation.

With all the code explained in Chapter 4 one can make comparisons as so for a large amount of data to come to conclusion on which is the most efficient for their specific application. This thesis aims to show how we can make these comparisons between parameterization methods for specific applications (sets of boundary data). In summary the interpretation of the comparison depends on the specific needs of the application. If a total distance of 17 is acceptable the selected parameterization is good if a smaller distance is crucial, further analysis and potential adjustments to the approach may be necessary.

Chapter 6

Conclusions

The main purpose of this bachelor thesis was to be able to compare parameterization methods of a rotation matrices in robotics. The study aimed to investigate the advantages and limitations of each parameterization method in terms of efficiency and accuracy for practical implementation in robotic systems.

The thesis mainly involved a comprehensive review of existing literature on rotation matrix parameterization methods, that is a detailed analysis of Euler angles and axis-angle methods. The comparison involved multiple mathematical formulations for corresponding parameterization methods to help obtain and visualize data sets and analyze them to derive meaningful insights that would be useful in formulating conclusions from said comparisons.

Through simulations the thesis demonstrated the performance of each parameterization method showing how one could determine their strength and weaknesses. In conclusion when comparing parameterization methods finding the more efficient or accurate method could be tricky but mainly depends on the application and also the self discipline to gather general data sets for said application in order to truly simulate and determine which parameterization methods would be more accurate for the specific application it is being used for.

For future research one could firstly explore alternative parameterizations such as the quaternion representation which is known for avoiding gimbal lock and may offer different trade-offs. As this thesis is a theoretical approach, one could also apply it to the real world, that is if applicable the methods can be tested with real-world scenarios or data that closely resemble intended application. Real world conditions may reveal challenges or opportunities not evident in synthetic data sets.

Bibliography

[1] Group. Wolfram MathWorld, url = https://mathworld.wolfram.com/Group.html, 2023.

[2] Question on special orthogonal group so(3) - linear algebra and group theory - . Science Forums, url = https://www.scienceforums.net/topic/10352-question-on-special-orthogonal-group-so3, 2023.

[3] Touchdro - powerful modern digital readout system. TouchDRO , url = https://www.touchdro.com, 2023.

[4] Rotation matrix derivation (the 3-d global spatial data model). url = https://what-when-how.com/the-3-d-global-spatial-data-model/rotation-matrix-derivation-the-3-d-global-spatial-data-model, 2023.

[5] Mathias Brandstötter. *Adaptable Serial Manipulators in Modular Design*. PhD thesis, Carinthian University of Applied Sciences, 11 2016.

[6] 3drotations. motion.pratt.duke.edu, url = http://motion.pratt.duke.edu /Robotic-Systems/3DRotations.html, 2023.

[7] Euler angles. Wolfram MathWorld, url = https://mathworld.wolfram. com/EulerAngles.html, 2023.

[8] J.Craig John. *Introduction to Robotics: Mechanincs and Control*. Pearson Education, Edinburgh Gate,Harlow,Essex CM20 2JE,England, 2013.

[9] Rotations in three-dimensions: Euler angles and rotation matrices. `https://danceswithcode.net/engineeringnotes/index.html`, 2015.

[10] Redstone Alexander. Axis angle rotation. `https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/REDSTONE/AxisAngleRotation.html`, 11 2005.

[11] Watt Alan. *Advanced Animation and Rendering Techniques*. Addison-Wesley, Harlow,England, 1992.

[12] Symmetric matrix and skew symmetric matrix. url=https://byjus.com/maths/what-is-symmetric-matrix-and-skew-symmetric-matrix/, 2023.

[13] Ena Alejandro. Euler angles, rotations and gimbal lock. `https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/REDSTONE/AxisAngleRotation.html`, 2022.

[14] Understanding euler angles. url=https://liqul.github.io/blog/assets/rotation.pdf, 2022.

[15] Yuri. Range of angle in axis-angle representation of rotations. StackExchange,url = https://math.stackexchange.com/questions/969755/range-of-angle-in-axis-angle-representation-of-rotations, 2014.

[16] Grassia Sebastian, F. Practical parameterization of rotations using the exponential map. Carnegie Mellon University, url=https://www.cs.cmu.edu/ spiff/-moedit99/expmap.pdf, 1998.

[17] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, 1992.

[18] Gankoji. 3d rotationsaxis-angle and shortest paths. `https://math.stackexchange.com/questions/2726721/3d-rotations-axis-angle-and-shortest-paths`, 2018.