

Applications of Differential Geometry in Computer Graphics

Børge Garpestad

June 9, 2009

Contents

1	Introduction	1
2	Geometric Transformations	1
2.1	Introduction to Geometric Transformations	1
2.2	Homogeneous Coordinates	3
2.3	Rigid Body Transformations	3
2.3.1	Translation	4
2.3.2	Rotation	4
3	Group Theory	6
3.1	$SO(3)$ - Rotation Matrices	7
3.2	$SE(3)$ - Transformation Matrices for Rigid Body Positioning	7
4	Manifolds	8
4.1	Smooth Manifold	8
4.2	Smooth Mapping	8
4.3	Tangent Space	9
4.4	Vector Fields and Integral Curves	9
5	Lie groups and their Lie algebra	9
5.1	Lie algebra for $SO(3)$	11
5.2	Lie algebra for $SE(3)$	11
5.3	Basis for the Lie algebra of $SE(3)$ and $SO(3)$	11
6	Exponential Map	12
6.1	Closed-form formula for $\mathfrak{so}(3)$	13
6.2	Closed-form formula for $\mathfrak{se}(3)$	13
7	Metric, Covariant derivative	14
7.1	The Inner Product on \mathbf{R}^n	14
7.2	Riemannian Manifold	15
7.3	Choices of Metrics	16
7.4	Covariant Derivative and Parallel Transport	16
8	Cost function	17
8.1	Minimum Distance Curves	17
8.2	Minimum Acceleration Curves	18
9	SVD Projection Method	19
9.1	Projection on $SO(3)$	19
9.2	Projection on $SE(3)$	21
10	Interpolation in Euclidean Space	21
11	Implementation and usage of algorithm	22
12	Results and Conclusion	23

1 Introduction

In this paper, the problem of computing the placement of a object between key positions and orientations are addressed. In other words, given a three dimensional object that are situated in space at two key positions, the algorithm of interest, is one that computes all intermediate placements between the two key positions. A object placement in this context involves both a position $\in \mathbb{R}^3$, and a rotation of the object.

Usage of this type of algorithms is relevant in areas such as robotics and computer graphics. In this paper, the result focuses mostly towards computer graphics. A algorithm than can interpolate between orientations and placements has many applications in computer graphics. Example of this, could be to define a motion for objects or for positioning objects in a certain manner. Associated with this paper, is a algorithm that solves the problem of interpolating between placements [5]. The algorithm is written in C++, and uses OpenGL as graphics engine.

The paper is an overview of existing material corresponding to the problem. The main resource is from Calin Belta and Vijay Kumars many articles describing the problem [2,3,4,11]. The SVD based interpolation method they presented in [4] is also the method used for implementing the algorithm associated with this paper [5].

A short overview of the context in this paper is here given.

In the first section, geometric transformation that can describe a position and orientation in space is given. The result from this section, is a proper definition of the geometric transformations corresponding to translation and rotation. In the next sections, tools from differential geometry are presented. These tools make it possible to do computation and analysis on the set of all geometric transformations, resulting in methods to identify certain aspects that the interpolation should fulfill, and with this, outline possible algorithms that can solve the problem. In the last section, the resulting algorithm is presented in short, with some results. The source code for the interpolation is also included at the end of the document. It is released under the GNU General Public License (GPL).

2 Geometric Transformations

When describing a position and orientation of a three dimensional object, a Geometric Transformation is used. Geometric transformations is a basic concept in areas such as computer graphics, robotics and many more. The geometric transformations presented in this paper, are those that describe physical motion of a rigid body. These types of transformations are called rigid body transformations.

In this first section, the very basic of translation- and rotation matrices are explained. After this, some requirements of a rigid body transformation is presented. These requirements are then used to derive at definitions for the elements in a rigid body transformation. A reader familiar with basic geometric transformations, can surely skip this first section.

2.1 Introduction to Geometric Transformations

Translation

The simplest sort of transformations are pure translation. A point \mathbf{p} , can be translated to

another point \mathbf{p}' , by adding a vector \mathbf{d} . ($\mathbf{p}, \mathbf{p}', \mathbf{d} \in \mathbb{R}^n$)

$$\mathbf{p}' = \mathbf{p} + \mathbf{d}$$

Rotation

When considering rotations two elements needs to be considered. That is the angle θ of the rotation, and the axis of rotation. The rotation in \mathbb{R}^3 around the z-axis will here be explained. A reader interested in more in detail explanation of the different types of rotations are referred to [6].

Define the angle θ as the angle of rotation of a point \mathbf{p} ($\mathbf{p} = [x, y, z]^T$) around the z-axis and the resulting point as \mathbf{p}' ($\mathbf{p}' = [x', y', z']^T$). The angle ϕ is the angular position of the point \mathbf{p} , that is

$$x = r \cos(\phi) \tag{1}$$

$$y = r \sin(\phi) \tag{2}$$

By use of standard trigonometric identities the following formulas for a rotation around the z-axis can be derived.

$$x' = r \cos(\phi + \theta) = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$

$$y' = r \sin(\phi + \theta) = r \cos(\phi) \sin(\theta) + r \sin(\phi) \cos(\theta)$$

$$z' = z$$

Substituting for the angular expression (1,2)

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$

$$z' = z$$

Writing these equations on matrix form

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

where $R \in \mathbb{R}^{3 \times 3}$

Results for rotation around x and y-axis can be derived in a similar way. The notation that R is a rotation matrix will be used in the rest of this paper.

2.2 Homogeneous Coordinates

Above translation and rotation transformations were presented. A translation transformation involved adding a vector, while the rotation transformation involved a matrix multiplication. The idea about homogeneous coordinates is to combine these two actions into one single matrix operation. This is done by extending elements in \mathbb{R}^3 with one element called the homogeneous parameter. In our treatment this value will be set to 1. The homogeneous transformation matrix is a 4×4 matrix. The special type of homogeneous transformation matrices discussed in this paper consisting of the 3×3 rotation matrix R and translation vector $\mathbf{d} \in \mathbb{R}^3$, in the following manner:

$$\begin{bmatrix} R & \mathbf{d} \\ 0 & 1 \end{bmatrix} \quad (3)$$

Noting the informality of defining this type of matrix as a homogeneous transformation matrix, since the transformation itself has nothing to do with a homogeneous transformation. Also, for convenience, when "homogeneous transformation matrix" is referred for the rest of this paper, it means the special transformation matrix with rotation and translation part as described above.

The transformation matrices for pure translations T_1 and pure rotations T_2 will then look as follows

$$T_1 = \begin{bmatrix} I & \mathbf{d} \\ 0 & 1 \end{bmatrix}$$
$$T_2 = \begin{bmatrix} R & \mathbf{0} \\ 0 & 1 \end{bmatrix}$$

By multiplication of different types of T_1 and T_2 matrices we can achieve a matrix that both translate and rotates [6]. By combining these types we can also compute transformation matrices for more complex actions, like rotations around an arbitrary axis [6]. The composition of a rotation around a straight line, and translation in the line's direction was proved to be able to represent the transformation between any two positions, by Chasles and Poincaré in the early 1800s.

2.3 Rigid Body Transformations

The transformations we are describing in this paper are structure preserving transformations. That is, if we have a solid object represented by a multiple of points, doing a transformation on these points, should not alter the structure of the object, only move it. This type of transformation is called a rigid body transformation. In this section some key requirements for a rigid body transformation are given.

It is convenient to define a new description of the rigid body transformation, when defining these requirements. Define the points representing the object to be transformed as a subset O ($O \subset \mathbb{R}^3$). The transformation can then be defined as a single mapping g , of the elements in O .

$$g : O \rightarrow \mathbb{R}^3$$

Distance preserving

It is clear, when considering that a rigid body motion describes a physical movement, that

a transformation of O , should not alter the distance between elements in O . The distance between points before the transformation, should equal the distance after.

$$\|g(\mathbf{p}) - g(\mathbf{q})\| = \|\mathbf{p} - \mathbf{q}\| \quad \text{for all points } \mathbf{p}, \mathbf{q} \in \mathbb{R}^3$$

The cross product preserved

The requirement off a distance preserving map is not sufficient to ensure that the transformation is physically possible. An internal reflection, could still occur, and it is clear that this type of transformation is not physically realizable. A requirement that meet this problem is to make the cross product between vectors in the body preserved [7]. That is,

$$g(\mathbf{v} \times \mathbf{w}) = g(\mathbf{v}) \times g(\mathbf{w}) \quad \text{for all vectors } \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$$

2.3.1 Translation

It is easy to show that the distance preserving requirement is fulfilled for a translation transformation. Considering a transformatin $g(\mathbf{p})$

$$g(\mathbf{p}) = \mathbf{p} + \mathbf{d} \quad \mathbf{p}, \mathbf{d} \in \mathbb{R}^3$$

Inserting this transformation into the distance preserving requirement

$$\|g(\mathbf{p}) - g(\mathbf{q})\| = \|\mathbf{p} + \mathbf{d} - (\mathbf{q} + \mathbf{d})\| = \|\mathbf{p} - \mathbf{q}\|$$

This clearly proves that a translation transformation fulfills the distance preserving requirement.

2.3.2 Rotation

For the rotation part, two key results will be derived using the rigid body requirements. First a two frame coordinate system representation of a transformation will be explained. Next, specific requirements for the elements in a rotation matrix will be stated.

Two frame representation

Considering the expression for the dot product with the polarization identity [7] for two vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$

$$\mathbf{p}^T \mathbf{q} = \frac{1}{4}(\|\mathbf{p} + \mathbf{q}\|^2 - \|\mathbf{p} - \mathbf{q}\|^2)$$

Since the mapping g should be distance preserving, it is clear that

$$\|\mathbf{p} + \mathbf{q}\| = \|g(\mathbf{p}) + g(\mathbf{q})\| \quad \|\mathbf{p} - \mathbf{q}\| = \|g(\mathbf{p}) - g(\mathbf{q})\|$$

Inserting this result into the polarization identity, a relation between the original inner products and the transformed ones are given.

$$\mathbf{p}^T \mathbf{q} = \frac{1}{4}(\|g(\mathbf{p}) + g(\mathbf{q})\|^2 - \|g(\mathbf{p}) - g(\mathbf{q})\|^2)$$

$$\mathbf{p}^T \mathbf{q} = g(\mathbf{p})^T g(\mathbf{q})$$

By this result, it is clear that two orthogonal vectors, are transformed to orthogonal vectors. Using the two facts that the mapping both preserves the cross product and orthogonality, we

can state that a orthonormal reference frame get mapped into another orthonormal reference frame [7]. This result give us a new method of describing the motion of a rigid body. Since the mapping is both distance preserving, and that the cross product is preserved, points on the rigid body can rotate with respect to each other, but not translate. With this, a motion of the rigid body can be presented by use of two Cartesian coordinate frames. One is fixed under the motion, F, and one is fixed to the rigid body in motion, M. The mapping then involves translating points with the vector from the origin in F to origin in M, and then rotation corresponding to the rotation displacement between the two frames.

Requirements for the rotation matrix

Proposition 2.1. *Every rotation matrix is orthogonal, and have determinant of value +1.*

Proof Using basic linear properties, the distance preserving requirement and the property that a rotation transformation is linear [8] to prove the preposition. Considering a linear rotation transformation $L[\mathbf{v}]$

$$L[\mathbf{v}] = R \mathbf{v} \quad (4)$$

where $R \in \mathbb{R}^{3 \times 3}$, $\mathbf{v} \in \mathbb{R}^3$.

By linearity and the distance preserving requirement, the following relation between the transformation of two elements $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ is given

$$\|L[\mathbf{p}] - L[\mathbf{q}]\| = \|L[\mathbf{p} - \mathbf{q}]\| = \|\mathbf{p} - \mathbf{q}\|$$

By defining $\mathbf{v} = \mathbf{p} - \mathbf{q}$, the requirement for a linear function, preserving distance gets reduced to

$$\|L[\mathbf{v}]\| = \|\mathbf{v}\|$$

Evaluating the dot product of the rotation transformation (4), this requirement imposes that the following equation must hold.

$$\|R\mathbf{v}\|^2 = (R\mathbf{v})^T R\mathbf{v} = \mathbf{v}^T R^T R\mathbf{v} = \mathbf{v}^T \mathbf{v} = \|\mathbf{v}\|^2$$

It is clear, that for this relation to hold, $R^T R = I$. That is, the rotation matrix must be an orthogonal matrix.

Now knowing that a rotation matrix is orthogonal, information about its determinant can be achieved. To define the determinant, two results from Linear Algebra will be used. First, the relation between the determinants of two square matrices A,B of same size is given as

$$\det(AB) = \det(A)\det(B) \quad (5)$$

Secondly, the determinant of a orthogonal matrix R transposed, equals the determinant of the matrix R itself

$$\det(R^T) = \det(R) \quad (6)$$

Using these relations, the following equation can be expressed.

$$1 = \det(I) = \det(R^T R) = \det(R^T)\det(R) = \det(R)\det(R) = \det(R)^2$$

It is clear then that $\det(R) = \pm 1$.

From Linear algebra, the following equation for the determinant of a 3×3 matrix can be derived [7]

$$\det(R) = \mathbf{r}_1^T(\mathbf{r}_2 \times \mathbf{r}_3),$$

where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ corresponds to the column vectors of R . Now limit the coordinate systems to right handed. That is, for three orthogonal basis vectors $[\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ of a rotation matrix, the following relation must hold.

$$\mathbf{r}_2 \times \mathbf{r}_3 = \mathbf{r}_1$$

With this, the determinant of a rotation matrix can be expressed as

$$\det R = \mathbf{r}_1^T \mathbf{r}_1 = 1$$

wish concludes the proof.

Matrices both orthogonal, and with determinant +1 is denoted $SO(3)$, special orthogonal matrices, and corresponds to rotations. Orthogonal matrices with determinant of value -1 on the other hand, are obtained by reflection in a mirror. [8]

3 Group Theory

With the Rigid Body Transformations identified, tools for working with the set of all rigid body transformations needs to be defined. This is done, by use of group theory. In this section, group theory is represented in short. Also, the rotation matrices and homogeneous transformation matrices for rigid body placement is proved to have group structures.

Definition 3.1. *A group is a set G , equipped with a operation \circ , if it satisfies the following axioms:*

1. *Closure, if $p_1, p_2 \in G$ then $p_1 \circ p_2 \in G$*
2. *Identity, There exist an element e , such that for every $p \in G, p \circ e = e \circ p = p$*
3. *Inverse, For every $p \in G$, there exist an unique element $p^{-1} \in G$, such that $p \circ p^{-1} = p^{-1} \circ p = e$ and z of X ;*
4. *Associativity, if $p_1, p_2, p_3 \in G$ then $(p_1 \circ p_2) \circ p_3 = p_1 \circ (p_2 \circ p_3)$*

Proposition 3.2. *Both the set of rotation matrices, and transformation matrices equipped with matrix multiplication fulfills the structure requirements of a Group.*

Proofs are stated when defining the two groups $SO(3)$ and $SE(3)$ in the next to sections.

3.1 $SO(3)$ - Rotation Matrices

1. **Closure**, As stated in the last section a rotation matrix R is orthogonal, and have determinant $+1$. To prove closure, first the result of a matrix multiplication of two orthogonal matrices, is shown to be a orthogonal matrix

$$(R_1 \cdot R_2) \cdot (R_1 \cdot R_2)^T = R_1 \cdot R_2 \cdot R_2^T \cdot R_1^T = R_1 \cdot I \cdot R_1^T = I$$

Using (5), the matrix multiplication is shown to result in a matrix of $\det = +1$

$$\det(R_1 \cdot R_2) = \det(R_1)\det(R_2) = +1$$

2. **Identity**, The identity matrix I , is the identity element
3. **Inverse**, Since a rotation matrix is orthogonal, its transpose is its inverse. It is also easy to prove that the inverse of a rotation matrix R is also a rotation matrix.
4. **Associativity**, Follows from the associativity of matrix multiplication

With all the group axioms fulfilled for the rotation matrices, the Special Group of Rotations in three dimensions $SO(3)$ can be defined as a Group with the following requirements

$$SO(3) = \{R \in \mathbb{R}^{3 \times 3}, RR^T = I, \det R = +1\} \quad (7)$$

3.2 $SE(3)$ - Transformation Matrices for Rigid Body Positioning

A transformation matrix for rigid body positioning has the form as presented in (3).

1. **Closure**, Multiplying two homogeneous matrices S_1, S_2 , with rotation part $R_1, R_2 \in SO(3)$ and translational parts $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{R}^3$ respectively results in the following matrix.

$$\begin{bmatrix} R_1 \cdot R_2 & R_1 \mathbf{d}_2 + \mathbf{d}_1 \\ 0 & 1 \end{bmatrix}$$

From last section, $R_1 \cdot R_2 \in SO(3)$. It also clear that $R_1 \mathbf{d}_2 + \mathbf{d}_1 \in \mathbb{R}^3$. This proves that homogeneous transformation matrices for rigid bodies are closed under matrix multiplication.

2. **Identity**, The identity matrix I , is the identity element
3. **Inverse**, Can show that the inverse of a homogeneous transformation matrix for rigid bodies can be given as

$$\begin{bmatrix} R^T & -R^T \mathbf{d} \\ 0 & 1 \end{bmatrix}$$

From this it is clear that the inverse always exists, and is a homogeneous transformation matrix for rigid bodies itself

4. **Associativity**, Multiplying homogeneous matrices, the resulting matrix can be easily shown to be associative by the fact that matrix multiplication is associative.

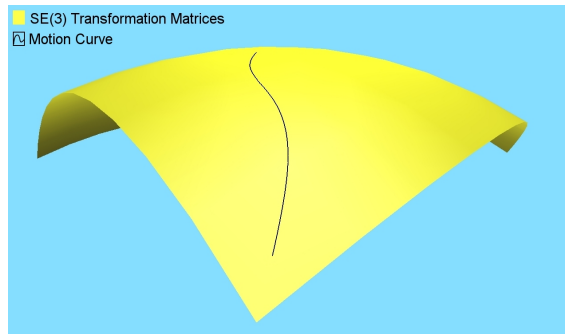
With the axioms fulfilled, the homogeneous matrices for rigid bodies can be defined as a group under matrix multiplication. Denote the group $SE(3)$ (Special Euclidean Group), the group of transformations corresponding to rigid body displacements.

$$SE(3) = \left\{ S \mid S = \begin{bmatrix} R & \mathbf{d} \\ 0 & 1 \end{bmatrix}, R \in SO(3), \mathbf{d} \in \mathbb{R}^3 \right\} \quad (8)$$

4 Manifolds

The definition of $SO(3)$ (7) and $SE(3)$ (8) shows clearly that they are restrictions on elements in the transformation matrices. Not all values are possible, and it can easily be shown that $SO(3)$ and $SE(3)$ are not vector spaces. Rather, the two sets of transformations $SO(3)$ and $SE(3)$, can be identified as manifolds [7]. In this section, the definition of a manifold and associated elements are presented. These definitions make it possible to define a curve $\gamma(t)$, ($t \in \mathbb{R}$) on the manifold, that describes a physical motion of a three dimensional object. The value of the curve at different parameter values corresponds to different elements of $SO(3)/SE(3)$.

A set (of points) M is defined to be a manifold if each point of M has an open neighborhood which has a continuous 1-1 map onto an open set of \mathbb{R}^n for some n . [9]. Another way of defining it, is to consider the manifold embedded in \mathbb{R}^n . With this, the manifold can be imagined as a d -dimensional surface ($d \leq n$), in which globally is curved and not suited for euclidean geometry, but locally the manifold is 'like' \mathbb{R}^n .



Visualization of $SE(3)$ as a manifold, with a curve representing a motion.

4.1 Smooth Manifold

Define coordinate charts on the manifold M . That is, a pair (ϕ, U) . ϕ is a function mapping the neighborhood of a point $p \in U \subset M$ to \mathbb{R}^n . When studying the overlapping areas of these coordinate charts, equations describing a relation between the mapping functions from the two coordinate charts can be found. If the partial derivatives exist for all orders, the overlapping charts are said to be C^∞ related. A collection of C^∞ related coordinate charts that completely cover the manifold is defined as a smooth manifold. A more in depth analysis of this concept can be found in [7].

4.2 Smooth Mapping

Considering the mapping F between two manifolds, M and N ($F : M \rightarrow N$). The manifolds M, N has coordinate charts (U, ϕ) and (V, τ) respectively. The mapping can be defined as

smooth or not by the composite mapping $F' = \tau \circ F \circ \phi^{-1} : \phi(M) \rightarrow \tau(N)$ [9]. If this mapping is smooth, then the mapping is said to be a smooth mapping.

4.3 Tangent Space

Denote the tangent space at a point p , on the manifold M , as T_pM . The tangent space T_pM is the collection of tangent vectors at that certain point. Intuitively, a tangent vector at a point x could be imagined as the the direction, in wish it is possible to pass x . Thus, the tangent space at a point would be the set of all possible directions at the specific point.

A formal definition of the tangent vector on a manifold can be derived using different methods. The definition presented here is obtained from [1]. Here differentiation, and the fact that a manifold is locally like \mathbb{R}^n is used to define the tangent vector. That is, we have a set of smooth real valued functions F_x defined on a neighborhood of a point p , on the manifold M . With this, the tangent vector ξ_p to a manifold M , at a point p , is a mapping from $F_p(M)$ to \mathbb{R} such that there exist a curve γ on M , with $\gamma(0) = p$ satisfying

$$\xi_p f = \dot{\gamma} f := \left. \frac{d(f\gamma(t))}{dt} \right|_{t=0},$$

for all $f \in F_p(M)$.

4.4 Vector Fields and Integral Curves

A vector field can be considered as a rule for assigning tangent vectors at each point on an manifold in a smooth manner. One type of vector field/rule for vector field, is a left invariant vector field. Considering the vector field X , on the manifold M . The value of X at a point $p \in M$ can be found by left translating p with a element from the vector space at the identity.

Define a curve on a manifold M , with the vector field X .

$$\gamma(t) : \mathbb{R} \rightarrow M, \frac{d\gamma(t)}{dt} = X(\gamma(t))$$

, where $X(\gamma(t))$ denotes the value of the vector field, at points on the curve $\gamma(t)$.

A curve with this property with respect to X , is called the integral Curve of the vector field X .

5 Lie groups and their Lie algebra

More tools are given when working with the groups $SO(3)$ and $SE(3)$ by introducing Lie groups. A Lie group, is a group G , which is also a smooth manifold, where the operator \circ gives a smooth mapping between elements in the Group [7]. The inverse mapping p^{-1} for all $p \in G$ should also be smooth.

The Group operation of $SO(3)$, and $SE(3)$ are matrix multiplication, and inverse is given by the matrix inverse. Considering the operations element wise, it is clear that they are smooth operations [7]. Knowing that both $SO(3)$ and $SE(3)$ are manifolds from the last section, it is therefor clear that they are both Lie groups.

Studying the structure of the tangent space at the identity of a Lie group, give rise to some useful tools when dealing with $SO(3)$ and $SE(3)$. We can show that the tangent space at the identity has the structure of a Lie algebra. Further, this can be used when defining tangentspaces on the manifolds $SO(3)/SE(3)$, which will be usefull when defining motions/curves on $SO(3)/SE(3)$. That is, the Lie algebra is usefull to study the Lie groups $SO(3)/SE(3)$. A short definition of the lie algebra will be given here.

Definition 5.1. *A Lie algebra g , is a vector space endowed with a bilinear operation (bracket) satisfying the following conditions, on $[A,B]$*

- $[A, B] = -[B, A]$, *skew symmetric*
- $[\alpha A + \beta B, C] = \alpha[A, C] + \beta[B, C]$, *bilinearity*
- $[A, [B, C]] + [B, [C, A]] + [C, [A, B]] = 0$, *Jacobi identity*

The relation between Lie algebra and a Lie group is through the tangent space at the identity of the Lie group [10]. One way of describing the Lie algebra, is by introducing a left invariant vector field on the Lie group. Denote the vector field X , and define it over the Lie group G by left translation of a tangent vector g at the identity of G .

$$X(P) = Pg, \tag{9}$$

where $P \in G$

Considering integral curves $\gamma(t)$ on the vector field.

$$\gamma(t) : \mathbb{R} \rightarrow M$$

For the left invariant vector field, such a curve would satisfy the following differential equation.

$$\frac{d\gamma}{dt} = \gamma g \tag{10}$$

This differential equation has an analytical solution [10]. For a curve that passes through the origin the solution is given as

$$\gamma(t) = \exp(tg) \tag{11}$$

This matrix exponential can be extended by use of Taylor series [8]

$$\exp(tg) = I + tg + \frac{1}{2}t^2g^2 + \dots$$

Differentiating $\gamma(t)$, and assigning $t=0$, it is clear that the the element g actually is the tangent space at the identity.

$$\begin{aligned} \frac{d\gamma(t)}{dt} &= g + tg^2 \dots \\ \left. \frac{d\gamma(t)}{dt} \right|_{t=0} &= g \end{aligned}$$

Summary of this result is that the Lie algebra g of a Lie group G is the tangent space at the identity, and that it can be studied by using a Taylor expansion of the exponential of elements in the Lie group. In the next section, this result is used to find the Lie algebra for $SO(3)$ and $SE(3)$.

5.1 Lie algebra for $SO(3)$

Considering an integral curve $\gamma(t)$ on the Lie group $SO(3)$. Elements on this group is orthogonal, hence

$$\gamma(t) \circ \gamma(t)^T = I$$

Writing this relation with the Taylor expansion

$$(I + tg + \frac{1}{2}t^2g^2 + \dots) \circ (I + tg^T + \frac{1}{2}t^2g^2 + \dots) = I$$

Multiplying

$$(I + tg^T + tg + O(t^2)) = (I + t(g^T + g) + O(t^2)) = I \Leftrightarrow g^T + g = 0$$

This result proves that the Lie algebra for $SO(3)$ consists of skew symmetric matrices. The Lie algebra for $SO(3)$ is denoted $so(3)$, using the convention of capital letters for Lie groups, and small letters for their Lie algebra.

$$so(3) = [\hat{\omega} | \hat{\omega} \in \mathbb{R}^{3 \times 3}, \hat{\omega}^T = -\hat{\omega}]$$

5.2 Lie algebra for $SE(3)$

The Lie algebra for $SE(3)$, contains the lie algebra of $SO(3)$. That is the Lie algebra of the rotational part of $SE(3)$ is $so(3)$. The Lie algebra of translational part of $SE(3)$, has no more requirements, than that it should be \mathbb{R}^3 [7]. With this, the definition of the Lie algebra $se(3)$, of $SE(3)$ is given as

$$se(3) = \left[S = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \mid \hat{\omega} \in \mathbb{R}^{3 \times 3}, v \in \mathbb{R}^3, \hat{\omega}^T = -\hat{\omega} \right]$$

5.3 Basis for the Lie algebra off $SE(3)$ and $SO(3)$

With the requirements for the Lie algebra for $SE(3)$ and $SO(3)$ given in the last section, it is easy to show the basis of their Lie algebras. First looking at how to present the skew symmetric matrix $\hat{\omega}$ with a vector $\omega \in \mathbb{R}^3$. Using notation standard from Belta and Kumar for the basis definitions.

A skew symmetric matrix has the requirement $\hat{\omega}^T = -\hat{\omega}$. With this requirement, it is easy to see that every skew symmetric matrix $\hat{\omega}$ can be written on the form

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

Define ω as

$$\omega = [\omega_1, \omega_2, \omega_3]^T$$

Since $so(3)$ consists of skew symmetric matrices, the basis can be associated with the euclidean three dimensional basis $[e_1, e_2, e_3]$. Using the $\hat{\cdot}$ operator to define the skew symmetric matrix corresponding to a arbitrary vector in \mathbb{R}^3 , we obtain the standard basis for $so(3)$, given by matrices L_1^0, L_2^0, L_3^0 :

$$\begin{aligned}
e_1 = [1, 0, 0]^T, \hat{e}_1 = L_1^0 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \\
e_2 = [0, 1, 0]^T, \hat{e}_2 = L_2^0 &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \\
e_3 = [0, 0, 1]^T, \hat{e}_3 = L_3^0 &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned}$$

For $se(3)$ the basis is represented in a similar way. The standard basis for $se(3)$ is given by matrices L_1, \dots, L_6 :

$$\begin{aligned}
L_1 &= \begin{bmatrix} L_1^0 & 0 \\ 0 & 0 \end{bmatrix} \\
L_2 &= \begin{bmatrix} L_2^0 & 0 \\ 0 & 0 \end{bmatrix} \\
L_3 &= \begin{bmatrix} L_3^0 & 0 \\ 0 & 0 \end{bmatrix} \\
L_4 &= \begin{bmatrix} 0 & e_1 \\ 0 & 0 \end{bmatrix} \\
L_5 &= \begin{bmatrix} 0 & e_2 \\ 0 & 0 \end{bmatrix} \\
L_6 &= \begin{bmatrix} 0 & e_3 \\ 0 & 0 \end{bmatrix}
\end{aligned}$$

6 Exponential Map

The mapping from the $so(3)/se(3)$ to $SO(3)/SE(3)$ and opposite can be useful in some interpolation algorithms. In this section, an efficient method of doing the mapping is presented. As described earlier (11), the solution to the differential equation describing an integral curve on a left invariant vector field is given as,

$$\gamma(t) = \exp(tg),$$

where $\gamma(t) \in SE(3)/SO(3), g \in se(3)/so(3), t \in \mathbb{R}$

In other words, $\exp(tg)$ is the mapping from an element of a Lie algebra, to the corresponding Lie group. Note that matrix exponential can be extended by use of Taylor series

$$\exp(tg) = I + tg + \frac{1}{2}t^2g^2 + \dots$$

If computation of the mapping is needed, this formula is not efficient. In this section, properties of the elements in $se(3)$ and $so(3)$ are used to derive closed-form formulas for these Taylor series.

6.1 Closed-form formula for $\mathfrak{so}(3)$

Proposition 6.1. *The exponential map of a $\mathfrak{so}(3)$ element $\hat{\omega}$, can be written on the form:*

$$\exp \hat{\omega} = I_3 + \sin(\|\omega\|)V + (1 - \cos(\|\omega\|))V^2,$$

where $\|\omega\| = \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}$, $V = \frac{\hat{\omega}}{\|\omega\|}$

This formula for the exponential map is called a Rodrigues' Formula.

Proof Identifying first an important relation with a skew symmetric matrix $\hat{\omega} \in \mathbb{R}^{3 \times 3}$. By simple matrix multiplication it can be proved that the following relation holds

$$\hat{\omega}^3 = -(\omega_1 + \omega_2 + \omega_3)\hat{\omega}$$

Define a matrix $V \in \mathbb{R}^{3 \times 3}$ as

$$V = \frac{\hat{\omega}}{\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}}$$

Important property of V is that $V^3 = -V$

$$V^3 = \frac{\hat{\omega}^3}{(\omega_1^2 + \omega_2^2 + \omega_3^2)^{\frac{3}{2}}} = \frac{-(\omega_1 + \omega_2 + \omega_3)\hat{\omega}}{(\omega_1^2 + \omega_2^2 + \omega_3^2)^{\frac{3}{2}}} = -\frac{\hat{\omega}}{\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}} = -V$$

Evaluating the exponential $\exp(\theta V)$, $\theta \in \mathbb{R}$

$$\exp \theta V = I_3 + \theta V + \frac{\theta^2}{2!}V^2 + \frac{\theta^3}{3!}V^3 + \frac{\theta^4}{4!}V^4 + \frac{\theta^5}{5!}V^5 + \dots$$

With the relation $V^3 = -V$, it is easy to show that odd power of V parts equals either $-V$ or V , and pair power parts equals either $-V^2$ or V^2 , that is

$$\exp \theta V = I_3 + (\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} \dots)V + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} \dots)V^2$$

These series can be expressed by use of the Taylor expansion for sin and cos. Using this results in

$$\exp(\theta V) = I_3 + \sin(\theta)V + (1 - \cos(\theta))V^2$$

Now, by assigning $\theta = \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2} = \|\omega\|$, the resulting expression is

$$\exp \theta V = \exp \hat{\omega} = I_3 + \sin(\|\omega\|)V + (1 - \cos(\|\omega\|))V^2,$$

which proofs the proposition

6.2 Closed-form formula for $\mathfrak{se}(3)$

Proposition 6.2. *The exponential map of a $\mathfrak{se}(3)$ element Z , with skew symmetric matrix $\hat{\omega}$ and translation vector \mathbf{d} can be written on the form:*

$$\exp Z = I_3 + \|\omega\|S + (1 - \cos(\|\omega\|))S^2 + (\|\omega\| - \sin(\|\omega\|))S^3$$

$$\exp Z = \begin{bmatrix} \exp \hat{\omega} & \|\omega\|\mathbf{d} + (1 - \cos(\|\omega\|))V\mathbf{d} + (\|\omega\| - \sin(\|\omega\|))V^2\mathbf{d} \\ 0 & 1 \end{bmatrix},$$

where $\|\omega\| = \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}$, $V = \frac{\hat{\omega}}{\|\omega\|}$, $S = \frac{1}{\|\omega\|}Z$

Justification for this Rodrigues' theorem and formulas for the mapping from a Lie group element, to their Lie algebra element with the logarithmic function can be found in [10].

7 Metric, Covariant derivative

To identify different properties of a curve on a manifold, a method of comparing elements to each other on the manifold is needed. Specifically, a method of defining a global notion of the length of tangent vectors are of interest. The reason for this will become apparent, when properties like acceleration is defined for a curve.

Since the manifolds of interest are not vector spaces, the standard euclidean way of comparing elements can not be used. To solve this problem, the concepts off a Riemannian metric is introduced. To get a understanding of this, the method of using the dot product in euclidean space to measure distance will first be introduced. These concepts are then used, when defining the length of tangent vectors on the manifolds of interest.

7.1 The Inner Product on \mathbb{R}^n

When dealing with distances on a vector space, the inner product is essential. The formal definition of the inner product will be useful when deriving the distance method for our manifolds.

Proposition 7.1. *An inner product on the real vector space V , is a pairing of two vectors $v, w \in V$, that produces a real number $\langle v, w \rangle \in \mathbb{R}$. The inner product is required to satisfy the following three axioms for all $u, v, w \in V$, and scalars $c, d \in \mathbb{R}$.*

1. *Bilinearity, $\langle cu + dv, w \rangle = c \langle u, w \rangle + d \langle v, w \rangle$, $\langle u, cv + dw \rangle = c \langle u, v \rangle + d \langle u, w \rangle$*
2. *Symmetry, $\langle v, w \rangle = \langle w, v \rangle$*
3. *Positivity, $\langle v, v \rangle > 0$ whenever $v \neq 0$, while $\langle 0, 0 \rangle = 0$*

For \mathbb{R}^n the most common inner product is simply the dot product. That is, for $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, the inner product is given as.

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^n v_i * w_i = \mathbf{v}^T I_n \mathbf{w}$$

Given a inner product, the associated norm of a vector $\mathbf{v} \in \mathbb{R}^n$, is defined as the positive square root of the inner product of the vector with itself.

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$$

The norm of this standard inner product, is usually called the length of a vector in euclidean geometry. Since \mathbb{R}^n is a vector space, it is easy to find a vector between two elements of this space by a simple subtraction. With this resulting vector, and the associated inner product, the norm can be computed, and thus it is possible to define the distance between two elements in the space.

The expression of the inner product with the use of a matrix will be used when deriving a concept of distance on the manifold. This matrix, has some specific requirements, that can be derived from the formal definition of the inner product.

Theorem 7.2. *Every inner product on \mathbb{R}^n is given by*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T K \mathbf{y} \text{ for } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

where K is a symmetric, positive definite matrix.

Proof The proof follows from the definition of an inner product. Two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ can be written in terms of the standard basis vectors

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \dots + x_n \mathbf{e}_n = \sum_{i=1}^n x_i * \mathbf{e}_i$$

$$\mathbf{y} = y_1 \mathbf{e}_1 + y_2 \mathbf{e}_2 + \dots + y_n \mathbf{e}_n = \sum_{j=1}^n y_j * \mathbf{e}_j$$

First, by using the bilinearity of the inner product, an expression of the inner product can be given as.

$$\langle \mathbf{x}, \mathbf{y} \rangle = \left\langle \sum_{i=1}^n x_i \mathbf{e}_i, \sum_{j=1}^n y_j \mathbf{e}_j \right\rangle = \sum_{i=1}^n \sum_{j=1}^n x_i y_j \langle \mathbf{e}_i, \mathbf{e}_j \rangle$$

Define a $n \times n$ matrix K , with entries

$$k_{ij} = \langle \mathbf{e}_i, \mathbf{e}_j \rangle$$

With this, the inner product can be written as

$$\sum_{i=1}^n \sum_{j=1}^n k_{ij} x_i y_j = \mathbf{x}^T K \mathbf{y}$$

This concludes that every inner product can be written on the form $\mathbf{x}^T K \mathbf{y}$. Applying the symmetry axiom of the inner product the following relation is apparent.

$$k_{ij} = \langle \mathbf{e}_i, \mathbf{e}_j \rangle = \langle \mathbf{e}_j, \mathbf{e}_i \rangle = k_{ji}$$

Consequently, the matrix K is a symmetric matrix. The final requirement is achieved by the positivity axiom.

$$\langle \mathbf{x}, \mathbf{x} \rangle = \mathbf{x}^T K \mathbf{x} > 0 \text{ for all } \mathbf{x} \neq 0, \text{ with equality if and only if } \mathbf{x} = 0.$$

A symmetric matrix fulfilling this requirement is called positive definite. This concludes the proof.

7.2 Riemannian Manifold

Now the inner product is introduced for the tangent vectors on a manifold. The tangent space on a manifold is a vector space, thus the general expression for the inner product from *Theorem 7.2* can be used here. That is, defining a smoothly varying inner product on the tangent space at each point on the manifold. An inner product defined in such a way, is called a Riemannian metric, and the manifold is called Riemannian [1]. A method to accomplishing this is to define an inner product on the Lie algebra, and extend it to the rest of the manifold using left (or right) translation. To illustrate this, consider two elements on the Lie algebra $se(3)$, $S_1, S_2 \in se(3)$.

$$\langle S_1, S_2 \rangle_I = s_1^T G s_2$$

Here s_1, s_2 is a 6×1 vector corresponding to the elements from the skew symmetric-, and translation part of $se(3)$ of S_1, S_2 accordingly. For an arbitrary point on the manifold A , the inner product of two tangent vectors V_1 and V_2 can be defined by left translation. Here the defined left invariant vector field (9), is used to get the associated Lie algebra element.

$$\langle V_1, V_2 \rangle_A = \langle A^{-1} V_1, A^{-1} V_2 \rangle_I = s_1^T G s_2$$

7.3 Choices of Metrics

The Riemannian metric presented in the last section is not unique. The matrix in the inner product has to be chosen. This choice depends on what properties the metric should fulfill. In [12], a metric that makes physical properties be more prominent in the equations are presented. Another property that one often would like a metric to have is left, right or bi-invariance. A left invariant metric, means that when translation of an element on the right side, the metric does not change. In the interpolation this means, that properties of the curve, defined by the metric is not dependent on where the curve starts. Right invariance means that the result is not dependent of origin of the body fixed reference frame [12]. A bi-invariant metric, is not dependent on any of these dependencies.

In this paper, a metric for the ambient space $Gl(n, \mathbb{R})$ will be presented. The group $Gl(n, \mathbb{R})$ is simply all $n \times n$ matrices equipped with matrix multiplication. In [9] this metric is shown to be able to be induced to both $SO(3)$ and $SE(3)$. For the algorithm derived in this paper, only the ambient metric is used, and thus the induced metric $SO(3)$ and $SE(3)$ will not be presented hear.

For a point P on the general linear group $GL(n, \mathbb{R})$, a metric is defined for $X, Y \in T_P GL(n, \mathbb{R})$ as

$$\langle X, Y \rangle_W = Tr(X^T Y W) = Tr(W X^T Y) = Tr(Y W X^T), \quad (12)$$

where W is a symmetric positive definite $n \times n$ matrix.

The use of this type of trace expression is just a method of rewriting the form for the inner product (Theorem 7.2), for matrices. This can be shown by describing the matrices as \mathbb{R}^{n^2} row vectors, using elements from W, to create a extended $\mathbb{R}^{n^2 \times n^2}$ matrix in the inner product expression [4]. The equality of the traces, is a result of $Tr(AB) = Tr(BA)$, which can be proved easily.

This metric can be used as a norm for matrices in the ambient space [2]. In section SVD projection, a method is presented that uses this induced norm to calculate $SO(3)$ and $SE(3)$ interpolations.

7.4 Covariant Derivative and Parallel Transport

For a curve $\gamma(t)$ on a manifold, the tangent vectors got defined with a left invariant vector field (9). These tangent vectors represents the velocity of a rigid body moving along $\gamma(t)$ [4]. To involve in a more detailed study of a curve on the manifold, physical values like acceleration and jerk is of interest. To define these concepts on the curve, a method of differentiating along a curve needs to be defined. Again, to accomplish this, a way of comparing tangent vectors from different tangent spaces is needed. Since the spaces of interest lie in curved spaces, this is not trivial. In [9] the non-triviality of transporting a tangent vector at the north pole of a sphere, along a curve, to the south pole is addressed. This problem can be summarized with the fact, there is no intrinsic notion of parallelism between vectors at different points on a differentiable manifold [9].

The solution presented in [9], is to define a rule for parallel transporting tangent vectors. This rule is called an affine connection. With this, it can be possible to compare two tangent vectors at different tangent spaces. For a curve $\gamma(t)$ on the manifold, with the vector field $X(t)$ ($X(\gamma(t))$) defined along it, the parallel transport, under the rule of the affine connection

is defined as;

$$X^{t_0}(t)$$

That is, $X^{t_0}(t)$ is the parallel transport of the tangent vector $X(t)$ at the point $\gamma(t)$ to the point $\gamma(t_0)$.

With a method of comparing vectors at different tangent spaces, the derivative at a point $\gamma(t)$ on the curve can be defined. This derivative is denoted covariant derivative, and is defined as

$$\left. \frac{DX}{dt} \right|_{t_0} = \lim_{t \rightarrow t_0} \frac{X^{t_0}(t) - X(t_0)}{t}$$

Now considering two vector fields, X and Y. If the derivative of X is taken along a integral curve of the vector field Y, the derivative of X with respect to Y is achieved. Denote this derivative $\nabla_Y X$ and define it as

$$\nabla_Y X = \left. \frac{DX}{dt} \right|_{t_0}$$

Using this definition, more properties of a curve on the manifold can be expressed.

The velocity is given by the tangent vector field, which was defined earlier using the Lie algebra (10).

$$\mathcal{V}(t) = \frac{d\gamma}{dt} \quad (13)$$

Taking the covariant derivative of the velocity along a integral curve of itself the expression for the acceleration \mathcal{A} is achieved

$$\mathcal{A} = \frac{d}{dt} \frac{d\gamma}{dt} = \nabla_{\mathcal{V}} \mathcal{V}$$

Further, the jerk of a rigid body moving along a curve on the manifold can be expressed as.

$$\mathcal{J} = \frac{d}{dt} \mathcal{A} = \nabla_{\mathcal{V}} \nabla_{\mathcal{V}} \mathcal{V}$$

8 Cost function

Considering interpolation between two matrices. Different solutions to this interpolation problem, corresponds to different curves on the manifold, with the two matrices as start- and endpoint. It is easy to see that there are infinitely many solutions to this problem. That is, there are infinitely many curves that can go between the two matrices. A cost function in this context, is a function that describes how good a certain curve is, according to some properties it should fulfill. In this paper, curves that minimize distance, and acceleration will be addressed.

8.1 Minimum Distance Curves

Defining the minimum distance curves among different curve choices, as the one that minimizes a distance cost function. This cost function is simply a integral computing the distance of the curves. For a curve $\gamma(t)$, with tangent vectors $V = \frac{d\gamma(t)}{dt}$ the length of a curve between to points $\gamma(a), \gamma(b)$ is given by

$$L(\gamma) = \int_a^b \langle V, V \rangle^{\frac{1}{2}} dt$$

It can be shown [12] that a curve that minimizes the function $L(\gamma)$ for all possible curves, can be described by the differential equations

$$\begin{aligned}\frac{d\omega}{dt} &= -G^{-1}(\omega \times (G\omega)) \\ \ddot{d} &= 0\end{aligned}$$

Here G denotes the chosen metric matrix. ω corresponds to the velocity vector field (13), and \ddot{d} the 2th order derivative of the translation part.

This system of equations has an analytical solution for $G = \alpha I$ [12], which will be restated here. Considering the interpolation problem of interpolating two matrices given as

$$\gamma(0) = \begin{bmatrix} R(0) & d(0) \\ 0 & 1 \end{bmatrix}, \gamma(1) = \begin{bmatrix} R(1) & d(1) \\ 0 & 1 \end{bmatrix}$$

at $t = 0$ and $t = 1$ respectively. The solution is given by

$$\gamma(t) = \begin{bmatrix} R(t) & d(t) \\ 0 & 1 \end{bmatrix}$$

where

$$\begin{aligned}R(t) &= R(0) \exp \hat{\omega}_0 t \\ d(t) &= (d(1) - d(0))t + d(0) \\ \hat{\omega}_0 &= \log(R(0)^T R(1))\end{aligned}$$

From this result it is possible to write an algorithm to interpolate between transformation matrices, using the Rodrigues' formulas (Proposition 6.1, 6.2) for the exponential map to the Lie group, and logarithmic map for the mapping to the Lie algebra [10].

8.2 Minimum Acceleration Curves

From physics it is clear that a curve minimizing the acceleration also minimizes the force used to create the movement. Generating this type of motion can then be done by taking a minimum acceleration cost function into account. This type of curve, that minimizes the magnitude of its derivative, also have the property that it has great smoothness [11].

Define it in much the same way as the minimum distance, only the cost function takes the acceleration into account.

$$L_a = \int_a^b \langle \nabla_V V, \nabla_V V \rangle dt \quad (14)$$

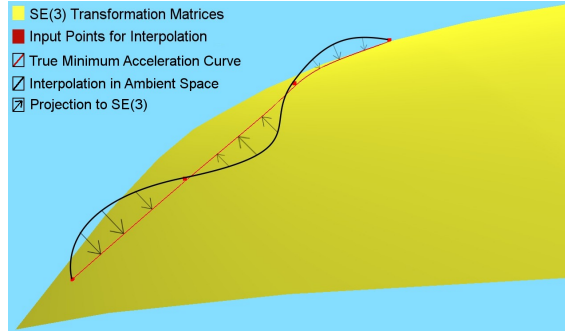
Like for minimum distance curve, a minimum acceleration curve could also be written using differential equations [12].

$$\begin{aligned}\omega^{(3)} + \omega \times \omega'' &= 0 \\ d^{(4)} &= 0,\end{aligned}$$

where $\omega^{(3)}$ is the 3th order derivative of the skew symmetric part of the velocity vector field (13) and $d^{(4)}$ the 4th order derivative of the translation part along the curve. For a special choice of initial and final velocities, these equations have an analytical solution. But the normal case is that numerical methods need to be used. In [3] the relaxation method is presented for solving the problem. Another method, that will be presented in this paper, uses a projection approach. It will be presented in the next section.

9 SVD Projection Method

Calin Belta and Vijay Kumar present in [2] a method to interpolate between matrices in the two groups $SO(3)$ and $SE(3)$ by use of Singular Value Decomposition (SVD). This method involves using common interpolation methods for vector spaces, to interpolate between two matrices. Formally said, the interpolation is done in the General Linear group of matrices $GL(n, \mathbb{R})$, that is the group off all $n \times n$ matrices equipped with matrix multiplication. The resulting matrices is then projected to the proper group. The projection of a matrix $M \in GL(n, \mathbb{R})$ to $A \in SO(n)/SE(n - 1)$ is defined as the matrix being closest to M. The distance is measured by using a norm induced by the metric defined for $GL(n, \mathbb{R})$ (12). In the rest of this paper, the results derived by Calin Belta and Vijay Kumar will be presented. All the results here are restatements of their work.



Visualization of Projection Method.

9.1 Projection on $SO(3)$

Proposition 9.1. *The projection of $A \in GL(3, \mathbb{R})$ to $R \in SO(3)$ is given by UV^T , where U, V corresponds to the matrices in the singular value decomposition of AW ($AW = U\Sigma V^T$) and W is the positive definite matrix of metric (12).*

Proof The proof involves two steps. First, the problem of choosing the closest matrix R to A , is reduced to a maximization problem. The second step, involves using SVD to prove a closed-form solution to the maximization problem.

Step 1. Considering the projection of a matrix $A \in GL(3, \mathbb{R})$ to a matrix $R \in SO(3)$. Using the definition of the projection of A as the matrix beeing closest to A , the problem of selecting R becomes a minimization problem, using metric (12) as norm.

$$\min_{R \in SO(3)} \|A - R\|_W^2$$

Evaluating by use of the induced metric

$$\|A - R\|_W^2 = \langle A - R, A - R \rangle = Tr((A - R)^T (A - R)W)$$

Which by multiplication equals

$$\|A - R\|_W^2 = Tr(A^T AW - A^T RW - R^T AW + R^T RW)$$

The expression $R^T RW = IW$ and $A^T AW$ is clearly constants in the minimization problem, so they can be ruled out. Also, by use of the relations that $Tr(A) = Tr(A^T)$, $W = W^T$,

and $Tr(AB) = Tr(BA)$, the expressions $Tr(R^T AW)$, $Tr(A^T RW)$ can be shown to be equal. That is,

$$Tr(R^T AW) = Tr((R^T AW)^T) = Tr(WA^T R) = Tr(A^T RW)$$

With these results, the problem can be reduced to

$$\min_{R \in SO(3)} -Tr(A^T RW)$$

Which again can be written as a maximization problem

$$\max_{R \in SO(3)} Tr(A^T RW)$$

Step 2, Solving the maximization problem by use of a singular value decomposition. Defining first the values values in the problem.

- $\mathcal{A} = AW$, where U, Σ, V is the SVD of \mathcal{A} , ($\mathcal{A} = U\Sigma V^T$)
- $\Sigma = \text{diag}\{\sigma_1, \sigma_2, \sigma_3\}$
- $C = R^T U$
- Column wise partitioning, $C = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3]$, $V = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$

Using these definitions, and the simple rewriting $\mathcal{A}^T = WA^T$, the expression $Tr(A^T RW)$ in the maximization problem can be written as

$$Tr(A^T RW) = Tr(WA^T R) = Tr(\mathcal{A}^T R)$$

Using the SVD of \mathcal{A} , and $C = R^T U \rightarrow C^T = U^T R$

$$Tr(\mathcal{A}^T R) = Tr((U\Sigma V^T)^T R) = Tr(V\Sigma U^T R) = Tr(V\Sigma C^T)$$

By use of the outer product the matrix in the trace can be written with a sum

$$Tr(V\Sigma C^T) = Tr\left(\sum_{i=0}^3 \sigma_i \mathbf{v}_i \mathbf{c}_i^T\right)$$

Seeing that $\sigma_i \in \mathbb{R}$, the expression can be written as

$$Tr(V\Sigma C^T) = \sum_{i=0}^3 \sigma_i (Tr(\mathbf{v}_i \mathbf{c}_i^T))$$

It is easy to see that the trace of the outer product $\mathbf{v}_i \mathbf{c}_i^T$, equals the inner product. With this, the following relation is obtained.

$$Tr(\mathcal{A}^T R) = \sum_{i=0}^3 \sigma_i (Tr(\mathbf{v}_i \mathbf{c}_i^T)) = \sum_{i=0}^3 \sigma_i \mathbf{v}_i^T \mathbf{c}_i$$

Studying the expression $\mathbf{v}_i \mathbf{c}_i^T$. Every matrix V in an SVD are orthogonal, thus $|\mathbf{v}_i| = 1$. Can also show that C is orthogonal:

$$C \cdot C^T = (R^T U) \cdot (R^T U)^T = R^T U U^T R$$

Know that R is orthogonal, U in the SVD is also always orthogonal, thus

$$C \cdot C^T = R^T I R = I$$

Which concludes that also C is orthogonal, and thus $|\mathbf{c}_i| = 1$.

Further, both \mathbf{c}_i and $\mathbf{v}_i \in \mathbb{R}^3$, so the general relation from euclidean geometry $\mathbf{v}_i^T \mathbf{c}_i = |\mathbf{v}_i| |\mathbf{c}_i| \cos(\theta)$ holds. From this, it is clear that the expression $\mathbf{v}_i^T \mathbf{c}_i$, has it maximum for $\theta = 0$. Since $|\mathbf{v}_i| = |\mathbf{c}_i| = 1$, this is equivalent of $V = C$, or $V = R^T U$. With this result, it is clear that the solution to the maximization problem is $R = UV^T$, which ends the proof.

Refer the reader to [2], for proofs and statements of left and bi invariance of the projection.

9.2 Projection on SE(3)

For a matrix $A \in GL(4, \mathbb{R})$ given as

$$A = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}, B_1 \in \mathbb{R}^{3 \times 3}, B_2 \in \mathbb{R}^{3 \times 1}, B_3 \in \mathbb{R}^{1 \times 3}, B_4 \in \mathbb{R}$$

the projection to SE(3) A is given as

$$A = \begin{bmatrix} UV^T & B_2 \\ 0 & 1 \end{bmatrix}$$

U, V is the matrices from the SVD of $B_1 W$, where W is the positive definite matrix of metric (12).

The interpolation of the translation elements of $GL(4, \mathbb{R})$ produces elements that are approved for the translation part of SE(3). Combining this, with the projection of SO(3) it is clear that the projection of SE(3), has the result as given above. A formal proof of this projection can be found in [2]. The result can be interpreted as a method of combining the interpolation of SO(3) with a euclidean translation interpolating curve. In other words, the rotation interpolation is done between two matrices $A, B \in SO(3)$, along a interpolating curve between two points $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$.

Considering how the projection methods fulfills the minimum acceleration cost function (14), the interpolation technique used in the ambient space $GL(4, \mathbb{R})$ needs to be considered. An important result in this context, is that a curve generated by a cubic spline minimizes the integral of the norm of the acceleration [11]. In [4] the projection method is compared to the relaxation method. The results derived here, show that the minimum acceleration property in the ambient space, is preserved to a satisfying extent with the projection method, and thus, making the projection method a good alternative, to a minimum acceleration interpolation method in SO(3) and SE(3).

10 Interpolation in Euclidean Space

There are many different techniques for interpolating in the Euclidean Space. As mentioned above, cubic splines, have the property that it is suited if the minimum acceleration cost function (14) is to be taken into account. Thus, the interpolation types that possess that property

would be considered here. It is worth noticing that the projection opens up to any interpolation approved in Euclidean Space, which can give interesting results. In the algorithm related to this paper [5], the natural cubic spline, Bezier Spline and Cardinal Splines are used. A short introduction to these types of interpolations will be given. Literature describing these types of interpolations and other in more detail is highly available on the Internet, and also in books concerning for example computer graphics [6].

Natural Cubic Spline A Natural Cubic Spline is basically a curve, that goes through all the points specified in the interpolation, in a manner that has a great smoothness. That is, it is C^2 continuous [6]. The algorithm for computing this type of interpolation is derived by introducing the equations describing all the limitations C^2 continuity imposes. The primary results of these limitations, is a tridiagonal matrix system that needs to be solved, for every part polynomial of the interpolation. In the context of interpolating between matrices, the values in this matrix equation is interesting. The SE(3) has totally 12 elements in each matrix, that needs to be interpolated. Denote each of these elements as a dimension. In a subinterval between two input matrices, the three diagonals are not dependent on the dimensions, thus the elements in the matrix with the row operations for solving a tridiagonal system can be computed only once for each part polynomial of the interpolation. This improvement has been done in the code associated with this paper.

Cardinal Spline The Cardinal Spline involves estimating tangent vectors at a certain point by the two adjacent points, and using this as input to create a piecewise cubic polynomials. A useful property of the Cardinal Spline is that it can be adjusted locally. That is, the curve created, will only change in the neighborhood of a point inserted. Worth noticing is that the first and last point in a Cardinal Spline interpolation only specify the tangent vectors at the beginning and end of the code, and are not a point on the curve.

Bezier Spline A Bezier Spline is actually not defined as a interpolation curve. It is rather a approximation curve, that is, the points inserted are more weights, and the curve is not needed to actually go through them. The weighting of a specific point is determined by the Bezier Blending function [6]. For a given parameter curve parameter u , and a point index k , this function gives the weighting of point k , at parameter u . The Bezier Spline has degree one less than the designated number of control points, and is thus not necessary a good choice for the minimum acceleration cost function. It is included in the implementation, purely to see what affect this type of approximating curve will have on the projection method.

11 Implementation and usage of algorithm

A short introduction to the usage of the algorithm will here be given. This involves a short explanation of the main function calls of the algorithm. Note that the different parameter values in the function calls are not presented here. Specific implementation details can be found in the code at the end of this paper and at [5].

Input of data to the interpolation can be done by inserting manually a matrix through the `inputdata()` method. Output of interpolation data is through the function `getGLInterpolation()`. Specific methods, that uses OpenGL convention for input of interpolation data is also

available, they will briefly be explained here.

Adding Rotation and Translation

In OpenGL, the function calls `glTranslated()` and `glRotated()` will left multiply the current matrix with a translation or rotation matrix respectively. The function calls `IglTranslated()` and `IglRotated()` serves the same purpose in the implementation. A call to one of them, will left multiply a matrix `m_activeMatrix` with a rotation or translation matrix. A call to `IPushMatrix()`, will add the current `m_activeMatrix` to the interpolation data. The method `ILoadIdentity()` will replace the current `m_activeMatrix` with the Identity Matrix. With this implementation method, rigid body transformations created by combining translation and rotation matrices, can easily be added to the interpolation data.

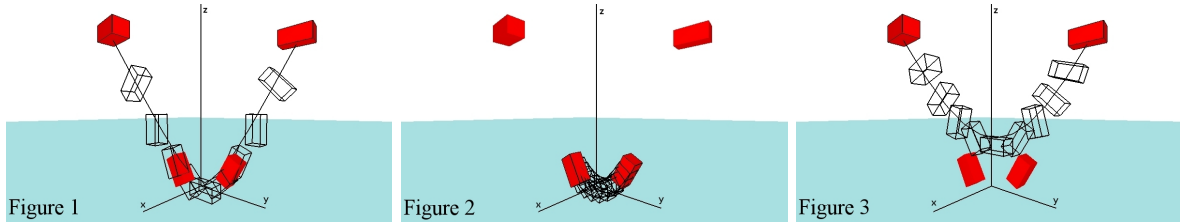
Adding Camera Position

In OpenGL the camera position can be defined with the `gluLookat()` method. What this method actually does, is to rotate and translate the entire world while the camera is kept at the same position. The matrix doing this transformation consists of rotation and translation element, and is therefore suited for use in the interpolation algorithm. In the implementation a function called `IgluLookat()` with the same parameters as the standard `gluLookat` OpenGL function, only this method adds the corresponding transformation matrix as input data for the interpolation. A call to the function `IgluLookat(float parameter)` will apply a transformation matrix at the given parameter value, and thus give exactly the same effect as calling the standard `gluLookat` function. Only difference is that the transformation matrix being applied corresponds to one computed in the interpolation.

A sample of the source code is available in the last pages of this document. Full source code with binary examples can be downloaded at [5]. The external library, The Template Numerical Toolkit was used when implementing the SVD, and for storing and computation with matrices. This library was developed at the National Institute of Standards and Technology (NIST), and the code is not subject to copyright protection and is in the public domain. Acknowledgment to their contribution is hereby given.

12 Results and Conclusion

The results are presented in a graphical manner, as screen shots from a OpenGL environment. A more mathematical explanation of the properties of the SVD projection can be found in [4]. In all the screen shots, the color read indicates orientations and positions corresponding to input data to the algorithm. Black lines, and objects corresponds to orientations and positions computed by the interpolation algorithm. A black line has also been drawn between \mathbb{R}^3 computed positions.



The input data for these three interpolations are the same. For the red cubes, from left to right, the rotation around the x axis is increased by 120 degrees, starting at 0. The positions of the cubes are given from left to right by the following vectors in \mathbb{R}^3 , $(4, 0, 5.5)^T$, $(2, 1, 1.5)^T$, $(1, 2, 1.5)^T$, $(0, 4, 5.5)^T$. 11 projections are computed on each figure. The matrix W in the metric (12), was the identity matrix.

From looking at the screen shots it is clear that what one would intuitively think the interpolation would compute, is the actual result. Also, the concepts from the different types of euclidean interpolation, seems to have been adopted quite well in the interpolations. Figure 1 is the result from using a Natural Cubic Spline in the ambient space. See that the interpolation is fitting with the start and endpoint. Figure 2 is the result of using the Cardinal Spline. Se here that the second and third input data have been nicely interpolated, and the intermediate computations corresponds to what one would expect. Finally, figure 3 is the result of the bezier spline. The approximation in the positions is clear to see, and the rotation seems reasonable.

As a description of how fast the algorithm works, a small informal test was done. The input data was just the same as above, only the amount of points/projections to compute was set between 10 and 100 000. The testing method involved simply using a clock when running the algorithm on a average personal computer. For projections amount between 10 to 1000 no more time than $1/60$ of a second was used for the computation. I took about $1/6$ of a second to compute 10 000 projections, and 100 000 projections took about 1.5 seconds. This test, although informal, shows that the algorithm is suited for use in real time applications.

Conclusion

In this paper the problem of interpolating between points with placement in \mathbb{R}^3 and rotation where addressed. Different tools from differential geometry have been defined to be able to do analysis of the different interpolation solutions. Results from the work of Vijay Kumar, and Calin Belta where used when defining solutions to the interpolation problem. By applying a cost function, two different interpolation types where defined, minimum distance and acceleration. As an approximation to the minimum acceleration, the SVD projection method presented in [2] where restated and implemented. The implementation was based on well known interpolation techniques in the ambient space $GL(4, \mathbb{R})$, together with a projection to $SO(3)$ involving an SVD. This implementation method was relatively simple to implement, and gave a great deal of freedom, by the ability to use different interpolation techniques in the ambient space. In a informal test, the algorithm showed promising results, with respect to real time applications in computer graphics. To show the use of interpolating transformation matrices, the implementation includes function calls resembling OpenGL standard function for programing with transformation matrices. With this, the interpolation algorithm should be

easy to understand for a OpenGL software developer. This includes both when placing objects with rotation and translation transformations, and for camera movement.

References

- [1] P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.
- [2] C. Belta and V. Kumar. New metrics for rigid body motion interpolation. In *Ball 2000 Symposium, University of Cambridge, UK*, 2000.
- [3] C. Belta and V. Kumar. On the computation of rigid body motion. *Electronic Journal of Computational Kinematics*, 1:2002, 2001.
- [4] C. Belta and V. Kumar. An SVD-based projection method for interpolation on SE (3). *IEEE transactions on Robotics and Automation*, 18(3):334–345, 2002.
- [5] Børge Garpestad. Source files for implementation. "http://garpestad.dyndns.org/Applications_DiffGeom_ComputerGraphics.zip".
- [6] D.D. Hearn and M.P. Baker. *Computer graphics with OpenGL*. Prentice Hall Professional Technical Reference, 2003.
- [7] R.M. Murray, Z. Li, and S.S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.
- [8] P.J. Olver and C. Shakiban. *Applied linear algebra*. Prentice Hall, 2006.
- [9] B.F. Schutz. *Geometrical methods of mathematical physics*. Cambridge University Press, 1980.
- [10] JM Selig. *Geometrical methods in robotics*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1996.
- [11] M. Zefran and V. Kumar. Interpolation schemes for rigid body motions. *Computer Aided Design*, 30(3):179–190, 1998.
- [12] M. Zefran, V. Kumar, and CB Croke. On the generation of smooth three-dimensional rigid body motions. *IEEE Transactions on Robotics and Automation*, 14(4):576–589, 1998.