3D Shape Deformations
A Lie Group Based Approach

by

SUMUKH BANSAL
201421002

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

July, 2020

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Doctor of Philosophy at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

Sumukh Bansal

Sumukh Bansal

Certificate

This is to certify that the thesis work entitled 3D SHAPE DEFORMATIONS has been carried out by SUMUKH BANSAL for the degree of Doctor of Philosophy at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.

Dr. Aditya Tatu
Thesis Supervisor

Acknowledgments

It has been an exciting journey, both personally and professionally. I want to take this opportunity to express my gratitude to the people who have supported me during this journey.

This thesis is dedicated to Prof. Aditya Tatu, who has been a constant source of inspiration for me. He has shown me how to be. No amount of words would be enough to pay my regards to him.

I would like to thank Prof. Pratik Shah for his insightful discussions. I am thankful to Prof Manjunath V. Joshi and Prof Gautam Dutta for their valuable comments on my thesis. They both have helped me to shape this thesis in the right direction. Prof. V. Sunitha, Prof. Rahul Muthu, and Prof. Srikrishnan Divakaran, It was great just to have them on campus. I will always be grateful to Prof. Bharani Kollipara for helping me out at a time when nobody could.

Life has been beautiful with beautiful people around. I feel blessed to have a loving family and wonderful friends. I would like to express my heartfelt thanks to temporary roommates: Kamal, Parth, Nirmesh, and Gaurav (yahi baatein to yaad aayegi); to dibba party: Jignesh, Prashant, and Rishikant; to Vandana (didi), Milind (bhaiya), and Madhu. Thank you everyone for your support.

Contents

Abstract

3D shapes are ubiquitous in many fundamental tasks of computer graphics and geometry processing. For many applications, new shapes have to be generated from the existing ones, for which it it imperative to understand and model shape of an object and its deformation. This thesis focuses on shape deformations and its applications. Real world 3D objects undergo complex, often non-rigid deformations. One way to model such deformations is using local affine transformations.

It is thus important for applications like 3D animation, to understand the structure of affine transformations and come up with robust and efficient computational tools on the set of affine transformations. With such tools, applications like interactive shape deformation and mesh interpolation can be effectively dealt with.

In this thesis, an interpolation framework for affine transformations, based on a Lie group representation of a tetrahedron is proposed. The proposed framework provides a intuitive closed form interpolation in all cases in contrast to existing approaches and preserves properties like isometry, reversibility, and monotonic change of volume. The proposed Lie group representation of the tetrahedron is extended to represent triangular and tetrahedral meshes. A detailed analysis of the invariance of the representation and interpolation to some choices made, is provided in the thesis.

We demonstrate the applicability of the framework for several applications like interactive shape deformation, shape interpolation, morphing, and deformation transfer. The proposed interactive shape deformation algorithm is close to being real-time, while the mesh interpolation algorithm is able to handle non-registered meshes and large deformation cases. The interactive shape deformation algorithm is amenable to data-driven methods, and we hope to explore data-driven methods using our mesh representation in near future.

CHAPTER 1

Introduction

Since the inception of computers, researchers have been trying to make machines intelligent. The core of this process is to make machines mimic the way humans perceive the world. The world around us is 3D and almost all the objects in the world have geometrical structures. One important aspect of human perception is our ability to process this geometrical information. But for machines, it is a fairly difficult task.

Until recently, the generation and reconstruction of 3D data was costly and involved computationally demanding processes. Most of the geometrical information came in the form of 2D imaging data. The last decade has seen a drastic change in the technology through which machines perceive the world. With emerging technology and new-age sensors, we now have relatively cheaper camera tools that provide an overwhelming amount of $RGB - D$ or 3D data. Hence, it has become necessary to develop tools to process and analyze such data. The branch of research which deals with acquiring, processing, and analyzing this 3D geometrical data is known as geometry processing.

Geometry processing as a research field is closely related to the fields of computer vision, computer graphics, and image processing. While computer vision is focused on extracting information about the 3D world through 2D imaging data, computer graphics focuses on the issues related to rendering 3D data in the form of 2D images. Image processing and geometry processing consist of models, algorithms, and tools to process 2D and 3D data respectively. Even though many tasks in image and geometry processing are similar, but the algorithms from image processing are usually not directly applicable to 3D data since, unlike 2D images, 3D

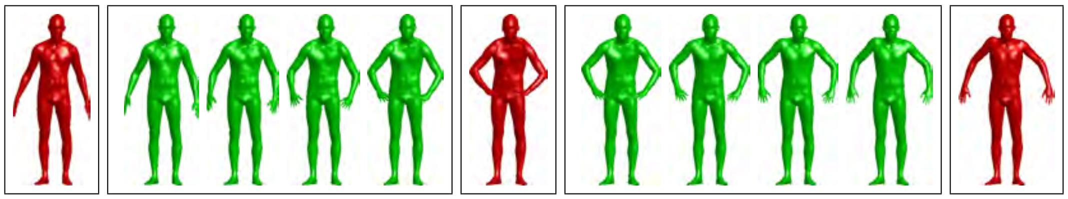

Figure 1.1: Column-1: Given 3D model. Key frames for the action sequence are shown in red while interpolated key frames are shown in green.

data is irregular.

Typically, 3D data, obtained from sensors, is in the form of 3D point clouds, depth scans, or volumetric images. A lot of preprocessing, in the form of cleaning, aligning, and registering the data, is needed which results in representations like polygonal or volumetric meshes. Each of these representations comes with its own set of pros and cons, and the choice of using a particular representation boils down to the applications under consideration. Most basic and popular shape representations are triangular and tetrahedral meshes, which are used for surface-based and volume-based shape modeling, respectively.

From an abstract point of view, geometry processing can be divided into two parts: (1) data acquisition and (2) data modeling. While data acquisition involves procedures like noise removal, alignment, registration, and topological corrections, data modeling involves choosing an appropriate representation and modifying /editing the data to get the desired shape. Usually, shape editing is done by a digital artist who puts a lot of time and effort in the process. Research in shape editing is focused on designing tools to produce realistic shapes while minimizing user input and intervention. Taking prior knowledge about shapes into account helps to produce realistic outcomes. Typically, one would want to capture the *shape space* and restrict the deformations to that space. Applications based on modifying and generating new shapes are the focus of this thesis.

Let us consider an example in a typical animation pipeline. A digital artist is given a digital model of a 3D object, say a humanoid, to animate and generate an action sequence (Figure 1.1). One may proceed by first generating important poses (called *key-frames*) in the action sequence followed by an interpolation process to generate the required sequence. The digital artist interactively deforms the model to generate the key-frames (Figure 1.2). Without the assistance from geometry processing algorithms, a digital artist may need to put in a lot of effort to design the key-frames carefully, and the process is exhaustive. The key-frames are then interpolated while preserving several properties so that the interpolation looks realistic. An associated goal is to use an animation sequence for one 3D object to produce a similar animation sequence for a different 3D object (Figure 1.3). This is referred to as *deformation transfer*.

The raw 3D data captured through sensors is in the form of a point cloud, which consists of a number of points, say N. Each point V_i is a sample on the surface of the 3D object and represents the 3D coordinates (x_i, y_i, z_i). While a dense point cloud provides a good approximation of the 3D object, a better representation of the 3D object is in the form of a mesh, where information about how a point is related to it's neighbors is taken into account. There are several ways to capture this information. In order to represent the mesh as a graph, an edge e_{ij} between each pair of adjacent points V_i and V_j is created. In triangular and other polyhedral meshes, the surface is approximated locally via triangular and/or polyhedral elements. For example, in a triangular mesh, a triangle F_{ijk}, called face, represents the surface between points V_i, V_j, and V_k (also known as vertices of the mesh). In cases where the volume of the 3D object is to be considered, tetrahedral elements F_{ijkl} are used to capture the volume between points V_i, V_j, V_k, and V_l. Thus, a point cloud, a graph, a triangular mesh, and a tet-mesh can be thought of as 0D, 1D, 2D, and 3D approximations of the actual object.

Once a digital representation, say a mesh P, of the 3D object is given, new shapes for the 3D object can be generated by computing the positions (coordinates) of the points for the new mesh P'. For each point V_i on mesh P, a new point V_i' is generated by applying a transformation $\mathcal{T}_i$ on the original point. In practice, only a set of key points are selected and transformed to the final positions on the desired mesh. *Interactive shape deformation* algorithms are used to compute transformations for the rest of the mesh. In cases where a pair of meshes P and P' is given and is to be interpolated, the transformation on each point is interpolated while preserving several important characteristics of the shape.

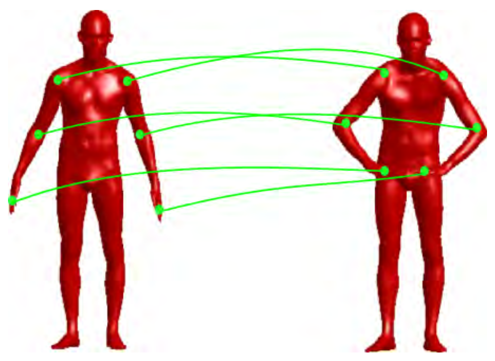
Figure 1.2: Interactive shape deformation. To generate key-frames from the given 3D model several handle markers (for example set of vertices) are picked on the given 3D model (left) and moved. The desired key-frames are obtained by using algorithms for interactive shape deformation (right).

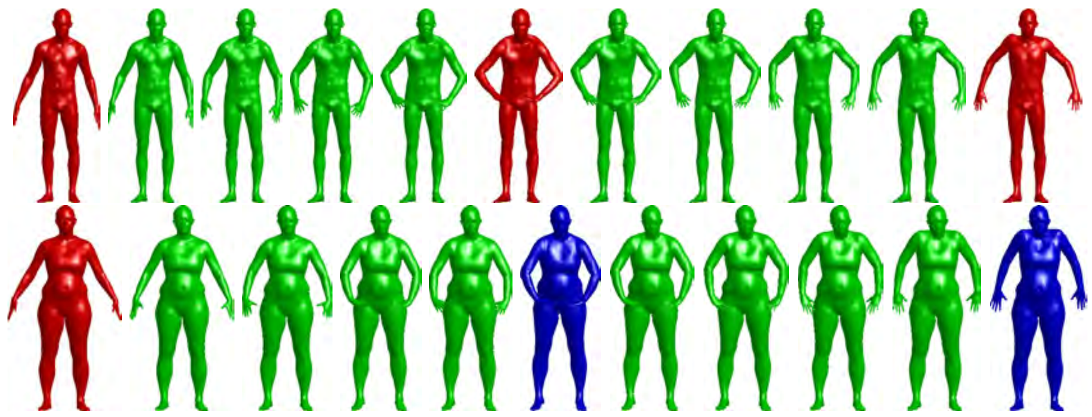
Figure 1.3: Deformation Transfer. The animation of the source model (row-1) is used to generate a similar animation of the target model (row-2). The key-frames of the source model (row-1, red) are used to generate the key frames of the target model(row-2, blue). The animation is computed by interpolating between the generated target key-frames.

The deformation required to be captured between two shapes may belong to a complex and non-linear deformation space. One way to handle such complex deformations is to decompose the deformation into locally linear/ affine deformations. Hence, deformation modeling depends on how the affine transformations are blended and interpolated. In literature, this is known as *affine interpolation* and is not a trivial task.

In this thesis, we propose algorithms for interactive shape deformation, mesh interpolation, deformation transfer, and affine interpolation using computational tools available on appropriate Lie groups. We begin with a brief survey followed by an overview of the thesis contribution.

1.1 Related Work

Geometry processing deals with 3D data acquisition, pre-processing, shape analysis, shape modeling, deformation, and other related problems. In this thesis, we focus on shape modeling and deformation.

Objects are characterized by their geometric information and external appearances, which leads to a variety of shape representations. Typically objects are represented by their boundaries. For 2D objects, parametric curves, level-set functions, medial axis, or a collection of landmark points etc. are used to represent shapes. For a recent survey refer [32]. In case of 3D objects, popular representations are point clouds, splines, triangular, polygonal, and tetrahedral meshes.

Kendall [53] defined shape as the information that is invariant to the transformations of interest. Typically, the transformations are assumed to be rigid transformations, while in case of 3D objects, for example humanoids exhibiting articulation, some non-rigid transformations may be included.

A widely used approach, proposed by Grenander and Miller [40] for object representation, known as morphable models, is to use transformations acting on the space than the object itself. Each shape is represented as a deformed version of a template shape. For 3D objects represented by its surface, a deformation can be characterized by a deformation field, where at each point of the surface, deformation is provided in form of a vector. Constraints on the deformation field yield the set of allowable shapes, and an energy-like function on the set of deformations interpreted as a cost function/metric gives rise to a *shape space*. Since translation has no effect on the shape, a simple way to represent deformation is to use gradient of displacement vectors, called *deformation gradient* [95]. For complex and large deformations, piece-wise (local) linear or affine transformations are used [82].

In what follows, we give an overview of the related work done in the problems that this thesis addresses, namely, interactive shape deformation, mesh interpolation, deformation transfer, and affine interpolation. More details on state-of-the-art methods are included in the corresponding chapters. To begin with, for the sake of completeness, we provide a brief but nonexhaustive survey on shape statistics.

1.1.1 Shape Statistics

In many applications of deformation modeling, prior information on the shapes is helpful. This information may be captured in the form of shape statistics or shape space. Given a collection of example shapes, shape variation is computed by comparing each shape to a template shape. Some of the earliest contributions to shape statistics are by Thomson [86] and Kendall [53, 32]. Traditional approaches use principal component analysis (PCA) on the deformation space. For example in the work by Allen et al. [6], a PCA on mesh vertex displacement was used and in the work by Anguelov [7], known as the SCAPE model, a PCA on the deformation between triangular faces of example and template mesh was used. In [34], a new Lie group representation for triangular meshes was proposed. Each mesh is represented by a set of transformations needed to deform the mesh to a fixed template mesh. The set of transformations forms a Lie group. Then the shape statistics are computed by performing a principal geodesic analysis (PGA) in the Lie group space.

1.1.2 Interactive Shape Deformation

Given a 3D shape, interactive shape deformation refers to the task of generating a realistic new shape with user specified constraints. Free-form deformation is one of the earliest methodologies for shape deformation [14]. The 3D space in which the object is embedded is deformed itself to achieve the desired object deformation; for surveys refer to [36, 78].

A typical approach for shape deformation is to estimate the deformed shape by minimizing a cost function based on the differences in important characteristics between the original and deformed shape, while satisfying the user specified constraints. Several linear methods are obtained by approximating the cost based on the differences between the two fundamental forms of the surface [91], gradients of the position functions [98] or Laplacian of the position functions [3]. For parametric surfaces, the new surface S' is found as a constrained minimizer of the cost function:

$$\mathcal{E}(S') = \int_{\Omega} \mathcal{D}(P(S), P(S'))\, d\Omega, \text{ such that } M(S') = 0, \tag{1.1}$$

where S is the original surface, $P(S)$ is the chosen property of the surface to be preserved (First & Second fundamental forms, for example), and $\mathcal{D}$ is an appropriate metric for the property chosen, Ω is the domain of definition for the parameters, and $M(\cdot) = 0$ are the given modeling constraints. A linear approximation followed by discretization yields the deformed mesh as a solution to a system of linear equations.

It is well known that linear methods fail to preserve geometric details under large deformations, especially deformations induced by translation (translation insensitive); for details see the survey by Botsch & Sorkine [18]. In order to overcome this shortcoming, geometric details are separately added using deformation transfer techniques as suggested in [19, 71]. Another approach consists of finding local transformations that preserve geometric details [80]. In the work by Lipman et. al. [60], known as Linear Rotation-invariant (LRI) coordinates, a rotation invariant mesh representation based on local frames and relation between adjacent local frames was proposed. A linear mesh deformation algorithm is given which solves for the unknown mesh vertices (frames followed by vertices) that satisfy the user constraints, such that the relation between adjacent local frames is preserved.

In the last decade, a few nonlinear methods have come up that alleviate some of the issues with linear methods. In the work by Sumner et al. [83], a graph is embedded in the given mesh, and the deformed mesh is obtained by blending rigid transformations computed on graph nodes via minimizing a smoothness/compatibility and modeling constraints based cost function. The density of graph nodes in the mesh affects the quality of the deformation. In another popular framework, known as As-Rigid-as-Possible(ARAP) [79], every vertex is equipped with a local rotation. Every deformation is explained using a collection of rigid transformations, as far as possible.

Another successful nonlinear method - PriMo [17], relies on local rigid transformations on a collection of prisms surrounding a given mesh to impart a global deformation. The local deformations are computed as minimizers of a non-linear elastic energy between adjacent prisms. It provides intuitive deformations, but the process is computationally expensive. The authors in [25] proposed an iterative scheme to compute an elastic deformation whose differential at any point is as close as possible to some rotation.

In order to mitigate the high costs associated with nonlinear methods while preserving the quality of deformations as far as possible, minimization of energies of the form given in Equation (1.1) is restricted to specifically chosen/designed linear subspaces. A coarser mesh leads to a reduction in the dimension of the search space. This is exploited for mesh editing in [44], where Laplacian coordinates based energy minimization is performed for a coarser control mesh. In [42], a subspace using the lower eigenmodes of the Hessian of the deformation energy plays a major role in forming the desired subspace. The deformation subspace obtained was shown to have a global presence over the mesh, to alleviate which sparse and localized deformations were computed in [65].

Skinning based methods form another class of mesh deformation algorithms; for a recent survey refer to [73]. Typically, deformations of a skeletal structure for a mesh are provided, whose weighted combinations are used to deform each vertex of the mesh. Rotations represented by Quaternions, blended using algorithms such as SLERP [76, 23] are popular approaches for skinning but can result in artifacts as one needs to handle translations separately. Unit dual quaternions [51] provide a similar representation for rotations and translations, and by separately blending the center of rotation take care of some of the artifacts found in quaternion blending. Issues like blending along true shortest paths and handling non-rigid transformations still persist.

1.1.3 Mesh Interpolation and Deformation Transfer

Mesh interpolation is the task of interpolation between a pair of meshes representing deformed shapes of an object. Mathematically, it can be seen as a boundary value problem, where given the boundary constraints, the task is to generate intermediate points. Approaches based on intrinsic shape representation such as differential coordinates, Laplacian coordinates, deformation gradients, and Linear rotation invariant coordinates [3, 60, 81, 95, 43, 27], work well for small deformations but fail for large deformation interpolation.

Complex large deformations can be interpolated on a local scale, using locally linear [4] or affine transformations [84]. Linear/ affine transformations are usually decomposed into rotational and scale/shear components to handle the intrinsic non-linearity of the individual components, where translation, scale, and shear components are interpolated linearly, while the rotational component is interpolated either via the exponential map on rotation matrices or via quaternions. When the local elements are interpolated independently, the interpolated mesh is prone to discontinuities and hence extra constraints or post-processing is needed to preserve the topology [4, 84]. Neighboring elements in a mesh are likely to undergo similar deformations, hence using a smoothness prior on the deformation yields realistic results [57]. In approaches based on interpolating dihedral angles and edge lengths [35] of the mesh, interpolated mesh has to be reconstructed which is a computationally expensive process. In some cases, patch based interpolation is used, where the mesh is clustered into patches undergoing similar deformations. Each patch is interpolated independently and the final mesh is reconstructed via stitching the interpolated patches together [85].

Another popular class of algorithms are based on representing the shapes in a shape space followed by an interpolation process. The shape space may be either explicitly specified as in the case of Lie bodies [34] or implicitly defined via an energy minimization framework, as is the case with As-Rigid-As-Possible(ARAP) approach [4]. Other approaches based on energy minimization framework [54, 9, 101, 41], involve solving of differential equations at a very fine time step and are also computationally expensive. An approach for computing discrete-time geodesics in discrete-shell shape space is proposed in [41], while Brandt *et al.*[20] generalize this for shape spaces. The geodesic is iteratively computed via a curve smoothing/fairing method and thus is not real-time. A multiresolution approach has been used to reduce the computational cost.

Approaches based on volumetric modeling of 3D objects, either as tetrahedral meshes or distance field [29] also exist. A distance field, usually based on signed distance function, is computed for source and target meshes and blended to compute distance field for intermediate objects [92, 94]. While volumetric approaches are able to deal with the topological differences in source and target meshes, these are typically computationally expensive.

A recent trend is to use *data driven* approaches which rely on a large number of examples meshes of an object for guiding the interpolation process [38, 39, 13, 88, 35]. These approaches are known to work well for large deformations but rely on a large number of similar examples.

A more involved task is of deformation transfer. Given a pair of shapes of a source object the idea is to generate similar shapes of a target object. In many cases, in order to get meaningful results, source and target meshes are assumed to be of the same topology. In other cases, a semantically meaningful dense correspondence is desired between source and target meshes, which is not trivial to compute and makes the problem challenging [19, 99]. In skinning methods, rig and weights used for the source shape are transferred to the target shape [8]. Spectral methods are based on representing meshes via harmonic basis and coefficients corresponding to low frequency basis are transferred to the target mesh [58]. Other methods are based on modifying the deformation gradients for the source model to be used for the target shapes [82].

1.1.4 Affine Interpolation

In applications related to shape and deformation modeling, affine transformations are key building blocks. Thus, a reliable and efficient affine transformation interpolation algorithm gives a framework in which tasks like mesh deformation and interpolation can be carried out.

Affine transformation consists of linear transformations like rotation, reflection, scale, translation, and shear. Given a pair of affine transformations $A(0)$ and $A(1)$, affine interpolation refers to the task of computing a smooth path $A(t), t \in [0, 1]$, such that $A(t)$ is an affine transformation. Affine interpolation can also be seen as the task of generating intermediate objects related by an affine transformation. Mathematically the task is to find the roots of a affine transformation. There are two prominent issues with affine transformations: (1) The set of affine transformations is not a linear space, assuming so produces artifacts, and (2) tools needed for interpolation on the space of affine transformation are either not available in closed form or are not defined on the complete space. A popular way of dealing with these issues is to decompose the affine transformation into simpler transformations like rotation and shear/ scale transformation [77]. But such a decomposition is not always intuitive. In [72], a closed form solution was proposed, but the solution does not exist in all the cases.

1.2 Contribution

The contribution of the thesis is as follows:

1. A framework for interpolation of affine transformations is proposed, which is based on decomposing the affine transformation into three constituent parts, i.e. rigid, scale, and shear. The proposed framework provides a intuitive closed form interpolation in all cases in contrast to existing approaches. The proposed interpolation algorithm preserves properties like isometry, reversibility, and monotonic change of volume [12].

2. A Lie group based representation for a tetrahedron is proposed which can also be used to represent a triangle as a special case. The proposed representation can be used to represent tetrahedral and triangular meshes. A detailed analysis of the representation in term of invariance to certain choices involved is provided [12].

3. The proposed mesh representation is used for several shape editing tasks like interactive shape deformation, shape interpolation, morphing, and deformation transfer [10, 11].

1.3 Thesis Organization

In Chapter 2, we introduce a Lie group representation of a tetrahedron and use it to interpolate any given affine transformation. The work done in Chapter 2 is published as [12]. In Chapter 3, the Lie group representation of tetrahedron is used to represent tetrahedral and triangular meshes based on which an interactive shape deformation framework is proposed. In Chapter 4 and 5, two applications, deformation transfer and shape interpolation, are discussed respectively. The work included in Chapter 4 and 5 has been published as [10, 11]. We conclude with a discussion and possible future work in chapter 6.

The work in this thesis requires a basic understanding of differential geometry and Lie group theory. We provide a summary of the mathematical background needed to understand the material in this thesis in Appendix A.1 and refer readers to standard texts [37, 69] for detailed discussions.

CHAPTER 2

Affine Interpolation

Affine transformations are of vital importance in many tasks pertaining to motion design and animation. Interpolation of affine transformations is non-trivial. Typically, the given affine transformation is decomposed into simpler components which are easier to interpolate. This may lead to unintuitive results, while in some cases, such solutions may not work. In this chapter, we propose an interpolation framework which is based on a Lie group representation of the affine transformation. The Lie group representation decomposes the given transformation into simpler and meaningful components, on which computational tools like the exponential and logarithm maps are available in closed form. Interpolation exists for all affine transformations while preserving a few characteristics of the original transformation. A detailed analysis and several experiments of the proposed framework are included.

2.1 Introduction

3D affine transformations play a pivotal role in many applications pertaining to computer vision, computer graphics and geometry processing. The need to interpolate affine transformations arises ubiquitously in applications involving animation design [77, 93], inverse kinematics [84, 31], motion estimation and averaging [100], image morphing, and robotics [74]. It is well known that the set of matrices $\mathbb{T}$ representing affine transformations forms a Lie group[66], interpolation on which is non-trivial. Moreover, as far as possible, the interpolated transformation should preserve the properties of the original affine transformation, such as orthogonality and area preservation.

Typical approaches decompose affine transformations into rotational and shear/ scale components, after which each component is handled separately. Steady Affine Motion (SAM) [72] for affine interpolation is found to maintain several desired properties. But in cases exhibiting large shear and rotation, the SAM interpolation does not exist.

The proposed approach decomposes any given 3D invertible, orientation-preserving affine transformation[1] into a series of intuitive transformations (each coming from a Lie group) needed to deform a tetrahedron into a fixed canonical tetrahedron. This gives a Lie group representation of the given affine transformation. Thus, an orientation- preserving 3D affine transformation can be represented as a mapping between two specific tetrahedrons, and conversely, a mapping between two oriented tetrahedrons with given correspondence as a unique 3D orientation- preserving affine transformation. This is a generalization of the Lie bodies representation of 3D triangular meshes introduced by Freifeld and Black [34]. The Lie bodies approach represents each triangle of a mesh via a specific set of transformations needed to deform a corresponding triangle in a given template mesh to the triangle under consideration. The approach proposed in this chapter represents any orientation-preserving 3D affine transformation as a decomposition into three components: a 3D rigid transformation, uniform scaling, and a specific 3D-shear, refer to Figure 2.1. The interpolation of the affine transformation is obtained using interpolation of the three components. The advantage of this decomposition is that closed-form solutions are known for interpolations of the three components. Moreover, several properties of the original affine transformation are preserved by the proposed scheme.

To summarize:

1. The proposed approach interpolates any orientation-preserving 3D affine transformation, unlike the state-of-the-art approach[72].

2. We provide a detailed analysis of the proposed interpolation scheme and show that it has several desirable properties like: (a) preserves isometry, (b)

[1]An affine transformation $\mathbb{T} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is *orientation-preserving* if $det(\mathbb{T}) > 0$.

preserves volume, and (c) yields a monotonic change in the volume.

3. The proposed interpolation scheme can also interpolate two tetrahedrons (and as a special case, triangles) related by any orientation-preserving 3D affine transformation. The interpolation is unique given a correspondence and a vertex ordering. Variation with respect to vertex ordering is analyzed in detail in this chapter.

In the next section, a review of the related work is provided. In Section 2.3, we provide details of the proposed representation of an affine transformation/ tetrahedron, along with a discussion on its existence and uniqueness. Section 2.4 describes the proposed interpolation algorithm and properties of the given affine transformation that are preserved by the interpolations obtained. Two important invariance properties related to our approach are analyzed in Section 2.5, followed by experiments and results in Section 2.6. We conclude the chapter in Section 2.7.

2.2 Related Work

There are two scenarios where the proposed approach is useful, (a) when two objects $A(0)$ and $A(1)$, also referred to as source and target, (as meshes, point clouds or triangles in 3D) are related by a unique orientation-preserving 3D affine transformation and (b) when an orientation- preserving 3D affine transformation $\mathbb{T}$ is given. In the former, intermediate objects $A(t)$, $0 < t < 1$ need to be computed, while in the latter intermediate affine transformations $\mathbb{T}(t)$, $0 < t < 1$, such that $\mathbb{T}(0) = I$ and $\mathbb{T}(1) = \mathbb{T}$ need to be computed. We first deal with the former case, and discuss ways to map the latter into the former, later in the chapter.

The set of affine transformations forms a non-linear manifold and is not compact [37]. Closed-form expressions of differential geometric tools like the exponential map are also not available. The interpolation of affine transformations is typically carried out by decomposing the given transformations into simpler transformations for which interpolation is well understood. The decomposition itself may not be intuitive, and each decomposition comes with its pros and cons. A popular decomposition consists of rotation with scale and/or shear transformations. Interpolation of rotations (SO(3)) is a well-studied problem, often solved using quaternions or matrix exponential and log maps.

In [77], Shoemake propose to decompose the affine transformation, $\mathbb{T}$, using polar decomposition $\mathbb{T} = QS$, where Q and S are rotation and stretch components (stretch being represented by symmetric positive definite matrices). Polar decomposition was found to be more stable than SVD decomposition (which is costly to compute and sensitive to small perturbations) and QR decomposition (which is stable under small perturbations but produces unintuitive rotational components). Interpolation of the rotational component is achieved using the SLERP method for quaternions [76], while the stretch component is interpolated linearly. For 2D affine transformations, authors in [50] use polar decomposition followed by exponential maps on rotations and symmetric positive-definite matrices to interpolate the rotation and stretch components, respectively. The limitation of polar decomposition comes in form of its inability to handle shear directly. A similar approach to model non-rigid deformation of meshes is used in [4].

In [1] the use of matrix exponential and log map was advocated. The matrix exponential and log map used are not in closed form and a solution based on Taylor series approximation is used. In [84], a combination of two approaches is used. The affine transformation is decomposed into rotation and stretch transformations, and while the rotation transformation is interpolated using closed-form matrix exponential and log map based on Rodrigues's formula [64], the stretch component is interpolated linearly.

A direct interpolation of a given 3D affine transformation is introduced by defining a matrix exponential and log map in [72]. The maps defined are given in closed-form, and are computationally efficient to implement. The interpolated transformations are also shown to preserve important properties like isometry and volume. However, in cases exhibiting large shear and rotation, the solution does not exist.

Similar cases arise in the context of skinning where the blending of transformations is involved. Linear blend skinning (LBS) [61] and other linear methods are known to produce candy-wrapper artifacts. Decomposition based linear blending approaches are also failure-prone as the linear blending of rotations does not necessarily produce a rotation. To alleviate limitations of linear methods, non-linear methods are proposed, where the rotation transformations are treated using quaternions [76, 52], while for Euclidean transformations dual quaternions were introduced [51]. For interpolation, the SLERP algorithm [76] is used which is iterative in nature, but for practical purposes, an approximation is found to be sufficient. While the quaternion based approaches work well for relatively large rotations, they are not well suited to model non-rigid deformation, for quaternions can only capture euclidean/rigid transformations. For a detailed survey refer to [48].

In [34], Freifeld *et al.*propose a Lie group based mesh representation for triangular meshes, henceforth mentioned as *Lie bodies*, where each triangular face of the mesh is represented by a set of transformations needed to deform a corresponding triangle on a template mesh to the triangle under consideration.

The approach proposed in this chapter decomposes a given 3D orientation-preserving affine transformation into a series of simpler transformations (each from a Lie group) needed to deform a tetrahedron to a given fixed tetrahedron, analogous to the Lie bodies framework. While the difference between our representation and the one used in the Lie bodies framework may appear minor, it is important to note that the Lie bodies framework provides a representation of triangles and not tetrahedrons. Thus, it may be used for computing interpolations between triangles (and hence a subset of orientation-preserving 3D affine transformations), but not for interpolating all possible orientation-preserving 3D affine transformations. Also, the translation component of the transformation is ignored in the Lie bodies framework, as the framework is proposed for computing shape variations and shape statistics of triangular meshes, which need to be translation-invariant. However, in applications like designing motions or interpolating transformations, translation plays a crucial role. Approaches that handle translation and the linear component of the affine transformation separately [1, 84, 77], fail to capture an intuitive interpolation path.

We also prove that our interpolation scheme has several desirable properties like isometry preservation, volume preservation and monotonicity of volume, and analyze its invariance properties in terms of choice of canonical tetrahedron and vertex orderings. These theoretical contributions are missing (for planar transformations) in [34].

In the next section, we present the proposed representation of a tetrahedron with respect to a chosen canonical tetrahedron, which can also be used to represent the orientation-preserving 3D affine transformation between the two tetrahedrons.

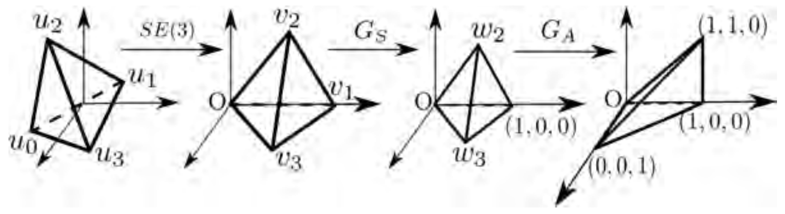

Figure 2.1: Lie group representation of a tetrahedron. (left to right): (1) A tetrahedron with vertices (u_0, u_1, u_2, u_3) (in this order) is transformed using $E = [R \;\; d] \in SE(3)$ to a tetrahedron with vertices $(\mathbf{0}, v_1, v_2, v_3)$ such that the face $(\mathbf{0}, v_1, v_2)$ lies in the xy plane, edge $(\mathbf{0}, v_1)$ aligns with the positive x axis, y coordinate of v_2 and z coordinate of v_3 are positive. (2) A 3D uniform scaling with scaling factor $s = 1/||v_1||$ is applied such that the length of the first edge vector becomes 1, to obtain the tetrahedron with vertices $(\mathbf{0}, (1, 0, 0), w_2, w_3)$. (3) Finally, a 3D shear transformation A which leaves the x-axis unchanged, maps vertex w_2 to the point $(1, 1, 0)$, and vertex w_3 to the point $(0, 0, 1)$ is applied in order to obtain the canonical tetrahedron. The tetrahedron can thus be represented as the triplet (E, A, s) with respect to the canonical tetrahedron.

2.3 Representation of a Tetrahedron using Transformations

In this work, we will denote the $n \times n$ identity matrix by I_n, a vector of n-ones or zeros by $1_n, 0_n$ respectively, and the transpose of a matrix A by A^T. We now describe the representation of a tetrahedron in $\mathbb{R}^3$. We call the tetrahedron formed by vertices $(0, 0, 0), (1, 0, 0), (1, 1, 0), (0, 0, 1)$ (in this order), a *canonical tetrahedron*. Any non-degenerate[2] tetrahedron with vertices u_0, u_1, u_2, u_3, (in this order) and

[2]A tetrahedron is *non-degenerate* if the volume enclosed by the tetrahedron is strictly positive.

same orientation[3] as the canonical tetrahedron, can be deformed into the canonical tetrahedron using a sequence of transformations described below. Refer to Figure 2.1 for a visualization of the process.

1. Rigid transformation in $\mathbb{R}^3$: The rotation component R of the rigid transformation aligns: (a) face (u_0, u_1, u_2) to the xy plane, with u_2 being mapped to a point with positive y coordinate, and (b) edge (u_0, u_1) to the positive x-axis. Due to the non-degeneracy and same orientation assumption, the vertex u_3 will get mapped to a point with positive z-coordinate. The translation vector d shifts the point Ru_0 to the origin. Let us denote the new vertices by $(\mathbf{0}, v_1, v_2, v_3)$.

2. Uniform scaling: An element of the Lie group $\mathbb{R}^+$, where the group operation is the usual multiplication between real numbers (henceforth denoted by G_S), uniformly scales the tetrahedron by $s = \frac{1}{||v_1||}$ so that the vertex v_1 is mapped to the point $(1, 0, 0)$. Let us denote the vertices of the resulting tetrahedron by $(\mathbf{0}, (1, 0, 0), w_2, w_3)$.

3. Shear: A 3D transformation that preserves the side joining vertices $\mathbf{0}$ and $(1, 0, 0)$, while aligning the vertex w_2 with the point $(1, 1, 0)$, and vertex w_3 with the point $(0, 0, 1)$. Borrowing notations from [34], we denote this subset of transformations as $G_A := \{A \in GL(3) \mid A[1\,0\,0]^T = [1\,0\,0]^T, xy$ plane is an invariant subspace of $A, A_{22}, A_{33} > 0\}$, or

$$G_A := \left\{ \begin{pmatrix} 1 & \alpha_1 & \alpha_3 \\ 0 & \alpha_2 & \alpha_4 \\ 0 & 0 & \alpha_5 \end{pmatrix} \mid \alpha_i \in \mathbb{R}, i = 1, \ldots, 5, \alpha_2, \alpha_5 > 0 \right\}.$$

Thus, each tetrahedron can be deformed into the canonical tetrahedron using the transformations described above. It appears that there are two choices involved in order to represent a tetrahedron: the choice of the canonical tetrahedron, and the choice of vertex ordering. This raises an important question: Does the interpolation path vary if we choose a different canonical tetrahedron and ordering of

[3]Two tetrahedrons with vertices $p_i \in \mathbb{R}^3, i = 0, \ldots, 3$ and $q_i \in \mathbb{R}^3, i = 0, \ldots, 3$ listed in this order, are said to have same *orientation* if $sign(det(P)) = sign(det(Q))$, where $P = [\tilde{p}_0\ \tilde{p}_1\ \tilde{p}_2\ \tilde{p}_3] \in \mathbb{R}^{4\times4}, Q = [\tilde{q}_0\ \tilde{q}_1\ \tilde{q}_2\ \tilde{q}_3] \in \mathbb{R}^{4\times4}$, and $\tilde{p}_i \in \mathbb{R}^4$ is the homogeneous coordinate representation of $p_i \in \mathbb{R}^3$.

vertices in a tetrahedron? These questions are addressed in Section 2.5, where we discuss the implications of the choices made.

2.3.1 Matrix Representation of Transformations

Since the individual transformations differ in the number of parameters, these need to be embedded in appropriate matrices before they can be composed to obtain a 3D affine transformation.

The rigid transformation $E \in SE(3)$, consisting of a rotation R and translation vector d, will be represented as a 4×4 matrix: $E = \begin{bmatrix} R & d \\ 0_3^T & 1 \end{bmatrix}$. The uniform scaling $s \in \mathbb{R}^+$ will be represented by the matrix $S = \begin{bmatrix} sI_3 & 0_3 \\ 0_3^T & 1 \end{bmatrix}$, while the 3×3 shear transformation matrices will be embedded in the upper left submatrix of I_4, i.e. $\begin{bmatrix} A & 0_3 \\ 0_3^T & 1 \end{bmatrix}$. We will denote both the 3×3 and 4×4 shear matrices with the same notation, typically A. The 3D affine transformation that deforms a given tetrahedron into the canonical tetrahedron is given by,

$$\mathbb{T} = ASE. \tag{2.1}$$

The transformation $\mathbb{T}$ is represented by the triplet $(E, A, s) \in \mathcal{M} := SE(3) \times G_A \times G_S$, and thus, any tetrahedron can be represented by an element of the set $\mathcal{M}$. Note that each of the three sets of transformations used above forms a Lie group[4], and thus the direct product $\mathcal{M}$ is also a Lie group.

2.3.2 Lie Group Representation of a Tetrahedron

The group operation on Lie group $\mathcal{M}$ is defined as: $(E_1, A_1, s_1) \cdot (E_2, A_2, s_2) = (E_1E_2, A_1A_2, s_1s_2)$. Henceforth the group operation $p \cdot q$, for $p, q \in \mathcal{M}$ will simply be denoted by pq. The Lie algebra $\mathfrak{m}$ of $\mathcal{M}$ is defined as $\mathfrak{m} = \mathfrak{se}(3) \times \mathfrak{g}_A \times \mathfrak{g}_S$, where

[4] $SE(3)$ and G_S are well known Lie groups, while the proof that G_A is a Lie group is given in Appendix A.2.

$\mathfrak{se}(3) = \left\{ \begin{bmatrix} \Omega & d \\ 0_3^T & 0 \end{bmatrix} \in M_4(\mathbb{R}) \mid \Omega \in M_3(\mathbb{R}), d \in \mathbb{R}^3, \Omega = -\Omega^T \right\}$,

$\mathfrak{g}_A = \left\{ \begin{bmatrix} 0 & \bar{\alpha_1} & \bar{\alpha_3} \\ 0 & \bar{\alpha_2} & \bar{\alpha_4} \\ 0 & 0 & \bar{\alpha_5} \end{bmatrix} \mid \bar{\alpha_i} \in \mathbb{R}, i = 1, \ldots, 5 \right\}$, and $\mathfrak{g}_S = \mathbb{R}$, are the Lie algebras of the Lie groups $SE(3), G_A$ and G_S, respectively. Thus $\mathcal{M}$ is a 12 dimensional Lie group (6 for $SE(3)$, 5 for G_A and 1 for G_S). The exponential and logarithm map for $\mathcal{M}$ is obtained by concatenating the exponential and logarithm map for each of three components: $SE(3), G_A$ and G_S, i.e., $\exp : \mathfrak{m} \to \mathcal{M} = (\exp : \mathfrak{se}(3) \to SE(3), \exp : \mathfrak{g}_A \to G_A, \exp : \mathfrak{g}_S \to G_S)$ and $\log : \mathcal{M} \to \mathfrak{m} = (\log : SE(3) \to \mathfrak{se}(3), \log : G_A \to \mathfrak{g}_A, \log : G_S \to \mathfrak{g}_S)$. The exponential and logarithm maps for the groups $SE(3)$ and G_S are known in closed form [37, 34] and are listed in the Appendix along with the derivations for exponential and logarithm maps for Lie group G_A.

Thus, a non-degenerate tetrahedron can be represented using an appropriate element of $\mathcal{M}$, with respect to a chosen canonical tetrahedron. Given an orientation-preserving affine transformation, we can apply the inverse of it on a chosen canonical tetrahedron, to obtain another tetrahedron, whose Lie group representation can be used to represent the given affine transformation. The existence and uniqueness of our representation for a tetrahedron, or for an orientation-preserving affine transformation rely on the following two results. (1) Given a pair of oriented tetrahedrons with correspondence between vertices, a unique orientation-preserving affine transformation exists transforming one tetrahedron into the other. (2) Additionally, given a fixed ordering of vertices of the tetrahedron, a unique decomposition in the proposed representation exists. These two properties allow us to represent any tetrahedron, or any orientation-preserving affine transformation, uniquely with the proposed framework. These results are discussed in the following subsection.

2.3.3 Existence and Uniqueness

We will represent a tetrahedron in $\mathbb{R}^3$ via its vertices (u_0, u_1, u_2, u_3), and we will collect the coordinates of the vertices as column vectors in a matrix. The ordering of vertices is assumed to be fixed, and with a little abuse of notation, we will use the same symbol to denote a tetrahedron, and the matrix (of size 4×4) containing its homogeneous vertex coordinates, wherever required.

Theorem 2.3.1. *Let Δ_X and Δ_Y represent two tetrahedrons in $\mathbb{R}^3$ with correspondence between vertices given, and vertices placed in a fixed order. Let $\mathbb{T}$ denote an invertible orientation-preserving affine transformation. Then the following is true:*

1. *For any pair of non-degenerate tetrahedrons Δ_X and Δ_Y with the same orientation, a unique orientation-preserving affine transformation $\mathbb{T}$ exists such that $\mathbb{T}\Delta_X = \Delta_Y$.*

2. *For a given orientation-preserving affine transformation $\mathbb{T}$, a pair of non-degenerate tetrahedrons Δ_X and Δ_Y exist such that $\mathbb{T}\Delta_X = \Delta_Y$. If the transformation $\mathbb{T}$ and one of the two tetrahedrons are given, the other tetrahedron can be computed uniquely.*

3. *Given an orientation-preserving affine transformation $\mathbb{T}$ between tetrahedrons Δ_X and Δ_Y, $\mathbb{T}$ can be uniquely decomposed into components (E, A, S) such that $\mathbb{T} = ASE$.*

Proof. Refer to Appendix A.2. □

Part (1) and (2) of Theorem 2.3.1 together show that there exists a bijection between the set of orientation-preserving affine transformations and the set of pairs of non-degenerate tetrahedrons (with one of the tetrahedrons fixed). With the decomposition given by part (3) of Theorem 2.3.1, a unique decomposition (corresponding to a fixed ordering of vertices) of the transformation $\mathbb{T}$ is possible.

2.4 Interpolation Algorithm & Properties

Given a 3D orientation-preserving affine transformation $\mathbb{T}$, let $p = (E, A, s) \in \mathcal{M}$ be its proposed Lie group representation. We propose to interpolate each of these three components on their respective Lie groups using appropriate exponential and logarithm maps, and combine the resultant components to obtain an interpolation of the given affine transformation. Note that the exponential and logarithm maps are known in closed form for each of the three components, but not for the group of general orientation-preserving affine transformations. Thus, the interpolated affine transformation is given as

$$\mathbb{T}_t = \exp(t \log A) \exp(t \log S) \exp(t \log E), \tag{2.2}$$

where $\log A$, $\log S$ and $\log E$ are logarithms of the matrices A, S and E respectively. The Lie group representation of the interpolated transformation between the identity I and transformation $\mathbb{T}$ at time $t \in [0, 1]$, is given by

$$p(t) = \exp(t \log p). \tag{2.3}$$

Note that for A and S, the exponential and log maps are considered in the form of matrices of compatible size. Closed-form expressions of these maps are given in Appendix A.3. Similarly, let Δ_p, Δ_q be two non-degenerate tetrahedrons in $\mathbb{R}^3$ with given vertex correspondence and vertex order to be interpolated, and let Δ represent the canonical tetrahedron. Let $p, q \in \mathcal{M}$ be the Lie group elements representing the tetrahedrons Δ_p, Δ_q respectively. The representation of the interpolated tetrahedron between Δ_p and Δ_q is given by

$$p \exp(t \log p^{-1} q), \;\; t \in [0, 1] \tag{2.4}$$

For an intuitive explanation refer Figure A.1 (Appendix A.1).

In [72], the interpolation of an affine transformation $\mathbb{T}$ is defined to be steady if $\mathbb{T}_t = \mathbb{T}^t, \;\; t \in \mathbb{R}$. The following result specifies conditions under which our interpolation is steady, whose proof is given in Appendix A.2.

Lemma 2.4.1. *(Steady interpolation) If the given transformation $\mathbb{T}$ consists either of only one of the three components: A, S or E, or a combination of A and S, then the proposed interpolation is steady.*

The interpolation scheme proposed in [72] is always steady, but does not exist in cases of large rotation and shear. Our solution, although not always steady, always exists. One can thus think of our scheme as a decomposition of the given affine transformation into the specified components, followed by a steady interpolation in each of the components, concluding with a composition of the components.

In the following lemmas, we provide a set of properties preserved by our interpolation algorithm. The proofs are given in Appendix A.2. These properties are essential in order to get intuitive interpolation paths.

Lemma 2.4.2. *(Volume Preservation) If the given transformation* $\mathbb{T}$ *is volume-preserving, i.e.,* $\det(\mathbb{T}) = 1$*, then the interpolated transformations* $\mathbb{T}_t, 0 \leq t \leq 1$ *defined in Equation* (2.2) *are also volume-preserving.*

Lemma 2.4.3. *(Monotonic variation of volume) If the given transformation* $\mathbb{T}$ *increases (decreases) the volume, i.e.,* $\det(\mathbb{T}) > 1$ $(\det(\mathbb{T}) < 1)$*, then the interpolated transformations* $\mathbb{T}_t, 0 \leq t \leq 1$ *monotonically increase (decrease) the volume.*

Lemma 2.4.4. *(Isometric Transformations) Assuming that the transformation* $\mathbb{T}$ *is isometric, i.e.,* $||\mathbb{T}(x-y)|| = ||x-y||, \forall x, y \in \mathbb{R}^3$*, all the interpolated transformations* $\mathbb{T}_t, 0 \leq t \leq 1$ *are also isometric.*

Lemma 2.4.5. *(Reversibility) Let* Δ_p *and* Δ_q *be two tetrahedrons. Let* $\Delta_p^q(t), t \in [0,1]$ *represent the interpolated tetrahedron at time t considering tetrahedron* Δ_p *as source, and tetrahedron* Δ_q *as target. Then* $\Delta_p^q(t) = \Delta_q^p(1-t), \forall t \in [0,1]$.

Intuitively, this rests on the fact that geodesics are reversible, which is the key construct of our interpolation algorithm.

2.4.1 Smooth Interpolation of Multiple 3D Affine Transformations

It is often required to interpolate a sequence of affine transformations $\mathbb{T}_i, i = 0, \ldots, n$ smoothly. Using the interpolation algorithm given in the previous section for each consecutive pair of transformations from the given sequence will produce a continuous but not differentiable interpolation. Lack of smoothness produces less appealing interpolations.

We summarize the algorithm proposed in [59] in order to obtain a $\mathcal{C}^1$ interpolation, and refer the reader to this chapter for higher-order smoothness. Let the decomposition of each $\mathbb{T}_i$ be given by $\mathbb{T}_i = A_i S_i E_i, i = 0, \ldots, n$. We first compute a $\mathcal{C}^1$ interpolation of individual components in the respective Lie group. We will present the algorithm for a smooth ($\mathcal{C}^1$) interpolation in the Lie group G_A; the algorithm for interpolation in the other Lie groups remain the same.

Let $N_i^A(t), t \in [0,1], i = 0, \ldots, n-1$ denote the i^{th} segment of the $\mathcal{C}^1$ interpolating curve in G_A. The constraints that ensure a $\mathcal{C}^1$ interpolation are:

$$N_0^A(0) = A_0, N_{n-1}^A(1) = A_n$$

$$N_i^A(1) = N_{i+1}^A(0) = A_{i+1}, \quad i = 0, \ldots, n-2$$
$$\frac{d}{dt} N_i^A(1) = \frac{d}{dt} N_{i+1}^A(0), i = 0, \ldots, n-2$$

These constraints represent the continuity conditions for the curve and derivative of the curve. In order to ensure that a solution to these constraints exists, we define the curve as a quadratic curve in terms of two free variables $B_i, C_i \in \mathfrak{g}_A, i = 0, \ldots, n-1$, as $N_i^A(t) = A_i \exp(tB_i) \exp(t^2 C_i), i = 0, \ldots, n-1$. The constraints on the curve yield a set of equations that the free variables must satisfy:

$$C_i = \log\left(\exp(-B_i) A_i^{-1} A_{i+1}\right), i = 0, \ldots, n-1, \tag{2.5}$$
$$B_{i+1} = A_{i+1}^{-1} A_i B_i A_i^{-1} A_{i+1} + 2C_i, i = 0, \ldots, n-2, \tag{2.6}$$

while $B_0 \in \mathfrak{g}_A$ is a free variable. In our experiments we set $B_0 = \log(A_0^{-1} A_1)$. Using appropriate exponential and logarithm maps, one can similarly determine $\mathcal{C}^1$ curves $N_i^S(t)$, $N_i^{SE(3)}(t)$ in the Lie groups G_S and $SE(3)$ respectively. The $\mathcal{C}^1$ affine transformation curve is then obtained by composing these individual curves: $\mathbb{T}_i(t) = N_i^{SE(3)}(t) \cdot N_i^S(t) \cdot N_i^A(t), i = 0, \ldots, n-1, t \in [0,1]$.

One can also use a subdivision scheme on a curve interpolating the given n transformations. The given curve can be first sampled to obtain mn transformations with $m > 1$, followed by the interpolation procedure described above to produce a smoother interpolation. This is demonstrated via experiments in Section 2.6.

In the next section, we discuss the invariance properties of the proposed interpolation scheme with respect to the choice of the canonical tetrahedron and choice of vertex ordering.

2.5 Choice of Canonical Tetrahedron and Re-Ordering

We first show that the proposed approach is invariant to the choice of canonical tetrahedron, followed by an investigation of the effects of change in vertex ordering.

2.5.1 Effect of Changing the Canonical Tetrahedron

In the proposed Lie group representation of a tetrahedron, the shear component assumes that the tetrahedron is aligned in a particular manner by the rigid component. In case a different canonical tetrahedron is chosen, the assumption on alignment may be violated. Rather than changing the structure of the shear matrix, we use the same rigid component used with the original canonical tetrahedron in the representation with respect to the new canonical tetrahedron. Thus, for a tetrahedron Δ_p, if the representation in terms of the usual canonical tetrahedron is (E_p, A_p, s_p), then its representation with respect to a new canonical tetrahedron Δ_1 is (E_p, A_{p_1}, s_{p_1}), while the actual affine transformation between Δ_p and Δ_1 is

$$\mathbb{T}_1 = E_{\Delta_1}^{-1} A_{p_1} S_{p_1} E_p, \tag{2.7}$$

where E_{Δ_1} is the rigid component in the representation of Δ_1 with respect to Δ.

Theorem 2.5.1. *(Invariance to Canonical tetrahedron): Let Δ_1 and Δ_2 be two arbitrary canonical tetrahedrons, and Δ_p and Δ_q be two tetrahedrons to be interpolated. Let (E_p, A_{p_i}, s_{p_i}) and $(E_q, A_{q_i}, s_{q_i}), i = 1, 2$ denote the Lie group representations of tetrahedrons Δ_p and Δ_q with respect to canonical tetrahedrons $\Delta_i, i = 1, 2$ respectively. Let*

$(E_{c_i}, A_{c_i}, s_{c_i}), i = 1,2$ *be the interpolated Lie group elements obtained using canonical tetrahedrons* $\Delta_i, i = 1,2$, *respectively. Then,* $(E_{c_i}, A_{c_i}, s_{c_i}), i = 1,2$ *represent the same tetrahedron.*

Proof. Refer to Appendix A.2. □

Intuitively, this is analogous to interpolation between two points in an n-dimensional Euclidean space $\mathbb{E}^n$. Once an origin is fixed, the points can be represented as vectors in $\mathbb{R}^n$ and interpolation can be carried out using the familiar convex combination of the vectors representing the points. With a different choice of origin, the representation of the two points and the interpolated point will change, but the points themselves remain the same. In our case, we interpolate between points of a manifold instead of a vector space, using a generalization of the convex combination operation in terms of the exponential and log maps. In effect, what we show is that the choice of canonical tetrahedron is analogous to the choice of origin in a Euclidean space, and therefore it does not affect the interpolation process.

2.5.2 Effect of Reordering

The Lie group representation of a tetrahedron in the proposed framework depends on the choice of the first face and edge (ordering of vertices in the face-list), since the rigid transformation is chosen to align the first face and edge. Let $O_i, i = 1,2$ represent two distinct orderings of vertices of any tetrahedron. Let $p_{O_i}, i = 1,2$ and $q_{O_i}, i = 1,2$ denote the Lie group elements representing the tetrahedrons Δ_p and Δ_q, in the vertex orderings $O_i, i = 1,2$, respectively. Let $V^p_{O_i}, i = 1,2$ and $V^q_{O_i}, i = 1,2$ denote the 4×4 matrices of vertices of the tetrahedrons Δ_p and Δ_q in homogeneous coordinates in column order $O_i,\ i = 1,2$.

Isometric Transformations & Uniform Scaling

Under the assumption that tetrahedrons Δ_p and Δ_q exhibit only isometric transformation and/or uniform scaling, there exists an $G_S \times SE(3)$ element, say (s, E)

such that $V^q_{O_i} = SEV^p_{O_i}, i = 1,2$. Thus the Lie group elements of the source and target tetrahedrons under different orderings are related by $q_{O_i} = p_{O_i}(E^{-1}, I_3, \frac{1}{s}), i = 1,2$. The tangent vectors for both orderings is then $\log((p_{O_i})^{-1} q_{O_i}) = \log((E^{-1}, I_3, \frac{1}{s})), i = 1,2$. Hence, the interpolation process remains invariant to re-ordering if the underlying transformation is Isometric and/or uniform scaling.

Non-isometric Transformations

In cases where the source and target tetrahedrons exhibit shear transformation other than rigid and uniform scaling, the interpolation path depends on the choice of the first face and edge. Let $\mathbb{T}^{O_1}_t$ and $\mathbb{T}^{O_2}_t$ be the interpolated transformations at time $t \in [0,1]$ corresponding to two different orderings of the tetrahedron vertices given by

$$\mathbb{T}^{O_i}_t = \exp(t \log A_{O_i}) \exp(t \log S_{O_i}) \exp(t \log E_{O_i}), i = 1,2,$$

where $(E_{O_i}, A_{O_i}, s_{O_i})$ are the corresponding Lie group representations, and we have assumed the canonical tetrahedron from Section 2.3. The change in the interpolated path can be represented by the change in the corresponding transformations i.e. $u_{O_1,O_2} = \max_{t \in [0,1]} ||\mathbb{T}^{O_1}_t - \mathbb{T}^{O_2}_t||_F$. We use the maximum and median of u over all orderings in order to capture the amount of change in interpolation paths under vertex re-orderings. In Figure 2.8, we empirically demonstrate that the change in path depends on the amount of shear component. A theoretical analysis of dependence of interpolation path on vertex ordering in the presence of shear needs to be carried out, so that users can make an informed choice of a particular vertex ordering.

We now provide details of the experiments and their results using the proposed framework and a comparison with other state-of-the-art approaches.

2.6 Results

In what follows, we will refer to the interpolation approach in [72] as SAM (Steady Affine Morph). Once we have the interpolated affine transformations, the results can be shown on different 3D objects. We present results using tetrahedrons and cuboids. The source code is available at `https://github.com/sumukhbansal/AffineInterpolation`.

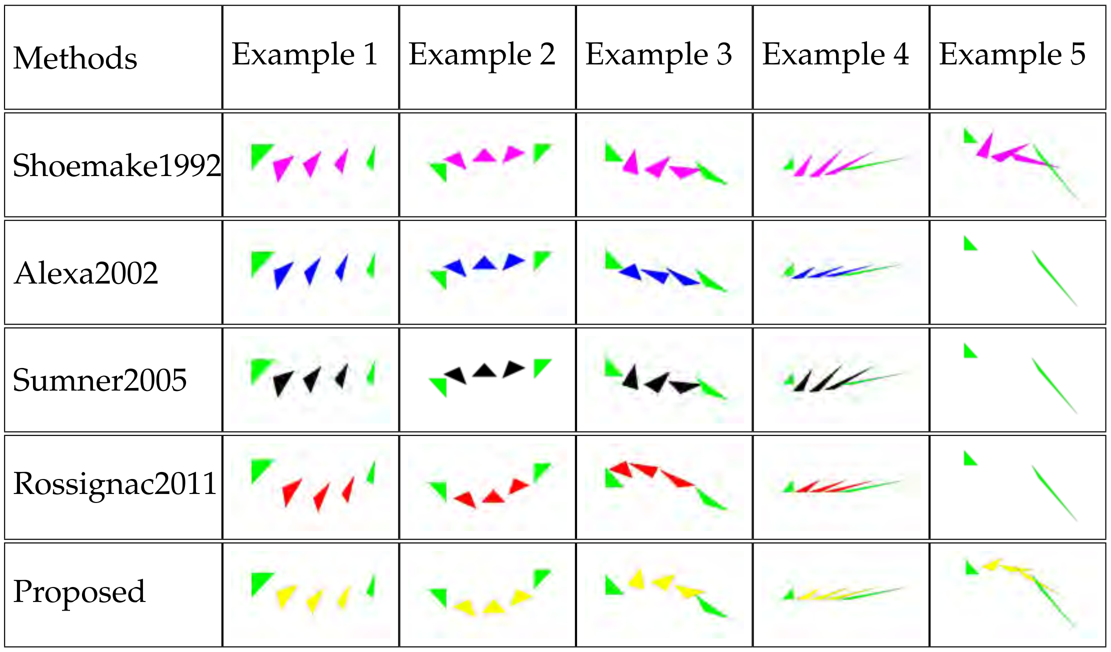

Figure 2.2: Examples of planar interpolation between triangles in $\mathbb{R}^3$. Row-wise (top-down) Interpolation results using method proposed in Shoemake1992 & Alexa2000 ([77] [4]), Alexa2002 [1], Sumner2005 [84], SAM [72] and proposed approach, for 5 different examples (column-wise). Triangles in green denote the source and target triangles. Interpolated results are produced for $t = 0.25, 0.5, 0.75$.

2.6.1 Comparison with the State-of-the-art

We begin with focusing on planar affine transformations. Such an affine transformation can be used as a map between triangles in $\mathbb{R}^2$. In Figure 2.2, we show results of planar interpolation on several cases for the following existing methods of interpolation: Shoemake1992 & Alexa2000 ([77] [4]), Alexa2002 [1], Sumner2005 [84] and SAM [72]. Translation is present in all examples. In addition, examples 2 and 4 have only rotation and shear component respectively, while examples 1, 3 and 5 have varying degrees of rotation and shear components. It can be seen that the results produced by the proposed approach and SAM produce very similar

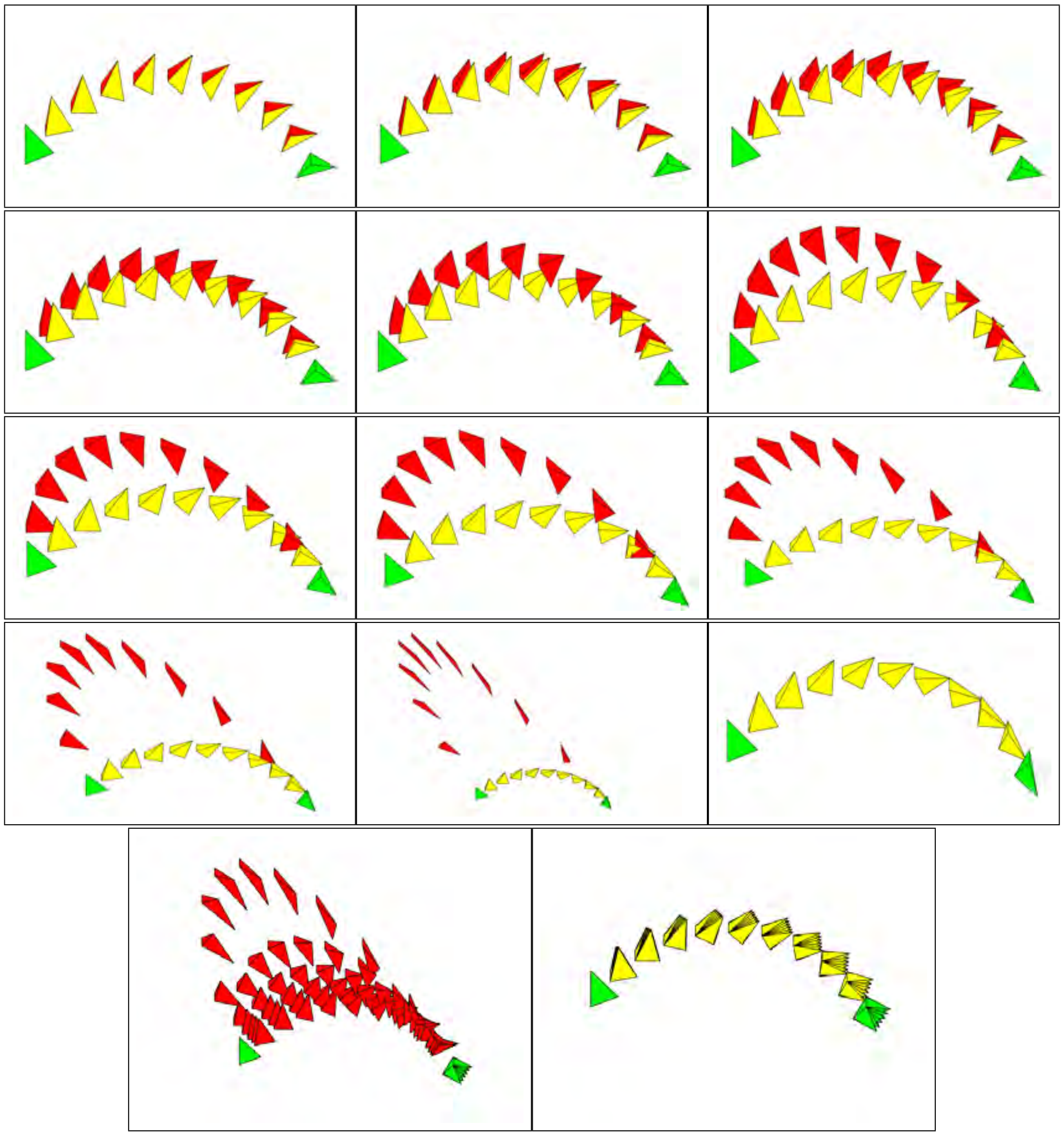

Figure 2.3: A comparison between SAM and the proposed approach. An example with a constant rotation about the z axis, while the shear (of x coordinate by y coordinate) component is gradually increased from top to bottom in Rows 1 to 4, and left to right. While the source and target tetrahedrons are shown in green, the interpolated tetrahedrons produced by SAM are shown in red and those produced by the proposed algorithm are shown in yellow. As expected, when the shear component becomes large, the SAM interpolation does not exist (Row 4, right). (Bottom row): 6 of the above 12 interpolations for SAM and proposed approach are superimposed in order to compare the difference in the obtained interpolations as the shear component increases. The trajectory corresponding to SAM changes considerably due to the steadiness requirement before it fails to produce an interpolation, while the proposed method does not produce significantly different interpolations.

and intuitive results. Also note that example 5 consists of a large shear and rotation transformation, in which case the SAM interpolation approach fails, while the proposed approach is able to produce an intuitive interpolation.

Moving onto 3D affine transformations, in Figure 2.3, we produce interpolations for affine transformations with constant rotation about the z-axis, and gradually increasing degree(top to bottom, left to right) of shear, using SAM and the proposed approach. When the shear component becomes large, SAM fails to produce an interpolation (Row 4, Column 3 in Figure 2.3). One can also observe in the bottom row of Figure 2.3 that the trajectory corresponding to SAM changes considerably before it fails to produce an interpolation, while the proposed method does not alter drastically, and produces an interpolation for all cases.

Table 2.1: Comparison of ARA measure proposed in [72], with the proposed approach for examples from Figure 2.2. ARA is computed by taking a grid sampled at an interval of .01 and for 50 intermediate poses.

Approaches	SAM	Proposed
Example 1	9.0282e-05	1.1218e-04
Example 2	2.7515e-06	2.3321e-06
Example 3	3.1247e-06	1.3392e-05
Example 4	6.9723e-05	6.9486e-05
Example 5	-	3.8634e-05

The Average Relative Acceleration (ARA) measure is used in SAM, to quantify the steadiness of the interpolation path. It is defined in [72] as: *the average magnitude (over space and time) of the acceleration vector by which the relative velocity expressed in the moving frame changes over time*. The ideal value of ARA is zero, and by construction, SAM achieves an ARA of 0. Since our algorithm does not produce steady interpolation in cases other than those mentioned in Lemma 2.4.1, we include an empirical comparison of the ARA measure of the proposed approach with SAM for the examples from Figure 2.2, in Table 2.1. The step size for the grid and the number of intermediate poses used in computing the ARA are .01 and 50, respectively. Although the proposed approach is not steady in all cases, the ARA measures are at most a magnitude off from those obtained by SAM for the corresponding discretization.

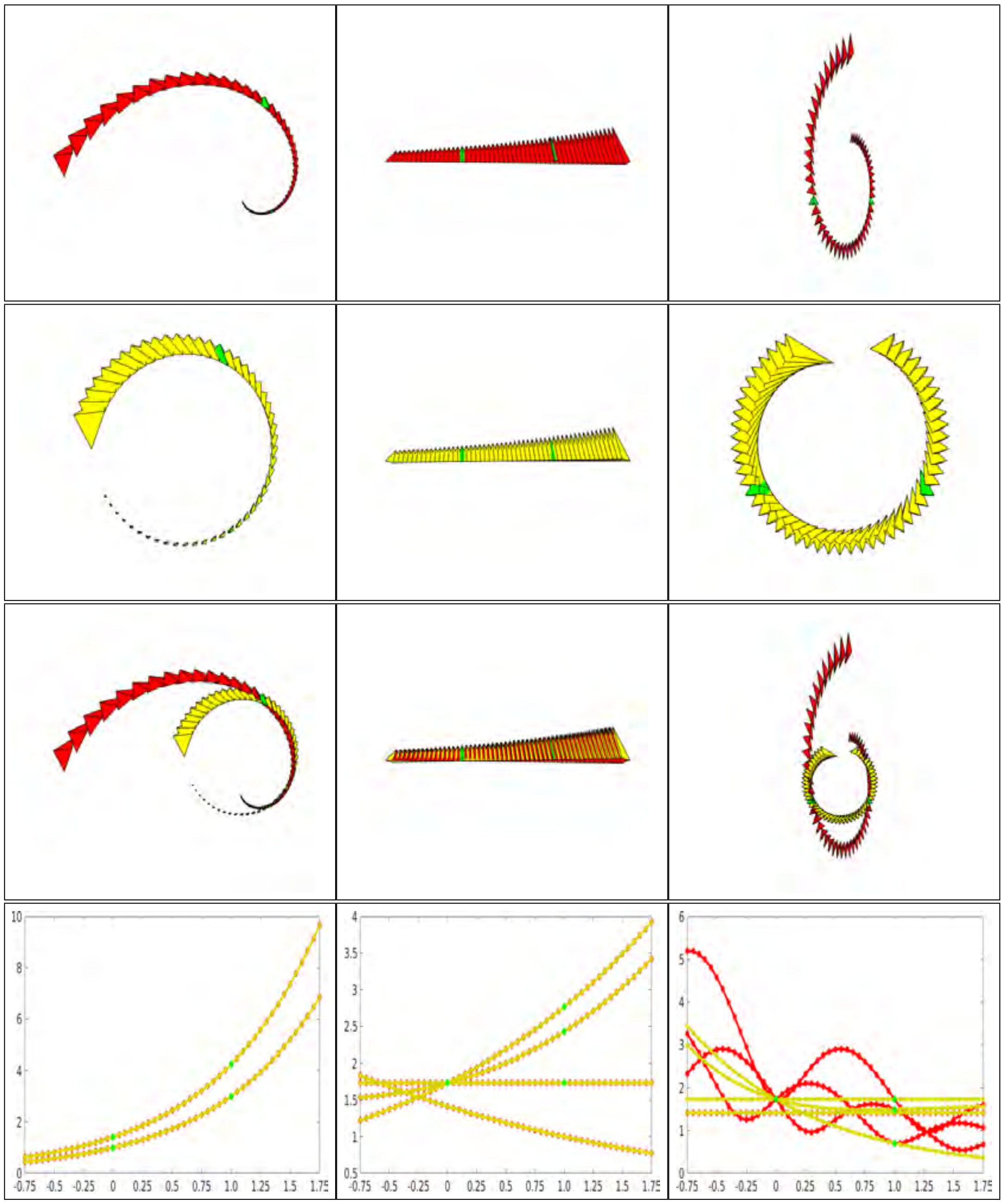

Figure 2.4: Extrapolation and interpolation. Source and target tetrahedrons are in green. Results using SAM(Row 1), the proposed approach (Row 2), comparison of SAM and the proposed approach (Row 3) and edge lengths corresponding to the edges of the interpolated base triangles and distance of fourth vertex from the base triangle (Row 4: SAM in red, proposed approach in yellow). The affine transformation used is: Column 1- Rotation, scale and translation, Column 2- shear and translation, and Column 3- Rotation, shear and translation. Results are produced for $t \in [-0.75, 1.75]$ with a sampling interval of 0.05.

2.6.2 Extrapolation

Rather than restricting the computation of $\mathbb{T}_t$ for $t \in [0,1]$, we can extrapolate the transformation by computing $\mathbb{T}_t$ for t beyond this range. In Figure 2.4, we show results of extrapolating the given 3D affine transformation using our approach and SAM. The two approaches produce the same result in case the transformation is a combination of shear and translation (refer to Figure 2.4, center column), while the translation component differs in case the transformation consists of rotation, scaling and translation (refer to Figure 2.4, left). This is evident from the last row in Figure 2.4, where we plot the edge lengths of one triangle (base) and the distance of the fourth vertex from the base triangle for each tetrahedron obtained. The edge lengths for tetrahedrons produced by SAM are shown in red, while those using our approach are shown in yellow. We emphasize the difference in the two approaches in case the transformation consists of rotation, shear, and translation. As can be seen in Figure 2.4 (right), a regular tetrahedron is rotated, translated and sheared to make it non-regular. When our approach is used to extrapolate this transformation, it yields more irregular tetrahedrons, while SAM reproduces a regular tetrahedron at some time $t > 1$, as can be observed from plots of lengths of tetrahedron sides in Figure 2.4 (bottom row, right). The lengths in the first two examples of each tetrahedron are same for both approaches (Figure 2.4, bottom row: left, center).

2.6.3 Smooth Interpolation

In Figure 2.5, smooth variations in the interpolation are shown. Interpolations and extrapolations between two pair of $3D$ affine transformations (shown in blue) are computed (shown in green). Each pair of interpolated/extrapolated transformations are then interpolated/extrapolated. The time parameter values used in all cases are $t = -.2, .2, .4, .6, .8, 1.2$. The result shows that the interpolated transformation varies smoothly in each direction. If, instead of generating interpolations row-wise and then column-wise, we first interpolate column-wise and then row-wise to generate the grid, the results only differ slightly. A theoretical analysis needs to be performed to analyze this behaviour.

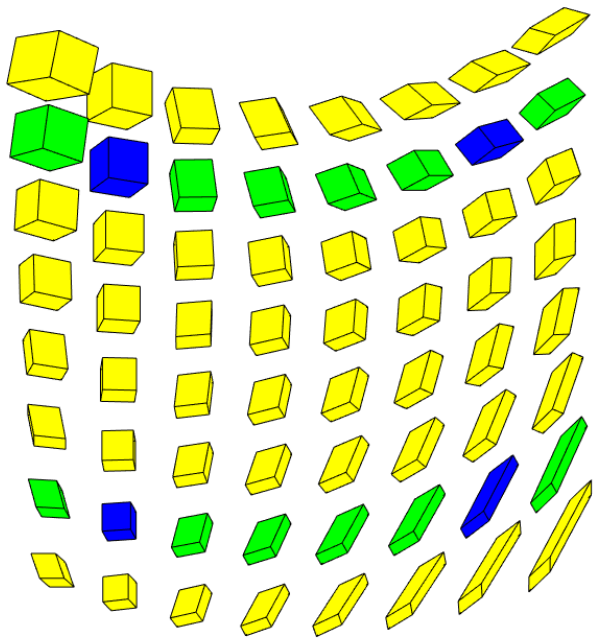

Figure 2.5: Smooth interpolation using the proposed method is shown. Firstly, affine transformations between cuboids given in blue are interpolated/extrapolated to generate cubes shown in green for time $t = -0.2, 0.2, .4, .6, .8, 1.2$. Then affine transformations between corresponding green cubes are interpolated/extrapolated.

2.6.4 Interpolating Multiple Affine Transformations

We demonstrate our algorithm to interpolate multiple affine transformations as discussed in Section 2.4.1. In Figure 2.6, we show two examples of a smooth interpolation between multiple 3D affine transformations containing scale, rigid and shear components. In order to demonstrate the subdivision scheme, we begin with continuous (but not necessarily differentiable) curves that interpolate the given affine transformations as shown in the left column of Figure 2.7. By adding transformations obtained at $t = 0.5$ to the set of given affine transformations followed by a smooth interpolation yields a smoother trajectory as shown in the right column of Figure 2.7.

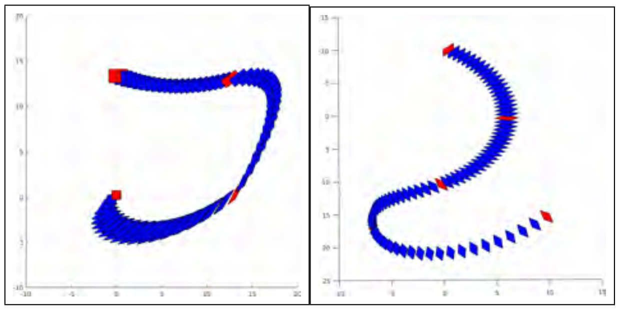

Figure 2.6: Smooth Interpolation of multiple affine transformations. A sequence of affine transformation is provided (shown in red). Each pair of consecutive affine transformations is interpolated for a time step of $dt = .05$ (shown in blue).

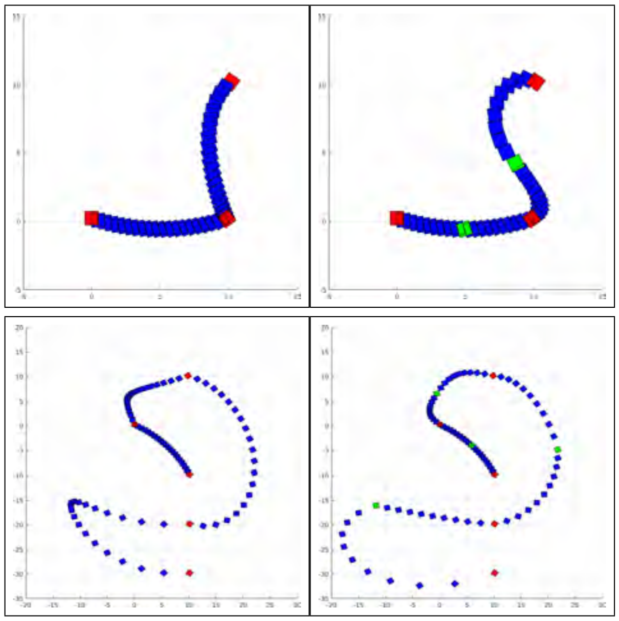

Figure 2.7: Subdivision scheme for smooth interpolation of multiple affine transformations. A sequence of affine transformation is provided (shown in red). Left column: Each pair of consecutive affine transformations sequence are interpolated for a time step of $dt = .05$ (shown in blue). Right column: In addition to the initial sequence of transformations additional intermediate transformation from column 1 (shown in green) are provided for interpolation. Each pair in the new affine transformation sequences are interpolated for a time step of $dt = .1$ As can be seen, the interpolation becomes smoother after providing additional transformations.

2.6.5 Effect of Reordering

To demonstrate the effect of reordering, we provide a comparison of the interpolation paths obtained with different vertex orderings as discussed in Section 2.5 by taking examples under two scenarios. In both scenarios, we use a regular tetrahedron inscribed in a unit sphere as the source tetrahedron. In the first case, the target tetrahedron is obtained by keeping two vertices of the source tetrahedron fixed and moving the third and fourth vertex along circles on orthogonal planes, lying on the unit sphere. For the second case, the third and the fourth vertex of the source tetrahedron are moved along the lines given by $x = constant$ and $y = constant$ respectively. Row 1 in Figure 2.8 demonstrates these deformations. The actual difference in any two orderings, say O_1 and O_2 is captured by $\max_{t\in[0,1]} ||\mathbb{T}_t^{O_1} - \mathbb{T}_t^{O_2}||_F$. Since there are 12 different orderings for a tetrahedron, we plot the maximum (Row 2) and median (Row 3) of differences of interpolation paths defined above, for each deformation, over all orderings in Figure 2.8. As can be observed from the figure, as the shear component increases (moving away from center of plot), the maximum and median deviation in path difference also increases. Actual interpolation paths for two instances of both the cases are included in Figures 2.9 (Case 1) and 2.10 (Case 2). It again points to the fact that higher the shear component in the affine transformation, higher is the difference in the paths obtained due to different re-orderings.

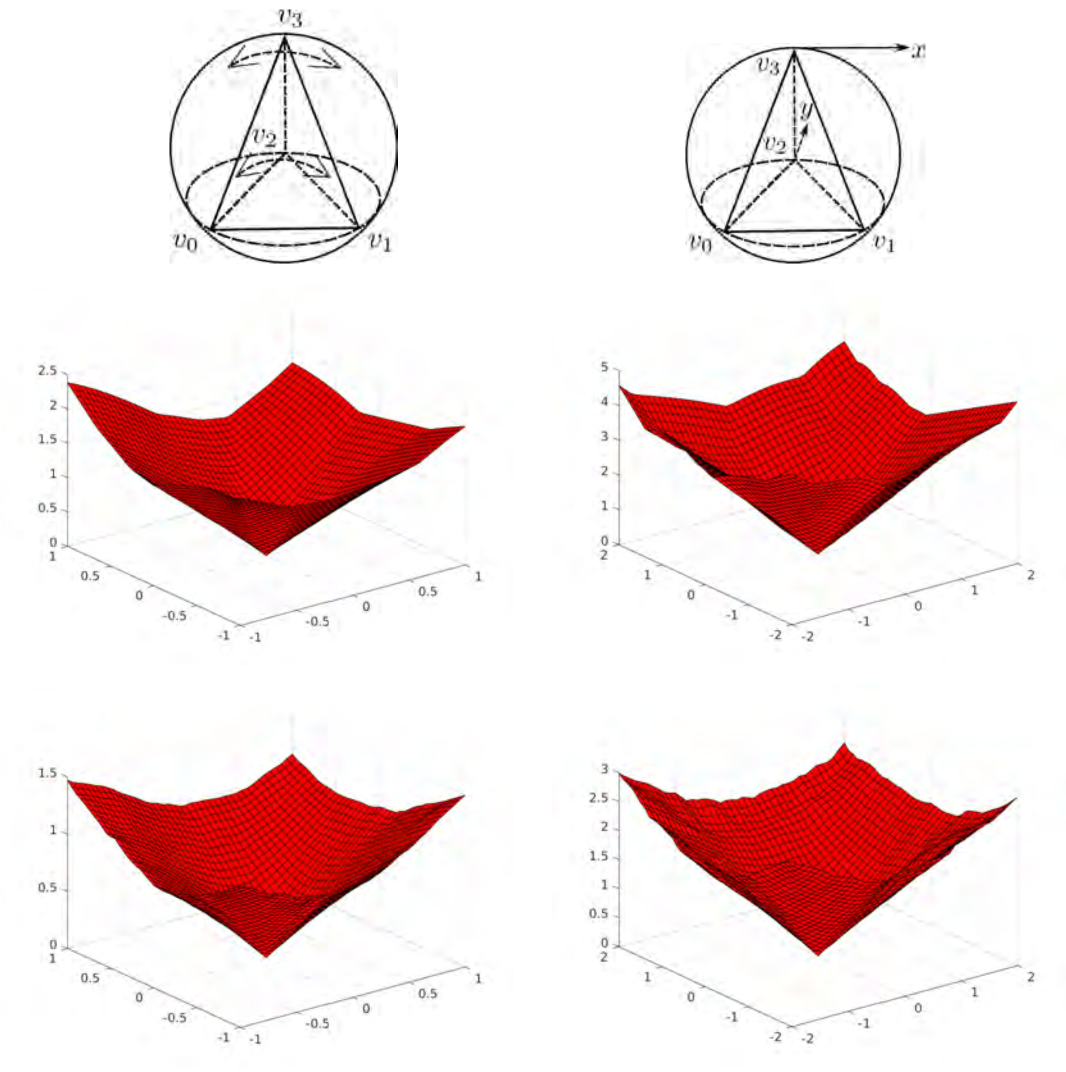

Figure 2.8: (Row 1): The target tetrahedron is obtained by (left) rotating vertex v_2 along the circle on the sphere in the plane of vertices v_0, v_1, v_2, and (right) translating vertices v_2 and v_3 along the x and y axis respectively. These deformations were chosen to directly affect the shear component in the representation. (Row 2) Norm of maximum path change for each deformation over all vertex reorderings for the two cases of deformations. (Row 3) Norm of median path change for each deformation over all vertex reorderings for the two cases of deformations. The maximum and median, both increase with the increase in the shear component. The x-axis and y-axis represent the change in moving vertices for each case (angles in radian along the orthogonal circles in case 1).

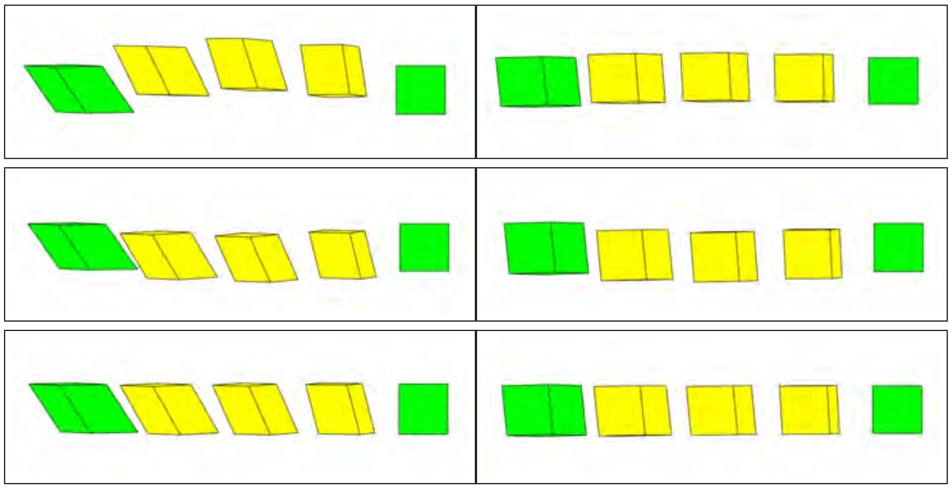

Figure 2.9: Interpolations corresponding to three vertex orderings(row-wise) for two deformations obtained by varying vertices on the circle. The two columns demonstrate results for transformations containing decreasing degree of shear. Interpolated transformations are shown for $t = .25, .5, .75$. It is evident that the path change is higher for a higher shear component. More results are provided in the supplementary material.

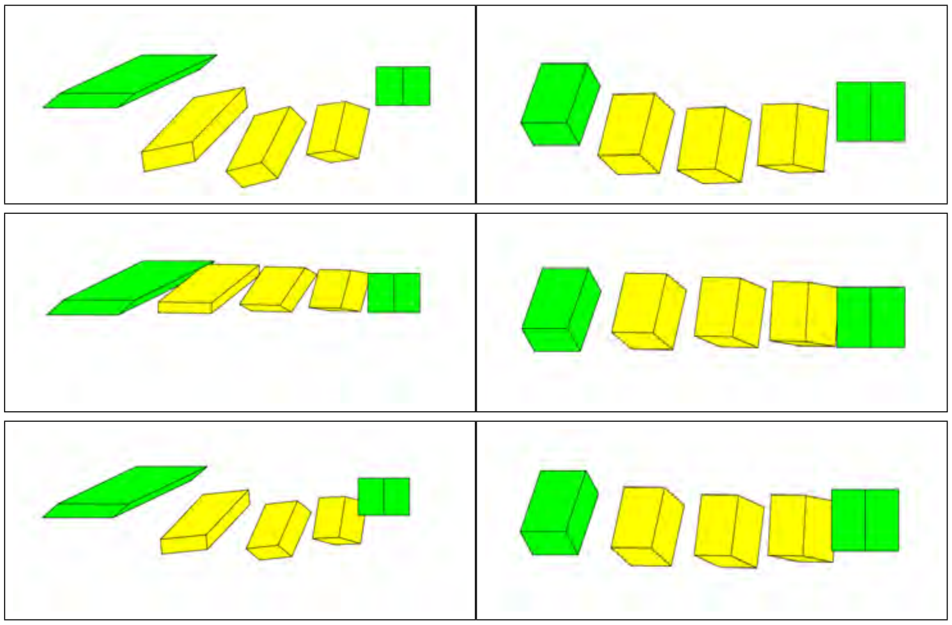

Figure 2.10: Interpolations corresponding to three vertex orderings(row-wise) for two deformations obtained by translating vertices on the coordinate axes. The two columns demonstrate results for transformations containing decreasing degree of shear. Interpolated transformations are shown for $t = .25, .5, .75$. It is evident that the path change is higher for a higher shear component. More results are provided in the supplementary material.

2.7 Conclusion and Discussion

In this chapter, we propose a framework to interpolate an orientation-preserving affine transformation, based on a Lie group representation of the transformation. The proposed framework always provides a solution, contrary to some prior work, while preserving several important properties of the original transformation, like isometry and volume preservation. The approach is invariant to the choice of canonical tetrahedron, while the degree to which vertex ordering affects the interpolation has been analyzed in detail.

Volumetric mesh deformation and interpolation methods can be developed based on our interpolation method. Our method can be applied on each pair of constituent tetrahedrons in case of mesh interpolation. This will in general not guarantee a valid volumetric mesh. An appropriate stitching process that binds individual tetrahedrons to form a valid volumetric mesh needs to be worked out. Similarly, for interactive volumetric mesh deformation, a blending process has to be developed in order to appropriately transform all tetrahedrons of the mesh.

In coming chapters, we discuss the mesh representation and it's use for several applications related to shape deformations.

CHAPTER 3

Interactive Shape Deformation

In last chapter, we discussed a Lie group representation for a tetrahedron. The proposed representation is used to define an interpolation framework for affine transformations. In this chapter we extend the proposed Lie group representation of tetrahedron for triangular meshes. Based on the new mesh representation, a framework for interactive shape deformation is provided.

Interactive shape deformation deals with algorithms and tools for deforming existing 3D models using minimum user input in a realistic or plausible manner. In this chapter, we present a one step algorithm based on a Lie group representation of triangular meshes. The user provides deformations for certain triangular faces of the mesh, based on which deformations for other triangles are interpolated. We show that our algorithm provides intuitive results in most large deformation scenarios, and in some cases better than some of the state-of-art methods. In most cases, the number of handles required is fewer than several other algorithms. We prove and analyse the invariance of our algorithm with respect to certain choices underlying the proposed framework. We also show the wide applicability of the proposed framework by demonstrating data driven deformations. While the framework in this chapter is discussed for triangular meshes, same framework is also applicable to tetrahedral meshes.

3.1 Introduction

Interactive shape deformation provides ways to deform 3D models with minimal user intervention. To interactively deform a mesh, a user chooses a subset of the mesh, called *handles*, and provides desired positions/deformations for elements of this set. A subset of the handles may be required to be fixed to their original position, while the rest are moved in space to deform the mesh. Given the positions of the handles in the new mesh (also known as the *modeling constraints*), the rest of the mesh is computed such that it has geometric properties as close as possible to the original mesh while satisfying the modeling constraints. Computing the desired deformation for a smooth surface is not difficult, the major challenge lies in preserving and correctly deforming the geometric details.

In this chapter, the proposed Lie group representation for tetrahedron is modified to represent triangular meshes and used for interactive shape deformation. The proposed framework is built on such tools that the nonlinearity does not pose any issues that plague deformation models like Linear Blend Skinning, or other surface deformation schemes that make local approximations and use vector space counterparts.

The proposed framework uses triangles as the atomic elements of the mesh; modeling constraints are defined over triangles. Triangles are a local 2D approximation to the underlying surface, and defining the overall surface deformation using triangles as handles is an alternative to vertex/point based handles. We also demonstrate possible extensions of the framework to the data driven deformations in this chapter. The proposed model achieves large deformations in one step, without having to break up the large deformations into a series of smaller deformations. Moreover, the only iterative component in the implementation is for solving a linear least square system.

To summarize,

- We propose a framework for interactive shape deformation which is based on Lie group mesh representation.

- The proposed algorithm is a one-step process; the only iterative component in the implementation is used to solve a linear least square system, which runs only once for computing the desired deformation. The algorithm is able to handle large deformations in a single step, without breaking it up into a series of small deformations.

- The algorithm is amenable to massive parallelization and thus is expected to produce real-time deformations, even for high resolution meshes. In its current serial implementation, it takes about a second to process a mesh with 40K vertices, refer to Table 3.1.

- The modeling constraints are imposed as hard constraints, and we provide several examples where fewer triangle based handles are able to achieve the deformation compared to vertex based handles.

- We discuss the invariance of the proposed framework with respect to the choice of canonical triangle and analyze the invariance under vertex re-indexing.

- The proposed framework is also amenable to possible extensions to data driven deformations. We discuss elementary (and not necessarily state of the art) algorithms for such deformations.

In the next section, we give an overview of related work on interactive shape deformation, followed by the proposed Lie group mesh representation and interactive shape deformation algorithm in Section 3.4. Section 3.5 discusses an algorithm to constrain the space of deformations by learning from a collection of deformed versions of an object. Various experiments are discussed in Section 3.6, and we conclude the chapter in Section 3.7.

3.2 Related Work

Interactive shape deformation is a crucial tool for applications like animation and object designing. The key issues are the computational efficiency of the algorithm to produce real-time and realistic results.

As discussed in Chapter 1, Linear methods are fast but fails in the case of large deformations. Refer the survey by Botsch & Sorkine [18] for more details. Rotation invariant mesh representations based on Linear Rotation-invariant (LRI) coordinates [60] and Discrete surface equations [90] were proposed, where relation between adjacent local frames was captured. A linear mesh deformation algorithm is given which solves for the unknown mesh vertices (frames followed by vertices in the former) that satisfy the modeling constraints, such that the relation between adjacent local frames is preserved. While both these methods are unable to handle large deformations in a single step, the LRI method is also translation insensitive. While non-linear approaches produce better results, the computational requirement is usually heavy. The existing state-of-the-art basically tries to mix the advantage of both linear and non-linear methods. The As-Rigid-as-Possible(ARAP) [79] is a popular framework in which every vertex is equipped with a local rotation. The algorithm explains every deformation by a set of rigid transformations, as far as possible. A regularized version, *Smooth Rotation enhanced ARAP* (henceforth SR-ARAP) encourages smoother rotation fields has been shown to be superior [57].

Linear blend skinning (LBS) is a very popular scheme for shape deformation due to the support it has in a graphics pipeline. But it suffers from defects like candy-wrapper effect. On the other hand, non-linear approaches based on minimizing ARAP (as-rigid-as-possible) energy produce appealing results but do not yield an interactive performance. A skinning based approach is given by Jacobson *et al*.[46] computes the skinning transformations by minimizing the ARAP energy. Bounded biharmonic weights [47] are used. To reduce the computations, vertices going through similar deformations are clustered. In the rest of the chapter, we refer to this approach with the acronym FAST (*Fast Automatic Skinning Transformation*). For computational efficiency, authors in [89] restricts the deformations to a smaller subspace by minimizing a smoothness based energy. The basis of the subspace is interpreted as weights used in blending while accepting the possibility of having negative weights. This approach of computing deformations by restricting the ARAP energy to a linear subspace is henceforth denoted by LS-ARAP in this chapter. While LS-ARAP allows vector-valued weights, weights used in FAST are scalars, non-negative and result in a partition of unity. Both these approaches are iterative in nature and are sensitive to the number of pseudo-handles. Moreover, these approaches produce non-intuitive results for large deformations.

Our proposed mesh representation is similar to the one used by Freifeld & Black [34], in which, a given mesh is represented with respect to a *template mesh* by encoding the set of transformations (rotation, scaling and 2D affine) required to deform each triangle of the template mesh to the corresponding triangle of the given mesh. These set of transformations form a Lie group on which manifold statistics are computed which relate to shape variations in a dataset. We refer to their mesh representation as the *Lie bodies*.

The representation used in the Lie bodies framework is insensitive to translation; we cannot afford such a characteristic, since the position of the mesh being deformed needs to be reproduced time and again. The proposed representation use 3D rigid transformations in place of 3D rotations used in the Lie bodies framework. Also, instead of encoding transformations between triangles of a given mesh and a template mesh, we encode transformations between triangles of the given mesh and a fixed canonical triangle, as discussed in Section 3.3. In triangular meshes, each triangle locally approximates the sampled surface, and hence to manipulate surfaces we use triangles as handles. The proposed algorithm is a one-step[1] procedure, and produces deformations with quality comparable against state-of-the-art methods like FAST, LS-ARAP and SR-ARAP in most situations. In fact, in most of the cases, our approach is able to handle larger deformations with fewer handles, as can be seen in Figure 3.6, and experiments provided in Section 3.6.

In the next section, we show that a triangle can be represented as a special case of the proposed representation for tetrahedron, which is then extended to obtain a Lie group representation of a triangular mesh.

3.3 Lie Group Representation of a Triangle

The triangle representation can be seen as a special case of the tetrahedral representation, where in addition to the three vertices of the triangle, a fourth vertex

[1]By one-step, we mean any deformation produced requires one iteration of the algorithm described in the chapter. We ignore implementation details such as the iterations that may be required for solving a least square problem and others.

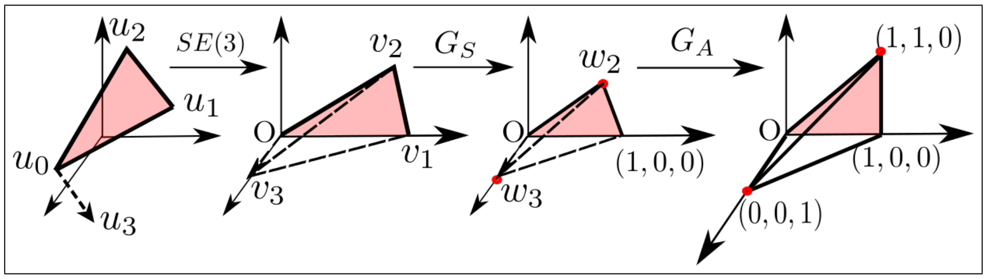

Figure 3.1: Triangle representation as a special case of tetrahedron representation.

along the normal of the triangle at a unit distance is assumed (Figure 3.1).

While the rigid and scale components of the original tetrahedron representation remain unchanged, shear component changes to

$$G_A = \left\{ \begin{pmatrix} 1 & \alpha_1 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_5 \end{pmatrix} \mid \alpha_i \in \mathbb{R}, i = 1, 2, 5, \alpha_2, \alpha_5 > 0 \right\}$$

Since the z-component of the vertex coordinates is not needed to compute the planar shear transform, the shear transformation is given by 2×2 matrix G_A (Figure 3.2).

$$G_A = \left\{ \begin{pmatrix} 1 & \alpha_1 \\ 0 & \alpha_2 \end{pmatrix} \mid \alpha_i \in \mathbb{R}, i = 1, 2, \alpha_2 > 0 \right\}$$

When composing the G_A with other components, G_A is embedded into upper left block of the identity matrix.

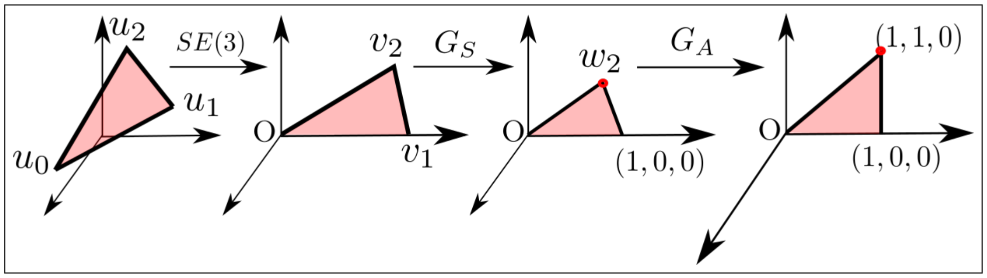

Figure 3.2: Triangle representation with out redundant parameters.

We are now ready to represent tetrahedral and triangular meshes as elements of a higher dimensional Lie group, as explained in the next subsection.

3.3.1 Lie Group Representation of a Mesh

Let P be a mesh with n faces (i.e. tetrahedral or triangular elements). Then, using the proposed representation, mesh is represented by concatenating the representations for all the tetrahedral (or triangular) elements, i.e. $(E_P, A_P, s_P) \in \mathcal{M}^n$, where n is the number of tetrahedral (or triangular) elements in the mesh.

$$E_P = \begin{bmatrix} E_{P,1} \\ E_{P,2} \\ \cdots \\ E_{P,n} \end{bmatrix}, \quad A_P = \begin{bmatrix} A_{P,1} \\ A_{P,2} \\ \cdots \\ A_{P,n} \end{bmatrix}, \quad s_P = \begin{bmatrix} s_{P,1} \\ s_{P,2} \\ \cdots \\ s_{P,n} \end{bmatrix}$$

where $(E_{P,j}, A_{P,j}, s_{P,j})$ represent the proposed representation for the j^{th} element of the P mesh. Using the fact that direct product of Lie groups is also a Lie group, $\mathcal{M}^n$ forms a Lie group. The Lie algebra of the Lie group $\mathcal{M}^n$ is similarly given by $\mathfrak{m}^n = (\mathfrak{se}(3) \times \mathfrak{g}_A \times \mathfrak{g}_S)^n$. Given an element from this Lie group, a mesh can be reconstructed by solving a least square system formed using the mesh connectivity graph [7]. The energy/cost to be minimized for the mesh reconstruction is given by

$$\sum_{i=1}^{n} ||T_i X_i - Y||^2$$

where T_i is the affine transformation corresponding to the Lie group representation of the i^{th} face in the mesh given by $T_i = A_i S_i E_i$, X_i represents the coordinates of vertices in the i^{th} face and Y represents corresponding vertex coordinates on the canonical triangle /tetrahedron. Each T_i, X_i, and Y are represented by matrices of size (4×4), (4×3), and (4×3) respectively. Equations corresponding to all the faces of mesh can be collected in a larger system of the form $TX = B$, where T, X and B are of sizes $(12n \times 4m)$, $(4m \times 1)$ and $(12n \times 1)$ with n and m being the number of mesh faces and vertices, respectively. Note that T is a sparse matrix. Solving the resulting system of equations provides the vertices of the reconstructed mesh.

In what follows, $M_P \in \mathcal{M}^n$ will denote the Lie group representation of a mesh P with n faces (tetrahedrons or triangles), while $M_{P,i}$ will denote the Lie group representation of face i in mesh P. Elements of the Lie algebra $\mathfrak{m}$ will be denoted by small letters such as d. We use exp and log to denote the exponential and logarithm map, both, for the Lie group $\mathcal{M}$ as well as for $\mathcal{M}^n$. The Lie group under consideration will be clear from the context.

Given two meshes, say P_0 and P_1, represented by M_{P_0} and M_{P_1} in the proposed representation, the log map on the $\mathcal{M}^n$ manifold is defined as $D_{M_{P_0},M_{P_1}} = \log_{M_{P_0}}(M_{P_1})$, where $\log_{M_{P_0}}(M_{P_1}) = (\log_{M_{P_0,1}}(M_{P_1,1}), \ldots, \log_{M_{P_0,n}}(M_{P_1,n}))$. The log map between the i^{th} faces is $\log_{M_{P_0,i}}(M_{P_1,i}) = \log((M_{P_0,i})^{-1}(M_{P_1,i}))$, which is in turn a concatenation of log maps in the three components: E, A and s. The vector $D_{M_{P_0},M_{P_1}}$ is a tangent vector in Lie algebra $\mathfrak{m}^n$. Similar to the log map, the exponential map (exp map) on the $\mathcal{M}^n$ manifold at point M_{P_0} is given by concatenating the exp map for individual faces, i.e., $\exp_{M_{P_0}}(D) = (\exp_{M_{P_0,1}}(d_1), \ldots, exp_{M_{P_0,n}}(d_n))$, where $D = (d_1, \ldots, d_n)$ denotes a tangent vector in Lie algebra $\mathfrak{m}^n$, and d_i is the tangent vector corresponding to the i^{th} face belonging to Lie algebra $\mathfrak{m}$. The exponential map $\exp_{M_{P_0,i}}(d_i)$ at point $M_{P_0,i}$, which is i^{th} face of the mesh P_0, is computed by taking exponential of vector d_i and then applying the left (or right) translation to point $M_{P_0,i}$ i.e. $\exp_{M_{P_0,i}}(d_i) = M_{P_0,i}\exp(d_i)$.

Lie Bodies Mesh Representation

The Lie bodies representation [34] was proposed to capture shape variations for triangular meshes and it is not defined for tetrahedral meshes. Lie bodies can be seen as a special case of the proposed Lie group mesh representation. There are two important points where Lie bodies differ from the proposed representation: (1) The translation component of the deformation is ignored in the Lie bodies representations, i.e. in place of rigid transformation, only rotational component of the transformation is used; (2) A canonical mesh is used in place of a canonical triangle. Because Lie bodies was proposed to capture shape variations with respect to a mean shape, the underlying assumptions were made. But for other applications, as will be shown in coming sections, these assumptions are not always desirable.

3.4 Interactive Deformation

To interactively deform a triangular mesh P with n faces, a user selects a set of h handles (each handle is a triangle) on the given mesh. These handles are then fixed or moved in space. Let P' be the mesh with updated handle positions that needs to be computed. We compute the Lie group representations of meshes P and P', i.e. M_P and $M_{P'}$. Here, we assume that the faces are ordered such that the first h faces correspond to the handles. The deformation of $M_{P,i}$ to $M_{P',i}, i = h+1, \ldots, n$ is then computed by propagating the deformations of the handles on the rest of the mesh. In other words, we blend the deformation prescribed on the handles to compute the deformation on non-handle faces. To model handle deformations, we propose to use deformation vector between $M_{P,k}$ and $M_{P',k}, k = 1, \ldots, h$.

In order to ensure a smooth weight function across the mesh corresponding to a particular handle, we use the bounded biharmonic weights proposed in [47]. These weights are minimizers of the Laplacian energy, non-negative, with constraints that the weight function should be 1 at the given handle (and zero at other handles), and the sum of weights of all handles at any vertex of the mesh should be 1. These constraints for weights, w_j, can be formalized as follows:

$$\begin{aligned}
&\arg \min_{w_j,\ j=1,\ldots,k} \sum_{j=1}^{k} \frac{1}{2} \int_{\Omega} ||\Delta w_j||^2 dV \\
&\textit{subjected to}: w_j|_{H_i} = \delta_{ji}, i = 1, \ldots, k \\
&\qquad \sum_{j=1}^{k} w_j(p) = 1,\ j = 1, ..., k\ \ \forall p \in \Omega \\
&\qquad 0 \leq w_j(p) \leq 1,\ j = 1, ..., k\ \ \forall p \in \Omega
\end{aligned}$$

where k is the number of handles, represented by H_i, used over the space Ω, while p is any point in Ω. Since our approach is based on computing deformations of each triangle, we first compute the dual graph of the mesh wherein each triangle is a vertex and triangles that share a side are connected via an edge. Biharmonic weights w_{ij} are then computed on this graph, where w_{ij} is the weight of handle $j = 1, \ldots, h$ for triangular face i.

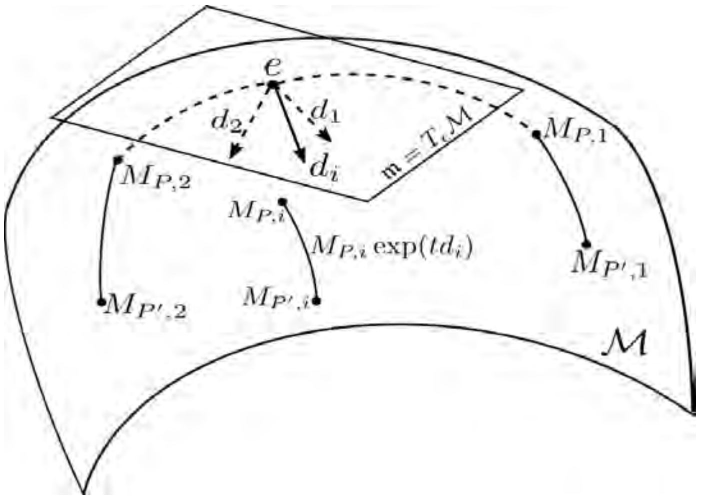

Figure 3.3: Deforming the i^{th} face in a mesh with two handles (index 1 and 2). The vectors $d_1 = log((M_{P,1})^{-1}M_{P',1}), d_2 = log((M_{P,2})^{-1}M_{P',2}) \in \mathfrak{m}$ are treated as tangent vectors of geodesics joining $M_{P,1}, M_{P',1}$ and $M_{P,2}, M_{P',2}$ respectively. The deformation tangent vector d_i is computed as a weighted average of d_1 and d_2, and deformed face i is given by $M_{P',i} = \exp_{M_{P,i}}(td_i)|_{t=1}$. The proposed algorithm tries to preserve the relation between handle and non-handle faces of the original mesh, in the deformed mesh.

To model handle deformations, we propose to use geodesics connecting $M_{P,k}$ and $M_{P',k}, k = 1, \ldots, h$. The log map defined in the previous section can be used to derive the necessary deformation vector $d_k = \log((M_{P,k})^{-1}M_{P',k})$ in the Lie algebra $\mathfrak{m}$. Such a choice is intuitive since the log map provides the group geodesics[2] connects $M_{P,k}$ with $M_{P',k}$. The vector d_k can thus be interpreted as the direction along which triangle k in mesh P deforms to triangle k in mesh P', on the manifold $\mathcal{M}$. Once the weights w_{ij} and deformation vectors $d_k, k = 1, \ldots, h$ are computed the next task is to blend them to obtain deformation vectors at non-handle mesh triangles. Note that all vectors $d_k, k = 1, \ldots, h$ belong to the same vector space $\mathfrak{m}$ making the task of blending simple.

For every non-handle face represented with Lie group element $M_{P,i}, i = h + 1, \ldots, n$, a weighted average of these tangent vectors is computed using the bounded biharmonic weights w_{ik} discussed above: $d_i = \sum_{k=1}^{h} w_{ik}d_k \in \mathfrak{m}$. The Lie group element for the corresponding deformed triangle face i in the new mesh is then computed as $M_{P',i} = \exp_{M_{P,i}}(d_i)$. This is repeated for every non-handle mesh

[2]It is also possible to directly specify $d_k \in \mathfrak{m}$, instead of providing $M_{P'_k}$. As an example, twisting the bar by rotating a handle by an angle of 3π and 2π is shown in Figure 3.8.

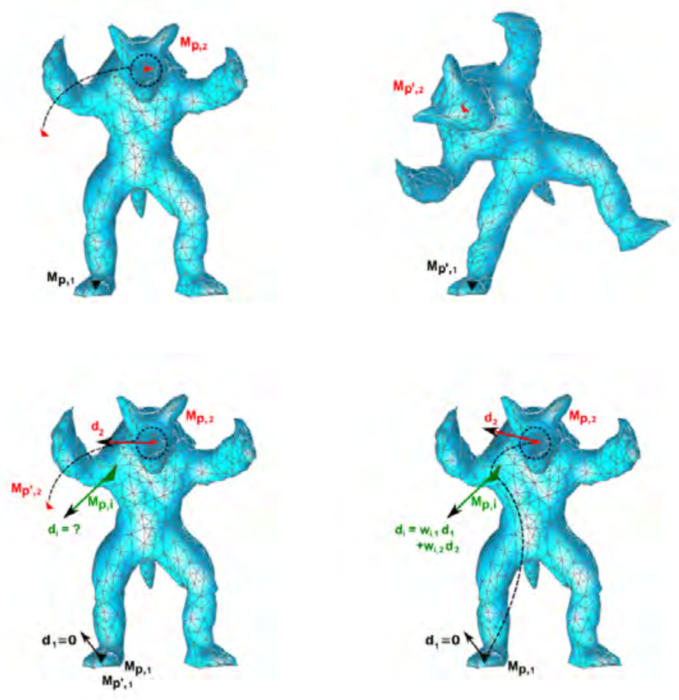

Figure 3.4: An interactive shape deformation example using two handles. Handle in black is fixed, while handle in red is moving. (From left to right) Row-1: Initial and final position of the selected handles on the original and deformed mesh i.e. $(M_{P,1}, M_{P,2})$ and $(M_{P',1}, M_{P',2})$. Row-2: Deformation vectors for handles (d_1, d_2), the deformation vector d_i at i^{th} face is computed as a weighted average of d_1 and d_2. The Lie group element $M_{P',i}$ is computed by taking the exponential of the vector d_i at the point $M_{P,i}$. A mesh reconstruction from the Lie group representation gives the deformed mesh.

face, and the mesh P' is reconstructed using the least square process mentioned earlier. An instance of deformation blending is shown in Figure 3.3 and Figure 3.4 and the process is summarized in Algorithm 1.

Note that our algorithm differs from that proposed in [2] in more than one way. The authors in [2], blend $\log(M_{P',k}), k = 1, \ldots, h$ compared to d_k in our work. Moreover, the log map that we use differs from the log map used in [2] as our log map is defined as the concatenation of log maps over a specific decomposition of a given transformation ($T \mapsto (E, A, s) \in \mathcal{M}$). For a subset of orientation preserving affine transformations in $\mathbb{R}^3$, a log map can be computed as shown in [72]. However, it does not exist for affine transformations consisting of large shear and large rotation. This is an advantage of decomposing the affine transformation, for example in the Lie group $\mathcal{M}$: $T \mapsto (E, A, s) \in \mathcal{M}$. The log map exists for all component transformations individually, thus implying that it exists for any affine transformation.

Algorithm 1 Interactive Deformation Algorithm

```
Require: Initial mesh $P$ with $n$ faces.
 1: Compute the Lie group representation $M_P$.
 2: Choose handle faces, re-order faces with handles being first $h$ faces.
 3: for all $k \in (1, \dots, h)$ do
 4:     Move or fix the handle to get the desired deformation.
 5:     Compute the Lie group representation of deformed handles: $M_{P',k}$.
 6:     Compute the tangent vector between initial and final position of the handles: $d_k = \log((M_{P,k})^{-1} M_{P',k}) \in \mathfrak{m}$.
 7: end for
    {Compute tangent vector for the non-handle faces, and Lie group representation of the deformed face.}
 8: for $i \in (h+1, ..., n)$ do
 9:     Blending: $d_i = \sum_{k=1}^{h} w_{ik} d_k$, where $w_{ik}$ is the bounded biharmonic weight of handle $k$ for face $i$.
10:     $M_{P',i} = \exp_{M_{P,i}}(d_i)$.
11: end for
    {Reconstruct the deformed mesh using the mesh tangent vector}
12: Reconstruct $P'$ from $M_{P',i}, i = 1, \dots, n$.
13: return $P'$
```

3.5 Data-driven Interactive Shape Deformation

In this section, we explore possibilities with our Lie bodies framework for guided shape deformation and shape interpolation/extrapolation. The aim is not to compete with state-of-art methods for these applications, but to show the wide applicability of the proposed framework. A collection of m meshes will be denoted by $P_j, j = 1, \dots, m$, while the corresponding Lie group representation will be denoted by $M_{P_j}, j = 1, \dots, m$. Elements of the Lie algebra $\mathfrak{m}^n$ will be denoted by capital letters such as D.

Although Algorithm 1 can help attain a wide variety of deformations, one may want to restrict the plausible deformations due to the nature of the object and the environment. For example, in the case of humanoids realistic deformations are rather limited. With the availability of data, deformation sequences, a constrained interactive deformation is feasible. In one of the popular approaches [84], a new pose is formed as a result of a nonlinear blend of the rotational deformation component and a linear blend of the shear component. By combining a discrete shell based mesh deformation with example driven deformation, Fröhlich and Botsch [35] are able to obtain visually pleasing deformations even when the modeling constraints deform the original mesh out of the span of provided mesh examples. A scale and topology independent data driven mesh deformation framework is proposed in [103].

Given the deformed meshes of an object, we learn the space of natural deformations using PCA on the deformation vectors, or what is also known as Principal Geodesic Analysis (PGA) in some fields [33, 67]. The idea is to capture the deformation between the source mesh and the given deformed meshes by principal deformation vectors. The proposed learning based approach is depicted in Figure 3.5. Given a source mesh P and deformed meshes $P_j, j = 1, \ldots, m$, com-

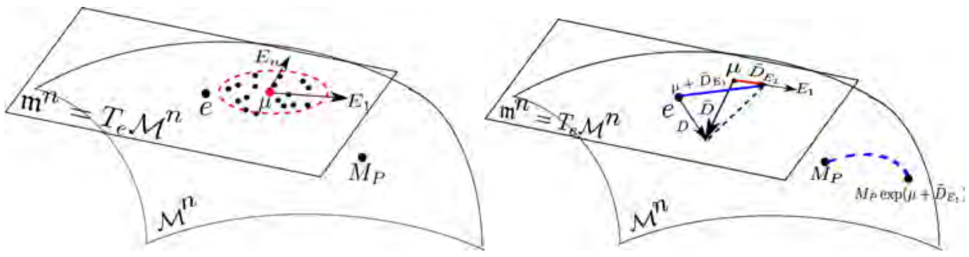

Figure 3.5: (left) PCA on the Lie algebra $\mathfrak{m}^n$, principal deformation directions $E_1, \ldots, E_n$. (right) Constraining the deformation to one principal direction. The vector $\tilde{D}_{E_1}$ (red) is obtained by projecting the vector $\tilde{D} = D - \mu$ onto the principal direction E_1, and the projected mesh is generated by using the deformation vector $\mu + \tilde{D}_{E_1}$ shown in blue, and is given by $M_P \exp(\mu + \tilde{D}_{E_1})$.

pute the corresponding deformation vectors $D_j = \log((M_P)^{-1} M_{P_j}), j = 1, \ldots, m$. Next, PCA is carried out on the set $\{D_j, j = 1, \ldots, m\}$ in the Lie algebra $\mathfrak{m}^n$. Given the handle deformations of mesh P, we compute the deformation vector D corresponding to the deformed mesh using Algorithm 1. Let μ denote the mean of the vectors $\{D_j, j = 1, \ldots, m\}$. The projection of vector $\tilde{D} = D - \mu$ onto n principal deformation directions $E_1, \ldots, E_n$ is $\Pi(\tilde{D}) = \sum_{i=1}^{n} \tilde{D}_{E_i} E_i$, where $\tilde{D}_{E_i} = \langle \tilde{D}, E_i \rangle = Tr(\tilde{D}^T E_i)$, where $Tr(\cdot)$ denotes the trace operator. The Lie group representation of the deformed mesh is then computed as $M_P \ \exp(\mu + \Pi(\tilde{D}))$, and the final mesh can be computed by the least square based reconstruction process mentioned earlier. Figure 3.5 demonstrates the algorithm with projection on the first principal deformation direction, and the algorithm is summarized in Algorithm 2. A disadvantage of this procedure is that the projection may violate the modeling constraints.

Algorithm 2 Interactive deformation algorithm using learning

```
Require: Source mesh $P$ and learning data meshes $P_j$, $j = 1, ..., m$.
 1: Find the tangent vector between $M_P$ and $M_{P_j}$.
 2: for all $j = 1, ..., m$ do
 3:     Compute $D_j = \log((M_P)^{-1} M_{P_j}) \in \mathfrak{m}^n$.
 4: end for
 5: Compute principal directions: $E_i = PCA(\{D_1, \ldots, D_m\}), i = 1, \ldots, q$.
 6: Using Algorithm 1, find the deformation vector $D$.
    { Project the computed tangent vector to the learned PCA space.}
 7: $\Pi(\tilde{D}) = \sum_{i=1}^{q} \tilde{D}_{E_i} E_i$, where $\tilde{D}_{E_i} = \langle D - \mu, E_i \rangle$.
 8: $M_{P'} = M_P \exp(\mu + \Pi(\tilde{D}))$.
 9: Reconstruct $P'$ from $M_{P'}$.
10: return $P'$.
.
```

3.6 Results and Discussion

Several experiments are performed on standard meshes (Armadillo, Cactus, Bar, Cylinder) provided in [17, 79], the Cigar mesh from [46], and Octopus mesh from [60], and high resolution Armadillo and Dragon meshes from the Stanford 3D scanning repository. For data driven interactive shape deformation, we have used humanoid meshes provided by Pons-Moll et al. [68]. All simulations were carried out on the Libigl framework [49], running on an Intel5 8GB RAM, 1.6 GHz system. The mesh resolution and timing details of each experiment are provided at the end of this section in Table 3.1. Some of the MATLAB figures for results shown in this section, along with real-time video recordings of interactive deformation on simple meshes, are provided at https://github.com/sumukhbansal/InteractiveShapeDeformation.

Justifying mesh representation: The choice of the Lie group $SE(3)$ in place of $SO(3)$ as proposed in the Lie bodies framework is to incorporate the transla-

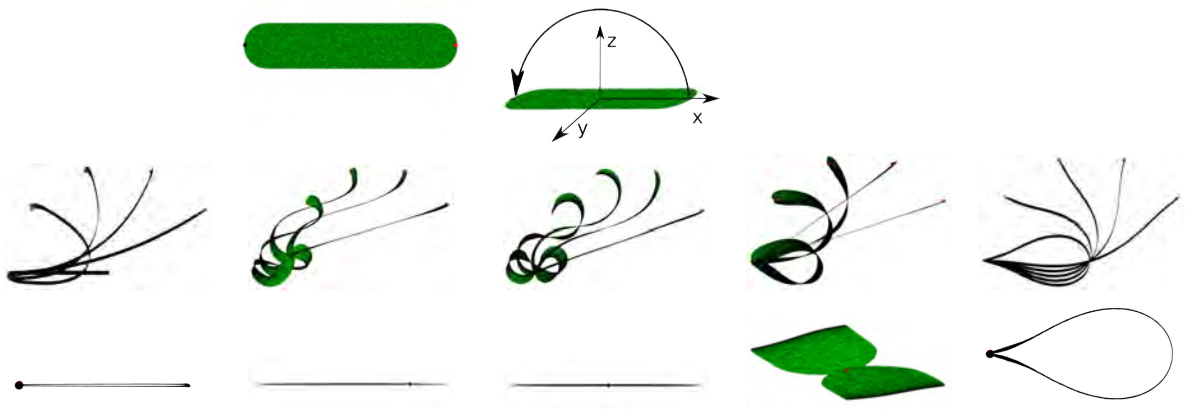

Figure 3.6: Folding the Cigar. (Top Row) Left: Original Cigar mesh, fixed handle (one) in shown in black and moving handle (one) is shown in red. Right: The moving handle is rotated about the y axis with origin being the fixed point by an angle of upto 180 deg. Results of the following algorithms used are shown from left to right: LRI, FAST, LS-ARAP, SR-ARAP and proposed approach. (Center row) Folding is achieved by breaking down the complete rotation of 180 deg in to 5 steps of 36 deg each. LRI and LS-ARAP produce a smooth and intuitive deformation, but fail in the last step for achieving a deformation of 180 deg; note the cigar collapsing in the $x - y$ plane in the last step. FAST and SR-ARAP are unable to keep the cigar from twisting, possibly due to insufficient handles. (Bottom row) Results of rotation of 180 deg in a single step. While LRI produce the same result (collapsed cigar) as earlier, FAST, LS-ARAP and SR-ARAP produces a different result as compared to the last incremental step from the center row. Here too the cigar collapses. Using our algorithm the cigar does not twist through the incremental rotation sequence, and produces the same result even in the one-step 180 deg deformation. For comparison on other deformations of the Cigar mesh, refer to Figures 3.10,3.11,3.12.

tion component in the process of mesh deformation. The information about the position of the mesh is not captured in the representation $(SO(3) \times G_A \times S)^F$ of a mesh in the Lie bodies framework. One may alleviate this particular problem with the Lie bodies representation by imposing handle positions as hard or soft constraints. When using hard constraints, the rest of the mesh still has an unknown translational component, thereby producing spikes and discontinuities, refer to Figure 3.7 (left), while soft constraints may lead to modeling constraints being not satisfied. With a large penalty on violating these constraints, the modeling constraints may be satisfied to a desired degree of accuracy, but it may lead to unpleasant shape distortion as shown in Figure 3.7 (center). Since the proposed framework explicitly captures the position of the mesh, it has no such issues, refer

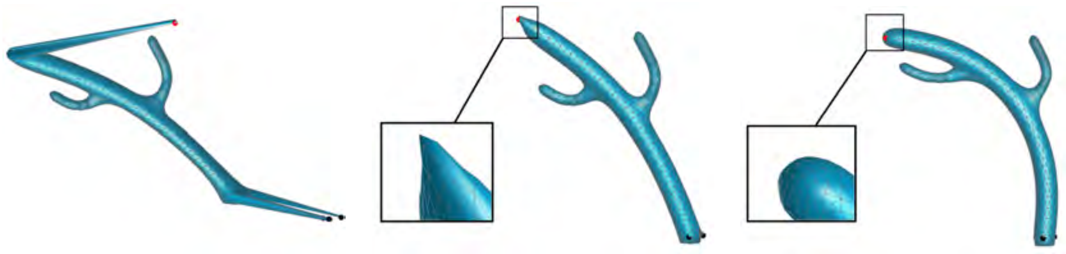
Figure 3.7: Deformation of the Cactus mesh by a rotation and translation of handles using the (left) Lie bodies framework with hard constraints for handles, (center) Lie bodies framework with soft constraints for handles with weight on handle constraints set to 1000, (right) proposed framework.

to Figure 3.7 (right).

Large deformation on standard datasets: The results of interactive shape deformations using Algorithm 1 on standard datasets are shown in Figure 3.8. These deformations are achieved with a small number of handles (typically less than 10), and no intermediate moving handle positions have been provided. As can be seen, the deformations obtained are intuitive. In the Armadillo case, we rotate the handle on the forehead by 180 deg which produces intuitive result with only 3 handles. Similarly for the Dragon mesh, with 4 handles (one on each foot) an intuitive result can be seen for a large deformation. The results of applying rotations of angle 3π and 2π are shown on the Bar mesh with only 2 handles (one each on top and bottom). Note that for applying a rotation of more than π deg, in place of giving the final handle position, a deformation vector for the handle is provided as discussed in the footnote of Section 3.4. Another example of a large deformation is shown in Figure 3.9, wherein one of the arms of the octopus has been rotated by approximately 145 deg about the arm centroid. Observe that the arm does not exhibit any folding.

Comparison with State-of-art: Next, we compare deformations produced by our algorithm with those produced using LRI [60], FAST [46], LS-ARAP [89] and SR-ARAP [57] algorithms. We use the planar cigar mesh as a test case. We select two triangles (6 vertex handles for the other algorithms) lying near opposite ends as handles. In addition to the results provided in Figure 3.6, we provide results of a one-step deformation by rotating the moving handle by an angle of 144 deg in Figure 3.10, (out of the plane containing cigar, same as in Figure 3.6). Again, FAST

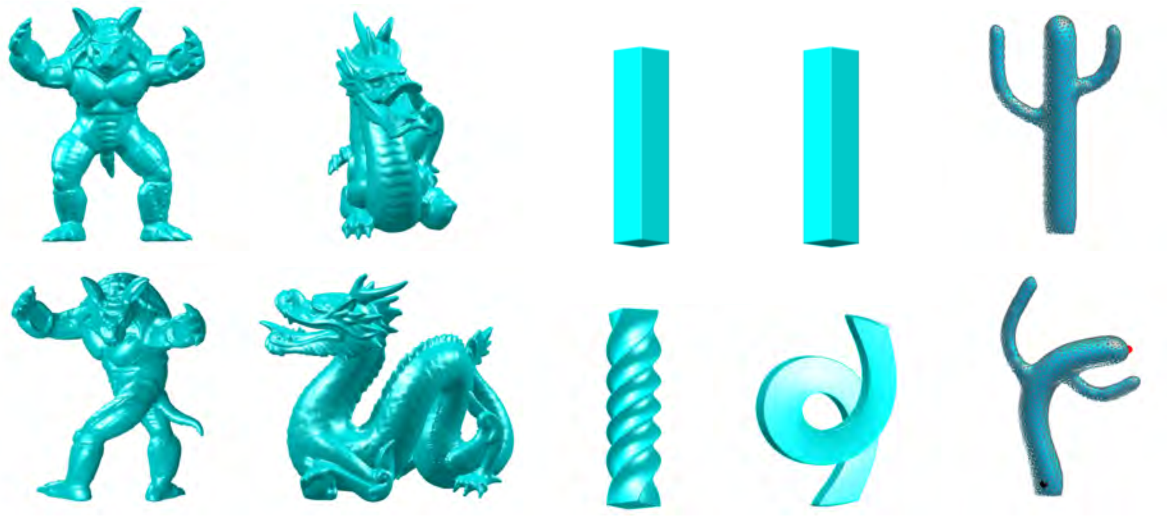

Figure 3.8: Deformation on Armadillo, Dragon, Bar & Cactus mesh. (Row 1) Original meshes, (Row 2) Deformed meshes. For Armadillo, 3 handles (one on each foot, one on the forehead) are used and deformation is achieved by rotating the handle on the forehead by 180 deg. We use 4 handles for the Dragon mesh to generate the large deformation. For the Bar mesh, results of applying rotations by 3π and 2π (col. 3 and 4) using only two handles (one on top and one on bottom) are shown, while the Cactus mesh is bent by 60 deg using 5 handles.

and SR-ARAP twist the cigar, and moreover, the result depends on the amount of deformation. The proposed algorithm yields intuitive results and does not vary with the amount of deformation. Instead of an out of the plane rotation, we next rotate the moving handle in the $x - y$ plane by different amounts (the handles are the same as used in the earlier case). Results are given in Figure 3.11. When rotated by an angle of 144 deg, the LRI produce a shrinking in the Cigar, while FAST, LS-ARAP SR-ARAP and the proposed algorithm produce intuitive results. When rotated by an angle of 180 deg, the LRI, FAST, and LS-ARAP produce undesirable effects in the Cigar, while even the SR-ARAP produces a counter-intuitive global rotation in the Cigar. While in some of these algorithms a better result can be expected by introducing more handles/pseudo-handles, our algorithm is able to achieve intuitive deformations using only two handles. In order to further compare our algorithm with SR-ARAP, we try to induce a twist deformation in the Cigar mesh, by rotating the moving handle by an angle of 180 deg about its own centroid. As can be observed in Figure 3.12, our algorithm produces a nice symmetrical twist, while with just two handles, SR-ARAP is not able to produce the

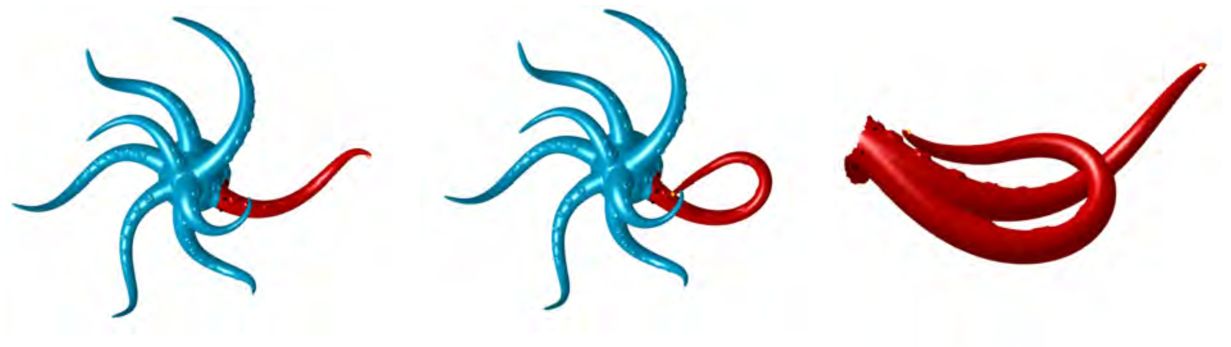

Figure 3.9: Large deformation on Octopus mesh. One of the arms (shown in red) of the octopus is rotated by approximately 145 deg about the arm's centroid. The arm does not exhibit any folding. Black denotes fixed handles, while yellow denotes moving handles.

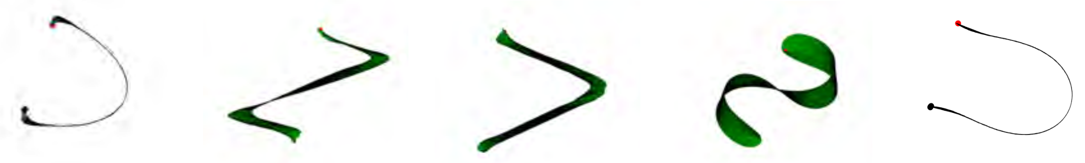

Figure 3.10: Cigar one-step deformation by rotating the moving handle out of the plane by an angle of 144 deg. (left to right) LRI, FAST, LS-ARAP, SR-ARAP, Proposed. While LRI produce intuitive results, FAST, LS-ARAP and SR-ARAP twists the cigar. The proposed algorithm yields intuitive results and does not vary with the amount of deformation. Results for FAST and LS-ARAP are computed with 80 auxiliary handles.

twist. Instead due to the desired smoothness in the rotations, it produces a discontinuity around the handle, leaving the rest of the mesh as it is. This situation can be alleviated by introducing extra handles, but the algorithm still struggles to produce a symmetrical twist.

Handle selection & translation insensitivity: Typically, interactive deformation algorithms rely on point based handles. Since the proposed algorithm uses face based handles, we analyze the effect of handles for translation and rotation deformations on the Spiky plane mesh. In Figure 3.13, different types of deformations are shown using different combination of transformations on handles. Here, the rotation of a handle (triangle) is with respect to its centroid. The handles shown in red and black are moving and fixed respectively. In cases where the deformation are performed with fixed and translated handles, some of the geo-

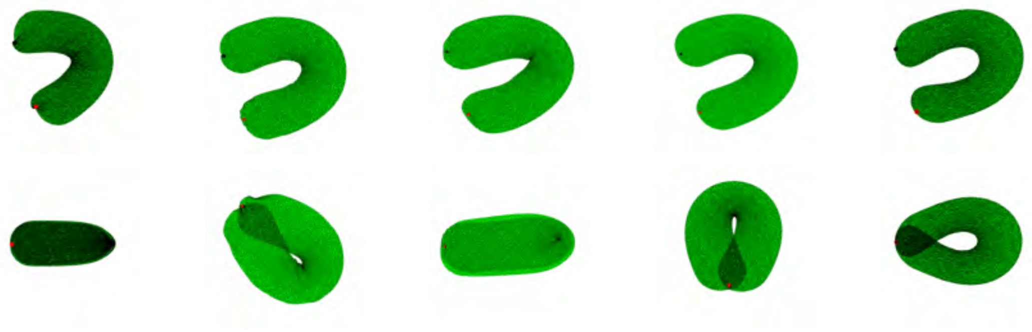

Figure 3.11: Deformation achieved by rotating moving handle by 145 deg (center row) and 180 deg (bottom row). (left to right) Deformation produced by LRI, FAST, LS-ARAP, SR-ARAP and proposed approach. When rotated by an angle of 180 deg, the LRI, FAST and LS-ARAP produce undesirable effects in the Cigar, while the SR-ARAP produces a counter-intuitive global rotation in the Cigar. Note that all deformations have been obtained in a single step, i.e., no intermediate handle positions have been used. Fixed handle marked in black, moving handle marked in red. Note that results included for FAST and LS-ARAP are chosen from a set of results with 40, 60 and 80 auxiliary handles.

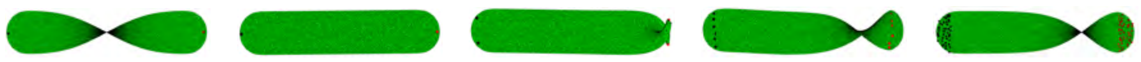

Figure 3.12: Twisting of Cigar about x-axis by 180 deg about the centroid. (left to right): Result using proposed method and results by SR-ARAP with different number of handles. Fixed handle marked in black, moving handle marked in red. The proposed method produces a nice symmetrical twist with just two handles, while with same handles SR-ARAP is not able to produce the twist. By introducing extra handles SR-ARAP is able to produce the twist but the algorithm still struggles to produce a symmetrical twist.

metric details are not deformed appropriately. This is visible in the results shown in Figure 3.13(e), where the algorithm fails to capture local rotations. We deal with this issue by introducing two additional handles as shown in 3.13(f). Since our framework uses the Lie group $SE(3)$, the translation and rotation components are treated independently. Thus a translation fails to induce any local rotations on the triangles, which leads to *translation insensitivity*.

Data driven deformations: A part of humanoid mesh sequence (hand motion sequence) is used to demonstrate the data driven interactive deformation suggested in Algorithm 2. Meshes from vertical and sideways hand motion sequences are used to learn the space of deformation vectors, as shown in Figure 3.15. The mean μ and the principal deformation vectors $E_1, \ldots, E_n$ are then com-

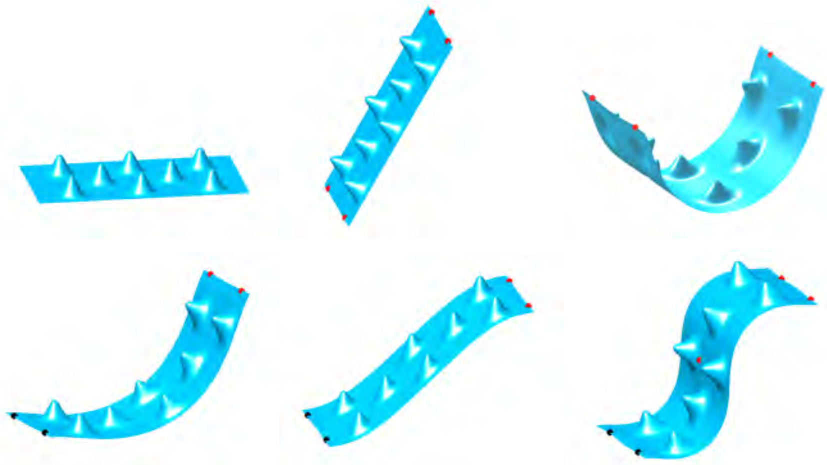

Figure 3.13: (Clockwise from top left.) (a) Original Mesh (b) Global rotation using both, handle translation and rotation, (c) Bending using handle rotations, (d) Lifting from a side using fixed handles and handle translations and rotations, (e) Lifting from a side using fixed handles and handle translations only, (f) Realistic lifting from a side using combination of fixed handles, handle rotations and translations, with 2 additional handles.

puted as discussed in Section 3.5. Once the handles are specified, Algorithm 1 is used to compute the deformation vector D. D is then used by Algorithm 2 to produce constrained deformations. Figure 3.15 shows the results for two training sets: (1) Vertical motion sequence, and (2) Vertical and sideways motion sequences. In both cases, the deformations correspond to the first principal deformation vectors. The user is given the flexibility to choose a weighted average of the deformation vector D and constrained deformation vector $\mu + \Pi(\tilde{D})$, to produce a deformed mesh as shown in Figure 3.16. The trade-off, as mentioned earlier is that the modeling constraints may be violated due to the PCA projection.

Effect of change of resolution: In the cylinder mesh, we alter the resolution in random contiguous areas and perform the same handle deformations. Figure 3.14 shows the deformations of the cylinder obtained with three variations in resolution: the original mesh, and both higher and lower resolution in chosen areas relative to the rest of the mesh. The handle deformations are the same in all three cases. It is evident from the example that the change of resolution does not qualitatively alter the deformation.

Computational performance: We provide the details of time taken by our algorithm for various results discussed above in Table 3.1. The total time taken

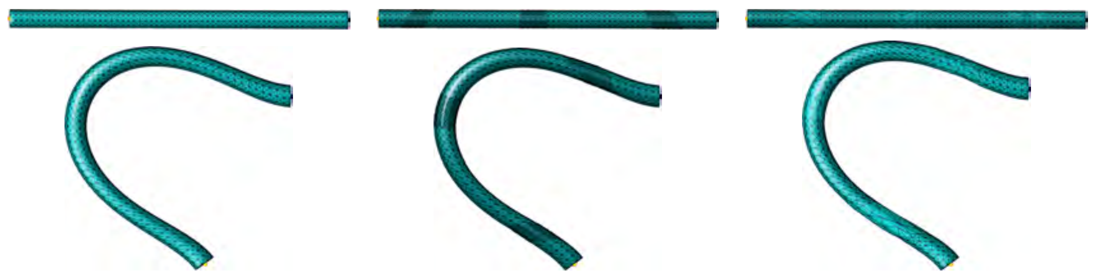
Figure 3.14: Effect of change in resolution. Row 1 shows original meshes, while Row 2 provides deformed meshes obtained via same handle deformations. The number of handles used is 2, shown in yellow (moving) and black (fixed). The left column demonstrates the deformation on the original cylinder mesh, while the center and right column demonstrates the deformation for the mesh with larger and smaller resolution in selected areas, respectively. The number of vertices and faces in these meshes from left to right is $(1212, 2420), (2022, 4036), (997, 1990)$.

is broken down into: (a) Time taken to convert a mesh into its Lie group representation, time taken for computing (b) Log maps and (c) Exponential maps on the Lie groups, (d) pre-processing for reconstruction which involves setting up a least squares system, and (e) Solving the least squares system. Note that the given total time includes time taken by other components not mentioned in the table, since it is negligible. We would like to emphasize the following: Note that computing the Lie group representation of the given mesh takes approximately 50% of the total time. Thus, once the Lie group representation has been computed (which can be done *offline*), the time taken for the actual deformation reduces drastically. Secondly, except for the final reconstruction, each triangle face can be treated independently in every computation (including computing the Lie group representation), our framework is a candidate for a massive parallelization. Our current implementation is serial in nature, with no explicit parallelization involved. Therefore, we expect our algorithm to yield real time deformations on high resolution meshes once the parallel nature of computations is exploited to its fullest.

3.7 Discussion and Conclusion

Translation insensitivity is an issue which we currently deal with by introducing additional handles. We are working towards an iterative algorithm to address

Table 3.1: Computational time for the proposed algorithm in seconds. Observe that about 50% time is spent in obtaining the Lie group representation, thus the time spent in mesh deformation is much less and this time increases approximately linearly with the number of faces in the mesh. Note that for the Octopus mesh, the resolution is given for only the deformed arm.

Mesh	Cactus	Cigar	Octopus	Spiky Plane	Armadillo	Dragon
Vertices	1856	3566	6508	24320	43243	50000
Faces	3708	6894	12924	47800	86482	100000
Lie Group Representation	0.0277	0.0466	0.1775	0.3477	0.6145	0.7466
Log Map	0.0060	0.0119	0.0365	0.0783	0.1498	0.1660
Exp Map	0.0112	0.0148	0.0320	0.0584	0.1070	0.1206
Reconstruction (Pre-processing)	0.0036	0.0066	0.0193	0.0456	0.0912	0.1100
Reconstruction (Linear system)	0.0011	0.0025	0.0095	0.0186	0.0594	0.0670
Total	0.0658	0.1065	0.3901	0.6780	1.2351	1.4673

this issue. We intend to implement the proposed approach in a more efficient manner by exploiting the obvious parallelism that exists by treating each triangle independently. With this improvement, the proposed algorithm should yield real time performance even for high resolution meshes. We are also working on incorporating better data-driven deformation schemes for our Lie group based mesh representation.

To conclude, we propose a Lie group representation for triangular meshes and use the same for interactive shape deformation. The proposed algorithm is able to achieve large deformations of comparable quality with state-of-art methods, often performing better, and with fewer handles. It is noteworthy that the algorithm is a one step process as compared to iterative state-of-art methods. Algorithm 1 allows for elastic stretching, and at the same time produces deformations that meet constraints imposed by the handle positions. The algorithm is invariant to choice of canonical triangle thus making it reliable. Also under the assumption that the user performs a rigid transformation and/or uniform scaling to obtain the deformed handle triangle, the algorithm is also invariant to vertex re-ordering (refer to Section 2.5). We also demonstrate the applicability of the proposed framework to data driven deformations.

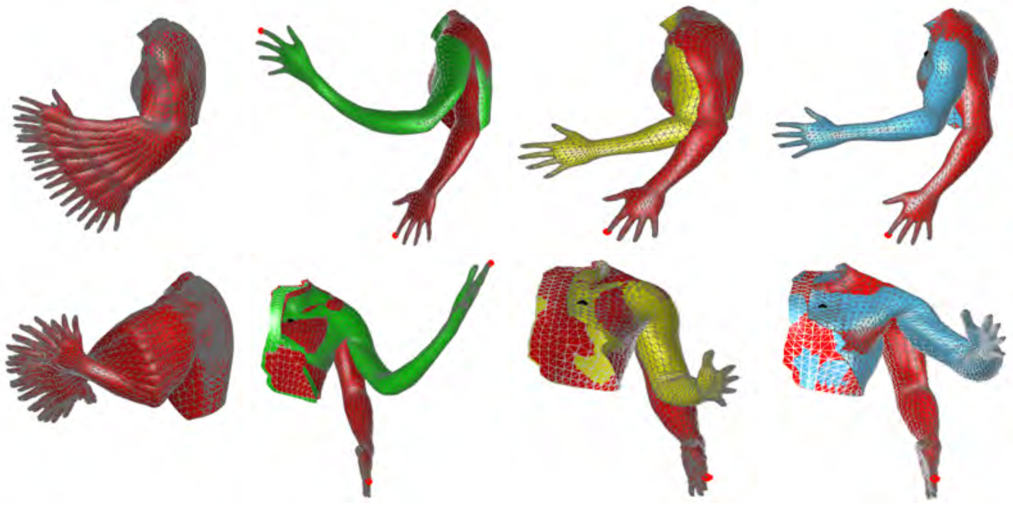

Figure 3.15: Example of Deformation Learning: (Column 1) The two deformation sequences used for learning the deformation space. (Column 2-4 Top and Bottom row) Mesh to be deformed is shown in red and deformed mesh produced by Algorithm 1 is shown in green. The deformed mesh corresponding to the projection on first principal direction generated by Algorithm 2 using only one deformation sequence (Column 1,Top row) is shown in yellow, while the deformed mesh corresponding to the projection on first principal direction using both deformation sequences is shown in blue. This example uses only two handles. The top row gives a side-view of the meshes, while the bottom row gives a front-view of the same results. Some sideways motion can be clearly seen in the blue mesh.

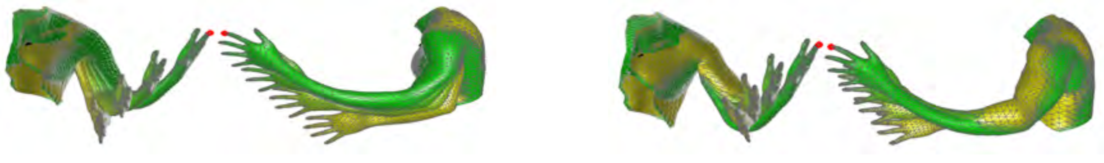

Figure 3.16: Trade off between handle constraint and training sequence subspace. The mesh in green is obtained using Algorithm 1 and the mesh in yellow is obtained using Algorithm 2. Using notations from Figure 3.5, the intermediate meshes are produced using a convex combination of the deformation vector D obtained using Algorithm 1 and the projected deformation vector $\mu + \tilde{D}_{E_1}$ computed using Algorithm 2. The meshes on the left are obtained using the first training sequence, and the meshes on the right are obtained using both training sequences given in Figure 3.15.

CHAPTER 4
Deformation Transfer

In computer animation, different objects may have to undergo similar deformations. Deformation transfer tries to mimic the deformation given for a source object, for a given target. In this chapter, we propose a deformation transfer approach based on the Lie bodies representation of triangular meshes. The approach does not assume that the target and source meshes are of the same resolution or are registered. Due to the Lie bodies representation, the non-linearity in the space of transformations is accurately modeled which enables the framework to handle large deformations. Furthermore, under certain assumptions stated in this chapter, the approach can also be used for the task of pose transfer.

4.1 Introduction

Computer animation of 3D models can be a lengthy and complicated task for an artist, if not supported by algorithms that either partially or fully automate this task. One particular framework to assist the animator, called deformation transfer, generates different shapes of a given target model by mimicking the deformation between given shapes of a source model. The need for deformation transfer arises naturally in many domains, for example, character animation, facial expressions modeling in computer graphics and anatomy transfer in medical applications [5, 82]. Typically, for deformation transfer, deformation between source example meshes are transferred to the target reference mesh. In a similar scenario, known as *pose transfer*, deformation between source and target reference meshes are used to generate a example meshes for target shape using example source meshes. Usually, algorithms addressing deformation transfer are not able to transfer poses, and vice-versa.

The proposed method uses the Lie bodies representation for triangular meshes and is able to handle larger deformations for deformation transfer compared to some of the state-of-the-art methods in the literature. This is mainly due to the Lie group/ algebra modeling of the mesh and deformation. The non-linearity in the space of local transformations, in order to deform meshes, is accurately handled via Lie groups and associated tools. In some cases where the source and target meshes do not differ too much, the approach can be used for pose transfer also. An example of a deformed pose constructed using deformation transfer and pose transfer from the given meshes in shown in Figure 4.1.

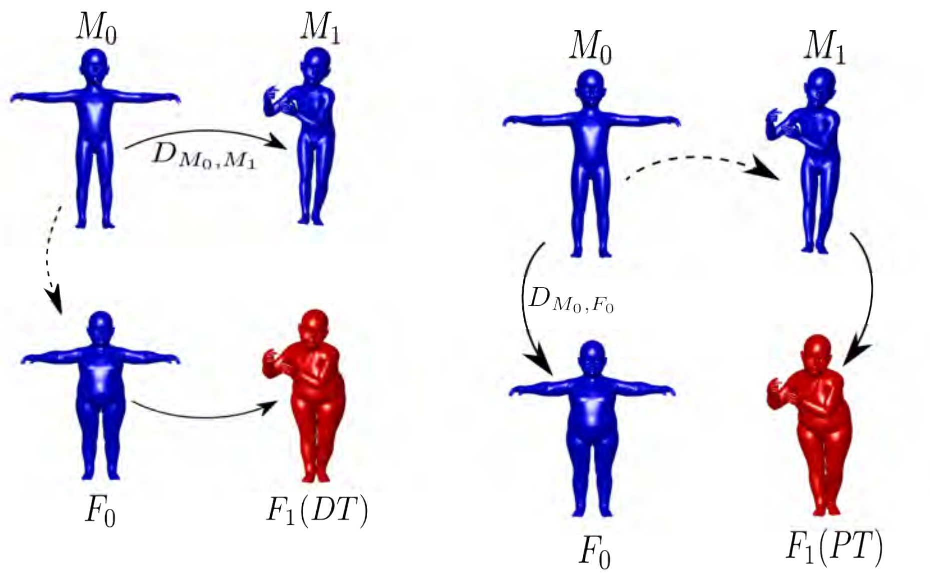

Figure 4.1: Example of (left) deformation transfer (DT) and (right) pose transfer (PT). M_0, M_1 represent given poses from one class, F_0 is the reference pose for the second class, while F_1 represents the mesh reconstructed using deformation transfer and pose transfer.

In the next section, we provide a brief review of the existing techniques in the domain of deformation transfer. Then we discuss the proposed approach for deformation transfer. In Section 4.4, we demonstrate the applicability of the proposed approach along with a comparison with some existing techniques. Finally, we conclude with limitations and future direction for the proposed work in Section 4.5.

4.2 Related Work

Deformation transfer methods can be classified based on the shape representation and the mathematical framework for representing deformation. Broadly, deformation transfer can be grouped into surface based and space based approaches [28]. Typically, triangular meshes are used in surface based approaches [82], while cage based control structure[26], and skeletons [24] are some popular examples of space based approaches. Examples of deformation models used are As-Rigid-As-Possible (ARAP) deformations [63], affine transformations on triangular faces [82], scaling and rotations of markers propagated via blending [99].

Sumner and Popović [82] describe the deformation between the source meshes via affine transformations. A set of markers are picked and deformation for these markers are transferred to the target mesh subject to a smoothness constraint. Zhou *et al.*[102] adapt this method for multi-component objects. In Zayer *et al.*[99], a dense correspondence between source and target mesh is obtained by using a sparse set of markers on source and target. Laplacian weights corresponding to these markers are computed and matched to compute the correspondence. Deformation between the source meshes on markers are decomposed into scaling and rotations, which are then transferred on the target mesh using the dense correspondence, followed by a reconstruction process involving a discrete Poisson equation. Deformation transfer using the ARAP model and rigidity control is proposed in [63]. Lévy and Bruno [58] use the spectral representation of a mesh, wherein low frequency components of the deformed source model and high frequency components of the target model are combined to obtain a deformed target model. One needs to compute common spectral bases[55, 97] for this purpose. A semantic deformation transfer scheme is proposed by Baran et al. in [13], wherein shape spaces for source and target models is built from a given set of example shapes. Deformation transfer is achieved by finding, for any new source mesh represented in source shape space, a corresponding point in the target shape space. The efficiency of the framework depends on the number of example poses, and how good the reconstruction process from shape space to the 3D model is. Shabayek *et al.*[75] use the Lie bodies [34] representation of triangular meshes, and transfer only the rotation (orientation) component of the deformation between the two source meshes to the target mesh. It also assumes a face-to-face bijective correspondence between the source and target meshes. In all of these approaches the correctness and denseness of the available correspondence between the source and target mesh is an important factor contributing towards the quality of results. Typically, it is either assumed that (dense) correspondence is available, or, sparse marker correspondences are used to obtain denser correspondences. For a survey on shape correspondence, refer [87].

In space based approaches, the deformation is modeled in a low dimension space defined by a control structure such as a cage[26, 15] or a skeleton[8]. Space based approaches dominate over surface based approaches in terms of simplicity of the modeling process, while the surface based approaches allow direct selection of markers making it easier to work with compared to cages and skeletons. Depending on the resolution of the cage/skeleton, space based deformation approaches may fail to capture and transfer fine-scale details. Moreover, such approaches will typically fail to transfer poses, especially in the case where the cage/skeleton are similar for both source and target models.

We propose a deformation transfer approach based on the Lie bodies mesh representation [34]. This enables the framework to handle large deformations in comparison with other approaches. We also suggest a mesh correspondence method based on the Bounded Biharmonic weights [47] to be used along with the deformation transfer procedure, which may be substituted by other methods of obtaining mesh correspondence.

4.3 Proposed Method

To begin with, we will assume that the source and target triangular meshes have the same number of vertices and are registered, and discuss a way to compute dense correspondence in Section 4.3.2.

4.3.1 Deformation transfer

Let P_0, P_1 and Q_0 denote the two example source meshes and the target mesh, each with n faces, respectively. We use the mesh P_0 as the reference mesh for both source and target meshes, in order to compute the representation in the Lie group $\mathcal{M}^n$ (refer Section 3.4). Again M_{P_i} and M_{Q_i}, $i = 0, 1$ are used to denote the Lie group representation.

The deformation between the source meshes is captured via the log map $D_{M_{P_0},M_{P_1}}$ on the $\mathcal{M}^n$ manifold. The tangent vector $D_{M_{P_0},M_{P_1}}$ is then used to compute the geodesic at identity, which is then translated to M_{Q_0} to get the Lie group representation of Q_1 i.e. M_{Q_1}.

The actual target mesh Q_1 is generated from its Lie bodies representation using a least-square approach, as mentioned earlier. This process is demonstrated in Figure 4.3 and summarized in Algorithm 3.

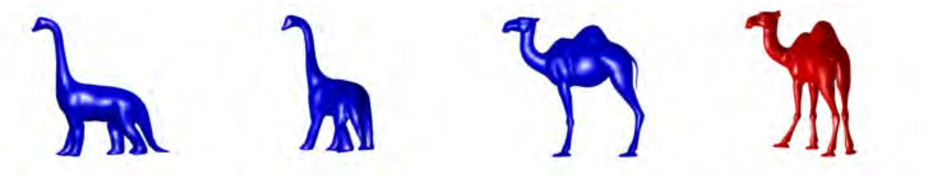

Figure 4.2: Pose transfer for similar shapes. (Left to Right) Reference pose and example pose of Dino model, Reference Camel pose and Camel pose generated using Pose transfer.

One may also compute the deformation vector $D_{M_{P_0},M_{Q_0}} = \log_{M_{P_0}}(M_{Q_1})$ that captures the difference in the geometry between the source and target meshes and use the deformation vector appropriately so as to provide a deformation of M_{P_1} into M_{Q_1}. This *pose transfer* approach can be used to produce the desired target pose M_{Q_1} as $M_{Q_1} = exp_{M_{P_1}}(\tilde{D}_{M_{P_0},M_{Q_0}})$. In practice, this approach produces the desired pose, as far as the source and target meshes are in a bijective and meaningful(semantic) correspondence, and the geometry is not too different, see Figure 4.2.

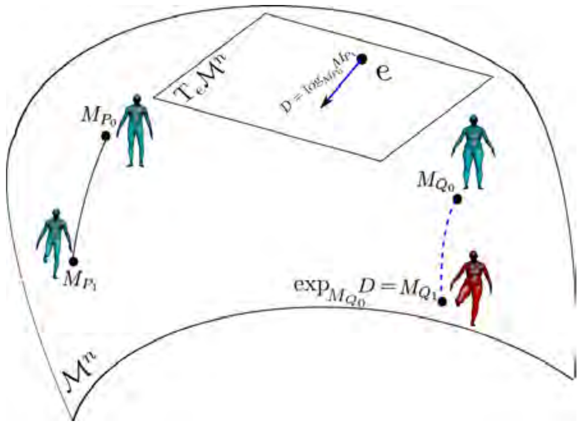

Figure 4.3: Deformation transfer. Compute the deformation vector D between M_{P_0} and M_{P_1}. Use the deformation vector to generate M_{Q_1} corresponding to M_{Q_0}.

As far as there is a bijective correspondence between the source and target meshes which is semantically meaningful, this deformation transfer approach works well. In the next subsection, we describe a method for obtaining a correspondence map (not necessarily bijective) between the source and target mesh, using which our deformation transfer can produce the unknown target mesh.

4.3.2 Correspondence Map Using Bounded-Biharmonic Weights

In order to estimate a dense correspondence between the source and target meshes P and Q (we drop the subscript indicating reference and desired pose in this subsection to simplify notations), we assume a sparse set of face correspondences provided by the user. On the dual mesh, treating the vertices as markers, we compute Bounded-Biharmonic weights proposed in [47] for each marker. Given a set of k markers on both the dual source and target meshes with n and m vertices respectively, weight vectors are computed at each vertex and are denoted as $w_i^P \in \mathbb{R}^k, i = 1, \ldots, n$ and $w_i^Q \in \mathbb{R}^k, i = 1, \ldots, m$. A matching is done in the weight space in order to obtain a correspondence map. For every target face Q^i, let C_i be the index set of corresponding faces from the source mesh to be computed, and let M_{P,C_i} denote the Lie group representations of the corresponding faces. This set is obtained using the rule : $j \in C_i$ iff $||w_j^P - w_i^Q||_2 \leq \epsilon_i$, where ϵ_i is a threshold given by $\epsilon_i = min_{j=1,\ldots,n}\{||w_j^P - w_i^Q||_2\}$. This ensures that none of the sets $C_i, i = 1, \ldots, m$ is an empty set, and thus each face in the target mesh has at least

Algorithm 3 Deformation Transfer Algorithm

```
Require: Source reference and deformed meshes $P_0$, $P_1$, and target reference mesh $Q_0$ with $n$ faces.
 1: Compute the Lie group representation of $P_0$, $P_1$ and $Q_0$, i.e. $M_{P_0}$, $M_{P_i}$, and $M_{Q_0}$.
 2: if Correspondence is available then
 3:     Compute deformation as:
        $D_{M_{P_0},M_{P_1}} = \log_{M_{P_0}}(M_{P_1})$.
 4: else
 5:     Compute Correspondence. {Algorithm 4}
 6:     Compute target deformation vector: $D_{M_{P_0},M_{P_1}} = (\tilde{d}_1, ..., \tilde{d}_n)$ {Section 4.3.2}
 7: end if
 8: $M_{Q_1} = \exp_{M_{Q_0}}(D_{M_{P_0},M_{P_1}})$. {Compute Lie group element representation of $Q_1$.}
    $exp_{M_{Q_0}}(D) = (\exp_{M_{Q_0,1}}(\tilde{d}_1), \ldots, \exp_{M_{Q_0,n}}(\tilde{d}_n))$
 9: Reconstruct the target mesh $Q_1$. {Using Least square reconstruction.}
10: return $Q_1$
```

one corresponding face in the source mesh. Refer Algorithm 4. Once the correspondence map is available, the previous deformation transfer algorithm can be used.

Let $M_{Q_0,i}$ denote the Lie group representation of the i^{th} face of the reference target mesh, with a set of corresponding faces on the reference source mesh P_0, whose Lie group representation is denoted by M_{P_0,C_i}. Let $d_j = \log_{M_{P_0,j}}(M_{P_1,j}), j \in C_i$ denote the deformation (tangent) vector on the Lie group element $M_{P_0,j}$ for the j^{th} face of the reference source mesh. These deformation vectors can be translated to the point $M_{Q_0,i}$ giving us $\tilde{d}_j = l_{M_{Q_0,i}}(d_j)$. In case the set C_i is singleton, the Lie group element of face i on deformed target mesh is obtained as $M_{Q_1,i} = \exp_{M_{Q_0,i}}(\tilde{d}_j)$, while if the set contains more than one element, an average of the deformation vectors in computed: $a_i = \frac{1}{|C_i|}\sum_{j \in C_i} \tilde{d}_j$, which then gives $M_{Q_1,i} = \exp_{M_{Q_0,i}}(a_i)$. The least square reconstruction process gives the deformed target mesh Q_1. Note that any other registration process can be used in place of the one suggested here, and the quality of the deformation transfer is expected to improve with better registration of meshes.

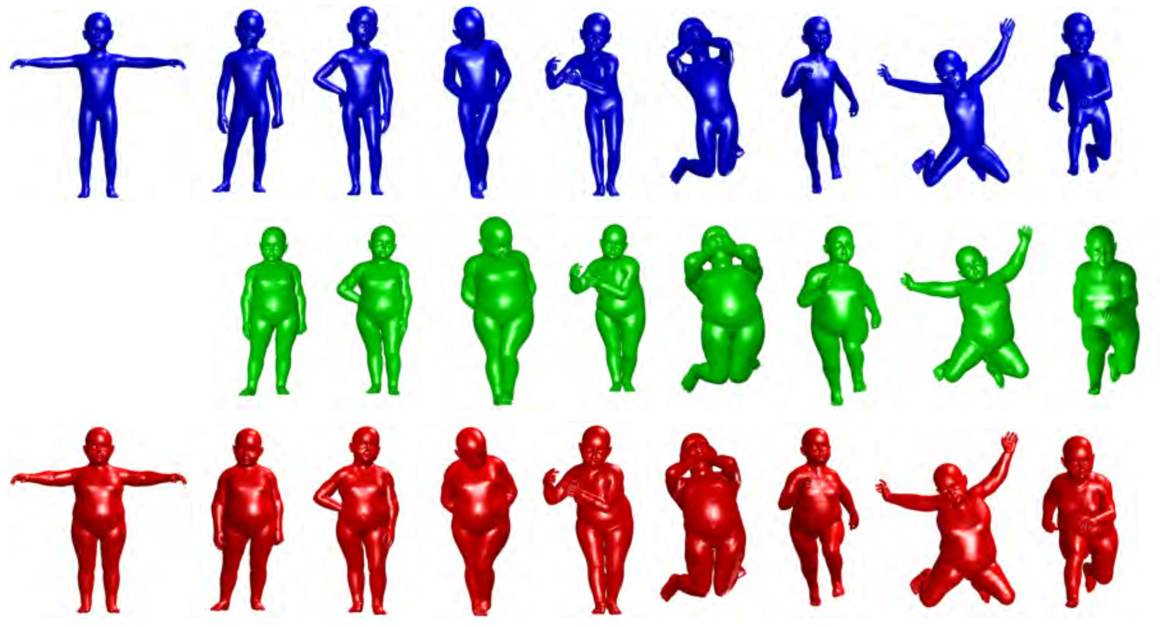

Figure 4.4: Deformation transfer on kids data. Correspondence is available between the two models. Top row: Reference and example poses of source model. Middle row: Ground truth for the target class. Bottom row: Results using proposed deformation transfer method.

4.4 Results

In this section, we validate our approach by performing deformation transfer on several datasets of triangular meshes, some with ground truth registration data available, while others without ground truth registration information and possibly of different resolutions. We then compare results of our approach with results from two methods from the class of surface deformation methods: Sumner & Popovic [82], and Shabayek *et al.*[75]. We also demonstrate the possibility of obtaining the deformed target mesh as a result of pose transfer.

1. *Registered meshes:* We demonstrate our algorithm on the Kids [70] and the FAUST [16] dataset. Both these datasets contain registered meshes with a resolution of approximately 120k and 14k faces, respectively. These datasets also contain the target meshes in the desired poses, which can be treated as ground truth. Some examples of deformation transfer obtained using our algorithm for both these datasets and corresponding ground truth meshes

Algorithm 4 Compute Dense Correspondence

```
Require: Source and target reference meshes $P$, $Q$, with $n$ and $m$ faces respectively.
1: Compute dual meshes of $P$, $Q$. {Dual source and target meshes contain $n$ and $m$ vertices. }
2: Select marker pairs $(H_i^P, H_i^Q)$, $i = 1, \dots, k$, on the dual source and target meshes.
3: $w^P = BBW(P, H^P)$,$w_i^P \in \mathbb{R}^k, i = 1, \dots, n$
   $w^Q = BBW(Q, H^Q)$, $w_i^Q \in \mathbb{R}^k, i = 1, \dots, m$.
   {Compute bounded biharmonic weights (BBW) corresponding to the selected markers.}
4: for $i \in (1, \dots, m)$ do
5:     $C_i = j$, iff $||w_j^P - w_i^Q||_2 = \epsilon_i$,
       where $\epsilon_i = min_{j=1,\dots,n}\{||w_j^P - w_i^Q||_2\}$.
6: end for
7: return $C$ {Set of correspondences for each face in the target mesh to source mesh.}
```

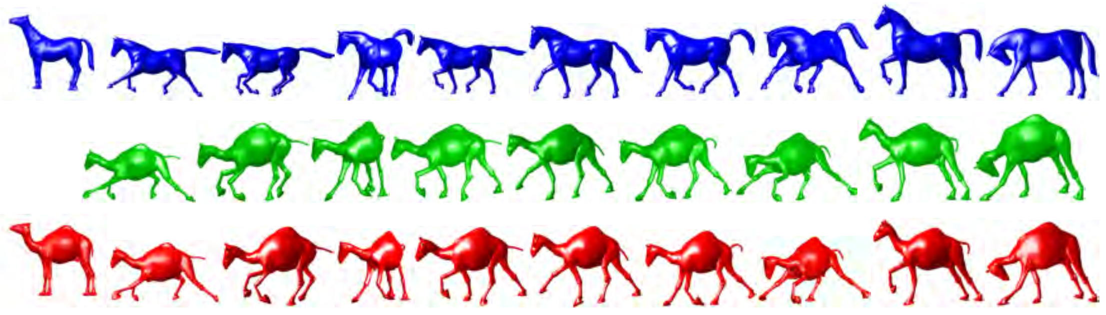
Figure 4.5: Marker based deformation transfer. Top row: Reference and example poses of source model. Middle row: Results using state-of-the-art[82]. Bottom row: Results using proposed deformation transfer method.

are shown in Figures 4.4 (Figure 2 in the supplementary material). As can be seen from these figures, our approach is able to produce visually reliable results even for large deformations. A comparison with the results from [82] and [75] on the FAUST dataset is given in Figure 4.7. For example, note that Sumner's approach misses out on body details (for example chest region) (Column 3). By selectively transferring only the rotation component, Shabayek's approach misses out on an important details of the deformation as shown in Figure 4.8. Also, as can be seen in the figure, our approach is able to mimic the deformation of the source examples more accurately.

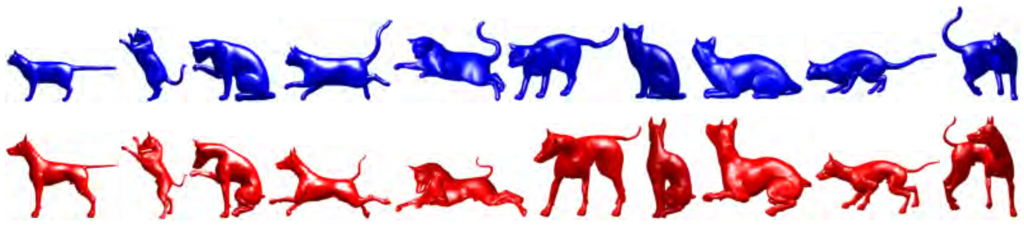

Figure 4.6: Deformation transfer on TOSCA data. Correspondence is not available. Top row: Reference and example poses of source model. Bottom row: Results using proposed deformation transfer method.

Videos of deformation transfer for a set of action sequences are provided at `https://github.com/sumukhbansal/LieBodiesBasedDeformationTransfer`, which clearly demonstrate the advantage of the proposed method.

2. *Meshes with different resolution:* In case the source and target meshes have a different resolution, we first obtain a correspondence map followed by deformation transfer, as described in Section 4.3.2 and 4.3.1. Note that in this case Shabayek's approach is not applicable as they assume that the meshes have the same resolution and are in correspondence. Figure 4.5 gives the result of our approach and the approach in [82] for transferring deformations of the *horse* mesh onto the *camel* mesh. The quality of results using both approaches is similar. Note that in our approach, we compute the correspondence between the two meshes only once, and use the same correspondence for generating all poses shown in the figure. We also provide results of our approach for transferring deformations of the *cat* mesh onto the *dog* mesh Figure 4.6. The meshes used in this experiment are taken from the TOSCA dataset[21].

3. *Pose transfer:* As desribed in Section 4.3.1, we can use the deformation D_{M_0,F_0} between the reference source and target poses to generate the desired deformed target pose, in case the meshes have the same resolution and a meaningful correspondence available. In Figure 4.7, the proposed approach is compared with Sumner's and Shabayek's approaches. While the arm of the model undergoes thinning using Sumner's approach, Shabayek's approaches fails to transfer the body-mass.

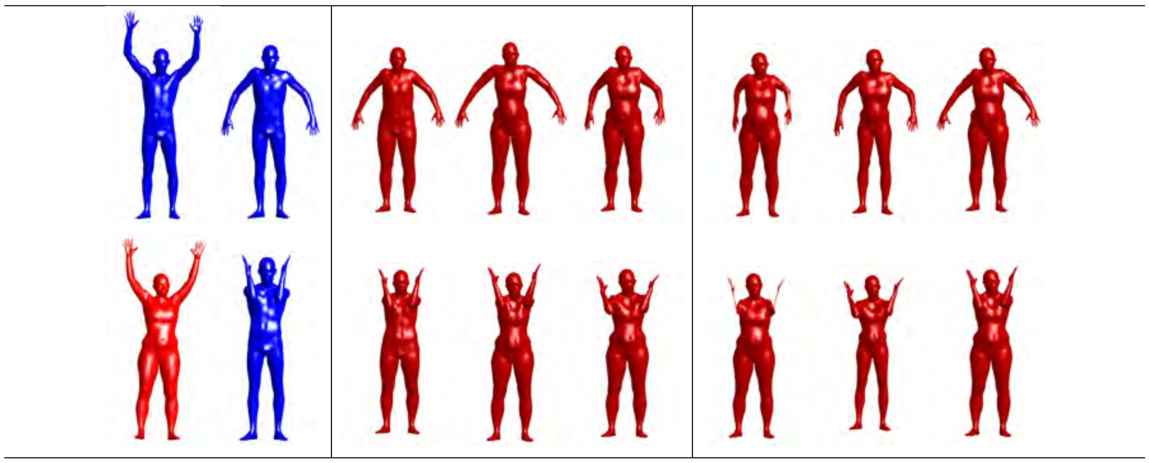

Figure 4.7: Comparison of deformation transfer (column 3-5) and pose transfer (column 6-8) using different methods. Correspondence is available. (Left to right column-wise) Source and Target reference poses(Column 1), Source examples, DT using Sumner's method, DT using Shabayek's method, DT using proposed method, PT using Sumner's method, PT using Shabayek's method, PT using proposed method (For larger images, refer supplementary material).

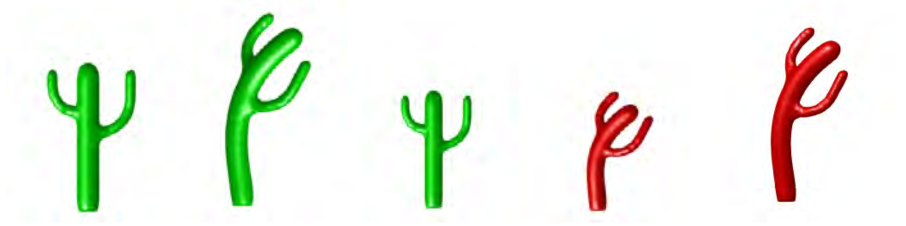

Figure 4.8: Limitation of Shabayek's method. (Left to Right) Reference and example mesh for source cactus model (Note that the deformation consists of both bending and stretching), Target cactus model (a scaled version of the source cactus model), Result of deformation transfer using Shabayek's method and Proposed method. As can be seen that Shabayek's fails to transfer the affine component of the deformation.

4.5 Discussion and Conclusion

We proposed a framework which is able to handle large deformations and work for both deformation and pose transfer. We propose to use a weight space to find a dense correspondence in case where its not available already. The availability of the correspondence or a successful registration algorithm is one of the keys to the success of the deformation transfer algorithm. Overall, the proposed method is able to achieve better or comparable results as compared to some of the other popular approaches. A serial implementation of the proposed method on Libigl framework [49], running on an Intel5 8GB RAM, 1.6 GHz system, for a mesh of $100k$ faces, took 1.5 sec, which involves computing log map (0.16 sec), exponential map (0.12 sec), pose reconstruction (.18 sec) and Lie body mesh representations (1 sec).

CHAPTER 5

Interpolation and Morphing

Interpolation and morphing algorithms aim to support an animator by automating animation in two scenarios. While morphing algorithms generate intermediate objects given two (or more) objects belonging to different classes, interpolation generates intermediate objects given two deformations of the same object. We propose a framework for morphing and interpolation based on the Lie bodies representation of triangular meshes. Without any physics based constraints on allowable deformations, the Lie group of transformations involved are able to handle both, morphing as well as interpolation. However, it does fails in case of large deformations. In such cases, we propose to segment the mesh and use the Lie bodies framework on individual components. Our segmentation scheme is based on detecting parts of meshes undergoing large deformations. The Lie bodies framework is thus able to handle large deformations, is able to produce any intermediate interpolation result directly, and is efficient due to the independent treatment of triangles in the mesh. We provide several interpolation and morphing results in support of our framework.

5.1 Introduction

Morphing and interpolation deals with algorithms that automate the process of producing visually appealing intermediate shapes between a pair of source and target shapes, while preserving some intrinsic properties of the shapes.

Morphing refers to the task of gradually deforming a source shape into a target shape, both belonging to different classes of object. On the other hand, Interpola-

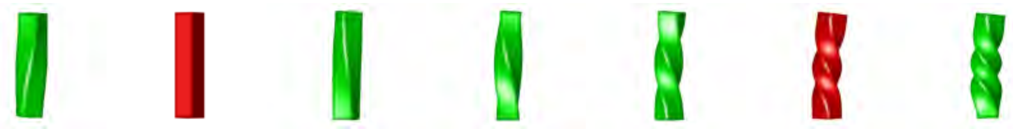

Figure 5.1: Twisting of a bar - An example of Interpolation and Extrapolation. From left to right, the meshes correspond to time $t = -.25, 0, .25, .5, .75, 1, 1.25$, where mesh at $t = 0$ corresponds to the initial mesh and mesh at $t = 1$ corresponds to the final mesh. Note that without the proposed segmentation based interpolation approach, this particular large deformation interpolation example will not produce intuitive results. Also, note that our approach is able to extrapolate the deformation.

tion refers to the process of generating intermediate shapes for source and target shapes belonging to the same class. Morphing and Interpolation can be thought of as computing geodesics between given points in an appropriate shape space. The difference arises in the choice of the given shapes. From this perspective, interpolation and morphing should be addressed via a common framework.

In this chapter, a unified non data-driven framework for morphing and interpolation, which is able to handle large deformations and is efficient in term of computational time is proposed. The proposed approach is applicable to triangular meshes and is based on the Lie bodies representation [34]. Since the meshes are represented as elements of a Lie group, the framework is able to work with the non-linear space of transformations with closed-form computational tools. For mesh interpolation, the framework is able to handle large deformations based on a segmentation of the mesh into components with small and large deformations. An example of a large deformation interpolation and extrapolation result of our framework is shown in Figure 5.1.

This chapter is organized as follows. We begin with a summary of related work in Section 5.2. The proposed common framework for morphing and interpolation and related algorithms is described in Section 5.3. Results of various experiments performed are provided in Section 5.4, followed by some concluding remarks in Section 5.5.

5.2 Related work

Morphing and Interpolation can be seen as special cases of interpolation between points on appropriately defined shape spaces. A correspondence map between the source and target meshes is needed in order to generate an intermediate object. The quality of morphing and interpolation depends on the quality of available/computed correspondence map (usually bijective). Based on the correspondence map, a non-rigid deformation framework between the source and target meshes is defined.

Many approaches are based using local rotations [4] or affine transformations [84] to model deformation and then interpolating these transformations. Linear/ affine transformations are usually decomposed into, translation, rotational, and scale/shear components, where translation, scale/ shear components are interpolated linearly, while the rotational component is interpolated in the intrinsic space either via the exponential map on rotation matrices or via quaternions. In approaches based on interpolating dihedral angles and edge lengths [35] of the mesh, interpolated mesh has to be reconstructed which is a computationally expensive process. In some cases, patch based interpolation is used, where the mesh is clustered into patches under going similar deformations. Each patch is interpolated independently and the final mesh is reconstructed via stitching the interpolated patches together [85].

Approaches based on energy minimization framework [54, 9, 101, 41], involve solving of differential equations at a very fine time step and are also computationally expensive. An approach for computing discrete-time geodesics in discrete-shell shape space is proposed in [41], while Brandt *et al.*[20] generalize this for shape spaces. The geodesic is iteratively computed via a curve smoothing/fairing method and thus is not real-time. A multi-resolution approach has been used to reduce the computational cost.

The authors in [56] propose a framework that carries out registration, statistical analysis and deformation tasks based on the spherical wavelet-based multi-resolution representation of Square-root normal fields (SRNF). Although interpolation in this representation is trivial (since it is a vector space), the reconstruction of the mesh from the representation is still computationally expensive even for low resolution meshes. While we acknowledge that computing correspondence map between models is a challenging problem, in this chapter, we assume that the correspondence map between the models is available, or can be computed using various registration algorithms, for example [96, 45]. For an exhaustive survey on various existing approaches for shape correspondence refer [87].

We propose a non data-driven framework for mesh interpolation and morphing for triangular meshes based on the Lie bodies [34] representation of meshes. The representation scheme is summarized in Section 3.3.1. We propose a segmentation based interpolation that is able to handle larger deformations than traditional non data-driven approaches and is amenable to real time performance.

Algorithm 5 LieBodyInterpolation/Morphing

```
Require: Source and target meshes $P_0$, $P_1$ with $n$ faces. {Source mesh is treated as reference mesh}
1: Compute Lie bodies representation of meshes $P_0, P_1$ i.e. $M_{P_0}, M_{P_1}$.
2: $D_{M_{P_0},M_{P_1}} = Log_{M_{P_0}} M_{P_1}$. {Tangent vector representing deformation.}
3: for $t \in [0,1]$ do
4:     $c(t) = Exp_{M_{P_0}}(tD_{M_{P_0},M_{P_1}})$ {Lie bodies representation of intermediate mesh.}
5:     $P_t = MeshReconstruction(c(t))$.
6: end for
7: return $P_t$
```

5.3 Proposed Approach

We begin by describing our framework for mesh interpolation and morphing under the assumptions that source and target shapes exhibit a small deformation. We describe a segmentation-based approach that caters to the case of large deformation in the following section.

5.3.1 Framework for Morphing and Interpolation

Given a pair of meshes P_0 and P_1, with n faces and same face connectivity, to be either interpolated or morphed, Lie bodies representations of both meshes are computed. Interpolation or Morphing between P_0 and P_1 reduces to computing geodesics between the Lie group elements M_{P_0} and M_{P_1}, which in turn are obtained by concatenating geodesics between $M_{P_0,i}$ and $M_{P_1,i}, i = 1, \ldots, n$.

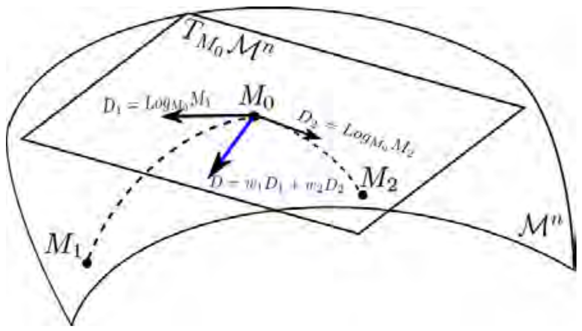

Figure 5.2: Multi component morphing.

The direction to move from M_{P_0} to M_{P_1} along the geodesic in the Lie group $\mathcal{M}^n$ is given by the Log map between M_{P_0} and M_{P_1}: $D_{M_{P_0}, M_{P_1}} = Log_{M_{P_0}}(M_{P_1})$. An intermediate mesh, for interpolation or morphing, is then obtained via sampling the geodesic given by $c(t) = Exp_{M_{P_0}}(tD_{M_{P_0}, M_{P_1}}),\ t \in [0, 1]$, followed by the least square mesh reconstruction process as mentioned earlier.

One can use this algorithm for morphing between multiple meshes also. Let P_0, P_1, and P_2 be the given meshes. In order to morph the mesh P_0 into a combination of meshes P_1 and P_2 we compute tangent vectors D_1 and D_2 corresponding to geodesic joining M_{P_0} to M_{P_1} and M_{P_2} respectively, using Log map. For intermediate meshes, a weighted average of the tangent vectors D_1, D_2 is taken and points on geodesic corresponding to the new tangent vector are computed using the Exp map. This procedure is demonstrated in Figure 5.2.

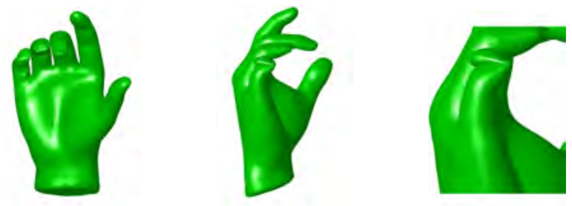

Figure 5.3: Incorrect direction of bending of the little finger due to large deformation between source and target hand meshes. The source and target mesh are the same as used in Figure 5.7. Front-view(left), side-view(center), and close-up of the little finger, of the interpolation result at time $t = 0.5$.

Algorithm 6 Shape Interpolation For Large Deformation

```
Require: Source and target meshes $P_0$, $P_1$ with $n$ faces. {Source mesh is treated as
    reference mesh.}
 1: Compute the face normals $N_0, N_1$ of meshes $P_0, P_1$.
 2: for $i \in \{1, \ldots, n\}$ do
 3:   if $dot(\ N_0(i), N_1(i)\ ) < 0$ then
 4:     $i \cup N_{1-ring}(i) \rightarrow$ LargeDefTriangles
 5:   end if
 6: end for
 7: BaseMesh = $\{1, \ldots, n\} \setminus$ LargeDefTriangles
 8: $Y_{\text{BaseMesh}} = \text{LieBodiesInterpolation}(P_0, P_1, \text{BaseMesh})$ {Algorithm 5.}
 9: for $j \in$ ConnectedComponent(LargeDefTriangles) do
10:   $Y_j = \text{LieBodiesInterpolation}(P_0, P_1, j)$ {Align and compute the interpolated
      mesh for $j^{th}$ component using Algorithm 5.}
11:   $Y_{\text{BaseMesh}} = \text{Merge}(Y_{\text{BaseMesh}}, Y_j)$ {Find a suitable rigid transformation
      for merging component $Y_j$ to base mesh $Y_{\text{BaseMesh}}$.}
12: end for
13: return $Y_{\text{BaseMesh}}$
```

5.3.2 Large Deformation Interpolation

The Lie bodies framework [34] was proposed for statistical analysis of *small* shape variations. The Log map on manifolds is a key component in this analysis. It is well known that the Log map on some manifolds is only a local diffeomorphism [37]. The 3D rotation Lie group $SO(3)$ which is a key component in the Lie bodies representation is a prime example where such a scenario occurs. Specifically, the Log map between two points in $SO(3)$ will always give a tangent vector corresponding to the shortest geodesic between them. In terms of the mesh triangles, if a triangle is rotated by more than 180 deg, the interpolation result will yield intermediate triangles rotated in the opposite direction by a total angle less than 180 deg. For an example of this phenomenon, refer Figure 5.3.

To address this issue, we segment the meshes into different components based on the amount of deformation in the faces from source to the target mesh. The segmentation process is based on locating triangles with large deformation. We say that a triangle is undergoing a large deformation from the initial to the final mesh if the unit normal (outward) of the two triangles have an angle difference of more than 90 deg. Connected components (based on adjacency relation) are formed on this set of triangles, and each component is treated individually. In practice, each triangle with large deformation is also extended to all triangles in its 1-ring to deal with very small connected components if any. The rest of the mesh containing triangles with small deformations is referred to as the *Base mesh* and is also segmented into connected components.

Our process of segmenting the mesh into components is different from the one used for deriving patch-based LRI [13], wherein a mesh is segmented into distinct patches based on the difference in deformation (rotation in particular) in triangles of the mesh compared to a given rest pose, irrespective of the amount of deformation. The patch-based LRI coordinates have also been used by Gao *et al.*for data-driven shape interpolation in [39], but the number of patches is typically large. For all experiments in this chapter, the number of the segments is less than 10 in our framework. In their framework, the number of patches is a parameter determined by the user. While Gao *et al.*[39] are able to handle large deformations using additional example poses, to some extent, we can handle the large deformations with our segmentation scheme without relying on additional examples. This is demonstrated using interpolation on two different *hand* poses, as shown in Figure 5.4, and can be qualitatively compared with Figure 1 in [39]. It is evident from this comparison that our framework can handle the large deformation present, while the patch-based LRI interpolation could not.

Next, for components undergoing large deformations, we perform a rigid alignment between corresponding components in the source and target meshes. Then,

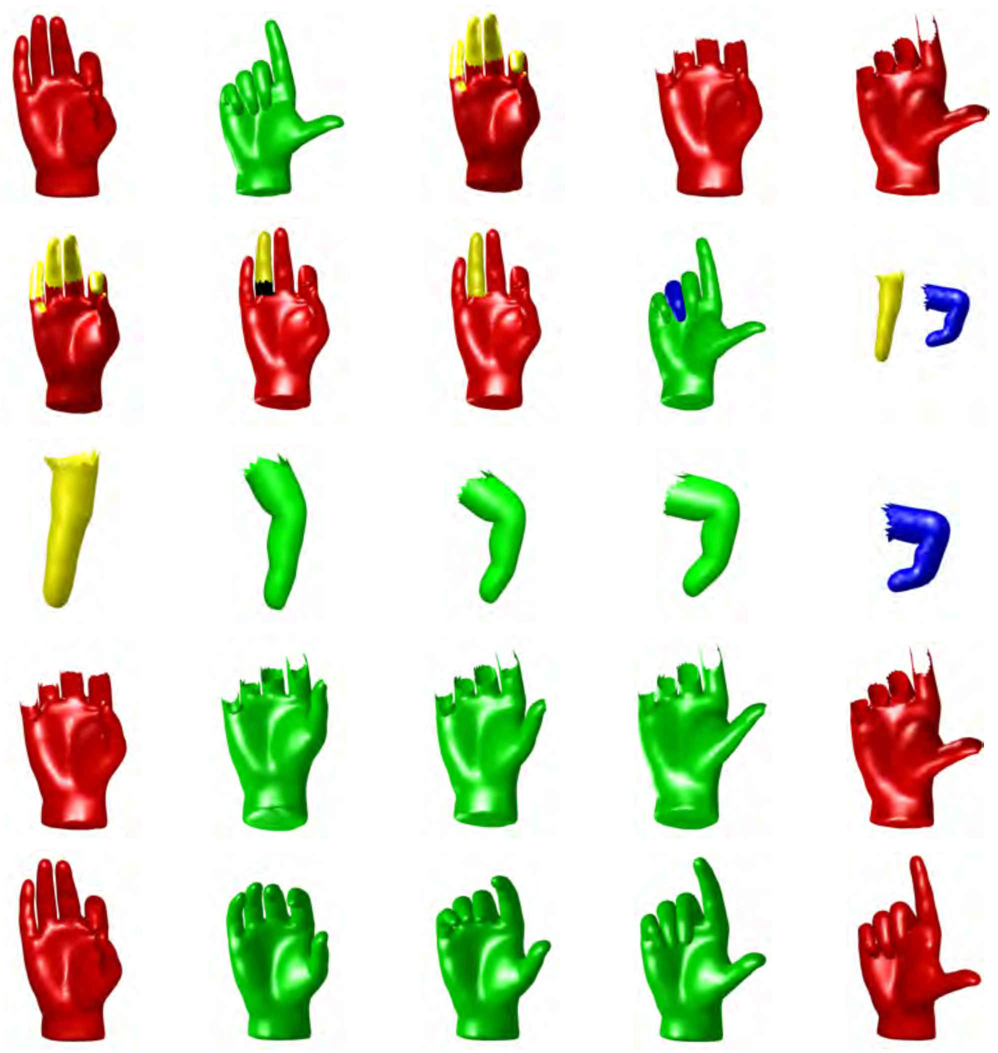

Figure 5.4: Large deformation mesh interpolation example. Row 1: (left to right) Initial pose (red), Target pose (green), Segmentation result with large deformation components shown in yellow, Base mesh component for source and target pose. Row 2: (left to right) large deformation components(yellow), One large deformation component (yellow) extended into neighboring component (black), Extended large deformation component on source pose (yellow), Corresponding component on target pose shown in blue, Rigidly aligned extended large deformation components on source and target poses. Row 3: Interpolation of the aligned extended large deformation component, for (left to right) $t = 0, .25, .5, .75, 1$. Row 4: Interpolation of the BaseMesh component, for (left to right) $t = 0, .25, .5, .75, 1$. Row 5: Final interpolation result, after interpolation and stitching independent components, for (left to right) $t = 0, .25, .5, .75, 1$. For a comparison with patch-based LRI approach refer Figure 1 in [39].

for all components, including components with small deformations, interpolation is performed independently of each other using Algorithm 5. The last step is to stitch small and large deformation interpolated components. This process may introduce discontinuities on the boundary. To alleviate this problem, we extend all large deformation components to include triangles from surrounding small deformation components within the 3-ring neighborhood of all large deformation component boundary triangles. Since the overlap between small and large deformation components is known, the interpolated components are stitched via a rigid alignment computed only on the overlapping 3-ring neighborhood. The rigid transformation is applied on the entire large deformation component interpolation result, while the stitching is done by selecting the overlapping region from the interpolated small deformation component, and ignoring the same from the aligned large deformation component. This segmentation based interpolation is summarized in Algorithm 6 and is demonstrated in Figure 5.4.

5.4 Results

In this section we provide results of our experiments on morphing and interpolation. Results have been computed for datasets from [96, 85, 68]. Morphing refers to computing intermediate deformations between two objects belonging to different classes. For all morphing related experiments, we do not use our segmentation based framework, while all interpolation results have been computed using our segmentation based framework.

1. Morphing results: Results for morphing between Humanoid meshes, Fandisk-Cube, Cat-Bunny, Bunny-Rabbit, and Head-Venus is shown in Figure 5.6. The correspondence used for the Head-Venus example is the one provided by [96]. We sample the geodesic at $t = 0.25, 0.5, 0.75$ and generate the corresponding intermediate meshes shown in this figure. Note that the humanoid example shows that our approach can simultaneously handle morphing (thin-fat) as well as the interpolation (position of hands).

2. Multiple model morphing: In Figure 5.5, we show the morphing of a source model (Camel) into multiple target models (Horse and Dinosaur).

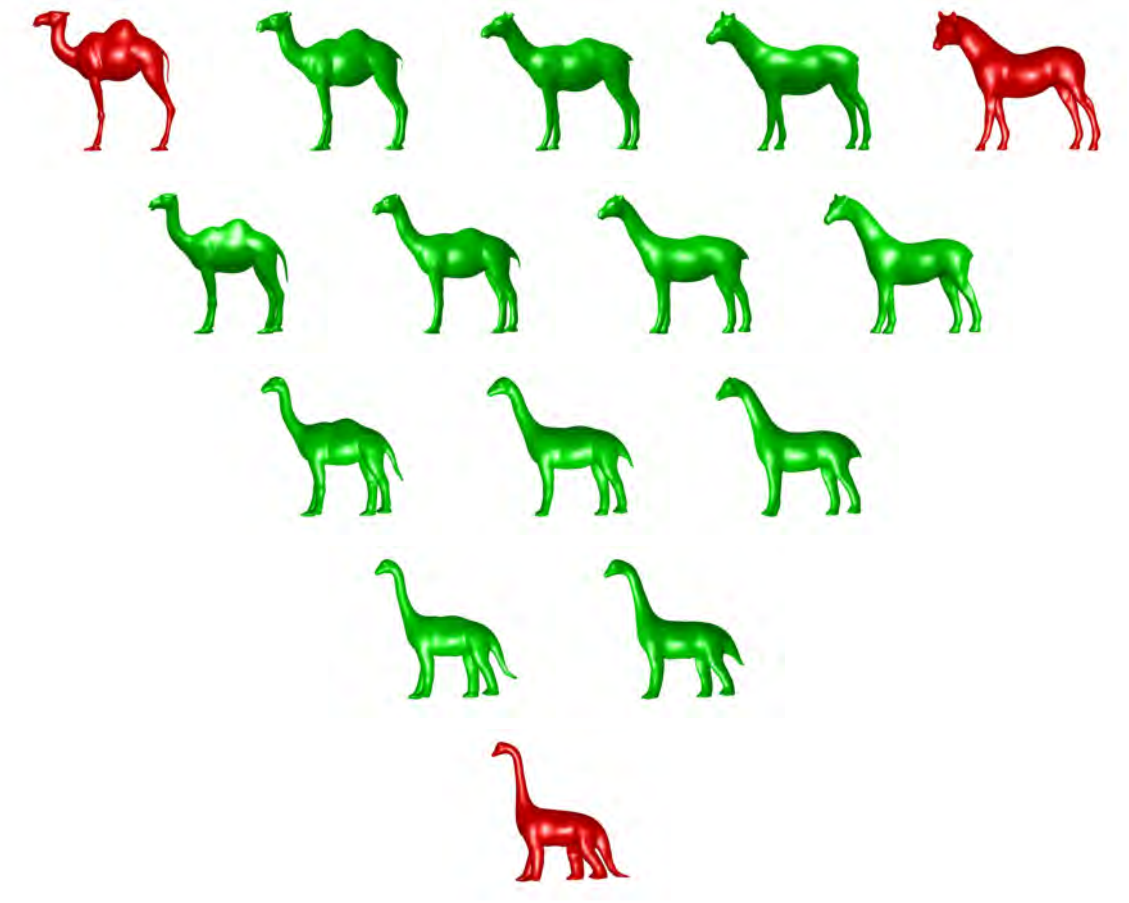

Figure 5.5: Morphing using multiple models.

3. Large deformation interpolation and extrapolation: Interpolation and extrapolation results for deformations of Horse and Elephant meshes are shown in Figure 5.7. In addition to intermediate meshes shown at $t = 0.25, 0.5, 0.75$, we extrapolate the deformation for the Horse and Elephant meshes and provide results for $t = -0.25$ and $t = 1.25$. These interpolations have been produced using our segmentation algorithm based on detecting large deformation triangles. It is important to note that the Lie bodies framework without our segmentation approach will produce effects similar to those shown in Figure 5.3. Similarly, results of interpolation and extrapolation of large scale deformations for the Bar are shown in Figure 5.1, while those for Cylinder and Helix meshes are shown in Figure 5.8. With this segmentation, the proposed approach is able to interpolate and extrapolate over multiple twists. As discussed in the previous section, interpolation results for the large de-

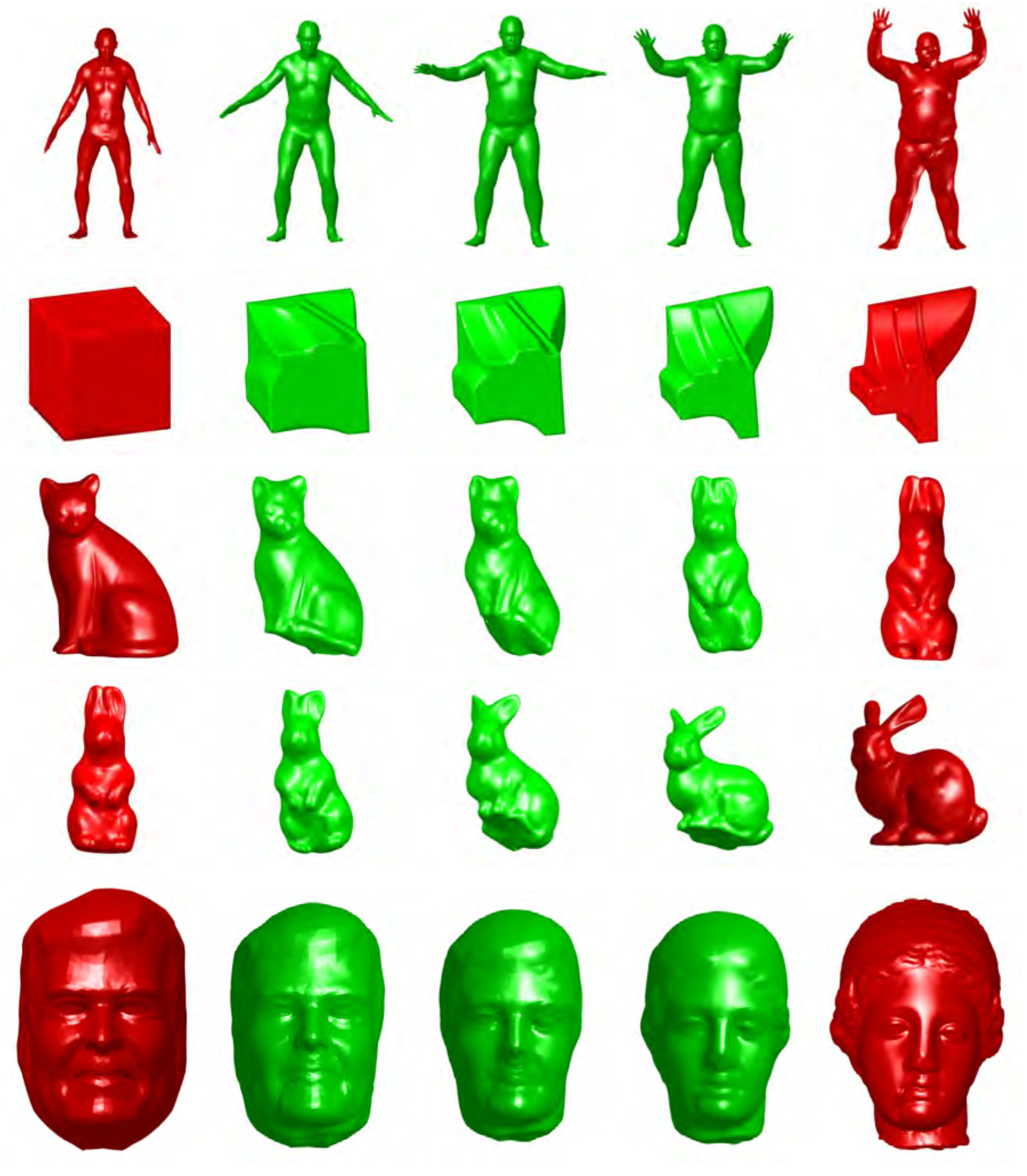

Figure 5.6: Morphing of different models. First column and last column show source and target models. Column 2-4 show morphing results by sampling the geodesic at $t = 0.25, 0.5, 0.75$. Notice the simultaneous interpolation and morphing in Row 1.

formation of *hand poses* is given in Figure 5.4, while results using patch-based LRI on similar examples can be found in Figure 1 in [39], where it is shown to fail.

The results produced using the proposed approach is visually comparable

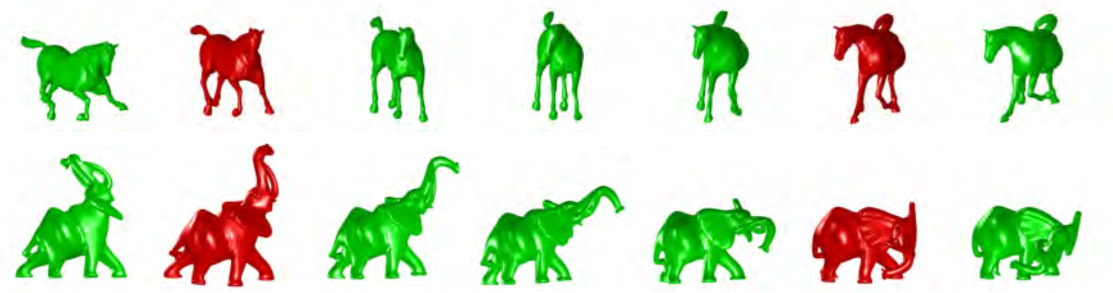

Figure 5.7: Mesh interpolation of Horse, Elephant and Hand meshes. Extrapolation results for Horse and Elephant meshes are included. (Row 1 & 2) Column-wise: Mesh at $t = -.25, 0, 0.25, 0.5, 0.75, 1, 1.25$. $t = 0, 1$ represent source and target mesh respectively, in all cases.

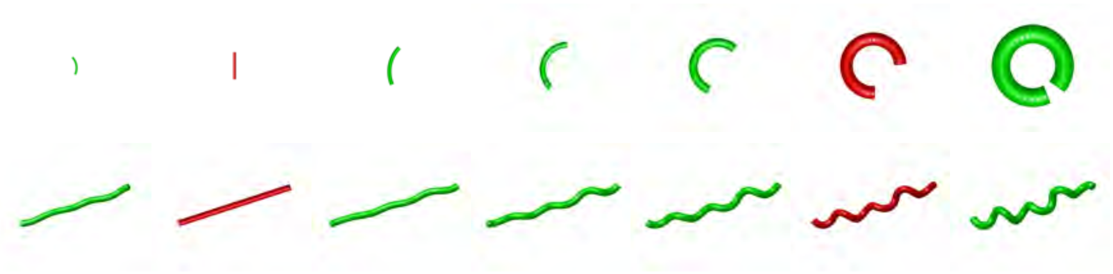

Figure 5.8: Example of mesh interpolation and extrapolation over large deformations of Cylinder and Helix mesh. Column-wise from left to right, meshes at $t = -.25, 0, .25, .5, .75, 1, 1.25$. $t = 0$ and 1 represent source and target mesh respectively. Note that both scaling and bending of the cylinder are captured in the interpolation and extrapolation.

to state-of-the-art approaches presented in [20, 101, 41, 85], though in these approaches, in order to generate intermediate mesh at some $t \in [0, 1]$, one needs to compute the mesh at $t - \Delta t$ for a small Δt. Hence, the interpolation process is serial in nature and ends up being computationally expensive. Our method, on the other hand, can sample the geodesic at an arbitrary parameter t without computing the mesh at parameter $t - \Delta t$. Computationally, our framework is thus faster. To give an idea of the computational time required for our framework, we provide the time required for our segmentation process and the average time required to interpolate one pose for several models in Table 5.1. Note that the interpolation time includes time for the least-square mesh reconstruction. Our current implementation use *Libigl* [49], runs on an Intel5 8GB RAM, 1.6 GHz system, without

Table 5.1: Computational time (in seconds) for the proposed algorithm for various meshes. Note that the interpolation time provided includes the time required for mesh reconstruction and no parallelization has been employed in our current implementation. For morphing results no segmentation process is used.

Mesh	Vertices	Faces	Segmentation	Segments (count)	Interpolation (per pose)
Cylinder	2442	4880	0.001	2	0.07
Bar	2882	5740	0.001	2	0.05
Helix	1212	2420	0.002	3	0.02
Hand	6094	12184	0.02	6	0.09
Horse	8431	16843	0.007	7	0.26
Elephant	39969	79946	0.22	5	0.35
Human	6890	13776	-	-	0.14
Cube	6583	13162	-	-	0.13
Cat	5618	11232	-	-	0.11
Rabbit	5618	11232	-	-	0.11
head	7702	15400	-	-	0.20
Camel	6013	12022	-	-	0.11

using any parallelization.

The robustness of our framework is based on how well each large deformation component can be stitched together. While, we have extended the large deformation component into its neighborhood components to achieve this robustness, the underlying assumption is that deformation between the shapes is smooth and not abrupt. This assumption is valid for most natural objects but there can exist cases where this assumption is not valid. For all the examples we have worked on, the framework has produced satisfactory results.

5.5 Conclusion

A unified framework for mesh interpolation and morphing is proposed, which models both these problems as finding geodesics between points in an appropriate shape space. With the proposed algorithms for segmentation based on large deformation components, the framework is able to handle interpolation between large deformations of objects, and is computationally efficient.

CHAPTER 6

Future Work and Discussion

A lot of research is taking place to develop a robust and efficient pipeline for processing 3D data, form content generation (3D reconstruction) and shape editing to rendering & visualization of 3D data.

There are several aspects of the work presented in this thesis, which need a more closer look and can be explored further. (1) Issues with Lie group mesh representation: (a) Handling topology changes. In applications where more than one shape is involved, all the shapes are assumed to share same topology, which is a very restrictive constraint. Specifically, for applications involving learning, fixing topology restrict the amount of data which can be used for learning. (b) Each element has an independent representation. While this gives an advantage in form of computational efficiency, treating each element independently does not fully exploit the neighborhood information. Furthermore, one possible implication of this, as shown in interactive shape deformation, is translation insensitivity, where the framework fails to represent the pure translation component of the deformation using local rotations. Finding a mesh representation which can be used without any constraint on the topology of the mesh and which captures neighborhood information could yield better performance. (2) Geodesics on Lie group: The proposed framework uses group geodesics which are defined in closed form on Lie groups under consideration. Again, while this makes the framework fast, a comparison with Remannnian geodesics is needed to fully justify the choice. (3) Data driven approaches for proposed representation and applications: As already shown in Chapter 3, the proposed framework is amenable to data driven techniques giving more realistic results. Data driven approaches can possibly be used to provide rigidity control in shape deformation by restricting the deformations to a shape space or to provide a realistic interpolation in shape space.

Apart from traditional mathematical modeling of geometry processing tasks, the focus is now shifting to data driven geometry processing. Till recently the data-driven techniques were not very practical for geometry processing tasks. This was mostly due to non-availability of 3D data and lack of computational power needed for data-driven geometry processing. With developing technologies for camera sensors, we now have cheap sensors which can produce a huge amount of 3D data. As in the field of image and signal processing, deep learning based methods have become popular in order to process 3D geometric data. This area, with its set of learning based tools and algorithms is called *geometric deep learning* [22]. The use of deep learning frameworks on geometric data, which is not regular, is not straightforward. There are two main issues: (1) Non-availability of the operators such as convolutions and pooling on geometric data. Existing approaches in geometric deep learning are based on generalizing classical definitions of convolution and pooling [30, 62]. (2) The representation of geometric data and topological inconsistency in it. While topological variations are not trivial to handle even for traditional geometry processing algorithms, more efforts are needed to fully leverage the power of deep learning [22].

6.1 Summary

To summarize, in this thesis, we proposed a Lie group representation of a tetrahedron. The proposed representation is used to interpolate affine transformations. The interpolation framework has several nice properties, which along with invariance properties have been analyzed in detail. It is shown that the tetrahedron representation can be used to represent triangles and in turn tetrahedral and triangular meshes. As applications of the proposed framework we contributed towards interactive shape deformation, deformation transfer, shape interpolation and morphing. The representation can be parallelized and, thus, is computationally efficient.

References

[1] M. Alexa. Linear Combination of Transformations. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 380–387. ACM, 2002.

[2] M. Alexa. Recent Advances in Mesh Morphing. In *Computer graphics forum*, volume 21, pages 173–198. Wiley Online Library, 2002.

[3] M. Alexa. Differential Coordinates for Local Mesh Morphing and Deformation. *The Visual Computer*, 19(2):105–114, 2003.

[4] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible Shape Interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.

[5] D. Ali-Hamadi, T. Liu, B. Gilles, L. Kavan, F. Faure, O. Palombi, and M.-P. Cani. Anatomy Transfer. *ACM Trans. Graph.*, 32(6):188:1—-188:8, nov 2013.

[6] B. Allen, B. Curless, B. Curless, and Z. Popović. The space of human body shapes: Reconstruction and parameterization from range scans. In *ACM transactions on graphics (TOG)*, volume 22, pages 587–594. ACM, 2003.

[7] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape Completion and Animation of People. In *ACM transactions on graphics (TOG)*, volume 24, pages 408–416. ACM, 2005.

[8] Q. Avril, D. Ghafourzadeh, S. Ramachandran, S. Fallahdoust, S. Ribet, O. Dionne, M. de Lasa, and E. Paquette. Animation Setup Transfer for 3D Characters. *Computer Graphics Forum*, 35(2):115–126, 2016.

[9] S.-Y. Baek, J. Lim, and K. Lee. Isometric Shape Interpolation. *Computers & Graphics*, 46:257–263, 2015.

[10] S. Bansal and A. Tatu. Lie bodies based 3D shape morphing and interpolation. In *The 15th ACM SIGGRAPH European Conference on Visual Media Production (CVMP 2018)*. ACM, 2018.

[11] S. Bansal and A. Tatu. Lie bodies based deformation transfer. In *11th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP 2018)*. ACM, 2018.

[12] S. Bansal and A. Tatu. Affine Interpolation in a Lie Group Framework. *ACM Trans. Graph.*, 38(4):71:1—-71:16, jul 2019.

[13] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic Deformation Transfer. *ACM Trans. Graph.*, 28(3):36:1—-36:6, jul 2009.

[14] D. Bechmann. Space Deformation Models Survey. *Computers & Graphics*, 18(4):571–586, 1994.

[15] M. Ben-Chen, O. Weber, and C. Gotsman. Spatial Deformation Transfer. In *Proceedings of the 2009 ACM SIGGRAPH*, SCA '09, pages 67–74, New York, NY, USA, 2009. ACM.

[16] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and Evaluation for 3D Mesh Registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[17] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. PriMo: Coupled Prisms for Intuitive Surface Modeling. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[18] M. Botsch and O. Sorkine. On Linear Variational Surface Deformation Methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, jan 2008.

[19] M. Botsch, R. Sumner, M. Pauly, and M. Gross. Deformation Transfer for Detail-Preserving Surface Editing. In *Vision, Modeling & Visualization*, pages 357–364, 2006.

[20] C. Brandt, C. von Tycowicz, and K. Hildebrandt. Geometric Flows of Curves in Shape Space for Processing Motion of Deformable Objects. *Computer Graphics Forum*, 2016.

[21] A. Bronstein, M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[22] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[23] S. R. Buss and J. P. Fillmore. Spherical Averages and Applications to Spherical Splines and Interpolation. *ACM Trans. Graph.*, 20(2):95–126, apr 2001.

[24] Y.-T. Chang, B.-Y. Chen, W.-C. Luo, and J.-B. Huang. Skeleton-driven Animation Transfer based on Consistent Volume Parameterization. In *Proceedings of Computer Graphics International 2006*, pages 78–89, 2006.

[25] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A Simple Geometric Model for Elastic Deformations. *ACM Trans. Graph.*, 29(4):38:1—-38:6, jul 2010.

[26] L. Chen, J. Huang, H. Sun, and H. Bao. Cage-based Deformation Transfer. *Computers & Graphics*, 34(2):107–118, 2010.

[27] H.-K. Chu and T.-Y. Lee. Multiresolution Mean Shift Clustering Algorithm for Shape Interpolation. *IEEE transactions on visualization and computer graphics*, 15(5):853–866, 2009.

[28] D. Cohen-Or. Space Deformations, Surface Deformations and the Opportunities In-Between. *Journal of Computer Science and Technology*, 24(1):2–5, jan 2009.

[29] D. Cohen-Or, A. Solomovic, and D. Levin. Three-dimensional Distance Field Metamorphosis. *ACM Trans. Graph.*, 17(2):116–141, apr 1998.

[30] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 2016.

[31] K. G. Der, R. W. Sumner, and J. Popović. Inverse Kinematics For Reduced Deformable Models. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1174–1179. ACM, 2006.

[32] I. L. Dryden and K. V. Mardia. *Statistical Shape Analysis, with Applications in R. Second Edition.* John Wiley and Sons, Chichester, 2016.

[33] P. T. Fletcher, C. Lu, S. M. Pizer, and S. C. Joshi. Principal Geodesic Analysis for the Study of Nonlinear Statistics of Shape. *{IEEE} Trans. Med. Imaging*, 23(8):995–1005, 2004.

[34] O. Freifeld and M. J. Black. Lie Bodies: A Manifold Representation of {3D} Human Shape. In *European Conf. on Computer Vision (ECCV)*, Part I, LNCS 7572, pages 1–14. Springer-Verlag, oct 2012.

[35] S. Fröhlich and M. Botsch. Example-Driven Deformations Based on Discrete Shells. *Computer Graphics Forum*, 30(8):2246–2257, 2011.

[36] J. Gain and D. Bechmann. A Survey of Spatial Deformation from a User-centered Perspective. *ACM Trans. Graph.*, 27(4):107:1—-107:21, nov 2008.

[37] J. Gallier and J. Quaintance. Notes on Differential Geometry and Lie Groups, feb 2017.

[38] L. Gao, Y.-K. Lai, Q.-X. Huang, and S.-M. Hu. A Data-Driven Approach to Realistic Shape Morphing. In *Computer graphics forum*, volume 32, pages 449–457. Wiley Online Library, 2013.

[39] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia. Efficient and Flexible Deformation Representation for Data-Driven Surface Modeling. *ACM Trans. Graph.*, 35:158:1—-158:17, jul 2016.

[40] U. Grenander and M. I. Miller. Computational anatomy: An emerging discipline. *Quarterly of applied mathematics*, 56(4):617–694, 1998.

[41] B. Heeren, M. Rumpf, M. Wardetzky, and B. Wirth. Time-Discrete Geodesics in the Space of Shells. In *Computer Graphics Forum*, volume 31, pages 1755–1764. Wiley Online Library, 2012.

[42] K. Hildebrandt, C. Schulz, C. V. Tycowicz, and K. Polthier. Interactive Surface Modeling Using Modal Analysis. *ACM Trans. Graph.*, 30(5):119:1—-119:11, oct 2011.

[43] J. Hu, L. Liu, and G. Wang. Dual Laplacian Morphing for Triangular Meshes. *Computer Animation and Virtual Worlds*, 18(4-5):271–277, 2007.

[44] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace Gradient Domain Mesh Deformation. *ACM Trans. Graph.*, 25(3):1126–1134, jul 2006.

[45] Q.-X. Huang, B. Adams, M. Wicke, and L. J. Guibas. Non-rigid Registration Under Isometric Deformations. In *Computer Graphics Forum*, volume 27, pages 1449–1457. Wiley Online Library, 2008.

[46] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine. Fast Automatic Skinning Transformations. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 31(4):77:1—-77:10, 2012.

[47] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded Biharmonic Weights for Real-time Deformation. *ACM Trans. Graph.*, 30(4):78:1—-78:8, jul 2011.

[48] A. Jacobson, Z. Deng, L. Kavan, and J. P. Lewis. Skinning: Real-time Shape Deformation. In *ACM SIGGRAPH 2014 Courses*, 2014.

[49] A. Jacobson, D. Panozzo, and Others. {libigl}: A simple {C++} geometry processing library, 2017.

[50] S. Kaji, S. Hirose, S. Sakata, Y. Mizoguchi, and K. Anjyo. Mathematical Analysis on Affine Maps for 2D Shape Interpolation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 71–76, Goslar Germany, Germany, 2012. Eurographics Association.

[51] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Trans. Graph.*, 27(4):105:1—-105:23, nov 2008.

[52] L. Kavan and J. Žára. Spherical Blend Skinning: a Real-time Deformation of Articulated Models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16. ACM, 2005.

[53] D. G. Kendall. A survey of the statistical theory of shape. *Statist. Sci.*, 4(2):87–99, 05 1989.

[54] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric Modeling in Shape Space. In *ACM Transactions on Graphics (TOG)*, volume 26, page 64. ACM, 2007.

[55] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel. Coupled Quasi-harmonic Bases. In *Computer Graphics Forum*, volume 32, pages 439–448. Wiley Online Library, 2013.

[56] H. Laga, Q. Xie, I. H. Jermyn, and A. Srivastava. Numerical Inversion of {SRNF} Maps for Elastic Shape Analysis of Genus-Zero Surfaces. *{IEEE} Trans. Pattern Anal. Mach. Intell.*, 39(12):2451–2464, 2017.

[57] Z. Levi and C. Gotsman. Smooth Rotation Enhanced As-Rigid-As-Possible Mesh Animation. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):264–277, feb 2015.

[58] B. Lévy and H. R. Zhang. Spectral Mesh Processing. In *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, pages 8:1—-8:312, New York, NY, USA, 2010. ACM.

[59] J. Li and P.-w. Hao. Smooth Interpolation on Homogeneous Matrix Groups for Computer Animation. *Journal of Zhejiang University-SCIENCE A*, 7(7):1168–1177, jul 2006.

[60] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear Rotation-invariant Coordinates for Meshes. *ACM Transactions on Graphics (TOG)*, 24(3):479–487, 2005.

[61] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint-dependent Local Deformations for Hand Animation and Object Grasping. In *In Proceedings on Graphics interface'88*. Citeseer, 1988.

[62] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017.

[63] L. Moser, D. Corral, and D. Roble. As-rigid-as-possible Deformation Transfer for Facial Animation. In *ACM SIGGRAPH 2016 Talks*, SIGGRAPH '16, pages 29:1—-29:2, New York, NY, USA, 2016. ACM.

[64] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.

[65] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. Sparse Localized Deformation Components. *ACM Trans. Graph.*, 32(6):179:1—-179:10, nov 2013.

[66] H. Ochiai and K. Anjyo. Mathematical Description of Motion and Deformation: From Basics to Graphics Applications. In *{SIGGRAPH} Asia 2013, Hong Kong, China, November 19-22, 2013, Courses*, pages 2:1—-2:47, 2013.

[67] X. Pennec, P. Fillard, and N. Ayache. A Riemannian Framework for Tensor Computing. *International Journal of Computer Vision*, 66(1):41–66, jan 2006.

[68] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A Model of Dynamic Human Shape in Motion. *ACM Transactions on Graphics, (Proc. SIGGRAPH)*, 34(4):120:1—-120:14, aug 2015.

[69] M. M. Postnikov. *Geometry VI: Riemannian Geometry*. Encyclopaedia of Mathematical Sciences. Springer-Verlag Berlin Heidelberg, 2001.

[70] E. Rodolà, S. Rota Bulo, T. Windheuser, M. Vestner, and D. Cremers. Dense Non-rigid Shape Correspondence Using Random Forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4177–4184, 2014.

[71] G. Rong, Y. Cao, and X. Guo. Spectral Mesh Deformation. *Visual Computing*, 24:787–796, 2008.

[72] J. Rossignac and Á. Vinacua. Steady Affine Motions and Morphs. *ACM Trans. Graph.*, 30(5):116:1—-116:16, oct 2011.

[73] N. A. Rumman and M. Fratarcangeli. *State-of-the-art in Skinning Techniques for Articulated Deformable Characters*, pages 200–212. SciTePress, 2016.

[74] J. M. Selig. *Geometric Fundamentals of Robotics*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[75] A. E. R. Shabayek, D. Aouada, A. Saint, and B. Ottersten. Deformation transfer of 3d human shapes and poses on manifolds. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 220–224. IEEE, 2017.

[76] K. Shoemake. Animating Rotation with Quaternion Curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, jul 1985.

[77] K. Shoemake. Matrix Animation and Polar Decomposition. *Graphics Interface'92, 1992*, 1992.

[78] D. Sieger, S. Gaulik, J. Achenbach, S. Menzel, and M. Botsch. Constrained Space Deformation Techniques for Design Optimization. *Computer-Aided Design*, 72:40–51, 2016.

[79] O. Sorkine and M. Alexa. As-rigid-as-possible Surface Modeling. In *Symposium on Geometry processing*, volume 4, 2007.

[80] O. Sorkine and D. Cohen-Or. Least-squares Meshes. In *Shape Modeling Applications, 2004. Proceedings*, pages 191–199. IEEE, 2004.

[81] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian Surface Editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.

[82] R. W. Sumner and J. Popović. Deformation Transfer for Triangle Meshes. *ACM Transactions on Graphics*, 23(3):399–405, aug 2004.

[83] R. W. Sumner, J. Schmid, and M. Pauly. Embedded Deformation for Shape Manipulation. *ACM Transactions on Graphics*, 26(3), jul 2007.

[84] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based Inverse Kinematics. *ACM Trans. Graph.*, 24(3):488–495, jul 2005.

[85] W. T., D. J., A. M., and H. K. MultiâĂŘScale Geometry Interpolation. *Computer Graphics Forum*, 29(2):309–318.

[86] J. A. Thomson. On growth and form. *Nature*, 100(2498):21, 1917.

[87] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A Survey on Shape Correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.

[88] C. Von-Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt. Real-time Nonlinear Shape Interpolation. *ACM Transactions on Graphics (TOG)*, 34(3):34, 2015.

[89] Y. Wang, A. Jacobson, J. Barbič, and L. Kavan. Linear Subspace Design for Real-time Shape Deformation. *ACM Trans. Graph.*, 34(4):57:1—-57:11, jul 2015.

[90] Y. Wang, B. Liu, and Y. Tong. Linear Surface Reconstruction from Discrete Fundamental Forms on Triangle Meshes. *Computer Graphics Forum*, 2012.

[91] W. Welch and A. Witkin. Variational Surface Modeling. *SIGGRAPH Comput. Graph.*, 26(2):157–166, jul 1992.

[92] Y. Weng, M. Chai, W. Xu, Y. Tong, and K. Zhou. As-Rigid-As-Possible Distance Field Metamorphosis. *Computer Graphics Forum*, 2013.

[93] B. Whited, G. Noris, M. Simmons, R. Sumner, M. Gross, and J. Rossignac. BetweenIT: An Interactive Tool for Tight Inbetweening. *Comput. Graphics Forum (Proc. Eurographics)*, 29(2):605–614, 2010.

[94] B. Wu, K. Xu, Y. Zhou, Y. Xiong, and H. Huang. Skeleton-guided 3D Shape Distance Field Metamorphosis. *Graphical Models*, 85:37–45, 2016.

[95] D. Xu, H. Zhang, Q. Wang, and H. Bao. Poisson Shape Interpolation. *Graphical models*, 68(3):268–281, 2006.

[96] I.-C. Yeh, C.-H. Lin, O. Sorkine, and T.-Y. Lee. Template-based 3D Model Fitting Using Dual-domain Relaxation. *IEEE Transactions on Visualization and Computer Graphics*, 99(RapidPosts), 2010.

[97] M. Yin, G. Li, H. Lu, Y. Ouyang, Z. Zhang, and C. Xian. Spectral Pose Transfer. *Comput. Aided Geom. Des.*, 35(C):82–94, may 2015.

[98] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh Editing with Poisson-based Gradient Field Manipulation. *ACM Trans. Graph.*, 23(3):644–651, aug 2004.

[99] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic Guidance for Surface Deformation. In *Computer Graphics Forum*, volume 24, pages 601–609. Wiley Online Library, 2005.

[100] M. Zefran and V. Kumar. Interpolation Schemes for Rigid Body Motions. *Computer-Aided Design*, 30(3):179–189, 1998.

[101] Z. Zhang, G. Li, H. Lu, Y. Ouyang, M. Yin, and C. Xian. Fast As-isometric-as-possible Shape Interpolation. *Computers & Graphics*, 46:244–256, 2015.

[102] K. Zhou, W. Xu, Y. Tong, and M. Desbrun. Deformation Transfer to Multi-Component Objects. *Computer Graphics Forum*, 29(2):319–325, 2010.

[103] F. Zhu, S. Li, and G. Wang. Example Based Materials in Laplace Beltrami Shape Space. *Computer Graphics Forum*, 34(1):36–46, 2015.

CHAPTER A

Appendix

A.1 Mathematical Background

The space of shape deformations is non-linear. Most of the time, either shape representations or deformation representations used in several of the state-of-the-arts methods for deformation modeling, use components from non-linear spaces. While almost all the representations for shape deformation use some form of rotation transformation, usually additional components are used which also come from non-linear spaces.

In this thesis most of the mathematical presentation is based on Lie groups, specifically, linear Lie groups which are also known as matrix Lie groups. Lie groups lie at the intersection of groups and manifolds, i.e, a Lie group $\mathcal{G}$ is a group $(\mathcal{G}, \cdot)$ and a smooth manifold, such that the group multiplication and inversion operations are smooth.

Intuitively, a manifold is a space such that a small neighborhood around each point of this space can be mapped to a Euclidean space. In more technical terms, a subset M of $\mathbb{R}^N$ is a manifold, if $\forall p \in M \; \exists \mathcal{U} \subseteq M$ and $\omega \subseteq \mathbb{R}^N$, such that $\phi(\omega) = \mathcal{U}$, where $\phi : \omega \to \mathcal{U}$ is a smooth homeomorphism. The map ϕ is called coordinate chart (chart in short). Generally, many such charts are needed to cover the entire manifold. In case two charts have some overlapping region, it is possible to have two different representations of the same region though these charts. A transition map is used to move between different representations. For **smooth manifolds**, the transition maps are infinitely differentiable (smooth) [37].

Lie groups. A subset $\mathcal{G}$ of $\mathbb{R}^N$ is a Lie group if:

1. $\mathcal{G}$ is a group.

2. $\mathcal{G}$ is a manifold in $\mathbb{R}^N$.

3. The group operation and inverse map defined as $. : G \times G \to G$ and $i : G \to G$ are smooth.

Although, it is not a trivial task to prove that a set is a Lie group, but in case of linear Lie groups, using the Theorem by Von Neumann and Carton (theorem 4.8 [37]) the task reduces to proving that the set is a closed subgroup of $GL(n, \mathbb{R})$. $GL(n, \mathbb{R})$ is the group of all $n \times n$ real-valued invertible matrices.
Theorem (Von Neumann and Carton): A closed subgroup $\mathcal{G}$ of $GL(n, \mathbb{R})$ is a linear Lie group. Furthermore the set $\mathfrak{g}$ defined as

$$\mathfrak{g} = \{X \in M_n(\mathbb{R}) | e^{tX} \in \mathcal{G} \text{ for all } t \in \mathbb{R}\}$$

is a non-trivial vector space equal to the tangent space $T_I\mathcal{G}$ at the identity I. $M_n(\mathbb{R})$ is the set of all $n \times n$ real-valued matrices.

Lets take a look at some linear Lie groups. (1) The special linear group defined as $SL(n, \mathbb{R}) = \{X \in GL(n, \mathbb{R}) | det(X) = 1\}$ is a Lie group. It is easy to show that $SL(n, \mathbb{R})$ is a subgroup of $GL(n, \mathbb{R})$. The $det(X) = 1$, for $X \in SL(n, \mathbb{R})$. Using the fact that *det()* can be shown to be a group homomorphism between $GL(n, \mathbb{R})$ and $\mathbb{R}^*$, where $\mathbb{R}^*$ be the multiplicative group of nonzero real numbers, the $SL(n, \mathbb{R})$ turn out to be a normal subgroup of $GL(n, \mathbb{R})$. (2) Orthogonal group defined as the group of distance-preserving transformations $O(n, \mathbb{R}) = \{X \in GL(n, \mathbb{R}) | X^T X = XX^T = I\}$, where I is the identity matrix, is a Lie group. Following simple definition of the subgroup, it can be shown that the $O(n, \mathbb{R})$ is a closed subgroup to the $GL(n, \mathbb{R})$. Other popular examples are special orthogonal group $SO(n, R)$ and group of symmetric positive definite matrices.
Lie algebra. Let $\mathcal{G}$ be a matrix Lie group with dimension m as a manifold. Then, the tangent space at identity $e \in \mathcal{G}$, denoted by $T_e\mathcal{G}$ can be identified with the Lie algebra. The Lie algebra is denoted by $\mathfrak{g}$ and is defined as $\mathfrak{g} = \{X \in M_n(\mathbb{R}) \mid \exp(X) \in \mathcal{G}\}$, where $M_n(\mathbb{R})$ denotes the set of all $n \times n$ real matrices, and exp denotes the

matrix exponential: $\exp(X) = \sum_{i=0}^{\infty} \frac{X^i}{i!}$. Note that $\mathfrak{g}$ is an m-dimensional vector space, and one can think of $\mathfrak{g}$ as a linearized approximation of $\mathcal{G}$ around the point e. The tangent space at an arbitrary point $p \in \mathcal{G}$ is defined using left (or right) translation as $T_p\mathcal{G} = \{pX \mid X \in \mathfrak{g}\}$.

Exponential and log maps. The Lie algebraic exponential and log maps coincide with the matrix exponential and log map (which is the inverse of the exponential wherever possible) for matrix Lie groups, and can be used to go back and forth between $\mathcal{G}$ and $\mathfrak{g}$: $\exp : \mathfrak{g} \to \mathcal{G}$ and $\log : \mathcal{G} \to \mathfrak{g}$. The Lie algebraic exponential map is a smooth map between $\mathfrak{g}$ and $\mathcal{G}$, such that it maps an element $v \in \mathfrak{g}$ to the element $\gamma(1) = \exp(v) \in \mathcal{G}$, where $\gamma : \mathbb{R} \to \mathcal{G}$ is the unique integral curve[1] generated by the left-invariant vector field generated by v. In what follows, both Lie algebraic and matrix exponential and log maps will be referred to simply as exponential and log maps.

Geodesics. A geodesic curve between e and an element $p \in \mathcal{M}$ is given by $\exp(t \log(p))$, while that between two points $p, q \in \mathcal{M}$ is given by first left translating the two points with p^{-1}, computing the geodesic between the resulting points e and $p^{-1}q$ and left translating the geodesic again by p, i.e., $c(t) = p \exp(t \log(p^{-1}q))$, refer Figure A.1.

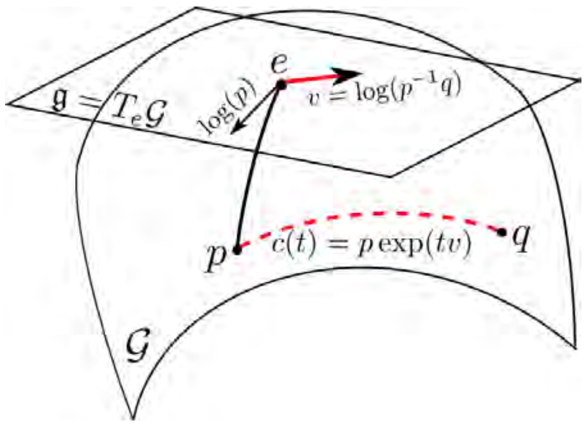

Figure A.1: Exponential & Log map on a matrix Lie group $\mathcal{G}$. The vector $v = \log(p^{-1}q) \in \mathfrak{g}$ shown in red, can be used to compute the geodesic between p and q as $c(t) = p\exp(tv)$, shown as a red curve with dashes.

[1]An integral curve of a vector field is a curve whose tangent vector at any point of the curve is equal to the given vector field at the corresponding point.

A.2 Proofs of Theorems & Lemmas

This appendix includes proofs of the statements, theorems and lemmas from Chapter 2.

A.2.1 G_A is a Lie group.

Proof. Using the fact that G_A is a subgroup in $GL(3)$ and Definition 4.6 in [37],

$$G_A = \left(\begin{bmatrix} 1 & \alpha_1 & \alpha_3 \\ 0 & \alpha_2 & \alpha_4 \\ 0 & 0 & \alpha_5 \end{bmatrix} | \alpha_2, \alpha_5 \in \mathbf{R}^+, \alpha_1, \alpha_3, \alpha_4 \in \mathbf{R} \right)$$

is linear Lie group as: $\forall A \in G_A, \phi : G_A \to \mathbf{R}^5$ given by $\phi(A) = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) \in \mathbf{R}^5$, (ϕ, G_A) is a smooth manifold in $\mathbf{R}^5$. $\square$

A.2.2 (Theorem 2.3.1)

1. An affine transformation $\mathbb{T}$ can be decomposed into a translation and a linear component, $\mathbb{T} = [L \mid d]$. The translation part d aligns the two centroids and is thus uniquely determined. The linear part can then be written as a map of the difference vectors of vertices with the centroid from one tetrahedron to corresponding difference vectors in the other tetrahedron. Since the tetrahedrons are non-degenerate, these two sets of difference vectors are linearly independent, implying the uniqueness of L. Moreover, since the two tetrahedrons have the same orientation, $det(\mathbb{T}) > 0$.

2. The translation part is invertible, while the linear part is invertible as far as the tetrahedrons are non-degenerate. Hence $\mathbb{T}^{-1}$ exists. Fixing tetrahedron Δ_Y, tetrahedron Δ_X is given by $\Delta_X = \mathbb{T}^{-1}\Delta_Y$ or fixing tetrahedron Δ_X, tetrahedron Δ_Y is given by $\Delta_Y = \mathbb{T}\Delta_X$.

3. We will first prove the statement for the case where Δ_Y is assumed to be the canonical tetrahedron, and later generalize it for any arbitrary tetrahedron. In order to write the affine transformation $\mathbb{T}$ as a unique product of elements from $SE(3), G_S$ and G_A, we show that the three components are unique. To begin with, there exists a unique $SE(3)$ element E that does the rigid alignment of the tetrahedron to the tetrahedron with vertices $(\mathbf{0}, v_1, v_2, v_3)$ (refer to Figure 2.1), since there exists a unique rotation matrix that aligns two orthogonal vectors, and a unique translation vector that aligns two points in $\mathbb{R}^3$. Then, a unique uniform scaling (in $\mathbb{R}^3$) that aligns the second vertex, say $v_1 = (v_{1_x}, 0, 0)$ to the vector $(1, 0, 0)$ is given by $s = \dfrac{1}{v_{1_x}}$. Let the third and fourth vertex coordinates after the uniform scaling be $(w_{2_x}, w_{2_y}, 0)$ and $(w_{3_x}, w_{3_y}, w_{3_z})$, respectively. The shear component can be computed by solving the linear system of equations: $A[w_2 \;\; w_3] = [(1\ 1\ 0)^T \;\; (0\ 0\ 1)^T]$. The solution is uniquely given by,

$$\alpha_1 = \frac{1 - w_{2_x}}{w_{2_y}}, \; \alpha_2 = \frac{1}{w_{2_y}},$$

$$\alpha_3 = -\frac{(1 - w_{2_x})w_{3_y} + w_{3_x} w_{2_y}}{w_{2_y} w_{3_z}}, \tag{A.1}$$

$$\alpha_4 = -\frac{w_{3_y}}{w_{2_y} w_{3_z}}, \; \alpha_5 = \frac{1}{w_{3_z}}. \tag{A.2}$$

Note that since the tetrahedron has been assumed to be non-degenerate and has the same orientation as the canonical tetrahedron, $w_{2_y} > 0$ and $w_{3_z} > 0$. Thus, the orientation-preserving affine transformation $\mathbb{T}$ can be decomposed into components (E, A, S) uniquely, and $\mathbb{T} = ASE$.

For the case where Δ_Y is not the canonical tetrahedron, let Δ represent the canonical tetrahedron. As discussed earlier, there exist unique orientation-preserving affine transformations $\mathbb{T}_X$ and $\mathbb{T}_Y$ such that $\mathbb{T}_X \Delta_X = \mathbb{T}_Y \Delta_Y = \Delta$. Since $\mathbb{T}\Delta_X = \Delta_Y$ is unique, $\mathbb{T} = (\mathbb{T}_Y)^{-1}\mathbb{T}_X$, and thus $\mathbb{T}$ also maps the tetrahedron $(\mathbb{T}_X)^{-1}\mathbb{T}_Y\Delta$ to Δ, thus getting us back to the previous case.

A.2.3 (Lemma 2.4.1 - Steady Interpolation)

In the case where the affine transformation consists of a single component, from Equation (2.2) and the identity $\exp(t \log M) = \exp(\log(M^t)) = M^t$, it is clear that $\mathbb{T}_t = \mathbb{T}^t$. When $\mathbb{T} = AS$ for some $A \in G_A$ and $s \in G_S$, since A and S commute,

$$\mathbb{T}_t = \exp(t \log A) \exp(t \log S) = A^t S^t = (AS)^t$$

Hence $\mathbb{T}_t = \mathbb{T}^t$.

A.2.4 (Lemma 2.4.2 - Volume Preservation)

Let $\det(\mathbb{T}) = 1$. Since $\mathbb{T} = ASE$, we have $\det(ASE) = \det(A)\det(S)\det(E) = 1$. Using the fact that $\det(E) = \det(R) = 1$, $\det(S) = s^3$ and $\det(A) = \alpha_2\alpha_5$, we get $\alpha_2\alpha_5 s^3 = 1$. For any $t \in [0, 1]$,

$$\begin{aligned}
\det(\mathbb{T}_t) &= \det\left(\exp(t \log A) \exp(t \log S) \exp(t \log E)\right) \\
&= \det(\exp(t \log A)) \det(\exp(t \log S)) \det(\exp(t \log E)) \\
&= \exp(t(\log \alpha_2 + \log \alpha_5)) \exp(t \log s^3) \\
&= (\alpha_2\alpha_5 s^3)^t = 1,
\end{aligned}$$

where the penultimate equation is obtained using the relation
$\det(\exp(Q)) = \exp(tr(Q))$, with $tr(\cdot)$ denoting the trace operator, the form of $\log A$ given in Appendix A.3, and the fact that $tr(t \log E) = 0$.

A.2.5 (Lemma 2.4.3 - Monotonic variation of volume)

In order to show monotonicity of the change of volume effected by the interpolated transformations, it is enough to show that the sign of $\frac{d}{dt}\det(\mathbb{T}_t), 0 < t < 1$, remains constant. Using the relation $\det(\mathbb{T}_t) = (\alpha_2\alpha_5 s^3)^t$ (derived in Lemma 2), we get

$$\begin{aligned}\frac{d}{dt}\det(\mathbb{T}_t) &= \frac{d}{dt}(\alpha_2\alpha_5 s^3)^t, 0 < t < 1\\ &= (\alpha_2\alpha_5 s^3)^t \log(\alpha_2\alpha_5 s^3).\end{aligned}$$

Since α_2, α_5, s and t are positive real numbers and $\det(\mathbb{T}) = \alpha_2\alpha_5 s^3$, we get $sign(\frac{d}{dt}\det(\mathbb{T}_t)) = sign(\log(\alpha_2\alpha_5 s^3))$
$= sign(\log(\det(\mathbb{T})))$. Thus $\forall t \in (0,1)$, $\frac{d}{dt}\det(\mathbb{T}_t) > 0 (< 0)$ if $det(\mathbb{T}) > 1 (< 1)$.

A.2.6 (Lemma 2.4.4 - Isometric Transformations)

Since $\mathbb{T}$ is an isometry, it is a rigid transformation, i.e. $\mathbb{T} = E \in SE(3)$. The rigid transformation consists of a rotation $R \in SO(3)$ and a translation vector $d \in \mathbb{R}^3$. The interpolated transformation $\mathbb{T}^t, t \in [0,1]$ computed using Equation (2.2) is also a rigid transformation denoted by $\mathbb{T}^t = E^t \in SE(3)$ with a corresponding rotation R^t and translation vector $d^t \in \mathbb{R}^3$. It is easy to see that $||\mathbb{T}^t(x-y)|| = ||R^t(x-y)|| = ||x-y||$, proving that $\mathbb{T}^t$ is also an isometry, as required.

A.2.7 (Lemma 2.4.5 - Reversibility)

Let the Lie group representations of the tetrahedrons $\Delta_p, \Delta_q, \Delta_p^q(t)$ be $p, q, r_p^q(t)$, respectively. Then
$r_p^q(t) = p\exp(t\log(p^{-1}q))$, and

$$\begin{aligned}r_q^p(1-t) &= q\exp\Big((1-t)\log(q^{-1}p)\Big)\\ &= q\exp\Big(\log(q^{-1}p)\Big)\exp\Big(-t\log(q^{-1}p)\Big)\\ &= p\exp\Big(t\log(p^{-1}q)\Big) = r_p^q(t).\end{aligned}$$

Thus, our algorithm is reversible.

A.2.8 (Theorem 2.5.1 - Invariance to Canonical tetrahedron)

Let Δ_p, Δ_q be the tetrahedrons to be interpolated, and let $\Delta, \Delta_1, \Delta_2$ be the original and two arbitrary canonical tetrahedrons.

Let (E_1, A_1, s_1) represent tetrahedron Δ_1 with respect to Δ_2, and let (E_2, A_2, s_2) represent tetrahedron Δ_2 with respect to Δ_1. Let $\Delta_{c_i}, i = 1, 2$ be the interpolated tetrahedrons at some time t obtained when using canonical tetrahedrons $\Delta_i, i = 1, 2$, with Lie group representations $(E_{c_i}, A_{c_i}, s_{c_i}), i = 1, 2$, respectively. We now prove that the two representations $(E_{c_i}, A_{c_i}, s_{c_i})$, $i = 1, 2$ represent the same tetrahedron. In the proposed framework (refer to Equation (2.4)), the components $E_{c_i}, A_{c_i}, s_{c_i}, i = 1, 2$ are given as:

$$\begin{aligned} E_{c_i} &= E_{p_i} \exp(t \log(E_{p_i}^{-1} E_{q_i})), \\ A_{c_i} &= A_{p_i} \exp(t \log(A_{p_i}^{-1} A_{q_i})), \\ s_{c_i} &= s_{p_i} \exp(t \log(s_{p_i}^{-1} s_{q_i})). \end{aligned} \tag{A.3}$$

Since the rigid transformation component does not depend on the choice of canonical tetrahedron, $E_{p_1} = E_{p_2} = E_p$ and $E_{q_1} = E_{q_2} = E_q$. Then, one can conclude that $E_{c_1} = E_{c_2}$. Let l_p, l_q, l_1, l_2 denote the lengths of the first edge vectors of tetrahedrons $\Delta_p, \Delta_q, \Delta_1$ and Δ_2 respectively. Then, $s_{p_1}^{-1} s_{q_1} = \frac{l_p}{l_1} \frac{l_1}{l_q} = \frac{l_p}{l_2} \frac{l_2}{l_q} = s_{p_2}^{-1} s_{q_2}$.

Similarly by working out the shear matrices $A_{p_i}, i = 1, 2$ and $A_{q_i}, i = 1, 2$ in general, it can be shown that $(A_{p_i})^{-1} A_{q_i}, i = 1, 2$ both depend only on the coordinates of the tetrahedrons Δ_p and Δ_q, and hence are equal. The above facts show that the tangent vector at identity between Lie group representations of tetrahedrons Δ_p and Δ_q is invariant to the choice of canonical tetrahedron.

The transformations between Δ_p and Δ_1, Δ_2 are

$$\begin{aligned} (E_1^{-1} A_{p_1} S_{p_1} E_p) \Delta_p &= \Delta_1 \\ (E_2^{-1} A_{p_2} S_{p_2} E_p) \Delta_p &= \Delta_2, \end{aligned}$$

respectively. Using

$$(E_2^{-1} A_1 S_1 E_1)\Delta_1 = \Delta_2, \tag{A.4}$$

in the equation above gives us

$$A_1 S_1 A_{p_1} S_{p_1} = A_{p_2} S_{p_2}.$$

Similarly, the transformations between the two interpolated tetrahedrons and their respective canonical tetrahedrons are

$$(E_1^{-1} A_{c_1} S_{c_1} E_{c_1})^{-1}\Delta_1 = \Delta_{c_1} \tag{A.5}$$

$$(E_2^{-1} A_{c_2} S_{c_2} E_{c_2})^{-1}\Delta_2 = \Delta_{c_2}. \tag{A.6}$$

Substituting Equation (A.4) in Equation (A.6) above yields

$$(E_1^{-1} S_1^{-1} A_1^{-1} A_{c_2} S_{c_2} E_{c_2})^{-1}\Delta_1 = \Delta_{c_2}. \tag{A.7}$$

Since $E_{c_1} = E_{c_2}$, in order to prove $\Delta_{c_1} = \Delta_{c_2}$, it remains to show that

$$A_1 S_1 A_{c_1} S_{c_1} = A_{c_2} S_{c_2}. \tag{A.8}$$

Using the definitions of the transformations from Equations (A.3) on the left hand side, and the fact that the diagonal scaling matrices S commute with the shear matrices A (both embedded in 4×4 matrices), we get the desired result:

$$\begin{aligned}
& A_1 S_1 A_{c_1} S_{c_1} \\
&= A_1 S_1 A_{p_1} \exp(t \log(A_{p_1}^{-1} A_{q_1}))\; S_{p_1} \exp(t \log(S_{p_1}^{-1} S_{q_1})) \\
&= A_1 S_1 A_{p_1} S_{p_1} \exp(t \log(A_{p_1}^{-1} A_{q_1})) \exp(t \log(S_{p_1}^{-1} S_{q_1})) \\
&= A_{p_2} S_{p_2} \exp(t \log(A_{p_1}^{-1} A_{q_1})) \exp(t \log(S_{p_1}^{-1} S_{q_1})) \\
&= A_{p_2} \exp(t \log(A_{p_2}^{-1} A_{q_2}))\; S_{p_2} \exp(t \log(S_{p_2}^{-1} S_{q_2})) \\
&= A_{c_2} S_{c_2}
\end{aligned}$$

Hence both representations $(E_{c_1}, A_{c_1}, s_{c_1})$ and $(E_{c_2}, A_{c_2}, s_{c_2})$ are representation of the same tetrahedron and the interpolation process remains invariant to the role of canonical tetrahedron.

A.3 Matrix Exponential and Logarithm maps for $SE(3), G_A$ and G_S

This section contains formulas for Exponential and Logarithm maps for Lie groups $SE(3), G_A$ and G_S.

A.3.1 Exponential map on $SE(3)$

Let $A = \begin{bmatrix} \Omega & t \\ 0_3^T & 0 \end{bmatrix} \in \mathfrak{se}(3)$. If $||\Omega||_F = 0$ then $\exp(A) = \begin{bmatrix} I_3 & t \\ 0_3^T & 1 \end{bmatrix}$, else $\exp(A) = \begin{bmatrix} \exp(\Omega) & Vt \\ 0_3^T & 1 \end{bmatrix}$, where $\exp(\Omega) = I_3 + \frac{\sin\theta}{\theta}\Omega + \frac{(1-\cos\theta)}{\theta^2}\Omega^2$ and $V = I_3 + \frac{(1-\cos\theta)}{\theta^2}\Omega + \frac{(\theta-\sin\theta)}{\theta^3}\Omega^3$, with $\theta = \sqrt{-\frac{1}{2}Tr(\Omega^2)}$.

A.3.2 Logarithm map on $SE(3)$

Let $A = \begin{bmatrix} R & t \\ 0_3^T & 1 \end{bmatrix} \in SE(3)$. $\log(A) = \begin{bmatrix} \log(R) & V^{-1}t \\ 0_3^T & 0 \end{bmatrix}$, where $\log(R) = \frac{\theta}{2sin\theta}\left(R - R^T\right)$ and $V = I_3 + \frac{(1-\cos\theta)}{\theta^2}R + \frac{(\theta-\sin\theta)}{\theta^3}R^3$, with $\theta = acos\left(\frac{Tr(R)-1}{2}\right)$.

A.3.3 Exponential map on G_A

Let $A = \begin{bmatrix} 0 & \bar{\alpha_1} & \bar{\alpha_3} \\ 0 & \bar{\alpha_2} & \bar{\alpha_4} \\ 0 & 0 & \bar{\alpha_5} \end{bmatrix} \in \mathfrak{g}_A$.

$$Exp(A) = \begin{cases} \begin{bmatrix} 1 & \bar{\alpha_1} & \dfrac{\bar{\alpha_1}\bar{\alpha_4}\exp(\bar{\alpha_5}) + \bar{\alpha_3}\bar{\alpha_5}\exp(\bar{\alpha_5}) - \bar{\alpha_1}\bar{\alpha_4}\bar{\alpha_5} - \bar{\alpha_1}\bar{\alpha_4} - \bar{\alpha_3}\bar{\alpha_5}}{\bar{\alpha_5}^2} \\ 0 & 1 & \dfrac{\bar{\alpha_4}(\exp(\bar{\alpha_5}) - 1)}{\bar{\alpha_5}} \\ 0 & 0 & \exp(\bar{\alpha_5}) \end{bmatrix}, \text{ if } \bar{\alpha_2} = 0 \text{ and } \bar{\alpha_5} \neq 0 \\ \begin{bmatrix} 1 & \dfrac{\bar{\alpha_1}(exp(\bar{\alpha_2}) - 1)}{\bar{\alpha_2}} & \dfrac{\bar{\alpha_1}\bar{\alpha_4}\exp(\bar{\alpha_2}) - \bar{\alpha_1}(1 + \bar{\alpha_2})\bar{\alpha_4} + \bar{\alpha_2}^2\bar{\alpha_3}}{\bar{\alpha_2}^2} \\ 0 & \exp(\bar{\alpha_2}) & \dfrac{\bar{\alpha_4}(\exp(\bar{\alpha_2}) - 1)}{\bar{\alpha_2}} \\ 0 & 0 & 1 \end{bmatrix}, \text{ if } \bar{\alpha_5} = 0 \text{ and } \bar{\alpha_2} \neq 0 \\ \begin{bmatrix} 1 & \bar{\alpha_1} & \bar{\alpha_3} + \dfrac{\bar{\alpha_1}\bar{\alpha_4}}{2} \\ 0 & 1 & \bar{\alpha_4} \\ 0 & 0 & 1 \end{bmatrix}, \text{ if } \bar{\alpha_2} = 0 \text{ and } \bar{\alpha_5} = 0 \\ \begin{bmatrix} 1 & \dfrac{\bar{\alpha_1}(\exp(\bar{\alpha_2}) - 1)}{\bar{\alpha_2}} & \dfrac{((\bar{\alpha_1}\bar{\alpha_4} + \bar{\alpha_3})\bar{\alpha_2} - \bar{\alpha_1}\bar{\alpha_4})\exp(\bar{\alpha_2}) + \bar{\alpha_1}\bar{\alpha_4} - \bar{\alpha_2}\bar{\alpha_3}}{\bar{\alpha_2}^2} \\ 0 & \exp(\bar{\alpha_2}) & \bar{\alpha_4}\exp(\bar{\alpha_2}) \\ 0 & 0 & \exp(\bar{\alpha_2}) \end{bmatrix}, \text{ if } \bar{\alpha_2} = \bar{\alpha_5} \neq 0 \\ \begin{bmatrix} 1 & \dfrac{\bar{\alpha_1}(\exp(\bar{\alpha_2}) - 1)}{\bar{\alpha_2}} & \dfrac{(-\bar{\alpha_2}(\bar{\alpha_1}\bar{\alpha_4} - \bar{\alpha_2}\bar{\alpha_3} + \bar{\alpha_3}\bar{\alpha_5})exp(\bar{\alpha_5}) + \bar{\alpha_1}\bar{\alpha_4}\bar{\alpha_5}\exp(\bar{\alpha_2}) + (\bar{\alpha_2} - \bar{\alpha_5})(\bar{\alpha_1}\bar{\alpha_4} - \bar{\alpha_2}\bar{\alpha_3}))}{(\bar{\alpha_2}\bar{\alpha_5}(\bar{\alpha_2} - \bar{\alpha_5}))} \\ 0 & \exp(\bar{\alpha_2}) & \bar{\alpha_4}\dfrac{(\exp(\bar{\alpha_2}) - \exp(\bar{\alpha_5}))}{(\bar{\alpha_2} - \bar{\alpha_5})} \\ 0 & 0 & \exp(\bar{\alpha_5}) \end{bmatrix}, \text{ otherwise.} \end{cases}$$

A.3.4 Logarithm map on G_A

Let $A = \begin{bmatrix}\begin{bmatrix}1 & \alpha_1 & \alpha_3\\ 0 & \alpha_2 & \alpha_4\\ 0 & 0 & \alpha_5\end{bmatrix}\end{bmatrix} \in G_A.$

$$log(A) = \begin{cases}
\begin{bmatrix}0 & \alpha_1 & \frac{\log(\alpha_5)(\alpha_3\alpha_5 - \alpha_3 + \alpha_1\alpha_4)}{(\alpha_5-1)^2} - \frac{\alpha_1\alpha_4}{(\alpha_5-1)}\\ 0 & 0 & \frac{\alpha_4\log(\alpha_5)}{(\alpha_5-1)}\\ 0 & 0 & \log(\alpha_5)\end{bmatrix}, & \text{if } \alpha_2 = 1 \text{ and } \alpha_5 \neq 1\\
\begin{bmatrix}0 & \frac{\alpha_1\log(\alpha_2)}{\alpha_2-1} & \frac{\alpha_1\alpha_4\log(\alpha_2)}{(\alpha_2-1)^2} - \frac{(\alpha_3 - \alpha_2\alpha_3 + \alpha_1\alpha_4)}{(\alpha_2-1)}\\ 0 & \log(\alpha_2) & \frac{\alpha_4\log(\alpha_2)}{(\alpha_2-1)}\\ 0 & 0 & 0\end{bmatrix}, & \text{if } \alpha_5 = 1 \text{ and } \alpha_2 \neq 1\\
\begin{bmatrix}0 & \alpha_1 & \alpha_3 - \frac{(\alpha_1\alpha_4)}{2}\\ 0 & 0 & \alpha_4\\ 0 & 0 & 0\end{bmatrix}, & \text{if } \alpha_2 = \alpha_5 = 1\\
\begin{bmatrix}0 & \frac{\alpha_1\log(\alpha_2)}{\alpha_2-1} & \frac{\alpha_3\log(\alpha_2)}{(\alpha_2-1)} + \frac{\alpha_1\alpha_4}{\alpha_2(\alpha_2-1)} - \frac{\alpha_1\alpha_4\log(\alpha_2)}{(\alpha_2-1)^2}\\ 0 & \log(\alpha_2) & \frac{\alpha_4}{\alpha_2}\\ 0 & 0 & \log(\alpha_2)\end{bmatrix}, & \text{if } \alpha_2 = \alpha_5 \neq 1\\
\begin{bmatrix}0 & \frac{\alpha_1\log(\alpha_2)}{\alpha_2-1} & \frac{\alpha_1\alpha_4\log(\alpha_2)}{(\alpha_2-\alpha_5)(\alpha_2-1)} - \frac{(\alpha_1\alpha_4 - \alpha_2\alpha_3 + \alpha_3\alpha_5)\log(\alpha_5)}{(\alpha_2-\alpha_5)(\alpha_5-1)}\\ 0 & \log(\alpha_2) & \frac{\alpha_4\log(\alpha_2)}{(\alpha_2-\alpha_5)} - \frac{\alpha_4\log(\alpha_5)}{(\alpha_2-\alpha_5)}\\ 0 & 0 & \log(\alpha_5)\end{bmatrix}, & \text{otherwise}
\end{cases}$$

A.3.5 Exponential and Logarithm map on G_S

Note that $G_S = \mathbb{R}^+$ and $\mathfrak{g}_S = \mathbb{R}$, therefore exponential and logarithm maps coincide with usual scalar exponential and logarithm maps.

A.3.6 Note

Exponential and Logarithm maps for A and s are converted into 4×4 matrices by embedding them in the top-left 3×3 block of the 4×4 identity matrix, whenever required to be composed with other transformations.