九州大学学術情報リポジトリ
Kyushu University Institutional Repository

A Study on 2D Shape Interpolation Using Affine Maps

松下, 昂平

https://doi.org/10.15017/1500513

出版情報：九州大学, 2014, 博士（機能数理学）, 課程博士
バージョン：
権利関係：全文ファイル公表済

A Study on 2D Shape Interpolation Using Affine Maps

A Dissertation Submitted to the Graduate School of Mathematics
in Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Functional Mathematics
Kyushu University

By

Kohei Matsushita

Supervisor: Associate Professor Yoshihiro Mizoguchi

2015

Abstract

In computer graphics (CG), shape interpolation techniques are widely used for many applications. In several interpolation techniques, As-Rigid-As Possible Shape Interpolation (ARAP) proposed by Alexa is known as a interpolation method which preserve rigidity of configurations. ARAP is suitable for character animation from this characteristics, and various methods have been proposed. There are many algorithm of shape interpolations, however, it is not enough to discuss a goodness of a given shape interpolation. It is an important issue to consider mathematical characterization of these image representations and construct efficient and tractable mathematical models. Specifically, it is one of our goals to define conditions mathematically and propose excellent interpolation methods by analyzing existing interpolation techniques. We also investigate fast algorithm of shape interpolation method.

In Chapter 2, we discuss a "goodness" of a two-dimensional shape interpolation. Although there are many interpolation methods, it appears that mathematical discussion for "goodness" of interpolations are not much. Our motivation is to construct a useful formulation for 2D shape interpolations. For a realization of a "good" interpolation, we examined and observed properties of interpolations and transformation matrices for local and global interpolations. 1) For a local interpolation, we define two "goodness" of interpolations in considering area and norm transforms. We obtained the mathematical conditions of the goodness for the known method according to our definitions. 2) For a global interpolation, we formulated an energy function which use different coordinate for each local transformation. As a result, the optimal solution of the energy function does depend on a choice of coordinates. To explain this fact, we show a example. We calculated a locus of the origin of the coordinate which attains the same value of energy functions.

In Chapter 3, we propose a faster algorithm of two-dimensional shape interpolation. It is often useful to compute a lot of matrix exponentials in computer graphics. The exponential of a matrix is used for the smooth deformation of 2D or 3D meshed CG objects [7, 2, 12, 1, 6]. Hence, we need to compute a large number of the exponentials of 3×3 rotational matrices and 3×3 real symmetric matrices. For rotational matrices, Rodrigues' formula [4] is known to compute their exponentials. We investigated the polynomial methods introduced by Moler and Van Loan [9] to compute an exponential of 3×3 real symmetric matrices, and we introduce an algorithm for eigenvalues of 3×3 real symmetric matrices. We introduce a simple formula for

the matrix exponential of a 3×3 real symmetric matrix using a formula in [6] and Viète's Formula. Since our matrix exponential algorithm does not use eigenvectors, we are able to reduce the computational cost using a fast eigenvalue computation algorithm. Then, we incorporated our implementation into a shape deforming tool developed in [6]. As a result, we achieved a notable performance improvement. In fact we show our algorithms for matrix exponentials is about 4.2 times faster than a standard algorithm for given 3×3 real symmetric matrices. For the deformation of a CG model, our algorithm was about 1.2 times faster than a standard algorithm.

Acknowledgments

I would like to express my deepest gratitude to Professor Yoshihiro Mizoguchi for his continuous support, useful discussion, suggestion and encouragement. I am also grateful to K. Anjyo, H. Ochiai, S. Kaji for their helpful comments. I would like to thank H. Hamada, S. Hirose, S. Yokoyama for useful discussions and comments.

Finally, I would like to give my special thanks to my parents for their love and support throughout my life.

Contents

Chapter 1

Introduction

2D shape interpolation is widely used in computer graphics (CG) . To generate shape interpolations, Shoemake and Duff[10, 11] suggested a way using transformation matrix. For a technique using transformation matrix, there is a technique called morphing. This technique is introduced by Beier and Neely [3] in 1992. For given two images, we define a method to transfer one image to another one smoothly. This technique is used for many visual effects. In 2000, Alexa and Xu suggested new algorithm preserving rigidity[2, 14] for algorithm of this interpolation. These algorithm consists of local interpolations and a global interpolation. In 2D shape interpolation, the local interpolation means an interpolation between one triangle mesh and another mesh. This deformation from one mesh to another one is decided by a unique affine transformation, and this transformation is described by a matrix representation. To generate intermediate shapes, we vary this affine matrix over time. Then, these local interpolations are composed into a global interpolation which minimizes the energy function defined on overall local interpolations. There are many algorithm of shape interpolations, however, it is not enough to discuss a goodness of a given shape interpolation. Even though "goodness" is a kind of sense, but we are trying to formalize these sense using mathematical formulas.

In chapter 2, we propose a "good" interpolation for a local interpolation and a "good" composition for a global interpolation. We start by determining a "good" interpolation between one triangle mesh and another one. Then, we find conditional equations of a good local interpolation. Next, for a global interpolation, we investigate a "good" composition. To construct a global interpolation, we solve a minimization problem of a given energy function. In this energy function, every triangle mesh has the same origin.

Generally, this origin is assumed the center of each triangle. On the other hand, for convenience, we choose a same origin for all local interpolations in the formula of an energy function of a global interpolation. The first problem we consider is that this small gap may cause different interpolations, and we would like to clarify conditions of an initial and target configuration which cause a critical errors for composing a global interpolation. We choose a simple figure with two triangles which we call them the first triangle and the second triangle. We consider the values of the global energy function when we fix the origin of the first triangle to zero and vary the origin of the second triangle. Then, we can show contours of the origin of the second triangle are quadratic curves. This result shows an answer of our first problem. The difference of origins cause a different global interpolations.

In chapter 3, we investigate a fast algorithm for matrix exponential computations which have been used for many applications in computer graphics. For example, *mesh-based inverse kinematics* provides a tool that simplifies posing task [12]. In this algorithm, a matrix exponential is used to calculate a deformation gradient of triangular meshes. Alexa investigated a new interpolation between transformations using operations, addition and scalar multiplication, which create weighted combination of transformations and interpolation [1]. For 2D shape interpolation and deformation, there are several algorithms that preserve rigidity [2, 5]. Kaji et al. improved those algorithms by using matrix exponentials [7]. Matrix exponentials are also used for smooth deformations of 2D or 3D meshed CG objects. Our goal is to provide a fast computation of matrix exponentials. Our motivation is to improve the performance of many applications using exponentials of 3×3 rotational matrices and 3×3 real symmetric matrices introduced in [12, 7, 6]. For the rotational matrices, Rodrigues' formula [4] is known to compute their exponentials. We consider an improvement of an exponential of a 3×3 real symmetric matrix rather than a rotation matrix. In general, there are many approaches for computing $n \times n$ matrix exponentials. Moler and Van Loan provided several methods to compute the matrix exponentials [9]. We focus on the matrix exponential of a special case such as 3×3 real matrices used for the affine transformations [6, 8]. First, we investigate the spectral decomposition method in [9] focusing on 3×3 real symmetric matrices. Our algorithm needs only eigenvalues of a given matrix. We note a method using diagonalization have to compute eigenvectors in addition to eigenvalues. Hence, our algorithm can compute matrix exponentials efficiently. In addition, we consider a faster algorithm for computing the eigenvalues of a 3×3 real symmetric matrix. For a given matrix, we need to solve the characteristic equation of the matrix to calculate its eigenvalues.

We use Viète's formula to compute the eigenvalues of a 3×3 real symmetric matrix. As a result, we introduce simple formulas of eigenvalues just using the trace and determinant of a given matrix. To evaluate the performance of our algorithm, we compare the average runtimes for matrix exponentials and eigenvalues. We show some experimental results to show the advantage of our algorithm.

Chapter 2

A "Good" Interpolation

In this chapter, we investigate useful formulations for 2D shape interpolations. For a local interpolation, we define two "good" interpolations from the view points of the changes of "area" and "norm". We consider conditions for a given initial and target configuration where the interpolation of this configuration is "good" in our sense. Next, we define a global interpolation as a minimum of an energy function defined by a sum of energies of local interpolations. Then, we consider the values of the global energy function when we fix the origin of triangles.

2.1 2D shape interpolation

For 2D shape interpolation, we follow the As-Rigid-As-Possible Shape Interpolation (ARAP) algorithm [2]. Figure 2.1 shows a example of 2D shape interpolation using the ARAP method. For a given input data which consists of a source and target configuration, this algorithm generate smooth intermediate configurations between this source and target configuration. This algorithm preserves rigidity of input configurations, and ARAP is useful for character animations. The ARAP algorithm consists of the following 3 steps.

1. **Compatible triangulation**: Two input shape data (source and target) are triangulated, and the meshes between the source and the target have a one-to-one correspondence.

2. **Local interpolation**: Let the source vertices be $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ and the target vertices be $Q = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3)$, where vertices with the same

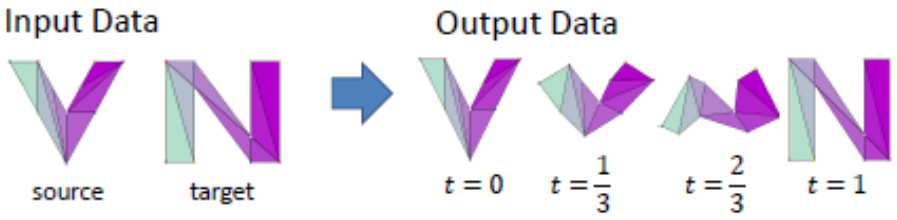

Figure 2.1: Example of 2D shape interpolation. Left: input configurations. Right: output configurations. They are intermediate configurations between an initial (source) and target configuration.

index correspond. We get an affine map denoted by A. In this paper, $\mathrm{Aff}^+(2)$ is defined by a set of affine maps which have positive determinant. Then, we construct a homotopy between the 2×2 identity matrix and $\boldsymbol{A}$. This homotopy $\boldsymbol{A}(t)$ is parameterized by $t \in \mathbb{R}$. $\boldsymbol{A}(t)$ satisfies $\boldsymbol{A}(0) = I$ and $\boldsymbol{A}(1) = A$. We follow [6], that is a local interpolation using the polar decomposition.

$$\boldsymbol{A}^E(t) := \boldsymbol{R}_\theta^t \boldsymbol{S}^t.$$

3. **Global interpolation**: Since most of vertices corresponding to more than one triangles, a mapping of all vertices could not (in general) be conforming with all the individual ideal transformations. The collection of local interpolations does not directly assign an ideal global interpolation. To construct a global interpolation, we use the minimizer of the energy function:

$$E_t(\boldsymbol{B}) := \sum_{i=1}^{m} E_i(\boldsymbol{A}_i(t), \boldsymbol{B}_(t)) + C(\boldsymbol{v}_1(t), \ldots, \boldsymbol{v}_n(t)),$$

where C is a constant function.

In this chapter, we assume that input configurations are already triangulated compatibly. For the next section, we denote a “good” interpolation and consider conditions.

2.2 A “good” interpolation

In this section, we consider conditions for a given source and target configuration where a local interpolation of this configuration is “good” in our

sense. We define two“good” interpolations from the viewpoints of the change of “area” and “norm”.

Definition 2.2.1. Let $\boldsymbol{A}(t)$ be a given local interpolation. $\boldsymbol{A}(t)$ is called a “good” interpolation for the change of area if $\det \boldsymbol{A}(t)$ is a monotone function and $\det \boldsymbol{A}(t) > 0$ for $0 \leq t \leq 1$.

Definition 2.2.2. Let $\boldsymbol{A}(t)$ be a given local interpolation, $\boldsymbol{v} \in \mathbb{R}^2$ and $\|\boldsymbol{v}\| = 1$. $\boldsymbol{A}(t)$ is called a “good” interpolation for the change of norm if the norm $\|\boldsymbol{A}(t)\boldsymbol{v}\|$ is a monotone function over $0 \leq t \leq 1$.

As a result, we obtain conditions of “goodness” for each known methods such as linear interpolation, SVD interpolation and polar decomposition interpolation.

2.2.1 Linear interpolation method

A linear interpolation method is a simple way to construct a local interpolation between a given initial and target configuration. This interpolation is defined as follows.

Definition 2.2.3. Let $\boldsymbol{A} \in \mathrm{Aff}^+(2)$ and $\boldsymbol{I}$ the identity matrix. A linear interpolation $\boldsymbol{A}_l(t)$ is defined by

$$\boldsymbol{A}_l(t) = (1-t)\boldsymbol{I} + t\boldsymbol{A}.$$

In this method, we obtain a condition of “good” interpolation from the viewpoint of the change of “area”.

Theorem 2.2.1. A linear interpolation $\boldsymbol{A}_l(t)$ is a good interpolation for the change of area if and only if $\boldsymbol{A}_l(t)$ satisfies

$$(\mathrm{tr}\boldsymbol{A} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{A} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{A} < 2\det\boldsymbol{A}).$$

Proof. The determinant of $\boldsymbol{A}_l(t)$ is

$$\begin{aligned}
\det \boldsymbol{A}_l(t) &= \begin{vmatrix} 1-t+ta & tb \\ tc & 1-t+tc \end{vmatrix} \\
&= (1-t)^2 + (1-t)(ta+td) + t^2ad - t^2bc \\
&= t^2(1-a-d+ad-bc) + t(a+d-2) + 1 \\
&= (1-\mathrm{tr}\boldsymbol{A} + \det\boldsymbol{A})t^2 + (\mathrm{tr}\boldsymbol{A} - 2)t + 1.
\end{aligned}$$

We note $\det\boldsymbol{A}_l(0) = 1$, $\det\boldsymbol{A}_l(1) = \det\boldsymbol{A} > 0$. The derivative of $\det\boldsymbol{A}_l(t)$ with respect to t is,

$$\frac{d}{dt}\det\boldsymbol{A}_l(t) = 2(1 - \mathrm{tr}\boldsymbol{A} + \det\boldsymbol{A})t + \mathrm{tr}\boldsymbol{A} - 2$$

(i) In the case of $1-\mathrm{tr}\boldsymbol{A}+\det\boldsymbol{A} = 0$, $\det\boldsymbol{A}_l(t)$ is a linear function. Then, $\det\boldsymbol{A}_l(t)$ is a monotone function and $\det\boldsymbol{A}_l(t) > 0$ for $0 \leq t \leq 1$. Therefore, $\mathrm{tr}\boldsymbol{A} = \det\boldsymbol{A} + 1$ is a condition that $\det\boldsymbol{A}_l(t)$ is a monotone function.

(ii) Next, in the case of $1 - \mathrm{tr}\boldsymbol{A} + \det\boldsymbol{A} > 0$, assume $\frac{d}{dt}\det\boldsymbol{A}_l(t_0) = 0$. Then, we have

$$t_0 = \frac{2 - \mathrm{tr}\boldsymbol{A}}{2(1 - \mathrm{tr}\boldsymbol{A} + \det\boldsymbol{A})}.$$

Since $0 \leq t \leq 1$, we have

$$(t_0 \leq 0) \vee (t_0 \geq 1) \Leftrightarrow (2 \leq \mathrm{tr}\boldsymbol{A}) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{A}).$$

In the case of $\det\boldsymbol{A} \leq 1$, we have $\mathrm{tr}\boldsymbol{A} < 2$ since $\det\boldsymbol{A} + 1 \leq 2$. Therefore, $2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{A} < 2$ is a condition that $\det\boldsymbol{A}_l(t)$ is a monotone function. In the case of $\det\boldsymbol{A} > 1$, we have $2\det\boldsymbol{A} > \mathrm{tr}\boldsymbol{A}$ since $2\det\boldsymbol{A} > \det\boldsymbol{A}+1$. Therefore, $2 \leq \mathrm{tr}\boldsymbol{A} < 2\det\boldsymbol{A}$ is a condition that $\det\boldsymbol{A}_l(t)$ is a monotone function.

(iii) Finally, in the case of $1-\mathrm{tr}\boldsymbol{A}+\det\boldsymbol{A} < 0$, assume $\frac{d}{dt}\det\boldsymbol{A}_l(t_0) = 0$. In the same way, we have

$$t_0 = \frac{2 - \mathrm{tr}\boldsymbol{A}}{2(1 - \mathrm{tr}\boldsymbol{A} + \det\boldsymbol{A})}.$$

Since $0 \leq t \leq 1$, we have

$$(t_0 \leq 0) \vee (t_0 \geq 1) \Leftrightarrow (2 \leq \mathrm{tr}\boldsymbol{A}) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{A}).$$

In the case of $\det\boldsymbol{A} \leq 1$, we have $2\det\boldsymbol{A} \leq \det\boldsymbol{A} + 1 < \mathrm{tr}\boldsymbol{A}$ since $(\det\boldsymbol{A} + 1) - 2\det\boldsymbol{A} = 1 - \det\boldsymbol{A} \geq 0$. Therefore, $\mathrm{tr}\boldsymbol{A} > \det\boldsymbol{A} + 1$ is a condition that $\det\boldsymbol{A}_l(t)$ is a monotone function. In the case of $\det\boldsymbol{A} > 1$, we have $2 < \det\boldsymbol{A} + 1 < \mathrm{tr}\boldsymbol{A}$ since $(\det\boldsymbol{A} + 1) - 2 = -1 + \det\boldsymbol{A} > 0$. Therefore, $\mathrm{tr}\boldsymbol{A} > \det\boldsymbol{A} + 1$ is a condition that $\det\boldsymbol{A}_l(t)$ is a monotone function.

According to (i), (ii) and (iii), we have

$$(\mathrm{tr}\boldsymbol{A} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{A} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{A} < 2\det\boldsymbol{A}).$$

□

2.2.2 SVD interpolation method

Let $\boldsymbol{A} \in \mathrm{Aff}^+(2)$ and $\boldsymbol{R}_\theta$ be a 2×2 rotation matrix for θ rotations. Then, $\boldsymbol{A}$ is decomposed into two rotation matrices and a diagonal matrix as below.

$$\boldsymbol{A} = \boldsymbol{R}_\alpha \boldsymbol{D} \boldsymbol{R}_\beta,$$

where $D = \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}$, $x > 0$, $y > 0$ and $\alpha, \beta \in [0, 2\pi)$. This decomposition is called as the singular value decomposition (SVD). By the SVD method, we define a SVD interpolation as follows.

Definition 2.2.4. Let $\boldsymbol{A} \in \mathrm{Aff}^+(2)$ and $\boldsymbol{I}$ the identity matrix. A SVD interpolation $\boldsymbol{A}_s(t)$ is defined by

$$\boldsymbol{A}_s(t) = \boldsymbol{R}_{t\alpha}((1-t)\boldsymbol{I} + t\boldsymbol{D})\boldsymbol{R}_{t\beta},$$

where $\boldsymbol{R}$ is a rotation matrix, and $\boldsymbol{D}$ is a diagonal matrix.

In this method, we obtain conditions of “good” interpolation from the viewpoints of the change of “area” and “norm”.

Theorem 2.2.2. $\boldsymbol{A}_s(t)$ is a good interpolation for the change of area if and only if $\boldsymbol{A}$ satisfies

$$(\mathrm{tr}\boldsymbol{D} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{D} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{D} < 2\det\boldsymbol{A}).$$

Proof. The determinant of $\boldsymbol{A}_s(t)$ is

$$\begin{aligned} \det \boldsymbol{A}_s(t) &= (\det \boldsymbol{R}_{t\alpha})(\det((1-t)\boldsymbol{I} + t\boldsymbol{D}))(\det \boldsymbol{R}_{t\beta}) \\ &= \det((1-t)\boldsymbol{I} + t\boldsymbol{D}). \end{aligned}$$

This corresponds to the case of the linear interpolation of $\boldsymbol{D}$. From Theorem 2.2.1, a condition of $\det \boldsymbol{A}_s(t)$ which satisfies a monotone and positive function in the range of $0 \leq t \leq 1$ is

$$(\mathrm{tr}\boldsymbol{D} \geq \det\boldsymbol{D} + 1) \vee (2\det\boldsymbol{D} \leq \mathrm{tr}\boldsymbol{D} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{D} < 2\det\boldsymbol{D}).$$

Then,

$$\det \boldsymbol{A} = (\det \boldsymbol{R}_\alpha)(\det \boldsymbol{D})(\det \boldsymbol{R}_\beta) = \det \boldsymbol{D}.$$

Therefore,

$$(\mathrm{tr}\boldsymbol{D} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{D} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{D} < 2\det\boldsymbol{A}).$$

□

Theorem 2.2.3. $\boldsymbol{A}_s(t)$ is a good interpolation for the change of norm if and only if $\boldsymbol{A}$ satisfies

$$((x \geq 1) \wedge (y \geq 1)) \vee ((0 < x \leq 1) \wedge (0 < y \leq 1)),$$

where, $\boldsymbol{D} = \begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}$, $x > 0,\ y > 0$.

Proof.

$$\begin{aligned} \|\boldsymbol{A}_s(t)\boldsymbol{v}\|^2 &= \|((1-t)\boldsymbol{I} + t\boldsymbol{D})\boldsymbol{v}\|^2 \\ &= ((x-1)t+1)^2\cos^2\theta + ((y-1)t+1)^2\sin^2\theta \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{1}{2}\frac{d}{dt}\|\boldsymbol{A}_s(t)\boldsymbol{v}\|^2 = ((x-1)^2\cos^2\theta + (y-1)^2\sin^2\theta)t \\ +(x-1)\cos^2\theta + (y-1)\sin^2\theta. \end{aligned}$$

For all $\theta \in [0, 2\pi]$, $\|\boldsymbol{A}_s(t)\boldsymbol{v}\|$ is a monotone function if

$$((x-1)\cos^2\theta + (y-1)\sin^2\theta \geq 0) \vee \left(\frac{d\|\boldsymbol{A}_s(t)\boldsymbol{v}\|^2}{dt}\ \Big|_{t=1} \leq 0\right)$$

For the equation $((x-1)\cos^2\theta + (y-1)\sin^2\theta \geq 0)$, we assume $x \geq y$. Then,

$$\begin{aligned} (x-1)\cos^2\theta + (y-1)\sin^2\theta &\geq (y-1)(\sin^2\theta + \cos^2\theta) \\ &= y - 1 \end{aligned}$$

Hence, $\|\boldsymbol{A}_s(t)\boldsymbol{v}\|$ is a monotone function if $y \geq 1$. In the same way, in the case of $y \geq x$, $\|\boldsymbol{A}_s(t)\boldsymbol{v}\|$ is a monotone function if $x \geq 1$.

Therefore

$$(x \geq 1) \wedge (y \geq 1).$$

We assume $x \geq y$.

$$\begin{aligned} \frac{1}{2}\frac{d\|\boldsymbol{A}_s(t)\boldsymbol{v}\|^2}{dt}\ \Big|_{t=1} &= (x-1)x\cos^2\theta + (y-1)y\sin^2\theta \\ &\leq (x-1)x(\cos^2\theta + \sin^2\theta) \\ &= (x-1)x \end{aligned}$$

Then, $\|\boldsymbol{A}_s(t)\boldsymbol{v}\|$ is a monotone function if $0 < x \leq 1$. In the case of $y \geq x$, $\|\boldsymbol{A}_s(t)\boldsymbol{v}\|$ is a monotone function if $0 < y \leq 1$ in the same way.

$$(0 < x \leq 1) \wedge (0 < y \leq 1).$$

Therefore,

$$((x \geq 1) \wedge (y \geq 1)) \vee ((0 < x \leq 1) \wedge (0 < y \leq 1)).$$

□

2.2.3 Polar decomposition interpolation method

From the SVD,

$$\boldsymbol{A} = \boldsymbol{R}_\alpha \boldsymbol{D} \boldsymbol{R}_\beta = (\boldsymbol{R}_\alpha \boldsymbol{R}_\beta)(\boldsymbol{R}_\beta^T \boldsymbol{D} \boldsymbol{R}_\beta).$$

We put $\boldsymbol{R}_\gamma = \boldsymbol{R}_\alpha \boldsymbol{R}_\beta$ and $\boldsymbol{S} = \boldsymbol{R}_\beta^T \boldsymbol{D} \boldsymbol{R}_\beta$. Then, $\boldsymbol{A}$ is decomposed into a rotation matrix and a positive definite symmetric matrix as below.

$$\boldsymbol{A} = \boldsymbol{R}_\gamma \boldsymbol{S},$$

where $\boldsymbol{S} = \begin{pmatrix} x & h \\ h & y \end{pmatrix}$, $x > 0$, $y > 0$ and $h \in \mathbb{R}$. This decomposition is called as the polar decomposition. By this decomposition, we define a polar decomposition interpolation as follows.

Definition 2.2.5. Let $\boldsymbol{A} \in \mathrm{Aff}^+(2)$ and $\boldsymbol{I}$ the identity matrix. We define a linear interpolation $\boldsymbol{A}_p(t)$ as follows:

$$\boldsymbol{A}_p(t) = \boldsymbol{R}_{t\gamma}((1-t)\boldsymbol{I} + t\boldsymbol{S}),$$

where $\boldsymbol{R}$ is a rotation matrix, and $\boldsymbol{S}$ is a positive definite symmetric matrix.

In this method, we obtain conditions of “good” interpolation from the viewpoints of the change of “area” and “norm”.

Theorem 2.2.4. $\boldsymbol{A}_p(t)$is a good interpolation for the change of area if and only if $\boldsymbol{A}$ satisfies

$$(\mathrm{tr}\boldsymbol{S} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{S} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{S} < 2\det\boldsymbol{A}).$$

where $\boldsymbol{S} = \begin{pmatrix} x & h \\ h & y \end{pmatrix}$, $x > 0$, $y > 0$, $h \in \mathbb{R}$.

Proof.

$$\begin{aligned}\det \boldsymbol{A}_p(t) &= (\det \boldsymbol{R}_{t\gamma})(\det((1-t)\boldsymbol{I} + t\boldsymbol{S})) \\ &= \det((1-t)\boldsymbol{I} + t\boldsymbol{S})\end{aligned}$$

This equation corresponds to the case of a linear interpolation of $\boldsymbol{S}$. From the result of Theorem 2.2.1,

$$(\mathrm{tr}\boldsymbol{S} \geq \det\boldsymbol{S} + 1) \vee (2\det\boldsymbol{S} \leq \mathrm{tr}\boldsymbol{S} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{S} < 2\det\boldsymbol{S}).$$

Then,

$$\det\boldsymbol{A} = (\det\boldsymbol{R}_\gamma)(\det\boldsymbol{S}) = \det\boldsymbol{S}$$

Therefore,

$$(\mathrm{tr}\boldsymbol{S} \geq \det\boldsymbol{A} + 1) \vee (2\det\boldsymbol{A} \leq \mathrm{tr}\boldsymbol{S} < 2) \vee (2 \leq \mathrm{tr}\boldsymbol{S} < 2\det\boldsymbol{A}).$$

□

Theorem 2.2.5. $\boldsymbol{A}_p(t)$ is a good interpolation for the change of norm if and only if $\boldsymbol{A}$ satisfies

$$((x-1)(y-1) \geq h^2) \wedge ((x+y \geq 2) \vee ((0 < x \leq 1) \wedge (0 < y \leq 1))).$$

where $\boldsymbol{S} = \begin{pmatrix} x & h \\ h & y \end{pmatrix},\ x > 0,\ y > 0,\ h \in \mathbb{R}$.

Proof.

$$\begin{aligned}\|\boldsymbol{A}_p(t)\boldsymbol{v}\|^2 &= \|((1-t)\boldsymbol{I} + t\boldsymbol{S})\boldsymbol{v}\|^2 \\ &= (((x-1)t+1)\cos\theta + th\sin\theta)^2 + (th\cos\theta + ((y-1)t+1)\sin\theta)^2\end{aligned}$$

Then,

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|\boldsymbol{A}_p(t)\boldsymbol{v}\|^2 = &(((x-1)\cos\theta + h\sin\theta)^2 + (h\cos\theta + (y-1)\sin\theta)^2)t \\ &+ (x-1)\cos^2\theta + h\sin 2\theta + (y-1)\sin^2\theta.\end{aligned}$$

For all $\theta \in [0, 2\pi]$,

$$(x-1)\cos^2\theta + h\sin 2\theta + (y-1)\sin^2\theta \geq 0 \tag{2.1}$$

or

$$\frac{d\|\boldsymbol{A}_p(t)\boldsymbol{v}\|^2}{dt}\bigg|_{t=1} \leq 0 \tag{2.2}$$

satisfies the condition of a monotone function.

For equation (2.1), we assume $x \geq y$.

$$\begin{aligned}
&(x-1)\cos^2\theta + h\sin 2\theta + (y-1)\sin^2\theta \\
&= (x-1)\frac{1+\cos 2\theta}{2} + (y-1)\frac{1-\cos 2\theta}{2} + h\sin 2\theta \\
&= \frac{x+y-2}{2} + \frac{(x-y)}{2}\cos 2\theta + h\sin 2\theta \\
&= \frac{x+y-2}{2} + \sqrt{\frac{(x-y)^2}{4} + h^2}\cos(2\theta+\alpha) \\
&\geq \frac{x+y-2}{2} - \sqrt{\frac{(x-y)^2}{4} + h^2}.
\end{aligned}$$

Then, $\|\boldsymbol{A}_p(t)\boldsymbol{v}\|$ is a monotone function if the following equation is satisfied:

$$\begin{aligned}
\frac{x+y-2}{2} \geq \sqrt{\frac{(x-y)^2}{4} + h^2} &\Leftrightarrow (x+y-2)^2 \geq (x-y)^2 + 4h^2 \\
&\Leftrightarrow 4xy - 4(x+y) + 4 \geq 4h^2 \\
&\Leftrightarrow (x-1)(y-1) \geq h^2.
\end{aligned}$$

Therefore,

$$((x-1)(y-1) \geq h^2) \wedge (x+y \geq 2).$$

For equation (2.2), we assume $x \geq y$.

$$\begin{aligned}
\frac{1}{2}\frac{d\|\boldsymbol{A}_p(t)\boldsymbol{v}\|^2}{dt}\bigg|_{t=1} &= (x-1)x\cos^2\theta + (y-1)y\sin^2\theta + h^2 + h(x+y-1)\sin 2\theta \\
&= (x-1)x\frac{1+\cos 2\theta}{2} + (y-1)y\frac{1-\cos 2\theta}{2} \\
&\quad + h^2 + h(x+y-1)\sin 2\theta \\
&= \frac{(x-1)x-(y-1)y}{2}\cos 2\theta + \frac{(x-1)x+(y-1)y}{2} \\
&\quad + h^2 + h(x+y-1)\sin 2\theta \\
&= \frac{(x-y)(x+y-1)}{2}\cos 2\theta + h(x+y-1)\sin 2\theta \\
&\quad + h^2 + \frac{(x-1)x+(y-1)y}{2} \\
&= (x+y-1)\left(\sqrt{\frac{(x-y)^2}{4}+h^2}\cos(2\theta+\alpha)\right) \\
&\quad + h^2 + \frac{(x-1)x+(y-1)y}{2} \\
&\leq |x+y-1|\sqrt{\frac{(x-y)^2}{4}+h^2} + h^2 + \frac{(x-1)x+(y-1)y}{2} \\
&\leq 0.
\end{aligned}$$

Then,

$$\begin{aligned}
&|x+y-1|\sqrt{\frac{(x-y)^2}{4}+h^2} \leq -h^2 - \frac{(x-1)x+(y-1)y}{2} \\
&\Leftrightarrow (x+y-1)^2\left(\frac{(x-y)^2}{4}+h^2\right) \leq \left(h^2 + \frac{(x-1)x+(y-1)y}{2}\right)^2 \\
&\Leftrightarrow h^4 + (x+y-2xy-1)h^2 + (x-1)(y-1)xy \geq 0 \\
&\Leftrightarrow (h^2-xy)(h^2+x+y-xy-1) \leq 0 \\
&\Leftrightarrow h^2 + x + y - xy - 1 \leq 0 \\
&\Leftrightarrow h^2 \leq xy - x - y + 1 \\
&\Leftrightarrow h^2 \leq (x-1)(y-1). \qquad (2.3)
\end{aligned}$$

In addition, we need

$$h^2 + \frac{(x-1)x+(y-1)y}{2} \leq 0 \Leftrightarrow h^2 \leq -\frac{(x-1)x+(y-1)y}{2}. \qquad (2.4)$$

If $x > 1$, $y \geq 1$ from (2.3). On the other hand, this condition does not satisfy the condition (2.4). In the same way, $y > 1$ does not satisfy the condition (2.3) and (2.4).

Therefore, we need

$$(0 < x \leq 1) \wedge (0 < y \leq 1). \tag{2.5}$$

Then,

$$\begin{aligned}(x-1)(y-1) + \frac{(x-1)x + (y-1)y}{2} &= xy - x - y + 1 + \frac{x^2 + y^2 - x - y}{2} \\ &= \frac{1}{2}(x^2 + y^2 - 3x - 3y + 2xy + 2) \\ &= \frac{1}{2}((x + y - 2)(x + y - 1)).\end{aligned}$$

We have $x + y \leq 2$ by (2.5). If $x + y \geq 1$, the condition (2.3) satisfies the condition (2.4) automatically.

In the case of $x + y < 1$, the condition (2.4) satisfies the condition (2.3) automatically.

However,

$$\begin{aligned}-\frac{(x-1)x + (y-1)y}{2} - xy &= -\frac{1}{2}((x+y)^2 - (x+y)) \\ &= -\frac{1}{2}((x+y)(x+y-1)) \\ &> 0.\end{aligned}$$

Since $S \in \mathrm{Aff}^+(2)$, the determinant S is positive, i.e. $xy - h^2 > 0$. Therefore,

$$h^2 < xy < -\frac{(x-1)x + (y-1)y}{2}.$$

This inequality shows that the condition (2.4) is always satisfied.

As a result, the condition which satisfies $\|\boldsymbol{A}_p(t)\boldsymbol{v}\|$ is a monotone function for all $\boldsymbol{v} \in S^1$ is

$$((x-1)(y-1) \geq h^2) \wedge ((x + y \geq 2) \vee ((0 < x \leq 1) \wedge (0 < y \leq 1))).$$

□

From the above results, there is a relation between the SVD method and the polar decomposition method.

Theorem 2.2.6. For a given $\boldsymbol{A} \in \mathrm{Aff}^+(2)$, the following are equivalent:

1. the SVD interpolation method $\boldsymbol{A}_s(t)$ is a good interpolation for the change of area.
2. the polar decomposition interpolation method $\boldsymbol{A}_p(t)$ is a good interpolation for the change of area.

Proof. By Theorem 2.2.2 and 2.2.4, it is enough to show $\mathrm{tr}\boldsymbol{D} = \mathrm{tr}\boldsymbol{S}$. In Section 2.2.3, we have $\boldsymbol{S} = \boldsymbol{R}_\beta^T \boldsymbol{D} \boldsymbol{R}_\beta$, where $\boldsymbol{D}$ is a diagonal matrix. $\boldsymbol{R}$ is a rotation matrix. $\beta \in [0, 2\pi)$. Then, we have $\mathrm{tr}\boldsymbol{S} = \mathrm{tr}(\boldsymbol{R}_\beta^T \boldsymbol{D} \boldsymbol{R})$. By $\mathrm{tr}(\boldsymbol{X}\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{Y}\boldsymbol{X})$ for all $\boldsymbol{X}$ and $\boldsymbol{Y}$, $\mathrm{tr}(\boldsymbol{R}_\beta^T \boldsymbol{D} \boldsymbol{R}) = \mathrm{tr}(\boldsymbol{R}\boldsymbol{R}_\beta^T \boldsymbol{D}) = \mathrm{tr}\boldsymbol{D}$ Hence, $\mathrm{tr}\boldsymbol{S} = \mathrm{tr}\boldsymbol{D}$. □

2.2.4 Example

For two given triangles $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$ and $Q = (\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3)$, we compare results of interpolations using linear, SVD and polar decomposition methods. Let $\boldsymbol{p}_1 = (0, 0)$, $\boldsymbol{p}_2 = (0, 3)$ and $\boldsymbol{p}_3 = (2, 0)$, $\boldsymbol{q}_1 = (0, 0)$, $\boldsymbol{q}_2 = (-9\sqrt{3}/4, -15/4)$ and $\boldsymbol{q}_3 = (-1/2, 3\sqrt{3}/2)$. Then, an affine map $\boldsymbol{A}$ from P to Q is $\begin{pmatrix} -1/4 & -3\sqrt{3}/4 \\ 3\sqrt{3}/4 & -5/4 \end{pmatrix}$. Let $\boldsymbol{A}_{\mathrm{svd}}$ be a SVD of $\boldsymbol{A}$ and $\boldsymbol{A}_{\mathrm{pd}}$ be a polar decomposition of $\boldsymbol{A}$. Then, $\boldsymbol{A}_{\mathrm{svd}} = \boldsymbol{R}_{\pi/3} \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \boldsymbol{R}_{-5\pi/3}$ and $\boldsymbol{A}_{\mathrm{pd}} = \boldsymbol{R}_{-4\pi/3} \begin{pmatrix} 5/4 & -\sqrt{3}/4 \\ -\sqrt{3}/4 & 7/4 \end{pmatrix}$.

Figure 2.2, 2.3 and 2.4 are interpolation results from a triangle P to a triangle Q using linear, SVD and polar decomposition method. In this example, a linear interpolation method is not a "good" interpolation for the changes of area and norm. Figure 2.5 shows values of $\det \boldsymbol{A}_s(t)$ over $0 \leq t \leq 1$ for a linear interpolation method. However, a SVD interpolation method is a "good" interpolation for the change of area. On the other hand, a polar decomposition is a "good" interpolation for not only the change of area but also the change of norm. This result show that a polar decomposition interpolation method is the best way from the perspective of our definitions of "good" interpolations.

2.3 The gap of compositions

For a global interpolation, we define a global interpolation as a minimum of an energy function defined by a sum of energies of local interpolations. Generally, we choose the origin of the local transformation as the center of

Figure 2.2: A interpolation result using a linear interpolation method.

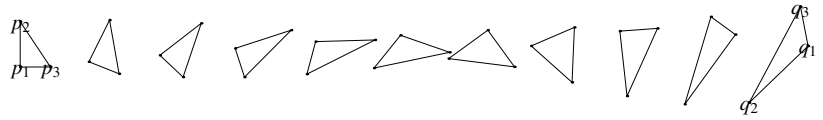

Figure 2.3: A interpolation result using a SVD interpolation method.

the source triangle. On the other hand, for convenience, we choose the same origin for each local transformation in the formula of an energy function of a global transformation. We consider that this small gap may cause different interpolations, and we would like to clarify conditions of a source and target configuration which cause a critical error with the global interpolation. In this section, we consider a simple figure consisting of a pair of triangles.

Let T_1 and T_2 be this pair of triangles. A_1 and A_2 are affine maps with respect to T_1 and T_2. we call T_1 and T_2 as the first triangle and the second triangle.

We assume the values of origins of the first triangle and the second triangle are zero. Then, we define an energy function E_1 as following.

Definition 2.3.1 (The origin is zero)**.** Let $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathrm{Aff}^+(2)$. The energy function E_1 is defined by

$$E_1(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2) := \|\boldsymbol{A}_1 - \boldsymbol{B}_1\|_F^2 + \|\boldsymbol{A}_2 - \boldsymbol{B}_2\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm.

Next, we consider a case if two triangles have different origins. Then, we define a new energy function E_2 as following.

Definition 2.3.2 (The origins are $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$)**.** Let $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathrm{Aff}^+(2)$, $\boldsymbol{o}_1$ be the origin's coordinates of the first triangle and $\boldsymbol{o}_2$ be the origin's coordinates of the second triangle. Suppose that $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ are represented by column

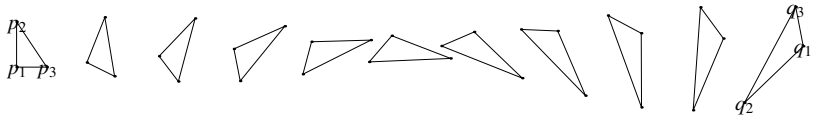

Figure 2.4: A interpolation result using a polar decomposition interpolation method.

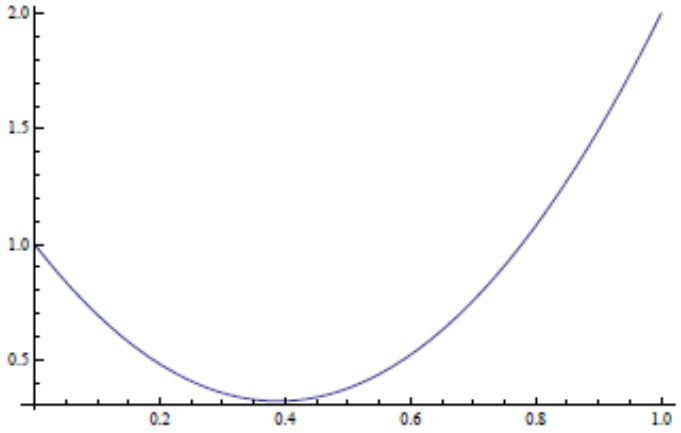

Figure 2.5: A graph of $\det A_l(t)$ for a linear interpolation method.

vectors. Then, the energy function E_2 is defined by

$$E_2(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{o}_1, \boldsymbol{o}_2) := E_1(T(\boldsymbol{A}_1, -\boldsymbol{o}_1), T(\boldsymbol{A}_2, -\boldsymbol{o}_2), T(\boldsymbol{B}_1, -\boldsymbol{o}_1), T(\boldsymbol{B}_2, -\boldsymbol{o}_2)).$$

That is,

$$E_2(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{o}_1, \boldsymbol{o}_2) = \|T(\boldsymbol{A}_1, -\boldsymbol{o}_1) - T(\boldsymbol{B}_1, -\boldsymbol{o}_1)\|_F^2 + \|T(\boldsymbol{A}_2, -\boldsymbol{o}_2) - T(\boldsymbol{B}_2, -\boldsymbol{o}_2)\|_F^2,$$

where $\|\cdot\|_F$ is the Frobenius norm and

$$T(\boldsymbol{A}_i, \boldsymbol{o}_i) := \begin{pmatrix} \boldsymbol{I} & \boldsymbol{o}_i \\ 0 & 1 \end{pmatrix} \boldsymbol{A}_i \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{o}_i \\ 0 & 1 \end{pmatrix}.$$

We investigate relations between two energies E_1 and E_2. Then, we have the following problem.

Problem 2.3.1. Let $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathrm{Aff}^+(2)$ and $\boldsymbol{o}_1 = (x, y)$ a point which satisfies the following equation. Does the equation

$$\min_{\boldsymbol{B}_1, \boldsymbol{B}_2} E_1(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2) = \min_{\boldsymbol{B}_1, \boldsymbol{B}_2} E_2(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{o}_1, \boldsymbol{o}_1)$$

hold or not?

We show a counter example which shows the above equation does not hold. We assume that the source and target configurations consist of four vertices and two triangles. For an intermediate configuration between the source and target configurations, this vertices are $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$ and $\boldsymbol{v}_4$. Then, the coordinates of $\boldsymbol{v}_i$ are described by $\boldsymbol{v}_i = (v_{ix}, v_{iy})$ Let $\boldsymbol{A}_1$ be an affine map of the first triangle from a source configuration to a target configuration and $\boldsymbol{A}_2$ be an affine map of the second triangle between them. In addition, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are affine maps of the first and second triangles from a source configuration to an intermediate configuration. Then, we have

$$\boldsymbol{A}_1 = \begin{pmatrix} 0 & 2 & -x+2y \\ -2 & 0 & 4-2x-y \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{A}_2 = \begin{pmatrix} 1 & 2 & 2y \\ -1 & 0 & 3-x-y \\ 0 & 0 & 1 \end{pmatrix},$$
$$\boldsymbol{B}_1 = \begin{pmatrix} -v_{1x}+v_{2x} & -v_{1x}+v_{3x} & b_{11} \\ -v_{1y}+v_{2y} & -v_{1y}+v_{3y} & b_{12} \\ 0 & 0 & 1 \end{pmatrix}, \boldsymbol{B}_2 = \begin{pmatrix} -v_{3x}+v_{4x} & -v_{2x}+v_{4x} & b_{21} \\ -v_{3y}+v_{4y} & -v_{2y}+v_{4y} & b_{22} \\ 0 & 0 & 1 \end{pmatrix},$$

where

$$\begin{aligned} b_{11} &= -x + v_{1x} + x(-v_{1x}+v_{2x}) + y(-v_{1x}+v_{3x}), \\ b_{12} &= -y + v_{1y} + x(-v_{1y}+v_{2y}) + y(-v_{1y}+v_{3y}), \\ b_{21} &= -x + v_{2x} + v_{3x} - v_{4x}x(-v_{3x}+v_{4x}) + y(-v_{2x}+v_{4x}), \\ b_{22} &= -y + v_{2y} + v_{3y} - v_{4y}x(-v_{3y}+v_{4y}) + y(-v_{2y}+v_{4y}). \end{aligned}$$

A solution of this minimization problem is

$$\begin{aligned}
v_{1x} &= \frac{1+x(-1+x+y)}{2(x^2+2xy+y^2-2x-2y+3)},\\
v_{1y} &= \frac{1}{2}\left(7+\frac{1+(x-1)(-1+x+y)}{x^2+2xy+y^2-2x-2y+3}\right),\\
v_{2x} &= \frac{1}{2},\\
v_{2y} &= 2,\\
v_{3x} &= 2,\\
v_{3y} &= \frac{7}{2},\\
v_{4x} &= \frac{1}{2}\left(5+\frac{1+x(-1+x+y)}{x^2+2xy+y^2-2x-2y+3}\right),\\
v_{4y} &= \frac{1}{2}\left(4+\frac{1+(x-1)(-1+x+y)}{x^2+2xy+y^2-2x-2y+3}\right).
\end{aligned}$$

Then, a minimum value of E_2 is given by

$$\min_{\boldsymbol{B}_1,\boldsymbol{B}_2} E_2(\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{B}_1,\boldsymbol{B}_2,\boldsymbol{o}_1,\boldsymbol{o}_1) = \frac{2+(x-1)x+(y-1)y}{x^2+2xy+y^2-2x-2y+3}.$$

Hence, $\boldsymbol{o}_1=(x,y)$ which satisfies $\min_{\boldsymbol{B}_1,\boldsymbol{B}_2} E_1 = \min_{\boldsymbol{B}_1,\boldsymbol{B}_2} E_2$ is

$$x^2-4xy+y^2+x+y-\frac{1}{4}=0.$$

In this result, we show that the contour of the origin which satisfies

$$\min_{\boldsymbol{B}_1,\boldsymbol{B}_2} E_1(\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{B}_1,\boldsymbol{B}_2) = \min_{\boldsymbol{B}_1,\boldsymbol{B}_2} E_2(\boldsymbol{A}_1,\boldsymbol{A}_2,\boldsymbol{B}_1,\boldsymbol{B}_2,\boldsymbol{o}_1,\boldsymbol{o}_1)$$

is a quadratic curve (Figure 2.6) . On the other hand, this result shows that the difference of origins cause a different global interpolation (Figure 2.7) .

2.4 Conclusions

We investigate useful formulations for 2D shape interpolations. For a local interpolation, we defined two "good" interpolations from the viewpoints of the changes of "area" and "norm". We consider conditions for given source and target configurations where the interpolation of this configuration is "good" in our sense. We obtained conditions of "goodness" for each

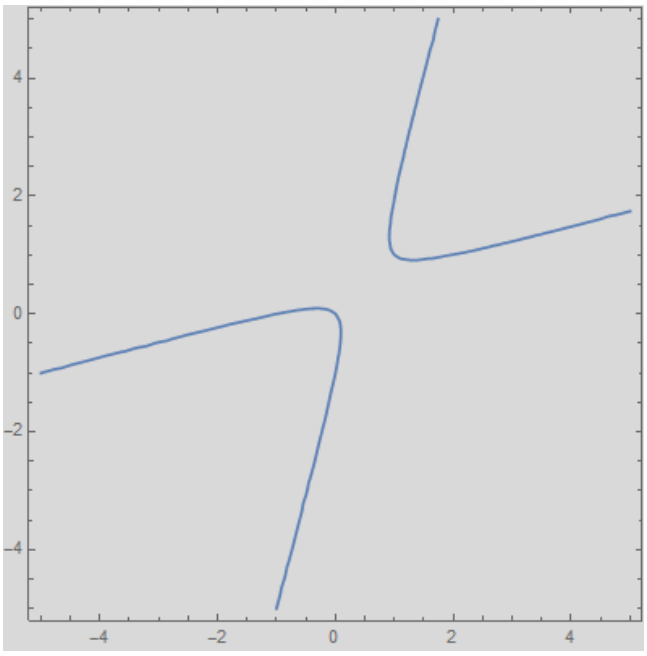

Figure 2.6: Contours of the origin which satisfies $\min_{B_1,B_2} E_1(A_1, A_2, B_1, B_2) = \min_{B_1,B_2} E_2(A_1, A_2, B_1, B_2, o_1, o_1)$.

known methods such as linear interpolation, SVD interpolation and polar decomposition interpolation.

In ARAP method, we define a global interpolation as a minimum of an energy function defined by a sum of energies of local interpolations. Generally, we choose the origin of the local interpolation as the center of the source configuration. On the other hand, for convenience, we choose the same origin for each local interpolation in the formula of an energy function of a global interpolation. The first problem we consider is that this small gap may cause different interpolations, and we would like to clarify conditions of the source and target configuration which cause a critical error with the ARAP method. We choose a simple figure with two triangles which we call them the first triangle and the second triangle. We consider the values of the global energy function when we fix the origin of the first triangle to zero and vary the origin of the second triangle. Then, we can show contours of the origin of the second triangle are quadratic curves. This result shows

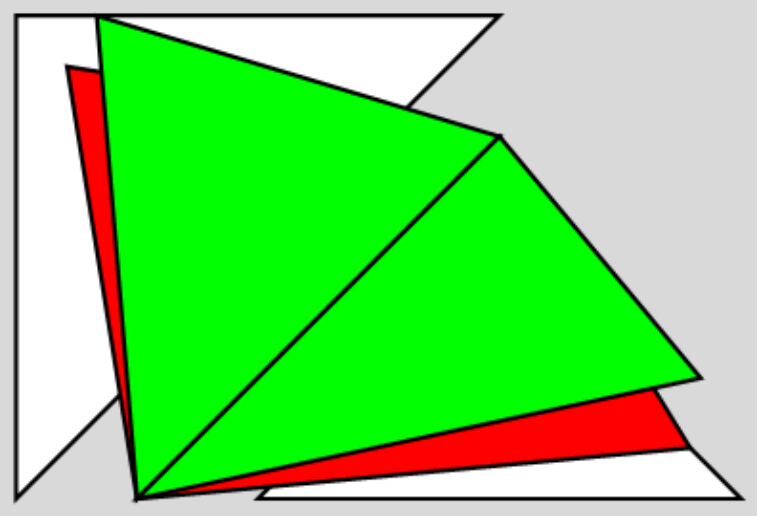

Figure 2.7: Results of global interpolations. Red: origins of two triangles are zero. Green: each origin of two triangles is barycenter of each triangle.

an answer of our first problem. The difference of origins cause a different global interpolations.

Chapter 3

Computation of 3×3 Matrix Exponentials

In this chapter, we investigate fast algorithms for the matrix exponentials and eigenvalues of matrix exponentials and eigenvalues of 3×3 real symmetric matrices. We show that matrix exponentials and eigenvalues of the 3×3 real symmetric matrix can be represented by specific formulas. Then, we achieve a notable performance improvement using our algorithm.

3.1 Algorithm for Matrix Exponential

Let $M_n(\mathbb{R})$ be the set of $n \times n$ matrices.

Definition 3.1.1. The set of 3×3 real symmetric matrices is defined by $\mathrm{Sym}(3) := \{S \mid S = {}^tS \in M_3(\mathbb{R})\}$, where ${}^tS$ is the transpose of S.

Definition 3.1.2. For $S \in \mathrm{Sym}(3)$, the matrix exponential of S is defined by

$$\exp(S) = \sum_{k=0}^{\infty} \frac{S^k}{k!}.$$

Although the sum of the infinite series converges, their rate of convergence may not be so high. So we can not have an efficient algorithm by direct computations.

3.1.1 Algorithm1. Diagonalization

First, we introduce the method using diagonalization. This method helps to compute their exponential easily.

Proposition 3.1.1. Let S be a real symmetric matrix. Then, S is decomposed by an orthogonal matrix P and a diagonal matrix D, and S is described as $S = PD^{\mathrm{t}}P$.

Proposition 3.1.2. Let S be an element of Sym(3). Then, the matrix exponential of S is $\exp(S) = P\exp(D)^{\mathrm{t}}P$, where P is an orthogonal matrix and D is a diagonal matrix. P and D satisfy $S = PD^{\mathrm{t}}P$.

Proof. Since S is a real symmetric matrix, S is also diagonalized with an orthogonal matrix P and a diagonal matrix D such that $S = PD^{\mathrm{t}}P$.

From definition (3.1.2), $\exp(S)$ is

$$\begin{aligned}\exp(S) = \exp(PD^{\mathrm{t}}P) &= \sum_{k=0}^{\infty} \frac{(PD^{\mathrm{t}}P)^k}{k!} \\ &= P\left(\sum_{k=0}^{\infty} \frac{D^k}{k!}\right)^{\mathrm{t}}P = P\exp(D)^{\mathrm{t}}P.\end{aligned}$$

□

So we can make an algorithm of a matrix exponential using the diagonalization. The algorithm needs to compute eigenvalues (D) and eigenvectors (P) to compute the matrix exponential. The computational cost of eigenvectors however is more expensive than the cost of eigenvalues. Next, we review the faster algorithm in [6] to improve this drawback. This method does not need to compute eigenvectors and is faster than the method using diagonalization.

3.1.2 Algorithm2. Spectral Decomposition

We describe the algorithm using spectral decomposition [6, 8]. Let λ_1, λ_2 and λ_3 be the eigenvalues of $S \in \mathrm{Sym}(3)$. They are the roots of the characteristic polynomial of S. The characteristic polynomial of S is defined as follows.

Definition 3.1.3. Let $\phi_S(\lambda)$ be the characteristic polynomial of $S \in \mathrm{Sym}(3)$. $\phi_S(\lambda)$ is defined as $\phi_S(\lambda) = \det(\lambda E - S)$, where det is the determinant operation.

From the Cayley-Hamilton theorem, the following proposition is satisfied.

Proposition 3.1.3. Let $\phi_S(\lambda)$ be the characteristic polynomial of $S \in \mathrm{Sym}(3)$ and we substitute S for λ and the identity matrix for 1 in this polynomial. Then, $\phi_S(S)$ is equal to 0.

Since $\phi_S(\lambda)$ is a third degree polynomial in λ, $\phi_S(S)$ is also a third degree polynomial in S. Therefore $\frac{S^k}{k!}$ can be described as $\frac{S^k}{k!} = Q_k\phi_S(S) + R_k$, where Q_k is a polynomial in S, and R_k is an at most second degree polynomial in S. Hence, $\exp(S) = \sum_{k=0}^{\infty}(Q_k\phi_S(S) + R_k)$. Then, $\sum_{k=0}^{\infty} Q_k\phi_S(S)$ is equal to 0 from the Cayley-Hamilton theorem, and $\sum_{k=0}^{\infty} R_k$ is an at most second degree polynomial in S. As a result, $\exp(S)$ can be described as $\exp(S) = xS^2 + yS + zE$, where E is the 3×3 identity matrix, and x, y and z are elements of $\mathbb{R}$.

Hence, $\exp(S)$ can be represented by a second degree polynomial in S. Next step is to decide $x, y, z \in \mathbb{R}$ to compute $\exp(S)$. In Proposition 3.1.2, $\exp(S)$ can be described as $\exp(S) = P\exp(D)^{\mathrm{t}}P$.

On the other hand, $xS^2 + yS + zE$ is described as $xS^2 + yS + zE = P(xD^2 + yD + zE)^{\mathrm{t}}P$. Therefore, $\exp(D) = xD^2 + yD + zE$.

Let λ_1, λ_2 and λ_3 be the eigenvalues of S. The elements of D are the eigenvalues of S. Then, we have

$$\exp(D) = \begin{pmatrix} e^{\lambda_1} & 0 & 0 \\ 0 & e^{\lambda_2} & 0 \\ 0 & 0 & e^{\lambda_3} \end{pmatrix},$$

$$xD^2 + yD + zE = \begin{pmatrix} x\lambda_1^2 + y\lambda_1 + z & 0 & 0 \\ 0 & x\lambda_2^2 + y\lambda_2 + z & 0 \\ 0 & 0 & x\lambda_3^2 + y\lambda_3 + z \end{pmatrix}.$$

Hence, we can decide x, y and z to solve the following equation.

$$\begin{pmatrix} e^{\lambda_1} \\ e^{\lambda_2} \\ e^{\lambda_3} \end{pmatrix} = \begin{pmatrix} \lambda_1^2 & \lambda_1 & 1 \\ \lambda_2^2 & \lambda_2 & 1 \\ \lambda_3^2 & \lambda_3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

The values of x, y and z depend on multiplicity of the eigenvalues of S. We consider two eigenvalues λ and λ' are same if $|\lambda - \lambda'| < 10^{-6}$.

Case 1. When all the three eigenvalues are same, put

$$x = y = 0,\ z = \exp(\lambda_1).$$

Case 2. When two of them are same, that is $\lambda_1 = \lambda_2$, put

$$x = 0,\ y = s - t,\ z = t\lambda_2 - s\lambda_3,$$

where

$$s = \frac{\exp(\lambda_2)}{\lambda_2 - \lambda_3},\ t = \frac{\exp(\lambda_3)}{\lambda_2 - \lambda_3}.$$

Case 3. When all of them are distinct, put

$$\begin{aligned} x &= s + t + u, \\ y &= -s(\lambda_2 + \lambda_3) + t(\lambda_3 + \lambda_1) + u(\lambda_1 + \lambda_2), \\ z &= s\lambda_2\lambda_3 - t\lambda_3\lambda_1 - u\lambda_1\lambda_2, \end{aligned}$$

where

$$s = \frac{\exp(\lambda_1)}{(\lambda_1 - \lambda_2)(\lambda_1 - \lambda_3)}, t = \frac{\exp(\lambda_2)}{(\lambda_2 - \lambda_3)(\lambda_1 - \lambda_2)}, u = \frac{\exp(\lambda_3)}{(\lambda_2 - \lambda_3)(\lambda_3 - \lambda_1)}.$$

Therefore, we have a simple formula for x, y and z. Our algorithm needs to compute only eigenvalues of a 3×3 real symmetric matrix so that we can compute a matrix exponential more efficiently than the algorithm using diagonalization.

Next, we will discuss an efficient algorithm for eigenvalues for this matrix in the next section.

3.2 Eigenvalues Using Viète's Formula

In this section, we investigate an algorithm using Viète's formula. This algorithm achieves a fast computation of eigenvalues in comparison with the algorithm using diagonalization.

If $S \in \mathrm{Sym}(3)$ is given, we need to solve the characteristic equation of S. Let λ_1, λ_2 and λ_3 be the eigenvalues of S. Then, the characteristic equation is $(x - \lambda_1)(x - \lambda_2)(x - \lambda_3) = 0$. This equation is expanded into

$$x^3 - (\lambda_1 + \lambda_2 + \lambda_3)x^2 + (\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1)x - \lambda_1\lambda_2\lambda_3 = 0.$$

Therefore, the characteristic equation of S is described as

$$x^3 - (\mathrm{tr}S)x^2 + \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2}x - \det S = 0,$$

where $\mathrm{tr}S$ is the trace of S, $\|S\|_F$ is the Frobenius norm of S.

This equation can be solved by using Viète's formula. For a given $S \in \mathrm{Sym}(3)$, let λ_1, λ_2 and λ_3 be the eigenvalues of S. Those are the roots of the characteristic equation of S:

$$x^3 - (\mathrm{tr}S)x^2 + \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2}x - \det S = 0.$$

Let a, b, c, and y be $a = -\mathrm{tr}S,\ b = \frac{(\mathrm{tr}S)^2 - \|S\|_F^2}{2},\ c = -\det S$ and $y = x + \frac{a}{3}$.

Then,

$$y^3 + \left(b - \frac{1}{3}a^2\right)y + \left(\frac{2}{27}a^3 - \frac{1}{3}ab + c\right) = 0.$$

Let p and q be $p = b - \frac{1}{3}a^2$ and $q = \frac{2}{27}a^3 - \frac{1}{3}ab + c$. Since all eigenvalues of real symmetric matrices are real numbers, the solutions are also real numbers. Hence, the discriminant satisfies $-(4p^3 + 27q^2) \geq 0$, especially $p \leq 0$. If $p = 0$, then we find $q = 0$. Then, λ_1, λ_2 and λ_3 are equal to $\frac{\mathrm{tr}S}{3}$. So we consider the case of $p < 0$. Let $t = \sqrt{-\frac{4p}{3}}$, $u = \frac{y}{t}$ and $k = -\frac{4q}{t^3}$. Then, the equation is described as

$$4u^3 - 3u - k = 0.$$

Since $4p^3 + 27q^2 \leq 0$, we have $|q| \leq \sqrt{-4p^3/27}$, $|4q| \leq t^3$ and $|k| \leq 1$. To solve the equation, we use the cosine's Triple-angle formula $\cos 3\theta = 4\cos^3\theta - 3\cos\theta$. Let u_1, u_2 and u_3 be the roots of the equation, and set $\theta = \frac{1}{3}\arccos(k)$. From this formula, the roots are

$$u_1 = \cos\left(\frac{1}{3}\arccos k\right), u_2 = \cos\left(\frac{1}{3}\arccos k + \frac{2}{3}\pi\right), u_3 = \cos\left(\frac{1}{3}\arccos k + \frac{4}{3}\pi\right).$$

From $y = x + \frac{a}{3}, u = \frac{y}{t}$, the eigenvalues of S are $\lambda_i = tu_i - \frac{a}{3}$ $(i = 1, 2, 3)$. Therefore, we find a simple formula for the eigenvalues of a 3×3 symmetric matrix.

$$\lambda_1 = \frac{r\cos\left(\frac{\arccos(k)}{3}\right) + \mathrm{tr}S}{3}, \lambda_2 = \frac{r\cos\left(\frac{\arccos(k)+2\pi}{3}\right) + \mathrm{tr}S}{3},$$
$$\lambda_3 = \frac{r\cos\left(\frac{\arccos(k)+4\pi}{3}\right) + \mathrm{tr}S}{3},$$

where

$$r = 3t = \sqrt{-12p},\ k = -\frac{108q}{r^3},$$
$$p = \frac{(\mathrm{tr}S)^2 - 3\|S\|_F^2}{6},\ q = \frac{5}{54}(\mathrm{tr}S)^3 - \frac{1}{6}\mathrm{tr}S\|S\|_F^2 - \det S.$$

As a result, we obtain the eigenvalues of S. We note that the eigenvalues of S can be represented by a simple formula using the trace, determinant

and Frobenius norm of S. The proposed method for computing eigenvalues involves computation of cosine and arccosine. Cosine is implemented in the Eigen library. The function comes from Julien Pommier's SSE math library which is known as an accurate math library, and arccosine is implemented in the C++ Standard Library. We consider the speed and accuracy of these functions are acceptable.

3.3 Applications

We incorporated our implementation of the algorithm discussed above into the shape deforming tools developed in [6]. For example, Cage-based deformer gives a target shape and a "cage" surrounding it. The cage can be any triangulated polyhedron wrapping the target shape. We want to deform the target shape by manipulating not directly on it but through the proxy cage. Essentially this deformation is based on polar decompositions and matrix exponentials of 3×3 real symmetric matrices. We examined our improvements replacing the exponential part in this deformation program.

3.4 Experimentation

In this section, we compared the computation times of our algorithms with standard techniques. We use a Intel Core i7 1.9 GHz CPU with 4 GB of RAM and Windows 8 (64bit) OS. In the experimentations, we implemented our algorithms in C++ programming language (Microsoft Visual Studio 2010 Ultimate), and the optimization option (/O2) of this compiler was maximization of the execution speed. Our implementation used the Eigen library which provides many functions of matrix operations.

First we compared our algorithm for the computation of the eigenvalues of a 3×3 real symmetric matrix with the QR algorithm [13] which uses the QR decomposition. This algorithm is implemented in the Eigen library as a class member function which computes the eigenvalues. Before call the function, we need to create an instance of this class by specifying the size of matrix. Then, the QR algorithm is restricted to 3×3 matrix. For given 10^7 three dimensional real symmetric matrices randomly, we compared the running times for the computation of eigenvalues of all given matrices. In the applications of CG, the eigenvalues of the given matrices should be more than 0. Hence, we need to obtain positive semi-definite matrices. The way of generating these matrices randomly is the following. First, we generate a real matrix M and choose the elements of M from the range $[-1, 1]$.

Next, we compute $S = {}^tMM$ to obtain a real symmetric matrix. Then, the eigenvalues of S are more than 0 because S is a positive semi-definite matrix. Figure 3.1 shows a comparison of the computation of the eigenvalues using the QR algorithm and our algorithm using Viète's formula. Our algorithm is about 3.2 times faster than the QR algorithm.

The Eigen library provides an algorithm computing the eigenvalues using Viète's formula. However, we implemented our algorithm independently using our formalized formula using the trace, determinant and Frobenius norm.

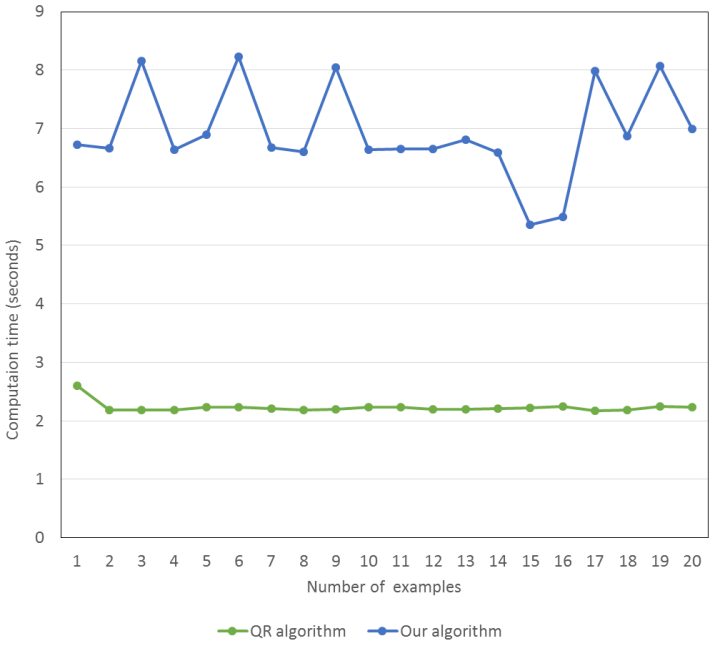

Figure 3.1: Computation time of the eigenvalues of the 3×3 real symmetric matrix. Upper: using the QR algorithm. Lower: using our algorithm.

Next, we compared our algorithm for the computation of the matrix exponential of a 3×3 real symmetric matrix with an algorithm based on diagonalization. For given 10^7 three dimensional real symmetric matrices chosen randomly by the same method in the previous experimentation, we compared the average running times for computing the matrix exponentials using the algorithm based on diagonalization (Diag), spectral decomposition

(SD) and improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues. Figure 3.2 shows a comparison of the computation of the matrix exponentials using the three algorithms. As a result, improved SD is the best algorithm among those algorithms and achieved about 4.2 times faster than Diag.

We consider eigenvalues between 0 and 3, because a transformation matrix induced by an interactive motion does not have large eigenvalues. By our experiments, the error rate is less than 10^{-6} and it is acceptable for making a deformation of computer graphics. Our method using Viète's formula may produce small numerical errors, but we have benefits of computing time using our exponential matrix without computing eigenvectors. Since we consider $\lambda \leq 3$, the error rate of the difference of cases in our algorithm are bounded. Experimentally, the computation of deformations are acceptable.

Finally we incorporated the implementations of Diag and improved SD into the Cage-based deformer algorithm in [6]. For a dragon model with 50,000 vertices, 100,000 triangles, we compared the average running times of deforming this model using two algorithms. Figure 3.3 shows the comparison of the computation times, and the average runtime of improved SD is about 1.2 times faster than Diag. Compared with the result for the eigenvalues, this improvement in speed for the application may not look like significant. This is because the application has other heavy computations. In more detail, the application computes affine transformations for all triangular meshes of a given model and logarithms of matrices obtained from the affine transformations. In addition, the application needs a weighted sum calculation to decide the shape deformation.

Therefore, our algorithm can be useful for a practical deformation tool.

3.5 Conclusions

We investigate fast algorithms for the matrix exponentials and eigenvalues of 3×3 real symmetric matrices. We show that the matrix exponential and eigenvalues of the 3×3 real symmetric matrix can be represented by specific formulas using the trace, determinant and Frobenius norm of the matrix. Our algorithms are implemented in C++ programming language as functions which compute the matrix exponential and eigenvalues of the 3×3 real symmetric matrix. For the computation time of the matrix exponential, we compare our algorithm with other algorithms. In addition, we incorporate the implementations into a practical deformation tool. Using the deformation tool, we evaluate the computation time of our algorithm

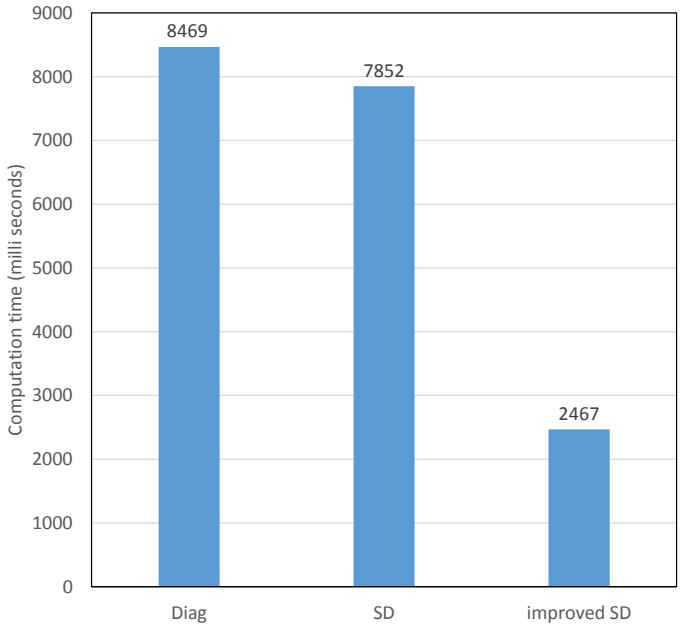

Figure 3.2: A comparison of average runtimes for the computation of the matrix exponential of 3×3. Left: using diagonalization (Diag). Center: using spectral decomposition (SD). Right: using the improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues.

for the deformation of a given model.

In conclusion, we achieve a notable performance improvement using our algorithm. In fact, we compare the computation of the eigenvalues using the QR algorithm and our algorithm using Viète's formula. Our algorithm is about 3.2 times faster than the QR algorithm. Next, we compare the computation of the matrix exponentials using an algorithm based on diagonalization, our algorithm based on spectral decomposition and our improved algorithm which includes our algorithm computing the eigenvalues using Viète's formula. Then, our improved algorithm is the best algorithm among those algorithms and achieved about 4.2 times faster than the algorithm using diagonalization. Finally we incorporate the implementations of the algorithm using diagonalization and our improved algorithm into the

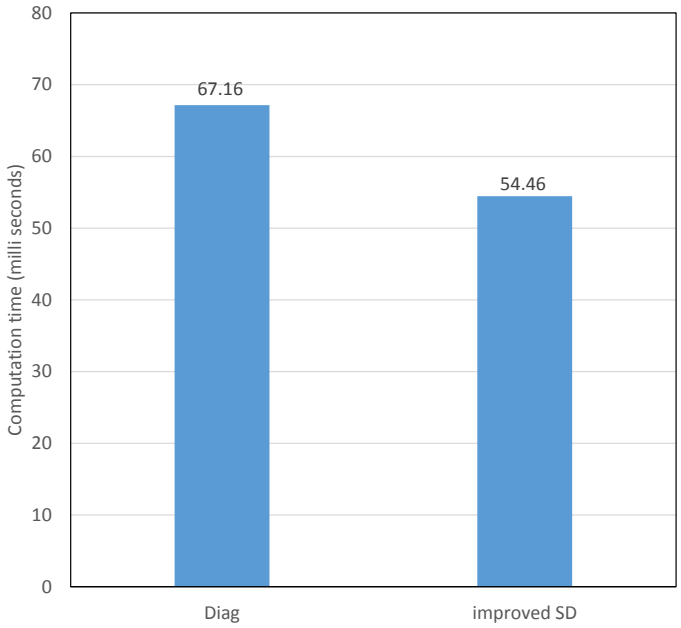

Figure 3.3: A comparison of average runtimes for the deformation of a dragon model. Left: using diagonalization (Diag). Right: using improved spectral decomposition (improved SD) which includes our algorithm for the computation of the eigenvalues.

Cage-based deformer. For a dragon model, we compare average running times of deforming this model using two algorithms. As a result, the average runtime of our improved algorithm is about 1.2 times faster than the algorithm based on the diagonalizaion.

We have not estimated the accurate numerical error of our matrix exponential algorithm. In this paper, we just show that the benefits of computational costs and acceptable results of applications of deformations.

Bibliography

[1] Marc Alexa. Linear combination of transformations. In *Proc. the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 380–387. ACM, 2002.

[2] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *Proc. the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 35–42, New York, NY, USA, 1992. ACM.

[4] Roger W. Brockett. Robotic manipulators and the product of exponentials formula. In P.A. Fuhrmann, editor, *Proc. the MTNS-83 International Symposium*, pages 120–129. Springer Berlin Heidelberg, 1983.

[5] Takeo Igarashi and Yuki Igarashi. Implementing as-rigid-as-possible shape manipulation and surface flattening. *J. Graphics, GPU, & Game Tools*, 14(1):17–30, 2009.

[6] Shizuo Kaji, Sampei Hirose, Hiroyuki Ochiai, and Ken Anjyo. A Lie theoretic parameterization of affine transformations. In *Proc. Symposium MEIS2013: Mathematical Progress in Expressive Image Synthesis*, volume 50 of *MI Lecture Note*, pages 134–140. Kyushu University, 2013.

[7] Shizuo Kaji, Sampei Hirose, Shigehiro Sakata, Yoshihiro Mizoguchi, and Ken Anjyo. Mathematical analysis on affine maps for 2d shape interpolation. In *Proc. the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 71–76. Eurographics Association, 2012.

[8] Kohei Matsushita, Hiroyasu Hamada, and Genki Matsuda. A fast computation of matrix exponential and its application in cg. In *Proc. Forum "Math-for-Industry" 2013*, volume 51 of *MI Lecture Note*, page 92. Kyushu University, 2013.

[9] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[10] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 245–254, New York, NY, USA, 1985. ACM.

[11] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[12] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.

[13] James H. Wilkinson, editor. *The Algebraic Eigenvalue Problem*. Oxford University Press, Inc., 1988.

[14] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, SPM '05, pages 267–274, New York, NY, USA, 2005. ACM.