



Run Run Shaw Library

香港城市大學
City University of Hong Kong

Copyright Warning

Use of this thesis/dissertation/project is for the purpose of private study or scholarly research only. ***Users must comply with the Copyright Ordinance.***

Anyone who consults this thesis/dissertation/project is understood to recognise that its copyright rests with its author and that no part of it may be reproduced without the author's prior written consent.

CITY UNIVERSITY OF HONG KONG

香港城市大學

High Quality Compatible Triangulations with Inconsistent
Rotation and Shape Self-occlusion Enhancement for Planar
Shape Morphing

基於提高旋轉不一致和形狀自遮擋的高質量同構三角剖
分方法在平面形狀變形中的研究

Submitted to
Department of Computer Science
電腦科學系
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
哲學博士學位

by

Liu Zhiguang
劉志廣

September 2016
二零一六年九月

Abstract

Planar shape morphing, also known as metamorphosis or shape blending, is the gradual transformation of one shape into another. Shape morphing techniques have been used widely in animation and special effects packages, such as Adobe After Effects and HTML5. With these morphing methods, we can transform a human to a bird or some other objects that people may never experience in real life. Thus, we want to build an interactive system that blends the human silhouette and other shapes such that the users can see these interesting transformations. To build such a system, (1) we need to employ compatible triangulation method to compute the correspondence between two shapes. (2) we need to apply shape interpolation method to transform one shape to another. (3) we need to use posture reconstruction method to address the transformation that involves self-occlusion.

Computing compatible triangulation can build the one-to-one correspondence between both the boundary and interior of two shapes. In this thesis, we propose a new method to compute compatible triangulation of two polygons in order to create a smooth geometric transformation between them. Compared with existing methods, our approach creates triangulations of better quality, that is, triangulations with fewer long thin triangles and Steiner points. This results in visually appealing morphing when transforming the shape from one to another. Our method consists of three stages. First, we use the common valid vertex pair to uniquely decompose

the source and target polygons into pairs of sub-polygon, in which each concave sub-polygon is triangulated. Second, within each sub-polygon pair, we map the triangulation of a concave sub-polygon onto the corresponding sub-polygon using linear transformation, thereby generating the compatible meshes between the source and the target. Third, we refine the compatible meshes, which can create better quality planar shape morphing with detailed textures.

Shape interpolation algorithms determine the path that transforms the source shape into the target one. Traditional image space interpolation methods use different features, points or line segments for example, to discretize the image. To achieve realistic morphing results, users need to carefully draw the corresponding features on both the source and target images. Previous work has shown that rigid shape interpolation methods can maximize the rigidity of a blended shape, which results in sensible transformations. However, the rigid shape interpolation approaches will suffer from inconsistent rotations whenever the rotation is more than π . We offer an efficient algorithm that gives a unique rotation assignment with minimum rotation angle. We create a graph with each original rotation angle as one vertex of the graph. During the searching process, we fix any jump that is larger than π by adding or subtracting multiple 2π . All the correct rotations in the thesis are generated by this efficient scheme.

It is still challenging to accurately recognize postures from a single depth camera due to the inherently noisy data derived from depth images and self-occluding action performed by the user. In this thesis, we propose a new real-time probabilistic framework to enhance the accuracy of live captured postures that belong to one of the action classes in the database. We adopt the Gaussian Process model as a prior to leverage the position data obtained from Kinect and marker-based motion capture

system. We also incorporate a temporal consistency term into the optimization framework to constrain the velocity variations between successive frames. To ensure that the reconstructed posture resembles the accurate parts of the observed posture, we embed a set of joint reliability measurements into the optimization framework. A major drawback of Gaussian Process is its cubic learning complexity when dealing with a large database due to the inverse of a covariance matrix. To solve the problem, we propose a new method based on a local mixture of Gaussian Processes, in which Gaussian Processes are defined in local regions of the state space. Due to the significantly decreased sample size in each local Gaussian Process, the learning time is greatly reduced. At the same time, the prediction speed is enhanced as the weighted mean prediction for a given sample is determined by the nearby local models only. Our system also allows incrementally updating a specific local Gaussian Process in real time, which enhances the likelihood of adapting to run-time postures that are different from those in the database.

Experimental results show that our method can create compatible meshes of higher quality compared with existing methods, which facilitates smoother morphing process. The proposed algorithm is robust and computationally efficient. Our posture reconstruction system can generate high quality postures even under severe self-occlusion situations. Our system can be applied to produce convincing transformations such as interactive 2D animation creation and special effects in movies.

CITY UNIVERSITY OF HONG KONG
Qualifying Panel and Examination Panel

Surname: LIU
First Name: Zhiguang
Degree: PhD
College/Department: Department of Computer Science

The Qualifying Panel of the above student is composed of:

Supervisor(s)

Dr. LEUNG Wing Ho Howard Department of Computer Science
City University of Hong Kong

Qualifying Panel Member(s)

Dr. CHAN Antoni Bert Department of Computer Science
City University of Hong Kong

Prof. LI Qing Department of Computer Science
City University of Hong Kong

This thesis has been examined and approved by the following examiners:

Dr. WONG Hau San Department of Computer Science
City University of Hong Kong

Prof. LI Qing Department of Computer Science
City University of Hong Kong

Dr. LEUNG Wing Ho Howard Department of Computer Science
City University of Hong Kong

Dr. SUN Hanqiu Department of Computer Science and Engineering
The Chinese University of Hong Kong

Acknowledgements

I would like to express my heartfelt gratitude and respect to my supervisor, Dr. Howard Leung for his inspirational guidance, constant support and continuous encouragement during my Ph.D. study. He always gives me great ideas and professional suggestions when I am facing difficult research problems. His influence on me will never fade away in my research and life for sure.

I would also like to express my sincere gratitude to my Ph.D. qualifying panel members Prof. Qing Li and Dr. Antoni B. Chan who often give me their precious suggestions that substantially contribute to my doctoral studies throughout these years. I would also like to thank Dr. Hubert P. H. Shum from Northumbria University for his suggestions that have inspired me a lot.

I am so thankful to my supportive lab-mates and friends, including Dr. Jacky Chun Pong Chan, Dr. Liuyang Zhou, Dr. Lingling Yang, Dr. Liang Tao, Mr. Meng Li, as well as Dr. Song Fang, Mr. Tian Mao for their fruitful discussion in both academic research and social life in Hong Kong. I am appreciative as well of the close friendships with Dr. Limin Li from Department of Physics and Materials Science, and Mr. Yuanbin He from Department of Electronic Engineering, for the time we spent together in the Jockey Club House these years. In particular, warm thanks go to Mr. Hamilton Chan from the CS general office for his kind and timely help during the whole period of study.

I wish to express my deepest gratitude to my family: my father, my mother, my brother and my wife. Their endless love, financial support, understanding, encouragement and most of all their patience have helped me to grow up all these years. I dedicate this thesis to my parents.

Table of Contents

Abstract	i
Acknowledgements	v
List of Figures	xi
List of Tables	xix
List of Algorithms	xxi
1 Introduction	1
1.1 Compatible Triangulation	3
1.2 Planar Shape Morphing	5
1.3 Posture Reconstruction	6
1.4 Outline of the Thesis	10
2 Related Work	11
2.1 Compatible Triangulation	11
2.1.1 Common Space based Compatible Triangulation	12
2.1.2 Divide and Conquer based Compatible Triangulation	12
2.2 Planar Shape Interpolation	13

2.2.1	Image Space Shape Interpolation	13
2.2.2	Object Space Shape Interpolation	15
2.3	Posture Reconstruction	16
2.3.1	Posture Reconstruction from Low Dimensional Signals	16
2.3.2	Data-Driven Posture Reconstruction	17
2.3.3	Regression based Posture Reconstruction	18
3	Compatible Triangulation	21
3.1	Compatible Triangulation Overview	22
3.2	Compatible Decomposition of the Source and Target Polygons	24
3.3	Compatible Triangulations Mapping	28
3.3.1	Mapping Steiner Points on the Link Path of Source Polygon	30
3.3.2	Mapping Steiner Points within the Source Polygon	30
3.4	Compatible Mesh Refining	33
3.5	Method Complexity	33
3.6	Experimental Results	35
3.6.1	Compatible Triangulation	35
3.6.2	Mesh Quality Evaluation	37
3.7	Summary	44
4	Planar Shape Interpolation	47
4.1	Choosing Shape Interpolation Approaches	47
4.1.1	Line Segment based Shape Interpolation	48
4.1.2	Shape Interpolation Using Moving Least Squares	49
4.1.3	Rigid Shape Interpolation	52
4.2	Assigning Consistent Rotation for Rigid Shape Interpolation	53

4.2.1	Previous Methods on Tackling Inconsistent Rotation	53
4.2.2	Our Method for Consistent Rotation Assigning	55
4.3	Experimental Results	59
4.3.1	Line Segment based Shape Interpolation	59
4.3.2	Shape Interpolation Using Moving Least Squares	61
4.3.3	Rigid Shape Interpolation	62
4.3.4	Regulating the Rotation for Rigid Shape Interpolation	66
4.4	Summary	72
5	Human Posture Reconstruction	73
5.1	Data Acquisition and Preprocessing	74
5.1.1	Data Acquisition	74
5.1.2	Posture Budgeting	75
5.2	Posture Reconstruction	77
5.2.1	Spatial Prediction	77
5.2.2	Incremental Learning of Local Gaussian Processes	80
5.2.3	Temporal Prediction	86
5.2.4	Reliability Embedding	88
5.2.5	Energy Minimization Function	89
5.3	Experimental Results	90
5.3.1	Posture Reconstruction	91
5.3.2	Qualitative Analysis	92
5.3.3	Quantitative Analysis	94
5.3.4	Comparison Between Randomized Forests and Our Method	99
5.3.5	Effects of Optimization Terms	101
5.4	Summary	102

6	Application: An Interactive Shape Morphing System	103
6.1	Interactive Shape Morphing System	103
6.2	Shape Morphing with Self-occlusion	107
6.2.1	Problem of Shape Morphing with Self-occlusion	107
6.2.2	Shape Morphing with Self-occlusion Enhanced	110
6.3	Summary	112
7	Conclusions and Future Directions	119
7.1	Conclusions	119
7.2	Future Directions	121
	List of Publications	125
	Bibliography	127

List of Figures

1.1	Overview of the system design.	3
1.2	Example of an inaccurately tracked posture from Kinect. The blue skeleton is the tracked result by Kinect.	8

3.1 Overview of the proposed framework to compatibly triangulate two simple polygons. (a) The target polygon Q . (b) The source polygon P . (c) We compute the valid vertex pairs for both the source and target polygons. (d) We collect the common valid vertex pairs. (e) We use the common valid vertex pair for compatible decomposition if the common vertex pair exists; otherwise we calculate the link path, e.g. the 2-link path between vertex u_2 and u_5 with blue color shown in (h). (f-h) We use the polyline found in (e) that maximizes the minimum angle to decompose the source and target polygons. (i) We triangulate each sub-polygon p_i of source polygon P using Delaunay triangulation. (j) We may need to add some Steiner points on the edge of sub-polygon q_i to keep equivalent topology. (k) We solve a linear system to map the triangulation of sub-polygon p_i onto the corresponding sub-polygon q_i of target polygon Q . (l-m) We finally refine the compatible meshes by operations such as splitting long edges and flipping interior edges so as to improve the interior angles of the mesh. 22

3.2 A valid vertex pair (1, 4) used to partition the source polygon, which yields four interior angles between vertex u_1 and u_4 25

3.3 Comparison between Liu et al., 15 and our method 29

3.4 Mapping Steiner points within the source sub-polygon onto the target sub-polygon. (a) The source sub-polygon with Steiner points u_1 and u_2 . (b) The corresponding target sub-polygon with unknown Steiner points v_1 and v_2 31

3.5 Compatible triangulations comparisons. We compare our results with those of [71], [7] and [43]. While we generally use less Steiner points than the others, our algorithm creates high quality compatible mesh in terms of the proportion of long thin triangles. 36

3.6 Compatible triangulation results. (a) The initial tessellations of two polygons. (b) mesh refinement and morphing. Note that our compatible mesh can be used to blend shapes with large rotations, e.g. shapes in the third row. 37

3.7 Morphing of Tai-chi motions: we adopt the compatible meshes generated by our algorithm to blend Tai-chi motions. 38

3.8 Mesh deformation evaluation. (a) Dog and cat. (b) Alligator. 41

3.9 Texture mapping comparison. (a1-a2) Source shape with texture. Adding texture to the target shape using compatible meshes generated by methods of (b1-b2) Surazhsky-Gotsman, 04. (c1-c2) Baxter et al., 09. (d1-d2) Liu et al., 15. (e1-e2) Ours. 42

3.10 Measure the angle changes. 43

3.11 Measure the angle changes. 43

4.1 Single line-segment-pair mapping 48

4.2 Multiple line-segment-pair mapping 49

4.3 Deformation with control points shown in purple 50

4.4 The problem of assigning rotations for rigid shape interpolation method: the triangle T_1 could either rotate ε clockwise or $-(2\pi - \varepsilon)$ counter clockwise that introduces ambiguity. 54

4.5 Inconsistent rotations: before fixing inconsistent rotations (top) and after (bottom). The color represents the rotation magnitudes of clockwise (red) and counter-clockwise (cyan). 55

4.6 Additional large rotation example. The coloration represents rotation angle calculated from rotation matrix, red and blue color indicate counter-clockwise and clock-wise respectively. The original rotations are inconsistent (left column), after applying our rotation fix the rotations are consistent (right column). 56

4.7 Method of [6] tends to create excess rotations when there is no solution for consistent rotations (top). The fixed results of our method (bottom). The color indicates the rotation magnitudes of clockwise (green) and counter-clockwise (purple). 57

4.8 Making rotation consistent. (a) The triangulation of a polygon. (b) The graph of rotation angles β that has equivalent topology to (a). (c) We start from one boundary rotation such as β_1 . (d-f) We traverse the graph and fix the inconsistency. 58

4.9 Blending of the human and airplane using line segment based interpolation. 59

4.10 Generating undesired distortion by reversing the line segment direction between left elbow and hand in Fig.4.9. 59

4.11 Additional example of shape interpolation using line segment based interpolation. 60

4.12 Shape deformation with sparse control points using MLS. 61

4.13 Fold back happens when two parts approach to each other. 62

4.14 Image deformation with dense control points using MLS. 62

4.15	Additional example of shape interpolation using MLS.	63
4.16	Rigid shape interpolation.	64
4.17	Additional example of blending shapes using rigid shape interpolation.	65
4.18	Computing consistent rotations by starting from the 1 st triangle. (a) The boundary rotations of the swirl-like shape to stick, (b) human to butterfly.	67
4.19	Fixing inconsistent rotations for swirl-like shape to stick-like shape by starting from the (a) 5 th and (b) 15 th triangles respectively.	69
4.20	Shape transformations with rotation fix that starts from the vertices at the (a) 1 st , (b) 5 th , and (c) 15 th triangles. The case of (b) and (c) need a larger amount of total rotation than (a).	70
4.21	Additional consistent rotation examples using our method.	71
5.1	Human motion capture with Kinect and an optical motion capture system.	75
5.2	Posture budgeting: We can shrink up to 40% of the training data while the mean error almost remains constant.	76

- 5.3 Overview of the local mixture of GP models. (a-b) We capture postures by the *MOCAP* system and Kinect at the same time to generate the training samples. (c) At the training stage, we partition the samples into $Q = 4$ local regions by K-means and learn Q local GPs with $\mathcal{S} = 10$ training samples independently. (d) During prediction, we extract feature of the $i^{th} = 7$ (left hand) joint, $x_*^7 = (\tilde{x}_*^7, N(\tilde{x}_*^7))$. (e) For a given test sample, x_*^i , shown in red star, we find the nearby $L = 3$ local models by similarity measurement defined in (5.10). (f) We compute the local predicted mean by the l^{th} local GP model and then generate the weighted mean prediction $\mu(x_*^i)$ using L nearby local models given by (5.13). 81
- 5.4 Data set I: postures from Kinect and their corresponding reconstructed postures. In each picture, the upper half shows the RGB and depth images, in which the blue skeleton is the tracked results from Kinect. The lower left and right parts represent the 3D Kinect posture and our reconstructed posture respectively. (a) Bending over; (b) Crossing arms; (c) Rolling hands forward and backward; (d) Rolling hands up and down 92
- 5.5 Data set II: postures from Kinect and their corresponding reconstructed postures. In each picture, the upper half shows the RGB and depth images, in which the blue skeleton is the tracked results from Kinect. The lower left and right parts represent the 3D Kinect posture and our reconstructed posture respectively. (a) Clapping hands; (b) Bending leg; (c) Golf swinging; (d) Waving left hand. 93

5.6	The perceptual score for the correctness of postures from Kinect, Shen et al. 2012, Zhou et al. 2014, proposed method, and an optical motion capture system.	95
5.7	Influence of the training size and the number of local GP models on the 3D joint reconstruction error.	96
5.8	Trajectory of the left hand when performing golf swinging motion. . .	97
5.9	Examples of the reconstruction error of one joint across frames. The green curve corresponding to our method, brown curve is the reconstruction error of randomized forests. (a) Bending leg motion. (b) Bending over motion.	100
6.1	The setup of the interactive shape morphing system.	104
6.2	Prototype of our interactive shape morphing system	105
6.3	Additional example of the interactive shape morphing system.	106
6.4	The compatible triangulation between a shape with self-occlusion (a) and target shape (b)	107
6.5	Examples of shape interpolation with self-occlusion.	108
6.6	Additional example of shape interpolation with self-occlusion.	109
6.7	Overview of compatible triangulation for shapes with self-occlusion. Our inputs are (a) The source shape with self-occlusion and (c) The target shape. (b)We introduce calibration shape. (d-e) We build the compatible triangulation between the calibration shape (b) and target shape (c). We deform the calibration mesh (e) into the source shape with occlusion (a) using the reconstructed joint positions estimated from our posture reconstruction method.	111

6.8 The triangles are not stored in order for compatible triangulation of the source (a) and target shape (b). 112

6.9 Collision detection and depth adjustment. Without appropriate depth assignment, one can see interpenetration (b). We detect overlapping regions and adjust depth on the fly (c). 113

6.10 Shape interpolation with self-occlusion enhanced using posture information. 114

6.11 Additional shape morphing example I: enhancing shape interpolation with self-occlusion. 115

6.12 Additional shape morphing example II: transforming a man to one wolf beast. 116

6.13 Additional shape morphing example III: transforming a man to one bat monster. 117

7.1 Turning around motion. (a) The RGB image of turning around motion (facing backward); (b) The tracking result of Kinect corresponding to (a) (facing forward). 123

List of Tables

3.1	Computational complexity: the main computational cost of our method is computing the link paths, where N is the total number of boundary vertices of source polygon P , C_p is the number of concave vertices of P , L and H are the number of sub-polygon pairs created by Liu et al. and our method, N_i and S_i are the number of boundary vertices and the number of Steiner points of the i -th sub-polygon respectively.	35
3.2	Quantitative comparisons between triangulation quality	39
5.1	Computational complexity: The main computational cost of our method is the offline learning while the incremental updating is fast. .	86
5.2	The number of frames for each type of training motion database used in our method, Shen et al. 2012, and Zhou et al. 2014.	98
5.3	Reconstruction error of Kinect, Shen et al. 2012, Zhou et al. 2014, and the proposed method on the testing data sets.	98
5.4	Reconstruction error of randomized forests and our method using five-fold cross validation (cm).	100
5.5	Reconstruction error of the proposed framework with different constraint terms.	101

List of Algorithms

3.1	Compatible decomposition of the source and the target polygons . . .	27
5.1	Local mixture of GP models and prediction	84
5.2	Incremental learning of local models	87

Chapter 1

Introduction

Planar shape morphing is the process of smoothly transforming a source shape into a target one [80, 15, 17]. Shape morphing techniques have been widely used in animation and special effects packages, such as Adobe After Effects and HTML5. Computing compatible triangulation tackles the vertex correspondence problem for shape morphing. Previous work [2, 23, 71, 7] has shown that computing compatible triangulations can successfully create smooth transformations for both the boundary and interior of a shape. However, we observed that the majority of existing compatible triangulation approaches may either create a large number of skinny triangles or are too complex for real-time shape morphing. We thus want to study how to generate compatible meshes with few long thin triangles and a small number of Steiner points, which enables smooth transformations from one shape to another. In order to determine the vertex path of each vertex in the mesh, we employ the rigid shape interpolation method that preserves the local rigidity during morphing process.

Recent advances in depth camera based motion tracking devices such as the Microsoft Kinect has enabled efficient human-computer interaction using body movement. We use Kinect as the input device and capture the source shape of

the user in the scene. In order to deal with the shape with self-occlusion, we propose a new method for accurately reconstructing human postures using Kinect. With the estimated human body joint positions, we deform a calibration pose into the user performed pose with self-occlusion, which implicitly builds the compatible triangulation between the source shape with self-occlusion and the target shape in the database.

Thus, we want to build an interacting system that can blend the user's shape and other shapes using Microsoft Kinect as the input device of source shape. We use compatible triangulation to build the bijection between the user and the other shapes for both the boundary and interior points, see Chapter 3. We apply rigid shape interpolation method to our system due to its natural transformation and capacity for texture mapping, see Chapter 4. In order to handle the posture with self-occlusions, we leverage the joint position using posture reconstruction method, see Chapter 5. Fig 1.1 shows our system design. The user stands in front of Kinect and then we could extract the shape of the user using Kinect SDK in step 1. In step 2, we compute compatible triangulation between the human and the shape in the database where we have predefined many shapes. We apply rigid shape interpolation algorithm to transform a source shape into the target one in step 3. For complex motion with self-occlusion, we utilize the posture information to generate sensible transformation detailed in Chapter 5 and 6.

The calibration shape in Fig 1.1 is used to build compatible triangulation for shapes that involve self-occlusion. For example, we perform morphing between the human and the butterfly, then the user could see that the human body is gradually transformed into a butterfly.

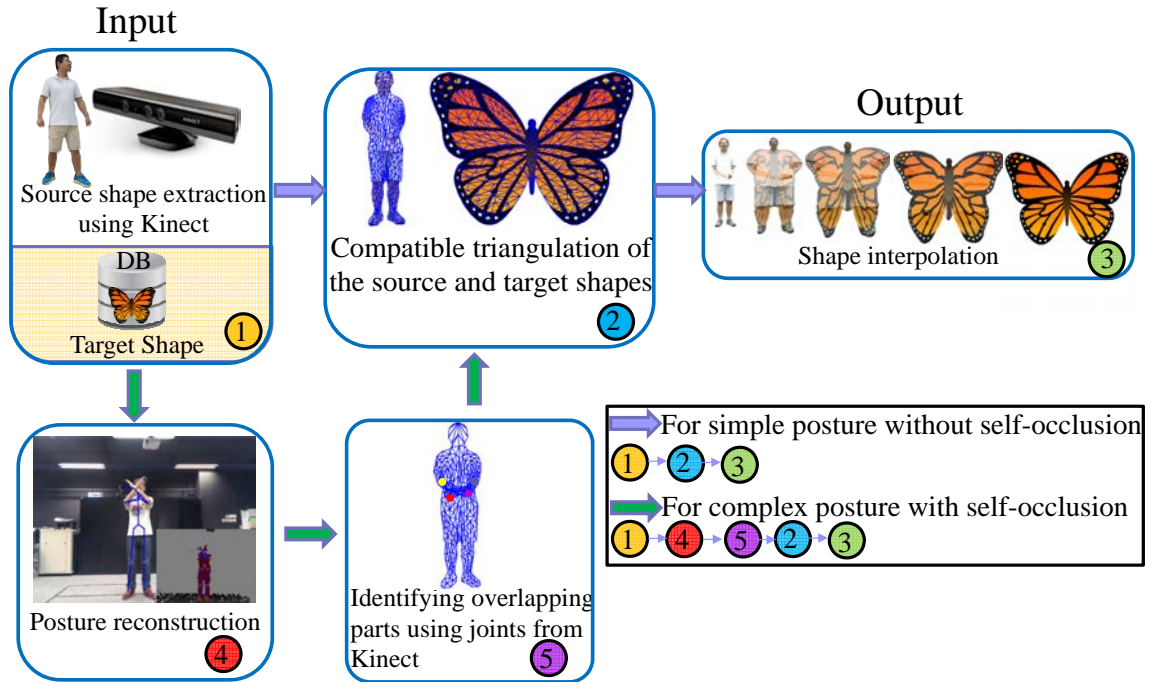


Figure 1.1: Overview of the system design.

1.1 Compatible Triangulation

Two triangulations are compatible if they have the same combinatorial structure, i.e., if their face lattices are isomorphic. However, in many situations, compatible meshes can be generated only if additional points (Steiner points) are added. Thus, one challenge of building compatible triangulation is using a small number of Steiner points such that we can reduce the shape morphing complexity. Another challenging problem of computing compatible triangulation is to avoid the generation of some long thin triangles, which may cause inconsistent rotation problem and create artifact when applying to shape interpolation algorithms. Therefore, a good compatible triangulation contains a small number of Steiner points and keeps a small percentage of long thin triangles.

Previous work [2, 23, 71, 7] has shown that computing compatible triangulations can successfully create smooth transformations for both the boundary and interior of a

shape. [3] first started the study of compatible triangulations by introducing at most $O(N^2)$ Steiner points, where N is the number of vertices of the polygon. Although the algorithm is conceptually simple, it introduces a large number of Steiner points that increases the morphing complexity. On the other hand, this method generates many long thin triangles, which can result in inconsistent rotations for shape interpolation algorithms such as [2]. [71] constructed compatible meshes based on link paths, which requires a small number of Steiner points, but a high computational cost that is prohibitive.

We observed that the majority of existing compatible triangulation approaches may either create a large number of skinny triangles or are too complex for real-time shape morphing. In this thesis, we propose an efficient framework for computing compatible triangulations of two simple polygons, which are defined as planar shapes with non-intersecting edges that form a closed path. Our method produces compatible meshes with few long thin triangles and a small number of Steiner points, which enables smooth transformations from one shape to another.

We conducted the preliminary research in [43], in which we presented a basic system to construct the compatible triangulations for two simple polygons. Compared with our previous work in [43], our new compatible polygon decomposition algorithm is more flexible that will lead to a better mesh quality with less number of Steiner points as illustrated in Fig. 3.5 and Table 3.2. The method of [43] generates different triangulation results if we start the convex decomposition from the source or target polygon. However, our method always produces the same triangulation results even if it starts from different directions. This is because we consider the source and target polygon at the same time using common valid vertex pairs. Generally, our algorithm is faster than that of [43], please refer to Section 3.5 for more details. We

have conducted extensive experiments to analyze the influence of the mesh quality on shape morphing.

The major contributions of the compatible triangulation method in this thesis are summarized as follows: First, we propose a new algorithm to calculate compatible polygon decomposition based on the common valid vertex pairs, which results in flexible decompositions of the source and target polygons. Second, for each iteration, we choose one common valid vertex pair that can maximize the minimum interior angle to compatibly partition the source and target polygons, which increases mesh quality. The increase in the ratio of regular triangles leads to a smoother transition during shape morphing and texture mapping. Third, we propose a new mesh quality evaluation metric to measure the quality of a mesh generated from the shape morphing algorithm. Lastly, our framework of compatible triangulations is simple to implement and computationally efficient.

1.2 Planar Shape Morphing

Planar shape morphing, also known as shape blending, aims at smoothly transforming a source polygon into a target polygon [80, 15, 17]. 2D morphing techniques have been used widely in animation and special effects packages, such as Adobe After Effects and HTML5. With the work on digital heritage, not only an ancient painting can be preserved and appreciated in a virtual reality environment, but also objects in paintings can be animated by shape morphing techniques to provide a more vivid viewing experience to the audience [1]. The key research focus in these applications is to create high-quality transformations that can avoid collapsing or overlapping of polygons during the morphing process.

2D image deformation algorithms such as the rigid shape deformation in [29, 55] have been extensively explored in the research community. Users can manipulate constrained handlers to deform a given image. However, such kind of image warping techniques offer a limited range of transformations. Transforming a shape to a significantly different one is difficult due to the lack of feature correspondence.

Planar shape morphing methods offer solutions to blend two shapes with different silhouettes. Previous attempts to tackle the shape morphing problem is to linearly interpolate the coordinates of each corresponding vertex pair between the source and the target polygons. However, simple linear interpolation sometimes creates intermediate polygons that intersect with each other, resulting in geometrically incorrect transformations. While other image space techniques such as [55, 18] achieve pleasant blending results, they usually suffer from the overlapping problems due to the lack of topology information.

Previous work [2, 67, 6] has shown that rigid interpolation algorithm can successfully create smooth transformations for both the boundary and interior of a shape. However, the rigid shape morphing algorithm may suffer from inconsistent rotations whenever the rotation is more than π . In this thesis, we identify the failure mode related to large rotations that is easily triggered in practical use, and we present a solution for this as well. Experimental results show that our method well handles large rotation for rigid shape interpolation algorithm.

1.3 Posture Reconstruction

Human motion recognition is an important component in interactive applications nowadays. Traditional motion-based systems such as those for dance training are

based on motion capture technology, where the user's movement is captured by an optical motion capture system [13]. While these applications can evaluate user performance with the accurately captured motions, they are not convenient since users have to wear capture suits with reflective markers. Moreover, these devices are relatively expensive and are not affordable for home use.

Depth image based motion sensing devices such as the Microsoft Kinect [45] serve as an alternative to capture human movement for interactive applications. Kinect is a controller-free device that infers 3D positions of human body joints from a single depth image with the help of a data-driven machine learning algorithm [60]. With such a device, it becomes possible to implement a natural user interface for virtual reality applications and gesture based systems [46]. While Kinect can robustly track the 3D postures of the user, the captured data suffer from poor precision due to self-occlusions and insufficient information provided by the Kinect sensor. Therefore, Kinect based interactive applications usually require the user to face the device so that individual body parts are observable, which greatly limits the system flexibility. In addition, the user has to minimize self-occluded postures, or Kinect would misrecognize body parts. As illustrated in Fig. 1.2, the blue skeleton represents the tracked result by Kinect SDK [45]. We can see the tracked arms are twisted due to self-occlusions. Therefore, it is essential to develop effective posture reconstruction strategies for interactive applications.

The occlusion problem and incompleteness of the tracked joints remain challenging despite the posture reconstruction research proposed in the past years. Generating postures from low dimensional signals is a potential solution for posture reconstruction [12, 42]. However, these methods assume the low dimensional signal to be stable and accurate, while joints tracked by Kinect are not. Hence, applying them



Figure 1.2: Example of an inaccurately tracked posture from Kinect. The blue skeleton is the tracked result by Kinect.

to reconstruct Kinect postures will create unsatisfying results. Shum et al. [62] apply reliability measurement to improve the posture reconstruction process. However, the reconstruction results depend heavily on the similarity between database postures and input ones. The system therefore requires a huge posture database.

In this thesis, we propose a new method using local mixture of Gaussian Process (GP) to reconstruct postures captured from Kinect, where the input motion belongs to one of the action classes in the database. We constrain the solution space such that the reconstructed posture is accurate while maintaining the originality of the input posture from Kinect. We follow [86] to adopt Gaussian Process model as a spatial prior distribution to predict the offset between Kinect and the ground truth, which aims at improving the accuracy of the postures in the case where there is sensor error from Kinect. Furthermore, since reconstructing each posture independently cannot ensure the temporal smoothness of the posture sequence, we follow [86] to introduce

a temporal consistency term to constrain the velocity variations between successive frames. Inspired by [62], we embed the reliability of each joint into the optimization framework to ensure that the reconstructed posture resembles the accurate parts of the Kinect tracked posture. Lastly, we propose a new method based on local mixture of Gaussian Processes to alleviate the cubic learning complexity of a regular GP model such that our system can deal with a large variety of movement. The experimental results demonstrate that the proposed approach is effective in reconstructing a number of motions containing self-occlusions. For example, as illustrated in Fig. 5.4a, our method accurately reconstructs the posture of bending over with a number of joints occluded.

The major contributions of the posture reconstruction algorithm in this thesis are summarized as follows:

1. We propose a new unified framework for posture reconstruction using Kinect. The system optimizes an occluded posture live captured by Kinect, which maintains the correctness of the posture while preserving temporal smoothness between frames. The proposed system performs well with significant smaller training sets comparing with previous work in the field.
2. We propose a new method based on local mixture of Gaussian Processes that partitions training samples into local regions to relieve the cubic learning complexity problem of Gaussian Process. With the proposed framework, the prediction speed is enhanced as only a few local models are considered for each input posture. It also enables incremental updating of local models in real time, which enhances the likelihood to adapt to the postures that are different from those in the database.

Compared with our previous work [85], we have significantly improved the spatial

prediction algorithm. Firstly, with the newly proposed method based on local mixture of GP models, our method generates postures of similar quality to that of [85] with significantly less training data. Secondly, we design a new algorithm to incrementally update a specific local Gaussian Process in real time, which enables the system to adapt to run-time postures that are different from those in the database. Lastly, because of the use of local models, our new framework only needs to consider a few local models that are close to an input posture rather than the whole database. Such an enhancement in efficiency allows us to combine all types of motion as a single database, while in [85] a separate database is built for each type of motion.

1.4 Outline of the Thesis

The organization of this thesis is as follows. In Chapter 2, we discuss related work in compatible triangulation, planar shape morphing, and posture reconstruction. In Chapter 3, we propose a new algorithm to calculate compatible triangulation based on the common valid vertex pairs. In Chapter 4, we offer an efficient algorithm that gives a unique rotation assignment with minimum rotation angle for rigid shape interpolation method. In Chapter 5, we present a probabilistic framework for human posture reconstruction such that we can handle shapes with self-occlusion. In Chapter 6, we present an interactive shape morphing system using the techniques discussed in the thesis. We conclude this thesis and discuss about future work in Chapter 7.

Chapter 2

Related Work

Planar shape morphing involves two sub-problems: vertex correspondence and vertex path computation [56]. Vertex correspondence determines how the vertex u of the source polygon P matches the vertex v of the target polygon Q . The vertex path determines the trajectory along which vertex u will travel to vertex v . The compatible triangulation approaches tackle the vertex correspondence problem and the shape interpolation methods deal with the vertex path problem. In order to generate sensible transformation, we need to build compatible triangulation with self-occlusion. To do so, we can identify overlapping body parts using joints from Kinect. However, data from Kinect are noisy and unreliable, thus, we employ posture reconstruction method to enhance the quality of joint position obtained from Kinect.

2.1 Compatible Triangulation

Two triangulations are compatible if they have the same combinatorial structure, i.e., if their face lattices are isomorphic. Previous methods for computing compatible triangulations usually fall into two categories: (1) Transforming source and target

polygons into another common space [3, 2, 35]. (2) Iteratively partitioning the source and the target polygons until both inputs become a set of triangles [72, 25, 71, 7].

2.1.1 Common Space based Compatible Triangulation

[3] constructed the compatible triangulations by overlaying the triangulations of the source and target polygons in a convex polygon. The intersections of the two triangulations built a piecewise-linear homeomorphism, which introduced a large number of Steiner points. To solve this problem, [2] employed Delaunay triangulations to reduce the Steiner points. [35] proposed another method that the number of Steiner points can be determined by the number of inflection vertices. While their method can reduce the number of Steiner points, the algorithm sometimes results in Steiner points on the edge of polygon. Furthermore, although these methods are conceptually simple, they require high computational cost and are not suitable for real-time applications.

2.1.2 Divide and Conquer based Compatible Triangulation

[25] used the divide-and-conquer method to iteratively partition the source and target polygons. Their algorithm introduced a small number of Steiner points by using link paths. However, their method is not suitable for polygons with a small number of vertices. [71] simplified the algorithm of [25] and they proposed a new remeshing method to greatly improve the mesh quality by adding a few Steiner points. Their algorithm requires implementation of many data structures and algorithms in [72] that makes their method algorithmically complex. [7] proposed a new way of finding compatible link paths. Based on this new link path generation algorithm, they used a similar scheme as in [71] to compatibly partition two polygons. Although their

algorithm of computing link paths is faster than that of [71], the proportion of regular-shaped triangles (as opposed to long thin triangles) still needs to be improved.

In this thesis, we propose a new framework to construct the compatible meshes of two simple polygons. Our method draws inspiration from [19], which uses barycentric coordinates to map a spatial surface triangulation to planar triangulation. However, [19] demands that every Steiner point of the target polygon Q must be a strict convex combination of its neighbors, which cannot always be satisfied in practice. As a solution, we propose an efficient convex decomposition algorithm that partitions the target polygon Q into a set of convex polygons such that we can solve the compatible mapping from a sparse linear system.

2.2 Planar Shape Interpolation

Shape interpolation determines the vertex trajectory along which each source vertex will travel to each corresponding target vertex. Shape interpolation approaches usually fall into two categories, image space and object space interpolation. Image space shape interpolation uses different features, points or line segments for example, to discretize the image. Object space shape interpolation applies explicit polygons to represent a 2D object.

2.2.1 Image Space Shape Interpolation

Image space shape interpolation uses different features, points or line segments for example, to discretize the image. Basically, we could employ many kinds of image space morphing methods to warp the image such as feature based deformation [8], skeleton based deformation [39], free form deformation(FFD) [38, 44].

Beier and Neely [8] used multiple pairs of line segments (one specified the source image, the other one specified the destination image), which defined the local coordinate system for the mapping. They demonstrated a complex transformation between two different faces with multiple pairs of lines. However, they noted this method may sometimes encounter some unexpected distortion due to the unforeseen combination of different line segment pairs; the user would have to check where this undesirable interpolation comes from and solve it by adding or deleting some pair of line segment. Schaefer, McPhail [55] presented a closed-form deformation method based on Moving Least Squares. Although this method is fast, the algorithm may generate some white space when two control points approach to each other. Lipman, Kim [41] introduced a new warping method using 4 control points. They aimed to keep nice conformal distortion while preserving the scaling and local bijection. However, the deformation was restricted to the region of interest, and users cannot simultaneously deform two regions that have an overlapping edge.

Interpolating boundary curve is another topic of shape morphing. [31] represented curves by sequences of symbols. The curve morphing problem is formulated as computing a weighted mean of two strings. [68] introduced a square-root velocity representation for analyzing shapes of curves, which can generate natural deformation along the geodesic path. [82] proposed a new structure called *part figure* to represent the shape. Their method can create smooth transition between the source and target shapes by interpolating the *part figure*. However, these curve interpolation methods only solve the boundary vertex path problem and cannot deal with detailed texture.

Generally, image space morphing methods require the source and target object to be of similar shape or geometrical structure [8, 80, 55, 33]; or require laboriously user manipulation on the image. However, the morphing between two very different

shape is a challenging task due to the different geometrical structures.

2.2.2 Object Space Shape Interpolation

Much of work has been proposed for interpolating two shapes using compatible mesh. [2] proposed a method that attempted to preserve rigidity. They separately interpolated the rotation and scale/shear components of an affine transformation matrix, which generated pleasing results with small rotations for most of cases. Inspired by [2], [81] presented a 3D morphing method based on Poisson equation that generated visual pleasing morphing sequences. However, their method suffered from the inherited problem of rigid interpolation methods that the rotations may be incorrectly interpolated. In order to fix this problem, [6] proposed a method to consistently assign rotations. [69] proposed a method that transferred the 3D deformation of a source triangle mesh onto a different target triangle mesh. However, their algorithm is designed for the case where there is a clear semantic correspondence between the source and target. [40] introduced a new type of coordinates for Hermite interpolation that can be applied to shape deformation. Other methods such as [14] trying to preserve certain properties like smoothness and distortion for 2D shape interpolation.

Previous work [2, 67, 6] has shown that rigid interpolation algorithm can successfully create smooth transformations for both the boundary and interior of a shape. However, the rigid shape morphing algorithm may suffer from inconsistent rotations whenever the rotation is more than π . In this thesis, we identify the failure mode related to large rotations that is easily triggered in practical use, and we present a solution for this as well. Experimental results show that our method well handles large rotation for rigid shape interpolation algorithm.

2.3 Posture Reconstruction

With the advancement in real-time depth cameras such as Kinect, human motion recognition and posture estimation have become a popular research topic in recent years. Kinect is based on motion recognition technology proposed by [60], where they use per-pixel classification method to quickly predict 3D joint positions from a single depth image. A number of research domains have benefited from Kinect, such as human-machine interaction [73], natural user interfaces [46], and 3D reconstruction [30]. A recent review on human activity analysis with Kinect can be found in [26]. Bailey and Bodenheimer [5] investigated the perceived differences in the quality of animation generated using motion capture data and a Kinect sensor, which clearly showed that the data recorded from Kinect was of lower quality compared with motion capture data from a Vicon motion capture system. Hence, it is essential to develop an effective posture reconstruction method to enhance the posture quality of Kinect.

2.3.1 Posture Reconstruction from Low Dimensional Signals

Full body postures can be represented by a set of low dimensional signals [12]. Some research work has been proposed to reconstruct a full posture with a subset of the signals. [34] reconstructed human motion from 3D motion sensors on a performer using kernel CCA-based regression. Given the input data from sparse motion sensors, they retrieve similar postures from the motion capture database and transform the low dimensional signal into the full posture space using an online local model. [12] employed a small set of retro-reflective markers to capture performance animation in real time. In their system, the low dimensional control signals from the user's performance were supplemented by a pre-recorded human motion database. At run

time, the system automatically learned some local models from the retrieved motion capture data that were close to the marker locations recorded by the camera. Their system only needs video cameras and a small set of markers, which makes it low cost and practical for home use. However, the majority of markers have to be tracked by the cameras to provide enough information for posture reconstruction.

Liu et al. [42] used a small number of motion sensors to control a full-body human character. They constructed online local dynamic models from pre-recorded motion capture database and used them to construct full-body human motion in a Maximum-a-Posteriori framework, in which the system tried to find the most similar postures from database for reconstruction. [27] adaptively fused inertial and depth information in a hybrid framework for posture estimation. Although these methods can be used to reconstruct postures from low dimensional signals, there is an assumption that these low dimensional signals are reliable and stable. It is therefore not applicable to noisy Kinect data.

2.3.2 Data-Driven Posture Reconstruction

Data-driven approaches usually reconstruct postures by evaluating the similarity between the input posture and a large posture database. [64] presented a data-driven model based method for 3D torso posture estimation from RGB-D image sequence. Although their method can extract the upper body posture of users without an initialization phase, they did not cope with full body posture recovery nor handle the occlusion problem. [61, 62] proposed a unified framework to control physically simulated characters with live captured motion from Kinect by searching for similar postures in a marker-based motion database. They constructed a latent space with a small number of retrieved similar postures, and applied optimization in the space

to reconstruct the input postures. [4] introduced a data-driven approach for full body reconstruction from a depth camera. They proposed an efficient algorithm for extracting posture features from the depth data. However, for fast movements, the proposed system required all five extremities to be visible. [59] introduced an exemplar-based method to correct the postures from Kinect using marker-based motion data.

[83] introduced a model based framework for full body reconstruction from 2D video data using motion capture database as the prior knowledge. The postures were reconstructed in an optimization framework, in which similar motion capture postures were retrieved through nearest neighbor searching. However, the accuracy is not robust because the 2D features projected from 3D motion induce posture ambiguity. [79] solved the reconstruction problem by registering a 3D articulated model with depth information. They formulated the registration problem into a Maximum-a-Posteriori framework to register a 3D articulated human body model with monocular depth via linear system solvers. To tackle the problem of manual initialization and failure recovery, they combined 3D pose tracking with 3D pose detection.

In general, these data-driven methods requires large database as prior, and the reconstruction results depend heavily on the retrieved postures.

2.3.3 Regression based Posture Reconstruction

Structured regression models for posture estimation such as [10] and [9] can model the correlations between multivariate output and input. [10] presented the Twin GP model that employs GP priors to model input and output relations. The output postures were estimated by minimizing the Kullback-Leibler divergence. [9]

optimized an output-associative functional that incorporates outputs and inputs using primal/dual formulations and adapts the model to kernel ridge regression and support vector regression. Shakhnarovich et al. [58] estimated upper body posture, interpolating k-nearest-neighbor postures matched by parameter sensitive hashing. [51] presented an inference machine to estimate articulated human pose. Their method allows learning a rich spatial model and incorporating high-capacity supervised predictors, which results in substantially improved pose estimation performance. Recently, it has been shown that deep learning methods such as [75] and [74] generate high precision pose estimates compared to state-of-art methods.

Gaussian Process (GP) models are flexible probabilistic nonparametric models. [85] presented a new probabilistic framework based on Gaussian Process to enhance the accuracy of the postures live captured by Kinect. Their method can generate high-quality postures even under severe self-occlusion situations. GP models are usually applied to small data sets of a few hundred samples due to its $O(N^3)$ training complexity, where N is the size of training data. In contrast, our incremental sparsification method can efficiently handle large data sets.

Previous attempts to solve the cubic learning complexity problem of GP involve sparse Gaussian Process (SGP) [50, 37, 66] and mixture of experts (ME) [76, 52, 77, 11]. SGP approximates the covariance matrix with a small subset of training data [37] or a set of inducing variables [66]. While SGP can greatly reduce the computational complexity, it utilizes a global voting scheme in which all training samples contribute to the prediction of a new test input. In contrast, ME applies gating network to partition the input space into different subspaces, where each GP expert is trained independently [76]. Compared with a regular GP model, the computational complexity of ME is reduced due to the significantly decreased sample

size in each subspace [65, 49]. However, for simple expert, the gating network has to be more complicated to model the function, which results in a higher risk of getting stuck in local minima or a slower learning process [54].

We follow [85] to reconstruct postures using GP model, which achieves high reconstruction accuracy. However, the high computational complexity of GP makes it not suitable for a real-time application. To solve the problem, we propose a new method based on a local mixture of Gaussian Processes, in which Gaussian Processes are defined in local regions of the state space. Due to the significantly decreased sample size in each local Gaussian Process, the learning time is greatly reduced. At the same time, the prediction speed is enhanced as the weighted mean prediction for a given sample is determined by the nearby local models only.

Chapter 3

Compatible Triangulation

The compatible triangulation methods tackle the first problem of shape morphing, that is, they deal with the vertex correspondence between the source and target polygons. Existing compatible triangulation methods such as [3, 2, 71, 7] may suffer problems such as (1) Generating too many Steiner points that slows down the speed of shape interpolation methods. (2) Creating mesh with many skinny triangles, which may cause numerical problems or lead to inconsistent rotation for rigid shape interpolation method. In Section 3.1, we give an overview of our compatible triangulation algorithm. In Section 3.2, we explain our algorithm for compatibly decomposing the source and target polygons. In Section 3.3, we show how to map the compatible triangulations using a linear system. In Section 3.4, we propose an efficient scheme to further refine the initial compatible triangulation. In Section 3.5, we discuss the time complexity of our algorithm. In Section 3.6, we show various compatible triangulation results using our method.

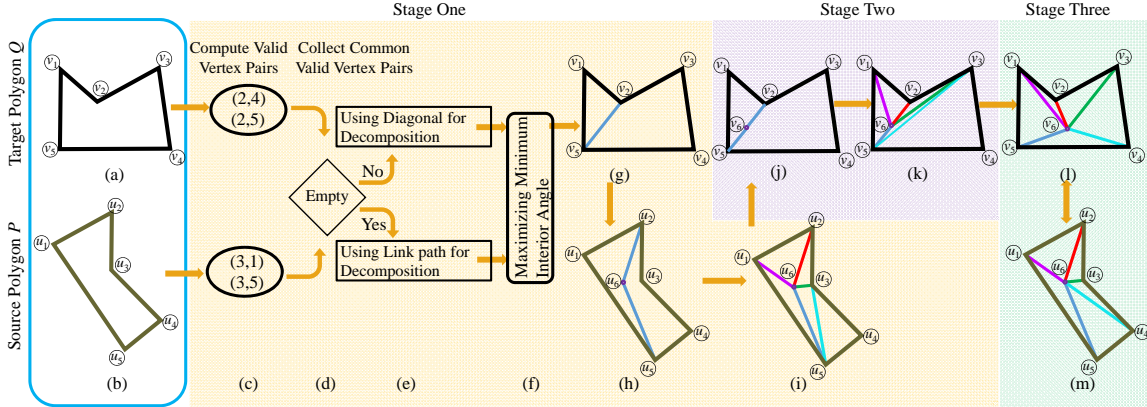


Figure 3.1: Overview of the proposed framework to compatibly triangulate two simple polygons. (a) The target polygon Q . (b) The source polygon P . (c) We compute the valid vertex pairs for both the source and target polygons. (d) We collect the common valid vertex pairs. (e) We use the common valid vertex pair for compatible decomposition if the common vertex pair exists; otherwise we calculate the link path, e.g. the 2-link path between vertex u_2 and u_5 with blue color shown in (h). (f-h) We use the polyline found in (e) that maximizes the minimum angle to decompose the source and target polygons. (i) We triangulate each sub-polygon p_i of source polygon P using Delaunay triangulation. (j) We may need to add some Steiner points on the edge of sub-polygon q_i to keep equivalent topology. (k) We solve a linear system to map the triangulation of sub-polygon p_i onto the corresponding sub-polygon q_i of target polygon Q . (l-m) We finally refine the compatible meshes by operations such as splitting long edges and flipping interior edges so as to improve the interior angles of the mesh.

3.1 Compatible Triangulation Overview

As illustrated in Fig. 3.1 (a-b), the input data of our system are two simple polygons P and Q with corresponding vertices ordered in counter-clockwise. We denote $P = \{U, E^P\}$ and $Q = \{V, E^Q\}$ as the source and target polygons with point set $u \in U$ and $v \in V$, together with the edge set E^P, E^Q respectively. P and Q are assumed to be simple polygons without holes, in which the edges do not cross each other and form a closed contour enclosing each polygon. We define \mathcal{T}_P and \mathcal{T}_Q as the triangulations of polygon P and Q . \mathcal{T}_P and \mathcal{T}_Q are compatible if they have equivalent topology that is defined as:

1. There is an one-to-one correspondence between the vertices of \mathcal{T}_P and that of

\mathcal{T}_Q .

2. There is an one-to-one correspondence between the edges of \mathcal{T}_P and \mathcal{T}_Q , meaning that if there is an edge connecting two vertices of \mathcal{T}_P , then there is an edge connecting the corresponding vertices of \mathcal{T}_Q and vice versa.
3. The boundary vertices of both \mathcal{T}_P and \mathcal{T}_Q are traversed in the same clockwise or counter-clockwise order.

However, in many situations, compatible meshes can be generated only if additional points (Steiner points) are added. Thus, one challenge of building compatible triangulation is using a small number of Steiner points such that we can reduce the shape morphing complexity. Another challenging problem of computing compatible triangulation is to avoid the generation of some long thin triangles, which may cause inconsistent rotation problem and create artifact when applying to shape interpolation algorithms. Therefore, a good compatible triangulation contains a small number of Steiner points and keeps a small percentage of long thin triangles.

Given two simple polygons P and Q with a boundary vertex correspondence as illustrated in Fig. 3.1 (a-b), our algorithm works in three stages. First, we compatibly decompose the source polygon P and the target polygon Q into sub-polygon pairs $(p, q) = \bigcup(p_i, q_i)$ as shown in Fig. 3.1 (c-g), where either the target sub-polygon q_i or the corresponding source sub-polygon p_i is convex. Consider a sub-polygon, e.g. p_i of P , we triangulate p_i using Delaunay triangulation as illustrated in Fig. 3.1 (h-i). Second, we map the triangulation \mathcal{T}_{p_i} of source sub-polygon p_i onto corresponding target sub-polygon q_i using a sparse linear system as shown in Fig. 3.1 (j-k). Third, we refine the compatible mesh to improve the mesh quality shown in Fig. 3.1 (l-m), which is important for high quality morphing in 2D animation, special effects for

movies and texture mapping.

3.2 Compatible Decomposition of the Source and Target Polygons

In order to maintain a small number of Steiner point during the convex decomposition, we want to choose a pair of vertex indices that are visible to each other in both the source and target polygons. The core of our framework is that we propose a new algorithm using common valid vertex pair for partitioning the source and target polygon pairs, which is more flexible to increase the mesh quality with a small number of Steiner points.

In the first phase, we compatibly decompose the source and target polygons, P and Q , into pairs of sub-polygons. In a simple polygon, a vertex $u \in U$ is *convex* if the angle α formed by two edges at u is less than π radians; otherwise u is considered to be *concave*. Our goal is to turn some concave vertices into convex ones through the decomposition and construct pairs of sub-polygons from the source and target polygons such that each of a sub-polygon pair contains at least one convex sub-polygon.

Without loss of generality, we assume the source and target polygons P and Q each to be a simple polygon with N vertices arranged in counter-clockwise order. Here, we label the concave vertices of Q as v_1, \dots, v_C and the convex vertices v_{C+1}, \dots, v_N . Similarly, we label $u_1, \dots, u_{C'}$ as the concave vertices and $u_{C'+1}, \dots, u_N$ as the convex vertices of P . We call a vertex pair (i, j) of P valid if u_i is visible from u_j and at least one of the two vertices is a concave vertex, e.g. $(1, 4)$ is valid as shown in Fig. 3.2. If two vertices are visible to each other but they are not a valid pair, then

it implies that both vertices are convex such as vertex pair $(2, 4)$ as illustrated in Fig. 3.2. A diagonal $u_a u_b$ of P is a line segment that joins vertex u_a and u_b of P and remains strictly inside P . A diagonal such as $u_2 u_4$ shown in Fig. 3.2 that connects two convex vertices is redundant in our compatible decomposition algorithm because it can be removed and the two convex sub-polygons on its sides can be merged into a convex polygon. Therefore, for the construction of a compatible decomposition, we consider only the diagonals that connect two vertices that belong to valid vertex pairs.

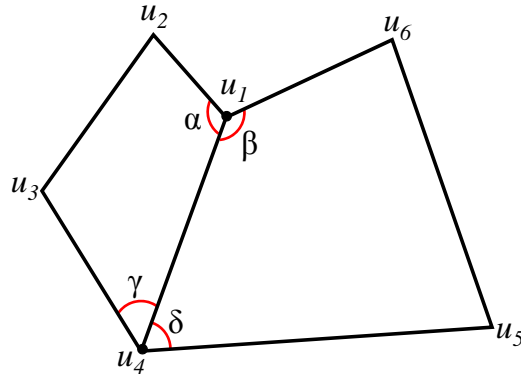


Figure 3.2: A valid vertex pair $(1, 4)$ used to partition the source polygon, which yields four interior angles between vertex u_1 and u_4 .

In some cases, compatible triangulation can only be constructed if Steiner points are added. In order to introduce the minimum number of Steiner points, we need to search for all the potential decomposition combinations in the solution space. Thus, there can be an exponential number of ways of decomposing a simple polygon into convex sub-polygons using the valid vertex pair, which forbids the practical use of the algorithm. Previous work converted the compatible triangulation problem into a common base domain [3, 2] or used a divide-and-conquer methods [70, 7, 43] to iteratively partition the source and target polygons. However, these methods may either need high computational time cost or produce mesh with poor quality.

Therefore, we want to find an efficient compatible triangulation algorithm with an improved mesh quality compared with the existing work.

We start from the source polygon P and find all the valid vertex pairs VP_P for P , similarly, we find the valid vertex pairs VP_Q for the target polygon Q . Among all the valid vertex pairs in VP_P and VP_Q , we collect the common valid vertex pairs $VP = VP_P \cap VP_Q$ that appear in both VP_P and VP_Q . The best partition for P and Q is the common valid vertex pair that generates the maximum minimum interior angle $IntAng$ by:

$$(a, b) = \arg \max_{\substack{v_a, v_b \in V \\ u_a, u_b \in U \\ a \neq b}} \min\{IntAng_P(a, b), IntAng_Q(a, b)\} \quad (3.1)$$

where the $IntAng_P(a, b)$ contains four angles formed by the intersection of the source polygon P and the diagonal $u_a u_b$ that connects a valid vertex pair (a, b) . For example, $IntAng_P(1, 4)$ contains $\angle\alpha$, $\angle\beta$, $\angle\gamma$ and $\angle\delta$ in Fig. 3.2.

Decomposing polygons with Equation 3.1 generates a balanced angle partition for both the source and target polygons, which maximizes the interior angle for both the source and target sub-polygons in the current iteration. [43] only considered a balanced angle partition for the target polygon; however, the source polygon may still generate small interior angles. [71, 7] only considered balanced index partition of the source and target polygons, which are likely to decrease the mesh quality in terms of the proportion of small angles of compatible meshes discussed in Section 3.6.2.

In practice, the common valid vertex pair may not be always available in some cases. For example, as shown in Fig. 3.1(c-d), the intersection of two valid vertex pair sets $\{(2, 4), (2, 5)\} \cap \{(3, 1), (3, 5)\}$ is empty. Here, we apply link path to determine the partition line between two vertices instead of using common valid vertex pair. A

link path between vertex u_a and u_b is a polyline within the polygon that joins the vertex pair (a, b) such as vertex pairs $(2, 6)$ and $(6, 5)$ in Figure 3.1(h) that define a 2-link path between vertex u_2 and u_5 . A minimum link distance for vertex pair (a, b) , $linkDist(u_a, u_b)$, is the minimum number of line segments in a polyline, for example, the minimum link distance for vertex pair $(2, 5)$ in Figure 3.1(h) is 2. We follow [7] to compute the link path with minimum link distance for all vertex pairs in $O(H \cdot N_i^3)$, where H is the number of sub-polygon pairs and N_i is the number of vertices for the i -th sub-polygon. Algorithm 3.1 summarizes our polygon decomposition algorithm in an iterative sense.

Algorithm 3.1 Compatible decomposition of the source and the target polygons

Input: The source and target polygons, P and Q

Output: A decomposition of P , $p = \bigcup p_i$, and Q , $q = \bigcup q_i$, where either p_i or q_i is a convex sub-polygon

convexDecomposition(P, Q)

if P or Q is convex **then**

 exit

end

 Compute valid vertex pairs VP_P and VP_Q

 Find common valid vertex pairs

$VP = VP_P \cap VP_Q$

if VP is not empty **then**

 Calculate the best partition by:

$(a, b) = \arg \max_{\substack{v_a, v_b \in V \\ u_a, u_b \in U \\ a \neq b}} \min\{IntAng_P(a, b), IntAng_Q(a, b)\}$

 Decompose P and Q using (a, b) that creates two sets of sub-polygons:

$\{p_i, p_{i+1}\}, \{q_i, q_{i+1}\}$

else

 Decompose P or Q using link path that creates two sets of sub-polygons:

$\{p_i, p_{i+1}\}, \{q_i, q_{i+1}\}$

end

convexDecomposition(p_i, q_i)

convexDecomposition(p_{i+1}, q_{i+1})

Our method simultaneously decomposes the source and target polygons and we will stop partitioning a sub-polygon pair if either of the sub-polygon in the sub-polygon pair is convex. However, [43] will keep partitioning the target polygon until all the target sub-polygon are convex. The drawback of the method of [43] is that the decomposition results will be very different if we start from the source or target polygon. The choice of the convex decomposition strongly influences the resulting compatible triangulations and the number of Steiner vertices. As shown in Fig. 3.3a, for sub-polygon pair (P_i, Q_i) , our method will stop partition them, while [43] will keep partitioning P_i until all the sub-polygon of P_i are convex. Fig. 3.3a shows that our method generates better partition results than that of [43]. Fig. 3.3b shows that our method computes compatible mesh with better mesh quality.

By this stage, we have compatibly decomposed the source polygon P and target polygon Q into sub-polygons $\{p_i = (U^{p_i}, E^{p_i})\}$ and $\{q_i = (V^{q_i}, E^{q_i})\}$, where (p_i, q_i) is a pair of sub-polygons and either p_i or q_i is convex. We apply Delaunay triangulation as the initial triangulation of a sub-polygon, which can maximize the minimum interior angle with no extra Steiner points in $O(N_i \log N_i)$ [21]. Here, we denote \mathcal{T}_{p_i} as the triangulation of the sub-polygon p_i and aim at constructing compatible triangulation \mathcal{T}_{q_i} of q_i based on \mathcal{T}_{p_i} .

3.3 Compatible Triangulations Mapping

The compatible decomposition process may introduce Steiner points on the link path of either the source polygon P and target polygon Q . In addition, in order to improve the mesh quality, the mesh refinement process detailed in Section 3.4 will create Steiner points within each sub-polygon. Therefore, we have two types of Steiner

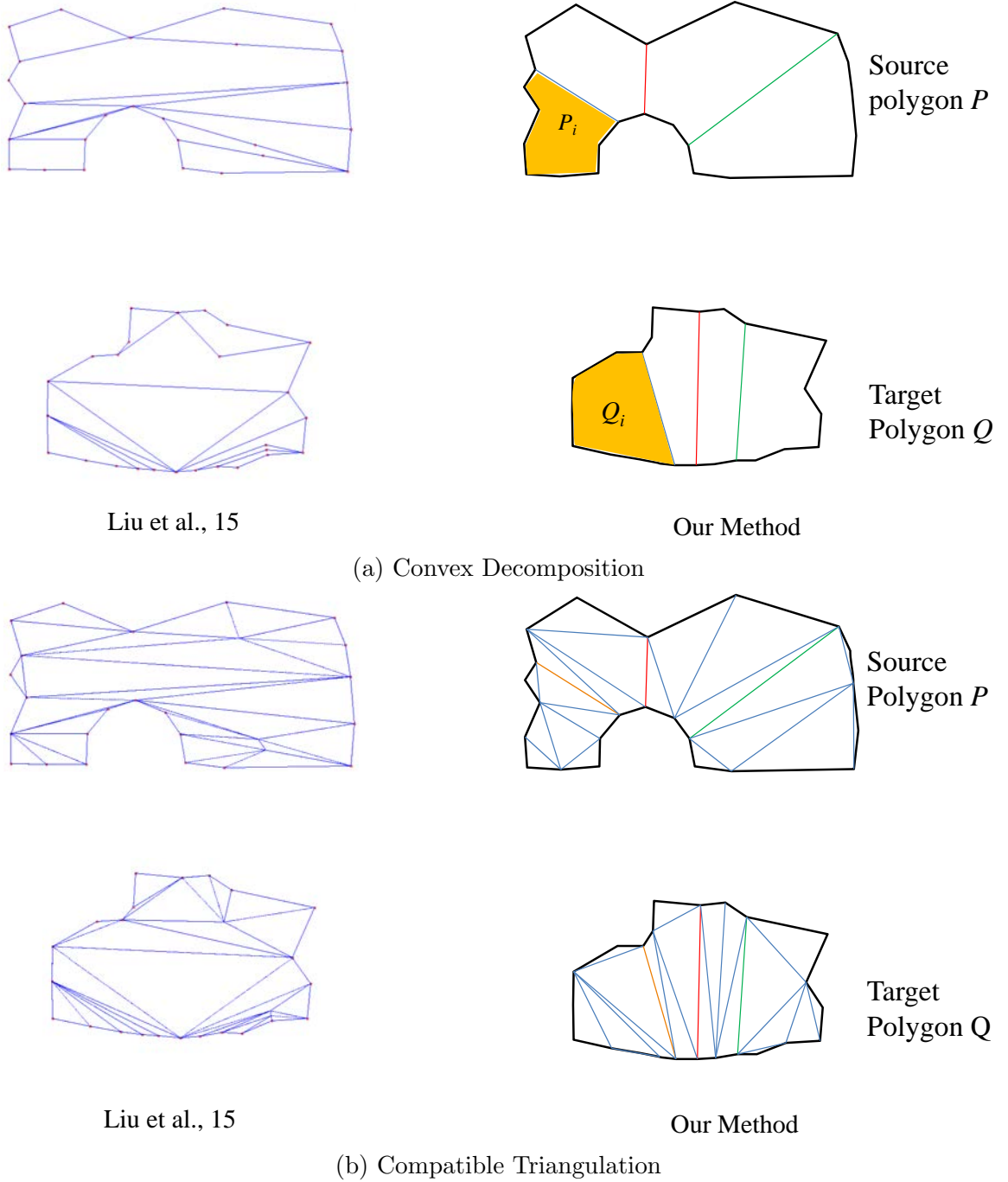


Figure 3.3: Comparison between Liu et al., 15 and our method

points: (1) Steiner points that lie on the link path of source sub-polygon p_i , and (2) Steiner points that lie within p_i . For (1), we map the Steiner points onto the corresponding edges of target sub-polygon q_i based on the simple line-segment-length proportion principle. For (2), we solve the mapping by a sparse linear system.

3.3.1 Mapping Steiner Points on the Link Path of Source Polygon

We denote u_s as a Steiner point lies on the link path between vertex u_a and u_b in the source sub-polygon p_i such as the vertex u_6 for vertex pair (u_2, u_5) in Figure 3.1(h). We add a Steiner point v_s for target sub-polygon q_i on the corresponding line segment $v_a v_b$ based on the linear ratio with the following equation:

$$v_s = \frac{\text{polylineLength}(u_b, u_s)}{\text{polylineLength}(u_a, u_b)} v_a + \frac{\text{polylineLength}(u_s, u_a)}{\text{polylineLength}(u_a, u_b)} v_b \quad (3.2)$$

where $\text{polylineLength}(u_a, u_b)$ is the summation of the length of all line segments on the link path between u_a and u_b .

As shown in Figure 3.1(h), the length of the polyline for vertex pair (u_2, u_5) is $\text{polylineLength}(u_2, u_5) = \text{polylineLength}(u_2, u_6) + \text{polylineLength}(u_6, u_5)$. We would place the vertex v_6 on the line segment $v_2 v_5$ based on the Equation (3.2).

3.3.2 Mapping Steiner Points within the Source Polygon

In this section, we will explain how to map the Steiner points inside the source polygon onto the corresponding locations inside the target polygon. As shown in Figure 3.4, we have to decide how to map the Steiner point u_1 and u_2 onto v_1 and v_2 inside the target polygon. Here, we calculate the barycentric coordinates of u_1 and u_2 . We then compute the proper locations for Steiner point v_1 and v_2 using the barycentric coordinates found in the source polygon.

Denoting $u_j, j \in \{1, \dots, S_i\}$ as a Steiner point that lies within the source sub-polygon p_i , where S_i is the number of Steiner points within p_i . We use barycentric coordinates λ to map the Steiner point u_j of source sub-polygon p_i

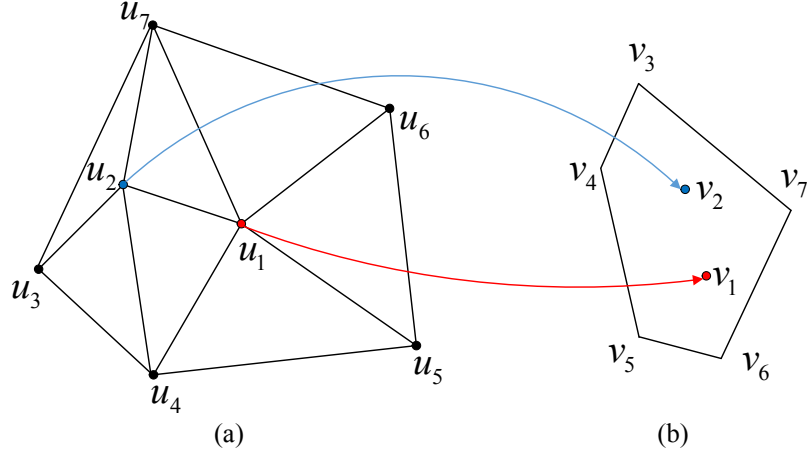


Figure 3.4: Mapping Steiner points within the source sub-polygon onto the target sub-polygon. (a) The source sub-polygon with Steiner points u_1 and u_2 . (b) The corresponding target sub-polygon with unknown Steiner points v_1 and v_2 .

onto the Steiner point v_j of target sub-polygon q_i . Here, we employ Floater’s mean value coordinates [20] to calculate the barycentric coordinates λ . The barycentric coordinates λ of vertex u_j can be seen as a weight of its neighboring vertices, which allows us to generate continues data from these adjacent vertices. We represent the Steiner point u_j as a weighted average of its neighboring vertices:

$$u_j = \sum_{k=1}^M \lambda_{j,k} u_k, \quad \sum_{k=1}^M \lambda_{j,k} = 1 \quad (3.3)$$

where M is the total number of points including boundary vertices and Steiner points for source sub-polygon p_i , i.e. $M = N_i + S_i$.

We now explain how to map the Steiner point $u_j \in U^{p_i}$, $j \in \{1, \dots, S_i\}$ of source sub-polygon p_i onto the corresponding Steiner point $v_j \in V^{q_i}$ of target sub-polygon q_i , where S_i is the number of Steiner points within p_i . We define v_1, \dots, v_{S_i} to be the solutions of linear equations with S_i variables.

$$v_j = \sum_{k=1}^M \lambda_{j,k} v_k, \quad \sum_{k=1}^M \lambda_{j,k} = 1 \quad (3.4)$$

where

$$\begin{aligned}\lambda_{j,k} &= 0, & (j,k) &\notin E^{q_i} \\ \lambda_{j,k} &> 0, & (j,k) &\in E^{q_i}\end{aligned}$$

Note that the barycentric coordinates $\lambda_{j,k}$ can be uniquely determined by Equation (3.3).

We rewrite Equation (3.4) by breaking the summation term into two sub-terms:

$$\begin{aligned}v_j &= \sum_{k=1}^{S_i} \lambda_{j,k} v_k + \sum_{k=S_i+1}^{S_i+N_i} \lambda_{j,k} v_k, j \in \{1, \dots, S_i\} \\ v_j - \sum_{k=1}^{S_i} \lambda_{j,k} v_k &= \sum_{k=S_i+1}^{S_i+N_i} \lambda_{j,k} v_k\end{aligned}\tag{3.5}$$

where S_i is the number of Steiner points within the target sub-polygon q_i and N_i is the number of boundary vertices of q_i .

Denoting $v_j = (x_j, y_j)$ to be a Steiner point within target sub-polygon q_i that we want to solve, Equation (3.5) is equivalent to the following form:

$$Ax = b_1, \quad Ay = b_2\tag{3.6}$$

where $x = (x_1, \dots, x_{S_i})^T$, $y = (y_1, \dots, y_{S_i})^T$, and matrix $A_{S_i \times S_i}$ is in the form:

$$\begin{aligned}a_{j,j} &= 1, j \in \{1, \dots, S_i\} \\ a_{j_1, j_2} &= -\lambda_{j_1, j_2} (j_1, j_2 \in \{1, \dots, S_i\}, j_1 \neq j_2).\end{aligned}$$

This linear system in Equation 3.6 has S_i unknown variables and S_i equations.

The solution to Equation (3.6) is unique as the matrix A is non-singular. We apply LU decomposition to solve Equation (3.6) in $O(S_i^3)$ [48], where S_i is the number of Steiner points within target sub-polygon q_i .

3.4 Compatible Mesh Refining

While the compatible meshes generated by our method introduce a very small number of Steiner points, there may still be some long thin triangles such as the second row in Figure 3.6(a). In practice, we found that these long thin triangles can cause numerical problems such as inconsistent rotations for shape morphing. Therefore, we have to refine the compatible meshes to avoid numerical problems.

To refine the compatible meshes, we apply a variation of the remeshing method in [71]. We only smooth those triangles with small interior angles and long edges. Specifically, we smooth the mesh using area and angle based remeshing, splitting long edges, and flipping interior edges to improve the interior angles. The smoothed results could be found in Figure 3.6(b).

3.5 Method Complexity

In this section, we will analyze the computational complexity of our method. It takes $O(N)$ time to determine the concave vertices and $O(N)$ time to find a valid vertex pair using visibility polygon algorithm [32], where N is the number of vertices of a polygon. Finding common valid vertex pairs using methods like hash table usually requires $O(1)$ time. Thus, the time cost of decomposing the source and target polygons into pairs of sub-polygons is $O(N^2)$. Finding a corresponding link path for a sub-polygon, e.g. p_i in source polygon P , is $O(N_i^3)$, where N_i is the number

of vertices of a source sub-polygon p_i . The Delaunay triangulation can be finished in $O(N_i \log N_i)$. Compatible mapping between a pair of sub-polygons requires solving a linear equation using LU decomposition that leads to $O(S_i^3)$ operations, where S_i is the number of Steiner points of sub-polygon p_i .

Table 3.1 compares the computational complexity between our method and alternative approaches. The main computation of our algorithm is dominated by computing link paths and solving a linear system, i.e. $O(H \cdot \max(N_i^3, S_i^3))$, where H is the number of sub-polygon pairs. In practice, the most time consuming part of our algorithm is building the link path as S_i is often smaller than N_i . Generally, our algorithm is faster than that of [43]. This is because our method simultaneously decompose the source and target polygon and we will stop partitioning a polygon pair if one of them is convex. However, [43] will keep partitioning the target polygon until all the target sub-polygon are convex. Our method is much faster than that of [71, 7] as we solve a small linear sparse system within each sub-polygon pair.

The matrix A in Equation (3.6) is sparse and non-symmetric, thus, we further speed it up by using iterative methods such as Bi-CGSTAB [78]. Here, we apply an open library Eigen [24] to solve the sparse linear system. The compatible mapping process of a sub-polygon pair can be even faster before mesh refinement operations and it can be completed in $O(S_i)$. This is because the Delaunay triangulation can triangulate the sub-polygon p_i with no Steiner points such that we only need to map the Steiner points on the link path as discussed in Section 3.3.1.

Table 3.1: Computational complexity: the main computational cost of our method is computing the link paths, where N is the total number of boundary vertices of source polygon P , C_p is the number of concave vertices of P , L and H are the number of sub-polygon pairs created by Liu et al. and our method, N_i and S_i are the number of boundary vertices and the number of Steiner points of the i -th sub-polygon respectively.

Surazhsky-Gotsman, 04			$O(N^3 \log N)$
Baxter et al., 09			$O(2N^3)$
Liu et al., 15			$O(L \cdot \max(N_i^3, S_i^3)), S_i, N_i \ll N$ $\left\lceil \frac{1}{2} C_\rho \right\rceil + 1 \leq L \leq C_\rho + 1$
Proposed Method	Convex decomposition	Common valid vertex pairs computation	$O(N^2)$
		Link paths generation	$O(HN_i^3), N_i \ll N$
	Linear system computation		$O(HS_i^3), S_i \ll N, H \leq L$

3.6 Experimental Results

In this section, we will show the experimental results and present the comparisons between alternative approaches including [71], [7] and [43]. Qualitative analysis is conducted to evaluate the mesh quality between the proposed method and other alternatives. The experiments are conducted on a Intel Core i3-2350M 2.3 GHZ PC with 4GB RAM.

3.6.1 Compatible Triangulation

To demonstrate the effectiveness of our method, we implemented the as-rigid-as-possible shape interpolation method introduced in [2]. Figure 3.5, 3.6 and 3.7 show some compatible triangulation results and some challenging polygon pairs that are quite different such as the shark and sea horse in the third row of Figure 3.6 and Tai Chi motions in Figure 3.7.

Figure 3.6(a) shows that our initial compatible triangulation contains few long

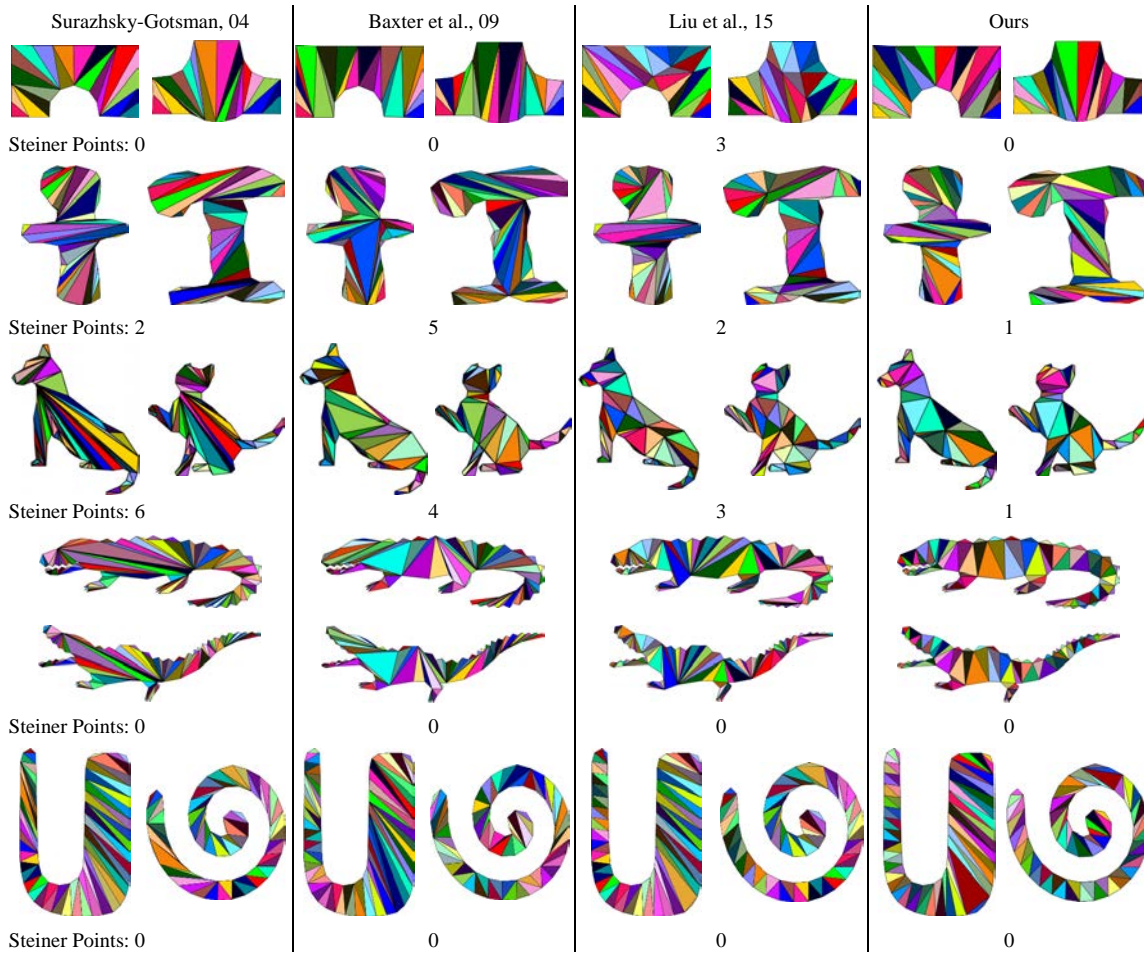


Figure 3.5: Compatible triangulations comparisons. We compare our results with those of [71], [7] and [43]. While we generally use less Steiner points than the others, our algorithm creates high quality compatible mesh in terms of the proportion of long thin triangles.

thin triangles and we may only need to flip some edges of triangles to enlarge the minimum interior angles. Figure 3.6(b) shows that our compatible meshes can be further refined by methods such as splitting long edges and average the area of adjacent triangles.

Given the compatible triangulations of two input polygons, shape interpolation can be applied to create animations showing the transitions from one shape to another. Figure 3.6(b) and 3.7 show some interpolation results using our compatible meshes.

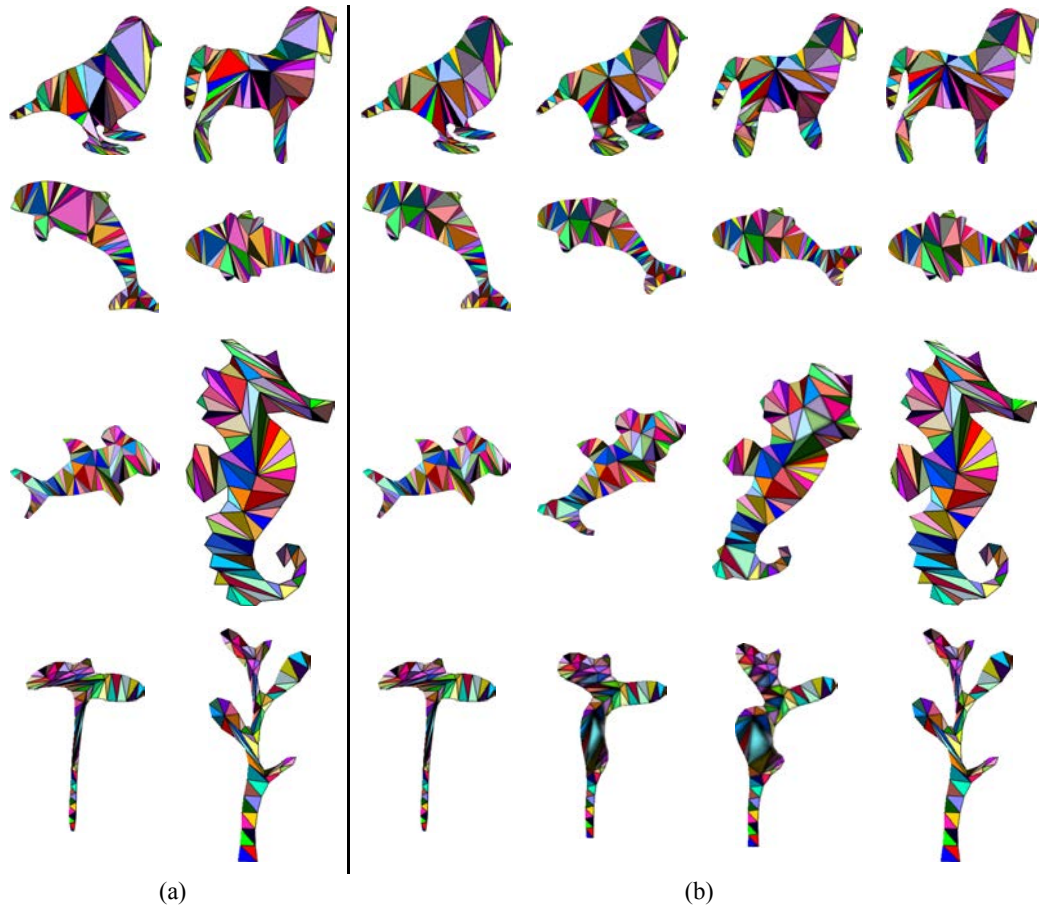


Figure 3.6: Compatible triangulation results. (a) The initial tessellations of two polygons. (b) mesh refinement and morphing. Note that our compatible mesh can be used to blend shapes with large rotations, e.g. shapes in the third row.

3.6.2 Mesh Quality Evaluation

The quality of the compatible meshes would greatly influence the intermediate shapes generated by morphing techniques. In particular, the mesh with those long and skinny triangles would suffer from the inconsistent rotation problem [2, 6].

We evaluate the individual and pair-wise mesh quality. We employ the following criteria to measure the individual mesh quality: (1) minimum interior angle of a given mesh. (2) The proportion of angles that are smaller than a certain constant value, which are known to be reasonable mesh quality criteria [53]. We measure pairwise mesh quality through the triangle deformation when we transform one triangle into

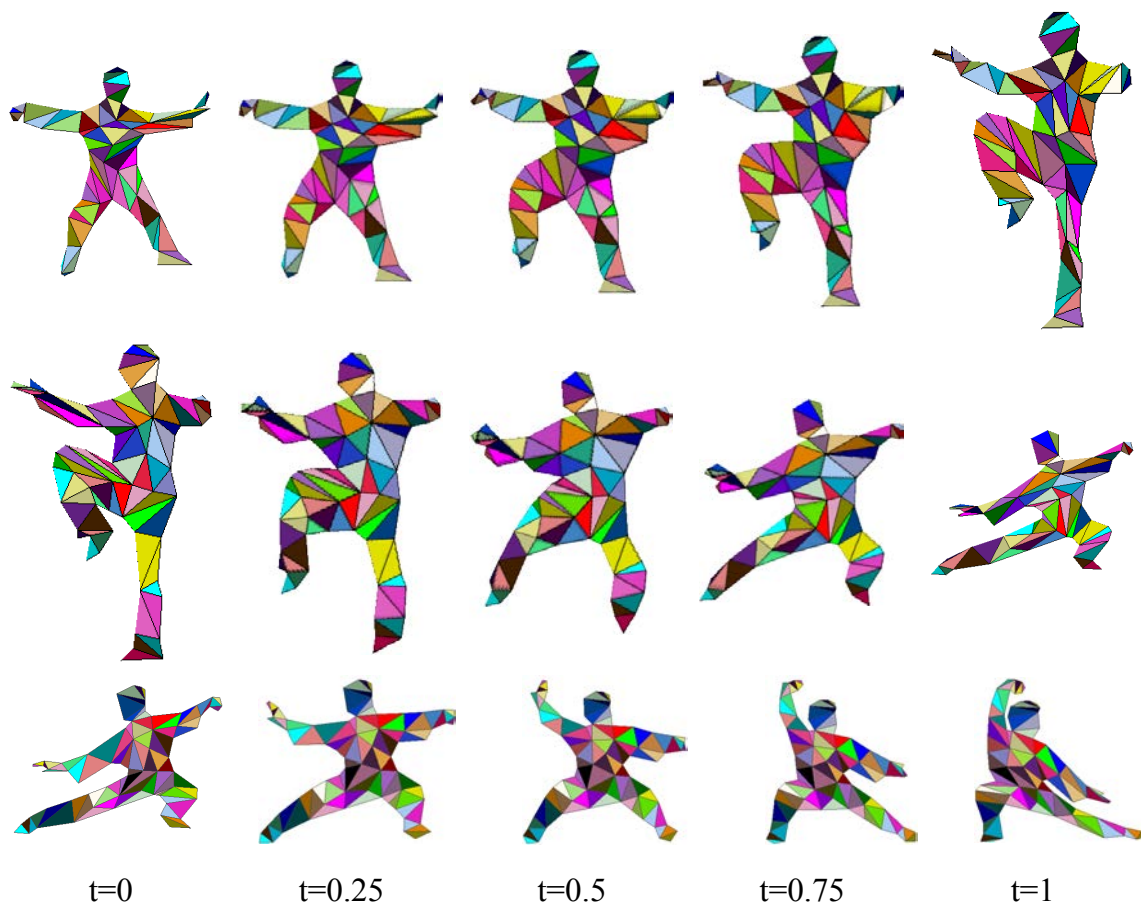



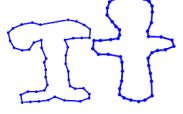



Figure 3.7: Morphing of Tai-chi motions: we adopt the compatible meshes generated by our algorithm to blend Tai-chi motions.

another one. We would like to increase the minimum interior angle of a mesh and decrease the percentage of small angles for individual mesh. Other other hand, for pairwise mesh, we want to create compatible mesh with less triangle deformation such that we can generate smooth texture mapping.

Table 3.2 shows a quantitative comparison between our algorithm and three other alternative methods. [71] tends to create more long thin triangles than the others. Compared with the results of [71], [7] improves the minimum interior angle. [43] enhances the proportion of regular triangles but sometimes introduce a few more Steiner points than [71]. While our results are similar to [71] in terms of the number of Steiner points, our algorithm creates a much smaller percentage of small angles

than [71, 7]. Compared with [71, 7, 43], the minimum angle of our method has been improved greatly while we generally add fewer number of Steiner points than the alternative methods.

Table 3.2: Quantitative comparisons between triangulation quality

Shape	Method	#Steiner Point	Minimum angle	Angles $\leq 10^\circ$	Angles $\leq 15^\circ$	Angles $\leq 20^\circ$	Computation time(second)
	Surazhsky-Gotsman, 04	0	1.6730°	11.57%	16.12%	31.27%	21
	Baxter et al., 09	0	3.3052°	10.61%	14.39%	30.30%	12
	Liu et al., 15	3	3.7557°	5.35%	11.90%	22.02%	7
	Ours	0	6.4161°	8.75%	12.87%	26.93%	3
	Surazhsky-Gotsman, 04	2	0.0441°	27.43%	36.81%	42.36%	24
	Baxter et al., 09	5	0.9779°	21.91%	29.32%	37.96%	14
	Liu et al., 15	2	0.9913°	15.27%	22.91%	32.29%	6
	Ours	1	1.3653°	12.49%	21.08%	26.37%	5
	Surazhsky-Gotsman, 04	6	0.4837°	8.60%	13.03%	20.59%	27
	Baxter et al., 09	4	0.5849°	6.49%	12.42%	18.64%	15
	Liu et al., 15	3	0.6120°	5.29%	11.64%	17.46%	8
	Ours	1	1.6855°	5.18%	9.11%	15.72%	7
	Surazhsky-Gotsman, 04	0	0.0347°	28.96%	35.47%	44.88%	35
	Baxter et al., 09	0	0.0229°	21.45%	29.21%	35.48%	18
	Liu et al., 15	0	0.3294°	16.01%	23.77%	29.54%	6
	Ours	0	5.6835°	3.99%	7.63%	12.11%	4
	Surazhsky-Gotsman, 04	0	0.8893°	12.43%	19.16%	24.13%	29
	Baxter et al., 09	0	2.1933°	10.95%	14.68%	21.89%	16
	Liu et al., 15	0	2.6746°	9.95%	14.18%	20.15%	9
	Ours	0	2.9338°	6.21%	10.94%	15.92%	5

Triangle Deformation Evaluation

We apply the rigid shape interpolation algorithm [2] to transform a source mesh \mathcal{T}_P into the target one \mathcal{T}_Q . Here, we define an edge deformation function to measure the deformation of each triangle face during the transformation. Given the vertices of a source triangle $\mathcal{T}_{P_1} = \{u_1, u_2, u_3\}$ and target triangle $\mathcal{T}_{Q_1} = \{v_1, v_2, v_3\}$, the edge

deformation function is defined as:

$$E_{u_a u_b} = \frac{|\|u_a u_b\| - \|v_a v_b\||}{\|u_a u_b\|}, \quad a, b \in \{1, 2, 3\}, \quad a \neq b \quad (3.7)$$

where $\|u_a u_b\|$ is the length of the edge that connects vertex u_a and u_b .

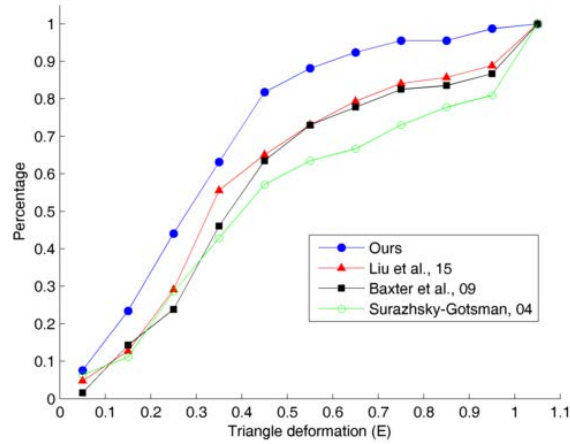
The deformation of a triangle is defined as:

$$E = \frac{1}{3} \sum_{u_a, u_b \in \mathcal{T}_{P_f}} E_{u_a u_b} \quad (3.8)$$

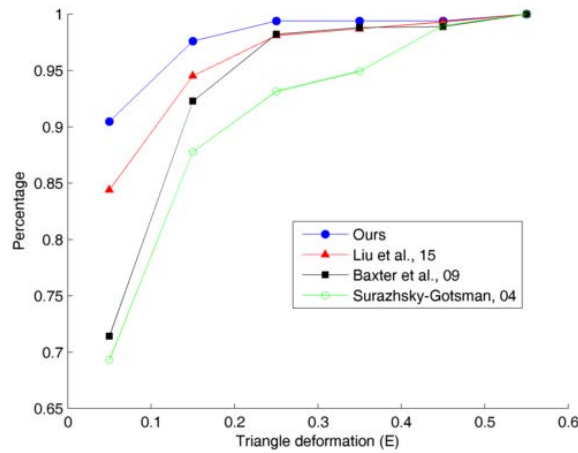
where u_a and u_b are vertices of the f -th source triangle in \mathcal{T}_P .

The deformation function E measures the deformation degree of each triangle. A source triangle \mathcal{T}_{P_f} will experience very small deformation to transform into the target triangle \mathcal{T}_{Q_f} as E approaches 0; Otherwise, a source triangle will experience a big distortion as E becomes larger. For a good compatible triangulation, a larger percentage of small deformation E is preferred, which benefits applications such as shape morphing and texture mapping. Fig. 3.8 shows that our method generally creates compatible mesh with higher percentage of triangles that experience small deformation E ; On the other hand, our method generates fewer triangles that need large deformation during the shape morphing.

It would be more informative to consider the area or angle changes for the triangle deformation. However, it is not sufficient to evaluate the triangle deformation using only the area changes. As show in Figure 3.10, the area of triangle $T_{P_{123}}$ retains nearly the same when we transform triangle $T_{P_{123}}$ into $T_{Q_{123}}$, while the triangle $T_{P_{123}}$ experiences a big distortion to be transformed into triangle $T_{Q_{123}}$. This indicates that only using area changes may not be a good indicator. We can enhance it by adding additional constraints such as the ratio of the minimum, L_{min} , to the maximum,



(a)



(b)

Figure 3.8: Mesh deformation evaluation. (a) Dog and cat. (b) Alligator.

L_{max} , length of the edges of a triangle defined by $q = \frac{L_{\min}}{L_{\max}}$.

On the other hand, measuring the angle changes is especially suitable for scaling transformations. Our edge deformation is able to capture all the triangle changes for rigid deformation, which does not include uniform scaling. However, for non-rigid transformations that involve scaling operation as shown in Figure 3.11, the angle changes could be a good complement for the edge deformation function defined in Equation 3.8

As illustrated in Fig. 3.9, we demonstrate the texture mapping using compatible

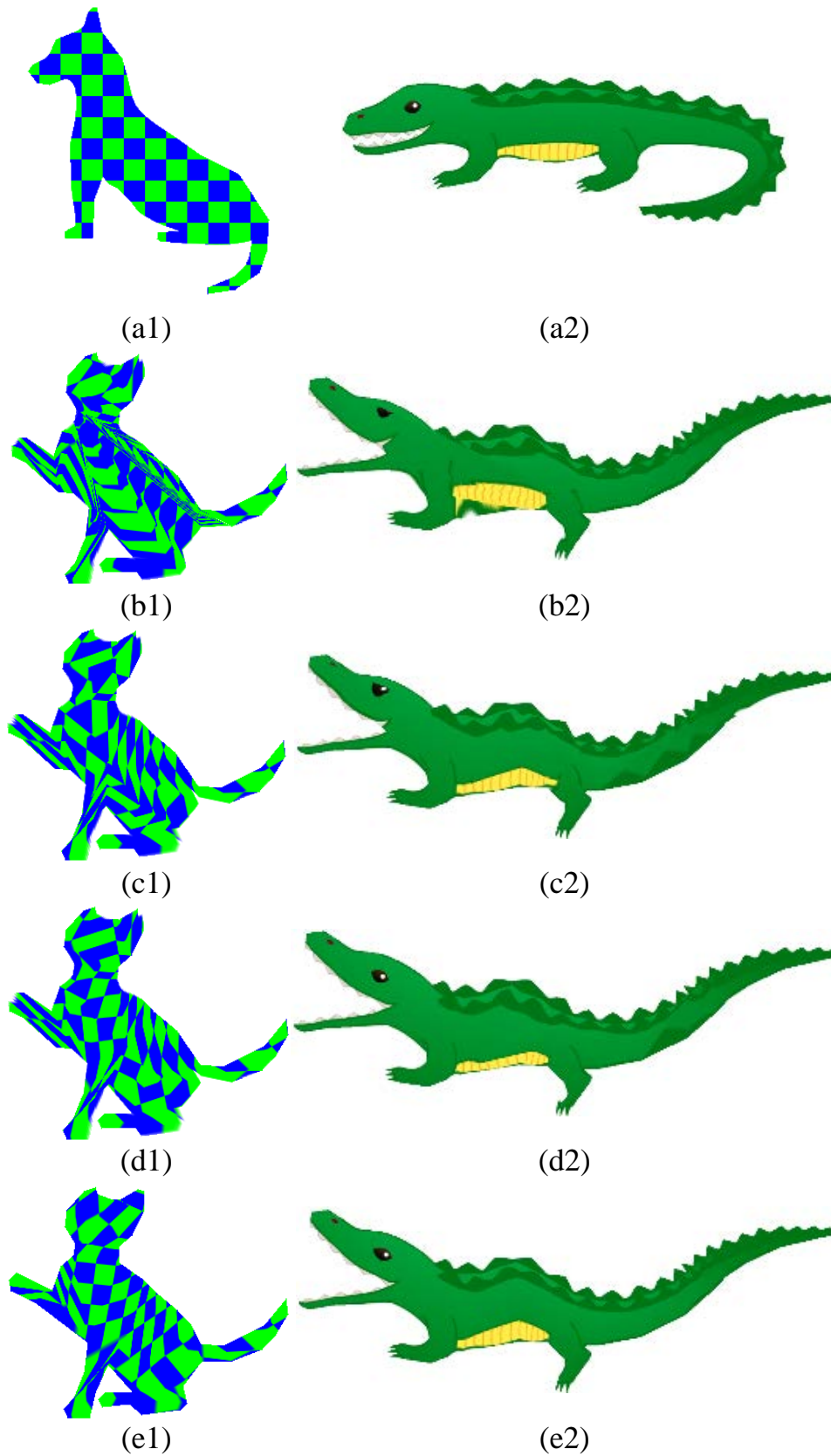


Figure 3.9: Texture mapping comparison. (a1-a2) Source shape with texture. Adding texture to the target shape using compatible meshes generated by methods of (b1-b2) Surazhsky-Gotsman, 04. (c1-c2) Baxter et al., 09. (d1-d2) Liu et al., 15. (e1-e2) Ours.

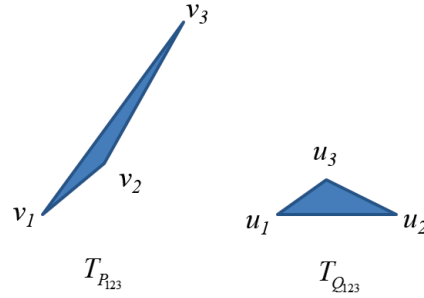


Figure 3.10: Measure the angle changes.

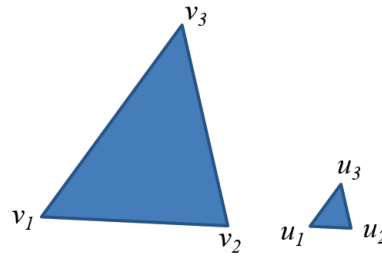


Figure 3.11: Measure the angle changes.

triangulations generated by methods of [71], [7], [43] and ours. The system input is a source polygon with texture and one target polygon without texture. We first build the compatible triangulations of two shapes with alternative approaches, as shown in the third and fourth row in Fig. 3.5. Based on the compatible mesh, we map the texture of a source shape onto the target one. In general, mapping the texture of a shape to another very different one always suffers from the texture stretching. As shown in the dog-cat example in Fig. 3.9 (b-e), nearly all the squares experience some distortion due to the creation of some long thin triangles as shown in the third example of Fig. 3.5. These long thin triangles need large distortion to transform into the target triangles. We can still observe that both [43] and our method preserved some regular squares around the upper body of the cat while our method only generates 1 Steiner point. We then try to map the texture of an alligator between two postures. For the method of [71], we can see some distortions appear

around the abdomen of the alligator. The stretched pattern can also be observed at the back of the alligator for both methods of [7] and [43]. Compared with the other methods, our method generates smoother pattern around the back of the alligator and the deformation of the abdomen makes more sense. This is because our method generates much more regular triangles that only involve small deformation during the shape morphing process, as shown in Fig. 3.8(b).

3.7 Summary

In this chapter, we propose a new method to compute compatible triangulation of two polygons in order to create a smooth geometric transformation between them. Compared with existing methods, our approach creates triangulations of better quality, that is, triangulations with fewer long thin triangles and Steiner points. This results in visually appealing morphing when transforming the shape from one to another.

Our method consists of three stages. First, we use the common valid vertex pair to uniquely decompose the source and target polygons into pairs of sub-polygon, in which each concave sub-polygon is triangulated. Second, within each sub-polygon pair, we map the triangulation of a concave sub-polygon onto the corresponding sub-polygon using linear transformation, thereby generating the compatible meshes between the source and the target. Third, we refine the compatible meshes, which can create better quality planar shape morphing with detailed textures.

Experimental results show that our method can create compatible meshes of higher quality compared with existing methods in terms of fewer long thin triangles and smaller triangle deformation values during shape morphing. These advantages

enable us to create more consistent rotations for rigid shape interpolation algorithm and facilitates smoother morphing process. The proposed algorithm is robust and computationally efficient. It can be applied to produce convincing transformations such as interactive 2D animation creation and texture mapping.

Chapter 4

Planar Shape Interpolation

The shape interpolation methods solve the second problem of shape morphing, that is, they determine the trajectory along which the source vertex u will travel to the target vertex v . In Section 4.1, we review three commonly used shape interpolation methods and determine which method is more suitable to our task. In Section 4.2, we discuss the problem of large rotations for rigid shape interpolation methods and present an efficient solution for this as well. We show the experimental results in Section 4.3.

4.1 Choosing Shape Interpolation Approaches

In this section, we review three commonly used shape interpolation methods: line segment based, control point based shape interpolation and the rigid shape interpolation method. Line segment based and control point based shape interpolation are representative image space shape morphing. The rigid shape interpolation method is an powerful object space interpolation, which can retain the original texture and keep rigidity during the transformation. With the line segments

or control points, it is more obvious for the user to understand what will happen when lines or control points are added and moved. Previous work [2, 69, 67] has shown that rigid shape interpolation methods can maximize the rigidity of a blended shape, which results in realistic transformations.

4.1.1 Line Segment based Shape Interpolation

Line segment based shape interpolation mainly solves the image warping problem. Fig 4.1 shows the idea of using a single pair of line segments. For each point X in the intermediate image, the corresponding point X_1 for the source and X_2 for the destination image are found based on the u and v . Finally, the cross dissolve is employed to get the pixel value of X in the intermediate image.

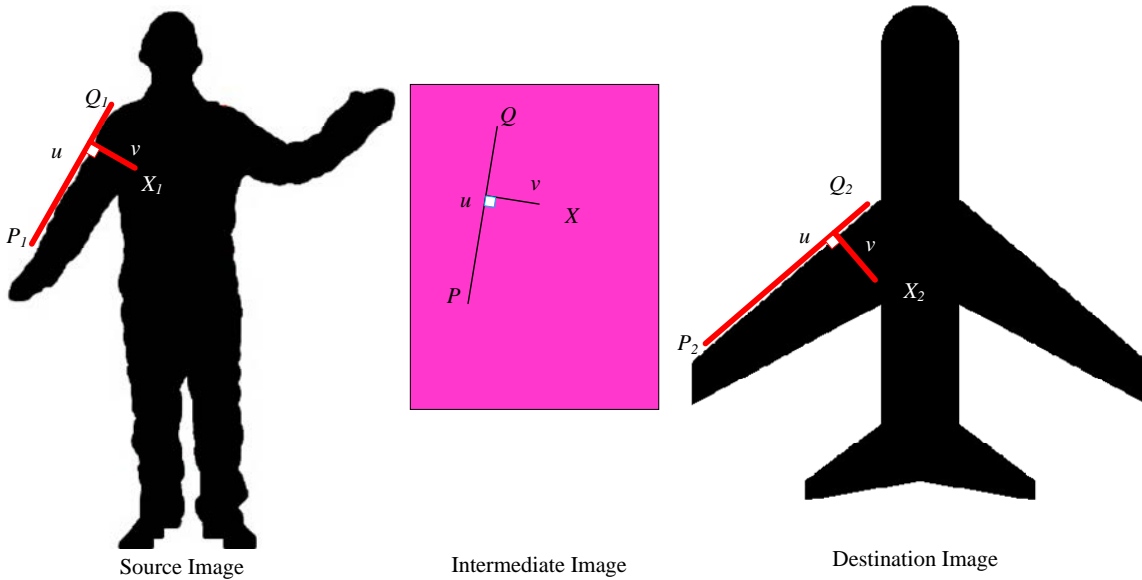


Figure 4.1: Single line-segment-pair mapping

The capital characters in Equations 4.1, 4.2 and 4.3 refer to coordinates of points such that a line segment are a pair of two points such as PQ ; u and v are two scalars, where u is the projection of X along the line segment PQ and v is the distance

between a pixel and line segment.

$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2} \quad (4.1)$$

$$v = \frac{(X - P) \text{Perpendicular}(Q - P)}{\|Q - P\|} \quad (4.2)$$

$$X' = P' + u(Q' - P') + \frac{v \text{Perpendicular}(Q' - P')}{\|Q' - P'\|} \quad (4.3)$$

Fig. 4.2 illustrates warping with multiple line segment pairs. The pixel X' in the source image is a weighted average of X'_1 and X'_2 calculated by two pairs of line segments. The weight is inversely proportional to v .

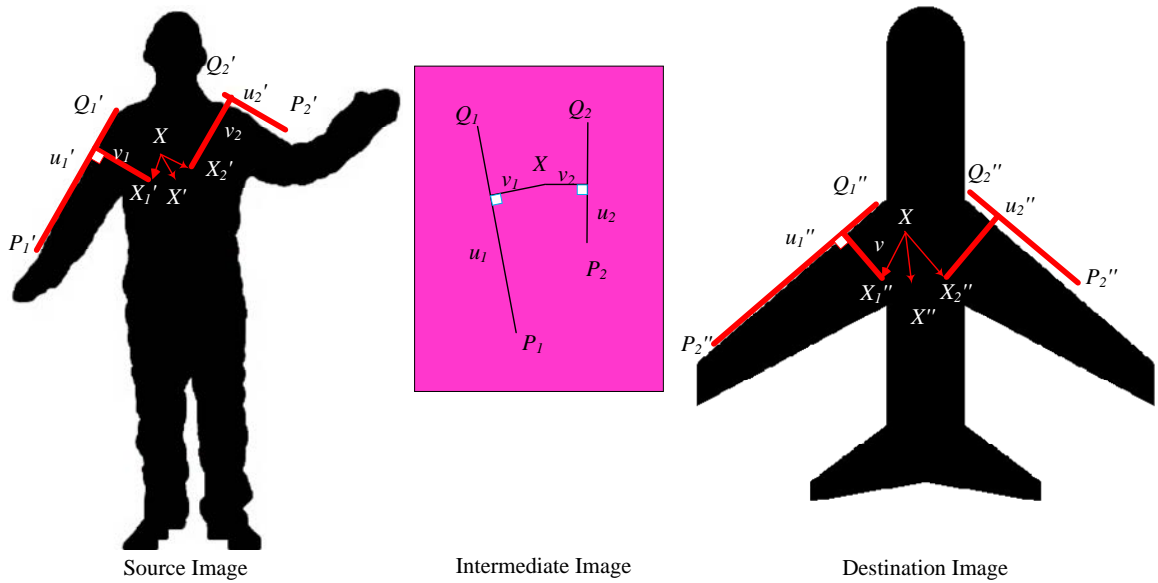


Figure 4.2: Multiple line-segment-pair mapping

4.1.2 Shape Interpolation Using Moving Least Squares

Schaefer et al. [55] proposed a deformation method that can smoothly transform between two images of similar structure using moving least square (MLS). As shown in Fig. 4.3, the user could specify a set of control points at p_i and a new deformed

position q_i of the control point at p_i , based on p_i and q_i the user could generate some realistic intermediate shapes as if they are manipulating the real object.

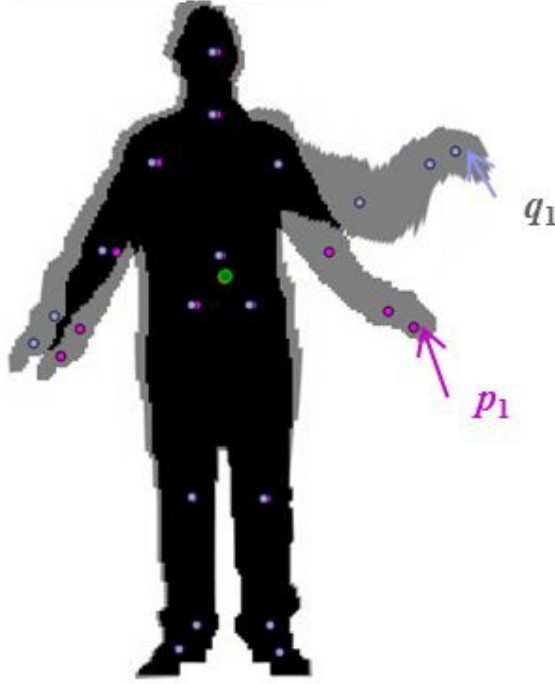


Figure 4.3: Deformation with control points shown in purple

The goal is to find an affine transformation M that could preserve the local rigidity as much as possible. We limit the transformation used in moving least squares as similar and rigid affine transformation. Consider a weighted least square problem for an image, we have the cost function defined in Equation 4.4:

$$\sum_i w_i |l_v(p_i) - q_i|^2, \quad w_i = \frac{1}{|p_i - v|^{2\alpha}} \quad (4.4)$$

where w_i is inversely proportional to the Euclidean distance between p_i and each pixel v in the image, $\alpha > 0$.

We aim to find the transformation $l_v(p_i)$, a linear combination of the transformation matrix M and a translation T , which can minimize the cost function

as:

$$l_v(x) = xM + T \quad (4.5)$$

To minimize the quadratic error of Equation 4.4, set the gradient over free variable to 0, we get the solution for $l_v(x)$, and define $f(x) = l_v(x)$

$$f_a(v) = (v - p_*) \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \sum_j w_j \hat{p}_j^T \hat{q}_j + q_* \quad (4.6)$$

Since the control point at p_i would not change its position, we could pre-compute the term A_j in Equation 4.6 to speed up the algorithm by:

$$f_a(v) = \sum_j A_j q_j + q_* \quad (4.7)$$

where $A_j = (v - p_*) \left(\sum_i \hat{p}_i^T w_i \hat{p}_i \right)^{-1} \sum_j w_j \hat{p}_j^T$.

The matrix M is a general transformation matrix that includes uniform transformation as well as non-uniform transformation, which might sometimes results in unrealistic distortion. Thus, we want to restrict the linear transformation matrix M only include rotation, translation, and uniform scaling by:

$$M^T M = \lambda I, \quad f_s(v) = \sum_i \hat{q}_i \left(\frac{1}{\mu_s} A_i \right) + q_* \quad (4.8)$$

where $A_i = w_i \begin{pmatrix} p_i \\ -p_i \end{pmatrix} \begin{pmatrix} v - p_* \\ -(v - p_*)^\perp \end{pmatrix}^T$, $\mu_s = \sum_i w_i p_i p_i^T$

Previous work [2, 29, 6] shows that transforming shape as rigid as possible could give users the impression that they were manipulating the real object. This rigidity means we need to keep the size of local shape unchanged, thus do not include uniform

scaling by:

$$M^T M = I \quad (4.9)$$

4.1.3 Rigid Shape Interpolation

In this section, we review the rigid shape interpolation method. We follow [2] to solve the rigid shape interpolation by minimizing a quadratic function. Given the vertices of two triangles $\mathcal{T}_1 = \{u_1, u_2, u_3\}$ and $\mathcal{T}_2 = \{v_1, v_2, v_3\}$, they transform \mathcal{T}_1 into \mathcal{T}_2 with an affine transformation $\mathcal{A}\mathcal{T}_1 = \mathcal{T}_2$. The matrix \mathcal{A} can be factorized into a rotation matrix \mathcal{R} and a scale-shear component \mathcal{S} using polar decomposition, i.e. $\mathcal{A} = \mathcal{R}\mathcal{S}$. We then independently interpolate the matrix \mathcal{S} and rotation angle β of \mathcal{R} to compute intermediate transformation matrix $\mathcal{A}(t) = \mathcal{R}(t\beta)((1-t)\mathcal{I} + t\mathcal{S})$, where $t \in [0, 1]$ is time. Finally, finding the vertex path of all triangles can be solved by minimizing a quadratic error between the desired matrix \mathcal{A} and actual matrix \mathcal{B} :

$$\mathcal{E} = \sum_{f \in \mathcal{T}_P} \left\| \mathcal{B}_f(t) - \mathcal{A}_f(t) \right\| \quad (4.10)$$

where $\mathcal{A}_f(t)$ is an affine transformation for the f^{th} triangle, \mathcal{B} is the actual matrix corresponds to $\mathcal{A}_f(t)$, and $\|\cdot\|$ is the Frobenius norm.

This rigid shape interpolation is capable of integrating texture for each intermediate shape. For each pixel PIX inside an intermediate triangle $\mathcal{T}_f(t)$, we calculate its three barycentric coordinates. The barycentric coordinates are applied back to the three vertices of the corresponding source and target triangles, \mathcal{T}_{f1} and \mathcal{T}_{f2} , to calculate the pixel on the original source and target images. We denote the color of pixel PIX within f^{th} triangle for \mathcal{T}_{f1} and \mathcal{T}_{f2} as $color(PIX_1)$ and

$color(PIX_2)$. The color of pixel PIX on the intermediate f^{th} triangle is assigned by:

$$color(PIX) = color(PIX_1)t + color(PIX_2)(1 - t) \quad (4.11)$$

4.2 Assigning Consistent Rotation for Rigid Shape Interpolation

In practical situations, orientation may be interpolated incorrectly by current rigid interpolation approaches such as [2, 81]. Inconsistent rotations can occur anywhere in a shape, either in the interior or on the boundary. To prove this, consider a single triangulated shape as shown in Fig.4.4a and a copy of it rotated ε radians as shown in Fig.4.4b. Now rotating a triangle T_1 to T'_1 has two choices: clockwise or counter clockwise. This will introduce a rotational discontinuity in the neighborhood of triangle T_1 . In this section, we first review previous methods on tackling inconsistent problem, and then we propose an efficient algorithm that gives a unique rotation assignment with minimum rotation angle.

4.2.1 Previous Methods on Tackling Inconsistent Rotation

[16] proposed an algorithm to generate pleasing interpolations but avoid inconsistent rotation. Specifically, for an arbitrarily selected triangle T_1 , select any admissible rotation angle $\alpha(T_1)$. Any time a triangle T has a rotation angle assigned, where the rotation angles lies in the interval $(\alpha_T - \pi, \alpha_T + \pi)$ to its adjacent triangles that have not got a rotation angle yet, until all triangles are exhausted. While this method can assign consistent rotations for triangulation without self-intersection, it need the user to first specify an admissible rotation.

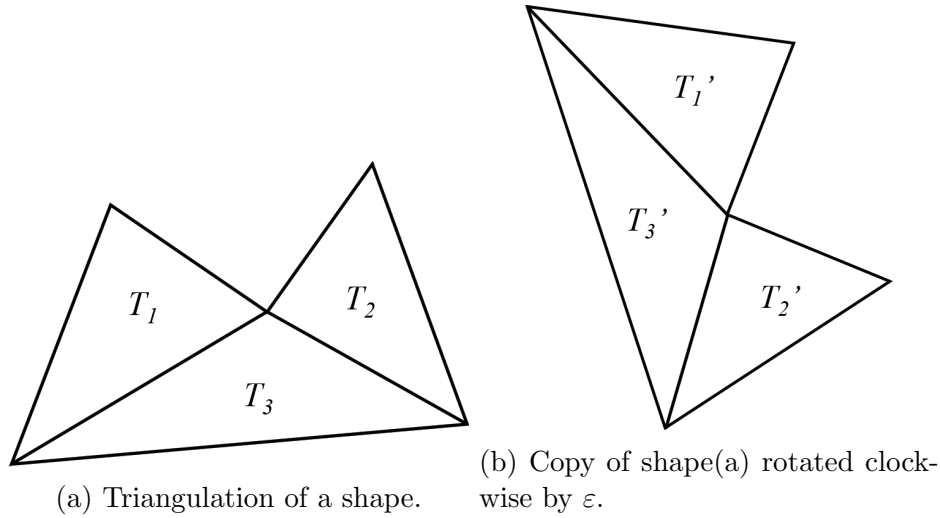


Figure 4.4: The problem of assigning rotations for rigid shape interpolation method: the triangle T_1 could either rotate ε clockwise or $-(2\pi - \varepsilon)$ counter clockwise that introduces ambiguity.

Similarly, [22, 6] design a simple scheme to choose coherent rotation angles. They first choose an arbitrary vertex v_j and compute its rotation angle α_j in $[0, 2\pi)$, and then assign the rotation angle for the next vertex (counter clockwise or clockwise) such that the difference between the two rotation angles is within $[-\pi, \pi)$. This assignment process terminates when all the vertices are visited. However, the resulting rotations are highly influenced by the starting rotation.

Global search is one way for solving discontinuity if there exist such a consistent rotation assignment. Removing discontinuities in sequence of rotations is in fact a classic problem. It is precisely the problem faced when extracting the phase from a discrete Fourier transform. We can employ the global unwrap method to minimize the average discontinuity. We put our rotations into an array and just pass it to such a function. The algorithm scans through the rotation angle and fixes discontinuities globally. In general, global optimization will result in the best solution for many optimization problems. While we minimize the average jump among all rotations using some global unwrap, we cannot guarantee the jump between two adjacent

rotations is less than π . On the other hand, the global optimization methods often take more computation time, which will slow down the shape transformations. Thus, based on the discussion above, we propose a local optimization method for solving the inconsistent rotation problem.

4.2.2 Our Method for Consistent Rotation Assigning

The rigid shape morphing algorithm may suffer from inconsistent rotations whenever the rotation is more than π . As shown in Fig. 4.5(top) and Fig.4.6(left column), some triangles rotate in opposite direction from their neighbors. This problem stems from the ambiguity of extracting the angle from rotation matrix \mathcal{R} . The rotation angle extracting techniques such as *atan2* would just naively return an angle β between $-\pi$ and π , which is the smallest magnitude rotation for each triangle. However, the desired rotation should be $\beta + 2l\pi$, where $l \in \mathbb{Z}$ is an integer.

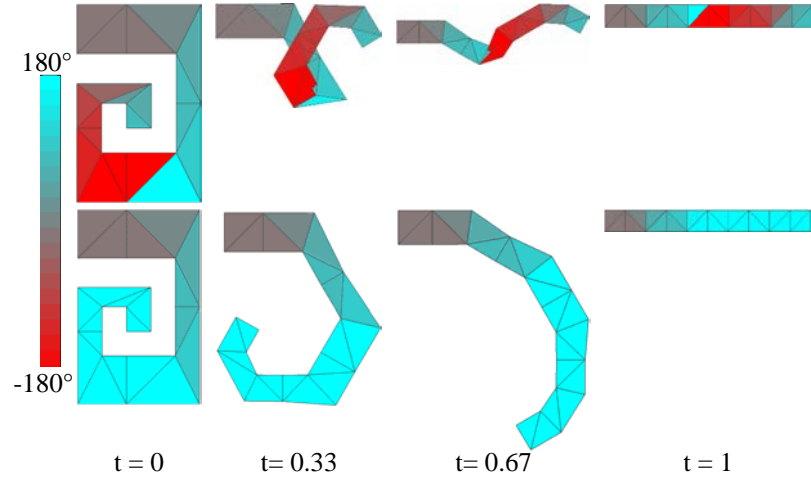


Figure 4.5: Inconsistent rotations: before fixing inconsistent rotations (top) and after (bottom). The color represents the rotation magnitudes of clockwise (red) and counter-clockwise (cyan).

Our input is a set of rotation angles β extracted from rotation matrix \mathcal{R} that lie in the closed interval $[-\pi, \pi]$. We want to generate consistent rotation angles such that

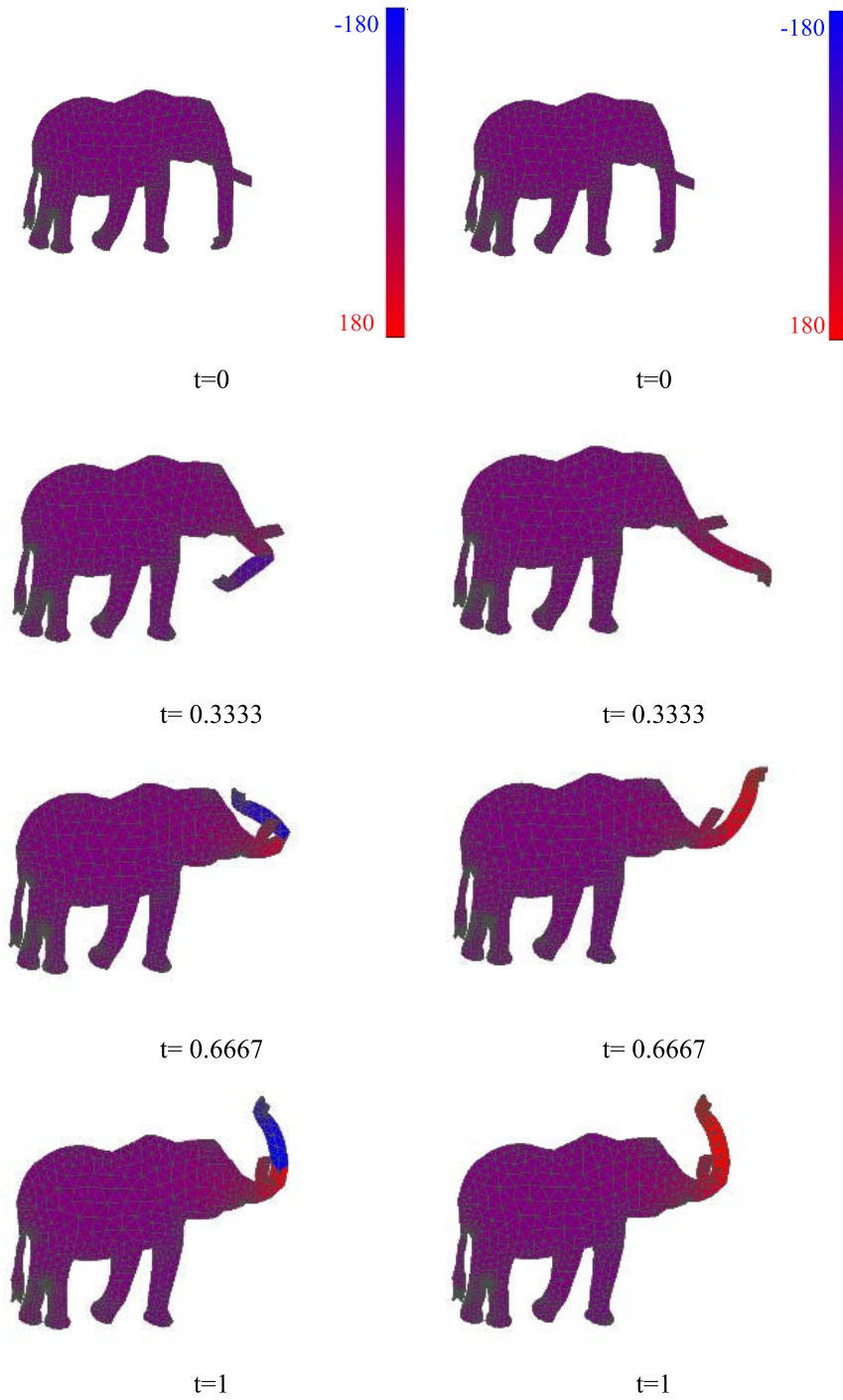


Figure 4.6: Additional large rotation example. The coloration represents rotation angle calculated from rotation matrix, red and blue color indicate counter-clockwise and clock-wise respectively. The original rotations are inconsistent (left column), after applying our rotation fix the rotations are consistent (right column).

the jump between adjacent angles is less than π . Previous work [2, 6] has shown that we may fail to find a solution for inconsistent rotation problem due to the numerical problem caused by the long thin triangles.

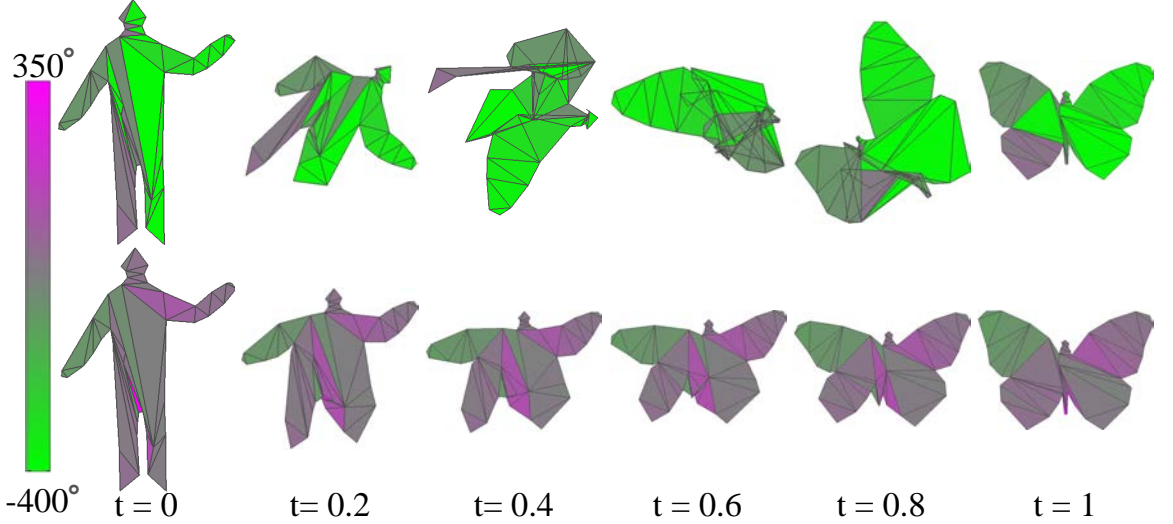


Figure 4.7: Method of [6] tends to create excess rotations when there is no solution for consistent rotations (top). The fixed results of our method (bottom). The color indicates the rotation magnitudes of clockwise (green) and counter-clockwise (purple).

Here, we offer an efficient algorithm that gives a unique rotation assignment with minimum rotation angles, even when consistent rotations do not exist. First, as illustrated in Figure 4.8(b), we treat each original rotation angle β as one vertex of a graph G . Second, we set the vertex connectivity of the graph G the same as the connectivity of the triangles in triangulation $\mathcal{T}_{\mathcal{P}}$ created in Section 3 such as the triangulation in Figure 4.8(a). Lastly, we start from one boundary rotation and examine all its neighbors, fixing any jump that is larger than π by adding $2l\pi$ such as Figure 4.8(c). We keep searching the graph and adjusting any adjacent rotation that is inconsistent with the current rotation until all the vertices have been traversed such as Figure 4.8(d-f).

Specifically, we push the rotations of a triangle on a queue, tagging it as *VISITED* and the others as *UNVISITED*. We start from the triangle with a small rotation and

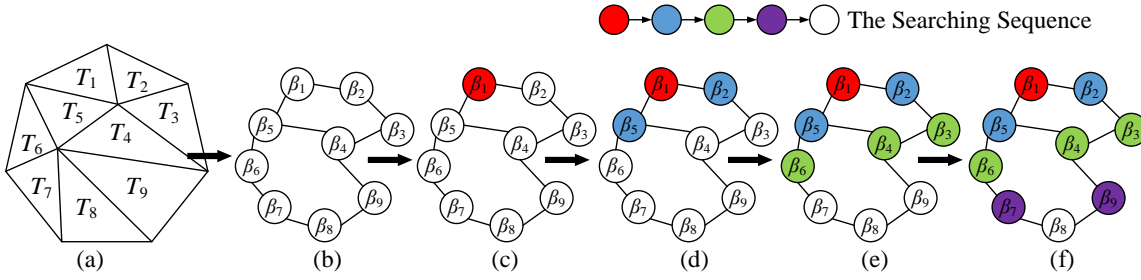


Figure 4.8: Making rotation consistent. (a) The triangulation of a polygon. (b) The graph of rotation angles β that has equivalent topology to (a). (c) We start from one boundary rotation such as β_1 . (d-f) We traverse the graph and fix the inconsistency.

enqueue it as the root node as triangles with small rotation will be very steady. We iteratively dequeue a node and check all its adjacent rotations that are *UNVISITED*. For any adjacent rotations that are inconsistent with the dequeued node, set their tag as *VISITED*, adjust the rotations and enqueue them. The whole process continues until the queue is empty. The corrected results in Fig. 4.5, 4.7 and 4.6 were generated in this way.

During the searching and fixing process, we keep the rotation of the long thin triangle unchanged if there is a jump of more than π . This is because the inconsistent rotations often stem from these long thin triangles, which results in numerical problem for extracting rotation angles. In addition, one triangle lies on the interior of a source polygon usually needs a small rotation, within $-\pi$ and π , to be transformed to the target one. After this process, our method would find a solution to fix these discontinuity such as the results in Figure 4.7(bottom). All the correct rotations in this paper are generated by this efficient scheme. Although we cannot prove our method can always find a consistent rotation assignment, it works well for all the results in this thesis. When finding consistent rotation is impossible, then we should probably look for a different compatible triangulations of the source and target before constructing a high quality morphing.

4.3 Experimental Results

4.3.1 Line Segment based Shape Interpolation

Fig. 4.9 illustrates the blending between an airplane and a human. This method calculates complicated warping using multiple pairs of lines. However, as shown in Fig. 4.10 ($t = 0.66$), it may suffer from undesired distortion caused by some unforeseen combination of line segment pairs. For example, we use the same line segment pairs as Fig. 4.9 but reverse the direction of the line segment between the left elbow and hand, we would see some undesirable distortions in Fig. 4.10. We need to check where these 'ghost' pixels originally come from in the source image and usually we could solve this problem by adding or deleting a line segment pair. Additional example of line segment based shape interpolation can be found in Fig. 4.11.

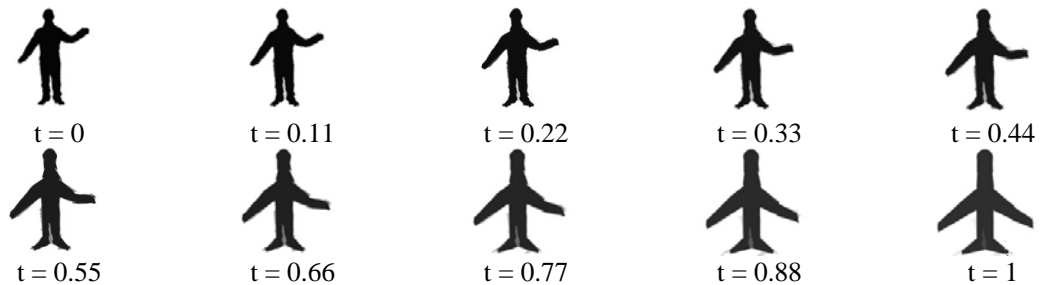


Figure 4.9: Blending of the human and airplane using line segment based interpolation.

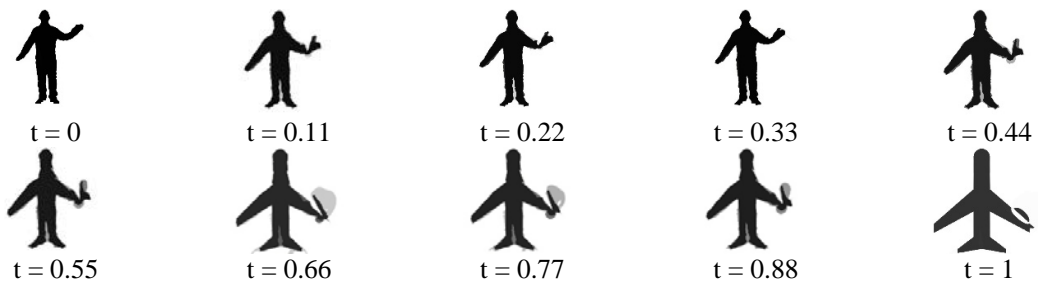


Figure 4.10: Generating undesired distortion by reversing the line segment direction between left elbow and hand in Fig.4.9.

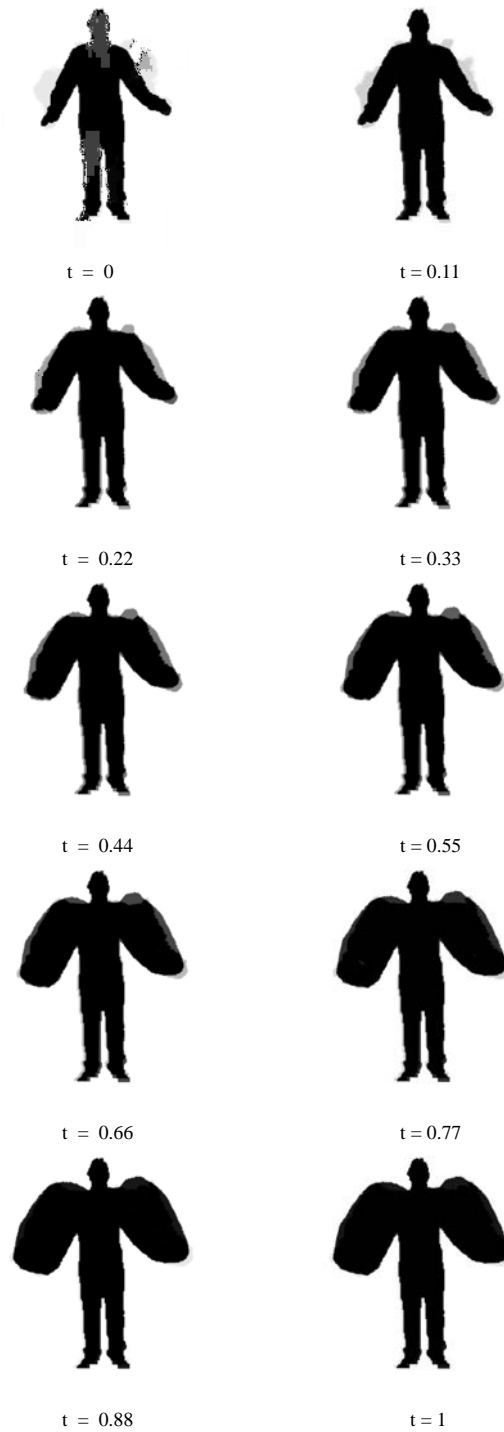


Figure 4.11: Additional example of shape interpolation using line segment based interpolation.

4.3.2 Shape Interpolation Using Moving Least Squares

Fig. 4.12 (a) and (b) shows two inputs, source and target image, we want to transform from Fig. 4.12 (a) to Fig. 4.12 (b) by some transformation function. Fig. 4.12 (c) shows the results of similar affine transformation, we can see that the upper arm is stretched due to some non-uniform scaling of M defined in Equation 4.5. Compared with MLS affine transformation, Fig. 4.12 (d) shows a more realistic result using similar affine transformation. Fig. 4.12 (e) shows the result using rigid affine transformation, the upper arm preserved better angle than the other transformations and it gives us an impression that we are operating the real object.

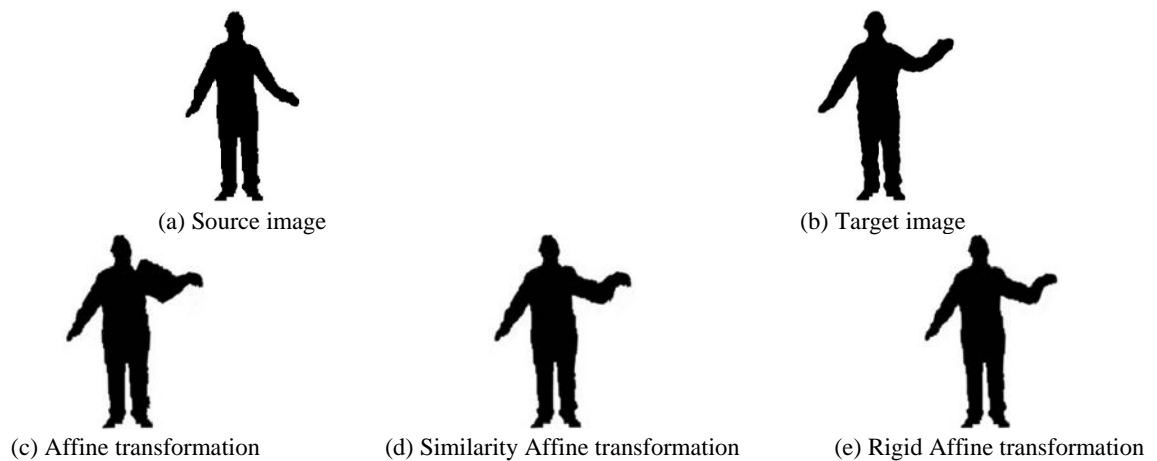


Figure 4.12: Shape deformation with sparse control points using MLS.

Fig. 4.13 shows that this method may generate some white space when the elbow approaches the waist, there are some blank between them. Fig. 4.14 blends the human and airplane with dense control points. Because this rigid MLS transformation only rotates and translates the shape, it is more suitable for blending two shapes of similar geometrical structure. We can see the jagged edge along the contour of the shape such as time slice $t = 0.66$ shown in Fig.4.14, which are caused by moving the dense control points that preserves local rigidity. Fig. 4.15 shows an additional example of transformation using MLS.

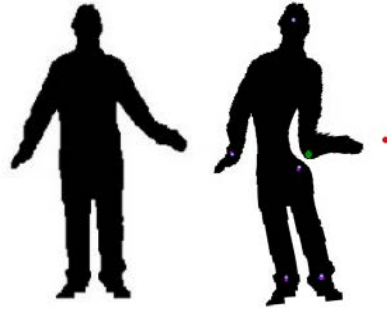


Figure 4.13: Fold back happens when two parts approach to each other.

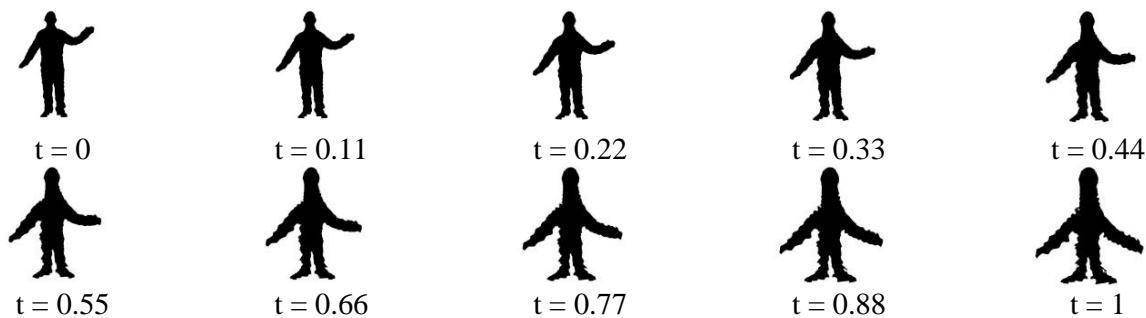


Figure 4.14: Image deformation with dense control points using MLS.

4.3.3 Rigid Shape Interpolation

The two-dimensional shapes for the rigid shape interpolation are generated by extracting a contour out of an image. For the correspondence of contours, we defined manually several vertex-to-vertex correspondences, while the remaining vertices were automatically aligned. A more intelligent way of computing the feature correspondence is using the geodesic distance to compute the similarity between two polygons.

As shown in Fig.4.16, we can see the human are smoothly transformed into an airplane. These local transformations are naturally interpolated over time and serve as the basis for composing a global coherent least-distorting transformation. Fig.4.17 shows one more example of rigid shape interpolation.

Compared with previous methods, i.e. line segment and control point based

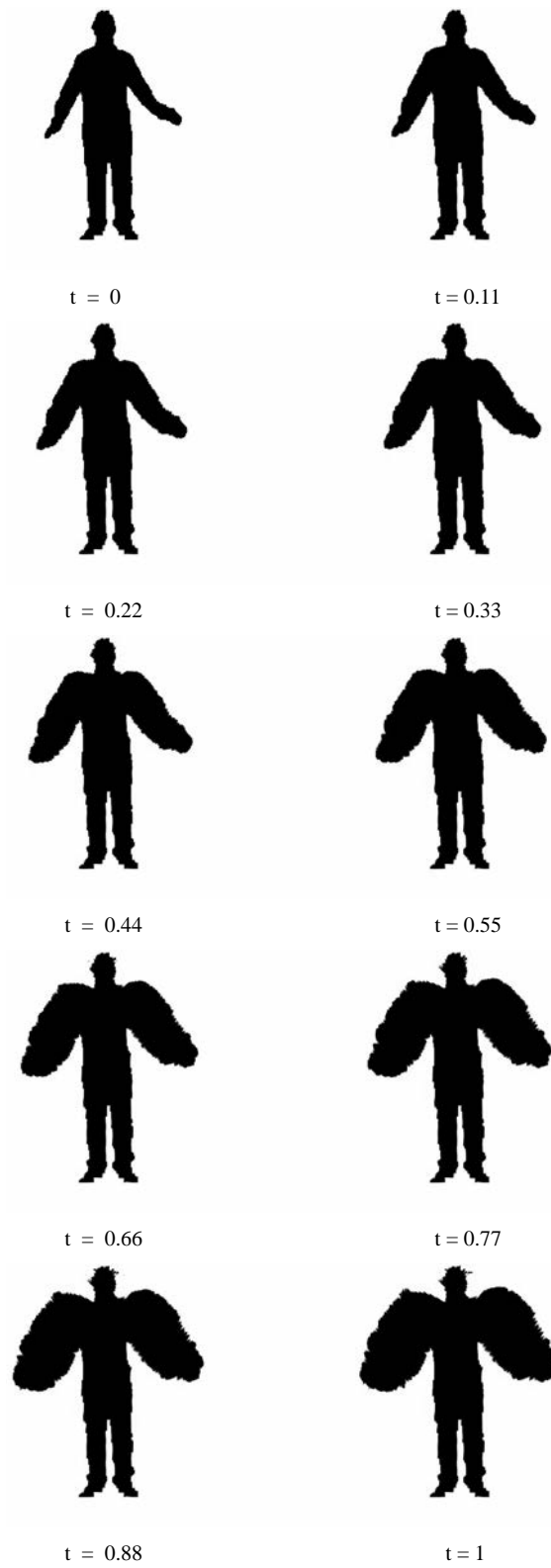


Figure 4.15: Additional example of shape interpolation using MLS.

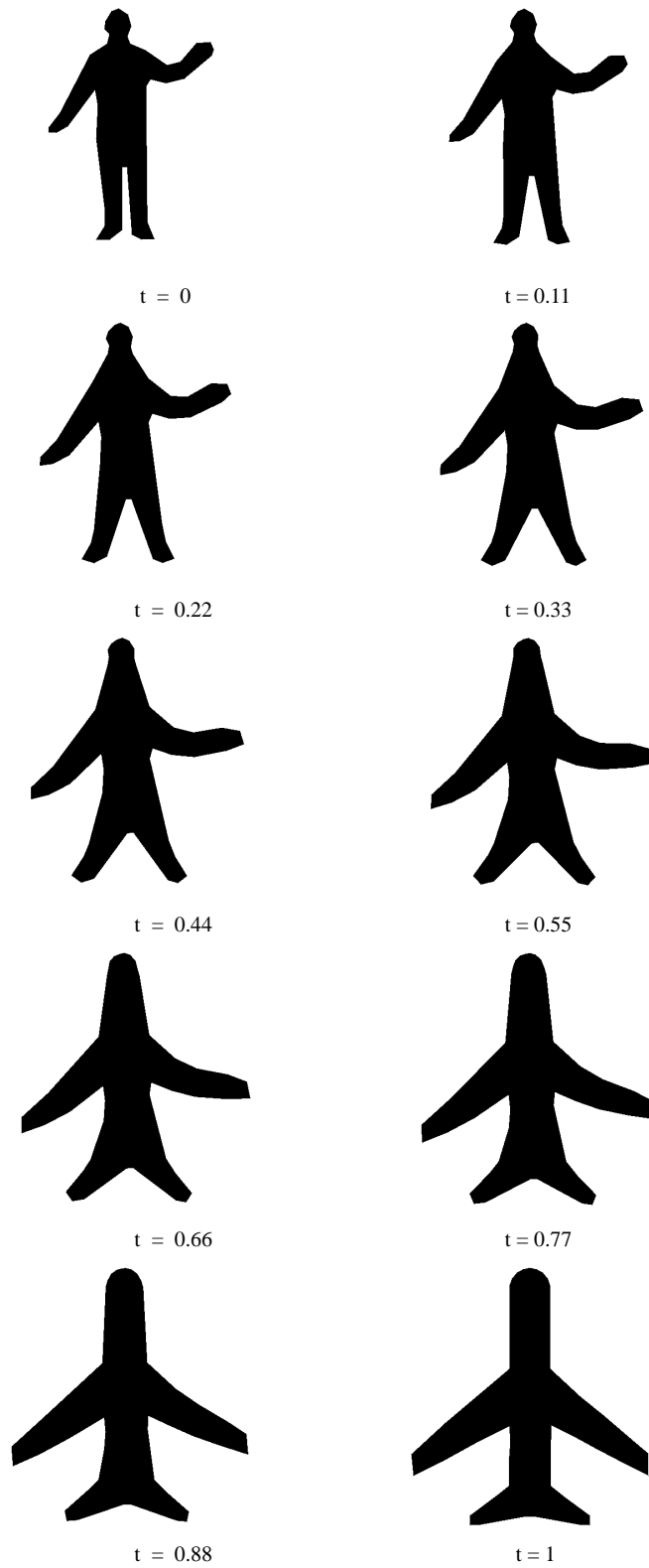


Figure 4.16: Rigid shape interpolation.

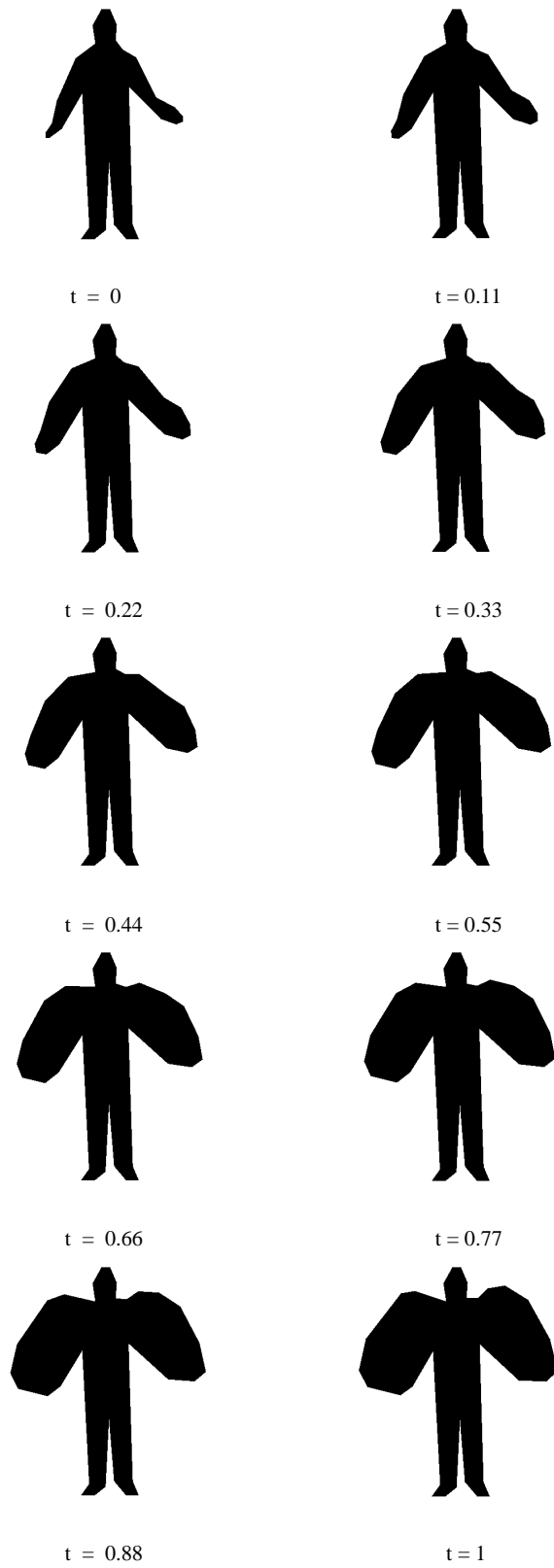


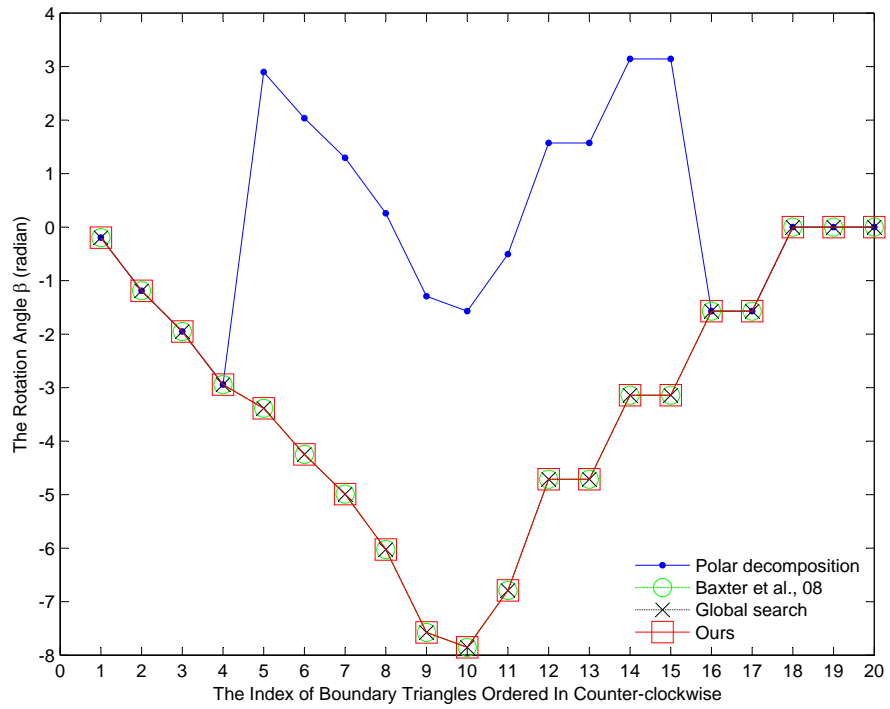
Figure 4.17: Additional example of blending shapes using rigid shape interpolation.

shape interpolation, rigid shape interpolation algorithm is able to blend shape of different geometrical structure and generate smooth transformations. In addition, rigid shape interpolation offers the possibility to add texture to the shapes such that shape blending becomes applicable to images.

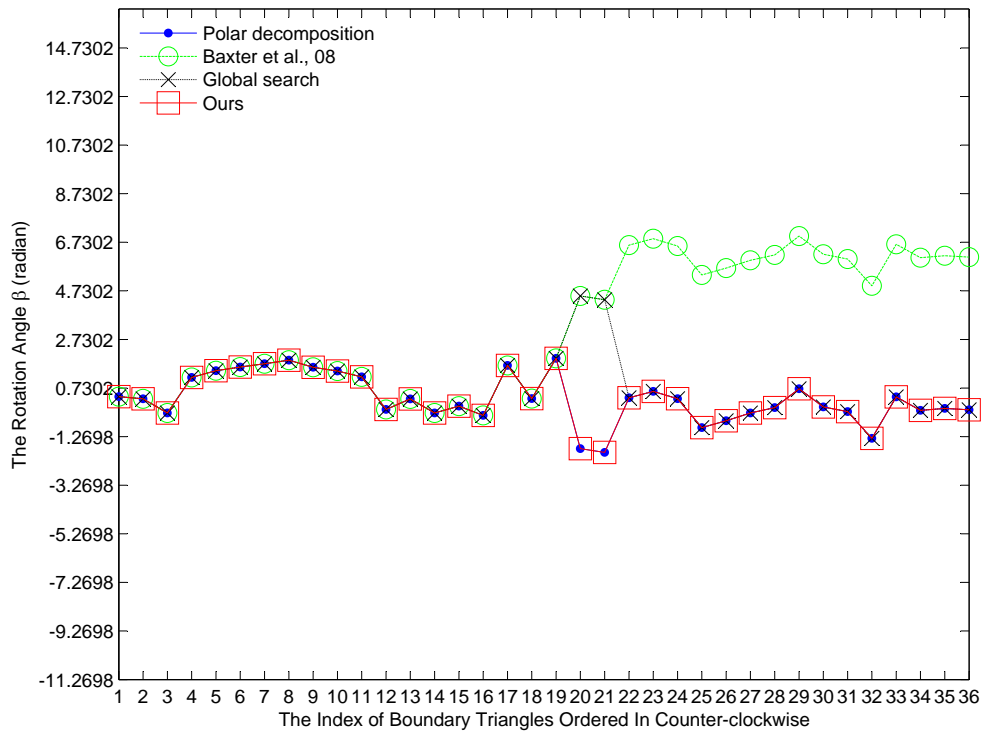
4.3.4 Regulating the Rotation for Rigid Shape Interpolation

As shown in Figure 4.18a, the rotations extracted from polar decompositions are the original rotations without any fix. More details about polar decomposition can be found in Section 4.1.3 and [2]. We can see that the rotations extracted from polar decompositions have big jump at the 5th and 15th boundary rotations in Figure 4.18a. [6] fixes this inconstant rotation problem through adding $2l\pi$, where l is an integer. The global search tries to find a global solution that each jump among adjacent rotation is within π . The results of our method are the same as [6] and the global search method.

Figures 4.18b shows another example that there is no consistent rotations, in which the compatible meshes are created by the method of [71]. There is a big jump at the 20th boundary triangle rotation. The problem stems from the rotation of the 19th boundary triangle, which is a long thin triangle. [6] starts from one boundary rotation and propagates the rotations inward either clockwise or counter-clockwise. In order to reduce the jump between adjacent rotation, their method tends to add additional rotations that lead to a big jump between the starting and ending rotation. In contrast, our method starts from one boundary rotation and searches both clockwise and counter-clockwise to fix inconsistent rotations, which can avoid the poor results by [6]. Our method *ignores* these long thin triangles and keeps these rotations unchanged, which minimizes the jump between all adjacent rotations.



(a)



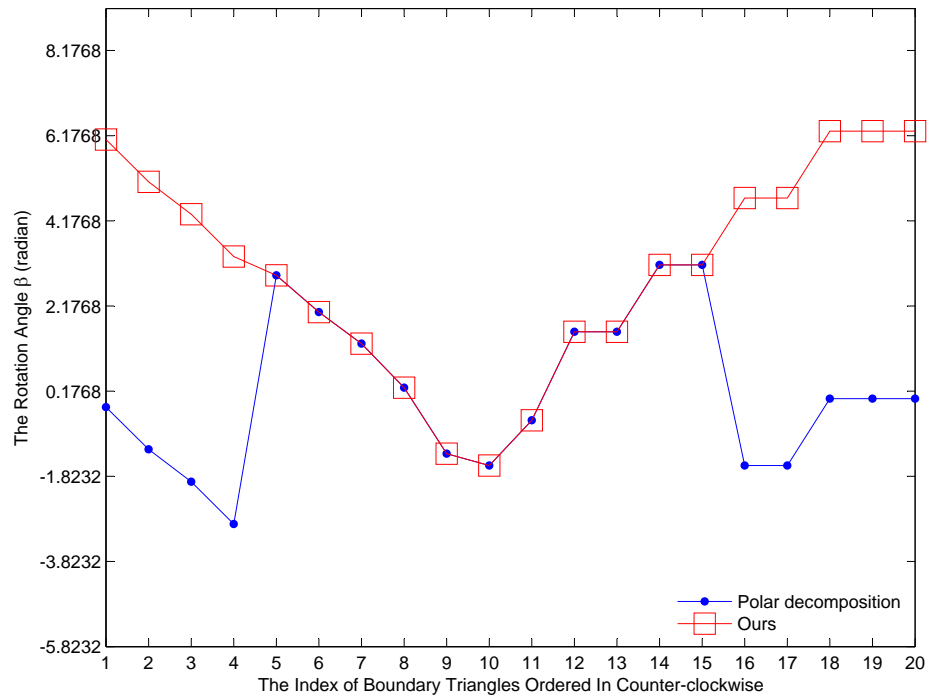
(b)

Figure 4.18: Computing consistent rotations by starting from the 1st triangle. (a) The boundary rotations of the swirl-like shape to stick, (b) human to butterfly.

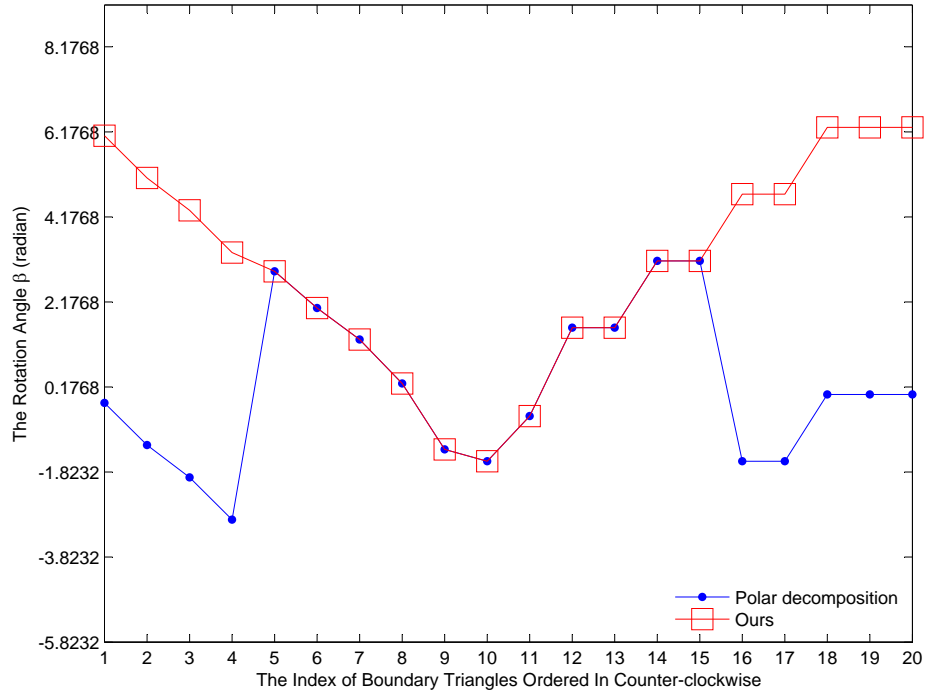
This is because (1) We observed that triangles lie on the interior of a source polygon usually need a small rotation, within $-\pi$ and π , to transform to the target one. (2) Long thin triangles are more likely to cause inconsistent rotations. Our results are similar to the global search method except for the 20th and 21th rotations. However, the total amount of jumps between our method and global search is very small for the example shown in Fig.4.18b.

Here, we discuss the influence of choosing the starting boundary vertex for solving inconsistent rotation problem. As shown in Fig.4.18a, with the polar decomposition, there are two jumps at the 5th and 15th vertex respectively. Now we test the starting vertex at the 5th and 15th vertex respectively. As illustrated in Fig.4.19a, while our method generates consistent rotation by starting from the 5th triangle, the total amount of rotation is larger than the solution of starting from the first triangle as shown in Fig.4.18a. Fig.4.19b shows the case of starting from the 15th vertex, which is the same as starting from the 5th triangle. The transformations with different starting vertices in Fig.4.19 can be found in Fig.4.20. We choose the rotation of the leftmost (1st) triangle of the stick-like shape as shown in Fig.4.5 because (1) The 1st triangle just needs a small rotation to be transformed into the target triangle and thus is more stable during shape morphing. (2) The total amount of rotations starting from the 1st triangle is smaller than the other solutions.

Compared with previous work such as [6], we offer an improvement over their method by taking additional steps to ensure that the rotation assignment chosen is unique and minimal. In addition, previous methods are slightly more succinct than ours, made so by the elimination of the special boundary handling step. Lastly, our method is able to deal with cases that consistent rotation assigning does not exist while the other methods [16, 22, 6] tend to generated excess rotations as illustrated



(a)



(b)

Figure 4.19: Fixing inconsistent rotations for swirl-like shape to stick-like shape by starting from the (a) 5th and (b) 15th triangles respectively.

in Fig.4.7(top) and Fig. 4.18b. More examples of assigning consistent rotation using our method can be found in Fig. 4.21.

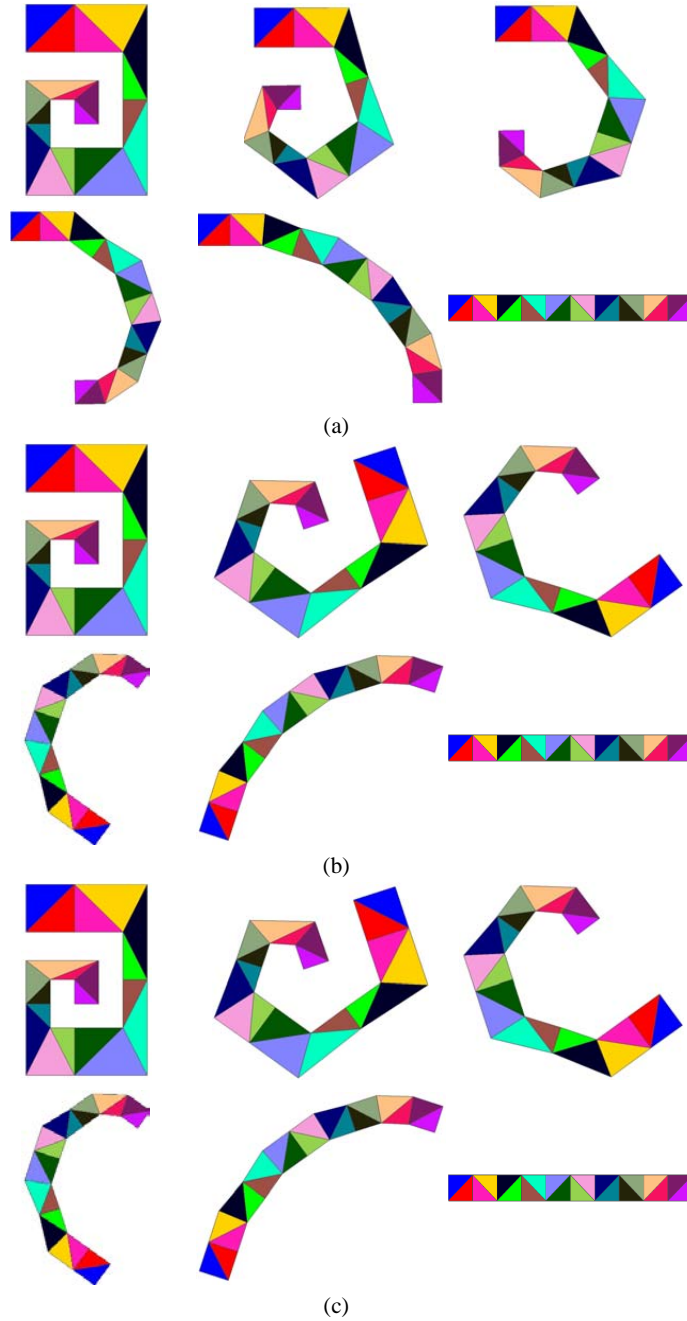


Figure 4.20: Shape transformations with rotation fix that starts from the vertices at the (a) 1st, (b) 5th, and (c) 15th triangles. The case of (b) and (c) need a larger amount of total rotation than (a).



Figure 4.21: Additional consistent rotation examples using our method.

4.4 Summary

In this chapter, we first discuss the image space and object space morphing methods. Specifically, for the image space morphing approaches, we apply the line segment based morphing and control point based shape deformation to blend an airplane and a human. The line segment based morphing can generate complicated shape morphing results, however, it may suffer from the undesired distortion due to the unforeseen combination of line segment pairs. While the control point based deformation creates smooth transformation between similar shapes, it is not suitable for blending two shapes of very different geometrical structure. For object space morphing, we apply the rigid shape interpolation method, which handles shape with different geometrical structure and keep the rigidity during morphing process. However, the rigid shape morphing algorithm may suffer from inconsistent rotation whenever the rotation is more than π . Finally we identify the failure mode related to large rotations that is easily triggered in practical use, and we present a solution for this as well. Experimental results show that our method well handles large rotation for rigid shape interpolation algorithm.

Chapter 5

Human Posture Reconstruction

Recent advances in depth camera based motion tracking devices such as the Microsoft Kinect has enabled efficient human-computer interaction using body movement, which enhances interactive systems such as motion-based gaming and sport training. In this thesis, we use RGBD camera (i.e. Kinect) as our purpose is for interactive applications and Kinect can estimate human poses in real time. While Kinect can be used to track the user and determine the user's 3D joint positions in a robust manner, the captured data suffer from poor precision due to self-occlusions. In this chapter, we propose a new real-time probabilistic framework to enhance the accuracy of live captured postures that belong to one of the action classes in the database. In Section 5.1, we illustrate the operation for data acquisition and preprocessing. In Section 5.2, we propose a new real-time probabilistic framework to enhance the accuracy of live captured postures. In Section 5.3, we show the posture reconstruction results using our method.

5.1 Data Acquisition and Preprocessing

For brevity, in this thesis we will use *MOCAP* to represent human motion data captured by an optical motion capture system. The postures obtained from Kinect are noisy and incomplete while *MOCAP* is accurate and stable. Hence, we can use *MOCAP* captured in an offline training stage to reconstruct postures captured by Kinect in real time.

5.1.1 Data Acquisition

We build a motion database captured from an optical motion capture system of Motion Analysis Corporation [47] with 7 cameras. Our database consists of different types of motions such as golf swinging and Tai Chi. The skeleton of the *MOCAP* system is a superset of that of the Kinect system, so we manually select 20 joints from the skeleton of the *MOCAP* system to match those of Kinect. Each posture in the database denotes a set of 3D positions of the body parts.

In this thesis, we model the relationship between Kinect data and *MOCAP* with Gaussian Process. Specifically, we capture motions with Kinect and optical motion capture system at the same time to identify their correspondence. The setup of this capturing procedure is shown in Fig. 5.1. The posture of Kinect at time t is denoted as $X_t = (x_t^1, x_t^2, \dots, x_t^n)$, $x_t^i \in R^3$, where x_t^i represents the 3D joint position of joint i over time t . There are 20 joints based on the skeleton definition of Kinect, i.e. $n = 20$. The corresponding *MOCAP* of X_t is denoted as $M_t = (m_t^1, m_t^2, \dots, m_t^n)$, $m_t^i \in R^3$.

To enhance the robustness of the spatial prediction model (Section 5.2.1) and to make the system invariant to different subjects, we follow [62] to conduct the normalization and retargeting processes, as they are simple yet effective. The posture normalization procedure is done by removing the rotation along the vertical axis and

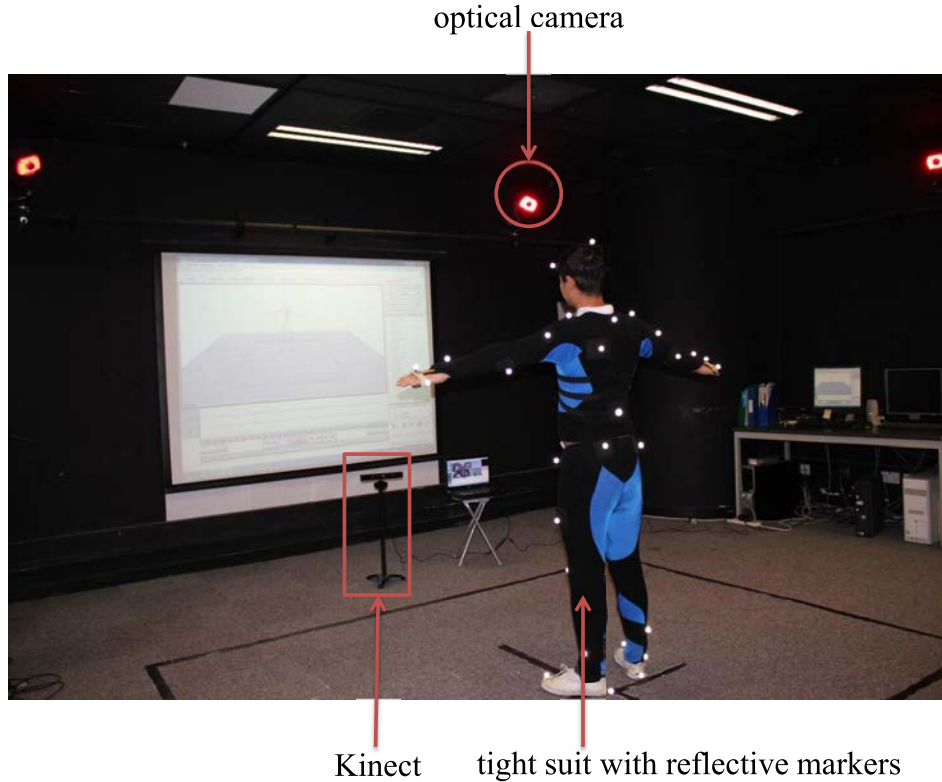


Figure 5.1: Human motion capture with Kinect and an optical motion capture system.

the global 3-D translation. The retargeting procedure ensures the system to be invariant to the skeleton size of the user.

5.1.2 Posture Budgeting

In this subsection, we introduce a data pruning scheme called posture budgeting to discard redundant samples.

We employ the probabilistic GPs to determine the samples that are informative to the model. We remove a specific training sample if such a sample can be precisely predicated by its neighbors in terms of the mean and variance. Specifically, we iteratively check each training sample (x_t^i, y_t^i) of joint i and determine its redundancy by calculating the relative entropy of the prediction of the training sample (x_t^i, y_t^i) with respect to the rest of the database. Following [77], we compute the Kullback-Leibler

(KL) divergence by:

$$D_{KL}(p(y_t^i|X^i, Y^i, x_t^i)||p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)) \quad (5.1)$$

where (X^i, Y^i) is a set of training samples, (x_t^i, y_t^i) is one of the samples in (X^i, Y^i) , and y_t^i is the difference between *MOCAP* data and Kinect data. Since both $p(y_t^i|X^i, Y^i, x_t^i)$ and $p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)$ are Gaussian Processes, we can solve the KL divergence in close form as:

$$\int p(y_t^i|X^i, Y^i, x_t^i) \log \frac{p(y_t^i|X^i, Y^i, x_t^i)}{p(y_t^i|X^i - x_t^i, Y^i - y_t^i, x_t^i)} dx \quad (5.2)$$

Further details of solving $p(y_t^i|X^i, Y^i, x_t^i)$ can be found in section 5.2.1 and 5.2.2.

Fig. 5.2 shows the impact of posture budgeting. The unfiltered Walking motion database includes 1120 training samples, which is reduced to 681. Nearly 40% of the training data can be pruned while maintaining a similar error level. For the details of the reconstruction error definition, please refer to Section 5.3.3.

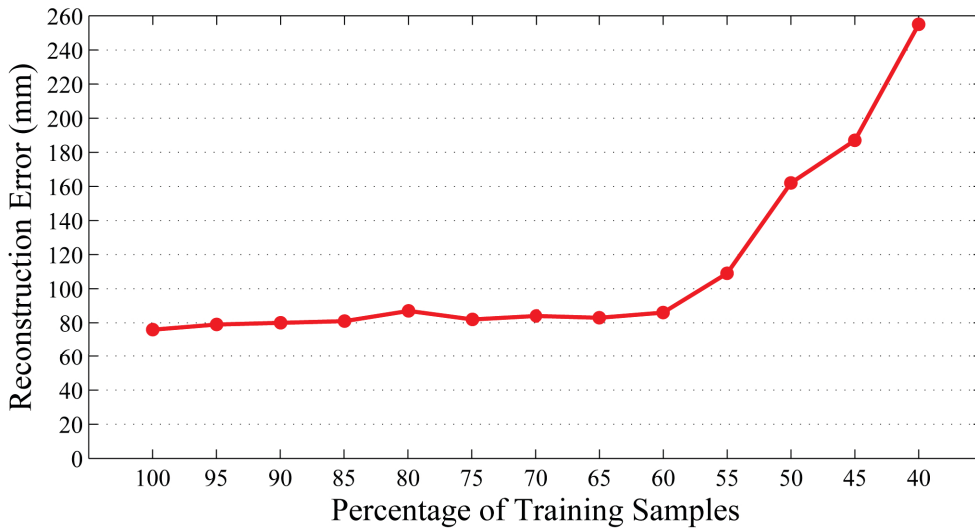


Figure 5.2: Posture budgeting: We can shrink up to 40% of the training data while the mean error almost remains constant.

5.2 Posture Reconstruction

To ensure that the reconstructed posture is accurate and resembles the input data from Kinect, we formulate the posture reconstruction as an optimization problem by minimizing an energy function. Such an energy function consists of three energy terms to constrain the solution space, which are the spatial prediction term, the temporal prediction term, and the reliability term. In the following, we will elaborate the definition and purpose of each term.

5.2.1 Spatial Prediction

Assuming that the *MOCAP* posture M_t is the corrected posture of the Kinect posture X_t , we design a spatial prediction term to evaluate how well the reconstructed posture fits with the *MOCAP* data, which implicitly favors solutions that are more similar to the correct posture.

Due to self-occlusions and sensor error, there exists a residual offset between X_t and M_t , which is calculated by $Y_t = M_t - X_t$, where $Y_t = (y_t^1, y_t^2, \dots, y_t^i), y_t^i \in R^3$. During run time, the objective is to predict the residual offset Y_t so that we can obtain the reconstructed posture M_t by appending Y_t to X_t .

In this thesis, we adopt the non-parametric GP as the predictor. More formally, let $X^i = [x_1^i, \dots, x_T^i]^T$ be the input data of an arbitrary joint i , where T is the total number of frames. Let $Y^i = [y_1^i, \dots, y_T^i]^T$ denote the output values such that y_t^i is the corresponding output of the input x_t^i . Here, we model the sensor error as the difference between the *Mocap* data and the Kinect data using Gaussian process, which transforms the input X^i of the i^{th} joint into the output Y^i by:

$$y_t^i = f(x_t^i) + \epsilon \quad (5.3)$$

where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$ is a noise variable, which is independent for each data point.

The joint distribution of the output Y^i conditioned on input X^i is given by:

$$p(Y^i|X^i) = \int p(Y^i|f^i, X^i)p(f^i|X^i)df = \mathcal{N}(Y^i|0, K) \quad (5.4)$$

where K is the covariance matrix, in which the element $k(x_a^i, x_b^i)$ is defined as:

$$\begin{aligned} k(x_a^i, x_b^i) &= \theta_0 \exp\left(-\frac{1}{2}(x_a^i - x_b^i)^T W (x_a^i - x_b^i)\right) \\ &+ \theta_1 + \beta^{-1}\delta_{ab} \end{aligned} \quad (5.5)$$

where a and b are indices of training samples of joint i , δ_{ab} is Kronecker's delta function, W is kernel width, θ_0 is signal noise, θ_1 is a constant bias. At the training stage, with the obtained training data from Kinect and *MOCAP*, we can learn the hyper-parameters of $\Phi = \{\theta_0, \theta_1, W, \beta\}$ by maximizing the log marginal likelihood:

$$\log p(Y^i|X^i, \Phi) = -\frac{1}{2}Y^{iT}K^{-1}Y^i - \frac{1}{2}\log|K| + \mathcal{C} \quad (5.6)$$

where K is the covariance matrix defined in (5.5) and \mathcal{C} is a constant. Obviously, the computational cost of learning GP is dominated by the cubic complexity of computing the inverse of covariance matrix K^{-1} .

Human body joints are highly coordinated and it is important to take into account the relationship between them. Here, given an arbitrary joint, we use its neighboring joints for prediction. Specifically, given a joint i at time t , x_t^i , its neighboring joints $N(x_t^i)$ are defined as the set of joints that are directly connected with the same bone segment as joint i . Therefore, the input feature x_t^i for obtaining y_t^i of joint i is the union set of \tilde{x}_t^i and $N(\tilde{x}_t^i)$, where $N(\tilde{x}_t^i)$ is the normalized position of the

neighboring joints for joint i . The input data of joint i is thus defined as $X^i = [(\tilde{x}_{t_1}^i, N(\tilde{x}_{t_1}^i)), \dots, (\tilde{x}_{t_n}^i, N(\tilde{x}_{t_n}^i))]^T$, where t_1, \dots, t_n are time slices. The output data Y^i correspond to X^i of the prediction model. To simplify notation, we use $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$ to denote new input of the i^{th} joint at time t_* and use y_*^i to represent the corresponding output of x_*^i in the remaining parts of the thesis.

With the learned model, we formulate the above prediction for y_*^i as a conditional probability distribution, yielding the spatial prediction energy term of the i^{th} joint as defined below:

$$E_S^i = \ln p(y_*^i | X^i, Y^i, x_*^i) \sim \mathcal{N}(\mu(x_*^i), \sigma(x_*^i)) \quad (5.7)$$

where

$$\mu(x_*^i) = k(x_*^i, X^i)K^{-1}Y^i = k(x_*^i, X^i)\alpha \quad (5.8)$$

$$\sigma(x_*^i) = k(x_*^i, x_*^i) - k(x_*^i, X^i)K^{-1}k(x_*^i, X^i)^T \quad (5.9)$$

are the predicted mean and variance respectively, K is the covariance matrix defined in (5.5), $\alpha = K^{-1}Y^i$ is the so-called prediction vector that can be pre-calculated from training samples, and the predicted mean is determined by the vector $k(x_*^i, X^i)$.

The term E_S ensures that the reconstructed postures are similar to the correct postures as much as possible. Predicting the offset of each joint reduces the searching space compared with inferring individual joints directly. The use of the weighted local GP models allows synthesizing variations in postures based on the motion database. There are several publicly available implementations of Gaussian Process. In this thesis, we used the library developed by Lawrence [36].

5.2.2 Incremental Learning of Local Gaussian Processes

The major problem of using full GP is its cubic learning complexity of the inverse covariance matrix K^{-1} in (5.6). Here, we propose a new method based on a local mixture of Gaussian Processes that has the following advantages: 1) The local GP models are created by partitioning the posture space into Q local regions using clustering algorithm, and training Q local GP models independently. This relieves the cubic computational cost for learning the full GP model. 2) Since we use the weighted average prediction of nearby local models in which only a small number of training samples are involved, the prediction process is fast. 3) With the use of local models, it becomes possible to incrementally update a specific local GP with the complexity of $O(\mathcal{S}^2)$, where \mathcal{S} is the size of local GP. With the newly added predicted samples, the system accuracy can be enhanced for run-time postures that are different from those in the database.

Our algorithm consists of three major parts: 1) learning the hyper-parameters of local GP models; 2) performing the weighted prediction of local GP models; 3) incremental updating of corresponding local GP models, that is, adding a new sample into the closest local model and updating the inverse covariance matrix K^{-1} in (5.8). Fig. 5.3 shows an example of applying the local mixture of GP models. To simplify illustration, we project the 3D joint positions onto the XZ plane.

Learning of local Gaussian Processes

We cluster the training samples into Q regions and learn the hyper-parameters at each local region. In our system, Q is empirically set as 30. Similar to Section 5.2.1, the hyper-parameters of Φ for each local model can be estimated by maximizing the log marginal likelihood in each local region as defined in (5.6).

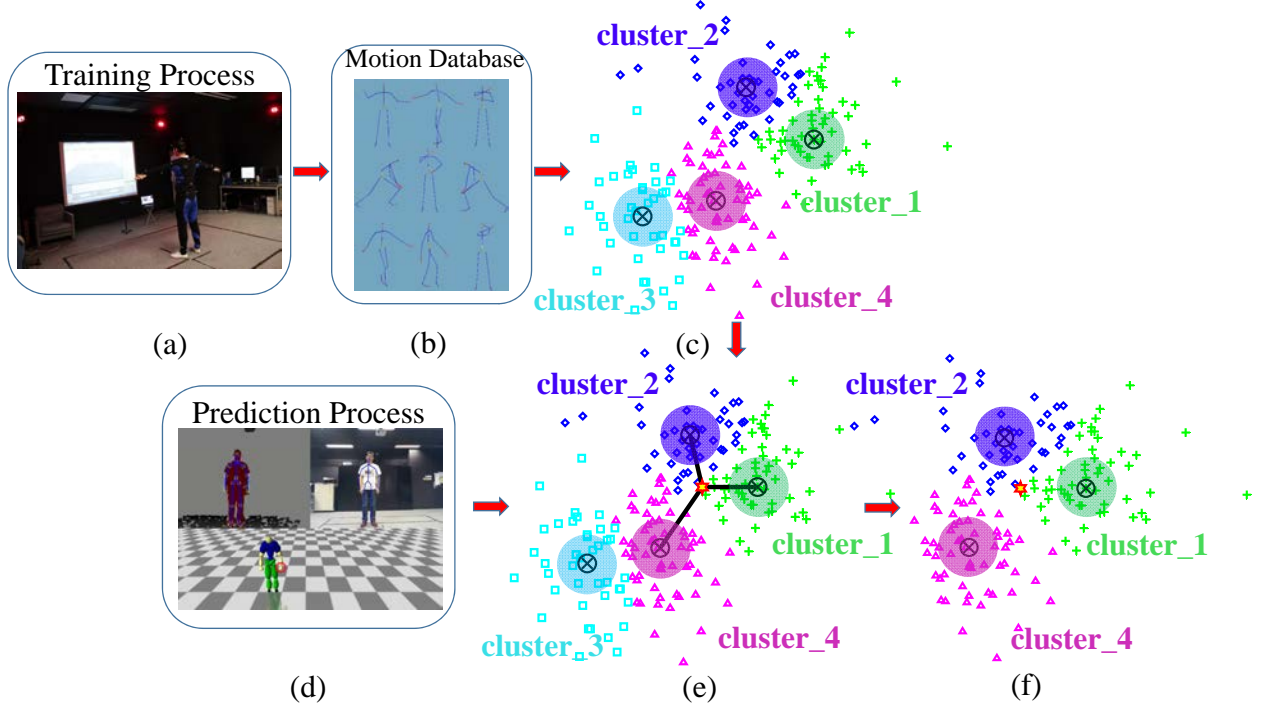


Figure 5.3: Overview of the local mixture of GP models. (a-b) We capture postures by the *MOCAP* system and Kinect at the same time to generate the training samples. (c) At the training stage, we partition the samples into $Q = 4$ local regions by K-means and learn Q local GPs with $\mathcal{S} = 10$ training samples independently. (d) During prediction, we extract feature of the $i^{\text{th}} = 7$ (left hand) joint, $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$. (e) For a given test sample, x_*^i , shown in red star, we find the nearby $L = 3$ local models by similarity measurement defined in (5.10). (f) We compute the local predicted mean by the l^{th} local GP model and then generate the weighted mean prediction $\mu(x_*^i)$ using L nearby local models given by (5.13).

Here, we explain how to partition the samples into different local regions. Given a new input x_*^i of joint i , assigning x_*^i to the q^{th} local region is straightforward by measuring the similarity between x_*^i and the center of cluster C_q . Here, we use the Gaussian kernel to measure the similarity, which is in the same form of (5.5):

$$\text{similarity}(q) = \exp\left(-\frac{1}{2}(x_*^i - C_q)^T W (x_*^i - C_q)\right) \quad (5.10)$$

where C_q is the center of the q^{th} cluster ($q \in Q$) and W is the kernel width.

To speed up the run-time computation, we learn these hyper-parameters as an

offline process named GP-offline. Its computational complexity is the summation of the complexity of clustering, $O(QdN)$, and the complexity of learning Q local GP models, $O(Q\mathcal{S}^3)$, where Q is the number of local models, d is the dimension of input, N is the total training data, and \mathcal{S} is the size of each local model. The first part of Algorithm 5.1 summarizes the training of local GP models, where the $kmeans(X^i, Q)$ function partitions the X^i into Q clusters, and returns an index set ς_q of samples for the q^{th} local model.

Prediction of local Gaussian Processes

Note that the mean of the prediction in (5.8) can be written as a function of Y^i :

$$\mu(x_*^i) = \sum_{t=1}^T w_t^i y_t^i \quad (5.11)$$

where w_t^i is the t^{th} element of $k(x_*^i, X^i)K^{-1}$, T is the total number of frames. In this view, the mean of the prediction distribution is determined by the weighted combination of the N training outputs. We therefore propose to interpret the full GP as a global voting process, in which all weighted training outputs contribute to the decision of the test sample x_*^i of joint i . We observed that local neighborhoods behave similarly such that nearby samples are likely to have similar output. With this insight, the full GP could be locally approximated by a small number of GPs near a given feature x_*^i of joint i , which could significantly reduce the computational cost and speed up the prediction.

Here, we explain how to determine the vote of local models. Similar to the model in [84], the contribution of each local model is determined by the distance to each local model. Given a new input $x_*^i = (\tilde{x}_*^i, N(\tilde{x}_*^i))$ of joint i , the weight of each local GP model can be determined by the normalized distance to l^{th} local

model. Specifically, we compute the averaged prediction of L nearby local models by $\mu(x_*^i) = \mathbb{E}\{\mu_l|x_*^i\} = \sum_{l \in L} \mu_l(x_*^i)p(l|x_*^i)$ where $\mu_l(x_*^i)$ is the predicted mean using the l^{th} local model given by (5.11) and $p(l|x_*^i)$ is the weight of each local GP, which is given by:

$$p(l|x_*^i) = \text{similarity}(l) / \sum_{\eta \in L} \text{similarity}(\eta) \quad (5.12)$$

where $\text{similarity}(l)$ measures the similarity between x_*^i and the center of the l^{th} local model given by (5.10). Hence, we calculate the weighted prediction by:

$$\begin{aligned} y_*^i &= \sum_{l \in L} \left(\frac{\text{similarity}(l)}{\sum_{\eta \in L} \text{similarity}(\eta)} \right) \mathcal{N}(\mu_l(x_*^i), \sigma_l(x_*^i)) \\ &= \sum_{l \in L} \sum_{\varsigma \in \mathcal{S}} \left(\frac{\text{similarity}(l)}{\sum_{\eta \in L} \text{similarity}(\eta)} \right) w_{l\varsigma}^i y_{l\varsigma}^i \end{aligned} \quad (5.13)$$

where L is total number of the nearby local models, \mathcal{S} is the size of training samples in each local model, ς is the index of local training samples in \mathcal{S} , $w_{l\varsigma}^i$ is the element of $k(x_*^i, X_\varsigma^i)K_{\varsigma, \varsigma}^{-1}$, and $y_{l\varsigma}^i$ is one of the offsets in \mathcal{S} that belongs to l^{th} local model. The prediction process is summarized in the second part of Algorithm 5.1.

The prediction of our model is computationally inexpensive as those local models are learnt from a very small set of neighborhoods. L and \mathcal{S} are parameters of our model, and typical small values are sufficient to generate satisfactory results. In our implementation, the size \mathcal{S} of each nearby local model is 50 and the number of nearby local models L is 9. The influence of L and \mathcal{S} are discussed in Section 5.3.3.

Incremental updating of local Gaussian Processes

One limitation of the data-driven method for posture reconstruction is that the reconstruction quality might drop significantly if we cannot find similar postures

Algorithm 5.1 Local mixture of GP models and prediction

Offline: Learning of hyper-parameters
 Q : total number of local GP models

 C_Q : the center of each local GP model

 ς_q : the index set of samples for the q^{th} local model

 $(C_Q, \varsigma_Q) = kmeans(X^i, Q)$
for $q = 1$ **to** Q **do**
 $\{\bar{\phi}^q\} \leftarrow \max(\ln p(Y_{\varsigma_q}^i | X_{\varsigma_q}^i, \bar{\phi}^q))$
end
Online: Prediction of a new input of joint i
Input: new input, x_*^i , of joint i
 L : the number of nearby local models

 S : the size of training samples for each local GP model

for $l = 1$ **to** L **do**

 Compute the similarity to the center of l^{th} cluster:

 $similarity(l) = \exp(-\frac{1}{2}(x_*^i - C_l)^T W (x_*^i - C_l))$

 Compute the local predicted mean by the l^{th} local model and ς_l is the index set of l^{th} local training samples:

 $\mu_l(x_*^i) = k(x_*^i, X_{\varsigma_l}^i) K_{\varsigma_l, \varsigma_l}^{-1} Y_{\varsigma_l}^i$
 $\sigma_l(x_*^i) = k(x_*^i, x_*^i) - k(x_*^i, X_{\varsigma_l}^i)^T K_{\varsigma_l, \varsigma_l}^{-1} k(x_*^i, X_{\varsigma_l}^i)$
end

 Compute the weighted prediction of new input x_*^i of joint i by the L local models:

$$y_*^i = \sum_{l \in L} (similarity(l) / \sum_{\eta \in L} similarity(\eta)) \mathcal{N}(\mu_l(x_*^i), \sigma_l(x_*^i))$$

in the database. To relieve this problem, our model should be able to learn from the newly estimated samples such that we are more likely to adapt to unknown postures that are different from those in the database.

Here, we explain the major process of incremental updating of local GP models. During the local GPs learning process, we learn the hyper-parameters of Φ and factorize the covariance matrix K by (5.14). At the prediction stage, the local models would predict the offset y_*^i given a new input x_*^i . During the incremental updating process, we preserve the samples (x_*^i, y_*^i) with high reliability and low predictive variance and append it into the nearest local model using the similarity measurement given by (5.10).

We calculate the similarity between x_*^i and the mean of each local Gaussian Process. If the similarity values with all local GPs are smaller than a predefined threshold $w_{similar}$, we create a new local model centers at x_*^i . Otherwise, we update the local GP with the highest similarity value. Notice that during the incremental learning process, the number of newly added samples can be further reduced by posture budgeting introduced in section 5.1.2.

To update a local GP, we need to first update both the prediction vector and the mean of the local model. To update the prediction vector $\alpha = K^{-1}Y^i$, we adapt [57] in which the K^{-1} is updated by adjusting Cholesky factorization. As K is a symmetric, positive-definite matrix, we can uniquely factorize K as:

$$K = U^T U \quad (5.14)$$

where U is a upper triangular matrix with positive diagonal elements. We then update the mean of the corresponding local model.

Given a new input x_*^i of joint i , we need to add additional rows and columns to K and U as follows:

$$K_{new} = \begin{bmatrix} K & \mathbf{k}_{new} \\ \mathbf{k}_{new}^T & k_{new} \end{bmatrix} \quad (5.15)$$

$$U_{new}^T = \begin{bmatrix} U^T & 0 \\ u^T & u_* \end{bmatrix} \quad (5.16)$$

where $\mathbf{k}_{new} = k(X^i, x_*^i)$, $k_{new} = k(x_*^i, x_*^i)$. Then, we can solve u and u_* by completing $K_{new} = U_{new}^T U_{new}$ as:

$$Uu = \mathbf{k}_{new}, u_* = \sqrt{k_{new} - u^T u} \quad (5.17)$$

Once we have solved U_{new} , we can update the prediction vector α in

$U_{new}^T U_{new} \alpha_{new} = Y^{i_{new}}$ through back-substitution. The cost of back-substitution for a local model is $O(\mathcal{S}^2)$, where \mathcal{S} is the number of training samples in a local model. Finally, we recalculate the corresponding local model using (5.8).

Table 5.1 compares the complexity of the full GP and our method. The computation of the Cholesky factorization is $O(Q\mathcal{S}^3)$, where Q is the number of local GP models and \mathcal{S} is the number of training samples in a local model. The prediction cost is $O(L\mathcal{S}^2)$, where L is the number of nearby local GPs given an input. Thus, the offline learning complexity, $O(2Q\mathcal{S}^3 + QdN)$, dominates the main computational complexity of our method. The cost of incremental updating is $O(\mathcal{S}^2)$ due to the update of Cholesky factorization, which enables our system to incrementally update a specific local Gaussian Process in real time. Algorithm 5.2 summarizes the incremental learning of local models.

Table 5.1: Computational complexity: The main computational cost of our method is the offline learning while the incremental updating is fast.

	Proposed Method	Full GP
Learning	$O(2Q\mathcal{S}^3 + QdN)$	$O(N^3)$
Prediction	$O(L\mathcal{S}^2)$	$O(N^2)$
Incremental Updating	$O(\mathcal{S}^2)$	N/A

5.2.3 Temporal Prediction

The above spatial prediction considers each posture independently. To ensure the temporal smoothness between consecutive frames, the relationship between frames is modeled as a second order temporal model, which has been verified to be effective in preserving temporal smoothness [63]. Specifically, we adopt a constant velocity variation to smooth velocity, which is formulated as below:

$$E_T = \ln p(M_t | M_{t-1}, M_{t-2}) \quad (5.18)$$

Algorithm 5.2 Incremental learning of local models

Input: new input, x_*^i , of joint i L : the number of nearby local models of x_*^i C_l : the center of the l^{th} local model, where $l \in L$ Q : total number of local GP models \mathcal{B} : the training samples $\mathcal{B} = (X^i, Y^i)$ and \mathcal{B}_q represents the samples of the q^{th} local modelPredict the offset, y_*^i , by L nearby local models (see Algorithm 5.1)**for** $l = 1$ to L **do**

$$similarity(l) = \exp(-\frac{1}{2}(x_*^i - C_l)^T W (x_*^i - C_l))$$

endFind the most similar j^{th} local model

$$max_{similar} = \max(similarity)$$

if $max_{similar} \leq w_{similar}$ **then**

Create a new local model:

$$C_{Q+1} = \{x_*^i\}$$

$$\mathcal{B}_{Q+1} = \{(x_*^i, y_*^i)\}$$

elseAppend $\{x_*^i, y_*^i\}$ to the nearest local model j

$$\mathcal{B}_{j_{new}} = \{\mathcal{B}_j; (x_*^i, y_*^i)\}$$

Update the mean of j^{th} model

$$C_{j_{new}} = mean(X_{j_{new}}^i)$$

Update $\alpha = K^{-1}Y^i$ of the j^{th} local model:Compute u , u_* , and U_{new} Compute α_{new}^i by back-substitution**end**

M_t , M_{t-1} , and M_{t-2} are the reconstructed postures at time slices t , $t-1$, and $t-2$. We have the following relationship between the reconstructed posture, input posture and the residual offset:

$$M_t = Y_t + X_t \tag{5.19}$$

Therefore, we can rewrite (5.18) as:

$$\begin{aligned}
E_T &= \ln p(Y_t + X_t | M_{t-1}, M_{t-2}) \\
&= \|(M_t - M_{t-1}) - (M_{t-1} - M_{t-2})\|^2 \\
&= \|M_t - 2M_{t-1} + M_{t-2}\|^2 \\
&= \|Y_t - (-X_t + 2M_{t-1} - M_{t-2})\|^2
\end{aligned} \tag{5.20}$$

which facilitates the continuity in the reconstructed motions.

5.2.4 Reliability Embedding

The accuracy of each tracked joint is different depending on the degree of occlusion. The incorrectly tracked joints from Kinect will incorrectly guide the system to infer the joint positions. The residual offset, $Y_t = M_t - X_t$, of the correctly tracked joints should be smaller as they are closer to the corrected posture, namely M_t . Thus, it is essential to consider the reliability of each joint to constrain the residual offsets of these joints with higher confidence during the prediction of Y_t . We use a reliability term E_R to penalize the residual offset of each joint based on its reliability, which implicitly ensures that the reconstructed posture resembles the input posture from Kinect as much as possible. More specifically, the residual offset value y_t^i of joint i should be smaller if the corresponding joint is with higher reliability.

We adopt the strategy proposed by [62] to evaluate the reliability of the tracked joints from Kinect. They evaluate the reliability in three aspects: behavior reliability, kinematics reliability, and tracking state reliability. The behavior reliability refers to abnormal behavior of a tracked joint, which is calculated by the cosine similarity between two consecutive displacement vectors of one joint. The kinematics reliability represents the kinematic correctness of the tracked joints, which measures the change

of bone length for bones connecting with the joint. The tracking state reliability tells if a joint is tracked, inferred or not tracked when it is completely occluded. More details about the calculation of the reliability of each joint can be found in [62]. As a result, the reliability rate of each joint is a value between 0.0 and 1.0 (inclusive). We embed the reliability of each joint into the optimization framework and formulate the following reliability term:

$$E_R = \|RY_t\|_F^2 \quad (5.21)$$

$\|\cdot\|_F$ is the Frobenius norm. The entry of R is the reliability of each joint, which ensures the reconstructed posture does not deviate from the input posture from Kinect. Intuitively, while minimizing the objective function, the value of y_t^i tends to be small when its reliability value is large.

5.2.5 Energy Minimization Function

With the terms defined in the above sections, the posture reconstruction problem is formulated as the following optimization function:

$$E = \arg \min_{Y_t} \{w_S E_S + w_T E_T + w_R E_R\} \quad (5.22)$$

where w_S , w_T , and w_R are the weights of the energy terms. In our implementation, they are empirically set to be 0.6, 0.2, and 0.2, respectively. We optimize (5.22) by using the gradient descent method. We sample a number of potential postures in the solution space in each iteration. The posture that minimizes the cost function will be considered as the initial posture sample in the next iteration. Our posture reconstruction system is frame-based. The initial posture for optimization at each frame is defined as the previous reconstructed posture, which allows the system to

have higher chance to find the optimized posture. The optimization procedure stops when an optimal solution is found or the number of iterations reaches a predefined threshold.

There are some principles to tune the values of the weights. The weight of the spatial prediction term should be set the largest, since this term drags the reconstructed posture to the corrected posture as closely as possible. Second, the temporal prediction term ensures the temporal stability of the posture sequences. The reliability term makes sure the reconstructed posture is as similar as the Kinect posture, since the primary purpose of the system is to reconstruct Kinect postures. We will evaluate how these terms affect the accuracy of the system in Section 5.3.5.

The proposed framework for posture reconstruction is summarized here. At the offline stage, we learn a spatial prediction model using Gaussian Process with pairwise Kinect data and marker-based motion capture data. It ensures that the reconstructed posture is as accurate as the *MOCAP* data. We also embed the temporal and reliability terms in offline process so as to generate temporal smoothness and reliable postures. At the online stage, the system obtains an optimized posture with live captured data from Kinect, which ensures the reconstructed posture resembles the input posture from Kinect while maintaining the temporal smoothness between previous frames.

5.3 Experimental Results

In this section, we will show the experimental results and present the comparisons with alternative approaches including Kinect SDK [45], as well as the algorithms proposed by [59] and [85]. We first show postures with severe self-occlusions

reconstructed by our approach. Qualitative and quantitative analysis were conducted to evaluate the accuracy.

The experimental results were conducted on a desktop computer with Intel Core 2 Duo 3.17 GHZ processor. If not otherwise mentioned, we use Kinect official SDK [45] to obtain posture data. Here, we consider the Kinect device as one additional reflective marker of the optical motion capture system to eliminate the interference between Kinect and the optical motion capture system. The setup environment of Kinect and optical motion capture system is shown in Fig. 5.1.

5.3.1 Posture Reconstruction

The proposed approach works for users with different body sizes and proportions, because we normalize and retarget the Kinect input posture as explained in Section 5.1.1. We evaluate our system on a wide range of human motions, including sports activity such as Tai Chi, bending, golf swinging, and daily actions such as crossing arms, waving right hand, clapping hands, rolling hands up and down, rolling hands forward and backward.

We choose these motions because all these motions contain severe self-occlusions, which are not well tracked by the Kinect system. However, the proposed method can well reconstruct these inaccurate postures even if a number of joints cannot be tracked by the Kinect sensor. Fig. 5.4 and 5.5 showed several frames of our results, the lower right avatar represents the postures reconstructed by our method and the lower left avatar corresponds to the estimated posture by Kinect SDK [45]. The upper half shows the RGB and depth images respectively. We can observe that certain parts of the postures from Kinect SDK are twisted when there exist occlusions while our method can reconstruct the postures very well.

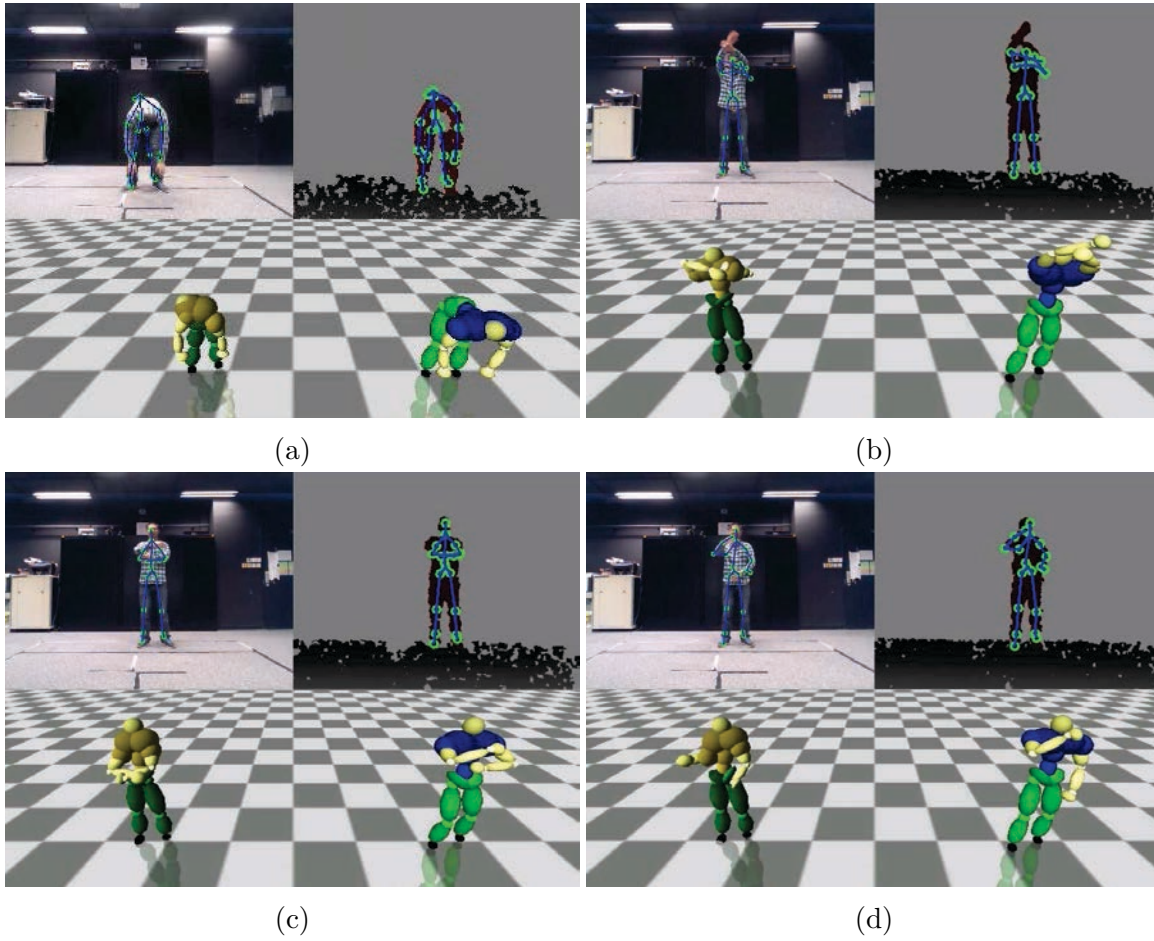


Figure 5.4: Data set I: postures from Kinect and their corresponding reconstructed postures. In each picture, the upper half shows the RGB and depth images, in which the blue skeleton is the tracked results from Kinect. The lower left and right parts represent the 3D Kinect posture and our reconstructed posture respectively. (a) Bending over; (b) Crossing arms; (c) Rolling hands forward and backward; (d) Rolling hands up and down

5.3.2 Qualitative Analysis

In this section, we evaluate the perceptual score for the correctness of postures reconstructed by our method, postures from Kinect, postures by the method proposed by [59], [85] and postures captured by an optical motion capture system.

In order to evaluate the perceptual correctness according to the user performed motion, we measure the perceptual score for the postures of each method using a survey-based evaluation. Such an experiment has also been performed in [62]

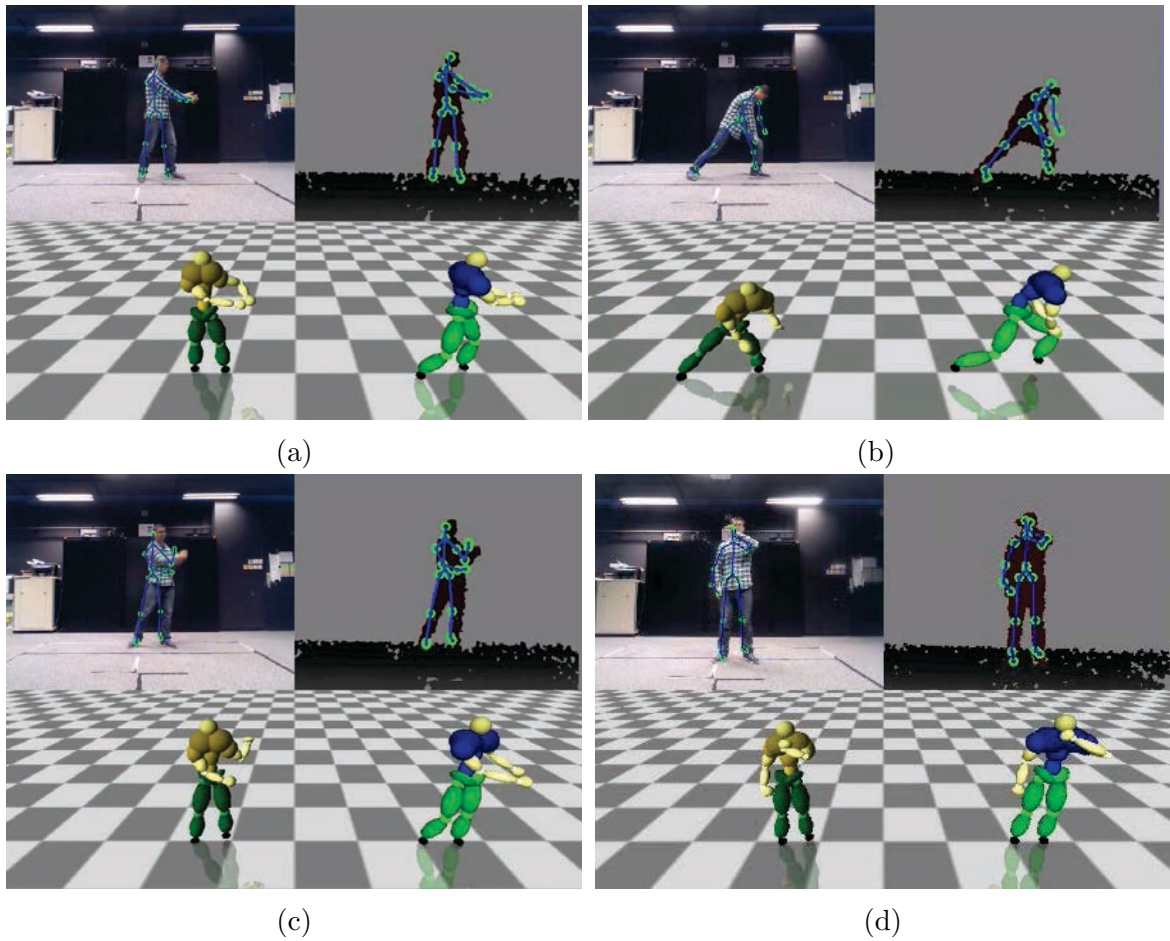


Figure 5.5: Data set II: postures from Kinect and their corresponding reconstructed postures. In each picture, the upper half shows the RGB and depth images, in which the blue skeleton is the tracked results from Kinect. The lower left and right parts represent the 3D Kinect posture and our reconstructed posture respectively. (a) Clapping hands; (b) Bending leg; (c) Golf swinging; (d) Waving left hand.

and [85]. Notice that while some recent research such as [28] analyzes how viewers perceive interactions between virtual characters, since the focus of our research is about posture reconstruction process from noisy data, we do not include detailed perceptual analysis in the scope of this research.

A total of 15 participants were invited to conduct this experiment. All of them had little or no experience about motion capture and 3D animation. The purpose of this experiment is to assess the relative correctness of the obtained postures from these five methods. We create a set of posture sequences with these five methods

together with the RGB video so that the participants know what the actual actions are. Participants were asked to give a score for each motion based on its correctness according to the performed motion without knowing what method is used. The score ranges from 1 to 10 (inclusive), where 1 means the most incorrect, and 10 means the most correct.

The score distribution for Kinect SDK [45], [59], [85], our method and *MOCAP* is shown in Fig. 5.6. The overall average scores of these five methods are 5.20, 6.42, 7.51, 7.49 and 9.16 respectively, and the standard deviations are 1.187, 0.578, 0.236, 0.240, and 0.255. As expected, *MOCAP* data achieve the best scores. We can see that our method performs better than Kinect and [59] in general. In particular, our method significantly outperforms Kinect and [59] for motions with more occlusions such as bending over and rolling hands, as shown in Fig. 5.4a, Fig. 5.4c, Fig. 5.6(b) and Fig. 5.6(h). The reason is that we embed the reliability term into our optimization framework, which implicitly ensures the system to recover these joints more than those with higher reliability. As shown in Table 5.2, our method generates postures of similar quality compared with [85] with a significantly smaller motion database. It should be noted that for motion that involves a large range of movement such as Tai Chi, our method could synthesize postures that are closer to the ground truth compared to [85]. This is because the weighted prediction of local models allows synthesizing postures that are not available in the motion database and more possible solutions are explored.

5.3.3 Quantitative Analysis

In this section, we quantitatively analyze the correctness of the proposed method. We assume the data from optical motion capture system is the ground truth data.

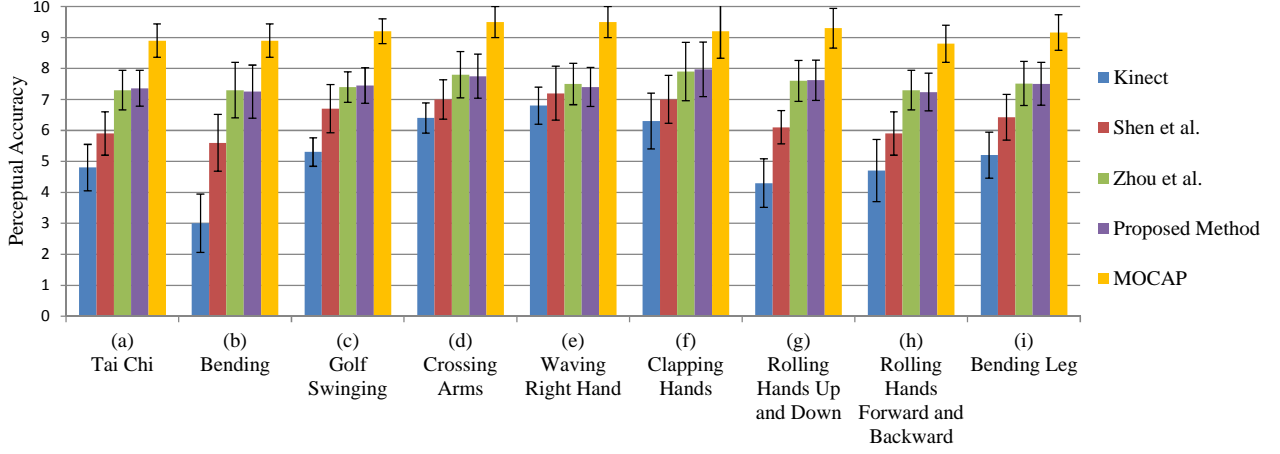


Figure 5.6: The perceptual score for the correctness of postures from Kinect, Shen et al. 2012, Zhou et al. 2014, proposed method, and an optical motion capture system.

To evaluate the accuracy of the reconstructed postures, we define an error function to measure the distance between reconstructed postures and ground truth postures:

$$E(F_1, F_2) = \frac{1}{IT} \sum_{i=1}^I E_i(F_1, F_2) \quad (5.23)$$

where F_1 and F_2 are the two sets of postures, I is the total number of joints, and T is the total number of postures. E_i is the reconstruction error of joint i between two set of postures, which is defined as:

$$E_i(F_1, F_2) = \sum_{t=1}^T D(F_{1t}^i, F_{2t}^i) \quad (5.24)$$

where F_{1t}^i is the i^{th} joint of the posture at time t from F_1 . D is the Euclidean distance between two joints of two postures:

$$D(P_1^i, P_2^i) = \sqrt{(P_{1x}^i - P_{2x}^i)^2 + (P_{1y}^i - P_{2y}^i)^2 + (P_{1z}^i - P_{2z}^i)^2} \quad (5.25)$$

With the error function defined in (5.23), we first study the influence of the

training size of each local model, \mathcal{S} , and the number of local models, L , on the reconstruction error. Fig. 5.7 shows that when we fix the \mathcal{S} for each local model, the 3D joint reconstruction error decreases as the L increases. Similarly, for any specific L , the system accuracy can be enhanced by increasing \mathcal{S} . However, the improvement is not significant when \mathcal{S} is raised to 50 and L reaches 9, because the postures become redundant and do not contribute to the reconstruction process. Thus, the value of \mathcal{S} and L are empirically set to 50 and 9 respectively.

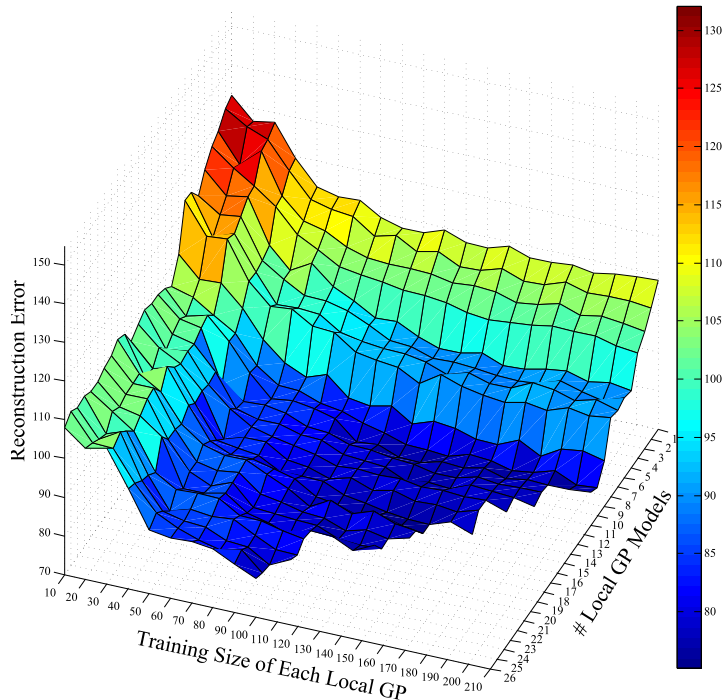


Figure 5.7: Influence of the training size and the number of local GP models on the 3D joint reconstruction error.

As an example, Fig. 5.8 shows the trajectory of the left hand in a golf swinging movement using offline local GPs (LGP-offline) and local GPs with incremental updating (LGP-incremental). We can see that the ground truth data (*MOCAP*) is smooth and the Kinect data is noisy due to the self-occlusions and sensor error. The mean error of Kinect, LGP-offline, and LGP-incremental is 12.36, 8.1, and 7.7 cm. Compared with LGP-offline, the LGP-incremental is closer to the ground truth in

general, which verify the effectiveness of the incremental learning framework. We can also see that a small number of local GP models ($L = 9$) is sufficient to reconstruct postures.

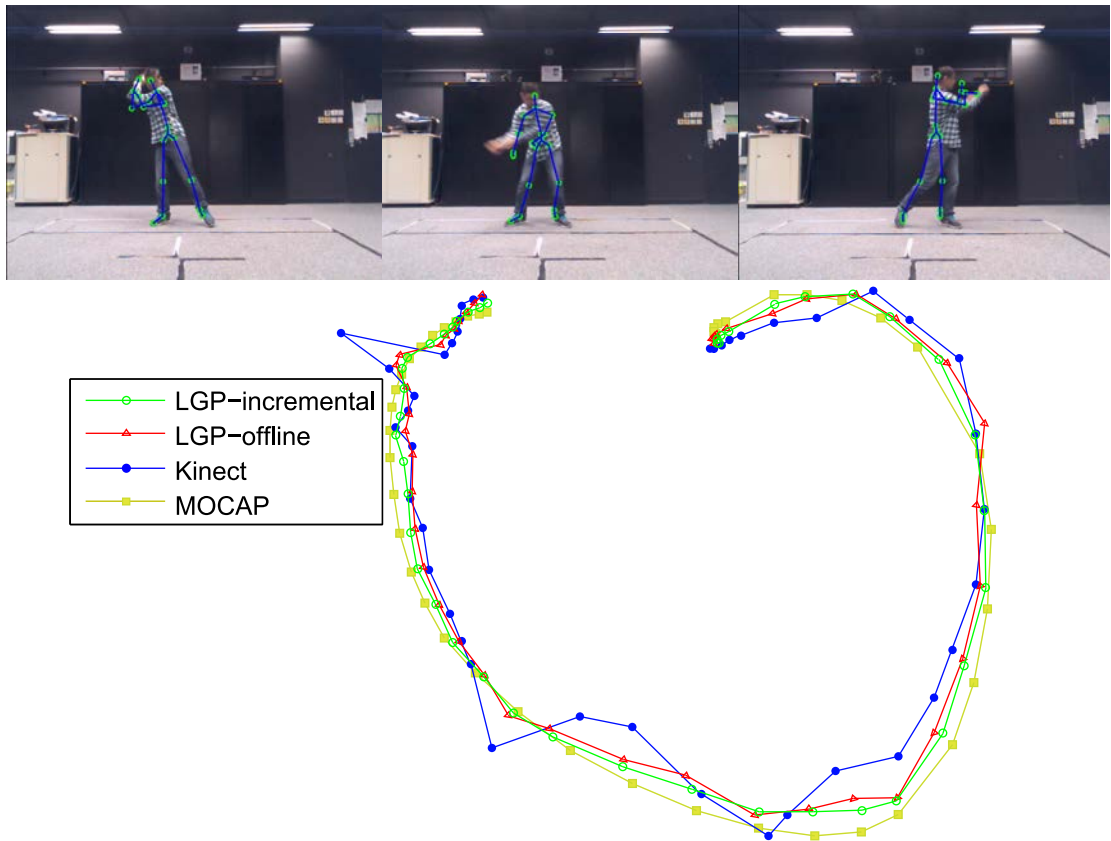


Figure 5.8: Trajectory of the left hand when performing golf swinging motion.

The number of frames in the training database used in our method, [59] and [85] are reported in Table 5.2, which shows that our database is 35% smaller than that of Zhou et al. [85], and 83% smaller than that of Shen. et al. [59]. In addition, our newly proposed local mixture of GPs algorithm allows us to combine all types of training motion in Table 5.2 as a single database, while in [85] a separate training database is built for each type of motion.

More comparisons of different type of testing motions between LGP-offline, LGP-incremental, [59], and [85] can be found in Table 5.3. Here we choose 5 types

Table 5.2: The number of frames for each type of training motion database used in our method, Shen et al. 2012, and Zhou et al. 2014.

Motions	Our method	Zhou et al. 2014	Shen et al. 2012
Tai Chi	411	650	2320
Bending	201	320	1580
Golf Swinging	296	460	1765
Crossing Arms	245	380	1685
Waving Right Hand	221	350	1650
Clapping Hands	270	420	1720
Rolling Hand Up and Down	308	480	1840
Rolling Hand Forward and Backward	306	475	2050
Bending Leg	243	385	1890
Mixed motion database	2501	-	16500

of motion for evaluation: clapping hands, crossing arms, bending, Tai Chi, and waving right hand. As expected, the error of Kinect was large in general. Our method outperforms [59] as we take into account the reliability of each joint such that the inaccurately tracked joints will not guide the system to infer the postures. For all classes of motions, our method consistently outperforms the Kinect and [59], which verifies the effectiveness of the proposed method in terms of reconstruction accuracy. It should be noted that the LGP-incremental can generate comparable system accuracy compared to [85] while the running time is less than [85]. The computational time of [85] and our system are 37 and 29 ms per frame, respectively.

Table 5.3: Reconstruction error of Kinect, Shen et al. 2012, Zhou et al. 2014, and the proposed method on the testing data sets.

Motion Type	Number of Frames for Testing	Kinect (cm)	Shen et al. (cm)	Zhou et al. (cm)	Proposed Method (cm)	
					LGP-offline	LGP-incremental
Crossing Arms	2052	12.5	9.8	7.2	7.9	7.4
Bending Over	1835	13.7	9.5	8.4	9.2	8.7
Tai Chi	2885	14.5	10.2	7.5	8.0	7.4
Waving Right Hand	1568	12.5	8.8	6.5	6.9	6.6

5.3.4 Comparison Between Randomized Forests and Our Method

In this particular experiment, we do not use Kinect SDK to extract joint positions. To ensure a fair comparison between our method and randomized forests, which is the method used to train Kinect, we construct a common training database for both methods.

Our database contains a large number of synthetic depth images that are created as follow. First, we create a 3D mesh model in which each body part is labeled. Second, we retarget *Mocap* data to drive the movement of the 3D mesh model. Third, we render depth information of the scene into depth images frame by frame. Since our 3D model comes with body part labels, we can automatically label body part information for each pixel in the rendered depth images. Finally, we trained the randomized forests with the labeled depth images and estimated the joint positions using mean shift. We also trained our GP models with the same data and the joint positions found by mean shift using the body parts estimated by randomized forests. Our training database consists of 17K synthesized depth images generated by *Mocap* data, including actions such as golf swing, waving hands, crossing hands and clapping hands.

We use five-fold cross validation to compare the performance of randomized forests [60] and our algorithm. Table 5.4 shows the comparison of average reconstruction error. It can be observed that both methods have similar performance for simpler motions such as T-pose. However, for more challenging motions that involve self-occlusion such as crossing hands and golf swing, our method generates better reconstruction results with smaller reconstruction error.

To better observe the relationship between reconstruction error and reconstructed

Table 5.4: Reconstruction error of randomized forests and our method using five-fold cross validation (cm).

Motion Type	Reconstruction Error with Randomized Forests	Reconstruction Error with Our Method
Crossing Hands	13.8	8.1
Golf Swinging	14.9	9.4
Waving Hands	13.5	7.3
Rolling Hand Left and Right	13.7	7.4
Bending Over	14.8	9.3
Clapping Hands	14.3	8.5
T-pose	7.8	7.1

postures. We use the error function defined in Equation 5.23 for each joint and plot the reconstruction error across frames. Figure 5.9 shows the joint errors of two example motions. We can observe that our method can achieve lower reconstruction errors compared with randomized forests.

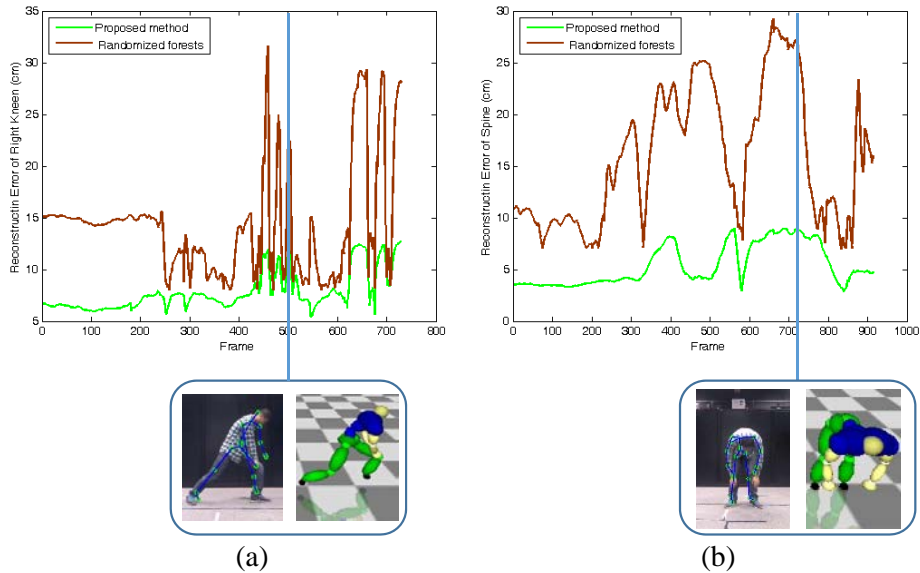


Figure 5.9: Examples of the reconstruction error of one joint across frames. The green curve corresponding to our method, brown curve is the reconstruction error of randomized forests. (a) Bending leg motion. (b) Bending over motion.

In this experiment, we feed the same training data to generate a fair comparison between randomized forests and our method. However, the reconstruction error of

randomized forest is larger than that of Kinect and this is mainly because of the difference in the depth and number of trees. Microsoft suggests that two trees each to depth 10 will generally achieve accepted results, which is the depth and number of trees we use in our experiment. They also reported that they trained six trees each to depth 20. This leads to much more accurate tracking results but with a large extra memory and computation resource, which takes them about a day on a 1000 core cluster. However, we do not have so much computation resource to do such an experiment, thus, we only train two trees each to depth 10, which will not result in the best accuracy. That is major reason why the accuracy of our random forest is slightly worse than that of Kinect.

5.3.5 Effects of Optimization Terms

In this section, we analyze the reconstruction accuracy by examining the effectiveness of different terms in the objective function of (5.22). We used Tai Chi motion, bending over and crossing arms for evaluation because of their complicated movement features. The results are reported in Table 5.5.

Table 5.5: Reconstruction error of the proposed framework with different constraint terms.

Setup	Terms Used	Reconstruction Error (cm)
(a)	E_S	12.0
(b)	E_S, E_T	10.5
(c)	E_S, E_R	9.7
(d)	E_S, E_T, E_R	7.9

We found that both the temporal prior term and the reliability term improve the reconstruction accuracy, especially for the movements with severe self-occlusions. Although setup (c) achieves better results than setup (b), the obtained movements are jerky, because setup (c) predicts postures independently without considering

relationship between consecutive frames.

5.4 Summary

In this chapter, we propose a new real-time probabilistic framework to enhance the accuracy of live captured postures that belong to one of the action classes in the database. We adopt the Gaussian Process model as a prior to leverage the position data obtained from Kinect and marker-based motion capture system. We also incorporate a temporal consistency term into the optimization framework to constrain the velocity variations between successive frames. To ensure that the reconstructed posture resembles the accurate parts of the observed posture, we embed a set of joint reliability measurements into the optimization framework.

A major drawback of Gaussian Process is its cubic learning complexity when dealing with a large database due to the inverse of a covariance matrix. To solve the problem, we propose a new method based on a local mixture of Gaussian Processes, in which Gaussian Processes are defined in local regions of the state space. Due to the significantly decreased sample size in each local Gaussian Process, the learning time is greatly reduced. At the same time, the prediction speed is enhanced as the weighted mean prediction for a given sample is determined by the nearby local models only. Our system also allows incrementally updating a specific local Gaussian Process in real time, which enhances the likelihood of adapting to run-time postures that are different from those in the database. Experimental results demonstrate that our system can generate high quality postures even under severe self-occlusion situations, which is beneficial for real-time applications such as motion-based gaming and sport training.

Chapter 6

Application: An Interactive Shape Morphing System

In this chapter, we build an interactive shape morphing system based on the three techniques discussed in the thesis: compatible triangulation, consistent rotation enhanced rigid shape interpolation and posture reconstruction. In Section 6.1, we present the interactive shape morphing system. We discuss the problem of shape interpolation with self-occlusion and our solution in Section 6.2.

6.1 Interactive Shape Morphing System

We have implemented a prototype of the proposed interactive shape morphing system. Fig. 6.1 shows the setup of our interactive shape morphing system. We use Kinect as the input device of the source shape. The user stands in front of the Kinect and the system can be controlled by gesture command. For example, the system starts to capture and extract the shape of the user when the user in the scene raising his/her left hand over the head. More commands such as raising two hands to go back to the

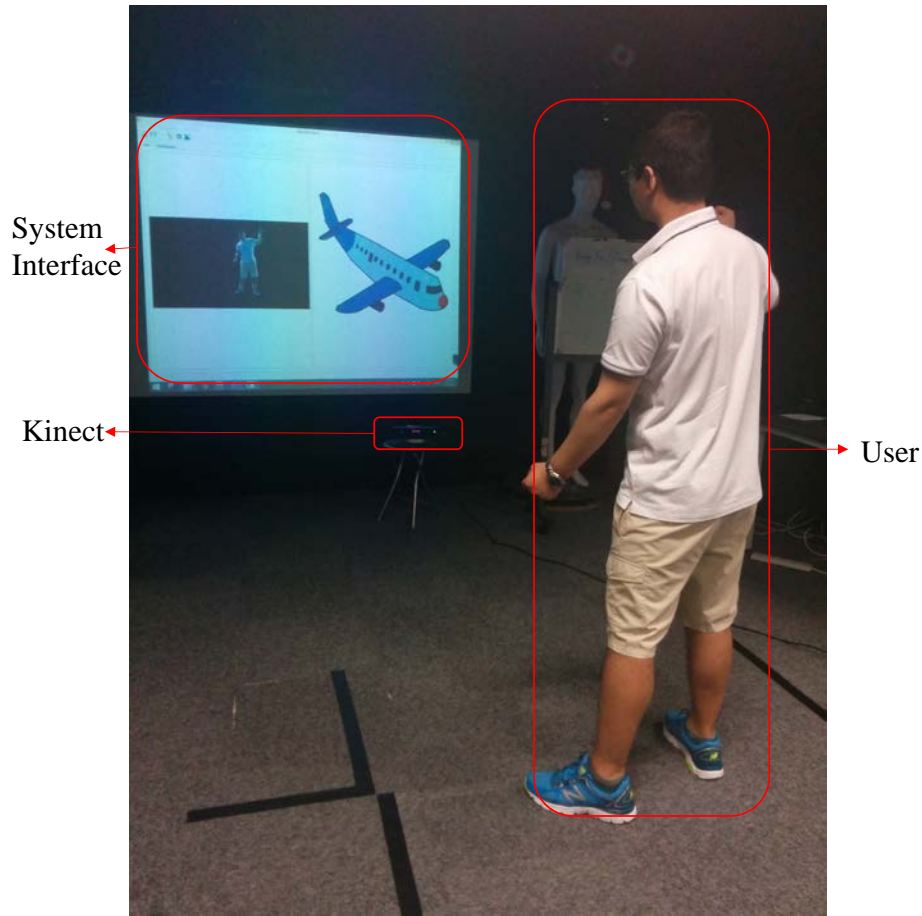
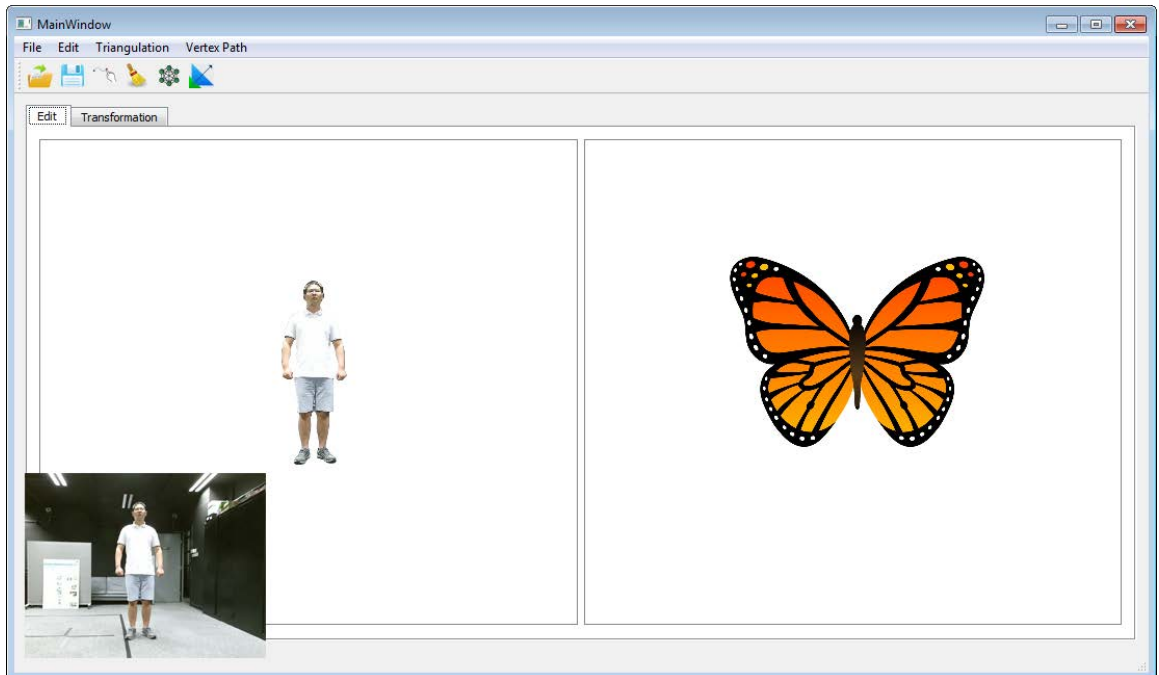


Figure 6.1: The setup of the interactive shape morphing system.

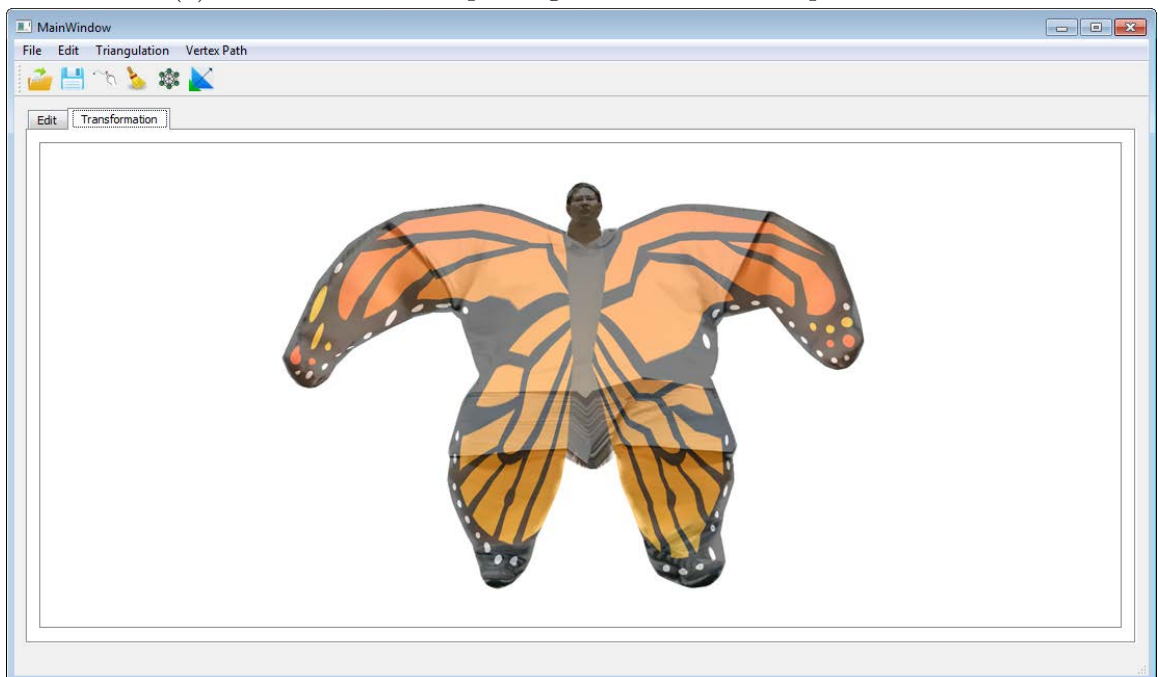
default capture view have been added to the system.

Fig. 6.2 shows the interface of the interactive shape morphing system. As shown in the bottom left of Fig. 6.2a, the user adopts a pose as the source input shape. The user could select the target shape in the shape database with certain gesture such as waving hand left, then the target shape is rendered in the right of *Edit* window in Fig. 6.2a. We then compute the compatible triangulation between the source and target shapes. Finally, we transform the source shape into the target shape based on the compatible mesh. The intermediate results are shown in the *Transformation* window as shown in Fig. 6.2b.

Our interactive shape morphing system can be applied to create animation, movie



(a) The interface for capturing the calibration shape of the user.

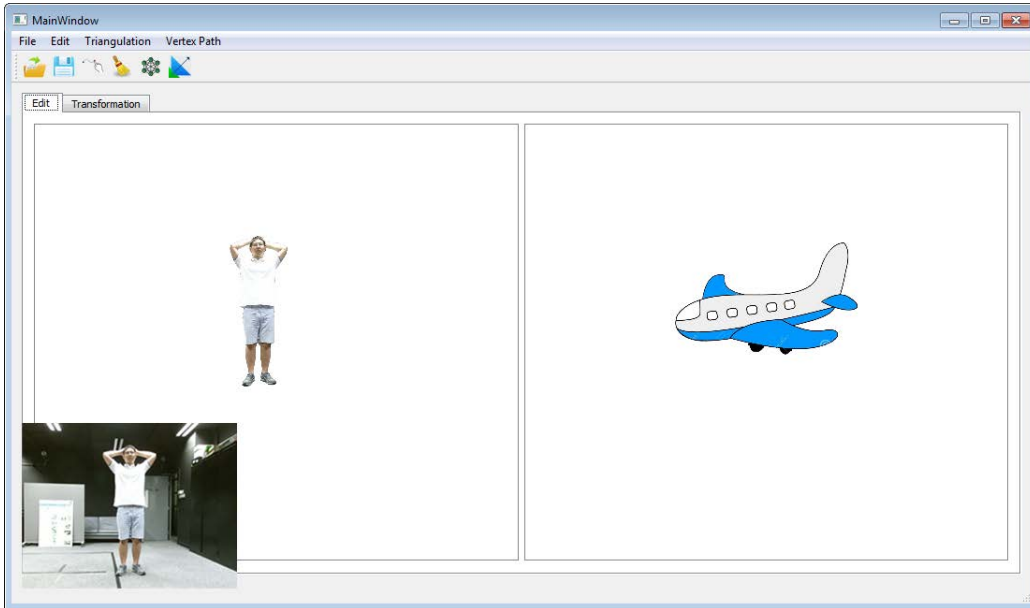


(b) The interface for showing the intermediate transformations.

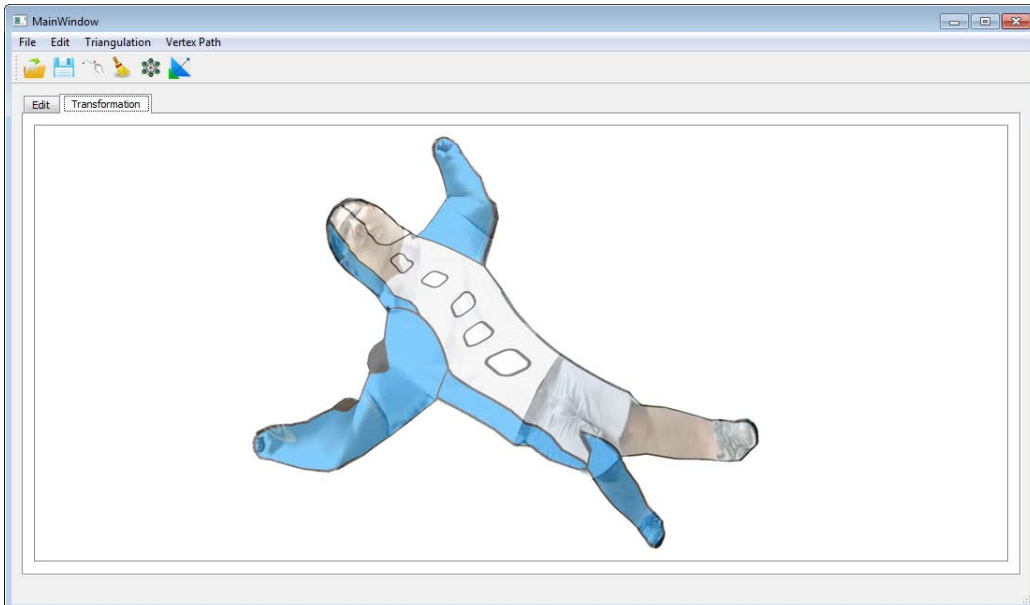
Figure 6.2: Prototype of our interactive shape morphing system

and even special effects software packages. For example, the Monkey King has seventy two transformations in the Journey to the West, we could apply our system to

generate these transformations. On the other hand, the normal users usually do not have professional resources to generate some interesting morphing video, our method will make it easier to generate the morphing video.



(a) The interface for capturing the shape of the user with self-occlusion.



(b) The interface for showing the blending results.

Figure 6.3: Additional example of the interactive shape morphing system.



(a) Shape with self-occlusion



(b) Target shape

Figure 6.4: The compatible triangulation between a shape with self-occlusion (a) and target shape (b)

6.2 Shape Morphing with Self-occlusion

6.2.1 Problem of Shape Morphing with Self-occlusion

As shown in Fig. 6.4a, the user adopts a pose with self-occlusions, e.g. with right hand placed in front of the torso and left hand behind it. We apply the compatible triangulation method discussed in Chapter 3 that generates the compatible mesh as shown in Fig. 6.4 with mesh highlighted in blue color. However, these meshes such as Fig.6.4a cannot distinguish the hands and the other body parts. We apply the rigid shape interpolation method introduced in Section 4.1.3 to blend the mesh as shown in Fig.6.4. The transformations are shown in Fig. 6.5. We can see the transformations of the in-between images such as the time slice $t=0.2857$ does not

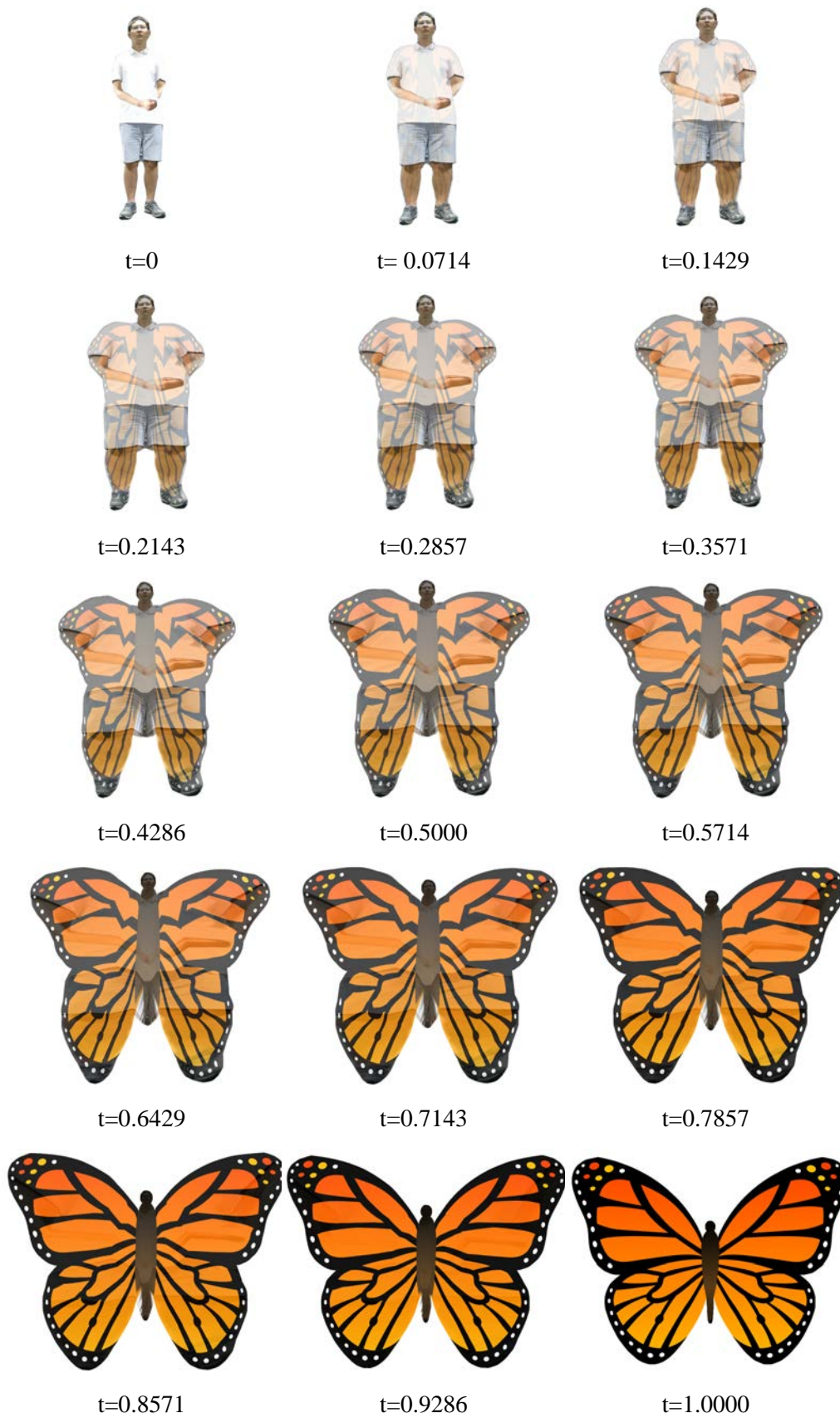


Figure 6.5: Examples of shape interpolation with self-occlusion.

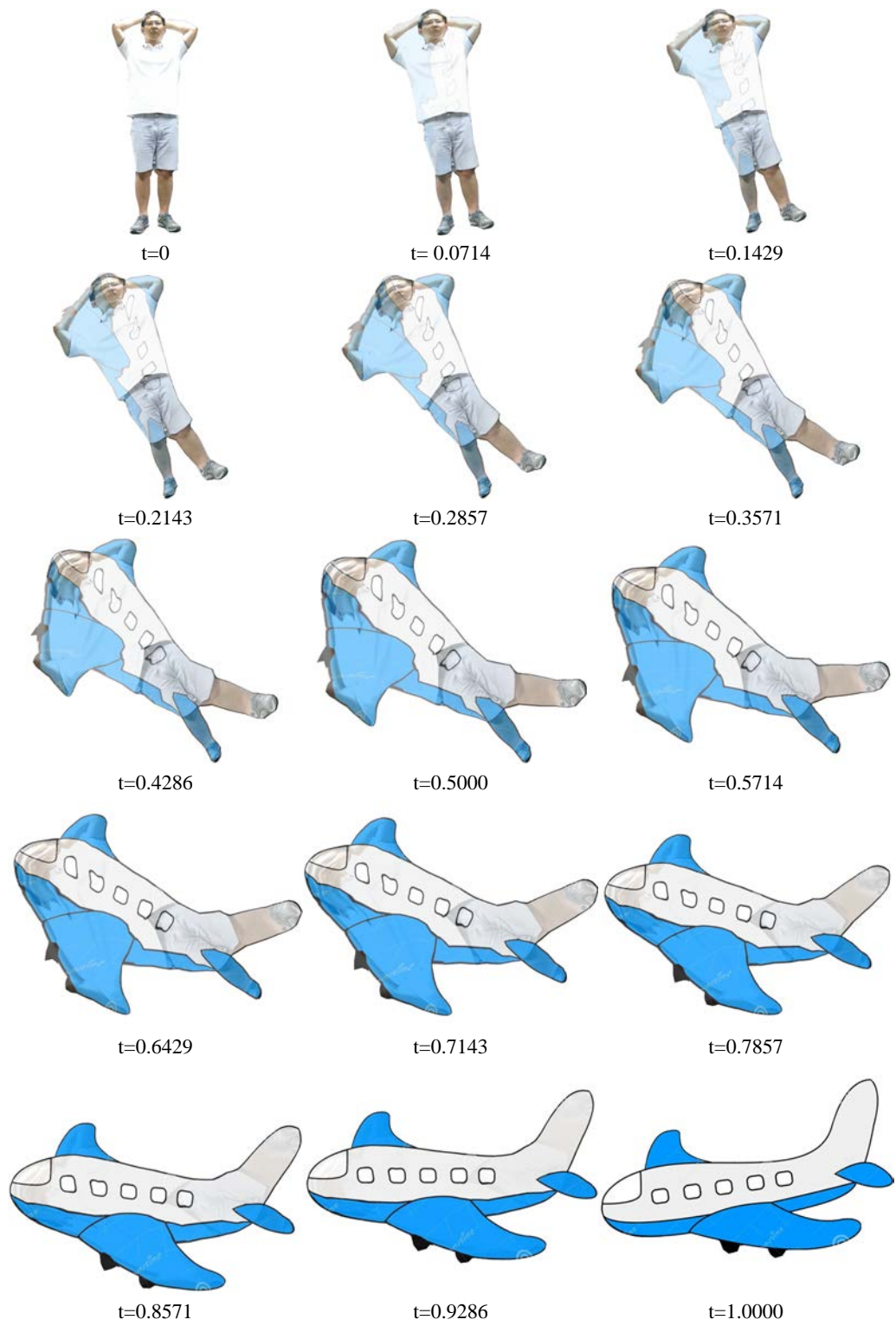


Figure 6.6: Additional example of shape interpolation with self-occlusion.

make sense, because we want the hand of the user to be gradually transformed into the wing of the butterfly. Fig. 6.6 shows another example of shape morphing with self-occlusions that has the similar problem to Fig. 6.5.

6.2.2 Shape Morphing with Self-occlusion Enhanced

In order to generate sensible transformation, we need to build compatible triangulation with self-occlusion. To do so, we can identify overlapping body parts using joints from Kinect. However, data from Kinect are noisy and unreliable, thus, we propose a posture reconstruction method to enhance the quality of joint position obtained from Kinect. In this section, we discuss how to combine compatible triangulation, rigid shape interpolation, and posture reconstruction techniques to compute shape morphing with self-occlusion such that the transformations make sense.

Compatible Triangulation with Self-occlusion. First, we capture a calibration shape of the user as shown in Fig. 6.7a and build the compatible triangulation between the calibration shape and target shape as shown in Fig.6.7c-d. Second, we deform the mesh of the calibration shape as shown in Fig.6.7c into the source shape that involves self-occlusion as shown in Fig. 6.7e, which implicitly build the bijection between the source shape with self-occlusion as shown in Fig.6.7f and target shape as shown in Fig.6.7b.

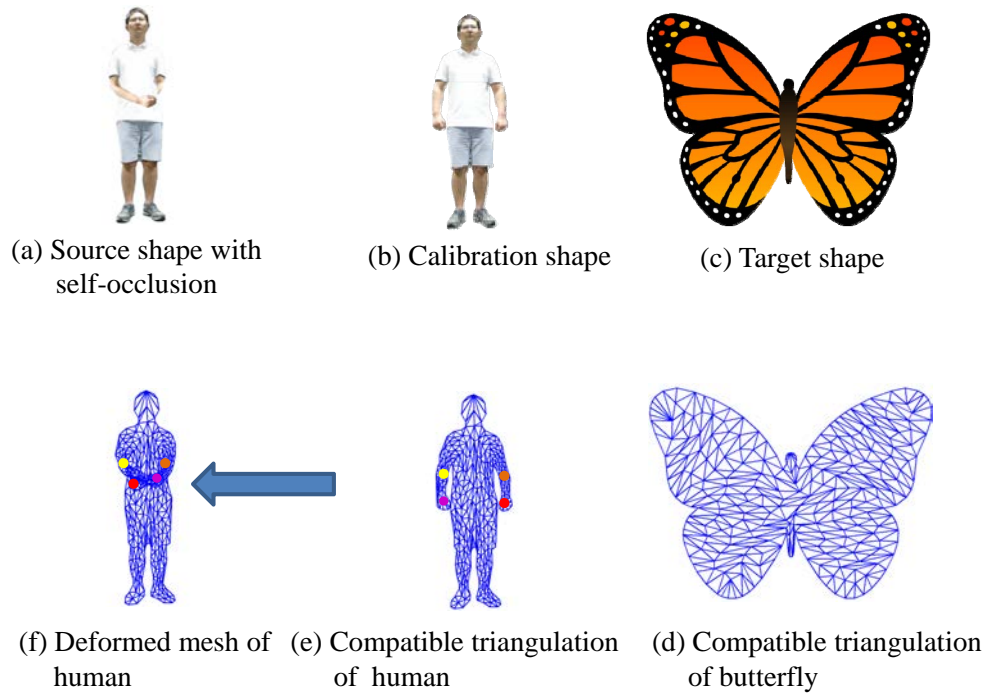


Figure 6.7: Overview of compatible triangulation for shapes with self-occlusion. Our inputs are (a) The source shape with self-occlusion and (c) The target shape. (b) We introduce calibration shape. (d-e) We build the compatible triangulation between the calibration shape (b) and target shape (c). We deform the calibration mesh (e) into the source shape with occlusion (a) using the reconstructed joint positions estimated from our posture reconstruction method.

Collision detection and depth adjustment.

As the order of polygon partition algorithm is similar to the breadth first search method, thus, the triangles are not stored in sequence as illustrated in Fig. 6.8. We must be careful when different parts of the shape overlap. If we assign depths inappropriately, the overlapping parts can interpenetrate as shown in Fig. 6.9b. We continuously monitor the mesh for self-intersection and assign appropriate depth values to the overlapping parts. The depth value we assigned to each triangle is estimated from the posture reconstruction algorithm. As we have recovered the joint positions for each human body joint, we know if the hands are in front of or behind the spine as shown in Fig. 6.9c.

As shown in Fig. 6.10, we blend the human with self-occlusion and the butterfly.

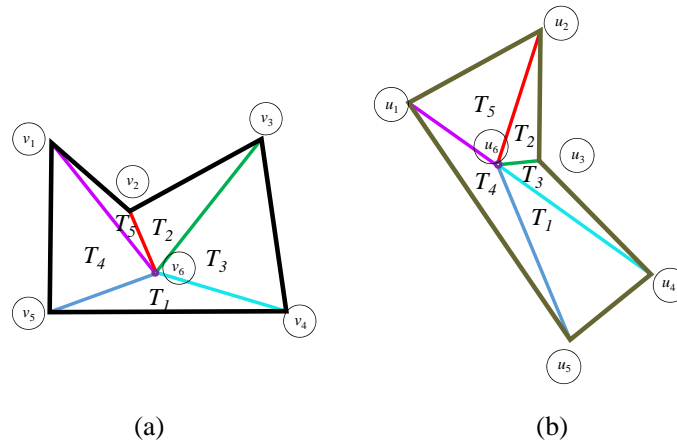


Figure 6.8: The triangles are not stored in order for compatible triangulation of the source (a) and target shape (b).

Compared with the transformations that do not consider body parts overlap as shown in Fig.6.5, the results in Fig. 6.10 make more sense as we now transform the human's limbs into the butterfly's wings. Fig. 6.11, 6.12 and 6.13 show additional examples of shape morphing using our method.

We can recommend shapes in the shape database that are similar to the user and this can be done by using shape matching methods such as calculating the geodesic distance, Hausdorff distance, and Frechet distance.

6.3 Summary

In this chapter, we present an interactive shape morphing system based on the compatible triangulation, consistent rotation enhanced rigid shape interpolation and posture reconstruction algorithm. The compatible triangulation determines the bijection between the source and target shapes. The rigid shape interpolation determine the vertex trajectory. The posture reconstruction method detects the body parts overlap and assigns appropriate depth to the occluded parts. Experimental results show that our system is easy to use and the transformations are sensible.



(a) Calibration shape



(b) Without depth adjustment



(c) With depth adjustment

Figure 6.9: Collision detection and depth adjustment. Without appropriate depth assignment, one can see interpenetration (b). We detect overlapping regions and adjust depth on the fly (c).

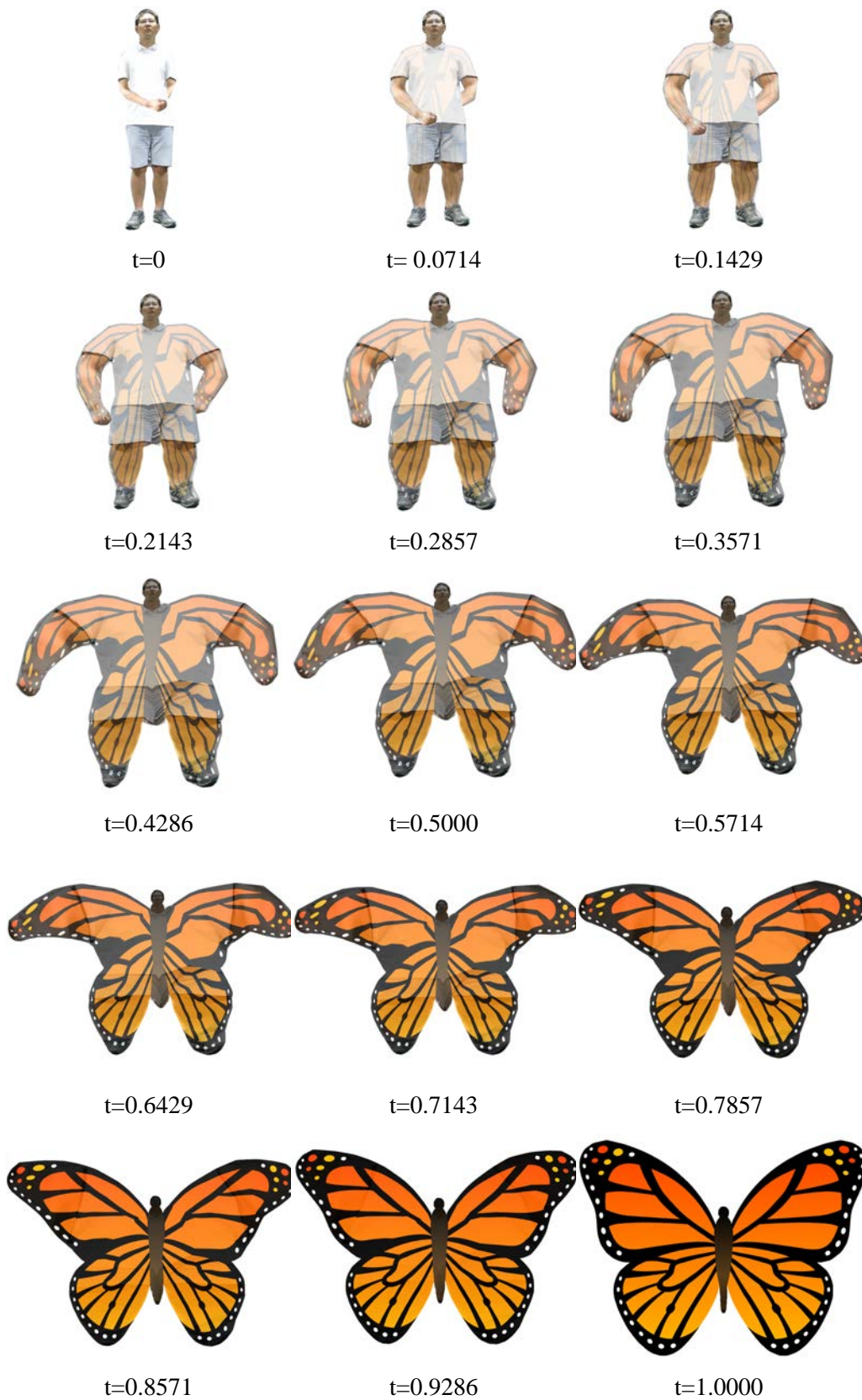


Figure 6.10: Shape interpolation with self-occlusion enhanced using posture information.

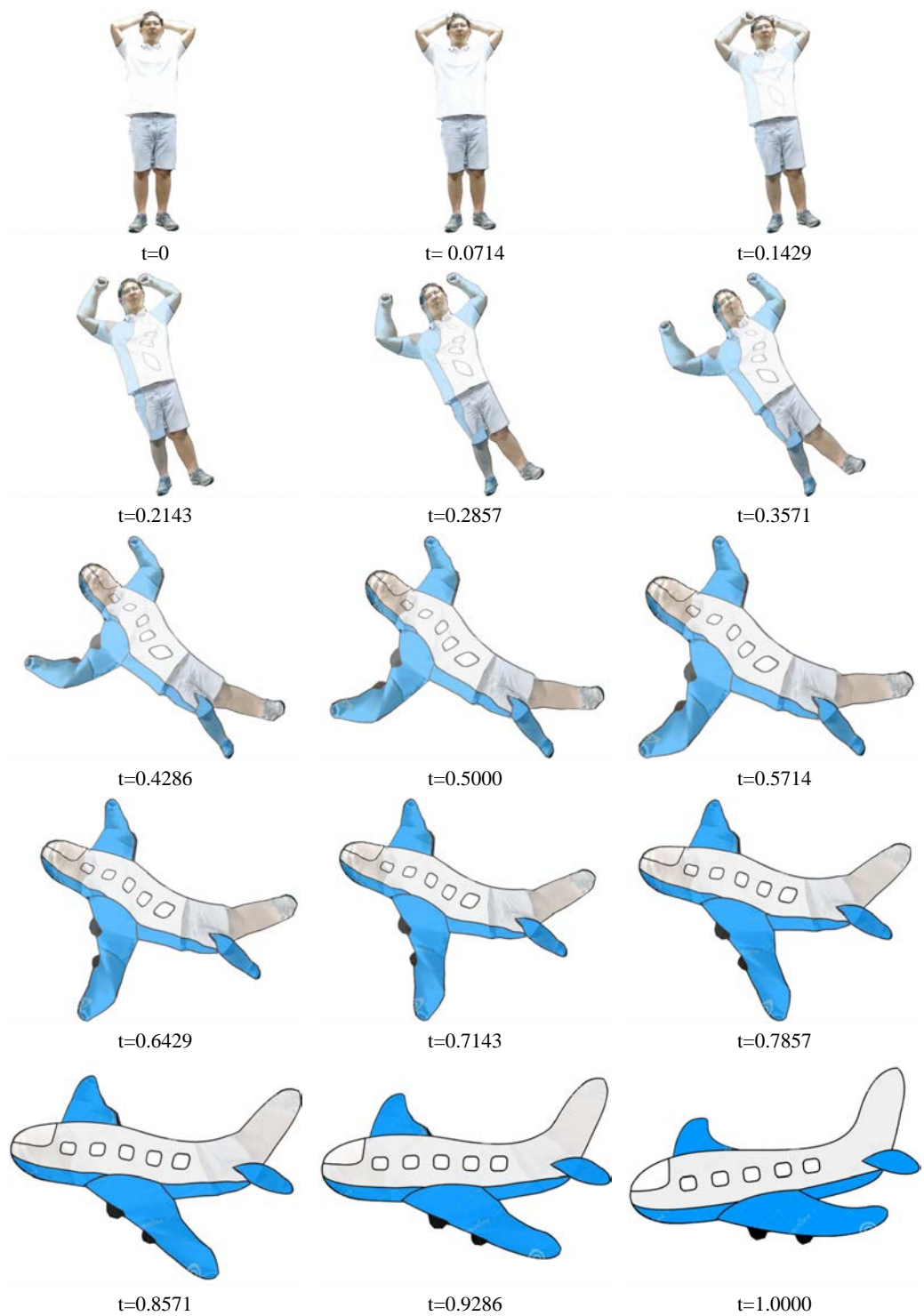


Figure 6.11: Additional shape morphing example I: enhancing shape interpolation with self-occlusion.

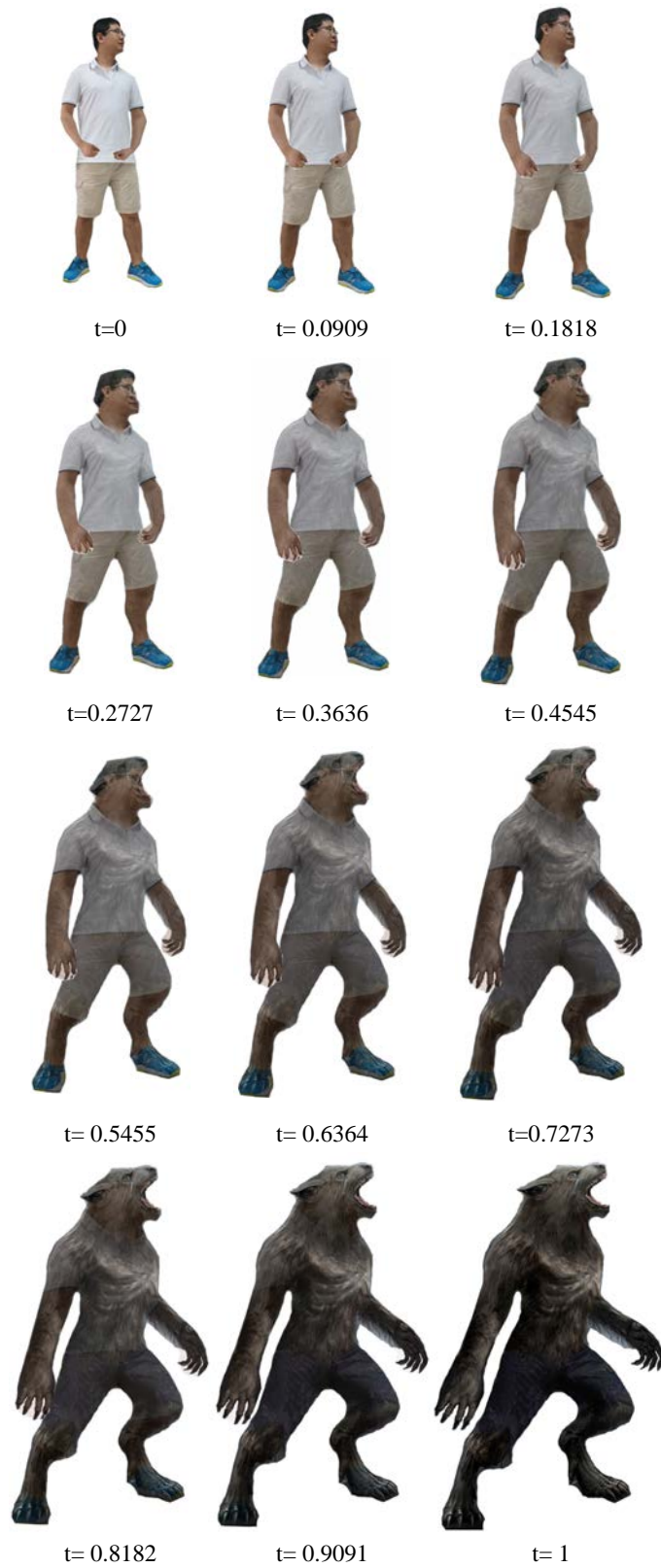


Figure 6.12: Additional shape morphing example II: transforming a man to one wolf beast.

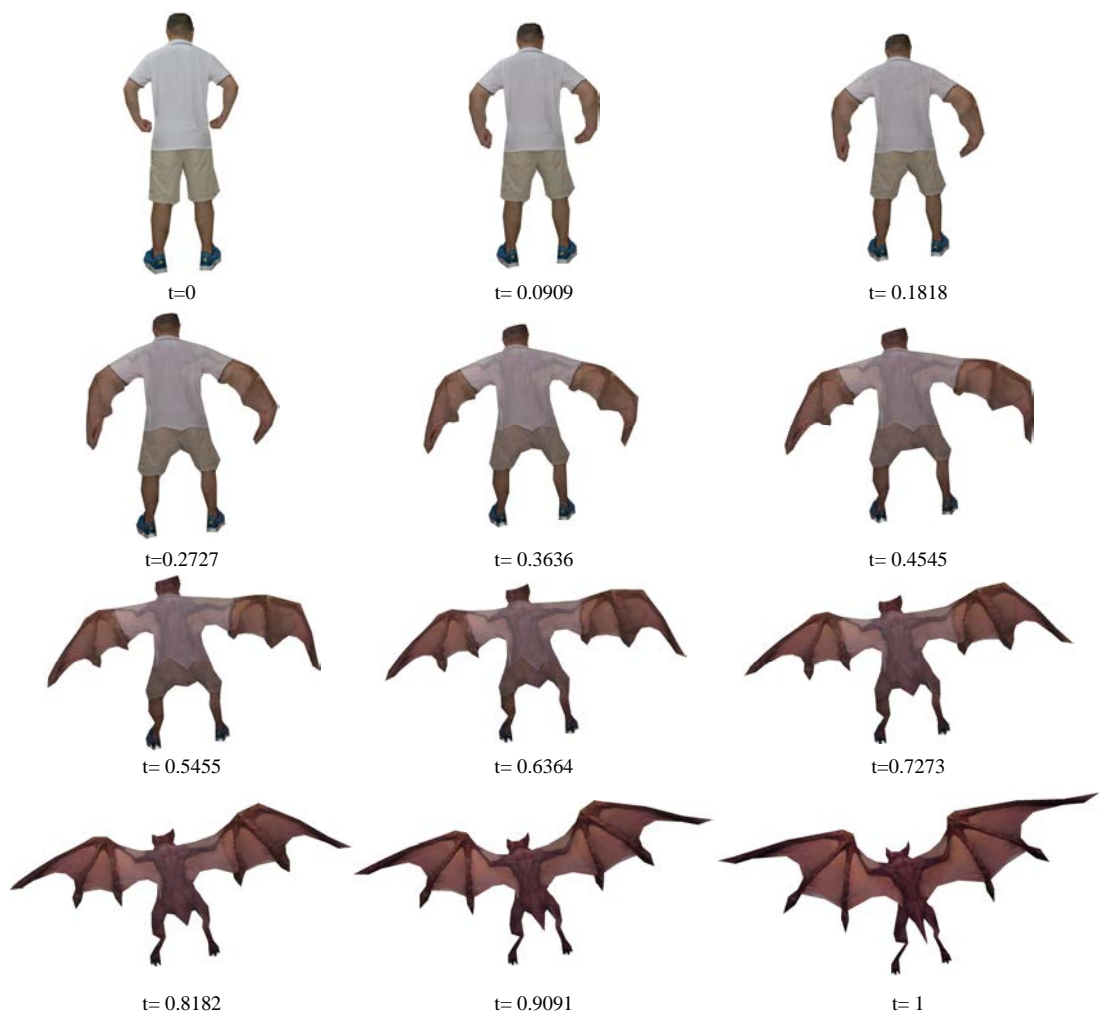


Figure 6.13: Additional shape morphing example III: transforming a man to one bat monster.

Chapter 7

Conclusions and Future Directions

This thesis has been devoted to the study of planar shape morphing with self-occlusion by using compatible triangulation, consistent rotation enhanced rigid shape interpolation, and human posture reconstruction techniques. As concluding remarks, we recap on our main contributions and discuss some interesting future work.

7.1 Conclusions

Here, we summarize the contributions of this thesis in three groups of techniques: compatible triangulation, rigid shape interpolation, and human posture reconstruction.

- 1. Compatible Triangulations.** We propose a new method for computing compatible triangulation of two simple polygons and apply them to 2D shape morphing and texture mapping. Our method compatibly decomposes the source and target polygons into sub-polygon pairs and map the triangulations between a pair of sub-polygons with a sparse linear system.

As an improvement from previous methods, our compatible polygon decomposi-

tion algorithm offers more flexible way of decomposing the source and target polygons such that the minimum interior angle can be maximized in each iteration. This leads to compatible triangulations with more regular-shaped triangles (as opposed to long thin triangles) as illustrated by the fact that there are fewer triangles whose minimum angles are small under our approach compared with other methods in [71, 7, 43]. Second, compared with our preliminary work [43], our method generates the same compatible mesh no matter we start the decomposition from the source or target polygon. Third, with our new mesh quality evaluation metric, the mesh generated by our method usually experiences less distortion for applications such as shape morphing and texture mapping. Another advantage is the simplicity of the three stages that all we need is to decompose a polygon, calculate link paths, and solve a sparse linear system, which enables real-time morphing.

2. Planar Shape Interpolation. The rigid shape morphing algorithm may suffer from inconsistent rotation whenever the rotation is more than π . We offer an efficient algorithm that gives a unique rotation assignment with minimum rotation angle. We construct a graph and treat each original rotation angle β as one vertex of the graph. We keep searching the graph and adjusting any adjacent rotation that is inconsistent with the current rotation until all the vertices have been traversed. After this process, our method would find a solution to fix these discontinuity. All the correct rotations in this thesis are generated by this efficient scheme. Although we cannot prove our method can always find a consistent rotation assignment, it works well for all the results in the thesis.

3. Human Posture Reconstruction. In this thesis, we present a probabilistic framework to reconstruct live captured postures from Kinect. Postures from Kinect are noisy, however, such a noise is not a random signal and we observed that there are

some underlying patterns in it. In this research, we can thus assume that the noisy data contain useful information in helping us to find the solution. Then, we apply a machine learning algorithm to learn the correlation between Kinect data and *Mocap* data so as to predict the offset given Kinect data. Finally, we verify our assumption with accurately reconstructed posture results.

To overcome the problem of incorrectly tracked and missing joints in Kinect, we adopt Gaussian Process (GP) model as a spatial prior to leverage position data obtained from Kinect and an optical motion capture system. Specifically, we model the residual offset between postures obtained from Kinect and *MOCAP* system instead of using pairwise posture relationship. While GP works well in small training data sets, it is not competent in systems that require a large database, such as motion-based gaming, due to its high computational complexity. To solve this problem, we propose a new method based on the local mixture of Gaussian Processes to speed up the learning and prediction. Our system allows incrementally updating of local models in real time, which boosts the reconstruction accuracy of run-time postures that are different from those in the database.

7.2 Future Directions

Although the methods presented in this thesis have achieved fruitful results on planar shape morphing, there are several possible directions for works presented in this thesis to be further studied.

- 1. Compatible Triangulations.** One drawback of our compatible triangulation algorithm is that we cannot deal with the polygon with holes. One possible solution is that we add a *bridge* between the outer polygon and inner polygons (i.e. the holes).

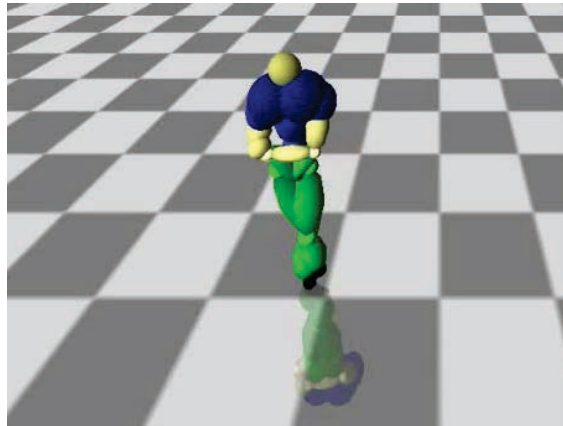
We connect the outer polygon with all the holes such that we can treat a polygon with holes as a single polygon. We can then apply our previous method to compatibly decompose the source and target polygons. While we have shown many examples of compatible triangulation in the thesis, we also want to test our algorithm on shapes with complex structure or completely different topology in the future.

2. Planar Shape Interpolation. While our compatible triangulation method well handles the mapping between shapes, the morphing results need to be further improved. As we focus on generating compatible mesh, we simply crossfade between textures in image space. More sophisticated texture blending or image warping algorithms such as [55] could be incorporated into our method. Currently, the intermediate images interpolated are uniquely determined by rigid interpolation method [2], which offers no means of controls. It would be desirable to modify some parts of the intermediate shapes if the users are not satisfied with them. We could explore possible solutions such as the linear constraints proposed in [6] to increase user creativity.

3. Human Posture Reconstruction. For our posture reconstruction method to work well, the motion performed by the user should belong to one of the action classes in the database. The proposed method is useful for real-time applications such as motion-based gaming and sport training where the user is expected to perform a motion from a set of common moves that are known in advance. While our system utilizes neighboring joints for prediction, it is difficult to deal with heavily occluded postures such as turning around, in which there are only few valid joints. As shown in Fig. 7.1, Kinect incorrectly recognizes the posture. The incorrect joint positions and the decrease of reliability of body joint greatly impact the recognition quality. In such cases, the amount of correct data present is so little that our system cannot



(a)



(b)

Figure 7.1: Turning around motion. (a) The RGB image of turning around motion (facing backward); (b) The tracking result of Kinect corresponding to (a) (facing forward).

produce very good result. One possible solution would be using multiple Kinects to capture postures from different directions.

There is still room to improve the proposed reconstruction system. The assigned weights for the terms in the objective function are empirically set to be fixed in the proposed system. However, the weights can be different for different types of motion to obtain optimal reconstructed postures. One possible solution would be to formulate the weights w as a function of the residual offset Y_t , which is used to measure the importance of each term. Assuming $Y_i(t)$, $Y_i(t+1)$, and $Y_i(t+2)$ to be the 3-D offset of a tracked body part i in three successive frames, we can calculate

the displacement vectors as:

$$\begin{aligned} d_i(t-1) &= Y_i(t) - Y_i(t-1) \\ d_i(t-2) &= Y_i(t-1) - Y_i(t-2) \end{aligned} \quad (7.1)$$

The acute angle between the two vectors can be calculated by their dot product as:

$$\theta_i(t) = \frac{d_i(t-1) \cdot d_i(t-2)}{|d_i(t-1)| |d_i(t-2)|} \quad (7.2)$$

We then define the temporal weight as $E_T = \theta(t)$. The smooth change of E_T for successive frames indicates small weight for temporal term. On the other hand, we use the variance $v_i = \text{variance}(Y_i(t))$ of all the offset value for joint i at time slice t to determine the importance of reliability term, which can be calculated by the normalized equation as:

$$E_R = \frac{v - \text{mean}(v)}{\text{std}(v)} \quad (7.3)$$

Therefore, the weights can be adaptively determined according to the type of motion. The incorporation of physical constraints into the proposed framework is another interesting direction as the reconstructed postures in this work are not necessary physically correct. One possible implementation would be modeling the physical attributes (i.e. force field) between the Kinect data and *MOCAP* data as a prior distribution, and embed them in the optimization framework to generate physically valid postures. Last but not least, integrating our system with other simple yet stable devices such as inertia-based Mocap system would be an interesting topic, because Kinect can only detect limited range of movements while motion sensor can be used as a complement, e.g. detecting the occluded body part.

List of Publications

Journal Papers

1. **Zhiguang Liu**, Liuyang Zhou, Howard Leung and Hubert P.H. Shum, "Kinect Posture Reconstruction based on a Local Mixture of Gaussian Process Models," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 22, no. 11, pp. 2437-2450.
2. Meng Li, Howard Leung, **Zhiguang Liu** and Liuyang Zhou, "3D Human Motion Retrieval Using Graph Kernels based on Adaptive Graph Construction," *Computers & Graphics*, vol. 53, pp. 104-112.

Conference Papers

1. Liuyang Zhou, **Zhiguang Liu**, Howard Leung and Hubert P. H. Shum, "Posture Reconstruction Using Kinect with a Probabilistic Model," In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST'14. New York, NY, USA: ACM, 2014, pp. 117-125.
2. **Zhiguang Liu**, Howard Leung, Liuyang Zhou and Hubert P. H. Shum, "High Quality Compatible Triangulations for 2D Shape Morphing," In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*. ser. VRST'15. New York, NY, USA: ACM, 2015, pp. 85-94.

Bibliography

- [1] “Riverside Scene at Qingming Festival,” https://www.youtube.com/watch?v=Ke_UwaUvGsk.
- [2] 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*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 157–164.
- [3] B. Aronov, R. Seidel, and D. Souvaine, “On compatible triangulations of simple polygons,” *Computational Geometry*, vol. 3, no. 1, pp. 27–35, 1993.
- [4] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt, “A data-driven approach for real-time full body pose reconstruction from a depth camera,” in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV ’11, 2011, pp. 1092–1099.
- [5] S. W. Bailey and B. Bodenheimer, “A comparison of motion capture data recorded from a vicon system and a microsoft kinect sensor,” in *Proceedings of the ACM Symposium on Applied Perception*, ser. SAP ’12, 2012, pp. 121–121.

- [6] W. Baxter, P. Barla, and K.-i. Anjyo, “Rigid shape interpolation using normal equations,” in *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. ACM, 2008, pp. 59–64.
- [7] W. Baxter, P. Barla, and K.-i. Anjyo, “Compatible embedding for 2d shape animation,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 15, no. 5, pp. 867–879, 2009.
- [8] T. Beier and S. Neely, “Feature-based image metamorphosis,” in *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2. ACM, 1992, pp. 35–42.
- [9] L. Bo and C. Sminchisescu, “Structured output-associative regression,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2403–2410.
- [10] L. Bo and C. Sminchisescu, “Twin gaussian processes for structured prediction,” *International Journal of Computer Vision*, vol. 87, no. 1-2, pp. 28–52, 2010.
- [11] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas, “Fast algorithms for large scale conditional 3d prediction,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.
- [12] J. Chai and J. K. Hodgins, “Performance animation from low-dimensional control signals,” in *ACM SIGGRAPH 2005 Papers*, ser. SIGGRAPH ’05, 2005, pp. 686–696.
- [13] J. Chan, H. Leung, J. Tang, and T. Komura, “A virtual reality dance training system using motion capture technology,” *Learning Technologies, IEEE Transactions on*, vol. 4, no. 2, pp. 187–195, April 2011.

- [14] R. Chen, O. Weber, D. Keren, and M. Ben-Chen, “Planar shape interpolation with bounded distortion,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 108, 2013.
- [15] C.-C. Chiang, D.-L. Way, J.-W. Shieh, and L.-S. Shen, “A new image morphing technique for smooth vista transitions in panoramic image-based virtual environment,” in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '98. New York, NY, USA: ACM, 1998, pp. 81–90. [Online]. Available: <http://doi.acm.org/10.1145/293701.293712>
- [16] J. Choi and A. Szymczak, “On coherent rotation angles for as-rigid-as-possible shape interpolation.” in *CCCG*, 2003, pp. 111–114.
- [17] N. Dym, A. Shtengel, and Y. Lipman, “Homotopic morphing of planar curves,” in *Computer Graphics Forum*, vol. 34, no. 5. Wiley Online Library, 2015, pp. 239–251.
- [18] H. Fang and J. C. Hart, “Detail preserving shape deformation in image editing,” in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 12.
- [19] M. S. Floater, “Parametrization and smooth approximation of surface triangulations,” *Computer aided geometric design*, vol. 14, no. 3, pp. 231–250, 1997.
- [20] M. S. Floater, “Mean value coordinates,” *Computer aided geometric design*, vol. 20, no. 1, pp. 19–27, 2003.
- [21] S. Fortune, “A sweepline algorithm for voronoi diagrams,” *Algorithmica*, vol. 2, no. 1-4, pp. 153–174, 1987.

- [22] H. Fu, C.-L. Tai, and O. K.-C. Au, “Morphing with laplacian coordinates and spatial-temporal texture,” in *Proceedings of Pacific Graphics*, 2005, pp. 100–102.
- [23] C. Gotsman and V. Surazhsky, “Guaranteed intersection-free polygon morphing,” *Computers & Graphics*, vol. 25, no. 1, pp. 67–75, 2001.
- [24] G. Guennebaud, B. Jacob *et al.*, “Eigen v3,” <http://eigen.tuxfamily.org>, 2015.
- [25] H. Gupta and R. Wenger, “Constructing piecewise linear homeomorphisms of simple polygons,” *Journal of Algorithms*, vol. 22, no. 1, pp. 142–157, 1997.
- [26] J. Han, L. Shao, D. Xu, and J. Shotton, “Enhanced computer vision with microsoft kinect sensor: A review,” *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1318–1334, Oct 2013.
- [27] T. Helten, M. Muller, H.-P. Seidel, and C. Theobalt, “Real-time body tracking with one depth camera and inertial sensors,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 1105–1112.
- [28] L. Hoyet, R. McDonnell, and C. O’Sullivan, “Push it real: Perceiving causality in virtual interactions,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 90:1–90:9, Jul. 2012.
- [29] T. Igarashi, T. Moscovich, and J. F. Hughes, “As-rigid-as-possible shape manipulation,” *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005.
- [30] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, “Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera,” in

- Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11, 2011, pp. 559–568.
- [31] X. Jiang, H. Bunke, K. Abegglen, and A. Kandel, “Curve morphing by weighted mean of strings,” in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4. IEEE, 2002, pp. 192–195.
- [32] B. Joe and R. B. Simpson, “Corrections to lee’s visibility polygon algorithm,” *BIT Numerical Mathematics*, vol. 27, no. 4, pp. 458–473, 1987.
- [33] Z. Karni, D. Freedman, and C. Gotsman, “Energy-based image deformation,” in *Computer Graphics Forum*, vol. 28, no. 5. Wiley Online Library, 2009, pp. 1257–1268.
- [34] J. Kim, Y. Seol, and J. Lee, “Human motion reconstruction from sparse 3d motion sensors using kernel cca-based regression,” *Computer Animation and Virtual Worlds*, vol. 24, no. 6, pp. 565–576, 2013.
- [35] E. Kranakis and J. Urrutia, “Isomorphic triangulations with small number of steiner points,” *International Journal of Computational Geometry & Applications*, vol. 9, no. 02, pp. 171–180, 1999.
- [36] N. Lawrence, “Gaussian processes tool-kit, <http://cran.r-project.org/web/packages/gptk/>,” 2014.
- [37] N. Lawrence, M. Seeger, and R. Herbrich, “Fast sparse gaussian process methods: The informative vector machine,” in *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, 2003, pp. 609–616.

- [38] S.-Y. Lee, K.-Y. Chwa, and S. Y. Shin, “Image metamorphosis using snakes and free-form deformations,” in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 439–448.
- [39] J. P. Lewis, M. Cordner, and N. Fong, “Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 165–172.
- [40] X.-Y. Li, T. Ju, and S.-M. Hu, “Cubic mean value coordinates,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. 126:1–126:10, Jul. 2013. [Online]. Available: <http://doi.acm.org/10.1145/2461912.2461917>
- [41] Y. Lipman, V. G. Kim, and T. A. Funkhouser, “Simple formulas for quasiconformal plane deformations,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 5, p. 124, 2012.
- [42] H. Liu, X. Wei, J. Chai, I. Ha, and T. Rhee, “Realtime human motion control with a small number of inertial sensors,” in *Symposium on Interactive 3D Graphics and Games*, ser. I3D ’11, 2011, pp. 133–140.
- [43] Z. Liu, H. Leung, L. Zhou, and H. P. H. Shum, “High quality compatible triangulations for 2d shape morphing,” in *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, ser. VRST ’15, 2015, pp. 85–94.
- [44] R. MacCracken and K. I. Joy, “Free-form deformations with lattices of arbitrary topology,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 181–188.

- [45] Microsoft Corporation, Kinect for Xbox 360, 2013. [Online]. Available: <http://www.xbox.com/en-US/xbox-360/accessories/kinect>
- [46] T. Morgan, D. Jarrell, and J. Vance, “Poster: Rapid development of natural user interaction using kinect sensors and vrpn,” in *3D User Interfaces (3DUI), 2014 IEEE Symposium on*, March 2014, pp. 163–164.
- [47] Motion Analysis Corporation, 2015. [Online]. Available: <http://www.motionanalysis.com>
- [48] K. Murota, “Lu-decomposition of a matrix with entries of different kinds,” *Linear Algebra and its Applications*, vol. 49, pp. 275–283, 1983.
- [49] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, “Local gaussian process regression for real time online model learning,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1193–1200.
- [50] J. Quiñonero-Candela and C. E. Rasmussen, “A unifying view of sparse approximate gaussian process regression,” *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.
- [51] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh, “Pose machines: Articulated pose estimation via inference machines,” in *Computer Vision—ECCV 2014*. Springer, 2014, pp. 33–47.
- [52] C. E. Rasmussen and Z. Ghahramani, “Infinite mixtures of gaussian process experts,” *Advances in neural information processing systems*, vol. 2, pp. 881–888, 2002.

- [53] J. Sarrate, J. Palau, and A. Huerta, “Numerical representation of the quality measures of triangles and triangular meshes,” *Communications in numerical methods in engineering*, vol. 19, no. 7, pp. 551–561, 2003.
- [54] S. Schaal and C. G. Atkeson, “From isolation to cooperation: An alternative view of a system of experts,” *Advances in neural information processing systems*, pp. 605–611, 1996.
- [55] S. Schaefer, T. McPhail, and J. Warren, “Image deformation using moving least squares,” in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 533–540.
- [56] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, “2-d shape blending: an intrinsic solution to the vertex path problem,” in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, 1993, pp. 15–18.
- [57] M. Seeger, “Low rank updates for the cholesky decomposition,” University of California at Berkeley, Berkeley, CA, USA, Tech. Rep. EPFL-REPORT-161468, 2007.
- [58] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast pose estimation with parameter-sensitive hashing,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 750–757.
- [59] W. Shen, K. Deng, X. Bai, T. Leyvand, B. Guo, and Z. Tu, “Exemplar-based human action pose correction and tagging,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1784–1791.

- [60] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR ’11, 2011, pp. 1297–1304.
- [61] H. P. H. Shum and E. S. L. Ho, “Real-time physical modelling of character movements with microsoft kinect,” in *Proceedings of the 18th ACM symposium on Virtual reality software and technology*, ser. VRST ’12, 2012, pp. 17–24.
- [62] H. P. H. Shum, E. S. L. Ho, Y. Jiang, and S. Takagi, “Real-time posture reconstruction for microsoft kinect,” *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1357–1369, 2013.
- [63] H. Sidenbladh and M. Black, “Learning image statistics for bayesian tracking,” in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, 2001, pp. 709–716 vol.2.
- [64] M. Sigalas, M. Pateraki, I. Oikonomidis, and P. Trahanias, “Robust model-based 3d torso pose estimation in rgb-d sequences,” in *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, Dec 2013, pp. 315–322.
- [65] E. Snelson, “Local and global sparse gaussian process approximations,” in *Proceedings of Artificial Intelligence and Statistics, 2007*, pp. 524–531.
- [66] E. Snelson and Z. Ghahramani, “Sparse gaussian processes using pseudo-inputs,” in *Advances in Neural Information Processing Systems*. MIT press, 2006, pp. 1257–1264.
- [67] O. Sorkine and M. Alexa, “As-rigid-as-possible surface modeling,” in *Symposium on Geometry processing*, vol. 4, 2007.

- [68] A. Srivastava, E. Klassen, S. H. Joshi, and I. H. Jermyn, “Shape analysis of elastic curves in euclidean spaces,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 7, pp. 1415–1428, 2011.
- [69] R. W. Sumner and J. Popović, “Deformation transfer for triangle meshes,” *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 399–405, 2004.
- [70] V. Surazhsky and C. Gotsman, “Explicit surface remeshing,” in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. Eurographics Association, 2003, pp. 20–30.
- [71] V. Surazhsky and C. Gotsman, “High quality compatible triangulations,” *Engineering with Computers*, vol. 20, no. 2, pp. 147–156, 2004.
- [72] S. Suri, “A linear time algorithm for minimum link paths inside a simple polygon,” *Computer Vision, Graphics, and Image Processing*, vol. 35, no. 1, pp. 99–110, 1986.
- [73] I. Tashev, “Kinect development kit: A toolkit for gesture- and speech-based human-machine interaction [best of the web],” *Signal Processing Magazine, IEEE*, vol. 30, no. 5, pp. 129–131, Sept 2013.
- [74] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, “Real-time continuous pose recovery of human hands using convolutional networks,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 5, p. 169, 2014.
- [75] A. Toshev and C. Szegedy, “DeepPose: Human pose estimation via deep neural networks,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1653–1660.

- [76] V. Tresp, “Mixtures of gaussian processes,” *Advances in neural information processing systems*, pp. 654–660, 2000.
- [77] R. Urtasun and T. Darrell, “Sparse probabilistic regression for activity-independent human pose inference,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.
- [78] H. A. Van der Vorst, “Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems,” *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, 1992.
- [79] X. Wei, P. Zhang, and J. Chai, “Accurate realtime full-body motion capture using a single depth camera,” *ACM Trans. Graph.*, vol. 31, no. 6, pp. 188:1–188:12, Nov. 2012.
- [80] G. Wolberg, “Image morphing: a survey,” *The visual computer*, vol. 14, no. 8, pp. 360–372, 1998.
- [81] D. Xu, H. Zhang, Q. Wang, and H. Bao, “Poisson shape interpolation,” *Graphical Models*, vol. 68, no. 3, pp. 268–281, 2006.
- [82] W. Yang, X. Wang, and G. Wang, “Part-to-part morphing for planar curves,” *The Visual Computer*, vol. 30, no. 6-8, pp. 919–928, 2014.
- [83] H. Yasin, B. Krüger, and A. Weber, “Model based full body human motion reconstruction from video data,” in *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications*, ser. MIRAGE ’13, 2013, pp. 1:1–1:8.

- [84] X. Zhao, Y. Fu, and Y. Liu, “Human motion tracking by temporal-spatial local gaussian process experts,” *Image Processing, IEEE Transactions on*, vol. 20, no. 4, pp. 1141–1151, 2011.
- [85] L. Zhou, Z. Liu, H. Leung, and H. P. H. Shum, “Posture reconstruction using kinect with a probabilistic model,” in *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, ser. VRST ’14. New York, NY, USA: ACM, 2014, pp. 117–125.
- [86] L. Zhou, Z. Liu, H. Leung, and H. P. Shum, “Posture reconstruction using kinect with a probabilistic model,” in *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2014, pp. 117–125.