

國立交通大學
資訊科學與工程研究所
博士論文

以統計方法為基礎之二維角色動畫合成

Statistical Approaches for 2D Character Animation



研究生：周芸鋒

指導教授：施仁忠 教授

中華民國九十九年四月

以統計方法為基礎之二維角色動畫合成
Statistical Approaches for 2D Character Animation

研究生：周芸鋒 Student：Yun-Feng Chou

指導教授：施仁忠 教授 Advisor：Prof. Zen-Chung Shih

國立交通大學
資訊學院
資訊科學與工程研究所
博士論文



A Dissertation
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in

Computer Science

April 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年四月

以統計方法為基礎之二維角色動畫合成

研究生：周芸鋒

指導教授：施仁忠 教授

國立交通大學資訊科學與工程研究所博士班

摘 要

傳統二維動畫製作是屬於一個勞動密集型態的製作過程，也就是以人工手繪的方式逐格繪製該動畫中每一格人物角色的姿勢，且以固定的畫面更新率，產生該人物角色的動作或行為，而製作過程中，耗費大量的人力與物力在繪製每格人物角色的姿勢，與為其所繪製之姿勢進行上色工作。為了節省上述傳統二維動畫製作所耗費的人力與成本，本論文提出一個新的動畫合成方法，取代傳統人工手繪的方式，我們的方法以統計分析與推論為基礎，以較為有限的人工介入，來合成逼真的二維角色動畫。我們透過統計學中的無母數迴歸分析，有效率地描述靜態影像中，預先採樣的角色位移資訊，藉此合成該靜態影像中角色的二維動畫。此外，二維角色動畫可以被視為一個三維的空間與時間轉換問題，我們根據數張連續的靜態影像中的同一人物，研究其在不同時間點個別姿勢之相對關係，我們採用時間序列的概念，來分析與預測該角色一連串適宜的連續動作。

在本論文中，我們把二維角色動畫製作分成不同的多媒體應用，包括新視角的合成、臉部表情與說話嘴形的模擬、肢體動作合成。如上述所示，我們透過無母數迴歸，產生出由另一個視點觀看影像中人物角色的效果，且進一步模擬該角色與輸入語音同步的說話嘴形和臉部表情。針對影像中該角色的輪廓資訊，本論文將介紹一種特殊的資料參數表示式：橢圓徑向基底函數，主要用於描述於橢圓表面採樣之資訊。我們利用無母數迴歸當中的橢圓徑向基底函數核迴歸去描述並

預測該人物角色形狀的改變，藉此產生角色動畫，而且，為了在角色變形之後，仍維持原有的角色細節或特徵，無母數迴歸中的局部加權迴歸則被用來加強區域細節的控制，藉此保有該角色的原有特徵。此外，我們進行時間序列分析，從數張連續影像中，針對影像中同一人物角色在不同時間點的姿勢，來分析該角色的肢體移動軌跡，我們提出一個無母數貝氏方法來估計代表該移動軌跡的時間序列，並依照所估計的時間序列，模擬該角色的行為或動作。本論文最終將更深入探討如何透過所提之統計方法來合成被動元件的動畫，也就是合成由自然界外力所造成的被動元件移動，如合成出因風吹拂，造成樹木搖曳與水起漣漪的效果。

本論文提出一個從靜態影像中，有效率地合成出二維角色動畫的方法。實驗成果充分驗證本論文所提方法之可行性與可塑性，不但能夠有效模擬出逼真的角色動作，所估計的移動軌跡能因應所提供角色不同時間點的姿勢而變化，產生出的動畫亦減少不自然的扭曲現象，另一方面，本論文所提方法特別適合於智能化的多媒體應用，可用於如虛擬人物的合成，我們也相信此方法能加速整個動畫製作的過程。



Statistical Approaches for 2D Character Animation

Student: Yun-Feng Chou

Advisor: Prof. Zen-Chung Shih

Institute of Computer Science and Engineering

National Chiao-Tung University



ABSTRACT

Traditionally, the production of 2D animation is a labor-intensive artisan process of building up sequences of drawn images by hand which, when shown one after the other one at a fixed rate, resemble a movement. Most work and hence time is spent on drawing, inking, and coloring the individual animated characters for each of the frames. Instead of the traditional animation generated by hand, we introduce a novel method by enhancing still pictures and making characters move in convincing ways. The proposed method is based on the statistical analysis and inference, while minimizing users' intervention. We adopt *nonparametric regression* to efficiently analyze the displacements of the pre-sampled data from characters in still pictures and use it to generate 2D character animation directly. Furthermore, 2D character animation is regarded as 3D transformation problem, which consists of a 2D spatial displacement and a 1D shift in time. Hence, we focus on the temporal relationship of different poses of the same character in these still pictures. *Time series* is applied to analyze the character's movement and forecast a sequence of the suitable limbs movement of the character.

In this dissertation, 2D character animation involves novel view generation, expressive talking face simulation, and limbs movement synthesis. Considering

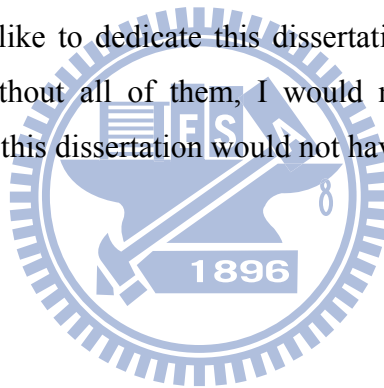
characters in still pictures, we focus on *nonparametric regression* to generate a novel view and an expressive facial animation synchronized with the input speech of a character. *Kernel regression* with *elliptic radial basis functions* (ERBFs) is proposed to describe and deform the shape of the character in image space. Note that the novel parametric representation, ERBFs, can be applied to represent the observations of the shape on the unit ellipse. For preserving patterns within the deformed shape, *locally weighted regression* (LOESS) is applied to fit the details with local control. Furthermore, *time series* is used to analyze the limb movement of a character and represent the motion trajectory. Note that a character's motion could be described by a series of non-continuous poses of a character from a sequence of contiguous frames. According to these poses, we investigate a nonparametric Bayesian approach to construct the time series model representing the character's motion trajectory. Then we can synthesize a sequence of the motion by using the motion trajectory. Last but not the least, we also investigate how to adopt the proposed statistical approaches mentioned above to animate passive elements. The movements of passive elements involving natural movements that respond to natural forces in some fashion like trees swaying and water rippling could be synthesized. Given a picture of a tree, we make it sway. Given a picture of a pond, we make it ripple.

The solutions are developed to animate photographs or paintings effectively. Experimental results show that our method effectively simulates plausible movements for 2D character animation. They also show that the estimated motion trajectory best matches the given still frames. In comparison to previous approaches, our proposed method synthesizes smooth animations, while minimizing unnatural distortion and having the advantages of being more controllable. Moreover, the proposed method is especially suitable for intelligent multimedia applications in virtual human generation. We believe that the provided solutions are easy to use, and empower a much quicker animation production.

Acknowledgements

First and foremost I would like to show my gratitude to my advisor, Prof. Zen-Chung Shih, for his support and guidance during the hard time of my stay at the Department of Computer Science, National Chiao Tung University. I would also like to thank the members of my oral defense committee, Prof. Shyan-Ming Yuan, Prof. Wen-Chieh Lin, Prof. Hao-Ren Ke, Prof. Tong-Yee Lee, Prof. Ming Ouhyoung, Prof. Shi-Nine Yang, and Prof. I-Chen Lin, for their insightful suggestions and for many enlightening discussions. Moreover, I am grateful to all the members in Computer Graphics and Virtual Reality Laboratory for their useful encouragement in these days.

Finally, I would like to dedicate this dissertation to my parents for their selfless love and support. Without all of them, I would never receive the Ph.D. degree in computer science, and this dissertation would not have been written.

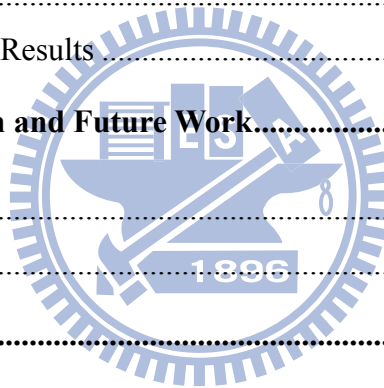


Contents

摘 要.....	i
ABSTRACT.....	iii
Acknowledgements.....	v
Contents	vi
List of Tables.....	ix
List of Figures.....	x
List of Symbols	xiii
Chapter 1 Introduction.....	1
1.1 Overview of Traditional 2D Animation Production.....	1
1.2 Motivation.....	4
1.3 Methodology.....	4
1.4 Primary Contributions.....	9
1.5 Auxiliary Multimedia Application.....	10
1.6 Dissertation Organization.....	11
Chapter 2 Literature Review	12
2.1 Image Morphing.....	12
2.2 Shape Deformation.....	14
2.3 Image Interpolation.....	15
2.4 View Interpolation.....	15
2.5 Expression and Viseme Synthesis.....	16
2.6 Motion Capture	16
2.7 <i>Time Series</i>	17
Chapter 3 Statistical Approaches.....	19

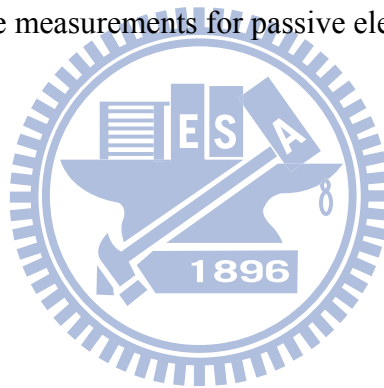
3.1 <i>Kernel Regression with Elliptic Radial Basis Functions</i>	19
3.2 <i>Locally Weighted Regression</i>	23
3.3 <i>Reversible Jump Markov Chain Monte Carlo (RJMCMC) Sampler</i>	26
3.4 Time Series Analysis	28
Chapter 4 Two-scale Image Abstraction	29
4.1 Color Space Transformation	29
4.2 Bilateral Filter	31
4.3 Image Abstraction	32
4.4 Experimental Results	33
Chapter 5 Novel View Generation	36
5.1 View Interpolation.....	36
5.2 Algorithm Overview	38
5.3 Character Extraction	39
5.4 Shape Deformation Using <i>Kernel Regression</i> with ERBFs.....	41
5.4.1 The Determination of Initial Values	41
5.4.2 Shape Deforming.....	43
5.5 Detail Preservation Using LOESS	45
5.6 Experimental Results	47
Chapter 6 Expressive Face with Speech Animation.....	54
6.1 Character and Features Extraction	54
6.2 Speech Animation	55
6.2.1 Algorithm Overview	56
6.2.2 Viseme Synthesis	56
6.3 Viseme Synthesis with Expressive Face	58
6.4 Experimental Results	61
Chapter 7 Limbs Movement Synthesis	68
7.1 Statistic-based Movement Synthesis	68
7.2 Algorithm Overview	70
7.3 Bayesian-based Limbs Movement Synthesis.....	72
7.3.1 Shape Structure.....	72

7.3.2 Point-to-point Correspondences	73
7.3.3 <i>Bayesian Inference</i>	74
7.3.4 The Time Series Model.....	76
7.3.5 Detail Preservation	77
7.4 Summary	78
7.5 Experimental Results	79
Chapter 8 Animating Passive Elements	85
8.1 <i>Simple Harmonic Motion</i>	85
8.2 Algorithm Overview	87
8.3 Passive Element Animation.....	89
8.3.1 Extraction and Specification.....	89
8.3.2 Water Waves	90
8.3.3 Trees	94
8.4 Experimental Results	95
Chapter 9 Conclusion and Future Work.....	100
9.1 Conclusion	100
9.2 Future Work.....	102
Bibliography	105
Vita.....	111



List of Tables

Table 5.1: Running times of figures for novel view generation and body movement synthesis.....	48
Table 6.1: Conversion table from phoneme to mouth shape and the corresponding phonetic alphabet.....	57
Table 6.2: Running times of figures for viseme and facial expression synthesis.....	62
Table 7.1: Performance measurements of limbs movement synthesis.....	79
Table 8.1: Performance measurements for passive element animation.....	98



List of Figures

Figure 1.1: The traditional 2D animation production.	2
Figure 1.2: Comparison of the number of basis functions using Gaussians	6
Figure 1.3: The overview of 2D character animation with the proposed statistical approaches	7
Figure 3.1: Comparison of ERBFs.....	21
Figure 3.2: Schematic diagram of an arbitrary directional <i>elliptic radial basis function</i> (ERBF)	21
Figure 4.1: Image abstraction computed using our two-scale decomposition.....	34
Figure 4.2: Another example of image abstraction.....	35
Figure 5.1: Novel view generation in a comic.....	37
Figure 5.2: This example shows the picture of Mona Lisa.....	38
Figure 5.3: Correspondences and initial value determination.....	42
Figure 5.4: Unnatural distortion without detail preserving.....	45
Figure 5.5: Novel view generation in a comic.....	49
Figure 5.6: Novel view generation from a painting.....	49
Figure 5.7: Body movement synthesis.....	50
Figure 5.8: Visual comparison of novel view generation.....	51
Figure 5.9: Visual comparison of body movement synthesis and detail preservation....	53
Figure 6.1: The overview of speech animation generation.....	55

Figure 6.2: Viseme segmentation of the given speech data	57
Figure 6.3: The overview of expressive face generation with the picture of Mona Lisa	59
Figure 6.4: Groupings of facial shape and features labeled as anchor points and relative curves.....	59
Figure 6.5: 2D character animation of Mona Lisa	63
Figure 6.6: Viseme synthesis for five vowels	64
Figure 6.7: Character animation with expression synthesis.....	65
Figure 6.8: Visual comparison of expression synthesis	66
Figure 6.9: Another comparison of expression synthesis	67
Figure 7.1: The overview of our approach for synthesizing limbs movements.....	70
Figure 7.2: Point-to-point correspondences	74
Figure 7.3: LOESS analysis	77
Figure 7.4: Two major diagrams of a motion trajectory	80
Figure 7.5: A motion trajectory of a ball	80
Figure 7.6: Limbs movement synthesis	81
Figure 7.7: Visual comparison of character animation	83
Figure 8.1: The overview of passive element animation with the example of the Japanese Temple named Temple of the Golden Pavilion (from Japanese term Kinkaku-ji)	88
Figure 8.2: The example of the Temple of the Golden Pavilion for segmentation and sampling	90
Figure 8.3: The example of the wind blowing downward	91
Figure 8.4: Schematic diagram of <i>simple harmonic motion</i>	91

Figure 8.5: The diagram of an arbitrary sample (P_{ij}) and its 4-neighbors ($P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, and $P_{i,j+1}$)	93
Figure 8.6: The diagram of the tree with <i>simple harmonic motion</i>	95
Figure 8.7: The result of the Temple of the Golden Pavilion.....	96
Figure 8.8: The result for swaying plants.....	97
Figure 8.9: Another example for a swaying tree	98



List of Symbols

General

d, i, j, t	indices
n, k, M, m, n_0, N, p, q	scalars
T	transpose matrix
ε	error term
$N(\cdot)$	Normal distribution function
$\text{Gamma}(\cdot)$	Gamma distribution function
$U(\cdot)$	Uniform distribution function
τ^2	noise variance
u	random variables (or pre-sampled data)
J	pre-sampled data
$\overline{J_1 J_2}$	line segment of two pre-sampled data
$\vec{u} = (x, y)$	a vector of the pre-sampled data with 2D coordinates
J', \vec{u}'	correspondences of the pre-sampled data
U	input space of points and the correspondences
\mathfrak{R}	set of real numbers
\mathfrak{R}^+	positive values of real numbers
\mathfrak{R}^m	m -dimensional Euclidean space
$\hat{\psi}$	a estimated vector ψ
$\bar{\psi}$	mean of a vector ψ
(μ_x, μ_y)	a vector of mean for two random variables x and y
(V_x, V_y)	a vector of variance for two random variables x and y
σ_{xy}^2	covariance for two random variables x and y
<i>Precision</i>	precision of the coefficient prior.
X	basis or polynomial terms of predictors in the matrix form
Y	responses of the regression model in the matrix form
P	matrix form of prior precision p
$L(\cdot)$	log-likelihood

CIELab Color Space

L^*	lightness
a^*	color opponent between red/magenta and green
b^*	color opponent between yellow and blue
$X_{CIE}, Y_{CIE}, Z_{CIE}$	the derived parameters from red, green, and blue colors in CIEXYZ
X_n, Y_n, Z_n	the normalized term with the reference white point in CIEXYZ
th	a scalar for threshold
P_{temp}	temporary variable

Bilateral Filter

I	an original image
Ω	the whole image range
I_p, I_q	intensity values of pixel p and q
G_{σ_s}	Gaussian function in the spatial domain
G_{σ_t}	Gaussian function in the intensity domain
σ_s	standard deviation in the spatial domain
σ_t	standard deviation in the intensity domain
$k(\cdot)$	a normalized function
$BF(I)_p$	the filtered result of the pixel p
B_{Li}	the i -level base layer
D_{Li}	the i -level detail layer

Arbitrary Directional *Elliptic Radial Basis Function* (ERBF)

$\vec{v} = (\mu_x, \mu_y)$	a center vector of an elliptic Gaussian with 2D coordinates
$\eta(\cdot)$	kernel function of ERBF
σ_i^2	covariance of Gaussian along i -axis
$\sigma_{j,i}^2$	covariance of the j -th Gaussian along i -axis
θ_i	angle between the major axis of ellipse and i -axis
a_i^2	aspect ratio of an ellipse in arbitrary directional ERBF along i -axis
A_{θ_i, a_i}	transformation matrix based on an angle and the aspect ratio
$\chi \subseteq \ell_2^N$	the space of square summable sequences of length N
σ_N^2	covariance of the N -dimensional Gaussians
Σ	diagonal matrix

Kernel Regression with ERBFs

\vec{r}	response representing the displacement of the pre-sampled datum
\vec{u}	predictor representing the position of the pre-sampled datum
$f(\cdot)$	regression surface
β_j	coefficient of the j -th elliptic Gaussian $\eta(\cdot)$
$T(\cdot)$	affine component
$R(\cdot)$	radial component
M_r	2×2 real matrix
B	matrix form of the coefficients of Gaussians
K	matrix form of the kernel functions
$\overline{f(\cdot)}$	matrix form of the function $f(\cdot)$

Locally Weighted Regression

x_0	center of a kernel
$x_i = (x_{i,x}, x_{i,y})$	dependent variable representing the i -th sampled point with 2D coordinates
$y_i = (y_{i,x}, y_{i,y})$	measurement of the dependent variable representing the new location of the i -th sampled point with 2D coordinates
$w_i(\cdot)$	weight of the i -th sampled point
s	a smoothing parameter
W_{kernel}	kernel width or bandwidth
$t_j(\cdot)$	the j -th term of polynomial functions
$t(\cdot)$	matrix form of polynomial functions
ζ_j	coefficient of the j -th term of polynomial functions
ζ	matrix form of coefficients of polynomial functions

Expressive Talking Face

V	set of moving templates for viseme synthesis
D	set of moving templates for mood synthesis
FE	lip-synch expressive face
$AT(\cdot)$	2D affine transformation of the head movements
\mathbb{N}	neutral expression
F_l	movements of lips
F_{nl}	movements of facial features except lips
D_l	movements of lips in a specific emotional state
D_i	movements of an individual facial feature except lips in a specific emotional state
L	lips movement synchronized with the input speech
γ	blending weight

Bayesian Inference

Θ	parameter space
Θ_t	current parameter space while there are t samples
D_{train}	training data
\vec{r}_{new}	response of an arbitrary new predictor representing the displacement of the pre-sampled datum
\vec{u}_{new}	arbitrary new predictor representing the position of the pre-sampled datum
β	matrix form regression coefficients
W	the diagonal matrix form of weights in LOESS

Bayesian Version of Autoregressive Moving Average

D_t	univariate time series datum at time t
$f_{TS}(\cdot)$	time series function
v_t	random variable at time t
C	constant
α_t	a multiple of the random variable in time t
ϕ_i, κ_i	coefficients of i -th term of ARMA
$U_i = (\vec{u}_{i,1}, \dots, \vec{u}_{i,n})$	a vector of the positions of n pre-sampled data from the i -th key-motion
(x,y,t)	the space of a motion trajectory with 3D coordinates
$f_{MT}(\cdot)$	one-dimensional representation of a motion trajectory

Simple Harmonic Motion

K_{Si}	the stiffness of the spring
X_{Dis}	displacement of a mass
$X_{Dis}(t)$	displacement of a mass at time t
A_{amp}	amplitude
ω	angular frequency
ϕ	phase shift
$v(t)$	velocity of a mass at time t
$a(t)$	acceleration of a mass at time t
f_{Fre}	frequency of the oscillation
T_{Per}	period of the oscillation

Water Rippling

P_S	point of impact
P_{TI}	sample of a water particle
x	position of the grid point in image space
y	water particle's displacement in vertical direction
v	propagation velocity
λ	wavelength
A	amplitude of the wave
P_{ij}	the sampled water particle
Δx_{Off}	distortion along the horizontal direction in image space
Δy_{Off}	distortion along the vertical direction in image space
$D_{Dis}(\cdot)$	displacement of its parameter

Tree Swaying

θ	the angle of sway
----------	-------------------

Chapter 1

Introduction

2D characters in motion have intrigued animators and computer graphics researchers for several decades. Most animations of 2D characters have been created using the traditional and often highly labor intensive keyframing technique, in which computers are employed to interpolate between animator-specified keyframes. It costs a lot of people and money to produce a sequence of animation. More recently, increasingly automated techniques for synthesizing realistic characters' motion have drawn much attention. In this dissertation, we will investigate the problem of producing animation which captures the smoothness of motion and behavior. These animations are intrinsically complex and present a challenge to the computer graphics practitioner. Animations of this sort are of interest not only because they attempt to recreate fascinating natural scenarios, but also because they have broad applicability. They can be used in the entertainment industry and advanced intelligent multimedia applications for next generation environments utilization, such as special effects in movies or in video games, real-time live performance [41], and enhancing graphical interface [63].

1.1 Overview of Traditional 2D Animation Production

In order to describe which parts of works during animation production most time or money is spent on, we give a brief overview of the traditional animation process in general first, as illustrated in Figure 1.1.

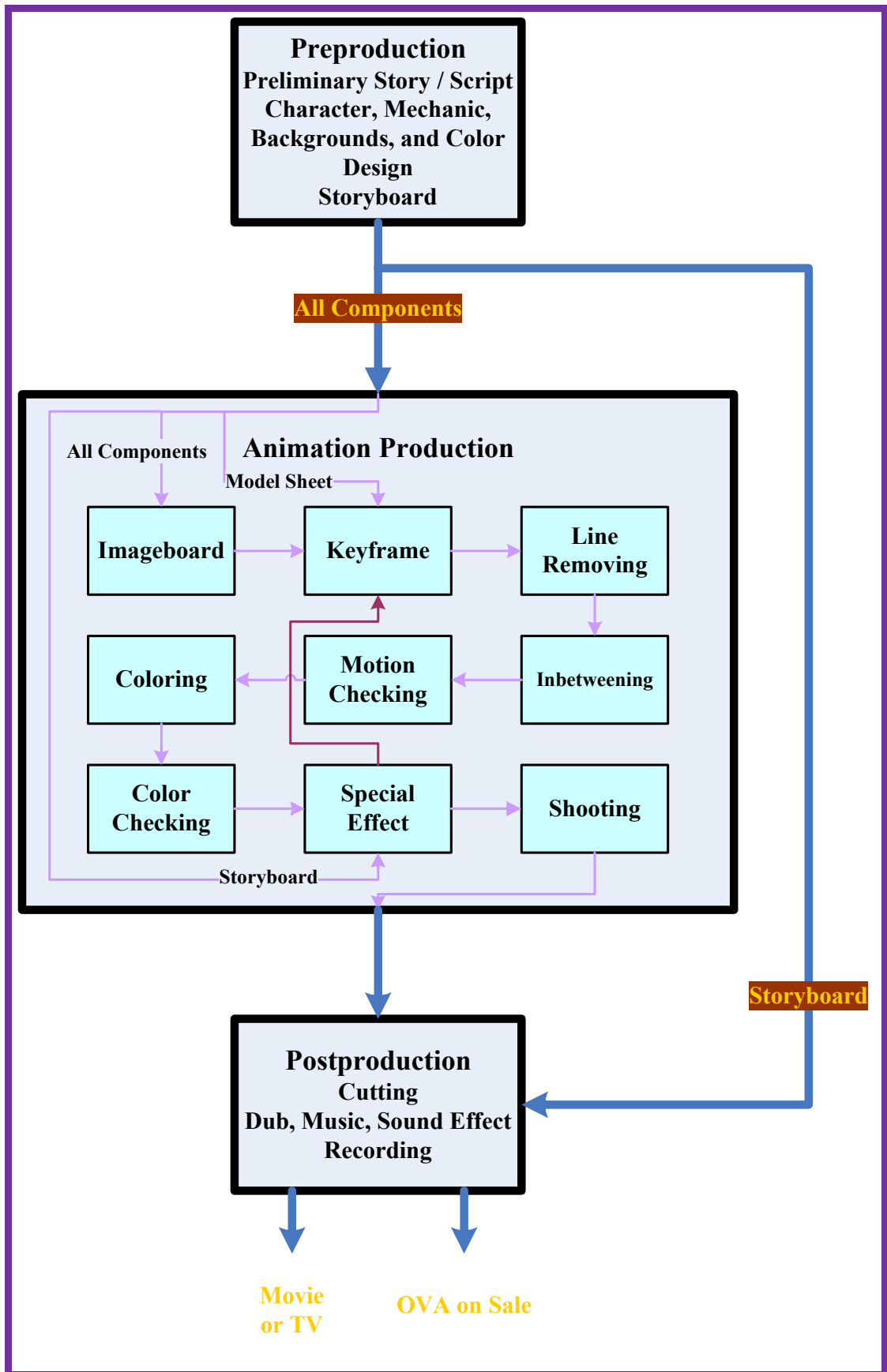


Figure 1.1: The traditional 2D animation production.

The traditional 2D animation production begins with preparing a story, including the related scenes, objects, and scripts designs. Then the story is refined into a storyboard. A storyboard is a visual layout of events allowing the animators to plan the flow of the plot and the composition of the imagery. Next to the storyboard, model sheets are also prepared. These sheets include the extreme poses of the characters to be drawn from which intermediate attributes can be interpolated, such as position, expression, color, size, etc.

According to model sheets and an imageboard which is a more detailed storyboard, key animators draw keyframes representing some scenes or several poses of a continuous action and behavior of the characters in the story. Moreover, keyframes show the major features of each character's action and behavior. After removing redundant lines in keyframes, assistant animators produce the in-betweens by considering those keyframes in order to yield a smooth animation. Unlike live action, where the camera is running continuously, each frame of an animation film is shot one by one. For each frame, a line drawing (not colored) is prepared to perform a line test for removing redundant lines or strokes. These drawings get mechanically aligned with the purpose of verifying that the movements are correct and that characters interact accurately. Then each frame is composed of several layers, which will be composited into a single frame in the compositing stage. Moreover, special effect artists can improve the constructed scene with some special effects, such as fog, smoke, or fire. These effects are created by compositing these layers. Furthermore, each frame is transferred or colored from paper onto a transparent sheet of celluloid by xerography. The transparent quality of celluloid allows for stacking up all sheets on each other. Thus, one sheet of celluloid can be seen underneath another sheet. The opaque background can be seen underneath all sheets of celluloid. Note that we propose a novel method by applying statistical concepts to animate 2D character from still keyframes or pictures instead of in-betweening step in traditional 2D animation production. In this dissertation, the coloring step could also be carried out by preserving details of the animated character while giving the colored keyframes or pictures.

Finally, an animation is produced by shooting frame by frame at a certain rate, such as 24 frames per second. A soundtrack is recorded, so that the animation may be more precisely synchronized to the soundtrack. In this last step, the soundtrack is

synchronized with the character’s action and behavior, and added to the video stream. Hence, a movie or an original video animation is produced.

1.2 Motivation

Traditionally, 2D animation production has been a labor-intensive artisan process of building up sequences of drawn images by hand. During the whole process, most work, and hence time, is spent on two tedious tasks: in-betweening (or drawing) and coloring (or inking) of each in-between frame, which take up approximately 60% of the total labor required in traditional animation production [79]. Instead of in-betweening and coloring, animating still keyframes or comics becomes a significant research direction to reduce the workload of animators and production costs. Computer graphics researchers have focused on making still pictures move in convincing ways and creating lively 2D animation, while minimizing users’ intervention. For example, when we view a photograph or painting, we perceive much more than the static picture before us. We supplement that image with our life experiences: given a picture of a tree, we imagine it swaying; given a picture of a pond, we imagine it rippling; given a picture of a human, we imagine him laughing, talking, or walking.

1.3 Methodology

This dissertation presents a method for animating still pictures, such as photographs and paintings. Generating a natural-looking animation from an image can be considered to analyze and simulate the motion of elements in that image. For example, Chuang et al. [16] animated passive elements, which are subject to natural forces like wind, by using stochastic motion textures to deform pictures. Besides, Hornung et al. [30] achieved the motion of photographed persons by projecting them to 3D motion data.

In this dissertation, 2D Character animation involves novel view generation, expressive talking face simulation, and limbs movement synthesis. We explore how a set of explicitly encoded pre-sampled data representing a character’s motion in still

pictures. Statistical approaches are applied to analyze that motion by using *nonparametric regression* which consists of *kernel regression* with *elliptic radial basis functions* (ERBFs) and *locally weighted regression* (LOESS). Note that *kernel regression* with ERBFs is used to fit the contours of a character and applied to infer the corresponding displacements to synthesize a novel view or an expressive talking face. Besides, LOESS is used to preserve important textures, patterns, or features within the synthesized outer contours of a novel view or an expressive talking face (that is filling in the color and texture information obtained from the original character in the given picture).

Our proposed approach is based on the prediction abilities of both *kernel regression* and LOESS [29, 42]. *Kernel regression* approximates the contours of the deformed character between two key-poses, which are two poses of a character in the given images, by the prior use of a set of kernel functions. Previously, researchers [70] presented image morphing techniques using *radial basis functions* (RBFs) with spatially-limited circular Gaussian distribution functions for the kernel. In contrast, circular Gaussian is not an appropriate choice to fit contours, which have noncircular structures, as shown in Figure 1.2. Figure 1.2 (a) is the original image, Figure 1.2 (b) using the circular Gaussians needs five kernels to fit the contour of the right arm of the character, and Figure 1.2 (c) using the elliptic Gaussians can fit the right arm and left leg with the same number of kernels. Using too many circular Gaussians increases the learning and fitting time. In this dissertation, we develop character deformation in image space using ERBFs specifically known as elliptic Gaussians, which provide less fitting time. Although ERBFs require more computation during optimization, better quality is obtained with fewer number of basis functions.

Except the globally smooth shape deformation with contours fitting mentioned above, the local-fitting methodology is also applied to preserve important features within the contour. For example, the wood grain of the character in Figure 1.2 (a). LOESS is used to preserve the features of details. LOESS is based on the minimized weighted sum of squared residuals. It is a way of estimating the regression surface through a multivariate smoothing procedure by fitting a function of independent variables locally.

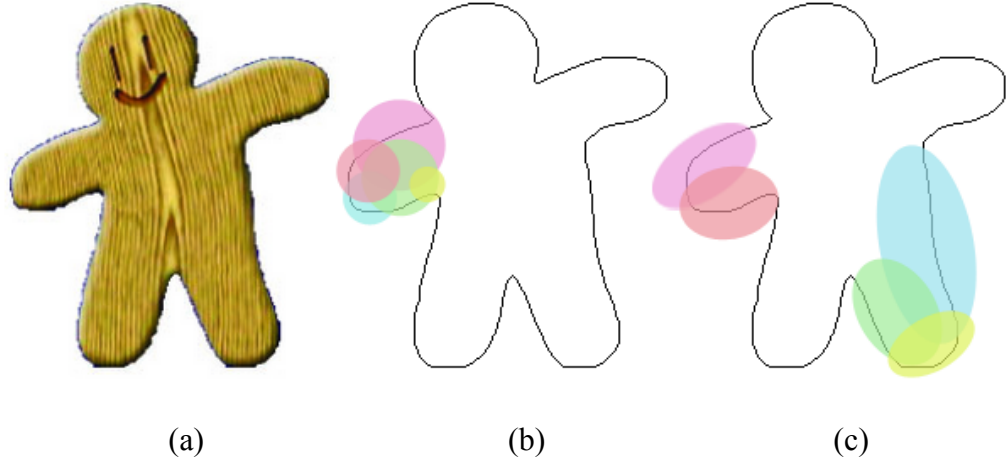


Figure 1.2: Comparison of the number of basis functions using Gaussians. (a) The original image. (b) Using RBFs to fit the contour of right arm with five kernels, and (c) using ERBFs to fit right arm and left leg with the same number of kernels.

In addition, a motion is essential by a 3D transformation problem which consists of a 2D spatial displacement and a 1D shift in time. *Time series* is integrated into the original model for forecasting the limbs movements of a character. We propose a Bayesian approach named by the Bayesian version of *autoregressive moving average* (BARMA) for time series analysis, which is based on *Bayesian inference* [78] by using the *reversible jump Markov chains Monte Carlo* (RJMCMC) method [14]. RJMCMC has advantages for parameters estimation. Note that RJMCMC generates a sequence or a chain of samples. Apart from the initial sample, each sample is derived from the previous sample, which allows the algorithm to find coefficients or parameters that satisfy the situation of current regression model. Moreover, ARMA is a useful time series model for human motion or stable time series data. Hence, BARMA is adopted to fit the motion trajectories of a character. BARMA, which integrates *Bayesian inference* with ARMA, is applied to predict the motion trajectories of the limbs. Then the trajectories are applied to synthesize the behaviors or limbs movements of the character.

In Figure 1.3, the outline reflects the structure of our proposed method for 2D character animation. Considering Figure 1.3, we briefly summarize our method in the following paragraphs.

1. Image Abstraction: This dissertation focuses on animating arbitrary still images for 2D character animation. Considering a real image, such as a photograph, it may have 16-bits or even 24-bits per color channel. Too many unimportant or unnecessary

contours of the character in that image are calculated to animate the character. It would waste the computation time and CPU power. Hence, we would simplify the color representation of the real image and acquire proper contours of the character in that image. The two-scale image abstraction is used to obtain the appropriate contours by eliminating redundancy information. The proposed two-scale image abstraction is based on the bilateral filter.

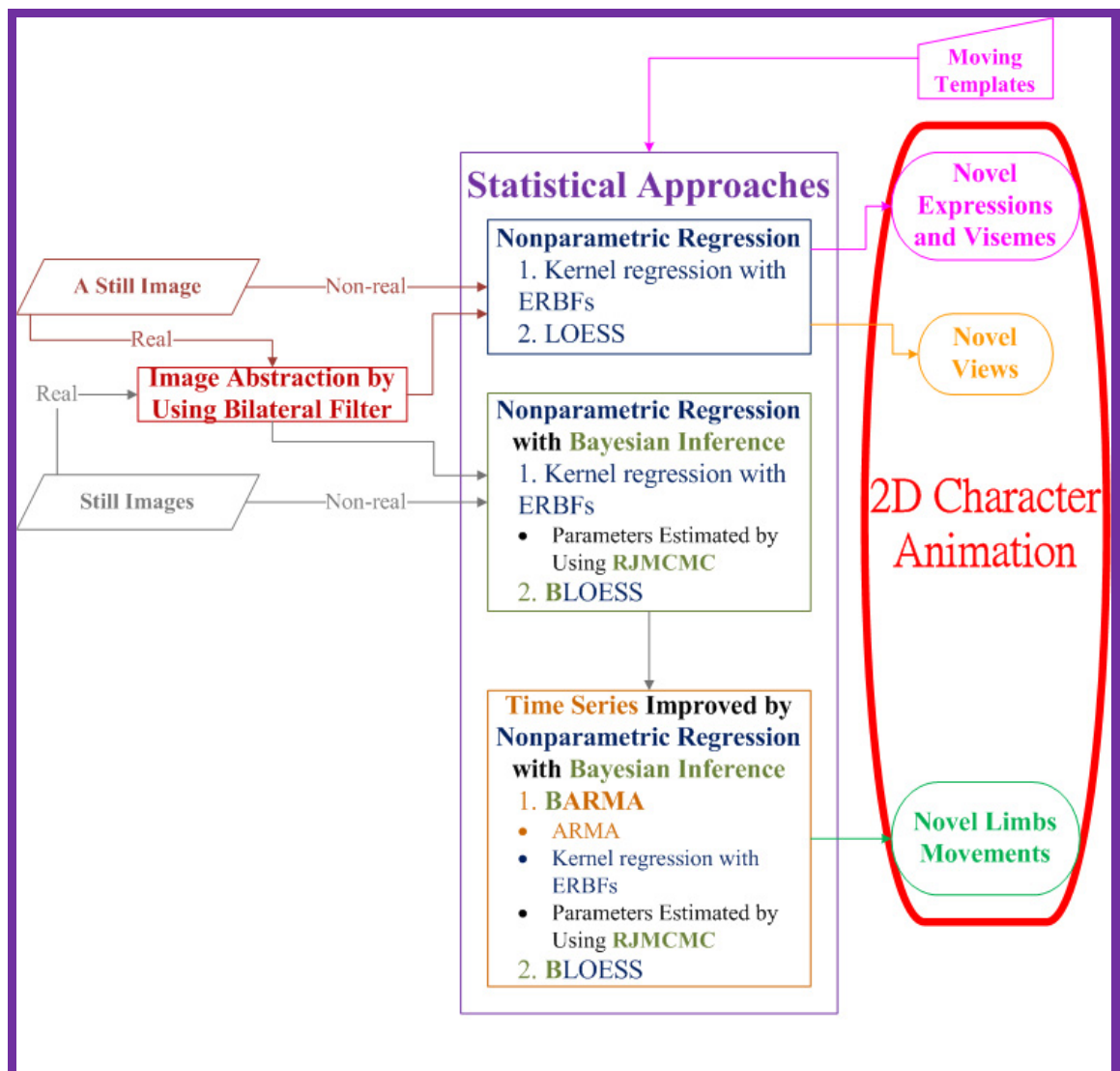


Figure 1.3: The overview of 2D character animation with the proposed statistical approaches.

2. Statistical Approaches: We adopt statistical approaches to animate 2D character. The proposed approaches include *nonparametric regression*, *Bayesian inference*, and *time series*. *Nonparametric regression* and *time series* are used to forecast a character's motion. *Bayesian inference* is employed to estimate the parameters or coefficients during the regression analysis and time series analysis. Hence, based on the prediction abilities of statistical analysis, we would synthesize a smooth and suitable character animation. The Bayesian approach improves meaningful convergences even with fewer data points than the equation parameters or coefficients.

3. Novel Views: In order to generate novel views, we adopt *nonparametric regression*, i.e. *kernel regression* with ERBFs and LOESS. For view morphing, shape deforming is carried out by using *kernel regression* with ERBFs, which is suited to the natural shape of characters like the character's head and body, as mentioned previously. Besides, features invariant is maintained during shape deformations by using LOESS while minimizing unnatural distortion.

4. Novel Expressions and Visemes: Given suitable moving templates, such as the training data set consisting of the mouth shape and the positions of other facial features for facial expression simulation and viseme synthesis, the proposed nonparametric regression model composed of *kernel regression* with ERBFs and LOESS would be trained and further applied to create lively animated talking faces and synthesize the stylistic variations of facial moods and expressions.

5. Novel Limbs Movements: Given two contiguous frames from a comic or a video, the contours of a character's key-motions are synthesized by using Bayesian estimation of *kernel regression*, which combines ERBF kernel with *Bayesian inference* through RJMCMC. As mentioned above, ERBF kernel is suitable to fit the natural shape of a character. Moreover, RJMCMC finds parameters that satisfy the situation of current moving model without leading to local minimization. Then BARMA is proposed to synthesize the contours of the whole character's motion by analyzing the motion trajectory from those key-motions. Note that a nonparametric Bayesian approach is constructed for improving ARMA and adding the smooth variety of the time series data by using ERBF kernel and RJMCMC. As similar as mentioned above, another estimation of LOESS called the Bayesian version of LOESS (BLOESS) is applied to preserve the details or features of the animated characters.

Note that the statistical approaches we proposed are adopted to fit the movement of the moving element including both characters and passive elements without limiting our domain. Motivated by the promising work on 2D character animation, this dissertation also provides principled techniques so that the proposed statistical approaches can be used in less restricted environments, model the movements of passive elements, and be applied to an exciting new application. As mentioned above, *nonparametric regression* is used to represent the displacements of passive elements from still pictures and infer a sequence of movements.

1.4 Primary Contributions

According to the provided information, this dissertation makes the following contributions for 2D character animation respectively.

Given a single pose of the character, making the character move:

- A novel approach for shape deforming based on *kernel regression* with ERBFs is proposed, which is suited to the natural shape of characters, such as a human's torso or an essentially human-like animal's limbs.
- By using a closed-form solution of LOESS, a new method for detail preserving is presented, which maintains features invariant during deformations while minimizing unnatural distortion.
- The proposed nonparametric regression model composed of *kernel regression* with ERBFs and LOESS would be applied to novel view generation. Besides, it is further used to create lively animated talking faces and synthesize the stylistic variations of facial moods and expressions for 2D character animation.

Given two contiguous poses of the character, making the character move and while matching these poses best synchronously:

- Given two contiguous poses of a character from a comic or a low-frame-rate video, the contours of a character's key-motions are synthesized by using a Bayesian

estimation of *kernel regression*, which combines ERBF kernel with RJMCMC. A key-motion is defined as the contour of an in-between pose between two given poses of a character. This approach can fit the shape of a character with parameters and coefficients adaptive to the current situation of the regression model.

- BARMA is proposed to analyze the motion trajectory of a character's limb through a nonparametric Bayesian approach. The Bayesian approach is constructed for adding the smooth variety of the time series data by using ERBF kernel and RJMCMC described above.
- BARMA is applied to synthesize the shape of the whole character's motion through contours fitting. Furthermore, BLOESS is applied to preserve the details or features of characters. The Bayesian approach improves meaningful regressions even with fewer data points than regression coefficients.

1.5 Auxiliary Multimedia Application

Another novel application of the introduced statistical approaches is proposed to fit the movement of the passive elements without limiting our domain in character animation, as described below.

- *Simple harmonic motion* is applied to estimate the displacements of the points sampled on the water wave or the tree in the next time-sliced scene first.
- *Kernel regression* with ERBFs is used to fit the contours of the water wave or the tree in the next time-sliced scene from the estimated positions of samples.
- Furthermore, LOESS is applied to preserve the details, features, or textures from the fitted contours interiors.

1.6 Dissertation Organization

The remainder of this dissertation is organized as follows. **Chapter 2** reviews the related literature on animating characters in still pictures. **Chapter 3** then summarizes statistical approaches that we employ to animate 2D characters. Considering arbitrary pictures, the two-scale image abstraction is used to eliminate redundancy information in **Chapter 4**. Next, **Chapter 5** describes how to apply *nonparametric regression* to generate a novel view of a character. **Chapter 6** deals with expressive talking face simulation. In addition, **Chapter 7** further infers limbs movements by integrating *Bayesian inference* and *time series* with the regression model. Moreover, **Chapter 8** demonstrates how to apply the proposed statistical approaches to animate passive elements for simulating natural phenomena. Finally, **Chapter 9** concludes this dissertation by summarizing our contributions and suggesting future research directions.



Chapter 2

Literature Review

2D Character animation involves novel view generation, expressive talking face simulation, and limbs movement synthesis in this dissertation. Many research areas are relevant to this dissertation. The following sections thus briefly review the techniques for 2D character animation.

2.1 Image Morphing

To animate characters from still pictures in image morphing community, several studies [28, 48] referred to as shape blending have been conducted. For example, Sederberg and Greenwood [55] employed an interpolation scheme that can interpolate the length of edges and angles between two keyframes. Furthermore, several methods [70] have extracted properties of the given key-poses, and used them to generate characters' motions. Xu et al. [74] synthesized an animal's motion by inferring its motion cycle representing the ordered motion snapshots. By morphing among the ordered poses and refining the appearances of in-betweens, an animal could be animated. Chuang et al. [18] adopted a wavelet curve descriptor combined with Lagrangian dynamics to implement the animation by image morphing. The wavelet coefficients could represent the shapes of images in different resolutions. Lagrangian dynamic equation could be applied to simulate periodic motions. They utilized a non-self-intersecting contour morphing to produce the motion of a similar nature by generating in-betweens. Shutler and Nixon [61] derived Zernike velocity moments from the video about a character's locomotion. Then they used Zernike velocity moments to reconstruct the silhouette of an occluded

character's locomotion which preserved a smooth transition. Our method only employs the correspondence of a character in several still images to synthesize the character's motion.

Besides, several studies [4, 52] referred to character motion synthesis have been conducted by using RBFs for image morphing. RBF is a weighted sum of the translation of a radially symmetric basic function augmented by a polynomial term. It is suitable for fitting smooth functions. It could be further used to warp facial expressions and animate images or drawings [2, 36]. In contrast, circular Gaussian is not an appropriate choice to fit noncircular structures. In this dissertation, we adopt ERBFs to fit contours of characters instead of RBFs. ERBF has the advantage of RBF-like smoothness. Moreover, ERBF is applicable to more general shapes than RBF. Nonlinear approximation of functions in certain general spaces with ERBF networks (referred to as *elliptic basis function* networks in [47]) was proposed. Furthermore, a volumetric approximation and visualization system was developed with ellipsoidal Gaussian functions for a 3D volume (referred to as *ellipsoidal basis functions* in [33]).

DeJuan and Bodenheimer [19] synthesized in-between contours and textures of a character based on RBF interpolation and elastic registration by two given keyframes of an animation. They generated a 3D mesh, which was fitted from the implicit surface generated by RBF interpolation, to obtain in-between contours. Contour points and the corresponding normals of a character in a keyframe were used in RBF method to interpolate an implicit surface. Then a 3D mesh describing the surface was generated. The mesh was sliced in the middle to create in-between contours. In-between textures were synthesized by using an elastic registration. Our approach fits contours with ERBF kernel in image space directly. As mentioned above, ERBF has the RBF-like smoothness and is suitable to more general shapes than RBF. Besides, in-between textures they created would be distorted in complex patterns made up of a few solid colors. LOESS we used could preserve the details without undesired distortion.

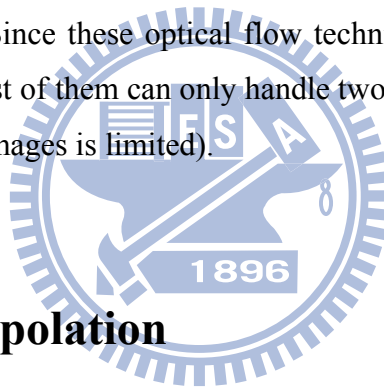
2.2 Shape Deformation

General research on solely image-based animation has recently been carried out based on the shape deformation of a single image. Recently, skeleton-based techniques [24, 76] have been used to deform the shapes by manipulating the space in which they are embedded. These techniques were very efficient in computation and easy to be implemented. However, they did not provide convenient or meaningful interaction tools for the user. Note that the weight tuning for rigging is a painful process for users. Besides, shape matching techniques have been used to shape deformation. Wang et al. [33] utilized uniform grids for 2D shapes and maintained the rigidity of each square in the grid by using shape matching during deformations. They implemented pure rotational transformation for each square. Note that the global area cannot be preserved. Botsch and Sorkine [7] deformed a 2D shape by discretizing the shape into finite elements. However, the computation time was dominated by the complexity of the discretization, and not by the intrinsic complexity of the shape itself.

Furthermore, Alexa et al. [1] considered that the shape deformation of an image should be as rigid as possible. Such deformations would minimize the amount of local scaling and shearing. Igarashi et al. [31] triangulated the input image and minimized the distortion of these triangles in the deformation process by solving a linear system of equations. Schaefer et al. [53] proposed a rigid transformation method by moving least squares. Their study concentrated on specifying deformation by using user-specified handles. In order to generate an animation, users needed to set the next pose by manipulating control vertices. Then the method deformed the entire image plane. Since it ignored the geometry of the shape, unnatural distortions or serious artifacts would be generated when the range of controlling handles were exceeded because of the limitation of the locally influencing extent by using moving least squares. Weber et al. [68] generalized the concept of barycentric coordinates and provided a few examples of known coordinates which could be used for planar shape deformations. Note that the inputs of these works are images and the outputs are also the edited and deformed images. In comparison, our input is just an image and the output is the whole sequence of interpolated frames.

2.3 Image Interpolation

Image-based animation has recently been carried out in computer vision community [27, 34, 35, 49, 62, 65, 73]. Optical flow techniques could be widely adopted for image interpolation. Baker et al. [3] created a collection of optical flow datasets with ground truth. They measured the flow accuracy and the interpolation quality of these optical flow algorithms adopted for image interpolation. While the primary focus of the optical flow algorithms was on evaluating the flow itself. Ghosting and blurring artifacts were visible in their interpolated images even though there were minor errors in the flows. Mahajan et al. [39] proposed an inverse optical flow method. They traced out the path of each pixel between two given images. Then the pixel in the interpolated frame was obtained by moving gradients along the corresponding path and using Poisson reconstruction. Note that they need to determine the flow of each pixel for constructing the path framework. Since these optical flow techniques are based on the disparity of two given images, most of them can only handle two similar images (the disparity or the motion between two images is limited).



2.4 View Interpolation

Several approaches [15, 25, 66] for view interpolation could be applied to generate 2D character animation. Seitz and Dyer [56] proposed a method known as view morphing. The input image was prewarped with the image points through the fundamental matrix computed by computer vision techniques or predefined. Then images were transformed onto the same plane such that their scan lines were aligned. Two views were then morphed, and the interpolated images were postwarped with the user-specified parameters to achieve better morphing quality. However, the quality depended on the number of line correspondences made by users.

2.5 Expression and Viseme Synthesis

Importantly, synthesizing a natural expression or viseme of a character from still pictures is a critical issue for 2D character animation. Chuang and Bregler [17] proposed an audio-driven synthesis technique for creating an expressive facial animation by extracting information from the expression axis of a speech performance. A statistical model based on *principal component analysis* for factoring the expression and visual speech was learned from video. With this analysis of the facial expression, the facial motion could be more effectively retargeted to another 3D face model. Moreover, there is a strong correlation between lips movement and speech [40], and a great number of studies have been conducted on facial animation involving lip-synching (short for lip synchronization). There have been multiple attempts at generating an animated face to match some given speech realistically [6, 8, 20, 21]. Incorporating speech therefore seems crucial to the generation of true-to-life animated faces. Our synthetic faces of the character are also driven by input speech. We reproduce small variations in facial expressions that convey the affective states, moods, and personality of the character. Furthermore, the strong interrelation between facial gestures and prosodic features has been reported in the speech processing literatures [10, 11]. However, the interrelation between facial gestures and individual phonemes is not obvious. Our main focus is to synthesize facial animation possibly driven by analyzing phonemes from input speech.

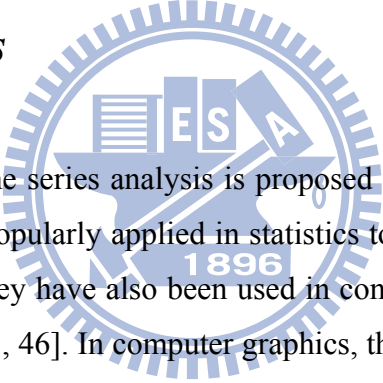
2.6 Motion Capture

Conversely, motion capture technology has enabled users to accumulate large database of human motion which makes the construction of empirical models of a motion feasible. In this technique, joint angles of a performing actor are recorded via sensors. These values are then used to create a character's motion [41]. A deal of research aimed at adapting the motion to different constraints while preserving the style of the original motion. Witkin and Popovic [69] developed a method in which the motion capture data was warped between keyframe-like constraints set by the animator. Warping was done

by overlapping and blending motion clips. Rose et al. [50] developed a method which used RBFs and low-order polynomials to interpolate new motions between example motions obtained from motion capture while maintaining inverse kinematic constraints.

As mentioned previously, Hornung et al. [30] accomplished the motion of photographed persons by projecting them to 3D motion data. However, they stipulated extra 3D information, including a 3D motion database and the corresponding model pose determination, thus increasing the overloads which did not belong to image reanimation. Although they could be applied to animate arbitrary characters from 2D images, their system did not work for motions where the character changed its moving direction, or where it turned its head. In this dissertation, the proposed time series scheme based on a nonparametric Bayesian approach does not have this limitation.

2.7 Time Series



In this dissertation, time series analysis is proposed to synthesize a character's motion. *Time series* has been popularly applied in statistics to forecast the trends in finance and marketing [22, 60]. They have also been used in control system, pattern recognition, or artificial intelligence [5, 46]. In computer graphics, they are adopted for aging trajectory prediction or character motion synthesis. For example, Scherbaum et al. [54] applied aging prediction to images of faces with 3D model reconstruction and *support vector regression* based on RBF kernel. Cai and Hodgins [12] generated animations from various user-defined constraints. Their system learned a state space model from motion capture data. This state space model was based on the deformed linear time series model, and was constructed from the concept of autoregressive model. They transferred constraint-based motion synthesis to a *maximum-a-posterior* (MAP) problem, and developed an optimization framework that generated a natural motion.

Furthermore, variants of *hidden Markov models* (HMMs) [4, 11] have been widely used to create the time series data of motion trajectories representing a character's motion. HMMs learned from human motion data have been employed to interpolate key frames, and synthesize a new style of motion. However, these statistical schemes required full information about a character's motion to train the initial statistical model.

For example, a large motion capture database of human body, or a large amount of user intervention for constraints was necessary. Our proposed approach learns a statistical dynamic model based on *time series*. Moreover, the dynamic behavior of the proposed model is predicted by *Bayesian inference*. More significantly, in contrast to previous methods, the proposed model allows the user to animate character smoothly without additional specified motion information.



Chapter 3

Statistical Approaches

In this chapter, we would introduce the statistical approaches that we use for 2D character animation summarily. We focus on the nonparametric regression model trained from key-poses of still images. *Kernel regression* with ERBFs is employed to train the model to represent the displacements of contours and fit the contours of deformed shape of a character. LOESS is adopted to fill in the color and texture information obtained from the original character in the given image. Thus, we introduce ERBFs in Section 3.1 first. *Kernel regression* with ERBFs is developed for regression prediction and analysis. Then LOESS is described in Section 3.2. Besides, for animating multiple limbs of a character simultaneously, *Bayesian inference* with RJMCMC is proposed to find parameters that satisfy the situation of the regression model. The sampling procedure of RJMCMC is outlined in Section 3.3. Furthermore, as mentioned previously, time series analysis is applied to predict the motion trajectory of the limbs. Hence, we give a description of the time series model in Section 3.4 briefly.

3.1 *Kernel Regression with Elliptic Radial Basis*

Functions

As mentioned before, researchers presented image morphing techniques using RBF for the kernel. RBF kernel is popular for interpolating scattered data. It is suitable for fitting smooth functions of the data and is used to warp facial expressions and animate images or drawings.

However, RBF is based on spatially-limited circular Gaussian distribution function. It has a limitation in fitting the data on long or high-gradient shapes, such as cylindrical shapes, the body, and the head of a character. The radius might reach the shortest boundary of the area and might require numerous small RBFs to fit one long shape, which would be matched to the shape of the character such as the body and the head. Therefore, we use ERBFs instead of RBFs. Note that ERBF has the advantage of RBF-like smoothness and is applicable to more general shapes than RBF. The *kernel regression* model with ERBFs is trained for the prediction of the deformed character's shape in image space.

Note that there are two kinds of ERBFs: axis-aligned and arbitrary directional ERBFs. A comparison of these two basis functions is shown in Figure 3.1. This figure shows a long diagonal data distribution (pixels along contours) and the influences of the two basis functions are drawn overlaid on the data. The data is approximated by two basis functions: axis aligned ERBF shown in Figure 3.1 (a) and arbitrary directional ERBF shown in Figure 3.1 (b). The major axis of the ellipse with arbitrary directional ERBFs is aligned along the contour of a character which is a long diagonal data distribution (gray region). For achieving more accurate quality with smaller number of basis functions, arbitrary directional ERBFs are applied to fit the contours of a character in a still picture.

In general, let $\vec{u} = (x, y)$ be a vector of the pre-sampled data and $\vec{v} = (\mu_x, \mu_y)$ be a center vector of an elliptic Gaussian. An arbitrary directional ERBF can be represented in a matrix form as follows:

$$\eta(\vec{u}, \vec{v}) = \exp \left\{ - \frac{(\vec{u}_x - \vec{v}_x)^T A_{\theta_x, \alpha_x} (\vec{u}_x - \vec{v}_x)}{2\sigma_x^2} - \frac{(\vec{u}_y - \vec{v}_y)^T A_{\theta_y, \alpha_y} (\vec{u}_y - \vec{v}_y)}{2\sigma_y^2} \right\}, \quad (3.1)$$

$$\vec{u} = \vec{u}_x + \vec{u}_y = [x \quad 0]^T + [0 \quad y]^T, \quad (3.2)$$

$$\vec{v} = \vec{v}_x + \vec{v}_y = [\mu_x \quad 0]^T + [0 \quad \mu_y]^T, \quad (3.3)$$

$$A_{\theta_i, a_i} = \begin{bmatrix} \cos \theta_i / a_i & \sin \theta_i / a_i \\ -a_i \sin \theta_i & a_i \cos \theta_i \end{bmatrix}, \text{ where } i \in \{x, y\}, \quad (3.4)$$

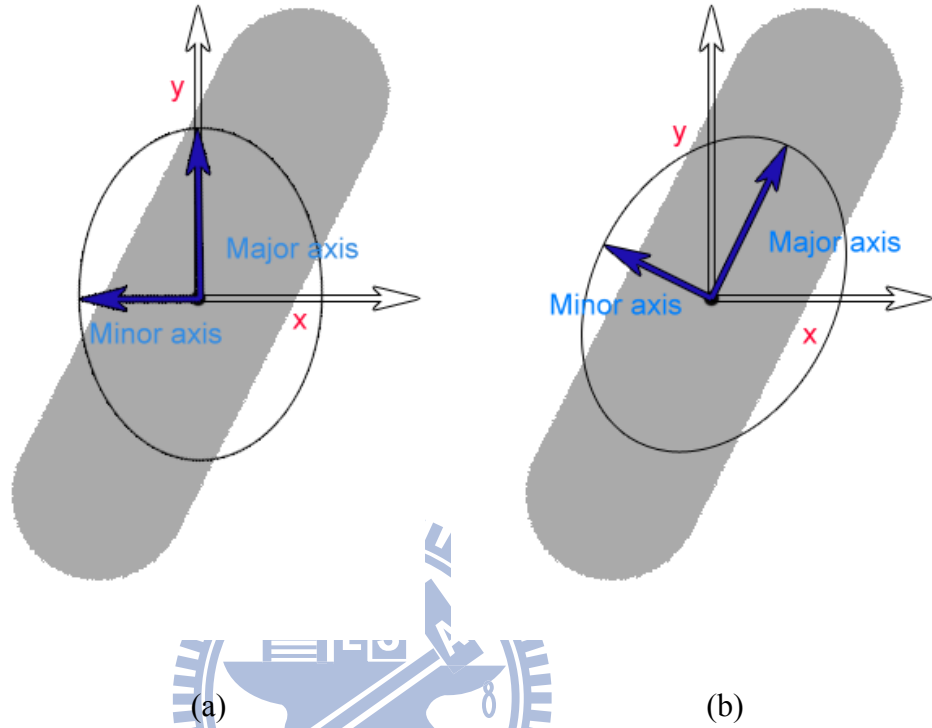


Figure 3.1: Comparison of ERBFs. (a) Axis aligned ERBF. (b) Arbitrary directional ERBF. The influence range of each basis function is shown as blue arrows and black curve.

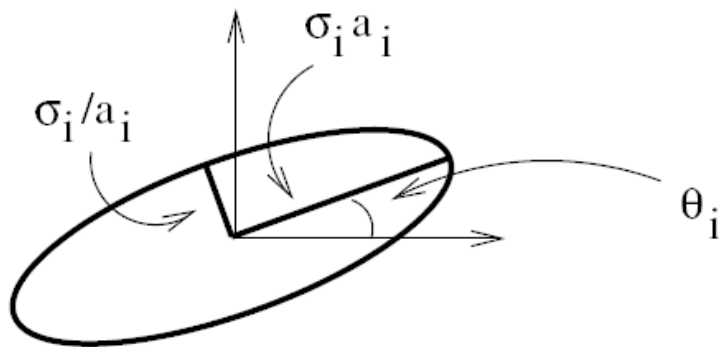


Figure 3.2: Schematic diagram of an arbitrary directional *elliptic radial basis function* (ERBF).

where σ_i^2 for $i \in \{x, y\}$ is the covariance of Gaussian along i -axis. The orientation θ_i (the angle between the major axis of ellipse and i -axis) and the aspect ratio a_i^2 are used to transfer to an arbitrary directional ERBF, as shown in Figure 3.2. Moreover, the transformation matrix A_{θ_i, a_i} , which contains a rotation and scaling component, is applied for alignment along the data distribution. In our work, the major axis of the ellipse is aligned along the contour of the character, as shown in Figure 3.1 (b). For the mathematical details of Equation (3.1), it can be derived from a *hyper radial basis function* (HRBF).

HRBF is computed by using the *Mahalanobis distance* [29], which is defined in the matrix form as follows:

$$\eta(\bar{u}, \bar{v}) = \exp\left(-(\bar{u} - \bar{v})^T \Sigma (\bar{u} - \bar{v})\right), \quad (3.5)$$

for $\Sigma = \text{diag}(\sigma_1^{-2}, \dots, \sigma_N^{-2})$ and $\sigma_1, \dots, \sigma_N \in \mathfrak{R}^+$,

where σ_N^2 should be the covariance of the multidimensional Gaussians rather than the single variance. HRBF differs from a standard RBF insofar each axis of the input space $\mathcal{X} \subseteq \ell_2^N$ (the space of square summable sequences of length N) has a separate smoothing parameter, i.e., a separate scale onto which the differences on this axis are viewed. It is worth mentioning that RBF kernels map the input space onto the surface of an infinite dimensional hyperspace. Note that $N = 2$ in arbitrary directional ERBF kernel represents the analysis of data distribution along the major axis and the minor axis in an ellipse. Along the orientation of arbitrary directional ERBF (the major axis and the minor axis), Equation (3.1) is constructed.

In this dissertation, we formulate the problem of 2D character animation as regression analysis. Given two key-poses of a character in still images, we analyze the contours of a character and represent the displacements of these contours as ERBFs. Then these ERBFs constructing an implicit regression surface can be used to predict the new position after deforming the shape of the character. In other words, we form a regression model trained from the given key-poses. The model is adopted to predict the motion of the character. Now, we derive the equation of *kernel regression* with ERBFs.

In general, the relationship of the response \bar{r} and the predictor \bar{u} can be described as

$$\vec{r} = f(\vec{u}) + \varepsilon. \quad (3.6)$$

Considering the above equation, $f(\cdot)$ denotes an unknown and smooth surface indicating the relationship between \vec{r} and \vec{u} , commonly termed the regression surface representing the shape deformation. Additionally, the error ε is assumed to come from a *Normal distribution* $N(0, \tau^2)$ in Equation (3.6), where τ^2 denotes the noise variance. Note that the regression surface is estimated by using *kernel regression* with ERBFs. ERBF is an appropriate choice to fit smooth functions for the form of $f(\cdot)$.

As mentioned above, \vec{v} denotes the center vector of elliptic Gaussian (ERBF). The proposed regression model consists of a radial component and an affine component. Moreover, a radial one is developed as a linear combination of a set of basis functions and their corresponding coefficients.

$$f(\vec{u}) = \sum_{j=1}^k \beta_j \eta(\vec{u}, \vec{v}_j) + T(\vec{u}), \quad (3.7)$$

where β_j denotes the suitably chosen coefficient of the j -th elliptic Gaussian $\eta(\cdot)$, \vec{v}_j is the related center vector, and k is the number of basis functions in the model. Note that there is the relative covariance of the j -th elliptic Gaussian along arbitrary i -axis $\sigma_{j,i}^2$. $\eta(\cdot)$ is the radial component chosen as an arbitrary directional ERBF. Moreover, $T(\cdot)$ represents the affine component. In our work, we would further train the model to predict the motion of the character by synthesizing the contours of the character's deformed shape.

3.2 Locally Weighted Regression

After synthesizing the contours of the character's deformed shape, we need to fill the contours and preserve details simultaneously. It is also motivated by following the process of the traditional 2D animation production. A similar issue occurs when the line art is scanned and goes to the next step of ink and paint. Hence, a local-fitting methodology called LOESS is applied to preserve the details or features of characters (that is filling in the color and texture information obtained from the original character

in the given image). Like *kernel regression*, LOESS is a procedure for fitting a regression surface to data through multivariate smoothing. LOESS uses the data from the neighborhood around a specific location. In other words, LOESS performs a linear regression on points in the data set, which are weighted by a kernel centered at that pre-defined location. It is much more strongly influenced by the data points that lie close to the location pre-defined according to some scaled Euclidean distance metric. This is achieved by weighting each data point according to its distance to the pre-defined location: a point very close to it is given a weight of one and a point far away is given a weight of zero. Note that the shape of the kernel is a design parameter for which many possible choices exist. The original LOESS uses the tri-cube weighting function. Nonetheless, we have used the Gaussian kernel to estimate the weights in the range of unit circle, as shown in Figure 3.3 (b).

During a LOESS prediction, the specific location (red dot) x_0 , which would be filled color or texture information, is supplied. LOESS performs a linear regression on the sampled contour points weighted by a kernel centered at x_0 . Given m pairs of points sampled along the contour (purple dots) of the character in the input image and the corresponding new locations of these points, the weight of the i -th sampled contour point x_i with Gaussian kernel is

$$w_i(x_0) = w(x_i - x_0) = \exp\left(-s \|(x_i - x_0)\|^2\right), \quad (3.8)$$

where $1 \leq i \leq m$, $s = 1/2W_{\text{kernel}}^2$, and $m = \sum_i w_i(x_0)$ for m data points. s is a smoothing parameter that determines how quickly weights decline in value as one moves away from x_0 , W_{kernel} is the kernel width or bandwidth which controls the amount of localness in the regression.

Let x_i be the predictor of the regression and y_i be the response. The regression function is specified by using an estimated local multivariate polynomial as follows:

$$\hat{y}_i = \zeta_1 t_1(x_i) + \zeta_2 t_2(x_i) + \dots + \zeta_M t_M(x_i), \quad (3.9)$$

where $t_j(\cdot)$ is a function that produces the j -th term in the polynomial, and ζ_j is the j -th term of coefficients to be estimated. Equation (3.9) can be rewritten for matrix manipulation, which can be easily extended to datasets with many inputs:

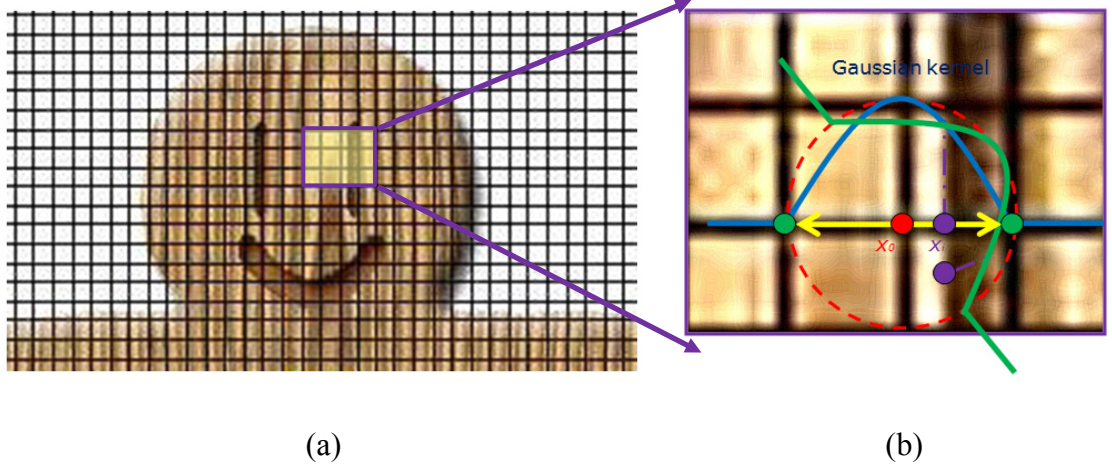


Figure 3.3: LOESS analysis. (a) Original image with a uniform grid. (b) The zoom-in view of the image. LOESS with Gaussian kernel is applied to estimate the weights.

$$\hat{y}_i = \zeta^T t(x_i), \quad (3.10)$$

where ζ is the matrix form of the coefficients vector $(\zeta_1, \zeta_2, \dots, \zeta_M)$ and $t(\cdot)$ is the matrix form of the polynomial terms $(t_1(x_i), t_2(x_i), \dots, t_M(x_i))$. Given m pairs of (x_i, y_i) , the general way to estimate $\hat{\zeta}$ is by minimizing the sum squared residuals.

$$\hat{\zeta} = \arg \min_{\zeta} \sum_{i=1}^m (y_i - \zeta^T t_i)^2, \quad (3.11)$$

where $t_i = t(x_i)$. Furthermore, note that the features of the original character interiors are considered as the specific locations to be preserved during a LOESS prediction. According to the distance to these specific locations, the warping degree is adapted to these features and is constrained by them. Unlike global deformation, LOESS can maintain local features invariant during deformations while minimizing unnatural distortion. Thus, $\hat{\zeta}$ is chosen by minimizing locally weighted sum of squared residuals.

$$\hat{\zeta} = \arg \min_{\zeta} \sum_{i=1}^m w_i(x_0)^2 (y_i - \zeta^T t_i)^2, \quad (3.12)$$

where $t_i = t(x_i)$ and $w_i(\cdot)$ is defined in Equation (3.8). The minimization can be obtained by the least-squares normal equations. In our work, we further fill in the color and texture information of the deformed character by using Equation (3.10) with the estimated regression coefficient vector $\hat{\zeta}$.

3.3 Reversible Jump Markov Chain Monte Carlo (RJMCMC) Sampler

Instead of using least-squares method to estimate unknown parameters during regression analysis, RJMCMC sampler is applied to estimate the optimized regression parameters. For instance, the procedure to estimate the parameters of *kernel regression* with ERBFs consists of three steps as follows:

1. Set Up Proper Priors: Recalling Equation (3.7), let \bar{v}_j be the mean of the j -th elliptic Gaussian, while β_j denotes the corresponding coefficient. We define $\sigma_{j,i}^2$ as the covariance of the j -th elliptic Gaussian along i -axis. We begin with a fairly flat Gaussian prior on the basis coefficient $\beta_j \sim N\left(0, \frac{\tau^2}{p_{precision}}\right)$, where $p_{precision}$ is the precision of the coefficient prior. τ^2 is the noise variance, and $\frac{1}{\tau^2} \sim \text{Gamma}(0.1, 0.1)$. A vague but proper Gamma prior distribution represents ignorance of the noise process and avoids inverting large matrices within each iteration of RJMCMC. We set $p_{precision} = 0.01$ and $\tau^2 = 1$ initially and they would be updated during RJMCMC process.

2. Determine Initial Parameter Value: Set the initial dimension k of the model equal to 3, that is intercept term plus the number of predictors. Then we use k -means clustering to set the starting center vector \bar{v}_j for each k -means group of anchor points. In addition, the covariance $\sigma_{j,i}^2$ is computed for each group. Besides, calculate β_j by using least-squares fitting.

3. Iterate RJMCMC Sampler Until Sufficient Samples: In the RJMCMC algorithm, we propose the next state of the chain representing a new basis function according to the following criteria. First, draw a uniform random variable $u \sim U(0,1)$. If $u < 0.33$, then perform the Birth step. In the Birth step, we would add a basis function (ERBF) in the model. Then the corresponding parameters are updated by k -means clustering simultaneously. Recalling Figure 3.2, for each k -means group, the transformation matrix A_{θ_i, a_i} is computed for adding this basis function. If $0.33 \leq u \leq 0.66$, then perform the Death step. In the Death step, we would lose a basis function. We just select one basis function at random and remove it. If $u > 0.66$, then perform the Move step. In the Move step, we choose a basis function from the model at random and reset its mean vector to

another random position. Next, the corresponding parameters are updated.

Then we would compute the marginal log-likelihood of the model and draw k new coefficients β_j . Given n pairs of predictors \vec{u}_d and corresponding responses \vec{r}_d , we would compute the marginal log-likelihood for the creditable change of state as follows:

$$L(\vec{r}|\Theta) = -n \log \tau - \frac{1}{2\tau^2} \sum_{d=1}^n \left\{ \vec{r}_d - \hat{f}(\vec{u}_d) \right\}^2, \quad (3.13)$$

where \vec{r} is a general representation for the response of regression, as defined in Equation (3.6).

Let X be the responses of basis functions in the matrix form. Y denotes the corresponding responses of the regression model in the matrix form. P represents the matrix form of prior precision $p_{precision}$. β represents the matrix form of k coefficients β_j defined in Equation (3.7). Furthermore, β is obtained from the marginal posterior distribution with posterior mean $\bar{\beta} = (X^T X + P)^{-1} X^T Y$ and modified standard deviation. Note that the initial standard deviation is drawn from the noise variance τ^2 and modified to be the upper triangle of posterior variance matrix $(X^T X + P)^{-1}$ obtained by using *Cholesky decomposition*.

Next, consider to accept the proposed change of next state. We draw a uniform random variable $u \sim U(0,1)$ first. If u is less than the ratio of the marginal likelihood of proposed next state to the marginal likelihood of original one, then accept the proposed change to the model and update the state of the *Markov chain*. Otherwise set the next state to be the current state. Then update prior precision $p_{precision}$ by drawing a random variable from a *Gamma distribution* and is modified by the sum of squares of β_j every 10 iterations. Recalculate the coefficients β_j from the marginal posterior distribution with the updated prior precision $p_{precision}$. Furthermore, draw a random variable τ^2 from a *Gamma distribution* for a new noise variance. Given response \vec{r} defined in Equation (3.6), τ^2 is modified by posterior sum of squares error for the next iteration.

Repeat RJMCMC process and record the number of states. An initial portion of the chain is discarded to ensure stability. If the number of states is greater than the discarded portion, then compute $\hat{f}(\cdot)$ defined in Equation (3.6) by the recorded parameters of the current model for synthesizing limbs movement. All the simulations are run with a

burn-in period of 5000 iterations of RJMCMC followed by 10000 samples.

3.4 Time Series Analysis

As mentioned before, *time series* is applied to generate smooth and continuous limbs' movements. In our work, ARMA is used to analyze limbs' movements of a character in several previous time slices for estimating the motion trajectories. Then we could synthesize the current movements following these estimated trajectories. The general form of a time series model is considered as

$$D_t = f_{TS}(D_{t-1}, \dots, D_{t-p}; \alpha_{t-1}, \dots, \alpha_{t-q}) + \alpha_t, \quad (3.14)$$

$$\alpha_t = C\nu_t, \quad (3.15)$$

where D_t denotes a univariate time series and $f_{TS}(\cdot)$ indicates an unknown function of time series. p and q represent non-negative integers. ν_t is a sequence of random variables assumed to come from a *Normal distribution* with mean zero and variance one. C is assumed to be a constant. Based on this general form, ARMA is formulated as follows:

$$f_{TS}(D_{t-1}, \dots, D_{t-p}; \alpha_{t-1}, \dots, \alpha_{t-q}) = \sum_{i=1}^p \phi_i D_{t-i} - \sum_{i=1}^q \kappa_i \alpha_{t-i}, \quad (3.16)$$

where ϕ_i and κ_i are the coefficients of parameters in this model. It is similar to the time series model proposed by Chen et al. [13], except that they assumed the functional form of $f_{TS}(\cdot)$ was a known linear function whereas we assumes $f_{TS}(\cdot)$ is estimated nonparametrically along with the Bayesian estimation of ERBFs already described in Section 3.1 and Section 3.3 in order to add smooth variety of the time series data, that is, we develop ERBF kernel in the original time series model with parameters inferred by using RJMCMC. We further use this nonparametric time series model to forecast the current limbs' movements of the character from his several previous poses.

Chapter 4

Two-scale Image Abstraction

Generating a natural-looking 2D character animation from still photographs or paintings can be considered to analyze and simulate the character's motion in that image. Note that a photograph contains redundant information. The raw format of a photograph may have 16-bits or even 24-bits per color channel. Using all contours of the character extracted from raw photographs for statistical analysis is not practical and useful. Hence, it is necessary to obtain contours of interests of the characters. We advocate the two-scale abstraction similar to progressive image abstraction proposed by Farbman et al. [23]. The proposed abstraction method is based on a two-scale decomposition of the image consisting of a base layer, which encodes large-scale variations of pixels, and a detail layer. The base and detail layers would be obtained by using an edge preserving filter called the bilateral filter [64]. Given photographs, the bilateral filter is applied to obtain regions of interest. The selected contours of a character from the detail layer, which represent important features, and the contours of that character in the base layer are used to estimate the character's motion. The redundant information of a photograph is filtered by the bilateral filter so as to animate 2D character from arbitrary still pictures by the proposed statistical approaches.

4.1 Color Space Transformation

In order to keep the regions of interest, we propose the two-scale image abstraction based on the bilateral filter. It classifies the image into a base layer and a detail layer. Important features can be preserved by adopting the contours selected from the detail

layer and the contours of the base layer to train the statistical model. In contrast, unimportant features can be filtered by applying the contours of base layer to the model only.

Tomasi and Manduchi [64] suggested computing the bilateral filter on a perceptually uniform feature space, such as CIELab [72]. Perceptually uniform means that a change of the same amount in a color value should produce a change of the same visual importance. Only perceptually similar colors are averaged together while bilateral filtering is carried out in CIELab color space. Moreover, only perceptually important details are preserved. The values used by CIELab color space are called L^* , a^* , and b^* . L^* component closely matches human perception of lightness, which is the luminance signal that can estimate the difference between light and dark. a^* represents the difference between red and green. b^* represents the difference between yellow and blue. Unlike RGB color space, CIELab is based on a large body of psychophysical data concerning color-matching experiments performed by human observers, and is designed a practical approximation to color processing in a human visual system model. In contrast, RGB models the output of physical devices rather than human visual perception. CIELab can thus be used to adjust the lightness contrast by using L^* component. Furthermore, CIELab can make accurate color balance corrections by modifying output signals in a^* and b^* components.

The three coordinates of CIELab represent the lightness intensity L^* , its position on a pure red and pure green scale a^* , and its position on a pure yellow and pure blue scale b^* . Note that $L^* = 0$ yields black and $L^* = 100$ indicates diffuse white. $a^* = -127$ indicates pure green and $a^* = 127$ indicates pure red. $b^* = -127$ indicates pure blue and $b^* = 127$ indicates pure yellow. The red/green and yellow/blue opponent channels are computed as differences of lightness transformations of cone responses. Note that the nonlinear relations for L^* , a^* , and b^* are intended to mimic the nonlinear response of the human eye. Furthermore, uniform changes of components in the CIELab color space aim to correspond to uniform changes in perceived color, so the relative perceptual differences between any two colors in CIELab color space can be approximated by taking the Euclidean distance between them. The Euclidean distance is directly proportional to the difference between the two colors as perceived by the human eye.

CIELab can be computed via simple formulas from nonlinearly-compressed CIEXYZ color space coordinates $(X_{CIE}, Y_{CIE}, Z_{CIE})$, which is not particularly perceptually uniform. Now, we would transform the initial color space RGB of a raw image into CIELab through CIEXYZ.

$$\begin{bmatrix} X_{CIE} \\ Y_{CIE} \\ Z_{CIE} \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (4.1)$$

Then three components of CIELab are obtained from CIEXYZ.

$$\begin{cases} L^* = 116\sqrt[3]{Y_{CIE}/Y_n} - 16 & \text{for } Y_{CIE}/Y_n > 0.008856 \\ L^* = 903.3 Y_{CIE}/Y_n & \text{otherwise} \end{cases}, \quad (4.2)$$

$$a^* = 500 [CIE(X_{CIE}/X_n) - CIE(Y_{CIE}/Y_n)], \quad (4.3)$$

$$b^* = 200 [CIE(Y_{CIE}/Y_n) - CIE(Z_{CIE}/Z_n)], \quad (4.4)$$

$$\text{where } \begin{cases} CIE(th) = \sqrt[3]{th} & \text{for } th > 0.008856 \\ CIE(th) = 7.787th + 16/116 & \text{otherwise} \end{cases}. \quad (4.5)$$

Here X_n , Y_n , and Z_n are the *tristimulus values* corresponding to the reference white point. They are specified respectively as 0.950456, 1.000000, and 1.088754.

4.2 Bilateral Filter

Next, we use the bilateral filter to classify the original image I into a base layer encoding large-scale variations and a detail layer. Note that the detail layer is applied to select the regions of interest and the features which should be preserved. The bilateral filter is a non-linear filter, where each pixel in the filtered result is a weighted mean of its neighbors, with the weights decreasing both with spatial distance and with difference in value. The bilateral filter can be defined as

$$BF(I)_p = \frac{1}{k(p)} \sum_{q \in \Omega} G_{\sigma_s}(\|p - q\|) G_{\sigma_l}(\|I_p - I_q\|) I_q, \quad (4.6)$$

$$k(p) = \sum_{q \in \Omega} G_{\sigma_s}(\|p - q\|) G_{\sigma_l}(\|I_p - I_q\|), \quad (4.7)$$

where Ω is the whole image range, and the subscripts p and q indicate spatial locations of pixels. I_p and I_q are intensity values of pixel p and q . The kernel functions G_{σ_s} and G_{σ_l} are typically Gaussians, where σ_s determines the spatial support, while σ_l controls the sensitivity to edges. $k(p)$ is a normalized function. $BF(I)_p$ is the filtered result of the pixel p . That represents the base layer of the original image. A bilateral filter allows combining three color channels in CIELab and measuring photometric distances between pixels in the combined space. Furthermore, the detail layer is the division of the original image by the base layer. The ratio is computed on each color channel separately and is independent of the signal magnitude. The ratio captures the local detail variation in the original image and is commonly called a quotient image [59] or a ratio image [37] in computer vision community.

4.3 Image Abstraction

We transform the color space of the base layer and the detail layer back to RGB for the further process. As mentioned previously, X_n , Y_n , and Z_n are specified respectively as 0.950456, 1.000000, and 1.088754. The reverse transformation to CIEXYZ is

$$X_{CIE} = X_n (P_{temp} + a^*/500)^3, \quad (4.8)$$

$$Y_{CIE} = Y_n P_{temp}^3, \quad (4.9)$$

$$Z_{CIE} = Z_n (P_{temp} + b^*/200)^3, \quad (4.10)$$

$$\text{where } P_{temp} = (L^* + 16)/116. \quad (4.11)$$

The color space RGB can be obtained by the transformation matrix.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} * \begin{bmatrix} X_{CIE} \\ Y_{CIE} \\ Z_{CIE} \end{bmatrix}. \quad (4.12)$$

Furthermore, the detail layer is attenuated to achieve a stylized abstract look. In this work, the short line segments in the detail layer are regarded as unimportant regions or noises. Hence, the median filter is applied to smooth or denoise the detail layer. The idea of median filtering is to calculate the median of neighboring pixels' values. It can be done by repeating these steps for each pixel in the image, as described follows:

1. Store the neighboring pixels in an array called the window. Note that the neighboring pixels are chosen by a box.
2. Sort the window in numerical order.
3. Pick the median from the window as the pixels value.

Finally, the base layer is overlaid with edges extracted from the filtered detail layer.

4.4 Experimental Results

The proposed method yields several results. Figure 4.1 shows the abstracted image by using the bilateral filter. Figure 4.1 (a) is the original real image. Figure 4.1 (b) is the coarsened image of the base layer with $\sigma_s = 12$ and $\sigma_l = 0.15$. In our experiment, we found these parameters to be better suited for applications that discarded or attenuated some of the details, such as image abstraction. Thus, we used these parameters throughout this dissertation for most real images. Figure 4.1 (c) is the selected contours from the detail layer by using the median filter. Moreover, Figure 4.1 (d) shows the image abstraction result.

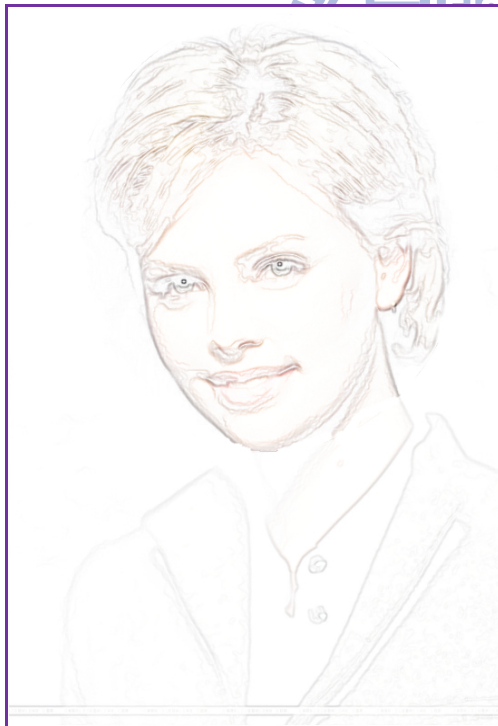
Figure 4.2 shows another example of the abstracted image. Figure 4.2 (a) is the original real image. Figure 4.2 (b) is the coarsened image of the base layer by using the bilateral filter. Figure 4.2 (c) is the selected contours from the detail layer by using the median filter. Finally, Figure 4.2 (d) shows the image abstraction result.



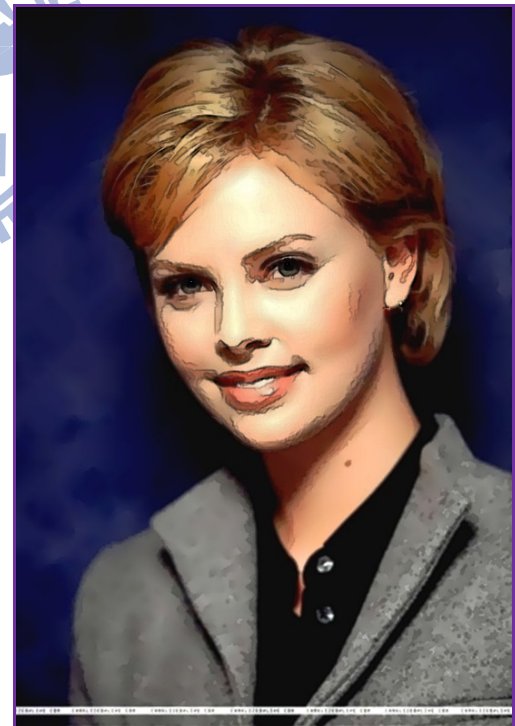
(a)



(b)



(c)



(d)

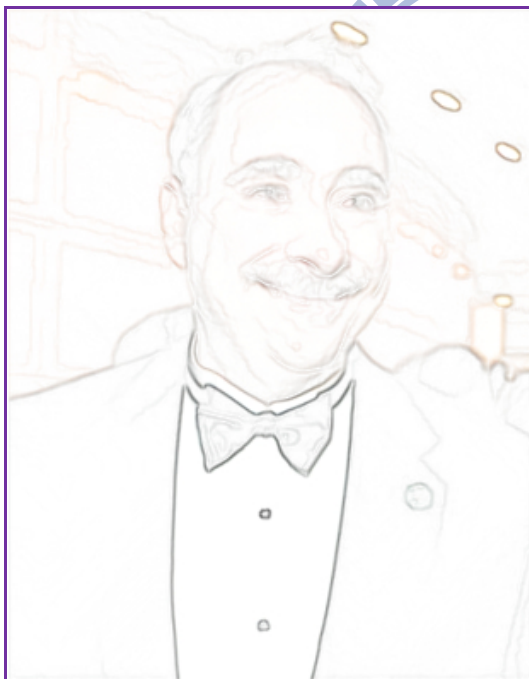
Figure 4.1: Image abstraction computed using our two-scale decomposition. (a) The original real image (the photo of Charlize Theron). (b) The base layer. (c) The selected contours from the detail layer. (d) The final abstracted image.



(a)



(b)



(c)



(d)

Figure 4.2: Another example of image abstraction. (a) The original real image (the photo of David Axelrod). (b) The base layer. (c) The selected contours from the detail layer. (d) The final result.

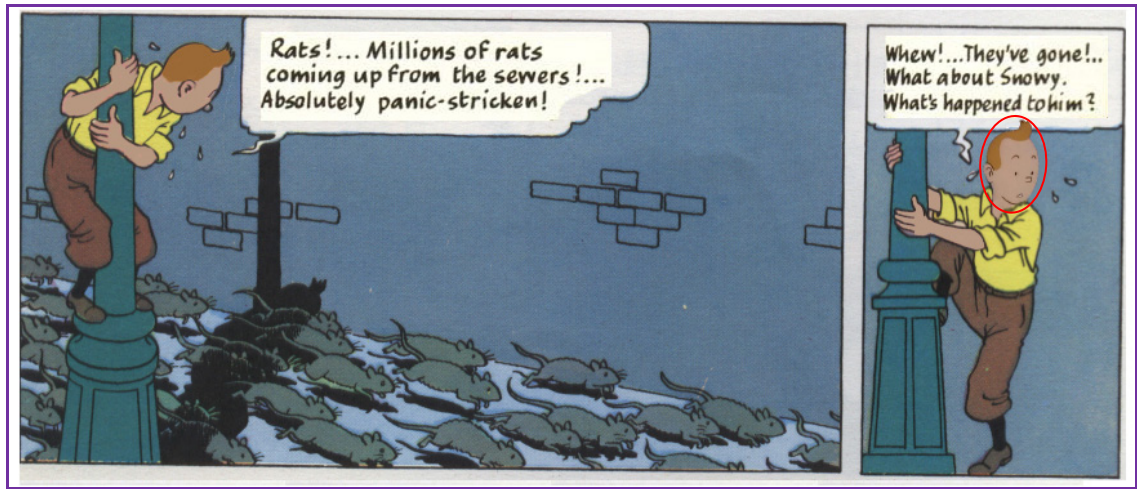
Chapter 5

Novel View Generation

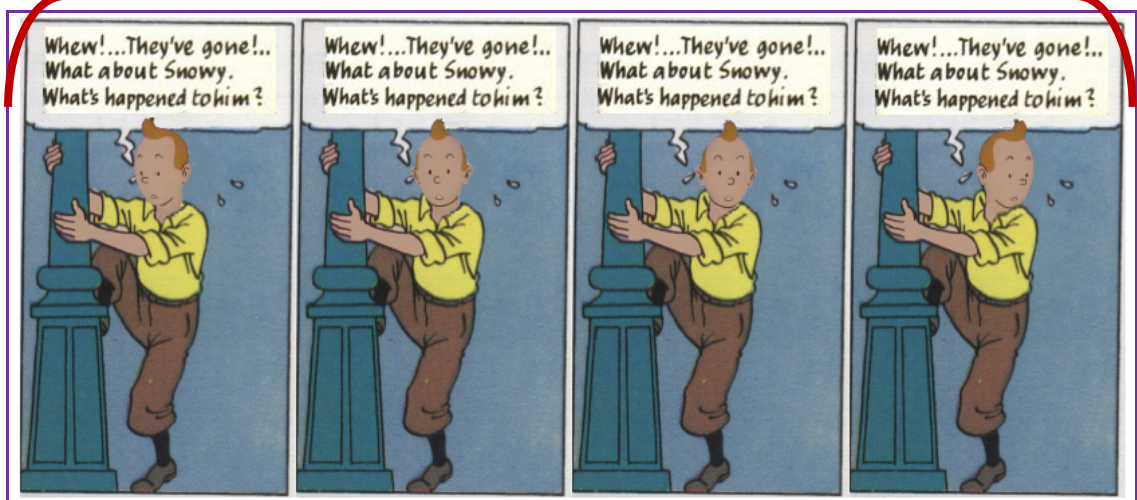
For animating 2D characters in still pictures, we proposed a statistical method based on nonparametric regression analysis to generate a novel view of a character. *Kernel regression* with ERBFs is introduced to fit the contours of a character and is applied to infer the corresponding displacements of the contours. LOESS is applied to fill in the color and texture information obtained from the original character in the given picture.

5.1 View Interpolation

In our work, we would take the idea of creating deformations directly in image space one step further by making 2D characters move. Actually, we propose a nonparametric regression model to animate the characters from still images. For instance, animating the character in a comic could be carried out by the creation of a novel view, as shown in Figure 5.1. It shows two continuous frames in the original comic, which can be regarded as two different scenes, and the synthesized frames from a single input frame. Figure 5.1 (a) are the original frames in the comic. Figure 5.1 (b) shows the synthesized novel views. Note that the model is trained to fit the shape and detail of the character between two key-poses from a given frame and its reverse, while minimizing unnatural distortion. Then the trained model is applied to synthesize the smooth transition between these key-poses.



(a)



(b)

Figure 5.1: Novel view generation in a comic. (a) Two continuous frames in a comic. (b) The frames synthesized from a single frame. © Georges Remi (Hergé)

As mentioned previously, the proposed model is based on the prediction abilities of *nonparametric regression*. *Kernel regression* approximates the shape of a deformed character between two key-poses or moving templates indicating different poses by the prior use of a set of kernel functions. Circular Gaussian distribution function is not an appropriate choice to fit contours, which have noncircular structures like characters or human-like subjects. Instead of RBF kernel based on spatially-limited circular Gaussian, *kernel regression* using *elliptic radial basis functions* (ERBFs), specifically elliptic Gaussians which provide less learning time, is applied for contours fitting during shape

deformation (defined as shape deforming). Although ERBFs require more computation during optimization, better quality is obtained with smaller number of basis functions. Furthermore, the local-fitting methodology is also applied to preserve important features within the deformed shape (that is filling in the color and texture information obtained from the original character in the given image). *Locally weighted regression*, or LOESS, is used to preserve the features or details by fitting a function of independent variables locally.

5.2 Algorithm Overview

In Figure 5.2, the outline reflects the structure of our proposed method for novel view generation. Considering Figure 5.2, we briefly describe our method in the following paragraphs.

1. Character Extraction: As mentioned in Chapter 4, in addition to the paintings or the comic, the input images like real images are filtered by using the bilateral filter first. Then in order to reduce the effects of the background upon deformations, we extract characters from the input image. We use level-set-based *GrabCut* to extract characters and features, as described in Section 5.3. Similar regions are extracted by the level set method. The bounding box of all regions is then used by *GrabCut*.

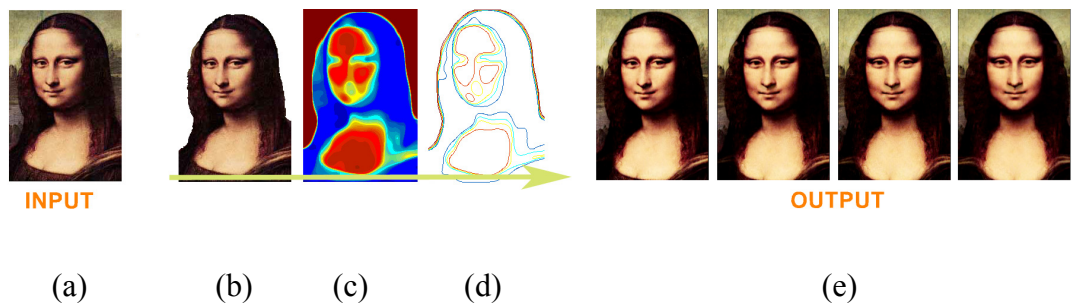


Figure 5.2: This example shows the picture of Mona Lisa. (a) The original input image. (b) The character is extracted, (c) who is described by the similar parts found by level-set-based *GrabCut*, and (d) the contours are applied to build the nonparametric regression model for shape deforming and detail preserving. After deforming the shape and preserving details, several resulting frames in the synthesized Mona Lisa's views are shown in (e).

The boundaries of regions corresponding to the matte produced automatically are further applied to obtain the final character matte. The foreground and background are separated successfully. Besides, the facial features are extracted simultaneously by the level set method. As shown in Figure 5.2 (b), Mona Lisa is extracted, which is described by the similar parts found by level-set-based *GrabCut* in Figure 5.2 (c), and the corresponding contours shown in Figure 5.2 (d) are applied to build the nonparametric regression model for shape deforming and detail preserving.

2. Statistical Approaches: Before we generate novel views of a character in a still image, we apply *nonparametric regression* for novel view generation mentioned in Section 3.1 and Section 3.2. Note that the proposed statistical approaches are not only used to generate novel views of a character, but also adopted to create an expressive talking face and synthesize smooth limbs movements. In practice, our framework for 2D character animation consists of these statistical approaches.

3. Novel View Generation: Given one key-pose of a character and its reverse, we deform the shape of the character by applying a trained nonparametric regression model for generating novel views of the character. The process can be divided into two steps: shape deforming and detail preserving. In the shape deforming, the correspondence in training data set is constructed first. *Kernel regression* with ERBFs is employed to train the model to represent and fit the contour of deformed shape, as described in Section 5.4. In the detail preserving step, as described in Section 5.5, LOESS is adopted to fit the details of the deformed shape. LOESS is suitable for detail preserving in accordance with the previously fitted contours. Figure 5.2 (e) shows finally the resulting images.

5.3 Character Extraction

In addition to the paintings or the comic, real images are filtered by using the bilateral filter. Then we adopt the level set method to extract regions with a similar color distribution in the image. The level set method, proposed by Osher and Sethian [38, 57, 58], is an approach for approximating the dynamics of moving curves and surfaces. Note that we choose HSV color space [72], it is not only close to the people understanding of colors, but also is regarded as the best option in judgment on the color

changes. It consists of three components, namely representatives of hue H (hue), saturation S (saturation), and brightness V (value). In practice, HSV color space allows combining the three color channels appropriately. Moreover, the combined color difference can be made to correspond to the distance between two points in the color space. It is suitable for image segmentation and analysis. We introduce the concept of color gradient information of images, instead of using gray gradient to update the curve evolution function of the level set method. Furthermore, these regions representing the facial features of a character are found simultaneously.

After feature extraction, *GrabCut* [51] is then applied to separate foreground (characters) and background. *GrabCut* is powerful object matting tool. However, it requires an initial incomplete trimap which represents the seeds of foreground and background for the underlying graph cut algorithm. That is, no hard foreground labeling is done at all. The region of background is determined by users as a strip of pixels around the outside of the marked rectangle. The Gibbs energy minimization is computed and the object matting for character extraction is applied.

We construct a bounding box of all these regions extracted by using the level set method. Then we use the bounding box for *GrabCut* instead of the initial incomplete trimap. Note that the extracted regions correspond to the regions of a character matte with the similar color distribution. The pixels inside the contours of the regions are considered the foreground distribution replacing users' refinement during the iterative minimization in *GrabCut*. Subsequently, the entire energy minimization process would be performed iteratively with the updated foreground distribution. The process is guaranteed to converge at least to a local minimization since the energy decreases monotonically. After convergence is achieved, the character matte is extracted successfully.

Note that we choose HSV color space. Due to the hue, saturation, and brightness of the three components to determine changes in color, the level set method with color gradient enriches the way that only uses gray gradient to judge whether at the border. Since joining the color factor, the character and feature extraction is robust for the images, which the gray level of the background is close to the gray level of the foreground. The final character and features matte is shown in Figure 5.2 (b).

5.4 Shape Deformation Using *Kernel Regression* with ERBFs

As mentioned previously, we use level-set-based *GrabCut* to extract the character and similar regions in the character. After extracting the characters in the input image and its reverse, we train a regression model with ERBF kernel. First, in Section 5.4.1, an initial solution to regression parameters is obtained. Then we discuss how to train the model and fit the character's deformed shape by the trained model, as described in Section 5.4.2.

5.4.1 The Determination of Initial Values

The initial guesses are important for further optimization convergence in model learning. Before setting the initial value of center and covariance, the correspondence with regard to feature alignment should be done. First, we create a window and use it to compute the curvature along each region boundary in the face. Note that these regions in the face are obtained by using the level set method. We choose the top five curvatures from the window interiors and sample points along these contours. The five bounding boxes of these sets of sample points are the feature blocks shown in Figure 5.3 (a) and Figure 5.3 (b). The structure of these feature blocks (that is the order of the feature blocks) is constructed to maintain the spatial relationship among these features, as shown in Figure 5.3 (c). Note that the structure is similar to the tree structure. However, there are no root and leaf nodes in our work. We only use the link between two nodes (feature blocks) to record the spatial relationship or the order of two nodes. Subsequently, *Tchebichef moments* (TMs) [43] of these blocks are used to determine the correspondence in the other key-pose, which is obtained by reversing the original input image, for spatial constraints, as shown in Figure 5.3 (d) and Figure 5.3 (e).

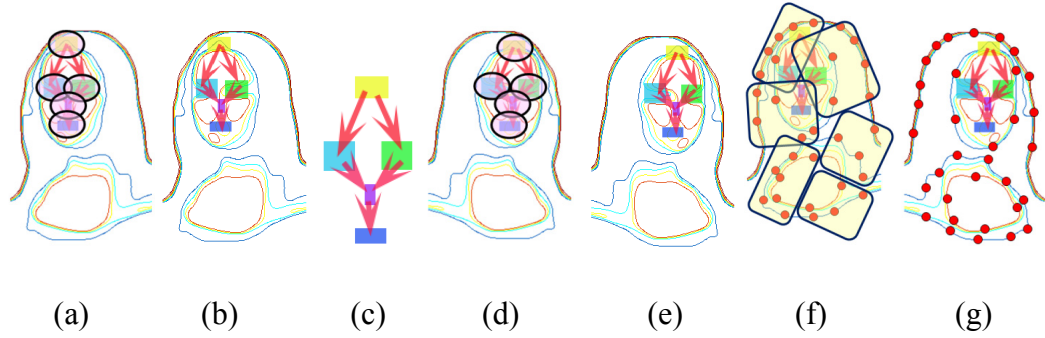


Figure 5.3: Correspondences and initial value determination. (a) Top five features are selected. (b) The structure is constructed from feature blocks. (c) The spatial relation is obtained from first key-pose. (d) (e) The correspondences in the other key-pose are extracted based on the structure of spatial relationship. (f) (g) The samples and correspondences are shown as red dots, and k -means clustering is employed to determine initial value of regression parameters.

TMs are translation, scale, and rotation invariant functions, which are useful for image retrieval and pattern recognition. For each feature block, we compute TMs of the block and compare with the other key-pose by using a window with the same size of the block. Since the minimal difference is found, the correspondences can be obtained. Moreover, the hard constraint is used to refine the correspondences found by TMs. According to the spatial relationship of the feature blocks, some correspondences are interchanged. For example, the correspondences of right eye and left eye found by TMs are interchanged. The correspondences based on the structure of the spatial relationship are constructed.

Owing to predicting the contours of the deformed character by the nonparametric regression model, we sample the contours and obtain the correspondences shown in Figure 5.3 (f) and Figure 5.3 (g) as red dots by using TMs. The contour samples, feature blocks mentioned above, and their correspondences are defined as n pairs of anchor points in the space $U = (\vec{u}_1, \vec{u}'_1, \dots, \vec{u}_n, \vec{u}'_n)$ for the training stage. K -means clustering is used to set the starting center values to the means of the training anchor points. In addition, the covariance for each k -means group shown in Figure 5.3 (f) as the block is computed.

5.4.2 Shape Deforming

The shape deformation of that character is constructed to infer the corresponding displacements of the contours for novel view generation. We define the deformation based on the mapping of n anchor points sampled along the contour of a character. We formulate this problem as regression analysis. Given n pairs of anchor points, we use arbitrary directional ERBFs to predict the contours by interpolating a smooth function. The response $\vec{r} = (r_{\Delta x}, r_{\Delta y})$ is the displacement of an anchor point, and the predictor $\vec{u} = (x, y)$ is the coordinate vector of the anchor point. The resulting ERBF interpolating function is defined as a transformation function $f: \mathfrak{R}^2 \rightarrow \mathfrak{R}$. For k pairs of the means of anchor points obtained by k -means clustering, $f(\cdot)$ contains the radial part $R(\cdot)$ and the affine part $T(\cdot)$. Equation (3.6) and Equation (3.7) can be rewritten as

$$\vec{r}_d = f(\vec{u}_d) + \varepsilon \quad \text{for } d = 1, \dots, n, \quad (5.1)$$

$$f(\vec{u}) = R(\vec{u}) + T(\vec{u}), \quad (5.2)$$

$$R(\vec{u}) = \sum_{j=1}^k \beta_j \eta(\vec{u}, \vec{v}_j), \quad (5.3)$$

$$T(\vec{u}) = M \vec{u}, \quad (5.4)$$

where \vec{r}_d denotes the displacement of the d -th anchor point \vec{u}_d . $f(\cdot)$ is the displacement of either the x -coordinate or the y -coordinate between the correspondences (the given n pairs of anchor points). Additionally, the error ε is assumed to come from a *Normal distribution* $N(0, \tau^2)$, where τ^2 denotes the noise variance.

Moreover, let $\vec{v} = (\mu_x, \mu_y)$ be the center vector of elliptic Gaussian (ERBF). The radial component $R(\cdot)$ is developed as a linear combination of a set of basis functions and their corresponding coefficients. β_j denotes the suitably chosen coefficient of the j -th elliptic Gaussian $\eta(\cdot)$. \vec{v}_j is the related center vector. k is the number of basis functions in the model. Note that $\eta(\cdot)$ is chosen as an arbitrary directional ERBF, as described in Equation (3.1). Recalling Section 5.4.1, we use k -means clustering to set the center vector \vec{v}_j for each k -means group of anchor points. In addition, the corresponding covariance is computed for each group.

Besides, $T(\cdot)$ is a 2D affine transformation, where M_r is a 2×2 real matrix. According to the correspondences of anchor points in feature blocks between the given picture and its reverse, controlling the affine component $T(\cdot)$ is carried out by a least-squares approximation procedure, perhaps using matrix pseudo-inverse techniques. After the affine component has been computed, the radial component $R(\cdot)$ satisfies the following equation:

$$R(\vec{u}) = f(\vec{u}) - T(\vec{u}). \quad (5.5)$$

The estimated weight of the radial component $\hat{\beta}_j$ is determined by solving the linear system.

$$\begin{aligned} & \hat{\beta}_1, \dots, \hat{\beta}_k \\ & = \arg \min_{\beta_1 \dots \beta_k} \sum_{d=1}^n \left\| \sum_{j=1}^k \beta_j \eta(\vec{u}_d, \vec{v}_j) - (f(\vec{u}_d) - T(\vec{u}_d)) \right\|^2. \end{aligned} \quad (5.6)$$

This can be solved by the least-squares normal equations to minimize the sum of the square difference. Equation (5.6) can be rewritten in a matrix form as

$$B = (K^T K)^{-1} K^T (f(\vec{u}) - T(\vec{u})), \quad (5.7)$$

where B is the matrix form of the vector β_j for $j = 1, \dots, k$, K is the matrix form of the vector $\eta(\vec{u}, \vec{v})$, and $(f(\vec{u}) - T(\vec{u}))$ is the matrix form of the vector $(f(\vec{u}) - T(\vec{u}))$.

After the weights $(\hat{\beta}_1, \dots, \hat{\beta}_k)$ are computed in the initial loop, we can compute the residual for nonlinear optimization. Since residuals are recomputed, the residuals update these parameters in the next iteration, which are centers, covariances, and weights, with a gradient descent. Optimization convergence is achieved when the residual is sufficiently small. The whole process is converged completely soon after in several iterative loops. Then the kernel regression model with ERBFs is trained. Note that we can use the model to fit the complete contours of a novel view. We can make predictions of the displacement for each contour point by using Equation (5.1). Furthermore, we use *Catmull-Rom splines* to connect new positions of the contour points. For each in-between frame in temporal domain, the contours of a novel view are synthesized by the model.

5.5 Detail Preservation Using LOESS

In addition to shape deforming for the whole animation process, the details from the character interiors have to be preserved by filling in the color and texture information obtained from the original character in the given image. After synthesizing the contours of the character's deformed shape by using *kernel regression* with ERBFs, LOESS is applied to preserve the details or features of characters. In this section, the detail is preserved within the deformed shape by using LOESS. Then novel views are synthesized completely.

Besides, Equation (5.1) is based on ERBFs and is trained to fit the contours. If we use Equation (5.1) to fill the interiors of the fitted contours with the color and texture information, the filled information will be warped to adapt to the shape of the fitted contours easily without considering the local features of the original character interiors. The result shows the unnatural distortion like the concentric shape effect, as shown in Figure 5.4. Figure 5.4 (a) shows the given key-pose. Another key-pose is shown in Figure 5.4 (d). Figure 5.4 (b) shows the result obtained by using LOESS. Moreover, Figure 5.4 (c) shows another result obtained by using Equation (5.1) directly. Note that there are unnatural distortions shown in the result obtained by using Equation (5.1).

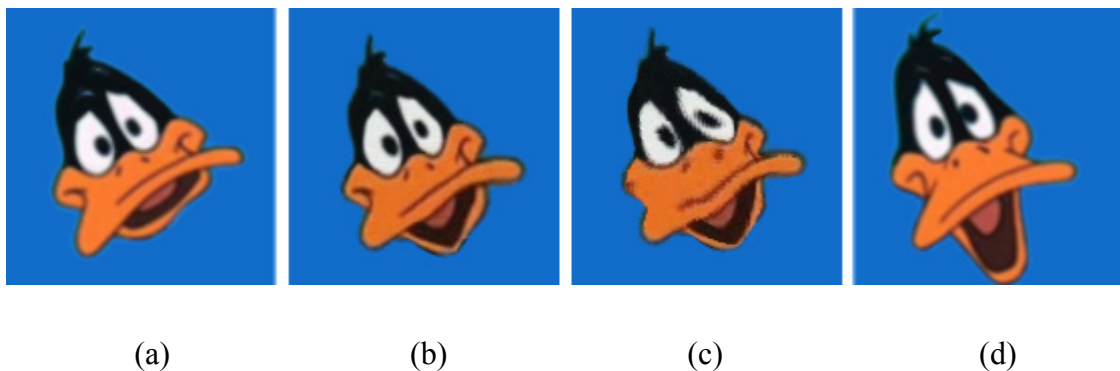


Figure 5.4: Unnatural distortion without detail preserving. (a) (d) The given key-poses. (b) The synthesized in-between obtained by shape deformation and detail preservation. (c) Another result obtained by shape deformation without detail preserving. © Warner Bros. Entertainment Inc.

Recalling Figure 3.3, to implement detail preserving, we sample the original image with a uniform grid (50×50). Given a grid point x_0 enclosed by the contours of a novel view, its filled color or texture information would be controlled by LOESS. In this work, let $x_i = (x_{i,x}, x_{i,y})$ be the i -th point sampled along the contours of the character in the given image, as shown in Figure 3.3 (b). Let $y_i = (y_{i,x}, y_{i,y})$ denote the measurement of the dependent variable representing the new location of the sample point x_i in a synthesized frame in the temporal domain. Suppose that the target coordinate \hat{y}_i is calculated by using Equation (3.9). Before calculating the target coordinate \hat{y}_i , we would estimate the regression coefficient vector $\hat{\zeta}$ in Equation (3.12) by the least-squares normal equations. Moreover, we could calculate the target coordinate \hat{y}_i directly from the closed-form solution as follows.

Recalling Equation (3.9), there is a 2D transformation model in our work. Thus, we have $t_1(x_i) = 1$ for ζ_1 which is a translation coefficient and $t_2(x_i) = x_i$ for ζ_2 which is a rotation coefficient. For brevity, we drop the argument x_0 for the weight $w_i(x_0)$. Given m pairs of (x_i, y_i) , we denote the approximated mean, variance and covariance in the following manner:

$$\mu_x = \frac{\sum_i w_i x_i}{m}, \quad (5.8)$$

$$V_x = \frac{\sum_i w_i (x_i - \mu_x)^2}{m}, \quad (5.9)$$

$$\sigma_{xy}^2 = \frac{\sum_i w_i (x_i - \mu_x)(y_i - \mu_y)}{m}, \quad (5.10)$$

$$\mu_y = \frac{\sum_i w_i y_i}{m}, \quad (5.11)$$

Then the new location of x_0 and the estimated target coordinate \hat{y}_j of the arbitrary new sample x_j can be computed as follows:

$$\hat{y}_j = \mu_y + \frac{\sigma_{xy}^2}{V_x} (x_j - \mu_x). \quad (5.12)$$

In practice, we approximate the character with a uniform grid, as shown in Figure

3.3 (a). We find the new location of each vertex in the grid (each grid point). Then we fill the resulting quad by using bilinear interpolation. Therefore, we can use Equation (5.12) to find the new location of the grid point x_0 and obtain the pixel values for filling in the color and texture information. Then we fill the resulting quad using bilinear interpolation. Note that we would reconstruct the details within the deformed shape with the contours fitted by *kernel regression* with ERBFs via a simple closed-form solution. After shape deforming and detail preserving, novel view generation is carried out. In order to maintain the 3D effect of the new view, it is sometimes combined with backward deformation by using color blending.

5.6 Experimental Results

In the shape deforming stage, the number of basis functions of all the examples fitting the contours was decided by residual analysis. The default setting was eighteen basis functions with better fitting results. Moreover, we applied only our ERBF model on the top five regions in contours to align the significant features in two key-poses or moving templates instead of the entire character because the prediction of unimportant features led to redundancy.

The proposed nonparametric model was implemented on an Intel Core 2 Quad 6600 2.40 GHz CPU and 3 gigabytes main memory that allowed efficient generation of novel views. The complete generation process consisted of two independent steps: shape deforming and detail preserving. Table 5.1 lists the resolutions and executions for the figures shown. Execution time is measured in each step. There is another performance measurement for body movement synthesis, which deformed the shape of the character to synthesize body movement by using the proposed model trained from two given key-poses of a character.

Our experiments were performed on digitized images obtained from “The Adventures of TinTin: The Shooting Star” which was originally produced by Georges Remi (Hergé). The results are presented in Figure 5.1 and Figure 5.5. They show different frames in the original comic, several synthesized frames of characters’ motion,

and the zoom-in views. They were only head movement. A user specification exists by which the head and the body can be segmented. For fitting the contours of the head component, the second key-pose involved reversing the contours of the head component and concatenating with the other contours. A novel view would then be synthesized using the trained model. Another example of Mona Lisa is shown in Figure 5.6.

Table 5.1: Running times of figures for novel view generation and body movement synthesis.

	Novel View Generation				Body Movement Synthesis	
	TinTin (head fitting)	Captain (head fitting)	Snowy (head fitting)	Mona Lisa	Cat (tail moving)	Object with Wood
Figure No.	5.1	5.5	5.5	5.6	5.7	5.9
Resolution	240×502	519×446	169×117	182×268	193×280	189×216
Shape Deforming ~ Training	6487	5876	9420	6434	3673	6002
~ Fitting (Millisecond)	806	721	1382	827	607	744
Detail Preserving ~ Training	3478	3299	3812	3413	2699	3442
~ Fitting (Millisecond)	440	369	745	566	341	421

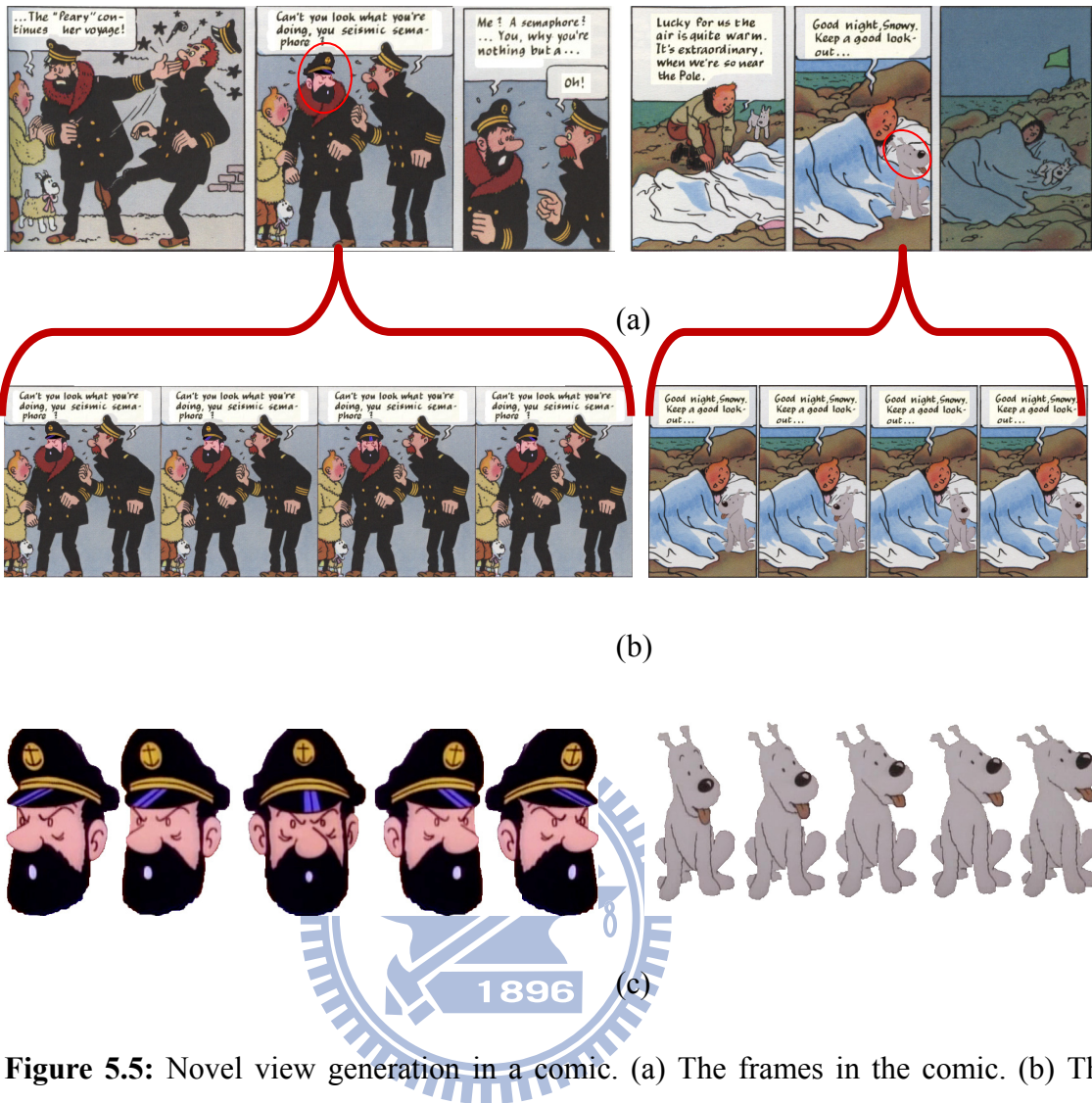


Figure 5.5: Novel view generation in a comic. (a) The frames in the comic. (b) The frames synthesized from a single frame. (c) The zoom-in views of the results. © Georges Remi (Hergé)

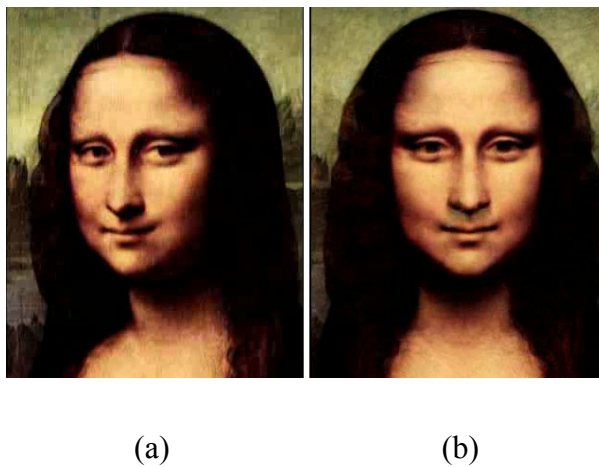


Figure 5.6: Novel view generation from a painting. (a) The picture of Mona Lisa. (b) Novel view generation.

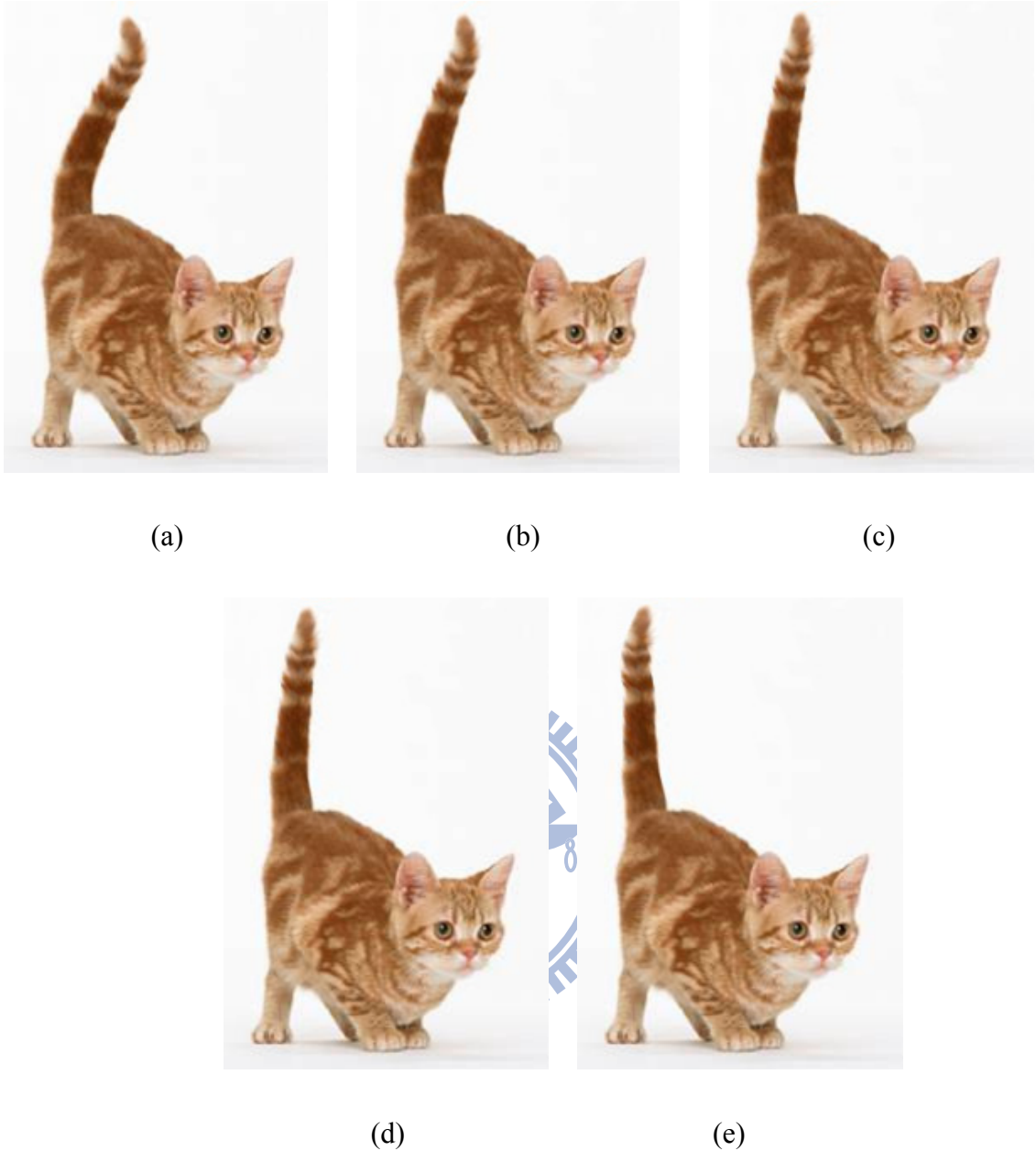


Figure 5.7: Body movement synthesis. (a) (e) Two key-poses of the cat. (b) (c) (d) The synthesized results.

The synthesized results of body movement are shown in Figure 5.7 and Figure 5.9. Body movement synthesis focuses on simulating a single limb's movement of a character. It is different from the proposed limbs movement synthesis. Note that limbs movement synthesis focuses on simulating simultaneously multiple limbs' movements which have motion trajectories respectively. Figure 5.7 (a) shows the original image of a cat representing one key-pose. Another key-pose is shown in Figure 5.7 (e). These two key-poses were employed to generate the tail's movement by the proposed statistical

approaches, as shown in Figure 5.7 (b), Figure 5.7 (c), and Figure 5.7 (d). Note that the pattern of fur shown in Figure 5.7 is preserved by using LOESS.

Since our goal is to do visually plausible 2D character animation, we focus on the qualitative analysis. We provide the results obtained by using *kernel regression* with RBFs, view morphing proposed by Seitz and Dyer [56], and image deformation using the moving least squares method proposed by Schaefer et al. [53], which suffice for direct visual comparisons.

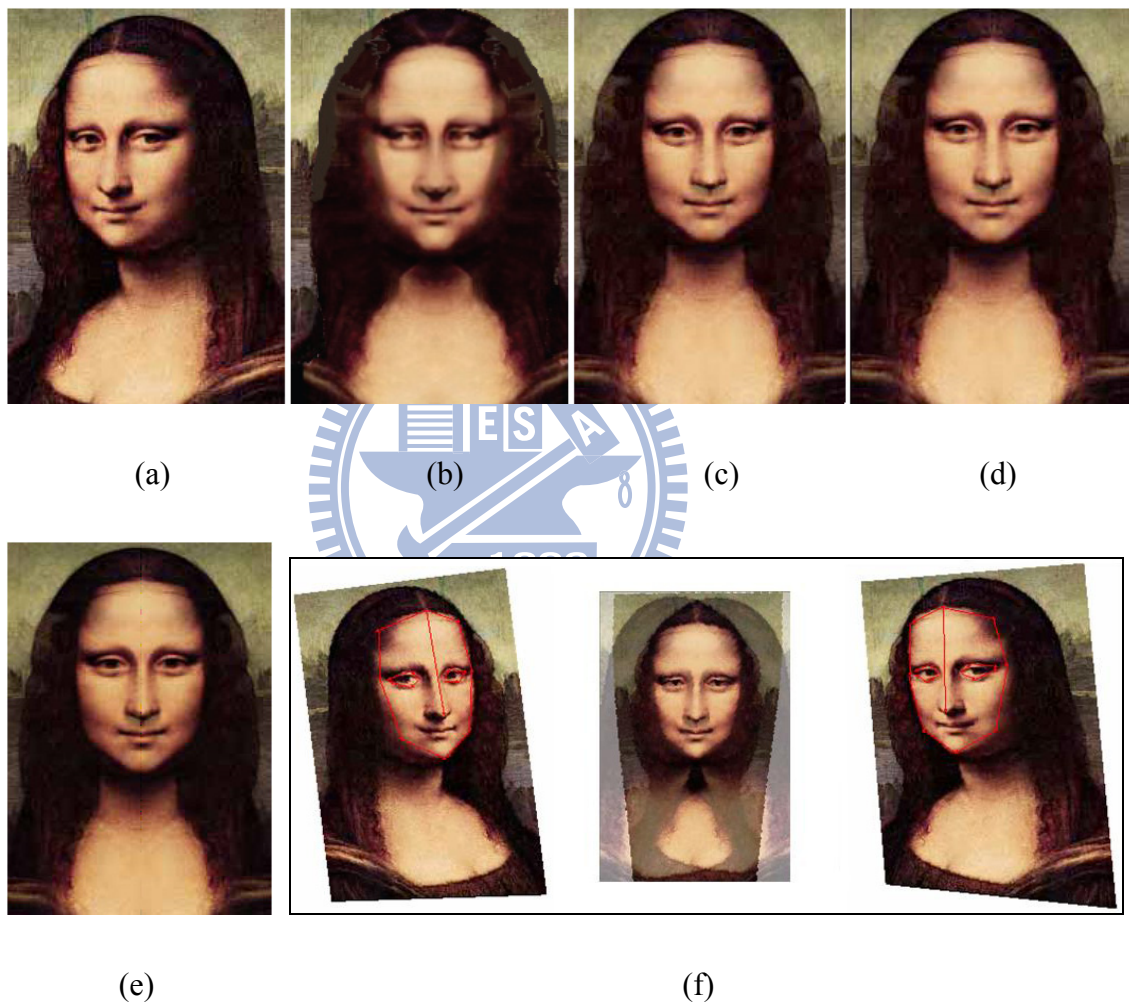


Figure 5.8: Visual comparison of novel view generation. (a) The picture of Mona Lisa. (b) The result obtained by using *kernel regression* with RBFs (18 *radial basis functions*). (c) Another result obtained by using *kernel regression* with RBFs (200 *radial basis functions*). (d) The result obtained by using *kernel regression* with ERBFs (our method with 18 *elliptic radial basis functions*). (e) The result obtained by using view morphing proposed by Seitz and Dyer [56] (f) Ghost occurrence in view morphing without enough correspondences (red lines are specified by users).

Figure 5.8 shows a comparison of novel view generation. Figure 5.8 (b) is obtained by using *kernel regression* with 18 RBFs, and Figure 5.8 (d) is obtained by our model with the same number of ERBFs respectively. Since shape fitting with RBFs contains more unnatural distortions during deformations, ghost effects are observed in the final result with color blending even though feature alignment is achieved in shape deforming. The quality of the final blending result with ERBFs is better. We also provide another result obtained by using *kernel regression* with 89 RBFs without apparent distortion, as shown in Figure 5.8 (c).

As mentioned before, the previous techniques like view morphing and shape deformation may be able to produce good quality results for 2D character animation. However, these techniques needed user intervention. Figure 5.8 (e) provides the comparison with view morphing proposed by Seitz and Dyer [56]. In view morphing, it is necessary to compute an additional estimated fundamental matrix for camera calibration. Further, many users' specifications are required for correspondences. Figure 5.8 (f) shows that the lack of users' specification would create ghost effects because of nonalignment. There are seventeen control lines on the face specified by users. A better result is obtained when more than thirty or forty control lines are specified.

Moreover, Figure 5.9 provides a comparison with the method proposed by Schaefer et al. [53]. Figure 5.9 (a) shows one key-pose of a human-like object. Another key-pose is the reverse of Figure 5.9 (a). The whole body was considered as the unit of movement. The whole body's movement was synthesized, as shown in Figure 5.9 (d). Note that the pattern of wood grain in Figure 5.9 (d) is preserved by using LOESS. Although the method of Schaefer et al. preserved the details of characters, such as wood grain shown in Figure 5.9 (b), the property may lead to an undesired result and unnatural distortions when users specify the moving handles, which exceed the control extent because of the constraint using moving least squares, as shown in Figure 5.9 (c). This man-made situation or interference would not occur in the proposed model. Our model is automatic in shape deformation process of body movement synthesis.

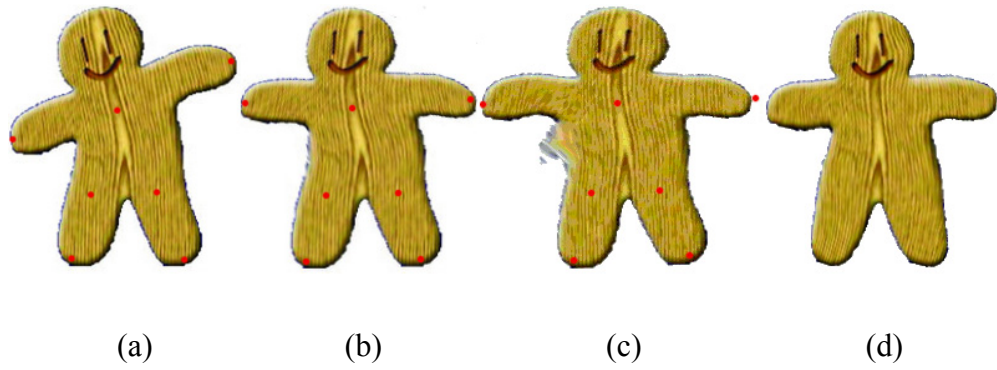


Figure 5.9: Visual comparison of body movement synthesis and detail preservation. (a) The character with handles (red dots). (b) The results created by image deformation using the moving least squares method proposed by Schaefer et al. [53]. (c) The undesired warp occurrence by using the moving least squares method (moving handles exceed the control extent). (d) The same pose with (b) created by using our method.



Chapter 6

Expressive Face with Speech Animation

As discussed in Chapter 5, we have applied *nonparametric regression* to generate a novel view of a character. With the exception of a novel view interpolation, several facial effects observed in 2D character animation, such as eye, nose, and mouth movements, could be created by using *nonparametric regression* mentioned in Section 3.1 and Section 3.2.

Moreover, as we now show, the same structure can be further applied to create lively animated talking faces and synthesize the stylistic variations of facial moods and expressions for generating facial animation. For novel view generation, the proposed nonparametric regression model is trained from one pose of a character and its reverse. Then the trained model is applied to synthesize the smooth transition between these two poses. For facial animation, the trained nonparametric model is employed to generate synchronized lips movement and drive the stochastic process for facial features movements by giving any speech data and the relative moving templates. Note that the statistical model is trained to fit the shape and detail of the character between these moving templates.

6.1 Character and Features Extraction

In order to reduce color complexity and the effects of the background upon deformations, we use the bilateral filter and level-set-based *GrabCut* mentioned previously to simplify and extract characters. Similar regions are extracted by the level

set method. Note that the facial features of the extracted character are obtained simultaneously by the level set method. By moving the facial features, we simulate the dynamics of the features to synthesize different expressions, such as wink, anger, or happy. Furthermore, an expressive facial animation with lips movement from speech could be simulated.

6.2 Speech Animation

After the pre-process of character and features extraction for reducing the effects of the redundancies, we generate the lip synchronization by a series of speech data.

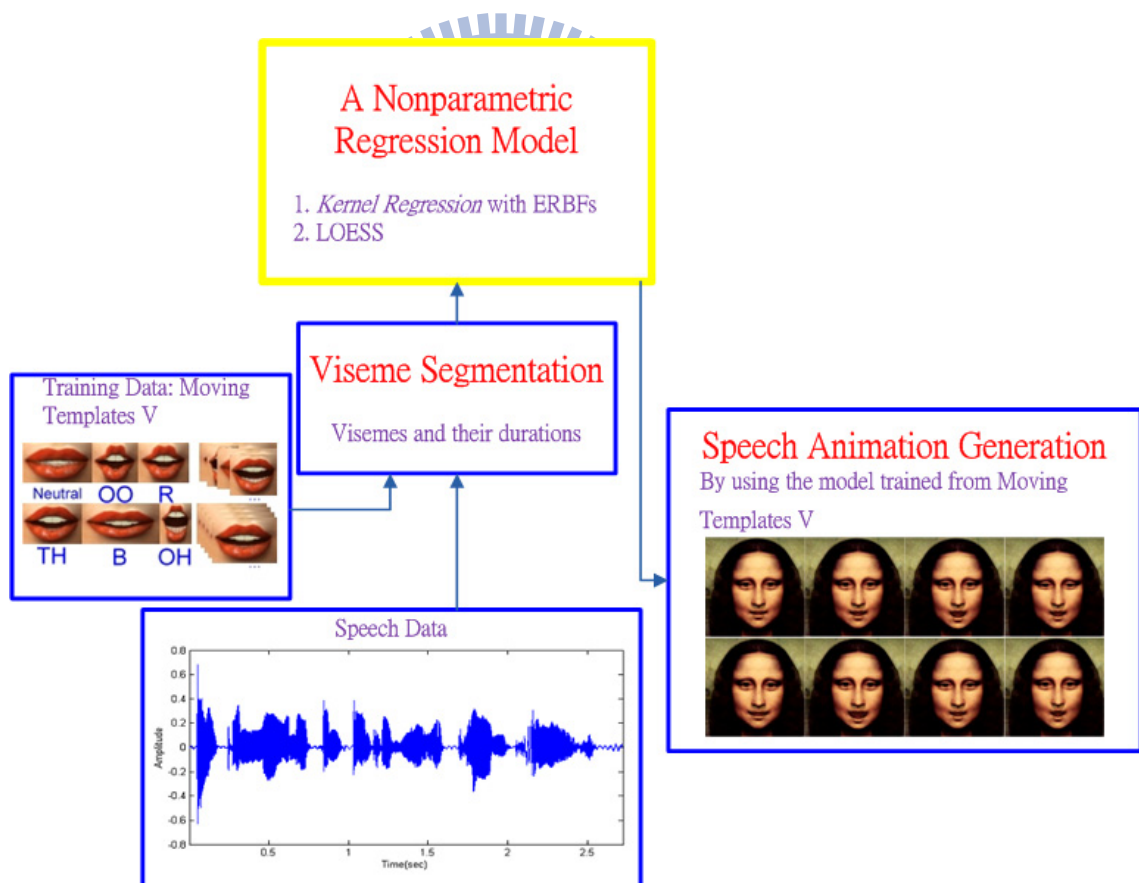


Figure 6.1: The overview of speech animation generation.

6.2.1 Algorithm Overview

In Figure 6.1, the outline reflects the structure of our proposed method for deforming mouth shape synchronized with input speech. Considering Figure 6.1, we briefly describe our method in the following paragraphs.

1. Training Data: For model learning, according to different multimedia applications, the nonparametric regression model is trained from different training data sets. As mentioned before, the model is trained from two key-poses of a character for novel view generation. To generate speech animation, the training data set consists of two extreme templates in moving templates V for visemes synthesis, which are the neutral mouth shape and the mouth shape with vowel /o/. The moving templates are the base target positions of the facial features to be animated. The positions of mouth shape are recorded in moving templates V , as shown in Figure 6.1.

2. Viseme Segmentation: In viseme synthesis, the goal is to model the correspondence between mouth shape variation and speech. Viseme segmentation, which means to determine all visemes and their durations, is done from the speech data. Note that it is to align phoneme labels to the audio stream, and use this information to label the corresponding lips movement.

3. Speech Animation Generation: For generating the lip-synch animation, we collect moving templates V and the speech data, whereas the speech data is the voice data. After viseme segmentation, we would convert to the corresponding mouth shape and synthesize speech animation by using the trained nonparametric regression model.

6.2.2 Viseme Synthesis

For viseme synthesis, viseme segmentation of the speech data is performed to determine all visemes and their durations. First of all, we employ a *Hidden Markov Model* (HMM [45]) speech recognizer, which is the high-precision speech recognition software in noisy environments, to analyze the given speech data. In practice, HMM speech recognizer is used to obtain the phoneme segments called phoneme samples, as shown in Figure 6.2.

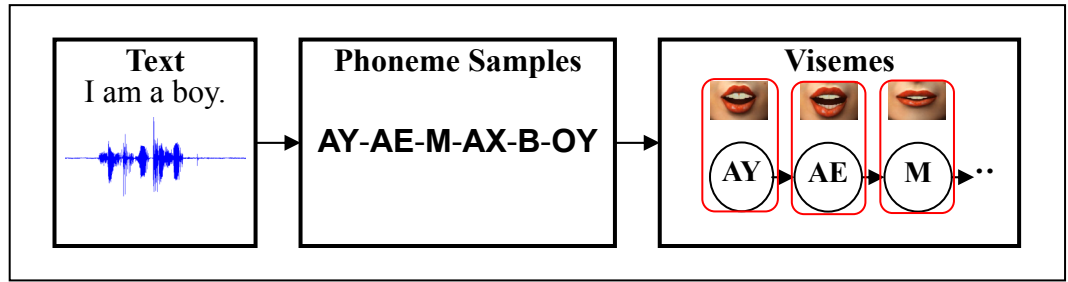


Figure 6.2: Viseme segmentation of the given speech data.

Table 6.1: Conversion table from phoneme to mouth shape and the corresponding phonetic alphabet.

Mouth Shape (Viseme) No.	Phoneme Samples	Phonetic Alphabet
1	AA, AE, AX, HH, SIL, and Y	/a/, /æ/, /ə/, /h/, /sil/, and /j/
2	B, M, and P	/b/, /m/, and /p/
3	CH, J, and SH	/tʃ/, /dʒ/, and /ʃ/
4	OO, OY, W, and UH	/u/, /ɔɪ/, /w/, and /U/
5	AY, EY, and ER	/aɪ/, /e/, and /ɜ/
6	F and V	/f/ and /v/
7	IH and IY	/ɪ/ and /i/
8	G and K	/g/ and /k/
9	N and NG	/n/, and /ŋ/
10	OH	/o/
11	R	/r/
12	S, TS, and Z	/s/, /ts/, and /z/
13	D, L, and T	/d/, /l/, and /t/
14	TH	/θ/

Besides, we design fifteen templates in moving templates V for viseme synthesis, which are fourteen common mouth shapes with their relative visemes and a neutral mouth shape for all other visemes, as shown in Figure 6.1. The templates in moving templates V are employed to record the positions of anchor points sampled on the contour of the lips, as shown in Figure 6.4. These anchor points are obtained from the

extracted features found by using the level set method, as mentioned before. Then we construct a phoneme–viseme mapping table by using a simple table lookup method [77] to find the relative moving template (viseme) of a phoneme sample in moving templates V directly, as shown in Figure 6.2. Table 6.1 shows the conversion from the phoneme to the mouth shape. For two continuous phoneme samples with the same neutral mouth shapes, one of the samples is redefined as the mouth shape with vowel /o/.

Given the positions of anchor points sampled from the contours of the extracted lips, moving templates V with their relative visemes, and the input phoneme samples, we could create the lip-synch animation. Next, we predict the motion of the character by using *nonparametric regression*: First, *kernel regression* with ERBFs is employed to fit the contour of a deformed shape. Then we preserve the details of the deformed shape by LOESS. We again consider the method to synthesize visemes.

Note that the nonparametric regression model consisting of *kernel regression* with ERBFs and LOESS is trained for viseme synthesis. In the shape deforming stage, the kernel regression model with ERBFs is trained. We would train the model with the training data, that is, n pairs of anchor points recorded in two extreme moving templates, which are the neutral mouth shape and the mouth shape with vowel /o/ in V , are the prior use of a set of kernel functions. In other words, these anchor points and their corresponding displacements are used to train the kernel regression model with ERBFs. Note that the regression coefficients are estimated by the least-squares normal equations. The trained model is applied to fit the variations of the mouth shape between arbitrary two visemes of the phoneme samples using the corresponding moving templates in V for lip-synch animation generation. Finally, the similar process to preserve details within the target mouth is carried out by LOESS.

6.3 Viseme Synthesis with Expressive Face

In addition to speech animation, we could further synthesize an expressive face of a character. By moving the facial features obtained from the structure of the spatial relation, which we constructed before, we simulate the dynamics of these features to synthesize different expressions. We could enhance the expression by shaking the

shoulders or wagging the character’s head. Even we could further retarget the expression onto another character, as illustrated in Figure 6.3. After the pre-process of character and features extraction, two emotional states in moving templates D are provided for facial expression simulation.

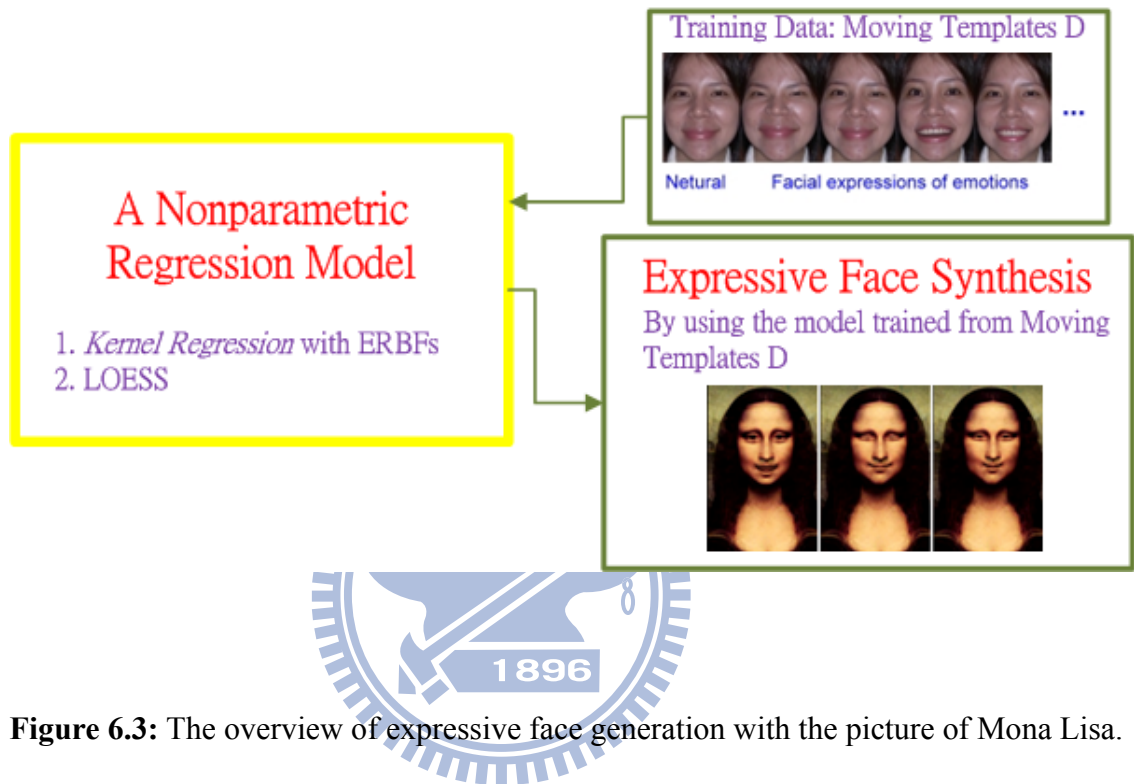


Figure 6.3: The overview of expressive face generation with the picture of Mona Lisa.

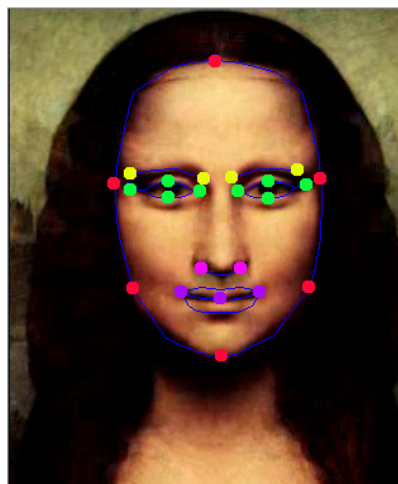


Figure 6.4: Groupings of facial shape and features labeled as anchor points and relative curves.

We collect moving templates D for facial mood and expression simulation, which consist of a neutral expression and several common expressions, such as happy, angry, sad, fear, surprise, and wink. Note that the moving templates are the base target positions of the facial features to be animated. Moving templates D focuses on the positions of all facial features. The templates in moving templates D are employed to record the positions of anchor points sampled and grouped from the contours of the extracted features, such as facial shape, eyebrows, eyes, nose, and lips, as shown in Figure 6.4. Given moving templates D , we can use *nonparametric regression* to synthesize facial expression of any emotional state through moving templates D . Furthermore, an expressive face with speech animation is created by combining with the lip-synch process described in Section 6.2.

As discussed in last section, the nonparametric regression model consisting of *kernel regression* with ERBFs and LOESS is similarly trained. In the shape deforming stage, the kernel regression model with ERBFs is trained. For facial expression simulation, note that we would train the model with a different training data, that is, n pairs of anchor points recorded in two moving templates which are the neutral expression and the specific emotional state in D . Then the trained model is applied to fit the movements of facial features between these two emotional states. Then the similar process to preserve details within the target face is carried out by LOESS.

Furthermore, an expressive face with speech animation could be created by combining with the lip-synch process simultaneously. Before the model with LOESS is trained for detail preserving, we would find the positions of anchor points for facial features in the target expressive face composed of specific emotion and visemes. Actually, we would like to identify facial expression simulation independently of the content (utterance of sentences and the corresponding lips movement). The target animated expressive face with lip-synch FE can be represented as follows:

$$FE = AT(\mathbb{N} + F_{nl} + F_l) = AT\left(\mathbb{N} + \sum_i D_i + \gamma D_i + (1 - \gamma)L\right), \quad (6.1)$$

where $AT(\cdot)$ is a 2D affine transformation of the head movements. The head movements are specified by users. \mathbb{N} is a neutral expression. F_l (lips) and F_{nl} (facial features except lips) are the movements of facial features. Note that F_l and F_{nl} are displacements from the neutral expression. So $\mathbb{N} + F_{nl} + F_l$ represents an individual facial expression in

a certain emotional state and viseme.

Instead of using F_l and F_{nl} , D_l is applied for the detailed movements of lips in a specific emotional state, and D_i for each i represents the movement of an individual facial feature except lips. Note that five non-overlapping features are identified for a specific emotional state, such as left eyebrow, right eyebrow, left eye, right eye, and nose. D_l and D_i are obtained through the fitted movements of facial features from the neutral expression to the specific expression. Besides, L is lips movement synchronized with the input speech. L is obtained through the fitted variations of mouth shape adapting to the audio stream. For the final mouth shape, a blending weight γ is considered to generate an expressive face with lip synchronization.

After finding the positions of facial features in the target expressive face, the model with LOESS is trained by these facial features. Then the trained model is employed to preserve details within the target expressive face. For example, after the mouth shape is obtained, the mouth cavity, which is the region between the upper lip and lower lip, is filled in the color and texture information obtained from the original character in the given image by using the trained model. Note that we use the color and texture information inside the mouth of the character to make the character appear realistic while talking. Thus, the character with a lively animated talking face is created.

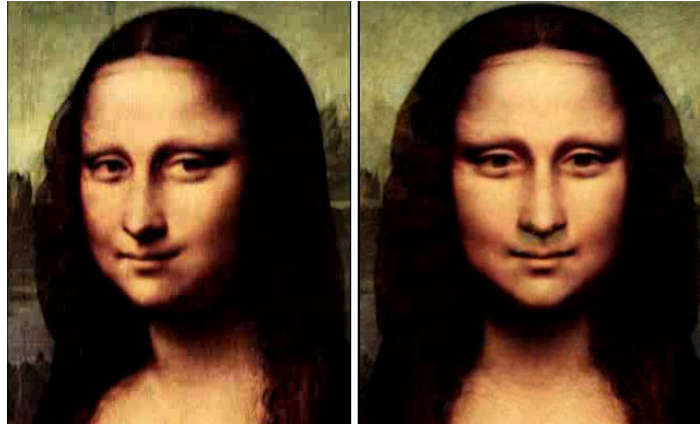
6.4 Experimental Results

The proposed nonparametric model was implemented on an Intel Core 2 Quad 6600 2.40 GHz CPU and 3 gigabytes main memory that allowed efficient generation of an expressive talking face. The complete generation process consisted of two independent steps: shape deforming and detail preserving. Table 6.2 lists the resolutions and executions for the figures shown. Execution time is measured in each step. We are interested in extending our concept to facial expression and viseme synthesis. Several facial effects observed in 2D character animation, such as eye, nose, and mouth movements, could be created, as shown in Figure 6.5, Figure 6.6, Figure 6.7, and Figure 6.8. By moving the facial features, we simulated the dynamics of the features to synthesize different expressions synchronized with the speech. Figure 6.5 shows the

visual result combining with novel view generation. Figure 6.5 (a) is the original picture of Mona Lisa. A novel view of Mona Lisa is shown in Figure 6.5 (b). For expression synthesis, Figure 6.5 (c) shows the synthesized facial expressions (smiley). Moreover, the lively talking face of Mona Lisa simulated by viseme and expression synthesis is shown in Figure 6.5 (d).

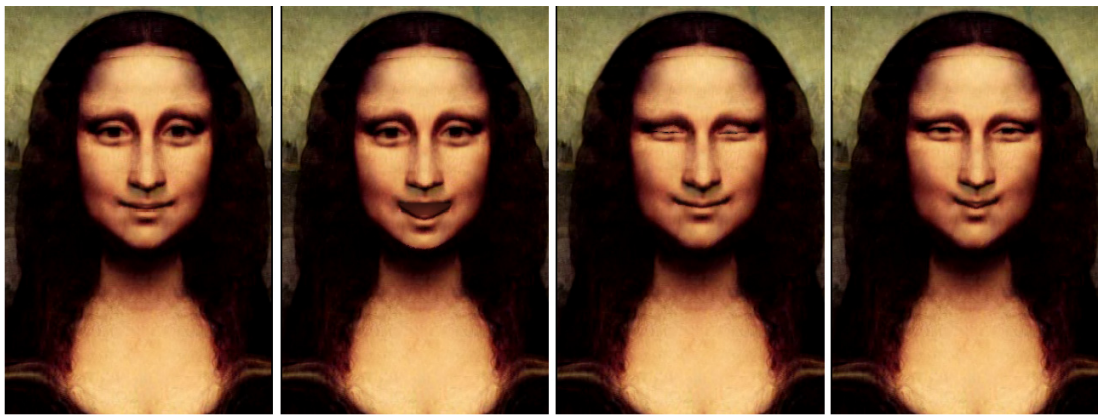
Table 6.2: Running times of figures for viseme and facial expression synthesis.

	Viseme Synthesis		Facial Expression Synthesis	
Figure Name	Mona Lisa	Lips	Mona Lisa	Self-Portrait
Figure No.	6.5	6.6	6.5	6.7
Resolution	182×268	565×281	182×268	505×582
Shape Deforming ~ Training	10432	12017	10860	14093
~ Fitting (Millisecond)	2009	2471	2204	4272
Detail Preserving ~ Training	3414	4294	3298	4836
~ Fitting (Millisecond)	1166	1707	1001	1864



(a)

(b)



(c)



(d)

Figure 6.5: 2D character animation of Mona Lisa. (a) The picture of Mona Lisa. (b) A novel view. (c) Different kinds of the smiles. (d) The lively talking face simulation.

Another synthesized lip-synch example is shown in Figure 6.6 for five vowels. Moreover, we used the model to predict other emotional states of another character in Vincent van Gogh' Self-Portrait for expression synthesis, as shown in Figure 6.7.

Since our goal is to do visually plausible character animation for novel view, viseme, and facial expression synthesis, we focus on the qualitative analysis. We provide a direct visual comparison with the results obtained by animating still images using the path-based method proposed by Mahajan et al. [39]. Figure 6.8 and Figure 6.9 show comparisons with the path-based method. Their method was based on an inverse optical flow and preserved the spatial frequencies of the input images. However, as mentioned before, the disparity or the motion that they could handle between the images was limited. In [39], the disparity or motion between the images was about 30 pixels. Given an image revealed in Figure 6.8 (a), the expressions (staring and smiley) were synthesized by using our method, as shown in Figure 6.8 (b) and Figure 6.8 (c). Figure 6.8 (d) shows the result obtained by using the path-based method.

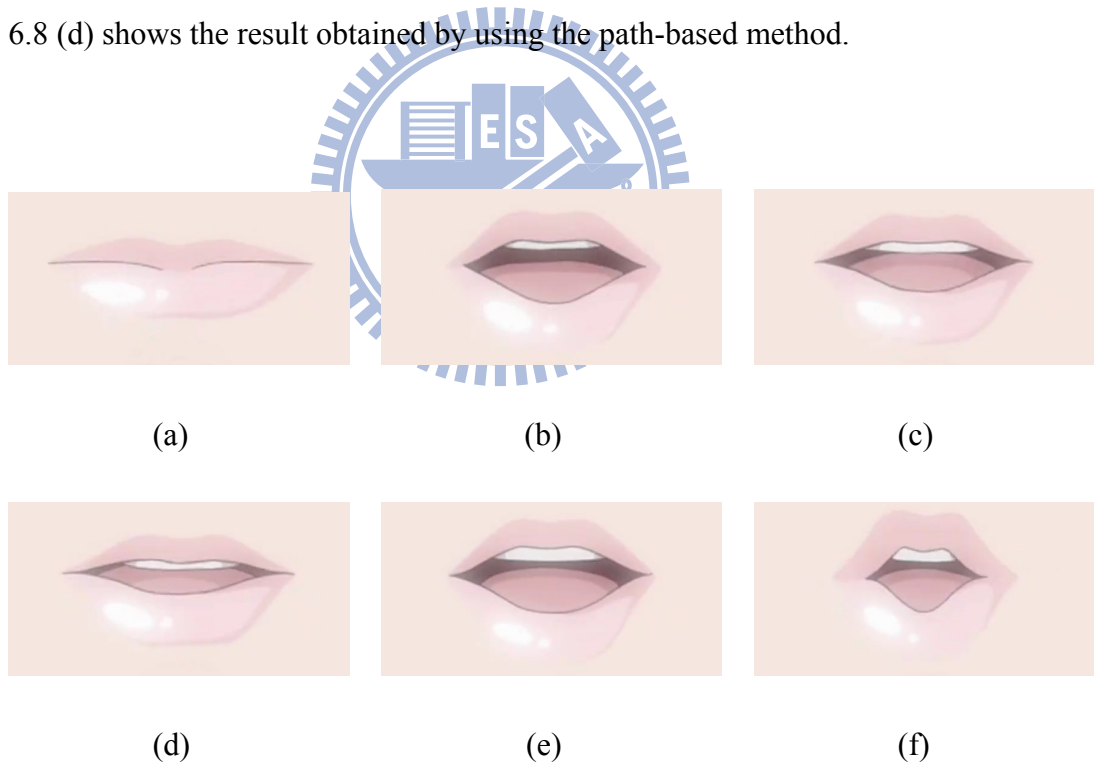
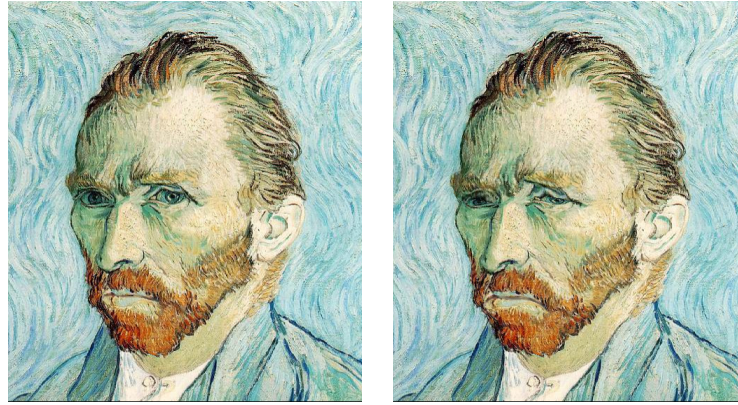
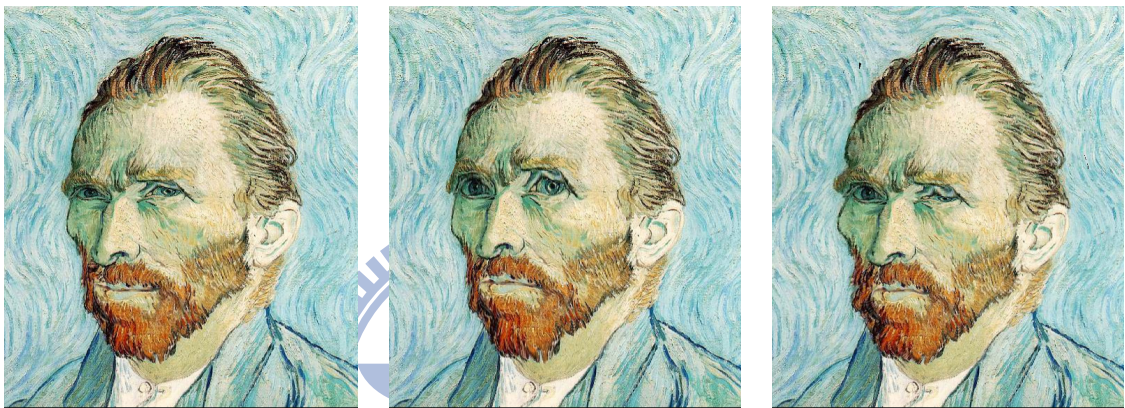


Figure 6.6: Viseme synthesis for five vowels. (a) The original mouth shape. (b) /a/. (c) /e/. (d) /i/. (e) /o/. (f) /u/.



(a)

(b)



(c)

(d)

(e)

Figure 6.7: Character animation with expression synthesis. (a) The original expression in Vincent van Gogh’s Self-Portrait. (b) Sad. (c) Smiley. (d) Staring. (e) Winking.

Moreover, given another example of the girl revealed in Figure 6.8 (e), facial expressions were synthesized by using our method, as shown in Figure 6.8 (f) and Figure 6.8 (g). The result obtained by using the path-based method is shown in Figure 6.8 (h). Furthermore, given Figure 6.8 (e) and its reverse, the novel views shown in Figure 6.8 (i) and Figure 6.8 (k) were generated by using our method. Note that the maximum disparity or motion between two images is 70 pixels. A few shades on the face are due to the fact that the inconsistent illumination or brightness on the face in the input image and the aforementioned color blending we used to maintain the 3D effect of the synthesized view. Another smiley example in the synthesized view is shown in Figure 6.8 (j).

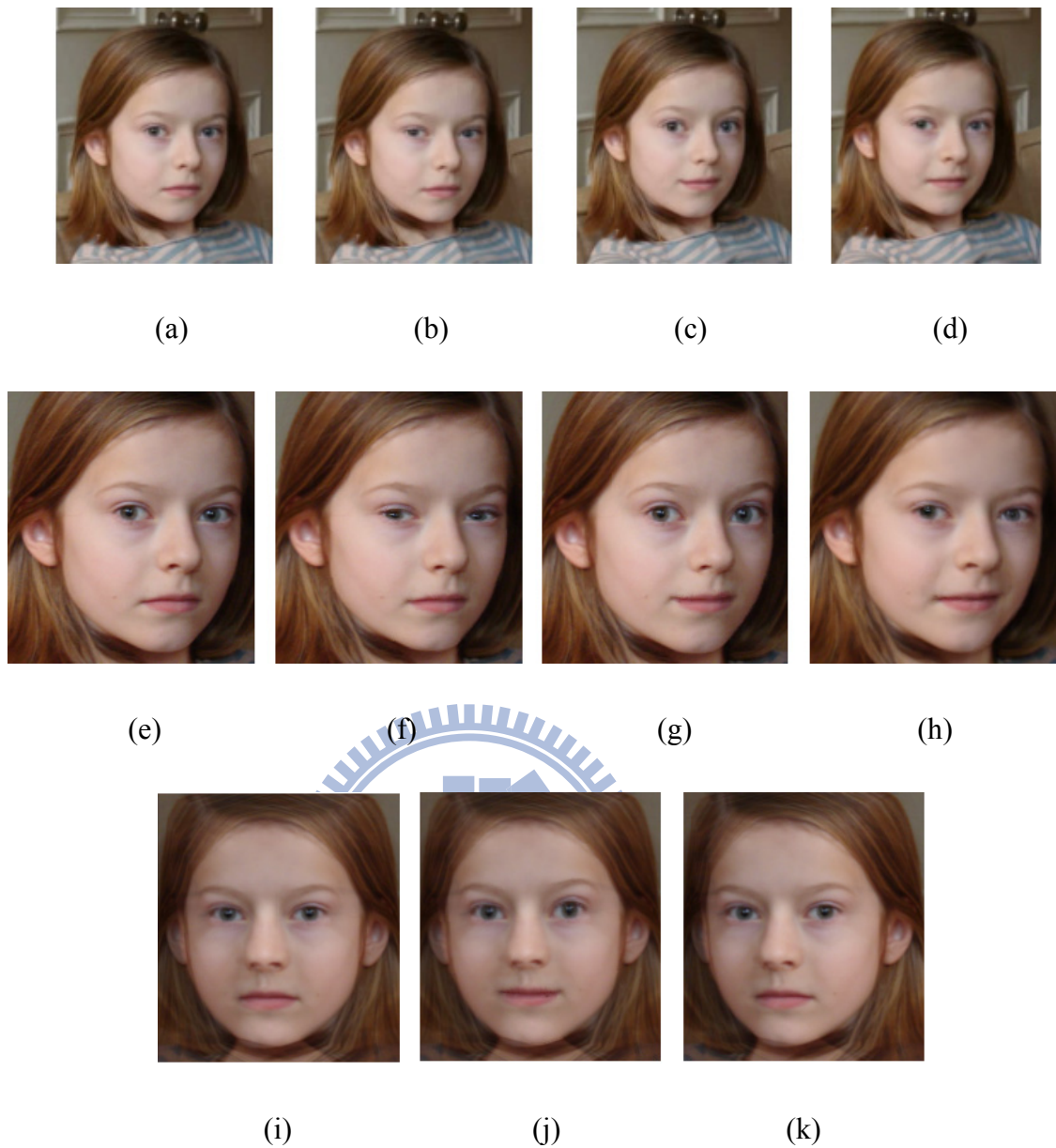


Figure 6.8: Visual comparison of expression synthesis. (a) The input image obtained from [39]. (b) (c) The face emotions synthesized by using our method (staring and smiley). (d) The result obtained by using the path-based method proposed by Mahajan et al. [39] (smiley). (e) Another given image (the zoom-in view of the girl’s face). (f) (g) The face emotions synthesized by using our method. (d) The result obtained by using the path-based method. (i) (j) Different facial moods in a novel view synthesized by using our method (neutral and smiley). (k) Another novel view synthesized by using our method.

We also provide a comparison with an example of a yawning cat obtained by using the path-based method proposed by Mahajan et al. [39]. The original picture of a cat is

shown in Figure 6.9 (a). Given the picture, the selected frames of the expression (wink) shown in Figure 6.9 (b) and Figure 6.9 (c) were generated by using our method. Figure 6.9 (d) shows the result obtained from [39]. Another example is the zoom-in view of the cat’s face, as shown in Figure 6.9 (e). Given the face, Figure 6.9 (f) and Figure (g) were synthesized by using our method. Figure 6.9 (h) shows the result obtained by using the path-based method. Note that the details are preserved explicitly by using LOESS, such as fur, whiskers, and tongue of the cat.

In general, we find that our method provides visually superior shape deforming or detail preserving with minimal artifacts in most cases. On the other hand, our method does not suffer from serious ghosting, blurring artifact, or unnatural warping, which exists in other methods, such as the path-based method [39], view morphing [56], and the moving least squares method [53] compared in Section 5.6. Moreover, our proposed synthesis process does not require user-specified correspondences. Considering view morphing and the moving least squares method, they require users’ intervention for correspondences or handling the deformation.

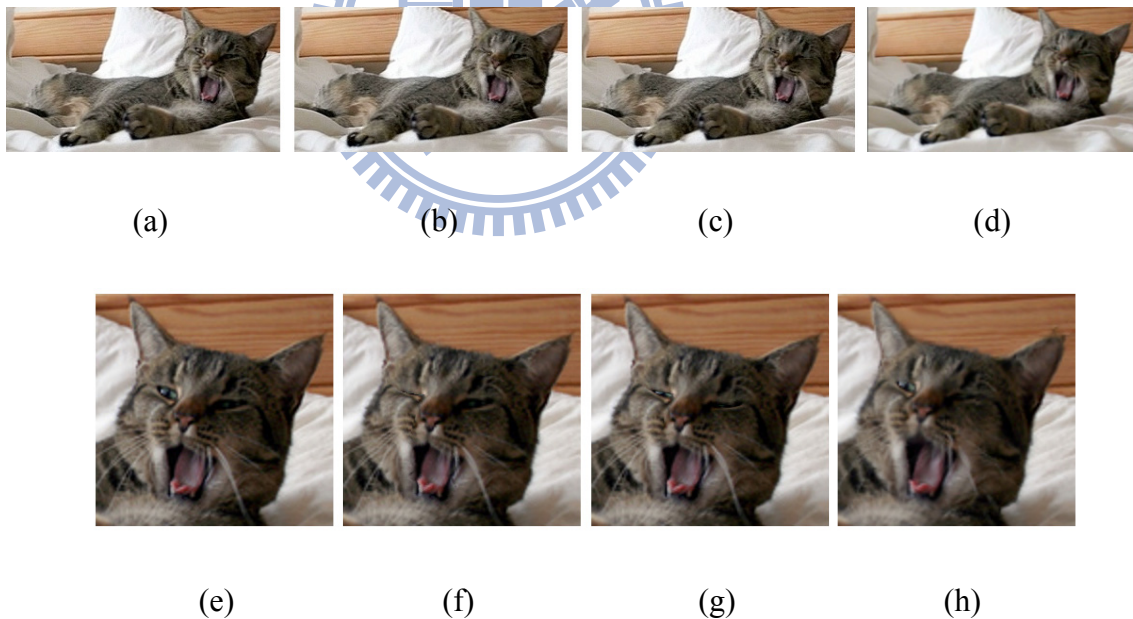


Figure 6.9: Another comparison of expression synthesis. (a) The input images obtained from [39]. (b) (c) The selected frames of a yawning cat synthesized by using our method. (d) The result obtained by using the path-based method proposed by Mahajan et al. [39]. (e) Another given image (the zoom-in view of the cat’s face). (f) (g) The expressive faces of the cat synthesized by using our method. (h) The result obtained by using the path-based method.

Chapter 7

Limbs Movement Synthesis

In a comic, a movement may be described by a series of non-continuous poses in a sequence of contiguous frames. While two contiguous frames represent two adjacent time-sliced poses in a continuous movement only, they cannot represent the movement completely. Thus, generating a natural-looking animation for limbs movement is still a major challenge in computer graphics. We synthesize limbs movements simultaneously according to still comic frames or low-frame-rate video frames. Basically, the movement is considered as an essential 3D transformation problem, which consists of a 2D spatial displacement and a 1D shift in time. So we construct a time series model to synthesize limbs movements.

In this chapter, we propose a model to analyze time series data of a character's motion by using a nonparametric Bayesian approach with ERBF kernel. Then we can automatically generate a sequence of motions by using the constructed time series model, which is adopted to fit the motion trajectories of a character.

7.1 Statistic-based Movement Synthesis

An approach of human or human-like subject movement synthesis is the constraint-based motion synthesis [12]. It was formulated in a *maximum-a-posterior* (MAP) framework. This statistical framework is approximated by only using the likelihood and prior terms, which is equivalent to the minimization of an error function. However, the framework only correlates with the training data. It does not necessarily

give a small error with respect to new data.

We also adopt a statistical method to synthesize the motion. First, we simulate key-motions of a character between two contiguous frames in a comic or a low-frame-rate video by using *kernel regression* with ERBFs mentioned in Section 3.1. A key-motion is defined as the contour of an in-between pose between the poses of a character in two contiguous frames of a comic. Note that ERBF kernel is suitable to perform scattered-data interpolation and is applicable to fit the human-like shape.

Besides, we obtain the regression parameters suitable for the current motion of a character by *Bayesian inference*, which is based on the RJMCMC method. Note that RJMCMC generates a sequence or a chain of samples, as mentioned summarily in Section 3.3. Apart from the initial sample, each sample is derived from the previous sample, which allows the algorithm to find coefficients or parameters that satisfy the situation of current regression model. We do not use least-square error to find the regression coefficients or parameters because the least-square method might lead to local minimization.

However, these simulated key-motions are described discretely in the temporal domain. For generating a smooth and continuous character animation, we synthesize the contours of a character's motion following the motion trajectory that is obtained by the proposed time series model called by the Bayesian version of ARMA (BARMA). BARMA integrates ARMA with a nonparametric Bayesian approach that is based on *kernel regression* with ERBFs and RJMCMC-based model estimation. Note that the model is trained from the key-motions.

After generating a sequence of motion, a local-fitting methodology is also applied to preserve important features within contours. LOESS, as mentioned in Section 3.2, is a way of estimating the regression surface through a multivariate smoothing procedure by fitting a function of independent variables locally, which maintains features invariant during deformations without unnatural distortion. Furthermore, the Bayesian version of LOESS (BLOESS) is proposed to improve meaningful regressions by using *Bayesian inference* to infer regression coefficients in LOESS.

7.2 Algorithm Overview

The proposed approach for generating 2D character animation from between two contiguous poses consists of the following four components: Shape Structure, Bayesian Regression, Time Series, and Detail Preservation.

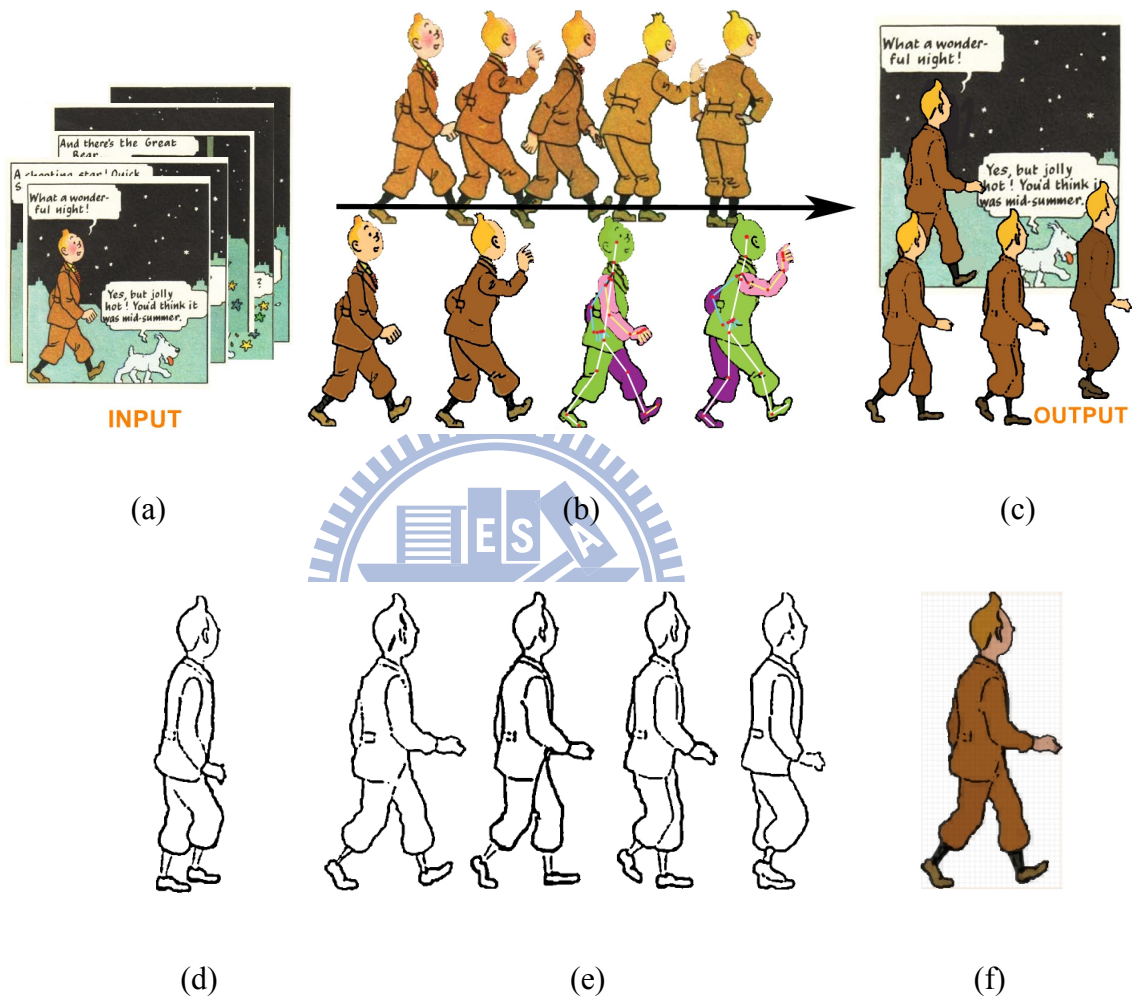


Figure 7.1: The overview of our approach for synthesizing limbs movements. (a) Considering several poses in consecutive frames of the source comic, (b) the character is extracted by level-set-based *GrabCut*. We construct the shape structure and refine it (here: the same color represents as the same level in the shape structure). (d) The key-motion is synthesized by Bayesian regression. (e) Then the time series is estimated to synthesize the whole motion. (f) The intermediate color is overlaid on the deformed contour by BLOESS. (c) The character animation in a comic is generated by using our method. © Georges Remi (Hergé)

1. Shape Structure: A hard character matte is obtained by using level-set-based *GrabCut*, as shown in Figure 7.1 (b). The foreground and background are adequately separated. The moving components are found simultaneously. Note that a moving component denotes the basic unit of a character's motion. To create convincing animations of a 2D character, its shape needs to be deformed plausibly, while maintaining the effort for generating animations on three generic body layers. These body layers denote the topological changes of a generalized character model for different camera viewpoints. Note that we abstract the character in order to construct the shape structure by using the bilateral filter, as mentioned in Chapter 4. The skeleton of a character is specified by using a predefined 2D skeleton structure.

2. Bayesian Regression: Anchor points are first sampled along the contours of a character in a frame. The shape deformation function between these samples and correspondences in another frame is trained by using Bayesian regression, which is based on *Bayesian inference of kernel regression*. Note that the ERBF kernel is adopted for regression analysis. Moreover, RJMCMC is applied to infer the optimized regression coefficients and parameters. The deformation function is used to fit the deformed contours for interpolating key-motions. For instance, Figure 7.1 (d) shows a key-motion obtained by using the deformation function. The function is trained from two poses of a character shown in Figure 7.1 (b) (below). Key-motions are applied to construct the time series model further.

3. Time Series: ARMA is a useful and stable time series model, as mentioned in Section 3.4. Given the key-motions, the entire limb movement is synthesized by using BARMA, which integrates Bayesian estimation with ARMA. BARMA is applied to predict the motion trajectories between the key-motions.

4. Detail Preservation: The trajectories are applied to fit contours for synthesizing a series of motions, as shown in Figure 7.1 (e). Then the details of character are preserved by using BLOESS. In other words, BLOESS improves meaningful regressions by using *Bayesian inference* during a LOESS prediction for filling in the color and texture information obtained from the original character, as shown in Figure 7.1 (f). The limbs movements are synthesized in accordance with the previously fitted contours and details, as shown in Figure 7.1 (c).

7.3 Bayesian-based Limbs Movement Synthesis

In this section, we explain our method in detail. The shape structure is constructed first. Then Bayesian regression with ERBF kernel and RJMCMC is applied for key-motions generations. Next, the time series model BARMA is constructed and used to estimate the motion trajectory. Finally, the movements of a character's limbs are synthesized by using BARMA and BLOESS.

7.3.1 Shape Structure

In this stage, similar regions are extracted by approximating the dynamics of moving curves. This method is known as the level set method mentioned in Section 5.3. After abstracting the character by using the bilateral filter, we apply the level set method to segment regions with the similar color distribution. Next, the bounding box of these regions is applied for *GrabCut* mentioned in Section 5.3 to separate the foreground and background. Furthermore, the moving components of a character are found simultaneously. Besides, the skeleton of each moving component is obtained by using morphology-based operations [32]. Given a predefined human skeleton structure, the skeleton of a character is specified by moving the bones of that predefined skeleton to align to the bones of the obtained skeletons of moving components. Furthermore, we can refine the skeletal bones and joints in occluded regions manually.

These moving components are further partitioned into three layers manually while animating characters from a side view. For instance, an animation might involve one layer for the foremost arm, one for the body and the foremost leg, and one for the remaining arm and leg. Moreover, these layers cannot move independently. They should be stitched together to convey the impression of a connected body when animating the character. Hence, every layer is composed of moving components, skeletal bones, and joints. Different layers are linked by the shared skeletal joints.

7.3.2 Point-to-point Correspondences

Basically, a shape deformation of a character is constructed for motion synthesis. Before defining the deformation, the point-to-point correspondences of anchor points are obtained. As illustrated in Figure 7.2, a character raises up his arm. We can construct the point-to-point correspondences from these bones (blue lines) and joints (purple dots). The anchor points are sampled along the contours of the character randomly. For example, there is a point u_i sampled along the contour of a right arm randomly. u_i is called an anchor point. First, we find the projection J_m of the anchor point u_i on the line segment $\overline{J_1J_2}$ or the extended line of $\overline{J_1J_2}$. Based on the predefined skeleton structure, the skeleton correspondence between two frames is obtained. Note that the corresponding joints J_1' and J_2' of J_1 and J_2 are known. According to the ratio of $\frac{J_1J_m}{J_1J_2}$, we can find the point J_m' which satisfies the constraint that $\frac{J_1'J_m'}{J_1'J_2'} = \frac{J_1J_m}{J_1J_2}$. Then we compute the normal vector on the point J_m' and find the intersection of the normal vector and the contour of the right arm. The intersection point u_i' is the correspondence of the anchor point u_i . Thus, we can obtain n anchor points sampled along the contours in a frame and their correspondences in another frame.

Then we define the deformation based on the mapping of n anchor points sampled along the contour of a character and the relative correspondences. The shape deformation function between these samples and correspondences in another frame is trained by using *kernel regression* with ERBFs and *Bayesian inference* with RJMCMC mentioned. Note that ERBF kernel is adopted for regression analysis, and RJMCMC is applied to estimate the optimized regression coefficients and parameters suitable for the current motion of a character. The deformation function is used to fit the deformed contours for interpolating key-motions.

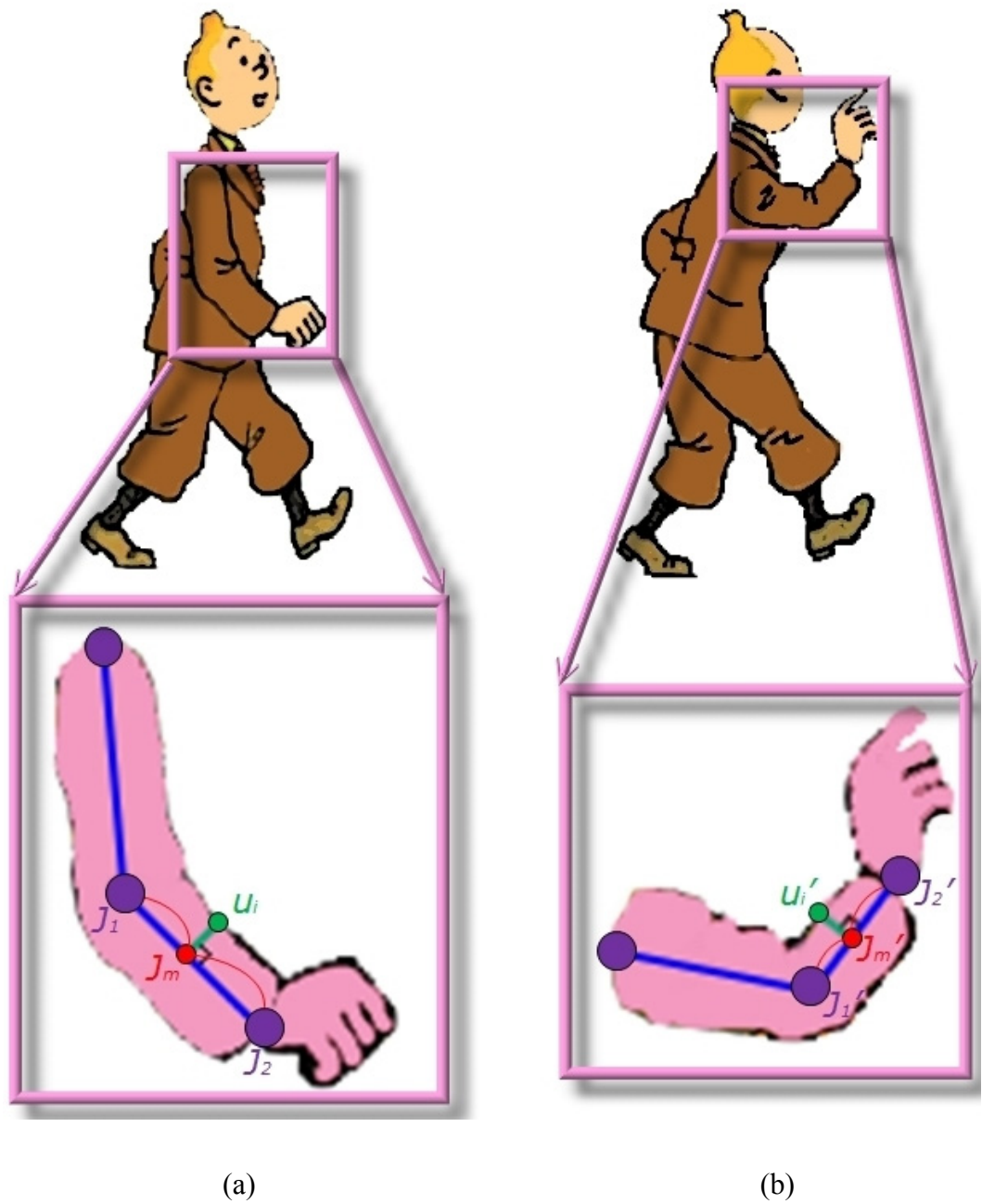


Figure 7.2: Point-to-point correspondences. (a) The character in a frame and the zoom-in view of his right arm. (b) The character in another frame and the zoom-in view of his right arm. © Georges Remi (Hergé)

7.3.3 Bayesian Inference

Now, we would describe how to estimate the most suitable coefficients and parameters

to fit the contour of a character by *Bayesian inference*. The central process of *Bayesian inference* is the calculation of probability distributions on the unknown parameter vectors. Recalling Equation (3.1) to Equation (3.4), Equation (3.6), and Equation (3.7) in Section 3.1, k is the number of elliptic Gaussians. \vec{v}_j denotes the mean of the j -th elliptic Gaussian, while β_j denotes the corresponding coefficient. We define $\sigma_{j,i}^2$ as the covariance of the j -th elliptic Gaussian along i -axis. θ_i is the angle between the major axis of ellipse and i -axis. A_{θ_i, a_i} is the transformation matrix with orientation θ_i and the aspect ratio a_i^2 . Let k, β_j, \vec{v}_j , and $\sigma_{j,i}^2$ be variables and a_i^2 term be fixed. θ_i and A_{θ_i, a_i} are set up according to the principal component of anchor points sampled from contours. Hence, the parameter space Θ can be written as $\Theta = \{\beta_j, \vec{v}_j, \sigma_{j,i}^2\}$ for $j=1, \dots, k$. Taking D_{train} to represent our training data, which are the anchor points and their displacements, we are interested in inference about the posterior probability of parameters Θ conditional on the data, i.e. $p(\Theta|D_{train})$. Recalling Equation (3.6), given a new contour point \vec{u}_{new} (any point on the contour of character), the target response \vec{r}_{new} (the displacement of that point) can be given as an expectation.

$$E[\vec{r}_{new} | \vec{u}_{new}, D_{train}] = \int \hat{f}(\vec{u}_{new}, \Theta) p(\Theta|D_{train}) d\Theta, \quad (7.1)$$

where $\hat{f}(\cdot)$ is the estimation of our ERBF model. However, the integral is intractable and untenable for asymptotic methods. We propose a Bayesian estimation of ERBFs. The proposed method imitates the ERBF procedure by RJMCMC which can approximate the integral of Equation (7.1), as described in Section 3.3.

RJMCMC proceeds by drawing samples of Θ in direct proportion to $p(\Theta|D_{train})$ and then approximates Equation (7.1) by

$$E[\vec{r}_{new} | \vec{u}_{new}, D_{train}] \approx \frac{\sum_{t=n_0+1}^N \hat{f}(\vec{u}_{new}, \Theta_t)}{N - n_0}, \quad (7.2)$$

where N is the number of samples generated, called the Markov chain length, and n_0 is the burn-in period. Θ_t denotes the current parameter space while there are t samples. The burn-in stage discards the samples generated by the Markov chain with unstable distribution of interest $p(\Theta|D_{train})$. Finally, we use Equation (7.2) to generalize the displacement of the character's contour. We can make predictions of the displacement

\vec{r}_{new} of an arbitrary new contour point \vec{u}_{new} by using Equation (7.2). Furthermore, we use *Catmull-Rom splines* to connect new positions of the contour points in in-betweens. So the contours of key-motions are synthesized by the model. In our implementation, we synthesize 10 key-motions between two contiguous frames in a comic or a low-frame-rate video.

7.3.4 The Time Series Model

As mentioned above, these key-motions are described discretely in the temporal domain. For generating a smooth and continuous limbs movement, we synthesize the movement following the motion trajectory that is obtained by *time series*. We propose a nonparametric Bayesian approach to analyze time series data representing the motion trajectory. Recalling Equation (3.16), the equation of ARMA can be rewritten as follows:

$$f_{TS}(D_{t-1}, \dots, D_{t-p}; \alpha_{t-1}, \dots, \alpha_{t-q}) = \sum_{i=1}^p \phi_i f(U_{t-i}) - \sum_{i=1}^q \kappa_i \alpha_{t-i}, \quad (7.3)$$

where D_t denotes a univariate time series. p and q represent non-negative integers. ϕ_i and κ_i are the coefficients of parameters in this model. α_t denotes the white noise. $f(\cdot)$ defined in Equation (3.6) is applied to estimate $f_{TS}(\cdot)$. We develop a Bayesian version of ARMA (BARMA) by combining ERBF kernel with Bayesian inference based on RJMCMC. In our work, $U_i = (\vec{u}_{i,1}, \dots, \vec{u}_{i,n})$ denotes the positions of n contour points sampled from the i -th key-motion. α_i is obtained by drawing a random variable. The coefficients of $f(\cdot)$ have already been inferred by using RJMCMC described in Section 7.3.3. Furthermore, we perform RJMCMC to estimate the optimal coefficients ϕ_i and κ_i . In our implementation, we already synthesize 10 key-motions between two given contiguous frames. We find that an appropriate number of frames is about 10 in our experiments (Note that $p = 10$ and $q = 10$). These key-motions are sufficient to predict the motion trajectory of key-motions effectively. We use BARMA to obtain the motion trajectories. Note that the entire limbs movement of a character is synthesized by fitting contours of each frame in the temporal domain through the trajectories.

7.3.5 Detail Preservation

The details of character are preserved by using LOESS. LOESS is employed to fill in the color and texture information obtained from the original character. A series of motions is synthesized in accordance with the previously fitted contours and details. As mentioned in Section 3.2, the specific location x_0 within the fitted contours, which would be filled in color and texture information, is supplied during a LOESS prediction. LOESS performs a linear regression on the sampled contour points weighted by a kernel centered at x_0 . The weight of the i -th sampled contour point x_i is defined by Gaussian kernel, as illustrated in Figure 7.3.

Recalling Section 5.5, we use the closed-form solution to find the regression coefficients in LOESS. For the same reason, we could apply RJMCMC to sample suitable coefficients of the current model. We use a special form of LOESS called BLOESS to build a model from the data. BLOESS allows meaningful regressions even with fewer data points than regression coefficients. Note that we assume a wide Gaussian prior on the coefficient vector $\zeta \sim N\left(0, \frac{\tau^2}{p_{precision}}\right)$ of the regression model in Equation (3.10) and a Gamma prior on the inverse of the noise variance $\frac{1}{\tau^2} \sim \text{Gamma}(0.1, 0.1)$ in common with RJMCMC sampler. $p_{precision}$ is the precision of the coefficient prior.

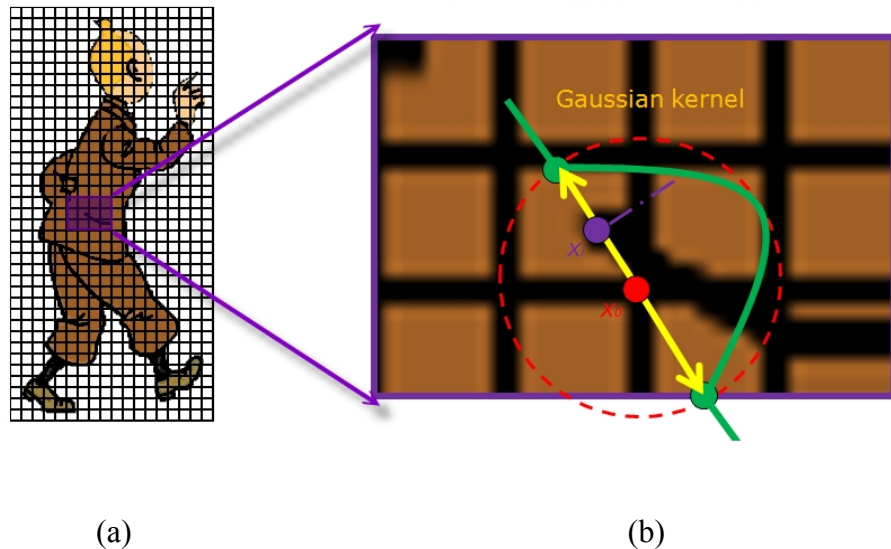
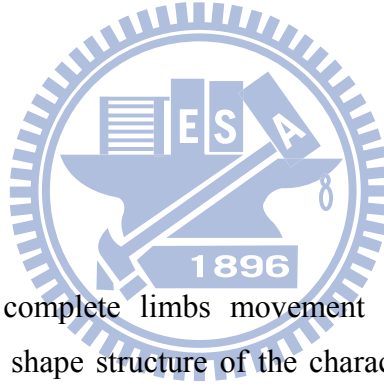


Figure 7.3: LOESS analysis. (a) The original character with a uniform grid (50×50). (b) The zoom-in view of the image. LOESS with Gaussian kernel is applied to estimate the weights. © Georges Remi (Hergé)

Let X be the polynomial terms of data samples in the matrix form. Y denotes the response representing the matrix form of the corresponding new locations. P represents the matrix form of the precision $p_{precision}$. W represents the diagonal matrix form of $w_i(x_0)$ for $1 \leq i \leq m$ in Equation (3.8). $\hat{\zeta}$ is obtained from the marginal posterior distribution with posterior mean $\bar{\zeta} = (X^T W^2 X + P)^{-1} X^T W^2 Y$ and modified standard deviation. Note that the initial standard deviation is drawn from the noise variance τ^2 and modified to be the upper triangle of posterior variance matrix $(X^T W^2 X + P)^{-1}$ obtained by using *Cholesky decomposition*. According to the estimated regression coefficient vector $\hat{\zeta}$, we can use Equation (3.10) to find the new location of x_0 and obtain the pixel values for filling in the color and texture information. In practice, we approximate the character with a uniform grid, as shown in Figure 7.3 (a). We find the new location of each vertex in the grid. Then we fill the resulting quad using bilinear interpolation.

7.4 Summary



In brief review, the complete limbs movement synthesis process consists of the following phases. The shape structure of the character is built first. Each layer of the shape structure of characters consists of several moving components, for instance, head, body, arm, and leg. The indications of moving components are then refined manually by the predefined 2D skeleton structure. The deformed contour of each moving component is synthesized through Bayesian regression. Key-motions are synthesized by combining all moving components using alpha-blending or the painter's algorithm with connective constraints formed from shape structures. As mentioned before, we actually create 10 key-motions between two contiguous frames in a comic or a low-frame-rate video by using ERBFs with the parameters estimated by RJMCMC. Then we construct the time series model BARMA to track the motion trajectory, which best matches key-motions and generates the entire limbs movement in contours. Bayesian regression and time series simulation are both constrained to the connection topology in the shape structure. Furthermore, BLOESS is applied to reconstruct the details within the deformed contours fitted from Bayesian regression and BARMA. The entire limbs movement is

synthesized after shape deforming and detail preserving. Actually, we forecast 300 frames to generate the 10 second character animation between two contiguous frames in a comic.

7.5 Experimental Results

The implementations were conducted on digitized images obtained from comics, such as “The Adventures of TinTin: The Shooting Star”, which was originally produced by Georges Remi (Hergé). The proposed time series model with nonparametric Bayesian inference was implemented on an Intel Core 2 Quad 6600 2.40 GHz CPU and 3 gigabytes main memory, which enabled smooth limbs movements. Table 7.1 lists performance measurements for the figures shown. Execution time is measured in each step. Shape Structure consists of the time of segmentation and the whole shape structure generation. Bayesian Regression comprises the time of ERBF kernel training and RJMCMC sampling. RJMCMC sampling took a lot longer. All the simulations were run with a burn-in period of 5000 iterations of RJMCMC followed by 10000 samples. Time Series indicates the time to construct the time series model.

Table 7.1: Performance measurements of limbs movement synthesis.

	TinTin	Ball	Man	Bunny
Figure No.	7.1	7.5	7.6	7.7
Resolution	240×502	100×75	368×583	319×646
Shape Structure (Second)	7	1	6	9
Bayesian Regression (Second)	1834	43	2046	2257
Time Series (Second)	102	21	104	128
UI (Minute)	1	0	1	1

Note RJMCMC sampling is carried out once to obtain the regression coefficients during Time Series step because the number of ERBFs is known (recalling $p = 10$ and $q = 10$ in Equation (7.3)). UI represents the time of users' intervention to refine shape structure further. Besides, Detail Preservation is not demonstrated since it takes less than 20 milliseconds for each frame and is quite fast. The time of animation generation is not shown because it varies significantly due to different numbers of frames generated.

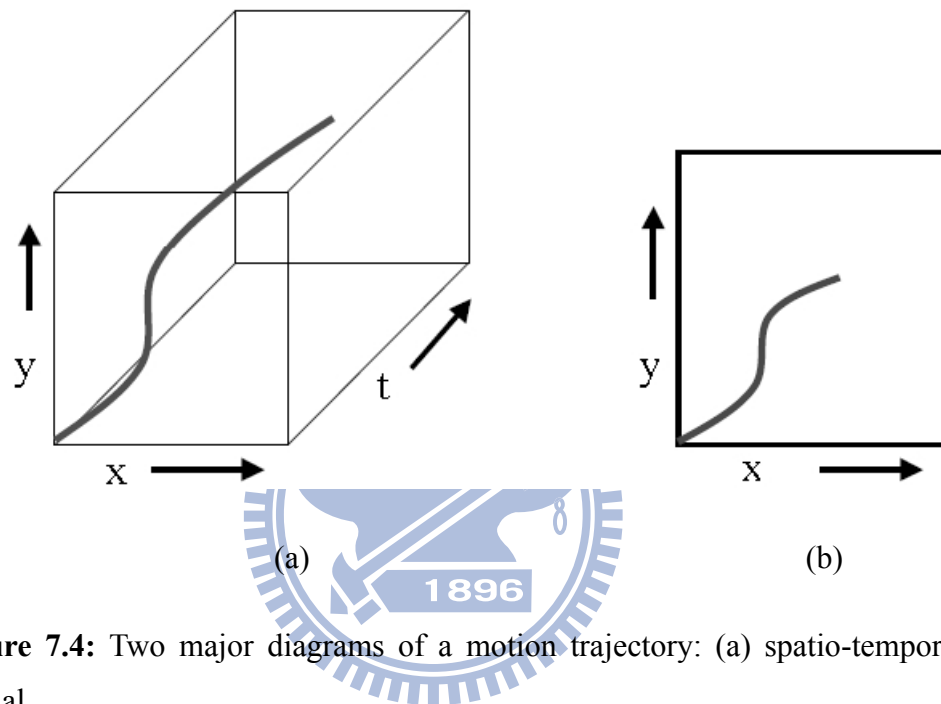


Figure 7.4: Two major diagrams of a motion trajectory: (a) spatio-temporal and (b) spatial.

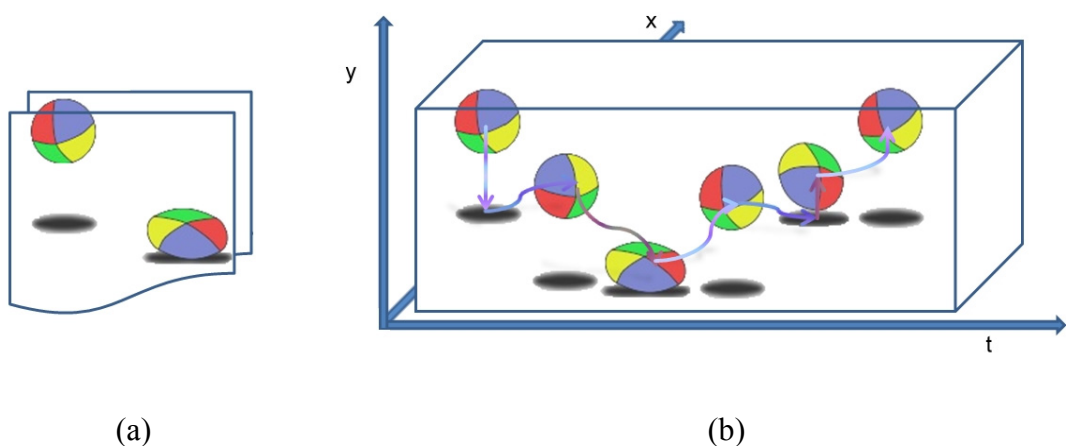
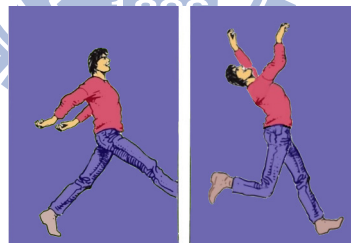


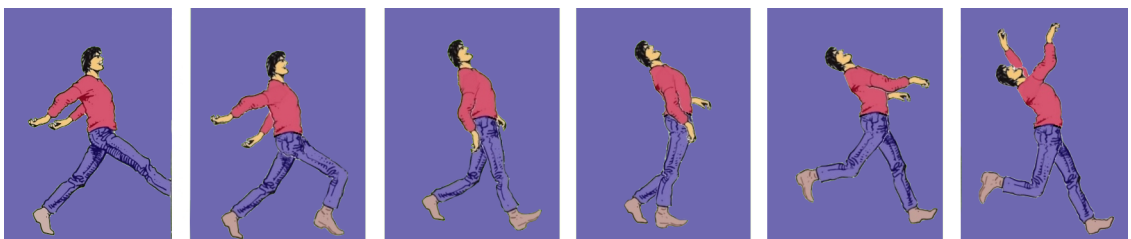
Figure 7.5: A motion trajectory of a ball. (a) Given two consecutive frames of a bouncing ball in a low-rate-frame video, (b) the synthesized frames of animation are shown with the motion trajectory.

The motion trajectory obtained by the proposed time series model is described in Figure 7.4. The spatio-temporal diagram, which is illustrated in Figure 7.4 (a), captures the movement of a character in time. Then it can be mathematically represented as a curve (x,y,t) in three-dimensional space or equivalently as a parametric curve $(x(t),y(t))$ in the two-dimensional space. The spatial diagram can be mathematically represented as a one-dimensional function $y = f_{MT}(x)$. As illustrated in Figure 7.4 (b), the spatial diagram is a projection of the spatio-temporal trajectory onto the image plane. Figure 7.5 (a) shows two consecutive key-poses of a bouncing ball in a low-frame-rate video. There is one moving component only. The motion of the bouncing ball was synthesized with its motion trajectory. As shown in Figure 7.5 (b), the trajectory is described by the movement of the ball's barycenter. The ball was animated along with the trajectory.

The results which are selected frames of 2D character animations generated by our method are presented in Figure 7.1 and Figure 7.6. There are 10 moving components representing the character shown in Figure 7.6 (a), such as head, right arm, right hand, torso, left arm, left hand, right leg, right foot, left leg, and left foot. Three motion trajectories of the man were obtained by BARMA. The extracted frames of the animation synthesized by using these motion trajectories are shown in Figure 7.6 (b).



(a)



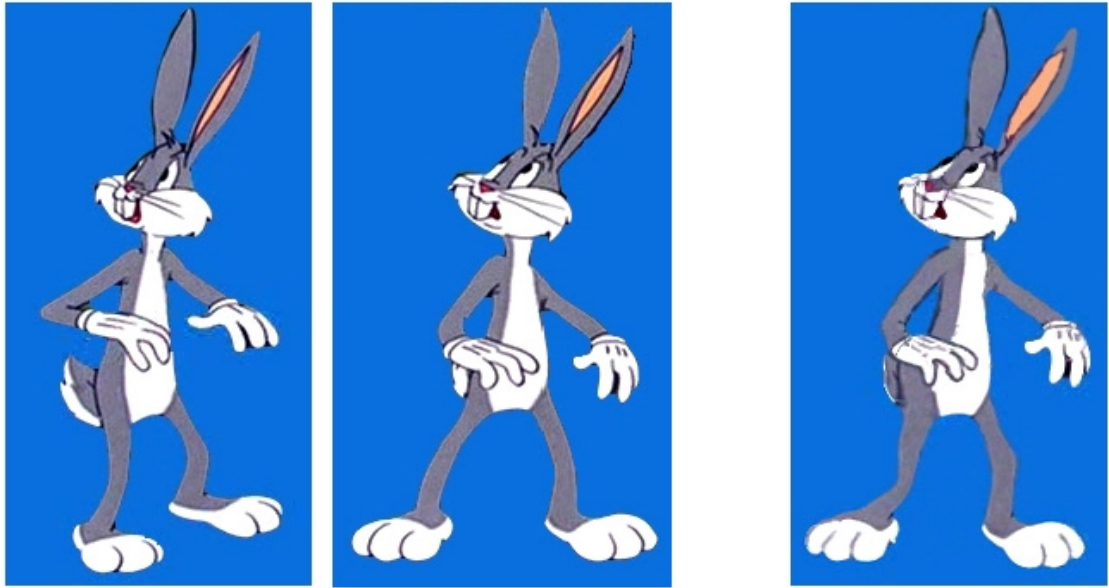
(b)

Figure 7.6: Limbs movement synthesis. (a) Given two consecutive frames in the comic, (b) limbs movement synthesis is carried out using the estimated time series.

Furthermore, Figure 7.1 (a) shows several digitized frames of comic obtained from “The Adventures of TinTin: The Shooting Star”. There are 8 moving components representing the character shown in Figure 7.1 (a), such as head, right arm, right hand, torso, left arm, left hand, left leg, and right leg. As similarly above, the character shown in Figure 7.1 consists of three layers in the proposed shape structure. Three motion trajectories were computed by BARMA from these three layers respectively and used to simulate the complete animation. The results reveal the strength of our method as the possibility of convincingly posing or animating any kind of comic characters, as shown in Figure 7.1 (c).

As mentioned before, DeJuan and Bodenheimer [19] synthesized in-between contours and textures by RBF kernel and elastic registration. However, for animating characters, their method would be operated iteratively. The whole animation was interpolated directly. Using such image-based interpolation method they proposed may introduce new artifacts. Our approach could generate a smooth and natural-looking 2D character animation by using the time series estimated from *Bayesian inference* with ERBF kernel. Note that the time series model is applied to represent the trajectory of a character’s motion. It should be noted that, the motion trajectory could be further used to predict a character’s movement, viseme, or facial expression by nonlinearly extrapolating reasonable deformations without the restriction of a purely interpolation method.

Figure 7.7 provides the comparison with the in-betweening technique proposed by DeJuan and Bodenheimer [19]. Given keyframes shown in Figure 7.7 (a), they generated the in-between bunny revealed in Figure 7.7 (b). Figure 7.7 (c) shows our synthesized result. Note that our method is based on ERBF kernel. It is more suitable than RBF kernel for fitting contours, which have noncircular structures, such as the arms and legs of the bunny. Note that there are 11 moving components representing the bunny, such as head including ears, right arm, right hand, torso, tail, left arm, left hand, right leg, right foot, left leg, and left foot. Two layers are constructed in the proposed shape structure. One layer consists of head, right arm, right hand, left arm, and left hand. Another layer consists of the other 6 moving components. The motion trajectories are also computed by BARMA for synthesizing character animation.

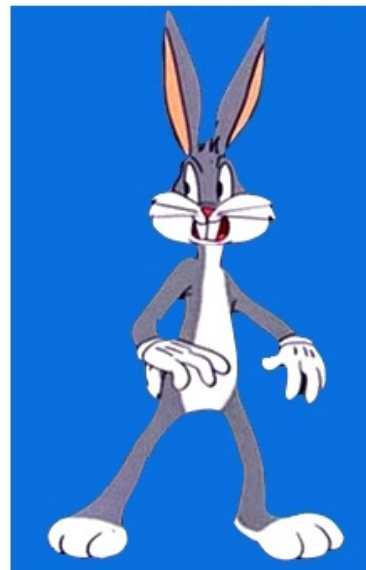


(a)

(b)



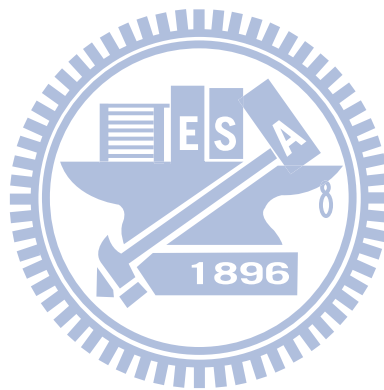
(c)



(d)

Figure 7.7: Visual comparison of character animation. (a) The inputs are two keyframes. (b) The in-between frame for Bugs Bunny are generated by using the technique proposed by DeJuan and Bodenheimer [19]. (c) Motion synthesis is carried out using the estimated time series for character animation. (d) A novel view and new expression is predicted by using the estimated time series. © Warner Bros. Entertainment Inc.

Furthermore, the character's novel view or expression could be forecasted directly by using the time series model, as shown in Figure 7.7 (d). In order to maintain the 3D effect of this new view, it sometimes uses color blending or cross dissolving to combine the features coming from two views. Given Figure 7.7 (c) and its reverse, the motion trajectory was estimated by the movements of facial components from Figure 7.7 (c) to its reverse. Figure 7.7 (c) was deformed forward to synthesize a frontal view by the motion trajectory. Then we combined the frontal view with another frontal view obtained by backward deformation from the reverse of Figure 7.7 (c) by using color blending. Note that the features of the bunny's head in the frontal view are simulated, such as the right ear of the bunny.



Chapter 8

Animating Passive Elements

The proposed statistical approaches have many advanced multimedia applications of next generation environments. This chapter presents a novel application of natural image for taking a still picture and making it move in convincing ways by simulating natural phenomena. Given a picture of a pond, we make it ripple. Given a picture of a tree, we make it sway. Considering the traditional 2D animation production, the background of the scene is the same as the character. The production process of building up sequences of drawn passive elements in the background is to resemble a movement by hand. It is still a labor-intensive artisan process. For the reason, we focus on simulating the movements of these passive elements. In this chapter, we explore how a set of explicitly encoded pre-sampled data representing the passive elements' movements in still pictures by using *kernel regression* with ERBFs mentioned in Section 3.1. *Simple harmonic motion* is applied to estimate displacements of samples of passive elements in next time-sliced scene. Then these positions are used to reconstruct the whole moving passive elements. The same process to preserve details is carried out by using LOESS mentioned in Section 3.2.

8.1 Simple Harmonic Motion

In physics, *simple harmonic motion* is the motion of a simple harmonic oscillator [71]. *Simple harmonic motion* is typified by the motion of a mass on a spring when it is subject to the linear elastic restoring force given by *Hooke's law*. To explore *simple harmonic motion*, let's take the example of a spring with a mass M in the absence of

gravity. If this is an ideal spring, the force is $(-K_{Sti}X_{Dis})$ where K_{Sti} is a measure of the stiffness of the spring and X_{Dis} is the displacement. The force is toward the origin if that is the equilibrium position of the spring. *Newton's second law* becomes

$$M \frac{d^2 X_{Dis}}{dt^2} = -K_{Sti} X_{Dis}. \quad (8.1)$$

The answer of this differential equation is

$$X_{Dis}(t) = A_{amp} \sin(\omega t + \phi), \quad (8.2)$$

where $X_{Dis}(t)$ denotes the displacement of a mass at any time t . A_{amp} denotes the amplitude representing the maximum displacement from equilibrium. ω denotes the angular frequency. In this dissertation, without loss of generality, we will take ϕ , also called the phase shift, to be zero (or just defining it where $t = 0$ is). The motion is periodic and sinusoidal in time. Each oscillation is identical, and thus the period, frequency, and amplitude of the motion are constant. The motion equation (8.2) for *simple harmonic motion* contains a complete description of the motion, and other parameters of the motion can be calculated from it. Moreover, using the techniques of differential calculus, the velocity and acceleration as a function of time are given by

$$v(t) = \frac{dX_{Dis}}{dt} = \omega A_{amp} \cos \omega t, \quad (8.3)$$

$$a(t) = \frac{d^2 X_{Dis}}{dt^2} = -\omega^2 A_{amp} \sin \omega t = -\omega^2 X_{Dis}(t). \quad (8.4)$$

Since $Ma(t) = -M\omega^2 X_{Dis} = -K_{Sti} X_{Dis}$,

$$\omega^2 = \frac{K_{Sti}}{M}. \quad (8.5)$$

Besides, $\omega = 2\pi f_{Fre}$ where f_{Fre} denotes the frequency representing the number of cycles per second,

$$f_{Fre} = \frac{1}{2\pi} \sqrt{\frac{K_{Sti}}{M}} = \frac{1}{T_{Per}}, \quad (8.6)$$

where T_{per} denotes the period representing the time required to complete a full cycle. These equations demonstrate that the period and the frequency are independent of the amplitude and the initial phase shift of the motion.

8.2 Algorithm Overview

The proposed approach for animating passive elements from a single still picture consists of the following five components: Passive Element Specification, Edge Extraction, Movement Selection, Displacement Estimation, and Movement Prediction. In Figure 8.1, the outline reflects the structure of our proposed method to animate passive elements. Considering Figure 8.1, we briefly describe our method in the following paragraphs.

1. Passive Element Specification: The user can scribble different colors in the interiors of various regions for marking the passive elements which would be animated. Note that each color indicates a specified movement style for that region. Real images are simplified by using image abstraction mentioned in Chapter 4. Once the user scribbles on the picture, the whole passive element is obtained by propagating that stroke through the level set method mentioned in Section 5.3.

2. Edge Extraction: As mentioned previously, the process to animate passive elements is base on shape deforming by applying the proposed statistical approaches to fit contours. Thus, passive elements are extracted by using the proposed level-set-based *GrabCut* mentioned in Section 5.3. Then the edges of these elements are extracted by using an edge detector.

3. Movement Selection: Our system provides two kinds of movement styles for natural phenomena simulation. For water waves, we focus on the ripple effect of water. Moreover, the branched and trunks of trees can be modeled as approximated physical systems like a simple pendulum. Wind force causes trees to sway. Another natural phenomenon is to model the periodical oscillation effect of trees. Besides, the dynamics of passive elements are driven by the wind force. The driving wind force causes the wave motion of these elements. Hence, the user could control the wind direction, the

wind speed, the amplitude, and the frequency of the wave motion.

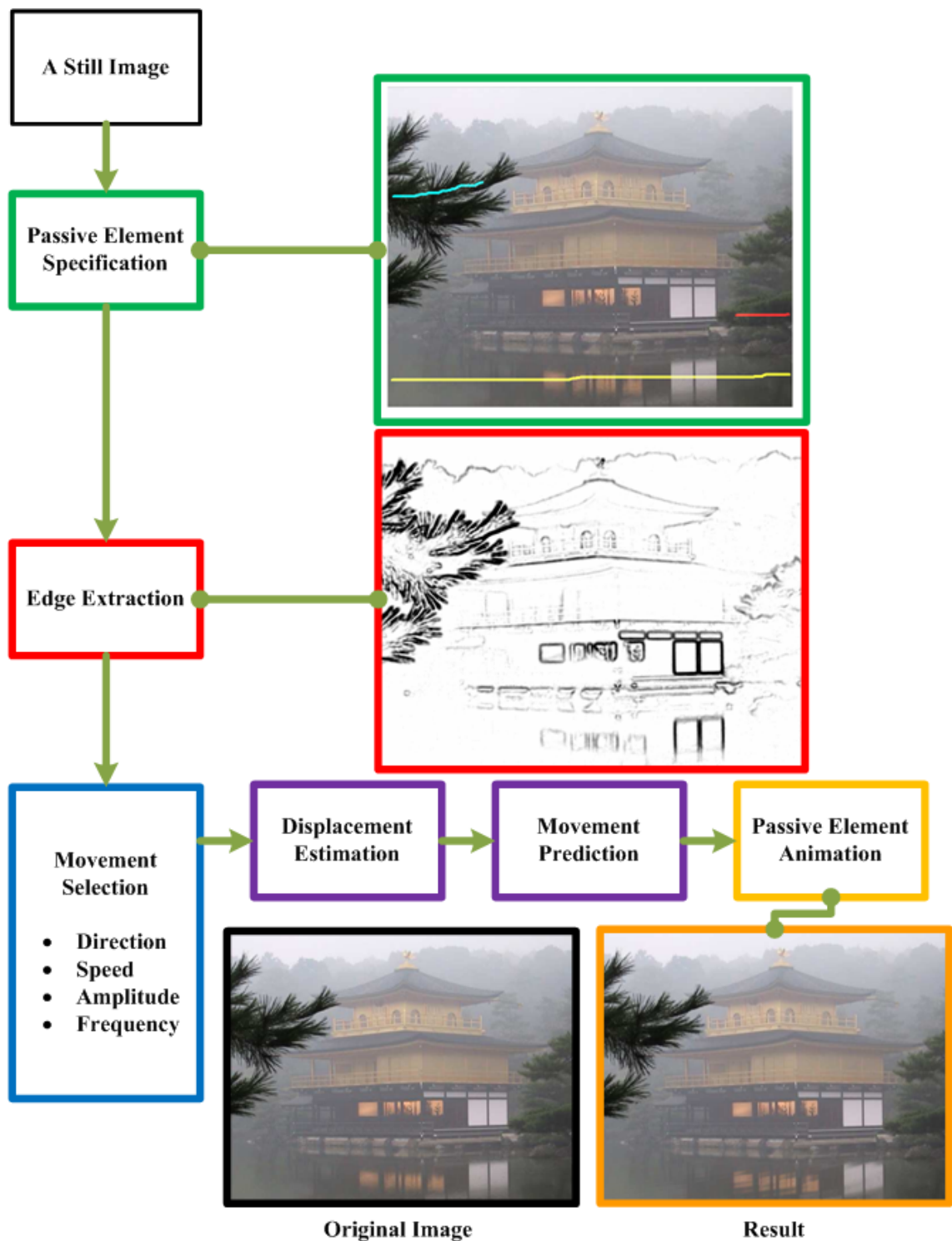


Figure 8.1: The overview of passive element animation with the example of the Japanese Temple named Temple of the Golden Pavilion (from Japanese term Kinkaku-ji).

4. Displacement Estimation: As mentioned above, our approximated physical system is modeled by *simple harmonic motion*. The displacements of the samples, which are sampled on the interiors of passive elements, are estimated by applying *simple harmonic motion* with the specified parameters.

5. Movement Prediction: All displacements of the samples are used to compute the distortions along x -axis and y -axis of pixels. These distortions are encoded explicitly by the regression surface using *kernel regression* with ERBFs. By using the representation, the movements of passive elements are simulated. Finally, LOESS is applied to preserve details if necessary.

8.3 Passive Element Animation

In this section, we describe our approach to simulate the movements of passive elements. We focus on the details of each movement style, i.e., water waves and trees.

8.3.1 Extraction and Specification

In addition to the paintings or drawings, real images are simplified by using the bilateral filter. Once the user scribbles on the image, regions with a similar color distribution of the drawing or the corresponding filtered image can be sensibly segmented by a single scribble propagation using the level set method. The segmented regions, which are propagated from the same scribble, indicate that they belong to the same passive element. Note that if the color distribution is too similar between the background and a passive element, the user should refine the scribble to mask the region of that element manually, as showed in Figure 8.2 (a).

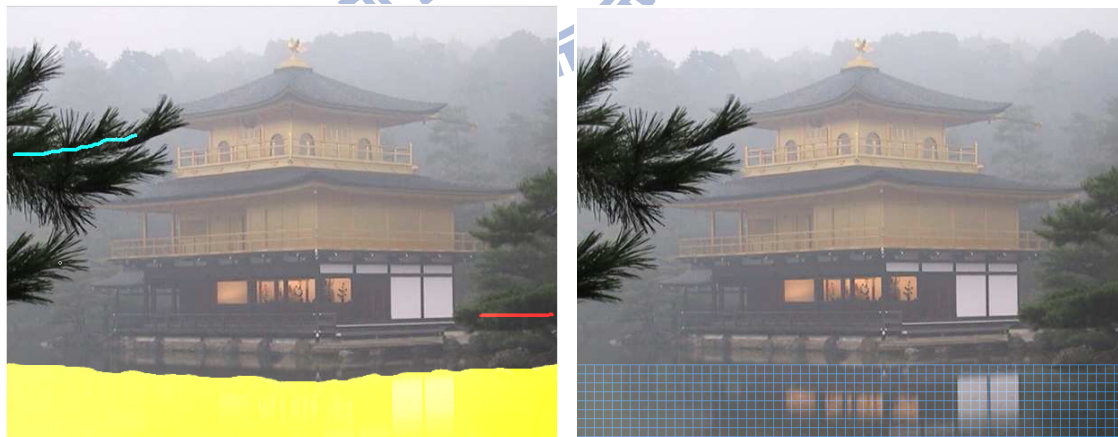
Level-set-based *GrabCut* is applied to obtain the passive elements, which we want to animate, through analyze these segmented regions iteratively. Next, the contours of these elements are extracted by using *Sobel Operator* [32] for further sampling. Besides, different scribbles with different colors represent various movement styles. The user could choose parameters of each motion type to control a specified motion, i.e., the

wind direction, the wind speed, the amplitude, and the frequency of the wave motion. Furthermore, the ripple effect of water and the oscillation effect of trees driven by the wind force are designed to simulate specified natural phenomena.

8.3.2 Water Waves

In order to simulate the movement of water effectively, we assume that the water surface is discrete and consists of water particles. We sample the water surface with a uniform grid (50×50), as shown in Figure 8.2 (b). Each sample is the water particle on the water surface defined as the grid point on the uniform grid. Next, according to the movement style, we estimate the displacements of these sampled water particles.

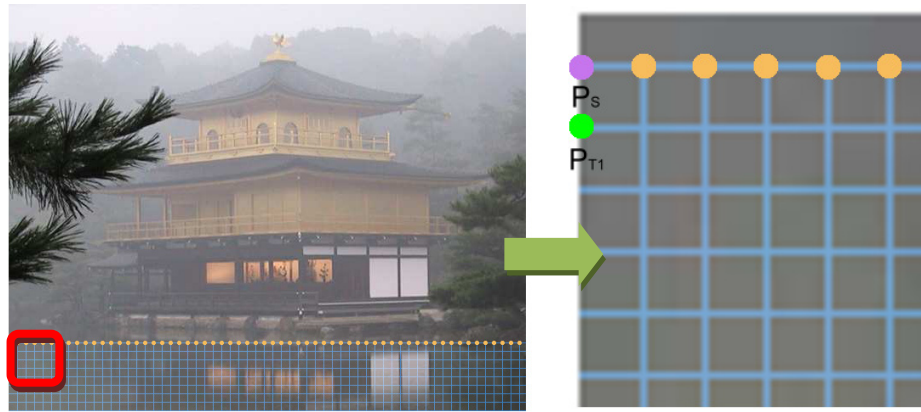
Based on the observation of the ripple effect, a pebble thrown into a pond will produce concentric circular ripples, which move outward from the point of impact. The ripple is the wave motion based on *harmonic oscillation*. Thus, the point of impact is determined by the user-specified wind direction first.



(a)

(b)

Figure 8.2: The example of the Temple of the Golden Pavilion for segmentation and sampling. (a) Users should mask the whole region of the pond, since its color distribution is too similar to the background. (b) The pond with a uniform grid (50×50).



(a)

(b)

Figure 8.3: The example of the wind blowing downward. (a) The sources of the wave (orange dots) are selected. (b) The zoom-in view of the image. Note that P_S (purple dot) denotes the point of impact. P_{T1} (green dot) denotes the sample of the water particle, whose displacement is estimated from P_S .

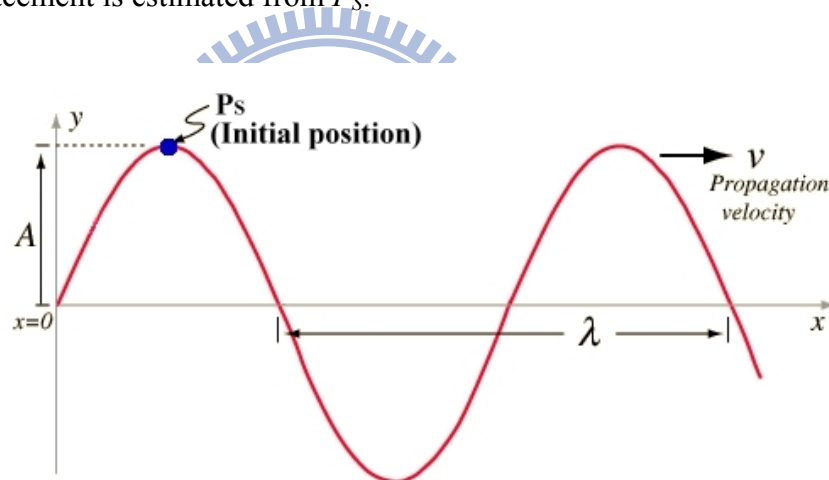


Figure 8.4: Schematic diagram of *simple harmonic motion*.

For example, the top point P_S of the intersection between the grid and the outer contour of the pond in image space is chosen as the point of impact when the wind direction is downward. Moreover, the user-specified amplitude is considered as the displacement of the water particle at this point. Besides, the grid points on the vertical, horizontal, or diagonal direction of P_S are determined to be the sources of the wave in terms of the wind direction. Figure 8.3 (a) shows P_S (purple dot) and the sources of the wave (orange dots) when the wind direction is downward. Note that the sources of the wave are selected according the direction vertical to the wind direction. The water particles' displacements at these sources are the same as the displacement at P_S , which

is the user-specified amplitude. Then we could estimate each sample's displacement by applying *simple harmonic motion*. Recalling the equations reviewed in Section 8.1, the motion equation is derived by the differential equation to time dimension in the spring system. Now, we simulate the traveling wave to estimate samples' displacements like the displacement of P_{TI} (green dot). As illustrated in Figure 8.4, x -axis indicates the position of the grid point in image space. y -axis indicates the water particle's displacement in vertical direction. Propagation velocity v equals to the user-specified wind speed. λ denotes the wavelength. A denotes the amplitude of the wave representing the displacement of the wave in the initial position. In other words, A represents the initial displacements of the point of impact P_S and the sources of the wave. We could estimate the displacements of samples at different time slices from initial displacements of P_S and the sources of the wave respectively.

$$y(x, t) = A \sin \frac{2\pi}{\lambda}(x - vt), \quad (8.7)$$

$$\frac{2\pi v}{\lambda} = 2\pi f_{Fre} = \omega, \quad (8.8)$$

where f_{Fre} and ω are the frequency and angular frequency respectively, as defined in Section 8.1. Moreover, a traveling wave would take the form of a sine wave. The motion relationship "distance = velocity × time" is the key to the basic wave relationship of the wave frequency, wavelength, and propagation velocity. With the wavelength as distance, this relationship becomes $\lambda = vT_{Per}$. Then using $f_{Fre} = 1/T_{Per}$ gives the standard wave relationship:

$$v = f_{Fre}\lambda. \quad (8.9)$$

Thus, we could estimate $\lambda = v/f_{Fre}$. Finally, we assemble Equations (8.7), Equation (8.8), and Equation (8.9) to compute all displacements of samples, which are the displacements of the water particles on all grid points. Note that the displacement of each sample would be computed from its the nearest source of the wave along the wind direction. After computing all displacements of samples, we further estimate the displacements of these samples' 4-neighbors. Then we would approximate the distortions of these samples. Note that the degree of distortion is mainly determined by the gradient of the water surface, the refraction, and the depth of the water.

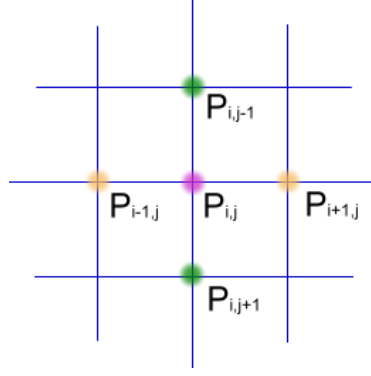


Figure 8.5: The diagram of an arbitrary sample ($P_{i,j}$) and its 4-neighbors ($P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, and $P_{i,j+1}$). Each point represents as a pixel in image space.

We only approximate them by analyzing the gradient since the distortion increases when the water surface is more declivous. Hence, the difference between the displacements of the water particle's 4-neighbors is represented as the distortion of that particle:

$$\Delta x_{Off} = D_{Dis}(P_{i+1,j}) - D_{Dis}(P_{i-1,j}), \quad (8.10)$$

$$\Delta y_{Off} = D_{Dis}(P_{i,j-1}) - D_{Dis}(P_{i,j+1}), \quad (8.11)$$

where $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, and $P_{i,j+1}$ is shown in Figure 8.5. Δx_{Off} denotes the distortion along the horizontal direction of $P_{i,j}$ in image space. Δy_{Off} denotes the distortion along the vertical direction in image space. $D_{Dis}(\cdot)$ is the displacement of its parameter. Note that these distortions represent the coordinate offsets along the horizontal direction and the vertical direction in image space (offsets of x -axis and y -axis).

Furthermore, we could obtain all samples' distortions along x -axis and y -axis. Each distortion is encoded explicitly by the regression surface using *kernel regression* with ERBFs. By using the representation, we could find the distortions of all water particles on the water surface. That means the distortions of all pixels in image space which belong to the passive element (water) are obtained. In the other words, these distortions are used to train the kernel regression model, as described in Section 3.1. In general, we formulate this problem as regression analysis. Recalling Equation (3.6), the relationship of the response $\vec{r} = (\Delta x_{Off}, \Delta y_{Off})$ and the predictor \vec{u} representing the coordinate of each pixel can be constructed. Then the trained model is used to fit the distortions of all

pixels in image space which belong to the passive element (water) by Equation (3.6). Hence, new pixel values of all pixels are determined by copying the pixel values at new positions (the original coordinate plus the offsets). The movement of the passive element is simulated.

8.3.3 Trees

Simple harmonic motion is found if there is a small enough deviation from stable static equilibrium. For example, touch the water surface lightly or knock the drum surface. For a small wind, the leaves on the tree show *simple harmonic motion*. For a stronger wind, branch of the tree will show *simple harmonic motion*. For a huge wind, the whole tree might show *simple harmonic motion*. Now, we provide an approximated physical system to simulate the swaying trees in spirit to the motion of a simple pendulum.

First, we find the bounding box of the extracted trees, which is obtained by the method mentioned in Section 8.3.1. As illustrated in Figure 8.6, the tree is enclosed by the bounding box (red rectangular). Then the skeleton of the tree named the major axis of the tree is chosen through the intersection between the tree and the bounding box. Note that the blue circle indicates the tip of the tree. Another end of the major axis is called the root.

Next, we could compute the displacement of the tip by *simple harmonic motion*. Based on the observation, the driving function that causes trees to sway is typically wind. Then the elastic restoring force of the tree makes the tree back to the equilibrium position. The motion of the tree can be estimated by using Equation (8.2) and the user-specified parameters. Note that the circular motion of the tip is considered as the horizontal motion since the angle of sway θ shown in Figure 8.6 (b) is smaller enough. Recalling Equation (8.2), the displacement of the tip is estimated. Then we compute the equation of the major axis and sample the axis uniformly. According to the extreme position of the tip (the equilibrium position plus the displacement) and the position of the root, the corresponding extreme positions and the displacements of other samples are estimated approximately.

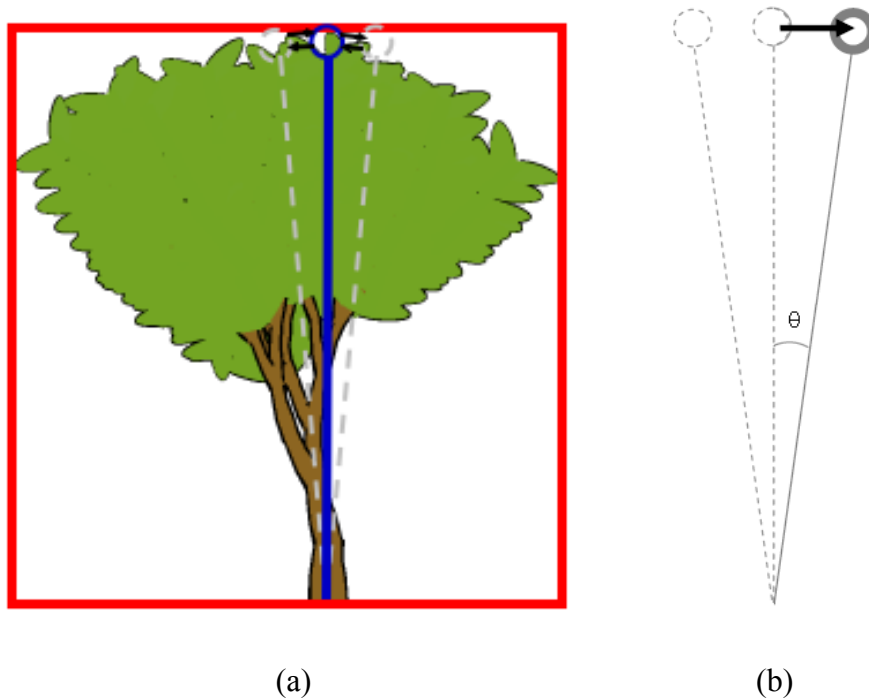


Figure 8.6: The diagram of the tree with *simple harmonic motion*. (a) The oscillation of major axis of the tree. (b) The zoom-in view of the motion of the major axis.

Furthermore, these displacements are encoded explicitly by the regression surface using *kernel regression* with ERBFs. It is similar to the process of the water waves. Moreover, the contours of the tree are extracted by using the method described in Section 8.3.1. By using the representation, we could find the displacements of the contours. That means deforming the shape of the tree in image space is carried out. Finally, the details, features, or textures from the fitted contours interiors are preserved by using LOESS. Hence, the movement of the passive element is simulated.

8.4 Experimental Results

We have applied our system to several photographs and famous paintings. Here, we show some of the results. First, we simulated the water ripple effect in the pond. We used the small amplitude of wave to give the ripples a fine-grained look. Figure 8.7 shows the extracted frame from the resulting passive element animation. We use this example to demonstrate that we can change the appearance of the water by controlling the specified parameters. Furthermore, we show another look of the water under

different wind speeds, directions, and frequency. Figure 8.7 (a) is the original real image of the Japanese Temple named Temple of the Golden Pavilion (from Japanese term Kinkaku-ji). Figure 8.7 (b) is the animated picture with higher wind speed. Figure 8.7 (c) shows the wind of different direction. Figure 8.7 (d) shows the rougher water surface.

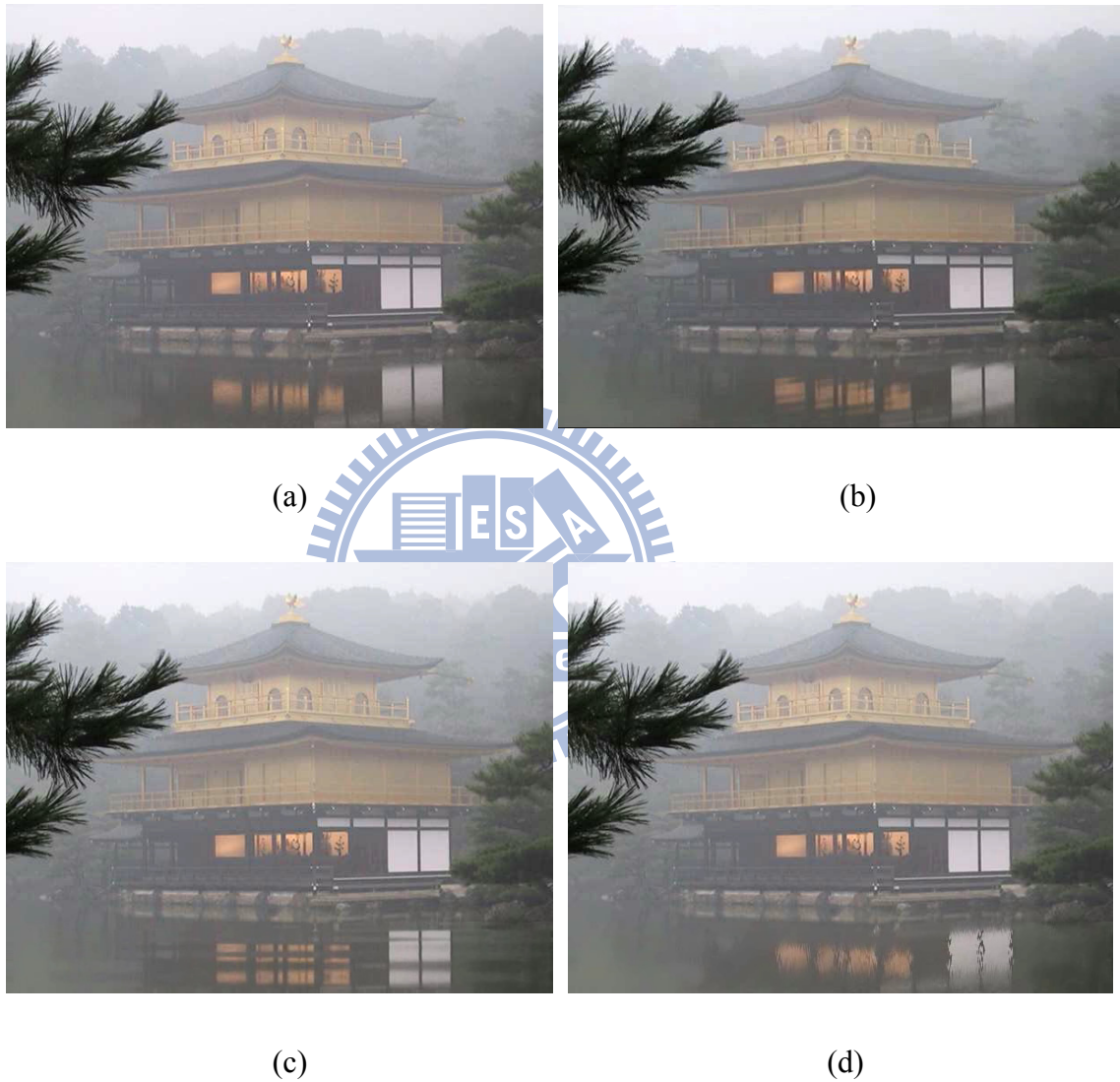


Figure 8.7: The result of the Temple of the Golden Pavilion. (a) The original image. (b) Simulating the ripple effect with higher wind speed. (c) The different wind direction. (d) The rougher water surface.



(a)

(b)

Figure 8.8: The result for swaying plants. (a) The original painting of Vincent van Gogh’s Still Life Vase with Fourteen Sunflowers. (b) The extracted frame from the animated painting.

Moreover, our model works even better with paintings. Figure 8.8 shows the result of the swaying plants. For Vincent van Gogh’s Still Life Vase with Fourteen Sunflowers, we used our model to animate 3 layers of plants. Note that the resulting animation is smooth and impressive. The viewer can find out that the flowers swaying in a natural way with the simple harmonic motion model. It is just like the flowers’ motions are driven by wind blowing.

Furthermore, Figure 8.9 provides an example to make a tree sway. Note that we estimate the motion of the tree with the angle of sway $\theta < 5$. Note that the original circular motion of *harmonic oscillation* can be considered as the horizontal motion Figure 8.9 (a) shows the original painting of Vincent van Gogh’s “Country Road in Provence by Night”. The extracted frame in a different time slice of the swaying tree is shown in Figure 8.9 (b). Moreover, the pattern of the tree is also preserved by using LOESS.

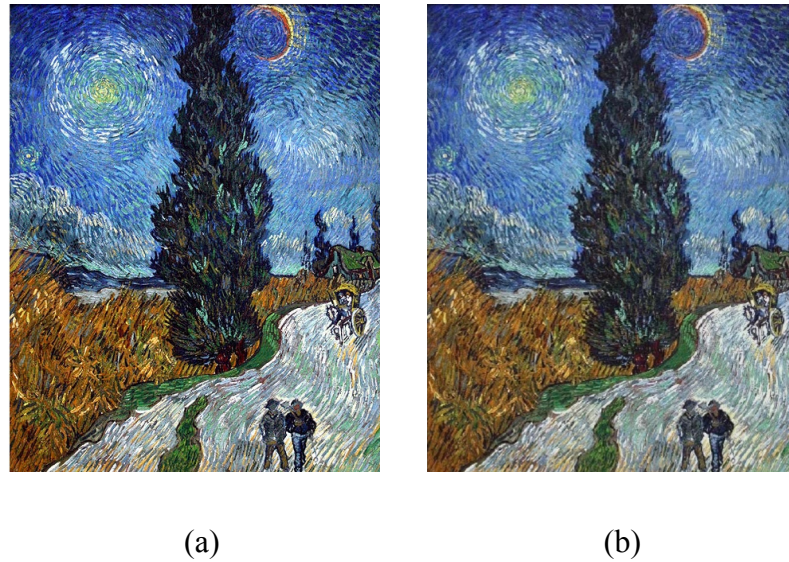


Figure 8.9: Another example for a swaying tree. (a) The original painting of Vincent van Gogh’s “Country Road in Provence by Night”. (b) The extracted frame from the animated painting.

Table 8.1: Performance measurements for passive element animation.

	The Temple of the Golden Pavilion	Still Life Vase with Fourteen Sunflowers	Country Road in Provence by Night
Figure No.	8.7	8.8	8.9
Resolution	608×480	632×849	467×599
Shape Deforming (Millisecond)	3096	3044	2602
Detail Preserving (Millisecond)	0	1443	980
UI (Minute)	1	2	1

Finally, Table 8.1 lists performance measurements for the figures shown. The proposed model for passive element animation was implemented on an Intel Core 2 Quad 6600 2.40 GHz CPU and 3 gigabytes main memory, which enabled to apply the proposed statistical approaches to animate passive elements for simulating natural phenomena effectively. It takes from several seconds to several minutes to animate a picture depending on the complexity of the input picture. Shape deforming represents the time for training and fitting with *kernel regression* with ERBFs. Detail preserving

represents the time for training and fitting with LOESS. UI represents the time of users' intervention to specify the passive element, refine the scribble to mask the region of passive element, and further indicate movement styles on different passive elements. Note that we only deformed the water surface shown in Figure 8.7. We simulated the ripple effect with the distortion of the water. Hence, we did not perform the detail preservation procedure in that scene. Furthermore, there are 3 layers specified by users in Figure 8.8 manually. Each layer represents different frequencies of movements.

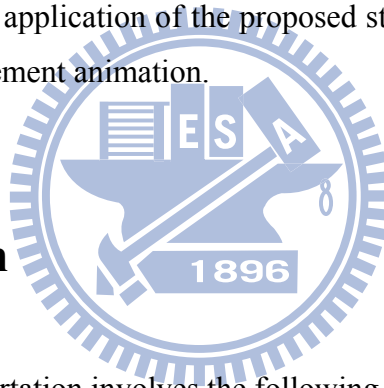


Chapter 9

Conclusion and Future Work

In this dissertation, we have introduced a novel set of statistical approaches which expand the working range of 2D character animation in three directions: novel view generation, expressive talking face simulation, and limbs movement synthesis. We have also presented a novel application of the proposed statistical approaches to animate still pictures for passive element animation.

9.1 Conclusion



In particular, this dissertation involves the following major works:

The Statistical Approaches. The statistical approaches are investigated to eliminate the time-consuming aspects of the traditional 2D animation production. By considering the traditional 2D animation, some key limitations and fallacies are identified. That is the problem of the repeated drawing and coloring of all characters in all frames. To this end, we particularly focus on automatically calculating the movements of characters in in-between frames. We use regression analysis to estimate and forecast the variations of the shapes of characters during deformations in image space. *Bayesian inference* is used for adding the smooth variety during regression analysis and improves meaningful regressions even with fewer data points than regression coefficients. Moreover, the motion of a 2D character is a 3D transformation problem in essence, which consists of a 2D spatial displacement and a 1D shift in time. Considering the temporal relation, time series analysis is applied to estimate the moving trajectory of a character's limb for the

smooth motion simulation.

Shape Deformation and Detail Preservation. This dissertation proposes a novel method for 2D Character animation. The character's motion is analyzed and predicted by *nonparametric regression*. That is, *kernel regression* with ERBFs is used to deform the shape of a character in image space directly for synthesizing the motion of a character. The extension to ERBFs decreases fitting time involves in alleviating the motion synthesis problems that are commonly observed for characters in noncircular structures. Moreover, LOESS is used to preserve the details from the deformed shape interiors by filling in the color and texture information obtained from the original character in the given image. The animation process is similar to machine learning in artificial intelligence or neural network community. In terms of different training data sets, we could synthesize different kinds of motions of a character, such as a novel view or an expressive face with lips movement from speech.

The Temporal Relation and Bayesian Estimation. We further consider the temporal relation between the given poses of a character. We use time series analysis, which is modeled by a nonparametric approach, and *Bayesian inference* to improve the original nonparametric regression model. Hence, Bayesian regression based on RJMCMC sampling which can be applied to choose the suitable parameters, BARMA, and BLOESS are proposed for synthesizing the movements of multiple limbs simultaneously. Note that the Bayesian framework provides a more principled solution and better results than previous methods. Moreover, BARMA is flexible and appropriate for any data distribution for data prediction in the temporal domain.

An Application of Animating Passive Elements. This work could also be extended to focus on another novel multimedia application. That is animating pictures for passive element animation which are subject to natural forces like wind. We apply our proposed statistical approaches to animate passive elements. With the help of the physics, we are able to synthesize natural phenomena with time-varying motions from still photographs or paintings.

Furthermore, the results reveal that the generated 2D character animations with minimizing unnatural distortions. However, the prediction performance of the proposed method is considered strongly by correspondences of the input pictures. The proposed

method will not produce a reasonable result for the lower degree of similarity between the input pictures. For example, one is the back-face view, and the other is the front-face view. It may be solved through the extra information needed to handle the motion of rotation with users' interaction. Besides, animating passive elements in the painting with the apparent strokes may cause unnatural movements. We sample the element uniformly and use these samples' displacements to predict the whole element's movement. Thus, it lacks the information to indicate each stroke's orientation or its moving direction. It could be solved by providing additional vector fields or painterly art maps [75] to guide the flow of the strokes.

9.2 Future Work

Future studies should address the following issues to build upon the ideas presented here.

A More Efficient Fitting Method. As shown in Table 7.1, RJMCMC sampling takes a lot longer since all the simulations are run with a burn-in period of 5000 iterations of RJMCMC followed by 10000 samples. RJMCMC sampling is the most time-consuming part in our system and usually consumes more than 90 percent of the overall time for synthesizing limbs movement. Hence, Enhancing the performance and quality of the scattered ERBFs, LOESS fitting algorithm, and *Bayesian inference* based on RJMCMC is an important task.

A Motion Retargeting Module. *Time series* is used to analyze and estimate the motion trajectory of the character. The estimated motion trajectory could be applied to retarget the motion onto any similar humans or human-like characters. The motion retargeting technique would empower a much quicker animation production.

Virtual Human Generation. Deforming characters in a 2D image has received lots of interests. Moreover, it is very useful for advanced intelligent multimedia applications for next generation environments utilization. Thus, the proposed method is especially suitable for animation production or intelligent multimedia applications, such as virtual human generation. The created virtual human can be treated as the spokesman or

substitute in various services of the next generation application domain, such as remote education, remote diagnosis, network gaming, virtual shopping, digital photo frame, video conference, and so on.

Furthermore, there are a number of research opportunities in other multimedia applications for the future.

Progressive Image Abstraction. In non-photorealistic rendering, a progressive image abstraction approach could be proposed by developing two-scale decomposition mentioned in Chapter 4. More specifically, let I denote the input image for which we would like to construct k -level decomposition. B_{L1} is the 1-level base layer, which is the filtered result of I by bilateral filtering. The corresponding detail layer D_{L1} is the division of the original image by B_{L1} . Then B_{L2} is 2-level base layer, which is the filtered result of B_{L1} by bilateral filtering. The corresponding detail layer D_{L2} is the division of B_{L1} by B_{L2} . For this reason, B_{Lk} is k -level base layer, which is the filtered result of $B_{L(k-1)}$ by bilateral filtering. The corresponding detail layer D_{Lk} is the division of $B_{L(k-1)}$ by B_{Lk} . Moreover, the relationship between I , an arbitrary base layer, and the corresponding detail layer are reversible. Based on the multi-scale decomposition, users can control various degree of detail of the scene in a spatially varying manner. That is, users can provide more detail in area of interest.

A Bobbing Boat and Flowing Clouds Simulation. A boat on the water surface simulated in Section 8.3.2 is rolling and moving vertically downward. It is almost in oscillatory motion. We can approximate the motion of the boat. According to the simulated wave shape of water, the displacement of the boat on the carrier wave is estimated through affine transformation. More specifically, we estimate the waterline by naval architecture or fluid mechanics. The displacements of the points sampled along the waterline equal to the displacements of the water particles at these samples. Then *nonparametric regression* mentioned in Section 3.1 and Section 3.2 could be use to fit the movement of the whole boat. Besides, another common passive element for scenic pictures could be simulated is cloud. We could specify a translational and rotational motion to cloud instead of *simple harmonic motion*. For global cloud movement, for example, a gradual translation along the wind direction emulates prevailing wind effects. Moreover, particles forming the cloud may rotate with velocity determined by the wind. Hence, we sample the contours of clouds and estimated the displacements of samples

through affine transformation. The proposed statistical approaches are applied to fit the motions of clouds.



Bibliography

- [1] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," In *Proceeding of ACM SIGGRAPH 2000*, pp. 157-164, 2000.
- [2] N. Arad, N. Dyn, D. Reifeld, and Y. Yeshurun, "Image warping by radial basis functions: applications to facial expressions," *CVGIP Graph Models Image Processing*, vol. 56, no. 2, pp.161-172, 1994.
- [3] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 1-8, 2007.
- [4] W. Baxter and K.-I. Anjy, "Latent doodle space," *Computer Graphics Forum*, vol.25, no. 3, pp. 477-485, 2006.
- [5] C. M. Bishop, *Neural networks for pattern recognition*. MIT Press, 1995.
- [6] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," *Computer Graphics Forum*, vol. 22, no. 3, pp. 641-650, 2003.
- [7] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213-230, 2008.
- [8] M. Brand, "Voice puppetry," in *Proceedings of ACM SIGGRAPH 1999*, pp. 21-28, 1999.
- [9] M. Brand and A. Hertzmann, "Style machines," In *Proceeding of ACM SIGGRAPH 2000*, pp. 183-192, 2000.
- [10] C. Busso, Z. Deng, M. Grimm, U. Neumann, and S. S. Narayanan, "Rigid head motion in expressive speech animation: analysis and synthesis," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 8, pp. 1075-1086, 2007.
- [11] C. Busso and S. S. Narayanan, "Interrelation between speech and facial gestures in emotional utterances: a single subject study," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 8, pp. 2331-2347, 2007.
- [12] J. Chai and J. K. Hodgins, "Constraint-based motion optimization using a statistical dynamic model," *ACM Transactions on Graphics*, vol. 26, no. 3, article 8, 2007.
- [13] C. W. S. Chen, R. E. McCulloch, and R. S. Tsay, "A unified approach to estimating and modeling linear and nonlinear time series," *Technical Report*. Graduate School

of Business, University of Chicago, 1996.

- [14] M.-H. Chen, Q.-M. Shao, and G. Ibrahimj, *Monte Carlo methods in Bayesian computation*. Springer, 2000.
- [15] S. E. Chen and L. William, "View interpolation for image synthesis," in *Proceeding of ACM SIGGRAPH 1993*, pp. 279-288, 1993.
- [16] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. Salesin, and R. Szeliski, "Animating pictures with stochastic motion textures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 853-860, 2005.
- [17] E. Chuang and C. Bregler, "Mood swings: expressive speech animation," *ACM Transactions on Graphics*, vol. 24, no. 2, pp. 331-347, 2005.
- [18] C.-H. Chuang, S.-F. Tsai, and C.-J. Kuo, "Cartoon animation and morphing by using the wavelet curve descriptor," in *Proceedings of 1994 IEEE International Conference on Image Processing*, pp. 666-670, 1994.
- [19] C. N. DeJuan and B. Bodenheimer, "Re-using traditional animation: methods for semi-automatic segmentation and inbetweening," in *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 223-232, 2006.
- [20] Z. Deng and U. Neumann, "Efase: expressive facial animation synthesis and editing with phoneme-isomap controls," in *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 251-260, 2006.
- [21] T. F. Ezzat, G. Geiger, and T. Poggio, "Trainable video realistic speech animation," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 388-398, 2002.
- [22] J. Fan and Q. Yao, *Nonlinear time series: nonparametric and parametric methods*. Springer, 2005.
- [23] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," *ACM Transactions on Graphics*, vol. 27, no. 3, article 67, 2008.
- [24] S. Forstmann, J. Ohya, A. Krohn-Grimberghe, and R. McDougall, "Deformation styles for spline-based skeletal animation," in *Proceeding of SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 141-150, 2007.
- [25] T. Fu and H. Foroosh, "Expression morphing from distant viewpoints," in *Proceeding of the International Conference on Image Processing*, pp. 3519-3522, 2004.
- [26] A. Galata, N. Johnson, and D. Hogg, "Learning variable length Markov models of behavior," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 398-413, 2001.
- [27] B. Glocker, N. Paragios, K. Komodakis, G. Tziritas, and N. Navab, "Optical flow estimation with uncertainties through dynamic MRFs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

- [28] E. Goldstein and C. Gotsman, "Polygon morphing using a multiresolution representation," in *Proceedings of Graphics Interface 1995*, pp.247-254, 1995.
- [29] R. Herbrich, *Learning kernel classifiers theory and algorithms*. The MIT Press, 2002.
- [30] A. Hornung, E. Dekkers, and L. Kobbelt, "Character animation from 2D pictures and 3D motion data," *ACM Transactions on Graphics*, vol. 26, no. 1, article 1, 2007.
- [31] T. Igarashi, T. Moscovich, J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 1134-1141, 2005.
- [32] A. K. Jain, *Fundamentals of digital image processing*. Prentice-Hall, 1989.
- [33] Y. Jang, R. P. Botchen, A. Lauser, D. S. Ebert, K. P. Gaither, and T. Ertl, "Enhancing the interactive visualization of procedurally encoded multifield data with ellipsoidal basis functions," *Computer Graphics Forum*, vol. 25, no. 3, pp. 587-596, 2006.
- [34] L. Lempitsky, S. Roth, C. Rother, "FusionFlow: discrete-continuous optimization for optical flow estimation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [35] Y. Li and D. Huttenlocher, "Learning for optical flow using stochastic optimization," in *Proceedings of the 10th European Conference on Computer Vision*, no. 2, pp. 379-391, 2008.
- [36] P. Litwinowicz and L. Willams, "Animating images with drawings," in *Proceeding of ACM SIGGRAPH 1994*, pp. 409-412, 1994.
- [37] Z. Liu, Y. Shan, and Z. Zhang, "Expressive expression mapping with ratio images," in *Proceedings of ACM SIGGRAPH 2001*, pp. 271-276, 2001.
- [38] S. Osher, J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12-49, 1988.
- [39] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving gradients: a path-based method for plausible image interpolation," *ACM Transactions on Graphics*, vol. 28, no. 3, article 42, 2009.
- [40] H. McGurk and J. MacDonald, "Hearing lips and seeing voices," *Nature*, vol. 264, no. 5588, pp. 746-748, 1976.
- [41] A. Menache, *Understanding motion capture for computer animation and video games*. Morgan Kaufmann Publishers Inc., 1999.
- [42] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. Wiley, 2006.

- [43] R. Mukundan, S. H. Ong, and P. A. Lee, "Image analysis by tchebichef moments," *IEEE Transactions on Image Processing*, vol. 10, no. 9, pp. 1357-1364, 2001.
- [44] T. Ngo, D. Cutrell, J. Dan, B. Donald, L. Loeb, and S. Zhu, "Accessible animation and customizable graphics via simplicial configuration modeling," in *Proceedings of ACM SIGGRAPH 2000*, pp. 403-410, 2000.
- [45] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in speech recognition*, pp. 267-296, 1990.
- [46] W. J. Palm, *Modeling, analysis, and control of dynamic systems*. Wiley, 1998.
- [47] J. Park and I. W. Sandberg, "Nonlinear approximations using elliptic basis function networks," in *Proceedings of the 32nd Conference on Decision and Control*, pp. 3700-3705, 1993.
- [48] V. Ranjan and A. Fournier, "Matching and interpolation of shapes using unions of circles," *Computer Graphics Forum*, vol. 15, no. 3, pp.129-142, 1996.
- [49] X. Ren, "Local grouping for optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [50] C. Rose, M. F. Cohen, and B. Bodenheimer, "Verbs and adverbs: multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32-40, 1998.
- [51] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309-314, 2004.
- [52] D. Ruprecht and H. Müller, "Image warping with scattered data interpolation," *IEEE Computer Graphics and Applications*, vol. 15, no. 2, pp. 37-43, 1995.
- [53] S. Schaefer, T. Mcphail, and J. Warren, "Image deformation using moving least squares," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 533-540, 2006.
- [54] K. Scherbaum, M. Sunkel, H.-P. Seidel, and V. Blanz, "Prediction of individual non-linear aging trajectories of faces," *Computer Graphics Forum*, vol. 26, no. 3, pp. 285-294, 2007.
- [55] T. Sederberg and E. Greenwood, "A physically based approach to 2D shape blending," in *Proceedings of ACM SIGGRAPH 1992*, pp. 25-34, 1992.
- [56] S. M. Seitz and C. R. Dyer, "View morphing," in *Proceeding of ACM SIGGRAPH 1996*, pp. 21-30, 1996.
- [57] J. A. Sethian, *Level set methods*. Cambridge University Press, 1996.
- [58] J. A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge University Press, 1999.

- [59] A. Shashua and T. Riklin-Raviv, "The quotient image: class based re-rendering and recognition with varying illuminations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 129-139, 2001.
- [60] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications: with R examples*. Springer, 2006.
- [61] J. D. Shutler and M. S. Nixon, "Zernike velocity moments for sequence-based description of moving features". *Image and Vision Computing*, 2006; vol. 24, no. 4, pp. 343-356, 2006.
- [62] D. Sun, S. Roth, J. P. Lewis, and M. J. Black, "Learning optical flow," in *Proceedings of the 10th European Conference on Computer Vision*, no. 3, pp. 83-97, 2008.
- [63] B. H. Thomas and P. Calder, "Animating direct manipulation interfaces," in *Proceedings of the 8th ACM Symposium on User Interface Software and Technology*, pp. 3-12, 1995.
- [64] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceeding of the International Conference Computer Vision 1998*, pp. 839-846, 1998.
- [65] W. Trobin, T. Pock, D. Cremers, H. Bischof, "Continuous energy minimization via repeated binary fusion," in *Proceedings of the 10th European Conference on Computer Vision*, no. 4, pp. 677-690, 2008.
- [66] S. Vedula, S. Baker, and T. Kanade, "Image-based spatio-temporal modeling and view interpolation of dynamic events," *ACM Transactions on Graphics*, vol. 24, no. 2, pp. 240-261, 2005.
- [67] Y. Wang, K. Xu, Y. Xiong, and Z.-Q. Cheng, "2D shape deformation based on rigid square matching," *Computer Animation and Virtual Worlds*, vol. 19, no. 3-4, pp. 411-420, 2008.
- [68] O. Weber, M. Ben-Chen, and C. Gotsman, "Complex barycentric coordinates with applications to planar shape deformation," *Computer Graphics Forum*, vol. 28, no. 2, pp. 587-397, 2009.
- [69] A. Witkin and Z. Popović, "Motion warping," in *Proceeding of ACM SIGGRAPH 1995*, pages 105-108, 1995.
- [70] G. Wolberg, "Image morphing: a survey," *The Visual Computer*, vol. 14, no. 8-9, pp. 360-372, 1998.
- [71] C. R. Wylie, *Advanced engineering mathematics*. McGraw-Hill, 1975.
- [72] G. Wyszecki and W. S. Styles, *Color science: concepts and methods, quantitative data and formulae*. Wiley, 1982.
- [73] L. Xu, J. Chen, and J. Jia, "Segmentation based variational model for accurate optical flow estimation," in *Proceeding of the 10th European Conference on*

Computer Vision, no. 1, pp. 671-684, 2008.

- [74] X. Xu, L. Wan, X. Liu, T.-T. Wong, L. Wang, and C.-S. Leung, “Animating animal motion from still,” *ACM Transactions on Graphics*, vol. 27, no. 5, article 117, 2008.
- [75] C.-R. Yan, M.-T. Chi, T.-Y. Lee, and W.-C. Lin, “Stylized rendering using samples of a painted image,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 468-480, 2008.
- [76] H.-B. Yan, S.-M. Hu, R. R. Martin, and Y.-L. Yang, “Shape deformation using a skeleton to drive simplex transformations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 3, pp. 693-706, 2008.
- [77] T. Yotsukura, S. Morishima, and S. Nakamura, “Model-based talking face synthesis for anthropomorphic spoken dialog agent system,” in *Proceeding of the 11th ACM International Conference on Multimedia*, pp. 351-354, 2003.
- [78] G. A. Young and R. L. Smith, *Essentials of statistical inference*. Cambridge University Press, 2005.
- [79] AIC Anime International Co. Inc. <http://www.aicanime.com/>.



Vita

Yun-Feng Chou was born on October, 1978, in Taipei, Taiwan, Republic of China. He received the B.S. and M.S. degrees in computer science from Soochow University in 2001 and 2003, respectively. In 2003, he decided to study a Ph.D. degree and joined the Computer Graphics Laboratory at the Department of Computer Science, National Chiao Tung University. During the Ph.D. degree, he has diligently pursued to his research about multimedia techniques and applications, such as image morphing, view morphing, motion synthesis, and shape deformation. After several years, he finally received the Ph.D. degree in computer science from National Chiao Tung University in 2010. His research interests mainly include computer graphics, computer vision, signal processing, image processing, and content-based multimedia indexing and retrieval algorithm.

