Distributed computing and NURBS in computer-assisted in-betweening

Zhang, Jian Feng

2005

Zhang, J. F. (2005). Distributed computing and NURBS in computer-assisted in-betweening.
Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/47492>

<https://doi.org/10.32657/10356/47492>

Nanyang Technological University

Downloaded on 26 Apr 2023 23:35:46 SGT

a357893

Distributed Computing and NURBS in Computer-Assisted In-Betweening

Zhang Jian Feng

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Master of Engineering

2005

TR
897.7
Z63
2005

Acknowledgements

First of all, I would like to express my grateful thanks to *Assoc Prof. Seah Hock Soon*, my research supervisor, who guided me patiently and enthusiastically through the academic research leading to this thesis. Without his great effort, my research work could not have been done properly and the thesis could not have been completed successfully. Since my research work started in October 2001, he kindly set and adjusted the research direction that I should follow. His research attitude and working ability gave me a great impression, which set up an example of how to become a good researcher to me. His valuable advice and sincere help had kept me on a firm direction towards the success. His rich and extensive knowledge on computer graphics and animation gave me a lot of precious suggestions and recommendations. Saying “thank you” is not enough to express myself to his sincere supervision, kindness and help.

My grateful thanks also go to *Assistant Prof. Tian Feng*, who kindly agreed to be my unofficial co-supervisor. He gave me wise advice and helpful assistance on my research work and my personal life as well. He is enthusiastic and provided a lot of insightful ideas to my research work. Without his help and assistance, my work could not have been done in a smooth way. His mindset and ways of doing things impacted me greatly for further study, work and future life as well.

My special and grateful thanks should go to Mr. Dilip Krishnan, who is the managing director of the company I was working for at the moment of doing this research. Throughout the time I was working on my research, he showed great understandings and gave me a lot of knowledge related to image processing. Whenever I approached him for any doubts, he was so patient to explain and illustrate the concepts to me. Without his help, I would face much more difficulty in my research.

Last but not least, my sincere thanks go to my friends Lu Ji, Li Li and Chen Quan. They helped me a lot in analyzing and understanding the problems I faced in my research. They also searched and collected valuable papers for me. Without their effort and help, my research work would be much more difficult than it was.

Contents

Acknowledgements	i
Table of Contents	ii
Abstract	v
List of Tables	vi
List of Figures	vii
1 Introduction	1
1.1 Background	1
1.2 Objectives	2
1.3 Research Overview	3
1.4 Thesis Organization	3
2 Survey and Problem Statements	5
2.1 Cartoon Animation Process	5
2.2 Animation Systems	7
2.2.1 Ink and Paint Systems	7
2.2.2 Automated In-Betweening Systems	9
2.3 Computer-Assisted In-Betweening	11
2.4 Feature-Based Matching Algorithm (FBA)	13
2.4.1 Displacement Vector (DV)	13
2.4.2 FBA Principle	14

2.4.3	Summary of FBA	17
2.4.4	Modified Feature-Based Matching (MFBA)	18
2.5	Analysis of MFBA-IB	18
2.5.1	Processing Performance	20
2.5.2	In-Between Motion Quality	22
2.6	Summary	26
3	Solution Analysis	27
3.1	Analysis of Distributed Image Processing	27
3.1.1	Resource Management	28
3.1.2	Parallel Image Processing on SMP	28
3.1.3	Distributed Image Processing	29
3.1.4	Related Work	31
3.1.5	Discussion	32
3.2	Computer Animation and NURBS	33
3.2.1	Animation System and Methods	33
3.2.2	Curve Representation	34
3.2.3	OpenGL and NURBS	37
3.2.4	Interactive Image Segmentation	38
3.2.5	Related Work	38
3.2.6	Discussion	39
3.3	Suggested Improvements	40
4	Distributed In-Betweening	41
4.1	Distributed and Parallel Communication Model	42
4.1.1	Design Overview	42
4.1.2	Communication Model Implementation	44
4.1.3	Discussion	46
4.2	Speed-up for MFBA-IB	46
4.2.1	Intra-Frame Mode	47

4.2.2	Result Analysis	50
4.2.3	Inter-Frame Mode	55
4.3	Summary	57
5	NURBS-Improved Matching	58
5.1	Line Thinning	59
5.2	Interactive Image Segmentation	62
5.3	Line Fitting via NURBS Modeling	62
5.3.1	Image Pixel Sampling	64
5.3.2	Curve Pixel Connectivity	66
5.3.3	Pixel Recursive Traversal Algorithm	70
5.3.4	Result Analysis and Discussion	75
5.4	NURBS In-Betweening	79
5.5	Interactive Feature Point Extraction	80
5.6	Generation of In-Between Curve	81
5.7	Integrated Solution for In-Betweening	83
5.8	Discussion	85
6	Experimental Results and Discussion	87
6.1	Graphical User Interface	87
6.2	Interactive NURBS In-Betweening	88
6.3	Summary	95
7	Future work	96
8	Conclusions	99
	Bibliography	102

Abstract

Producing in-between frames in 2D animation is a labor-intensive process in current cartoon production. An existing *Modified Feature Based Algorithm* (MFBA) automatically generates in-between frames between two key frames based on feature matching. This is referred to as MFBA-IB. However, the MFBA is computationally intensive. Even with current compute power, it is difficult to have massive in-between frame production by running the algorithm in a single computer. In order to speed up the processing of key frames, a distributed algorithm is proposed. The distributed algorithm has the potential to increase MFBA-IB processing throughput and it is implemented with our distributed communication model in Linux platform. The proposed distributed communication model is a transparent interface for underlying machine architecture or parallel programming and it eases the building of a distributed MFBA-IB system. A graphical user interface program is designed to demonstrate the use of these library routines for a distributed MFBA-IB system and in-between results and throughput of such a system are analyzed.

The matching result of MFBA used by MFBA-IB does not always yield complete satisfaction insofar as the image quality of the generated in-between frames is concerned. It is observed that certain image segments were not in-betweened well. An interactive *Non-Uniform Rational B-Spline* (NURBS) matching algorithm is proposed to solve such in-between problem in MFBA-IB. Expected improvements in the in-between are achieved. Effort in automatic classification of curves yielded certain results. The interactive NURBS matching algorithm could be extended to be automatic in the future and incorporated into the distributed in-betweening process to work together with MFBA-IB.

Keywords: Distributed Image Processing, Computer Clustering, MFBA, MFBA-IB, SMP, In-betweening, NURBS, Line Fitting, Feature Point Extraction, OpenGL, GLUI.

List of Tables

2.1	MFBA Processing Performance for Different Image Resolutions	20
2.2	MFBA Processing Performance for Different Image Contents	20
4.1	Distributed MFBA Processing versus Local MFBA Processing	48

List of Figures

1.1	Thesis Structure	4
2.1	The Ink and Paint Process	6
2.2	Displacement Vector Illustration	13
2.3	In-Between Frame Generation	14
2.4	MFBA Hierarchical Processing	17
2.5	Displacement Vector Field	18
2.6	MFBA In-Between Results - Sea	19
2.7	Different Image Contents with 720x586 Resolution	21
2.8	MFBA In-Between Sequence - Girl	23
2.9	MFBA In-Between Sequence - Cloud	24
2.10	MFBA In-Between Sequence - Fire	25
3.1	Symmetric Multiprocessor System	28
3.2	Mapping of the Parallel Construct	32
3.3	NURBS Animation System	34
3.4	A Bézier Curve of Degree 3.	35
3.5	Distortion in Length when Rotation is Simulated Linearly	39
4.1	Distributed Communication Model	43
4.2	Distributed Communication Components	43
4.3	Interactive Communication Diagram	45
4.4	Distributed Communication Structure	45
4.5	Central Partition of a Cartoon Character	47

4.6	Possible Performance Curves for MFBA	48
4.7	Distorted Distributed In-Between Frames	49
4.8	Non-Central Partition of A Cartoon Character	50
4.9	The Distributed MFBA-IB Results of Central Segmentation in <i>Intra-Frame</i> Mode	53
4.10	The Distributed MFBA-IB Results of Independent Characters in <i>Intra-</i> <i>Frame</i> Mode	54
4.11	The Composition of Independent Characters in Key Frames for Distributed Processing	54
4.12	The Distributed MFBA-IB Results in <i>Inter-Frame</i> Mode	56
5.1	Original Image versus Binarized Image	59
5.2	SUSAN Thinning Rules	60
5.3	SUSAN Thinned Image	61
5.4	Sampling Grid On An Image	64
5.5	Searching An On-Curve Pixel	65
5.6	Mapped Images of Valid Sampled Pixels	66
5.7	Pixel Connectivity	67
5.8	Coordinate System	68
5.9	Connectivity Search for A Valid Sampled Pixel	68
5.10	Branching Curve	70
5.11	Result of Connectivity Search in Eight Directions	73
5.12	Refined Result of Automatic Reconstruction	74
5.13	Key Frames - Wave	74
5.14	Automatically Reconstructed Key Frames - Wave	75
5.15	Auto-NURBS In-Between Test Results	76
5.16	Curve Reconstruction Error	77
5.17	Automatically Reconstructed Frame	78
5.18	Interactive Feature Point Extraction	80
5.19	NURBS Curves and Feature Points	82

5.20 Curve Removal 83

5.21 Integrated In-Betweening Process 84

5.22 Segmentation of Key Frames 85

6.1 Scheduling GUI for Distributed MFBA-IB Processing 88

6.2 GUI for Interactive NURBS In-Between Matching 89

6.3 Interactive Segmentation and Curve Fitting On Two Key Frames 90

6.4 NURBS In-Between Frames 91

6.5 MFBA In-Between Frames 91

6.6 Interactive NURBS In-Between Frames 93

6.7 MFBA In-Between Frames 94

7.1 Auto-NURBS In-Between Matching 97

Chapter 1

Introduction

In this chapter, our research standpoint is positioned by introducing the background of computer-assisted animation. The research scope of our work on computer-assisted cartoon animation is defined by presenting our objectives clearly. The limitations of an existing in-between algorithm (MFBA) provide the impetus for our research. A brief overview of distributed and parallel image processing as well as the computer graphics technique (NURBS) lay the ground for our research direction to improve MFBA-IB.

1.1 Background

In the past few decades, there has been quite a number of research topics on computer-assisted animation. Some systems have been built in an attempt to release the burden of animation artists in order to generate in-between frames from key frames. Key frames outline the motion for a sequence at certain key points in an action [4]. Although there has been some success and a great deal of optimism, the promise of higher output and quality using a computer has not been fully realized. The principal difficulty lays in the fact that hand drawings are two dimensional renditions of three dimensional characters, therefore depth information is not available in the images. It is a difficult image correspondence problem [6, 8] to infer the original object depth from its projections.

Various computer-assisted cartoon animation algorithms have been proposed in the worldwide research as in [1, 9, 11, 12, 34]. Among them, Seah [1] developed the *Modified*

Feature Based Algorithm (MFBA) to match a pair of key frames. Based on the matching result, the MFBA-IB (MFBA-based in-betweening) algorithm generates a parameter-driven number of in-between frames. This algorithm provides a generic solution aimed at improving productivity and quality of cartoon production with as less user manipulation as possible. As animation comes in many different forms and varieties, specific styles need special attention. Some of the problems with the generic in-between algorithm (MFBA) will be addressed in this thesis.

MFBA uses hierarchical feature-based image matching [2] to compare a pair of key frames and to determine the corresponding pixel displacement vectors automatically. Like many of the other image processing algorithms, it is a computationally intensive process. As an example, with an existing computer system having one 1.0 GHz Intel CPU, it can take five minutes or more to match two key frames of 720x586 resolution. This is not practical when there is a large number of key frames to be processed, especially in the cartoon production where high production throughput is required. A problem with MFBA-IB is that the generated in-between results are not perfect in image segments where disparate pixel motion changes occur, e.g. the matching of opening and closing eyes, where the upper and lower eyelids move in upward and downward directions respectively.

1.2 Objectives

One of the objectives of this project is to explore the possible ways of speeding up the MFBA in-between algorithm developed by *Seah* [2]. A distributed and parallel communication model will be investigated to handle the massive computation needed by MFBA when it becomes practical for use by cartoonists in the future. Another objective is to develop a complementary algorithm to solve the discouraging in-between motion problem in areas where the pixel displacement vectors are not determined properly by MFBA due to lack of features, inconsistency in drawings, complicated drawings, and noise.

1.3 Research Overview

This overview section introduces the areas that are focused in this research. A clear mindset about the research aspect of these areas is expected to be established.

Distributed and parallel processing have been widely used to cater for computationally intensive work. The high computational requirement and the time constraint associated with many image processing applications have resulted in considerable interest in the utilization of parallel computing resources [30]. There are many implementations at different application levels [21, 29, 28, 32, 30, 31]. It also has the potential to achieve high performance at a reasonable cost [28] for many animation applications.

NURBS is the abbreviation of *Non-Uniform Rational B-Spline* curves. NURBS is the de facto standard technique [35, 36] to model curves and surfaces in the industry. A lot of applications, e.g. CAD, use NURBS to model curves and surfaces [43].

When applying NURBS to improve MFBA-IB, the idea is to use it to model the image portions that cannot be in-betweened properly by MFBA, for example, due to complex or disparate motion changes in the matching portions. Therefore, the attempt is to apply computer graphics techniques like NURBS in the in-betweening process to achieve a better matching.

1.4 Thesis Organization

Figure 1.1 shows the structure of the thesis. A brief introduction is given in Chapter 1 on the research background, objectives, overview and thesis organization. Chapter 2 provides background information on cartoon animation system and computer-assisted in-betweening. It also analyzes the principle of MFBA-IB and reveals the problems that need to be tackled.

Through the study of distributed image processing and NURBS, Chapter 3 points out how these two techniques can be applied to improve MFBA-IB.

Chapter 4 contains the design and implementation of our distributed communication model API for speeding up MFBA-IB.

Figure 1.1: Thesis Structure

Chapter 5 illustrates the design and implementation of our NURBS-improved matching for MFBA-IB.

Chapter 6 presents and discusses the experimental results for our distributed API and NURBS-improved matching algorithm as well as the design and implementation of graphical user interfaces.

Chapter 7 discusses the possible future work for in-betweening process.

Chapter 8 concludes our research achievements in this thesis and points out the limitations with the current in-between research work.

Chapter 2

Survey and Problem Statements

In-betweening is an important step no matter in traditional cartoon animation process or in computer-assisted animation process. Such importance can be seen in review of these two animation processes in the following sections. Since computer becomes a powerful tool for people to accomplish labor-intensive work nowadays, people are keen to computerize cartoon animation process and improve production efficiency. In-betweening is an intelligent process and it challenges people to develop all kinds of algorithms and systems to make it computerized. Among these algorithms, the **Feature-Based matching Algorithm** (FBA) and its enhanced version, **Modified Feature-Based matching Algorithm** (MFBA), are analyzed and illustrated in detail. The strengths and weaknesses of these two algorithms led to the problems that are to be tackled in this research. For simplicity, it should be noted that the discussion and analysis on MFBA also apply to FBA if not otherwise specified.

[Keywords: **DV** - *Displacement Vector*, **FBA** - *Feature-Based matching Algorithm*, **MFBA** - *Modified Feature-Based matching Algorithm*]

2.1 Cartoon Animation Process

The traditional cartoon animation system uses a paper-based approach, which is tedious and labor-intensive. Figure 2.1 shows a simplified traditional animation process [13].

As it can be seen, most steps involve an exposure sheet, which lists all the frames in

Figure 2.1: The Ink and Paint Process

a scene. The lines include the phoneme pronounced by each character and the order and position in which the camera will shoot the figures and the background.

Each scene requires a set of stages, of which only the background can be painted in parallel with the animation stages (from key-frame to paint). The stages include:

- **Story Board:** Splits script into scenes with dialog and music.
- **Sound Track:** Records dialog and music in a prototype form.
- **Sound Detection:** Fills the dialog column of an exposure sheet.
- **Layout:** Manages the drawing of backgrounds and main character positions, with specifications for camera movements and other animation characteristics.
- **Background Painting:** Paints the background according to the layout.
- **Key Frame Animation:** Draws extreme positions of characters as specified by the layout. Provides instructions for the in-betweeners.

- **In-Betweening:** Draws the missing frames according to the key-frame animator's instructions.
- **Cleaning:** Cleans up the drawings to achieve final quality of the strokes.
- **Paint:** Photocopies the clean drawings onto acetate celluloid (cels) and paints zones with water color.
- **Check:** Verifies animation and backgrounds according to the lay-out and approves for shooting.
- **Record:** Shoots frame-by-frame on film or video, using a rostrum (explained later).

It can be seen that the traditional animation process requires many stages to complete a cartoon movie from start to end. Since computer systems become more and more powerful and cost-effective nowadays, people have been trying to automate cartoon animation production by developing algorithms on computer systems. The in-betweening process is an intelligent process that attracts more attention from people, it is also the part under our research consideration.

2.2 Animation Systems

Although people have automated the cartoon animation production significantly with the aid of powerful computer systems, the two main types of animation systems, which are identified as *Ink and Paint* and *Automated In-Betweening* [13], are still not fully computerized. In both systems, all artwork is drawn on paper and later digitized and managed by computer after the cleaning stage. These two types of systems are introduced in the following sections.

2.2.1 *Ink and Paint* Systems

Ink and Paint systems perform the following steps [13], starting from scanned images of each animated character:

- Remove noise from images;
- Close gaps between strokes to prepare for paint;
- Paint using seed-fill;
- Compose images of characters and backgrounds, analyzing zoom, rotation and pans as in a *rostrum*;
- Record on film or video.

The advantages of *Ink and Paint* systems are:

- Speeding up painting - A computer can speed up the hand painting process, which is the most tedious task in traditional animation. The average speed is 100 drawings per day depending upon the number of zones per drawing and the number of colors being used. In traditional animation, a maximum of 40 drawings can be painted per day, with an average of 20;
- Remove limits on the number of layers - Computers can handle more than the five composition layers found in traditional animation, which is limited by the opacity of acetate celluloid;
- Simplify shooting - A virtual rostrum is easier to use than a real rostrum;
- Allow images from other sources - Highest quality backgrounds are still hand-painted and scanned in. However, images from other sources - like paint programs, rendering programs or digitized live action - can also be imported.

The disadvantages of *Ink and Paint* systems are:

- Increase the cleaning time - Films require huge numbers of drawings, in the order of tens of thousands. Computer systems can add manual actions to traditional work and machine processing time can be slow. For example, *Ink and Paint* systems require cleaner drawings than traditional animation does. Animators must spend extra time cleaning to avoid having the computer spend more time filtering, which increases the cost of the hand-cleaning stage;

- Consume large resources - Reuses, though possible, are only made by large studios. With the exception of backgrounds, most images are usually removed from disk after recording. Pixel images require a lot of storage, especially if the final format is 35mm film. The technology required to manage such images remains expensive and without a planned long term production (over two years), the extra storage and network cost is not guaranteed to be balanced by the increase in productivity;
- Add medium change costs - Finally, several marginal costs result from the transition between paper and computer:
 - A special staff is required to scan and check drawings;
 - The exposure sheet must be typed into the system;
 - Information is not usually propagated back from the computer to the paper world. Since artists do not receive feedback, some problems recur.

The overhead of transferring between paper and computer accounts for a minimum of 10% for a well-organized process to 25% or more for less well-organized processes in the final cost.

2.2.2 *Automated In-Betweening Systems*

Automated In-Betweening systems begin with a scanned key frame and perform the following steps:

- Clean and vectorize, sometimes using semi-automatic methods;
- Match pairs of drawings or match drawings to a template;
- Paint, if colors are not part of the template;
- Interpolate in-betweens according to an exposure sheet;
- Render and compose images of characters and backgrounds, applying zoom, rotation and pans as in a rostrum;

- Record on film or video;
- Put in database for reuse.

The advantages of *automated in-betweening* systems are:

- Reduce the number of hand-drawn in-betweens - Computer-assisted in-betweening is designed to reduce the number of hand-drawn in-betweens and increase reuse of animation. Once a drawing has been vectorized and the correspondence specified, an animation can be tuned precisely. Some actions can be reused at different tempos, which also decreases the number of hand-drawn drawings;
- Allow procedural rendering - Since zones and edges composing successive key drawings are matched, the interpolation process can automatically maintain space coherence between procedurally computed attributes used for rendering. For example, the checked texture of a character's shirt will be correctly animated when he moves his arms. Unlike with traditional animation, many special effects can be animated automatically.

Like *Ink and Paint* systems, *Automated In-Betweening* systems allow an unlimited number of layers, an easier shooting and access to images from other sources. Painting takes a small portion of the key frame matching process. Although *Automated In-Betweening* systems have the above advantages, they also have certain disadvantages that need to be noticed:

- Change the nature of in-betweening and limit its complexity - Crafting an animation with an *Automated In-Betweening* system is slow, it is found that inputting and tuning simple in-betweens take about the same time as drawing them by hand. Even with recent advances in algorithms, it remains difficult to match a drawing to a template or two successive drawings to each other;
- Require a skill in modeling - Template-based systems require both 3D modeling as well as traditional animation skills. Crafting a good model is expensive and must

be taken into account when estimating the cost of the production. Given this cost, these systems are only cost-effective for productions with few characters;

- Does not provide as much reuse as expected - Animations are less reusable than might be expected. Reuse must be planned at the storyboard stage, which is not integrated with the computer part of the process. All these problems contribute to the perception by animation studios that computer-assisted in-between programs produce poor quality animation.

Moreover, like *Ink and Paint* systems, *Automated In-Betweening* systems also add medium change costs. However, despite the disadvantages of computer-assisted in-betweening systems that have been discussed [13], its advantages and difficulties challenge people to develop various algorithms and systems to automate in-betweening process and therefore reduce work load.

2.3 Computer-Assisted In-Betweening

From past research on computer-assisted in-betweening, a number of models and systems have been built to assist animation artists to ease the burden of generating in-between frames from key frames [9, 10, 11, 12]. Each of these systems or models was developed based on desired objectives, attempting to solve some aspects of the in-betweening problem from their own perspectives.

Fabian [9] proposed a solution based on novel 2.5D modeling and animation techniques. Within the context of a multi-level approach, the solution starts from basic 2D drawing primitives (curves) at level 0, to explicit 2.5D modeling structures at level 1, and the inclusion of 3D information by means of skeletons at level 2, to high-level deformation tools at level 3.

Burtnyk and Wein [10] proposed a skeleton control technique for animators to develop a complex motion sequence by animating a stick figure representation of an image. This technique tries to eliminate the shortcomings in key frame animation, which results

from incomplete control of motion dynamics, both in complexity and in smoothness or continuity.

In the computerized in-betweening process, William [11] proposed an approach to allow animators more control over an interpolation sequence than existing key frame techniques. It specifies a set of new constraints called *moving points* in addition to the existing set of key frame constraints. The animators are allowed to specify correspondences between key frames in an intuitive manner, hence the interpolation between key frames can be controlled with as many moving points as necessary.

In-betweens are not just interpolations between key frames [12]. There is some general knowledge that the in-betweener has to consider:

- the background knowledge of the physical rules of the world,
- the expert knowledge when to bend or ignore these rules and
- the idea what emotions should be evoked by the animation.

Owing to the reasons listed, the fully automatic in-betweening would probably remain an unreachable goal at least in the near future. As such, Kort [12] proposed an approach which does not aim at automatic in-betweening but at computer-assisted in-betweening. The approach proposed is not applicable to all kind of cel animations. It has some pre-defined assumptions. The summary of the algorithm [12] is:

- the approach utilizes the model of *stroke chains* to represent the outline of regions in a cartoon image;
- then it defines a set of rules to regulate which stroke in one drawing can be matched to which stroke(s) in the other drawing;
- a cost function is used to measure the quality of a pairwise matching of stroke chains;
- an optimization algorithm works out the best assignments between stroke chains in different drawings. The in-between stroke chains are generated from the best assignments.

The MFBA matching algorithm [2] is a hierarchical feature-based matching for motion estimation. The pixel correspondences of two key frames are computed as individual pixel displacement vectors [3], which are used to generate in-between frames. Due to unpredictable and complicated changes of cartoon characters, Quan [5] classifies two types of changes for a cartoon character as global movements and local deformation. MFBA works effectively on local changes while global changes are managed by key frame relocation [5].

2.4 Feature-Based Matching Algorithm (FBA)

2.4.1 Displacement Vector (DV)

Displacement vector indicates the coordinate change of a moving point in an image as shown in Figure 2.2.

Figure 2.2: Displacement Vector Illustration

The purpose of in-betweening is to determine the displacement vectors for all the pixels on two key frames and generate in-between frames in a mathematical way, e.g. linear interpolation or B-spline interpolation. Figure 2.3 illustrates the generation of in-between frames through linear interpolation method, i.e. the in-between DVs are obtained by linearly dividing the DVs of motion pixels in the key frames.

Figure 2.3: In-Between Frame Generation

2.4.2 FBA Principle

The key objective of the hierarchical feature-based matching algorithm (FBA) [7] is to determine the displacement vector for each pixel in a pair of key frames. There are several features on the key frames that can be used for matching. The main features include edgeness, cornerness, intensity and smoothness. Edgeness is defined as the magnitude of the gradient intensity: $e = \left\| \frac{\partial i(\mathbf{u})}{\partial \mathbf{u}} \right\|$. Corners are points where the gradient direction changes rapidly, they correspond to the physical corners of objects. The cornerness of a point may be defined by the instantaneous rate of change in the gradient direction along an edge curve that passes through the point. The positiveness (black corner against white background) and negativeness (white corner against black background) of cornerness should be taken care of. Gradient based approach is used to estimate the displacement vector $\mathbf{d}$ as

$$i(\mathbf{u}) = i'(\mathbf{u} + \mathbf{d}) = i'(\mathbf{u}) + \nabla i'(\mathbf{u})\mathbf{d}^T \quad (2.1)$$

where ∇ is the usual multi-dimensional gradient operator.

The intensity disparity, r_i (also known as residual), between two key frames at point $\mathbf{u}$ could be derived from 2.1 as:

$$r_i = i(\mathbf{u}) - i'(\mathbf{u}) = \nabla i'(\mathbf{u})\mathbf{d}^T. \quad (2.2)$$

Equations could be derived similarly for the edgeness, positive cornerness, and negative cornerness features respectively as:

$$r_e = e(\mathbf{u}) - e'(\mathbf{u}) = \nabla e'(\mathbf{u})\mathbf{d}^T \quad (2.3)$$

$$r_p = p(\mathbf{u}) - p'(\mathbf{u}) = \nabla p'(\mathbf{u})\mathbf{d}^T \quad (2.4)$$

$$r_n = n(\mathbf{u}) - n'(\mathbf{u}) = \nabla n'(\mathbf{u})\mathbf{d}^T. \quad (2.5)$$

The residual error function for orientation smoothness constraint is defined as:

$$r_o = \frac{\|\bar{\mathbf{d}} \times \mathbf{d}\|}{\|\mathbf{d}\|} = \|\mathbf{d}\| \sin \theta, \quad (2.6)$$

where $\bar{\mathbf{d}}$ is the refined displacement vector from the previous iteration during hierarchical processing. The residual error for displacement smoothness is given as:

$$r_d = \|\mathbf{d} - \bar{\mathbf{d}}\|. \quad (2.7)$$

Finally, the goal is to determine the displacement vector $\mathbf{d}$ such that the weighted sum of squares of residuals is minimized:

$$\sum_u \{ \lambda_i r_i^2(\mathbf{u}, \mathbf{d}) + \lambda_e r_e^2(\mathbf{u}, \mathbf{d}) + \lambda_p r_p^2(\mathbf{u}, \mathbf{d}) + \lambda_n r_n^2(\mathbf{u}, \mathbf{d}) + \lambda_o r_o^2(\mathbf{u}, \mathbf{d}) + \lambda_d r_d^2(\mathbf{u}, \mathbf{d}) \} \rightarrow \min$$

where λ_i , λ_e , λ_p , λ_n , λ_o and λ_d are weighting parameters that are dynamically adjusted according to different resolution levels.

If we define

$$r = [r_i, r_e, r_p, r_n, r_o, r_d]^T$$

and

$$\Lambda = \text{diag} \{ \lambda_i, \lambda_e, \lambda_p, \lambda_n, \lambda_o, \lambda_d \}$$

then we have

$$\Lambda r = \Lambda J \mathbf{d} \quad (2.8)$$

where the Jacobian matrix is

$$J = \begin{bmatrix} \frac{\partial i'}{\partial x} & \frac{\partial i'}{\partial y} \\ \frac{\partial e'}{\partial x} & \frac{\partial e'}{\partial y} \\ \frac{\partial p'}{\partial x} & \frac{\partial p'}{\partial y} \\ \frac{\partial n'}{\partial x} & \frac{\partial n'}{\partial y} \\ \frac{\partial y}{\|\mathbf{d}\|} & -\frac{\partial x}{\|\mathbf{d}\|} \\ \frac{\partial r_d}{\partial x} & \frac{\partial r_d}{\partial y} \end{bmatrix}$$

By using the hierarchical approach, setting an initial displacement vector $\mathbf{d}$ is to be zero at the highest level, and applying the pseudo-inverse (a form of minimization) together with 2.8, the updating formula for $\mathbf{d}$ is:

$$\mathbf{d} = (J^T \Lambda^2 J)^{-1} J^T \Lambda^2 r. \quad (2.9)$$

After $\mathbf{d}$ is estimated, $\mathbf{d}$ is refined to yield the displacement that is used in the next iteration, $\mathbf{d} \leftarrow \mathbf{d}^* + \mathbf{d}$ until no further reduction in error is observed. $\mathbf{d}^*$ denotes the value of the displacement vector at the beginning of the iteration. Figure 2.4 illustrates the hierarchical processing approach adopted by FBA.

Figure 2.4: MFBA Hierarchical Processing

2.4.3 Summary of FBA

The FBA algorithm uses a hierarchical approach to refine the displacement vector for each pixel in a 2D cartoon image. The steps for performing this hierarchical technique are summarized as follows:

1. Analyze and extract the attributes (intensity, edgeness, and cornerness) from a given pair of key frames at the original resolution (lowest level).
2. Recursively smooth the attributes for higher levels to generate lower resolution images.
3. Start by setting the highest level as the current level.
4. Initialize displacement vectors (DVs) to zero for all pixels.
5. Iterate for a pre-defined number of times, at all pixel locations:
 - (a) Compute the average displacement vector and hence the orientation and displacement residuals.

- (b) Compute and update the DVs.
- 6. Project the DVs downwards to the next lower level.
- 7. Set the current level to the next lower level.
- 8. Repeat step 5 until the lowest level is done.

2.4.4 Modified Feature-Based Matching (MFBA)

The modified feature-based matching algorithm enhances the original feature-based algorithm by improving the accuracy of matching and automatically assigning weightages to the various features to handle a wide range of image resolutions and contrast [2]. The attributes used for the matching process are normalized and thus are handled in a uniform way. This facilitates the determination of their relative weightages. Although MFBA enhances the original FBA algorithm, it faces the same problems as FBA regarding processing speed and in-between motion, which are to be seen in section 2.5.

2.5 Analysis of MFBA-IB

MFBA-IB uses the MFBA algorithm to generate motion vectors for each pixel in a pair of key frames. Figure 2.5 shows an example of using MFBA to generate displacement vector fields for the starting and ending frames. The field is shown superimposed on the edge image of the starting frame.

(a) and (c) are the starting and ending frames. (b) is the displacement vector field.

Figure 2.5: Displacement Vector Field

Based on the generated displacement vector field, the generated sequence of frames using linear interpolation [2] are shown in Figure 2.6, which includes the starting frame, the ending frame and the in-between frames.

(a) (f) are key frames. (b) (c) (d) (e) are in-between frames.

Figure 2.6: MFBA In-Between Results - Sea

Although MFBA-IB yields good results for certain cartoon images, e.g. the test key frames as in Figure 2.10, it is time consuming to compute displacement vectors for all the pixels in a pair of key frames, this gets worse when the image size increases. This imposes a barrier to the efficiency of animation production. Another problem observed is discouraging in-between motion for certain image segments that are of disparate motion change directions and those with complicated curves. This problem will be discussed in section 2.5.2.

2.5.1 Processing Performance

Table 2.1 shows MFBA matching performance for images of different resolutions. The matching time is the time taken to determine the DVs for all the pixels in a pair of key frames. Once DVs are determined, the generation of in-between frames is straightforward (Section 2.4.1) and requires little time. The experimental results in Table 2.1 show that MFBA processing time increases with the increase of image resolution. This is due to the numbers of DVs are different for these images of different resolutions.

Table 2.1: MFBA Processing Performance for Different Image Resolutions

Experiment	Image Size(width x height x bpp)	Matching Time(seconds)
1	320x286x32	101.0
2	420x472x32	130.0
3	700x500x32	190.0
4	720x586x32	243.0

The in-between matching time may also vary for different pairs of key frames that are of the same resolution but of different image contents as shown in Table 2.2. Figure 2.7 shows how these different images look like.

Table 2.2: MFBA Processing Performance for Different Image Contents

Experiment	Image Size(width x height x bpp)	Image Content	Matching Time(seconds)
1	720x586x32	Cloud	100.0
2	720x586x32	Fire	150.0
3	720x586x32	Wave	180.0
4	720x586x32	Gorilla	230.0

The reasons why the processing time is different (Table 2.2) are mainly due to the differences of effective cartoon character pixels and motion changes, which impose different CPU calculation requirements.

Figure 2.7: Different Image Contents with 720x586 Resolution

A simple calculation will show how this processing throughput would affect cartoon production. A desired 5-minute cartoon sequence of 24 fps contains $5 \times 60 \times 24 = 7200$ frames. If only 5%, or 360 of these frames are key frames to be matched, in the worst case as in the above experimental results, it needs $180 \text{ pairs} * 243 \text{ seconds} / 3600 = 12.15 \text{ hours}$ to complete. If 10% of these frames need to be matched, the time needed will be 24 hours. Generally one time rendering is not enough as an animation artist may adjust parameters and re-render the same sequence for a few times. Hence it will take a few days or even more time to complete a 5-minute sequence, out of which 10% are key frames. Therefore there is a need to increase the batch processing throughput for MFBA-IB in order to apply it in real cartoon animation production.

2.5.2 In-Between Motion Quality

Despite the fact that MFBA-IB yields good results for some 2D cartoon frames, e.g. Figure 2.10, it generates discouraging in-between motion for certain frames. As Figure 2.8 shows, the in-betweened eyes of the girl are not satisfactory because there are hardly-observable intermediate motion changes between the opened eyes and the closed eyes.

This problem is inherent with MFBA since it is using a general hierarchical approach to determine displacement vectors for every pixel in a 2D image. The DVs of lower-level (up-sampled) pixels have to base on other higher-level (down-sampled) pixels, this ignores the specialty of lower-level pixels. That is, certain local information related to parts of a character in the image is not taken into consideration. Also, the smoothness constraint forces neighbouring vectors to align with each other as much as possible. Therefore in certain situations when pixel motion changes are not in the same direction, it generates frames that have discouraging in-between motions. Figure 2.9 is another example of such case. The deformation of cloud contains disparate pixel motion directions and the generated in-between frames lack of intermediate motion changes.

(a) and (f) are key frames.

(b), (c), (d) and (e) are in-between frames.

Figure 2.8: MFBA In-Between Sequence - Girl

(a) and (f) are key frames.

(b), (c), (d) and (e) are in-between frames.

Figure 2.9: MFBA In-Between Sequence - Cloud

(a) and (f) are key frames.

(b), (c), (d), (e) are in-between frames.

Figure 2.10: MFBA In-Between Sequence - Fire

2.6 Summary

This chapter presents a general research survey on cartoon animation systems and a variety of algorithms on computer-assisted in-betweening as well. The advantages and disadvantages of *the ink and paint* systems and *automated in-betweening* systems show the importance and significance of automating the in-betweening process for animation artists. That is, an effective and efficient computer-assisted in-betweening system could greatly reduce artists' work load in cartoon animation production. For such an in-betweening system to be built, a good in-betweening algorithm is the key, therefore some of the existing in-betweening algorithms have been discussed in this chapter.

Among these algorithms, FBA and its modified version, MFBA, yield encouraging results on automatic in-betweening. And the detailed presentation on the principle of FBA and MFBA facilitates the in-depth understanding of FBA and MFBA, which is the primary foundation of MFBA-IB, the in-betweening process via MFBA. However, the analysis on the experimental examples of MFBA-IB reveals certain problems for applying MFBA on real cartoon in-between production in two perspectives: low processing throughput and discouraging in-between motion quality. By focusing our research effort on these two perspectives, the solutions to improve MFBA-IB are proposed in Chapter 3.

Chapter 3

Solution Analysis

In this chapter, unless otherwise mentioned, MFBA-IB would be taken to include its MFBA image matching engine and the subsequent IB, in-between frame generation process. The feasibility and effectiveness of proposed solutions for improving MFBA-IB are explored in two areas: distributed image processing and computer animation with NURBS. Based on the analysis and related work in these two areas, the solutions for improving MFBA-IB are proposed.

[Keywords: **SMP** - *S*ymmetric *M*ulti**P**rocessor, **NURBS** - *N*on-*U*niform *R*ational *B*-*S*pline]

3.1 Analysis of Distributed Image Processing

In this section, the concepts and models of distributed image processing will be analyzed in comparison to parallel image processing. Both of these two techniques are helpful in solving computationally intensive problems but some effort has to be made in order to apply them in a specific field, for us, the in-between frame generation. For the sake of understanding, some of the related research work on these topics will be introduced and illustrated. This facilitates the design of distributed communication model for MFBA-IB.

3.1.1 Resource Management

Distributed and parallel architectures can utilize a large number of commodity components working in parallel, i.e. parallel processing on several distributed PC's and parallel accessing to many resources, e.g. to disks, the potential of scalable processing power and scalable resource access [21]. The in-between image processing is computationally intensive and has frequent disk access, e.g. to reduce CPU memory usage, it generates a lot of intermediate files for its own use. When the in-between image processing capability is fully developed, lots of in-between frames can also be generated. In such a situation, there is a need to manage the distributed resources, e.g. PCs and disks, for MFBA-IB.

3.1.2 Parallel Image Processing on SMP

Parallel image processing could achieve a good speed-up if an image processing algorithm could parallelize its computationally intensive parts. The parallelism on SMP does not involve networking overhead since the multiple processors are sharing the same local onboard memory, the task scheduling between the multiple processors is done by the operating system running on top of these processors. Figure 3.1 shows such a working scenario. The operating system provides the service for a user application to utilize the multiple CPUs in the SMP configuration.

Figure 3.1: Symmetric Multiprocessor System

The multiple CPUs have direct communication with the operating system. This is

the advantage of SMP over a computer cluster because it eliminates the communication overhead imposed by the network. This is inherent in clustering computers connected via a network. Of course, the disadvantage is that it is not possible to add on new processors in SMP while it is easy to add on new computers in a cluster to enhance computation power. If the speed-up of a single pair of key frames is concerned, SMP will definitely be effective in reducing the amount of time needed for in-betweening. Also, currently a language called *cilk* for multi-threaded parallel programming based on ANSI C can be used to implement the parallelism on SMP efficiently. The information about *cilk* can be found in <http://supertech.lcs.mit.edu/cilk/>. In comparison to distributed image processing based on computer clusters, it is more difficult for parallel image processing based on SMP to achieve the required speed-up (with many pairs of key frames available) for MFBA-IB as illustrated in section 3.1.3.

3.1.3 Distributed Image Processing

Distributed image processing is implemented with a computer cluster, which contains a number of individual computer nodes. In such an environment, network communication overhead naturally exists. For some applications, the network overhead is not a concern, but it is a critical issue for implementing parallelism in time-critical algorithms like our in-between algorithms.

A computer cluster is more applicable in the situation where there is a large amount of work needed to be done and it can be separated into a number of equal amount jobs, which are distributed and processed on different computers. Using MFBA-IB for in-between frame generation, the computationally intensive part is the matrix calculation of equation 2.9. If the matrix calculation could be parallelized and distributed, the network communication overhead has to be considered. The following example shows the difficulty to achieve the speed-up for parallel matrix calculation via a computer cluster.

T = the matrix calculation on one single CPU computer.

$\frac{T}{n}$ = the ideal final matrix calculation time via distributing parallel parts among n single CPU computers.

t = the network communication overhead. $\frac{T}{n} + t$ = the network overhead incurred $\times$

The final speed-up via distribution would be $\frac{T}{\frac{T}{n} + t}$, or $\frac{n}{1 + \frac{nt}{T}}$.

The problem here is only when $t \rightarrow 0$, a speed-up of n times can be achieved. If t is very large, which results in $T < \frac{T}{n} + t$, slow-down could be caused instead of speed-up. The possible situations are summarized as:

1. speed-up of $\frac{T}{\frac{T}{n} + t}$ times, when $t < (1 - \frac{1}{n}) * T$;
2. no speed-up, when $t = (1 - \frac{1}{n}) * T$;
3. slow-down, when $t > (1 - \frac{1}{n}) * T$.

For example, if a matrix calculation takes 1000 clock cycles in one single CPU computer and four computers are available. After distribution, a 4-fold speed-up is expected, as each computer takes 250 cycles to finish. Assume the network overhead is t . Then the final speed-up would be $\frac{1000}{250+t}$. If t is 0, a 4-fold speed-up is achieved. If t is 250, a speed-up of 2 times is achieved. If $t = 750$, no speed-up is available. When $t > 750$, slow-down occurs.

From the example we could see that the speed-up depends on t and T , which could result in either a speed-up or slow-down. In parallelizing the MFBA-IB in-between algorithm over a distributed computer cluster, it is possible that a slow-down could result when the network communication overhead is high enough to make $t > (1 - \frac{1}{n}) * T$ be true. On the other hand, speed-up can be achieved when SMP machines are used as the inter-processor communication overhead is probably $t < (1 - \frac{1}{n}) * T$ within the same operating system.

The analysis above shows the difficulty to parallelize the MFBA-IB algorithm over a computer cluster. However, inter-pair processing can achieve a very good speed-up via distributed processing over multiple computers. The reason is that there is no information exchange while the pairs are processed. All the pairs are processed separately and their individual processing time is more or less equal depending on the complexity of images. A distributed framework is thus built up to realize the inter-pair speed-up for MFBA-IB as shown in Chapter 4.

In summary, the reasons why distributed processing technique is chosen are as follows:

1. key frame pairs can be processed separately without information exchange among them.
2. the network overhead concern and the difficulty to modify the codes to parallelize MFBA-IB over a computer cluster;
3. the system is more scalable as computer nodes can be added easier to the distributed framework compared to the fact that it is more difficult to increase the number of processors in a SMP system.

3.1.4 Related Work

In this section, some related work in distributed and parallel image processing are reviewed [21, 22, 23, 25]. Image oriented access and processing operations are often both compute and I/O intensive. Therefore, cluster computing is applied in many areas to realize distributed processing of image algorithms or applications. For example, Jeffrey [23] proposes a parallel skeleton for edge detection based on a cluster. The implementation of the skeleton is a client/server architecture, using UNIX-based C network programming mechanisms such as Sockets. The paper [25] presents the development of a parallel image matching system which uses a divide-and-conquer method to implement the proposed hierarchical matching scheme on a networked workstation cluster.

A computer aided parallelization tool (CAP), proposed by Benoit and Roger [21], is to let parallel application programmers have complete control on how their application is parallelized, and at the same time freeing them from the burden of explicitly managing a large number of threads and associated synchronization and communication primitives. The configuration of the system is shown in Figure 3.2.

Figure 3.2 maps the concept of CAP onto computing units. Each parallel CAP construct consists of a split function splitting an input request into sub-requests sent in a pipelined parallel manner to the operations of the available threads and a merging function to collect the results [21].

Figure 3.2: Mapping of the Parallel Construct
onto a Simple Multi PC Configuration

The *Parallel Image Processing Toolkit* (PIPT) library, designed and implemented by Jeffrey, Andrew and Brian [22], is a scalable, extensible, and portable collection of image processing routines. PIPT uses a message passing model based on the Message Passing Interface (MPI) standard and supports the parallel execution on heterogeneous workstation clusters. The PIPT uses a manager/worker scheme in which a manager program reads an image file from disk, partitions it into equally sized slices, and sends the pieces to worker programs running on machines in the cluster.

3.1.5 Discussion

The work done by other researchers focused more on a general approach to achieve image processing speed-up and performance improvement. In our case of in-betweening, a more specific design approach is required to utilize workstation clusters as our main purpose is to speed up the MFBA in-between algorithm which is presented with many key frame pairs. Our design and implementation are the modification and specialization of ideas selectively adopted from many other researchers, e.g. our scheduler/agent scheme (Section 4.1.1), compared to manager/worker scheme [22], is a specialized distributed architecture for improving MFBA-IB.

3.2 Computer Animation and NURBS

With increasing power of computers, computer animation is a hot topic [4]. It has evolved into a standard technique in computer graphics [44]. Most computer animation hitherto uses polylines and polygonal faceted representation of objects [37, 41] to ease the interpolation of points between positions in cells. NURBS curves are a powerful geometric tool and are gaining increased use and recognition in many computer aided designs (CAD) and graphics systems. Mathematically speaking, NURBS curves and surfaces are parametric functions that can approximate most types of curves or surfaces. In this section, computer animation and the way to represent the curves of a cartoon image and the tools to realize the representation in a user interactive manner will be introduced and discussed and this will facilitate and improve our design of interactive NURBS in-betweening for MFBA-IB.

3.2.1 Animation System and Methods

Many researchers have implemented a number of animation systems, which include the in-betweening process. In many such systems, the following methods are adopted for animation [37]:

- all objects are modeled as a set of points and surfaces;
- through time all points are interpolated via some methods;
- consequently the movie representation is a list of curves. When they are all evaluated at a particular time, they represent the scene at the time.

These methods use curves to model objects and transform them as and when required in making a movie which could be a cartoon movie. Figure 3.3 shows the system presented in [37].

Figure 3.3: NURBS Animation System

As NURBS can represent a wide range of surfaces, the system can present to users with the familiar objects, e.g. bézier curves, polygonals and triangle strips, while internally these objects are converted into NURBS surfaces for the purpose of in-betweening. In terms of 2D computer animation, the system could be re-designed to handle 2D curves, e.g. circles, ellipses and polylines. This complements the pixel-based MFBA-IB algorithm by handling those portions of the image that are difficult to tackle in a computer graphics technique - NURBS.

3.2.2 Curve Representation

The two most common methods to represent a curve or a surface are the implicit and parametric methods [14]. The implicit method describes a relationship between the axis variables. For example, the function $f(x, y) = x^2 + y^2 - 1 = 0$ represents a circle of radius 1. The parametric method could define the axis variables in a function of an independent parameter. For example, a curve could be defined with the independent variable u as:

$$C(u) = (x(u), y(u)) \quad a \leq u \leq b.$$

To represent the first quadrant of a circle in a parametric form, one can write:

$$C(u) = (\cos(u), \sin(u)) \quad 0 \leq u \leq \frac{\pi}{2}.$$

From these two forms, we see that there is no unique way of representing a curve in the parametric method.

NURBS is the generalization of B-splines, which are based on rational Bézier curves. The rational Bézier curve is a generalization of Bézier curves. A Bézier curve of degree n is defined as:

$$C(u) = \sum_{i=0}^n B_{i,n}(u) P_i \quad 0 \leq u \leq 1 \quad (3.1)$$

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i}. \quad (3.2)$$

Figure 3.4 shows a Bézier curve of degree 3:

Figure 3.4: A Bézier Curve of Degree 3.

However, Bézier curves cannot represent conic curves. A *rational function* can represent a conic curve such as a circle. It is defined as the ratio of two polynomials such as:

$$x(u) = \frac{X(u)}{W(u)} \quad y(u) = \frac{Y(u)}{W(u)} \quad z(u) = \frac{Z(u)}{W(u)}. \quad (3.3)$$

From Equation 3.3, the rational Bézier curve is defined as:

$$C(u) = \frac{\sum_{i=0}^n B_{i,n}(u) w_i P_i}{\sum_{i=0}^n B_{i,n}(u) w_i} \quad 0 \leq u \leq 1$$

where P_i and $B_{i,n}$ are defined as before, w_i are scalars called the *weights*.

The geometric coefficients P_i are called *control points*. The basis function $B_{i,n}$ are the classical n th-degree Bernstein polynomials given by 3.2.

A B-spline curve is defined as:

$$C(u) = \sum_{i=0}^n N_{i,p} P_i \quad a \leq u \leq b, \quad (3.4)$$

where P_i are the control points and $N_{i,p}$ are the p th degree B-spline basis functions. The B-spline basis function of p th degree is defined using the recurrence formula as:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (3.5)$$

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}. \quad (3.6)$$

A p th degree Non-Uniform Rational B-Spline (NURBS) defined by control points P_i , the knot vector $U = u_0, u_1, \dots, u_m$ and weights w_i is written as:

$$C(u) = \frac{\sum_{i=0}^n N_{i,p}(u) w_i P_i}{\sum_{i=0}^n N_{i,p}(u) w_i} \quad a \leq u \leq b. \quad (3.7)$$

With the homogeneous control points P_i written as $P_i^w = w_i x_i, w_i y_i, w_i z_i, w_i$ in a four dimensional space where $w \neq 0$ and dividing all the coordinates P_i by the fourth coordinate w_i , the NURBS curve can be redefined as:

$$C^w(u) = \sum_{i=0}^n N_{i,p}(u) P_i^w, \quad (3.8)$$

where $C^w(u)$ are the resulted NURBS points in the homogeneous coordinates. NURBS can also be used to represent surfaces. It is not considered here since only 2D line-drawing cartoon images are explored in this research work.

3.2.3 OpenGL and NURBS

OpenGL is a good tool to visualize NURBS curves and surfaces. OpenGL provides an easier interface to draw NURBS curves as well as Bézier curves and B-Splines from programmers' point of view [16]. Before drawing anything, a NURBS context object should be created because it will be used in all subsequent GL calls. The call is:

$$nurbs = gluNewNurbsRenderer().$$

The function to evaluates NURBS is *gluNurbsCurve*, which is defined as:

```
void gluNurbsCurve( GLUnurbs* nurb,
                   GLint knotCount,
                   GLfloat *knots,
                   GLint stride,
                   GLfloat *control,
                   GLint order,
                   GLenum type ).
```

The first argument of the call is the NURBS context object. The next two are the number of knots and a pointer to the knot vector. Then comes the width of each control point specifying 3D [26] or 4D and a pointer to the control points. The next argument is equivalent to the number of knots minus the number of control points. The last argument specifies the kind of a representation, whether it is going to define a curve drawn in 3D space or it is going to represent color values.

Simply manipulating OpenGL function calls is difficult and error prone. Fortunately, there is a library called NURBS++ [15] that provides a simpler interface to OpenGL. This allows users to interact with the drawn curves easier. In the next few chapters, there will be a more detailed explanation about the design and implementation of the NURBS matching algorithm.

3.2.4 Interactive Image Segmentation

Image segmentation is explored to solve the pixel-based MFBA-IB's discouraging in-between motion in image portions with disparate pixel motion directions (Section 2.5). There is no general rule for accurate image segmentation because image segments are subjective. One way to segment an image is based on the different colors in different regions. Automatic segmentation of images is not an easy topic because it is difficult for computer to understand or perceive, in the same way as human beings, how or what a segment should be. The existing segmentation algorithms, for example [20], could help to certain extent, but not completely. In order to determine the result of matching two localized segments of a pair of key frames, interactive segmentation is used in our research. Automatic segmentation is taken to be future work for in-betweening.

3.2.5 Related Work

Spline curves have been used in various research areas related to computer graphics, image processing and computer animation [27, 34, 40]. They are used to solve problems related to animation. Ravi [34] proposed a method for fast interpolation between animation key frames. The method allows automatic computer-generated improvements of the motion. Bert and Michael [40] proposed a linear control structure for piecewise rational motion suitable for geometric processing. NURBS is the generalization of spline curves and they have been applied to tackle problems that could hardly be solved by spline curves.

Linear interpolation has a few drawbacks [27] such as:

- the lack of smoothness in the motion;
- discontinuity in the speed motion;
- distortion in the rotation.

Figure 3.5 shows an example of distortion in length when the rotation is interpolated linearly. Due to the drawbacks of linear interpolation, Doris and Richard [27] proposed a new method for using cubic interpolating splines in a key frame animation system.

Figure 3.5: Distortion in Length when Rotation is Simulated Linearly

By introducing tension, continuity and bias control, a very general class of interpolating cubic splines is produced [27]. The flexibility provided by these parameters is especially useful in key frame animation, because it allows the animator to adjust the movement of objects without having to adjust or redraw key frames.

NURBS provides a convenient way to describe surfaces of almost any shape [43]. Trimming NURBS [41] makes it possible for developers to cut away the unneeded areas of a particular surface. Combining thousands or even ten thousands of trimmed NURBS surfaces makes it possible to describe very sophisticated objects like cars, airplanes or submarines. Therefore, Kahlesz [43] proposed a system that is capable of tessellating trimmed NURBS surfaces with a given error tolerance in Euclidean space. This approach makes it possible to sew the surfaces together into one mesh, making it suitable for simplification and manipulation.

The in-betweening animation process is concerned with the changing behavior of synthetic cartoon characters through time. Preston and Hewitt [37] proposed a technique for creating higher dimensional NURBS to model object deformation.

3.2.6 Discussion

From the past research work, it can be seen that NURBS has great potential in the computer animation area. It is known that NURBS is an industrial standard tool for the representation and design of geometry [17]. A few reasons for the use of NURBS are [17, 18]:

- one common mathematical form for both standard analytical shapes (e.g. conics) and free form shapes;
- the flexibility to design a large variety of shapes;
- reasonably fast evaluation by numerically stable and accurate algorithms;
- invariant under affine as well as perspective transformations;
- generalization of non-rational B-splines, non-rational and rational Bézier curves, and surfaces.

There are infinite kinds of curves in a hand-drawing cartoon image, which require sophisticated curve modeling. NURBS provides such flexibility and modeling ability. This facilitates the design of a novel in-between matching algorithm to address the concern of the discouraging in-between problem observed in existing in-betweening algorithms, e.g. MFBA-IB (Section 2.5).

The use of NURBS combines image processing and computer graphics together. This has the potential to improve the in-between quality of MFBA-IB which relies on low level image processing techniques. The result of such combination is presented in Chapter 5.

3.3 Suggested Improvements

According to our analysis of distributed image processing technique and computer animation technique with NURBS, two improvements for MFBA-IB are suggested as follows:

- a distributed in-betweening system for improving the computation throughput performance of MFBA-IB;
- an interactive NURBS-improved system for better MFBA-IB quality.

The design and implementation issues of these two improvements are presented in the next two chapters.

Chapter 4

Distributed In-Betweening

Although parallel and distributed image processing techniques have been applied in many areas, they lack of a suitable tool for software development [30]. As network technologies continue to mature, workstation clusters are becoming a viable and economical parallel computing resource [24], therefore it is important to utilize these environments for parallel image processing. Computer-assisted in-betweening production is a process that handles a large number of key frames. In order to achieve an overall processing performance for parallelizing the in-betweening process, there is a need to develop an appropriate application programming interface (API). The original FBA and MFBA algorithms have made it a reality to generate in-between frames automatically from two key frames. However, there are still certain problems related to processing speed and in-between motion (Section 2.5). In this chapter, a scheduling communication model is proposed to distribute and parallelize the MFBA in-betweening process. Such a model allows researchers to develop distributed and parallel in-betweening tasks transparently [31]. The communication model is designed in a flexible manner such that *Inter-Frame* and *Intra-Frame* modes can be handled at ease. The model should also be scalable [32] such that with more computers available, more computation power can be coordinated into the in-betweening process. Such scalability is important in subsequent work as it provides a framework foundation to improve throughput performance of MFBA-IB, NURBS-improved matching and in-between generation.

4.1 Distributed and Parallel Communication Model

Many image processing tasks exhibit a high degree of data locality and parallelism [24] and they can be mapped quite readily to specialized massively computing hardware. However, special-purpose machines (real parallel machines) have not been cost-effective enough (in terms of the necessary hardware and software investment) to gain wide-spread use. As network technologies continue to mature, clusters of workstations are increasingly viewed as a parallel computing resource [24]. The advantages of cluster-based parallel computing are low cost and high utility. The disadvantages are considerable communication latency and irregular load patterns on the distributed computer nodes.

The communication model to be designed and implemented is going to be Linux-based (cost-effective in terms of hardware and software APIs) and specialized for in-betweening. The model is implemented via TCP/IP-based *socket* in Linux, which is reliable and supports parallel execution on heterogeneous workstation clusters. Generic distributed computing APIs, e.g. *Common Object Request Broker Architecture* (CORBA) and *Distributed Component Object Model* (DCOM), are not used because they are more suitable for building large scale information communication models. Our model is designed specifically to provide dynamic scheduling [29] and monitoring ability for the in-between image processing.

4.1.1 Design Overview

The goals of the design are simple, to be able to schedule and monitor the distributed in-between image processing, to achieve speed-up without modifying image algorithms, to make it easier to integrate with user's image processing applications. The key design concern is to properly schedule distributed tasks and optimize the task allocations [29]. MFBA is used here as a practical example for the design and implementation.

A flexible and adaptive distributed communication API is helpful for developers to integrate the in-between components, e.g. a graphical user interface and backend in-between matching processing. It hides the underlying network communication issues from

image algorithm developers. The overall design paradigm is shown in Figure 4.1.

Figure 4.1: Distributed Communication Model

The design caters for modularity of each component with object-oriented techniques. The *scheduler* schedules and monitors the distributed jobs for the in-betweening process on each cluster machine. Multi-threading technique is applied for the concurrent cooperation of different components. The more detailed design is shown in Figure 4.2.

Figure 4.2: Distributed Communication Components

As Figure 4.2 shows, the distributed system consists of a scheduler and a number of distributed computer nodes. TCP/IP is adopted to be a reliable communication protocol for the scheduler and the cluster nodes.

The *scheduler* consists of a graphical user interface (GUI), a synthesis process and a monitor process. Users could schedule the work that is going to be processed by each cluster node and monitor the status of all the jobs as well. The monitor process is acting as the proxy for all the scheduled jobs. It distributes jobs to the available cluster nodes and collects the running status from all the cluster nodes. The status information collected will be shown to the users through the GUI.

Each of the cluster nodes consists of a *daemon* process and an *agent* process. If a cluster node has two CPUs, two in-betweening processes would be started at the same time by the *daemon* process. The *agent* process is integrated with the in-betweening process via multi-threading. The *agent* process acts as a proxy for the in-betweening process to communicate with the *scheduler* process.

Whenever users want to run a pair of key frames, the *scheduler* will connect to the port that is listened to by the *daemon* process running on a cluster node. The *daemon* process then launches the in-betweening process, which creates the *agent* process as its communication proxy with the *scheduler* process. The *agent* process has embedded procedures to communicate with the *scheduler* for control and status information. The in-betweening process talks with the *agent* process via inter-process communication (IPC) techniques. In order to ensure reliable availability to the *scheduler*, the *daemon* process remains on a cluster node for all time.

4.1.2 Communication Model Implementation

The implementation of the communication model is straightforward once the design is completed. In order to minimize the communication latency, gigabit port or 1000 Mbps port, a common built-in component in most of workstation motherboards nowadays, is one of the hardware requirements for running our APIs. And TCP/IP socket is used as the basic communication handler in the model because it guarantees the reliability of network communication. Figure 4.3 shows the interactive diagram for the implementation of our communication model. The diagram reveals our interactive message passing protocol, which defines the hand-shake between the distributed communication parties in the model.

Figure 4.3: Interactive Communication Diagram

There are five major components: *scheduler*, *socket*, *imagecom* (image communication interface), *agent* and *daemon*. The *scheduler* and the *agent* provide the necessary API interfaces to the GUI and the backend in-betweening process as illustrated in Figure 4.4. The API is implemented via IPC. Therefore the network communication is hidden from the GUI and the distributed in-betweening processes.

Figure 4.4: Distributed Communication Structure

The interactive communication diagram is shown in Figure 4.3. The *scheduler* creates a *socket* to inform the *daemon* process to start an in-betweening process. The in-between process has embedded procedures to create an *agent* process for itself. The *agent* process

connects to the *scheduler* process and requests image frames on behalf of the in-betweening process. Once the in-betweening process gets enough frames, it starts processing while the *scheduler* process and the *agent* process are in a two-way control mode to handle any asynchronous interrupt events, e.g. backend error or user commands to the in-betweening process. After the in-betweening process is completed, the resulting frames are sent back to the *scheduler* process. In the case when a pair of key frames is split into corresponding parts for distributed processing, the *scheduler* process synthesizes the different in-between parts into complete in-between frames.

4.1.3 Discussion

The *socket* and the *imagecom* components are separated to be two different classes as the *imagecom* is a more specialized communication interface while the *socket* is a basic interface. For the ease of future development, it is better to keep the *socket* as an independent component because there could be other specialized communication needs instead of merely transferring images as does the *imagecom*. If necessary, the *socket* could be passed into any specialized communication interface, therefore the flexibility of the design is maintained.

4.2 Speed-up for MFBA-IB

Via our distributed communication model, experiments were done to investigate the speed-up for MFBA-IB in partitioning a pair of key frames into four pairs using distributed processing versus a pair of whole key frames using local processing. *Intra-Frame* mode is named for the partition of a pair of key frames while *Inter-Frame* mode is named for the distributed MFBA-IB processing of key frame pairs without partition. These two modes are experimented with our distributed communication model and the results will be analyzed in the following sections.

The system configuration for the subsequent experiments was a cluster with each node having an Intel 1.6 GHz CPU and 512M RAM.

4.2.1 Intra-Frame Mode

Intra-Frame mode means partitioning key frames and processing the partitioned pairs in a distributed environment. An experiment was conducted to explore MFBA performance characteristics with the use of the distributed communication model as has been designed and implemented. The experiment partitioned a pair of key frames into four pairs, each of which contained an overlapped portion of the original pair as shown in Figure 4.5.

Figure 4.5: Central Partition of a Cartoon Character

There are three possible performance curves regarding the MFBA processing time

versus image size as shown in Figure 4.6.

Figure 4.6: Possible Performance Curves for MFBA

Case A shows the MFBA processing time increases with increasing partitioned image sizes. Case C is the reverse of Case A. Case B shows the linear relationship between the partitioned image size and the MFBA processing time. MFBA performance characteristics is to be determined by this experiment.

Table 4.1 shows the experimental results of distributed processing versus local processing for a pair of 472x420 line-drawing images. Processing time is computed for the partitioned image size against the original image size, the number of partitioned parts and the overlapped pixels. For example, an image of size 472x420 has four partitioned images of size 236x210 with no overlapped pixel. Therefore, an image of size 472x420 will have four partitioned images of size 246x220 with 20 overlapped pixels. The other partitioned image sizes are similarly computed.

Table 4.1: Distributed MFBA Processing versus Local MFBA Processing

Experiments	ImageSize	Overlapped	Partitioned Image Size	Distributed(s)	Local(s)	Speed-up
1	472x420	20	246x220=54120	42.0	130.0	3.09
2	472x420	40	256x230=58880	42.2	130.0	3.08
3	472x420	60	256x230=58880	44.2	130.0	2.94
4	472x420	100	246x220=54120	78.4	130.0	1.66

Table 4.1 shows, with the same pair of key frames, that as the overlapped pixel number increases, the distributed processing time increases and the speed-up drops. Plotting the column *Partitioned Image Size* in Figure 4.1 on the same coordinates as in Figure 4.6

shows that Case A is the MFBA performance curve. That means the increasing ratio of MFBA processing time is more than the increasing ratio of image size, hence the overall speed-up reduces with the increase of the overlapped pixel number.

On the other hand, the final synthesized in-between frames, generated by MFBA-IB algorithm, would do better with the increase of overlapped pixel number because the generated in-between parts can be synthesized better after cutting off the common image area during the synthesis process.

This approach could lead to certain problems as shown in Figure 4.7. The in-between frames are different from those generated by processing the pair of key frames without partition. It can be seen that a few final in-between frames are not synthesized properly in some parts of the cartoon character.

Figure 4.7: Distorted Distributed In-Between Frames

A particular reason is that when the cartoon character in the key frames is not in the central portion of the original pictures, they are partitioned as shown in Figure 4.8. The result of such partition is analyzed in detail in the following section 4.2.2.

Figure 4.8: Non-Central Partition of A Cartoon Character

4.2.2 Result Analysis

Ideally, the final synthesized results should be the same for both partitioned and non-partitioned processings. Unfortunately, as shown in Figure 4.7, the results are not what we expected. The main reason for this is that the cartoon character is not partitioned equally as it is not in the central area of the original key frame. MFBA uses a hierarchical matching approach (Section 2.4.2). Therefore, the difference between the partitioned and non-partitioned processings is the down-sized image information from which the initial

displacement vectors are determined. Figure 4.8 gives an example that after partitioning the pair of key frames and down-sampling them, the initial displacement vectors are determined from these parts separately since we distribute each pair of sub-images for MFBA-IB local processing. While without partitioning, the initial displacement vectors are determined from the down-sized key frames alone. For example, if a pair of key frames are partitioned into four pairs, there will be four sets of initial displacement vectors after down-sampling. There is no forced dependency between these parts, hence they each up-sample individually without exchanging information among them. The other reason about the unexpected results could be after down-sampling, some part of the key frames contains no information about the cartoon character at all, therefore the initial displacement vectors for the pixels in that part are not correct.

However, it is found that if a single cartoon character is segmented out as a sub-image and the partition is applied to this sub-image, the result is much better. Figure 4.5 shows that the sub-images are partitioned into four parts and most of the area in each part contains a large portion of the cartoon character. In this way, each down-sampled key frame still contains a significant amount of the character's information and hence have enough region of support. The initial displacement vectors obtained are more reasonable than those in Figure 4.8, where some parts of the partitioned images have very small region of support.

The quality of distributed in-betweening result with such adjustment does improve. As shown in Figure 4.9, the result is similar as MFBA-IB local processing but with a speed-up of almost four folds.

The limitation of this adjustment is that the movement of the character of object should not be too much otherwise each pair of the separated portions may not have enough region of support to match. In order to overcome this limitation, further analysis is required on finding out the relationship between the character's motion and the number of overlapped pixels suitable for automatic partition. Once this relationship is established, an adaptive algorithm can be developed to better quantify the required number of overlapped pixels and hence to assist the automatic partition, however this need future research effort.

Therefore this *Intra-Frame Mode* section only considers the effect of overlapped pixels on improving MFBA in-between processing quality and speed. The corresponding experimental analysis and results are given in this section, which can in turn help to evaluate such an adaptive algorithm in future. After partitioning by considering overlapped pixels, each pair of separated portions can be seen as a pair of key frames and therefore they can be processed in *Inter-Frame mode* (parallelized processing) as will be discussed in section 4.2.3.

Figure 4.9: The Distributed MFBA-IB Results of Central Segmentation in *Intra-Frame* Mode

Figure 4.10: The Distributed MFBA-IB Results of Independent Characters in *Intra-Frame* Mode

Figure 4.11: The Composition of Independent Characters in Key Frames for Distributed Processing

The conclusion is that in order to use the partition approach of key frames together with MFBA-IB, a cartoon character has to be segmented out as a sub-image before the partition is applied and this kind of partition is called the *central-partition* (Figure 4.5). The segmented area should fit in a cartoon character with motion in both the starting and ending key frames. The speed-up of MFBA-IB could be achieved in this way with better in-between quality, compared to the *non-central partition*.

If there are multiple independent characters in a pair of key frames as shown in Figure 4.10 (a) and (f), these characters can be interactively segmented and distributed for MFBA-IB processing. After MFBA-IB processing, the in-between frames of these independent characters can be composite into final in-between frames as shown in Figure 4.11. This reduces the overall processing time for this kind of key frames. Figure 4.10 shows the result of the distributed processing for independent characters in a pair of key frames.

The other problem with MFBA-IB is the discouraging in-between motion observed in certain portions of key frames (Section 2.5). A localized or segmented matching algorithm is proposed to solve this problem in Chapter 5.

4.2.3 Inter-Frame Mode

Inter-Frame mode speed-up for MFBA-IB can be achieved via our communication model when there are a number of key frames waiting to be processed. If the number of key frame pairs is equal to the number of computer nodes in a cluster, the overall MFBA-IB processing time is roughly equal to the amount of time needed to process one pair of key frames. In such situation, taking image complexity into account, the total amount of time for processing all the key frame pairs will be equal to the largest amount of time needed by any pair among them. Computer resources will not be held up by the slowest pair because the scheduler of the communication model takes care of the allocation and deallocation of computer nodes. The load balancing problem is also handled properly by the scheduler.

(a) and (f) are the first pair of key frames

(b), (c), (d) and (e) are the in-between frames

(a') and (f') are the second pair of key frames

(b'), (c'), (d') and (e') are the in-between frames

Figure 4.12: The Distributed MFBA-IB Results in *Inter-Frame* Mode

Figure 4.12 shows the distributed MFBA-IB results of two pairs of key frames in *Inter-Frame* mode. The timing statistics for these two pairs of key frames is: about eight minutes for local MFBA-IB in sequential order; about four minutes for distributed MFBA-IB in parallel order. This experimental result verifies that our distributed communication model does improve the overall processing time, which is about half of the original local MFBA-IB. However, the problem that the in-between frames lack of intermediate motion for certain image portions still remains the same as local MFBA-IB (Figure 2.8), i.e. the eyes of the in-between character.

4.3 Summary

With the distributed MFBA-IB system implemented via our communication model API, the original local MFBA in-betweening speed-up is achieved for a single pair of key frames or a large number of key frames, which have been discussed as *Intra-Frame* mode processing and *Inter-Frame* mode processing. The results of these two types of processing show encouraging speed-up improvements for MFBA-IB and prove our distributed API to be a success. However, there are limitations for our methods. For *Intra-Frame* mode, two key frames for matching should not have too much motion changes, otherwise after distributed processing of partitioned portions, the partitioned in-between frames cannot be synthesized properly. For *Inter-Frame* mode, the overall throughput performance of MFBA-IB is improved, but the motion quality of the final in-between frames is bounded to MFBA, i.e. in-between frames do not have satisfactory intermediate motion for certain cases of cartoon key frames. We are going to address this problem in Chapter 5.

From the experimental results and discussions of *Intra-Frame* mode and *Inter-Frame* mode for improving MFBA-IB, we can see that our distributed API is flexible and adaptive to different situations. Therefore our distributed API can also be applied and integrated with other in-between algorithms to make use of cluster computers to improve throughput performance for animation production.

Chapter 5

NURBS-Improved Matching

Non-Uniform Rational B-Spline (NURBS) is used in our research to model a pair of segmented drawings extracted from a pair of key frames. With enough control points, NURBS could approximate most types of shapes[14, 18]. One of MFBA-IB limitations is that it does not give satisfactory results when certain portions of key frames have displacement vectors of disparate motion change directions (Section 2.5). As shown in Figure 2.8, which is generated by MFBA-IB, by observation, it can be seen that the girl's eyes on the key frames are not in-betweened properly. Our objective is to use NURBS to model particular portions of key frames that would have problem with MFBA-IB and generate frames with better in-between motion. In this chapter, we propose a NURBS-improved matching algorithm, which is partitioned into a few steps as follows:

- line thinning;
- segmentation;
- line fitting via NURBS modeling;
- NURBS in-betweening;
- generation of in-between curves;
- post-processing of the in-between curves, e.g. line thickening, and
- synthesis of the in-between segments.

5.1 Line Thinning

For reasons that will be explained later, key frames have to be thinned before the subsequent steps can be applied on them. As one of the thinning steps, gray scale pixel values should be binarized in such a way that any pixel value equal or above a certain gray level (threshold) is set to the maximum gray value, while those below the threshold are pushed to zero. As an example, a gray scale image having eight bits to represent each of its pixels has a maximum pixel value of 255. If a threshold pixel value of 128 is set, all pixels with a value less than 128 are set to 0 while those with a pixel value greater than 128 are set to 255. Figure 5.1 shows an example of a binarized image.

Figure 5.1: Original Image versus Binarized Image

After being thinned, these key frames should be line-connected such that a continuous contour can be generated. This preprocessing is necessary since the key frames we are going to handle are hand-drawings done by cartoonists, which means that lines may contain breaks and consistent geometric information cannot be assured as well [13]. A continuous one-pixel-wide contour is a precondition to perform line fitting using NURBS as control points can then be determined more accurately. Control points are the points that define a NURBS curve mathematically [16, 18, 37]. These points are needed to generate the corresponding NURBS curves for matching those segments that would not be properly

in-betweened by MFBA-IB.

Figure 5.2: SUSAN Thinning Rules

SUSAN thinning rules [33] are used for line thinning. *SUSAN* is an acronym for *Smallest Univalve Segment Assimilating Nucleus*. The rules are shown in Figure 5.2. They are summarized as:

- zero neighbour - remove the edge pixel;
- one neighbour - search for the neighbour with the maximum (non-zero) edge response, to continue the edge, and to fill gaps in edges. The responses used are those found by the initial stage of the *Susan* edge detector, before non-maximum suppression. They are slightly weighted according to the existing edge orientation so that the edge will prefer to continue in a straight line. An edge can be extended by a maximum of three pixels;
- two neighbours - there are three possible cases:
 - 1. If the pixel is “sticking out” of an otherwise straight line, then compare its edge response to that of the corresponding pixel within the line. If the

potential pixel within the straight edge has an edge response greater than 0.7 of the current pixel's response, move the current pixel into line with the edge;

- 2. If the pixel is adjoining a diagonal edge then remove it;
 - 3. Otherwise, the pixel is a valid edge pixel.
- more than two neighbours. If the pixel is not a link between multiple edges then thin the edge. This will involve a choice between the current pixel and one of its neighbours. If this choice is made in a logical consistent way then a "clean" looking thinned edge will result.

These rules are applied to every pixel in an image sequentially left to right and top to bottom. If a change is made to the edge image then the current search pixel is moved backwards up to two pixels leftwards and upwards. This means that iterative changes of value to the image pixels can be achieved using only one pass of the algorithm. Figure 5.3 shows an example of the thinning result.

Figure 5.3: SUSAN Thinned Image

Binarizing and thinning smooth the way for subsequent curve pixels acquisition and other pixel-related processings, e.g. connectivity search of pixels.

5.2 Interactive Image Segmentation

Segmentation of a given image is one of the most important techniques for image analysis, understanding and interpretation [20]. It is known that image segmentation is important but difficult. Feature-based image segmentation is performed using two basic image processing techniques [20] as follows:

- the boundary-based segmentation - it is often referred to as edge-based and it relies on the generation of a strength image and the extraction of prominent edges;
- the region-based segmentation - it relies on the homogeneity of spatially localized features and properties.

Both of these two methods segment images based on image color or gray scale, which could hardly work for line-drawing cartoon images that contain lines only. The segmentation required for developing a NURBS matching algorithm is subjective and only necessary for the portions that are envisaged or observed to be poorly matched via MFBA-IB. Hence, in order to ease the development of the NURBS matching algorithm, interactive segmentation is used. With user's manipulation, certain parts of the key frames that are difficult to be matched properly through MFBA-IB are segmented via a graphical user interface. Automatic segmentation algorithm can be another research topic which is not of concern in this research.

5.3 Line Fitting via NURBS Modeling

In our context, line fitting is a method to extract pixel-to-pixel relationships on a curve and the geometric continuity [39] out of a group of pixels in a line cartoon image. What we are interested in is line fitting in 2D. While some other applications do 3D surface fitting [38], it is not considered here. A good method to generate a NURBS curve is the interpolation of pixels or points [15, 16]. For an image segment, there are possibly many pixels, some may even be noise. Therefore the global approximation method is used to generate a NURBS curve which does not necessarily go through all pixels but

approximate them within a certain error bound. With enough pixels in close proximity, as can be assured because the control points¹ are extracted from an image segment, a very good approximation of the segments to be matched can be generated. There are two methods for fitting a curve with a set of given pixels:

- interpolation - the curve to fit will pass through all the given pixels. The limitation of this interpolation method is that in order to go through all the given pixels, the required number of control points will be enormous, therefore it is not applicable for implementation;
- approximation - the curve to fit may not necessarily pass through all the given pixels, instead it will approximate the given set of pixels.

It is known that NURBS curves are defined by control points, a knot vector and weights. In order to minimize the uncertainty of results and the number of variables, we focus more on how control points affect the shape of NURBS curves, which is important for line fitting and subsequent in-betweening. Therefore, if the number of control points is n , then the number of knots is $m = n + p + 1$, where the NURBS degree p is fixed to be 3. The knot vector is defined as:

$$u_i = \begin{cases} 0.0 & i \in [0, p + 1) \\ \frac{i-p}{n-p} & i \in [p + 1, n) \\ 1.0 & i \in [n, m) \end{cases} .$$

The number of weights is equal to the number of control points and these weights will be calculated during approximation. Line fitting with NURBS could result in better approximation compared to Bézier curves since the weights can vary during approximation process while Bézier curves are the special NURBS curves when the weights are all one. A better approximation for the original cartoon character may result in improved MFBA in-betweening, which is what this research is going to explore. In the following sections, how to obtain control points and improve MFBA in-between results with NURBS curves will be discussed.

¹A NURBS curve is defined by a set of control points.

5.3.1 Image Pixel Sampling

In line fitting process, it makes approximation meaningless to use all pixels from a curve to form a group of NURBS control points, as in this case in between any two physically-adjacent pixels, there are no more pixels to approximate. In another word, the pixels to form image segments are discrete by nature while NURBS curves can be continuous in the mathematical domain. In order to approximate an image curve by NURBS, we need to sample a certain number of pixels out to give a set of control points. And by defining a sampling distance for a cartoon image, we can model a cartoon character by modeling different segments of the character. Figure 5.4 shows a sampled image.

Figure 5.4: Sampling Grid On An Image

The sampled pixels may not be on any of the curves. The next step is to associate each sampled pixel to its closest pixel on any curve, in another word, to adjust the coordinate values x and y of each sampled pixel to snap onto a pixel of any curve if possible. A search around all the sampled pixels gives us the corresponding closest pixels on all curves. Given a pre-defined maximum searching distance, say 10 pixels, the search starts from 0-pixel away from the given sampled pixel, or the sampled pixel itself, reaching to 10-pixel away from the given sampled pixel. Whenever the search reaches a non-background pixel, the search stops. Figure 5.5 shows the process of such a search. In another word, the search

starts from the pixel itself, then increases the distance until it reaches the maximum distance. Whenever a pixel on a curve is determined to correspond to a sampled pixel, the search stops and the corresponding pixel coordinates are stored inside its sampled pixel data structure.

Figure 5.5: Searching An On-Curve Pixel

The sampled pixel stored with its closest curve pixel coordinates is called a *valid sampled pixel*. The *valid sampled pixel* concept eliminates the confusion between the pixels stored with curve pixel coordinates and the pixels that do not have corresponding curve pixels as are determined by the searching algorithm illustrated in Figure 5.5. By this means, all the valid sampled pixels are mapped to their closest pixels on the curve. This process gives a sampled image consisting of pixels as shown in Figure 5.6.

Figure 5.6: Mapped Images of Valid Sampled Pixels

The left image of Figure 5.6 shows the sampled pixels that are mapped on the curves of the thinned image². The right image of Figure 5.6 shows only the pixels stored in their respective sampled pixel data structures. Comparing these two images, it can be seen that the sampled pixels obtained maintain the geometric information of the thinned image. To human beings, it is easy to figure out how to link these sampled pixels together and recover the thinned image. However, the computer does not know how the original image looks like, all it knows is the sampled pixels information that is mapped according to the pixel values of the thinned image. Therefore, how to link the mapped sampled pixels together and reconstruct the thinned image becomes a challenging problem to the computer. The next section discusses the attempt to solve this problem.

5.3.2 Curve Pixel Connectivity

The problem with line fitting is that a cartoon image can comprise many complicated curves, which are difficult for computers to automatically detect and generate the curve pixel mappings. Therefore an attempt has been made to automatically discover and detect curves on an image. As one of the pre-processing steps for our proposed algorithm later

²The original image has to be thinned for subsequent experiments and discussions. As such, in most of the cases, we compare our results with the thinned image instead of the original image.

on, the connectivity or relationship of curve pixels has to be studied and analyzed.

Figure 5.7: Pixel Connectivity

As shown on Figure 5.7, the central pixel is considered to have eight neighbouring pixels with a pre-defined neighbouring distance. The connectivity of all the sampled pixels on any curve should be established such that any two adjacent sampled pixels on any curve are inter-connected with each other, i.e. two sampled adjacent pixels regard each other as neighbours. A successful established inter-neighbour relationship of curve pixels provides necessary traversal information for the proposed *pixel recursive traversal* algorithm as will be seen later. There are two methods to start exploring the neighbouring relationships of curve pixels:

- random selection of any valid sampled pixel, i.e. any mapped pixel;
- systematical iterative selection of all valid sampled pixels.

The second approach has been adopted for implementation. For the purpose of identifying search directions, the OpenGL coordinate system convention is used as shown in Figure 5.8. The Z-axis is not used for the in-betweening process. Hence the search for pixel connectivity between valid sampled pixels starts from the bottom left of the sampled image, which is (0,0). The iteration of the search continues along the X-axis, then the Y-axis.

Figure 5.8: Coordinate System

Whenever a *valid sampled pixel* is reached, its eight neighbouring sampled pixels are involved in the connectivity search. The eight neighbouring sampled pixels under consideration have to be *valid sampled pixels* because only *valid sampled pixels* contain on-curve pixel information. Although a *valid sampled pixel* can have more than one valid neighbouring sampled pixel, it may not be on the same curve with those pixels due to non-zero sampling distance. Whether a *valid sample pixel* is on the same curve with its valid neighbours is determined as shown in Figure 5.9.

Figure 5.9: Connectivity Search for A Valid Sampled Pixel

In order to determine whether *pixel P* and *pixel NP* are on the same curve, a middle *pixel MP* between them is considered. A search around *pixel MP* is conducted in the same way as shown in Figure 5.5. If the search shows that there is a curve pixel inside the search area around *MP*, *P* and *NP* are considered to be on the same curve. This judgment is based on:

the hypothesis - pixel P and pixel NP are on the same curve and the curve variation between pixel P and pixel NP is small enough, so that the search around pixel MP is guaranteed to find a pixel which is on the same curve as pixel P and pixel NP .

The extreme situation when the assumption is no longer needed is to make the distance between P and NP to be zero, which means zero sampling distance, i.e. all image pixels are taken as sampled pixels. The smaller the sampling distance, the more precise the determination is, but more redundant processing. When the pixel connectivity search is over, a connectivity map of all valid sampled pixels is generated. This provides the information required for further processing.

Once the connection map is generated, there is a need to classify the curves that forms a cartoon character. A NURBS curve contains a chain of control points, which does not have any branches. In order to provide matching information for NURBS in-betweening process, the curves of a thinned image should be classified into non-intersecting curves, which do not have any branches or intersections. As shown in Figure 5.10, for a general line-drawing image, branching curves exist by nature, which implies the difficulty of curve classification. For the curves that have no intersections or branches, with the information obtained in the previous processing of connectivity search, ideally, if the pre-defined neighbouring distance is small enough, a systematic trace to all the pixels on a particular curve is possible. For the curves with intersections or splitting branches, a recursive search starting from the branching pixel can be conducted, which possibly results in a new curve if the further search discovers some of the neighbouring pixels that do not belong to any known curves.

Figure 5.10: Branching Curve

As shown in Figure 5.10, from the starting pixel P , there are many possible ways to traverse and classify the curves, e.g. $P-1-5$, $P-1-2-3-4-6$, or $P-1-2-4-3-7$. Therefore there can be different sets of curves that each can construct the same character. For a general line-drawing cartoon image, the number of such sets or combinations of curves can be huge. Therefore, there is a need to systematically classify the curves on a cartoon image for further matching purpose. Although this is not an easy task, we yielded encouraging results by designing and implementing a generic algorithm for this.

5.3.3 Pixel Recursive Traversal Algorithm

As said in the previous section, we are interested to design a generic method to automatically classify curves on a cartoon image to assist subsequent NURBS curve matching. Some of the curves on a cartoon image have no intersection pixels, while some are intersecting each other via common pixels. For these intersecting curves, they cannot be discovered straightforwardly as every intersection pixel is part of any curves that cross it over. And some curves may even have more than one intersection pixel. All these scenarios have to be handled in a systematical manner, therefore a *pixel recursive traversal* algorithm is proposed to automatically detect curves on a cartoon image as follows:

Algorithm 1 Pixel Recursive Traversal

1. Binarize and thin the image.
 2. Establish neighbouring point connectivity for all the valid sampling points.
 3. For all the valid sampling points, process as 4.
 4. Recursive function for any valid sampling *point p*:
 - while point p* has a connected *point a*,
 - remove *point a* from *point p*'s connectivity.
 - If point a* has connectivity to *point p*,
 - remove *point p* from *point a*'s connectivity.
 - end If*
 - If* there is no *global curve*,
 - make a new *global curve*, append *point p* and *point a*, *point p* will be the head of the new curve.
 - else*
 - append *point a* to the *global curve*.
 - If point a* is on any known or detected curve,
 - append *point a* to the *global curve* if there is any;
 - the *global curve* ends and mark no *global curve*;
 - continue process *point p*'s connections.
 - else*
 - process connections on *point a* (recursive call to 3).
 - end of while*
 - If* there is any *global curve*,
 - append *point p*;
 - add the *global curve* to curves list;
 - mark a flag to indicate no *global curve*.
 - end of If*
-

The algorithm uses a recursive approach to solve the branching problem of curves. It is explained as follows:

1. *Binarize and thin the image.* This step performs the necessary preprocessing for subsequent processes.
2. *Establish the neighbouring pixel connectivity.* This step determines the inter-relationship

between the valid neighbouring sampled pixels, which is critical for step 3. How to establish this connectivity has been explained in section 5.3.2.

3. *For all the valid sampled pixels, process as step 4.* This step simply calls a recursive function which is implemented in step 4.
4. *Recursive search for curves.*

Start from a valid sampled pixel p , for each of its valid neighbours a , one at a time, remove their neighbours from p 's connectivity such that the recursive function will stop even if the curve is circularly connected. Meanwhile, if pixel a has connectivity to pixel p , remove p from a 's connectivity. This is to prevent backward linking when pixel a is passed into the recursive function and becomes another pixel p .

During the recursive function call, a *global curve* will be updated with sampled pixels. If there is no *global curve*, a new curve will be created and set to act as the *global curve*. In that case, pixel p becomes the head of the new curve and is followed by pixel a . Otherwise, pixel a will be appended to the *global curve*.

If pixel a is on any known curve, which means the *global curve* has intersection with a known curve, append pixel a to the *global curve*. In this case, the *global curve* ends and the search stops. The search continues for the other neighbours of pixel p .

If pixel a is *not* on any known curve, it will be passed to the recursive function and the search starts from pixel a . This is the function of the recursive function, which enables the trace to link a set of *valid sampled pixels* into a curve.

When pixel p has no more neighbours to process, append it to the *global curve*, add the *global curve* to the curve list, set a flag to indicate there is no *global curve*.

At the end of the recursion, all the curves on an image are detected. Each of the curves is just a set of *valid sampled pixels*. By linking all these sampled pixels, the thinned image

is reconstructed. Figure 5.11 shows the result of searching and linking *valid sampled pixels* in eight directions. That is to say, eight neighbours are considered for searching the connectivity relationship of *valid sampled pixels* (Figure 5.7 provides an illustration).

(a) is the thinned image. (b) is the reconstructed image.

Figure 5.11: Result of Connectivity Search in Eight Directions

The reconstruction result shown in Figure 5.11 can be refined by comparing it with the thinned image such that the redundant curves can be removed. Figure 5.12 shows the refined result. The key feature about the reconstructed image in Figure 5.11 is that the computer extracts all the information from the thinned image, stores the information in a proper data structure and then reconstructs the whole image by using the stored data structure. Each of the curves is a list of pixels that are sampled from the thinned key frames.

(a) is the thinned image. (b) is the refined result.

Figure 5.12: Refined Result of Automatic Reconstruction

We now show another example with automatic curve reconstruction and propose simple rules for automatic in-betweening with computer reconstructed curve models. Figure 5.13 shows the original key frames and Figure 5.14 shows the automatic reconstructed result.

Figure 5.13: Key Frames - Wave

By comparing Figure 5.13 and Figure 5.14, we can observe that the auto-reconstructed frames in Figure 5.14 maintain a global resemblance to the original geometric information.

Figure 5.14: Automatically Reconstructed Key Frames - Wave

The following rules or matching criteria are used for matching the curves on the reconstructed frames:

1. The average pixel coordinates for two matched curves should be less than a pre-defined distance. The test in Figure 5.15 used 100 pixels for this distance.
2. The total pixel numbers of two matched curves should differ less than a pre-defined percentage of the pixels in the longer curve. The test in Figure 5.15 used 20% for this value.

The purpose of defining these two simple criteria is to gather heuristic statistics for future research reference, i.e. the result of using these two simple criteria helps us to refine our future research direction. Figure 5.15 shows the automatic in-between results and we can see that the automatic reconstructed key frames actually contain more curves than what human eyes should perceive, therefore the in-between results have more curves than subjectively acceptable. This shows the difficulty of automatic in-betweening with NURBS and for our present research, we focus on interactive in-betweening with NURBS.

5.3.4 Result Analysis and Discussion

Figure 5.11 shows that the reconstructed character maintains most of the original pixel geometric information, but it is not exactly the same as the thinned image. The reason is that in the thinned image, there are many small curves that are close in proximity to each

(a) and (f) are the automatic reconstructed key frames as in Figure 5.14.

(b), (c), (d) and (e) are the automatic in-between NURBS test results.

Figure 5.15: Auto-NURBS In-Between Test Results

other and this violates the assumption made for the pixel connectivity in section 5.3.2. For example, in Figure 5.16, the top image shows a scenario where some valid sampled pixels are close to each other like $P1$ and $P2$, $P3$ and $P4$, $P5$ and $P6$, $P7$ and $P8$. These pixels are considered to be neighbours to one another. This is because the sampling distance is not zero and neighbouring pixels here mean two pixels that are apart from one another by the amount of sampling distance, e.g. if the sampling distance is 10, two neighbouring pixels on a curve mean they are apart from one another by 10 pixels. Therefore, during the search of neighbouring pixels, they are regarded as neighbors, even though they are not on the same curve.

Figure 5.16: Curve Reconstruction Error

As shown in the bottom image of Figure 5.16, during the reconstruction of curves, the pixels $P1$ and $P2$, $P3$ and $P4$, $P5$ and $P6$, $P7$ and $P8$ are linked together, which results in redundant curves. This explains why Figure 5.11 has more curves than the thinned image. At the moment, this algorithm can work well for the simple images in which the curves are not close to each other. In Figure 5.17, the right image is one reconstructed from the left image. The *pixel recursive traversal* algorithm does work on this example. Also, Figure 5.14 is another example of better automatic key frame reconstruction.

Figure 5.17: Automatically Reconstructed Frame

Although our automatic reconstruction algorithm is not able to perfectly handle all kinds of key frames, it shows an attempt to reconstruct and model a cartoon character with NURBS curves. Figure 5.12 shows the refined result by removing redundant curves. It looks much better than Figure 5.11. A better result can be expected when the sampling distance reduces to zero and all the pixels of an image in such case would be processed. However, this is not implemented as when the sampling distance reduces, more noise effect appears and more effort is required to refine the result. If the distance reduces to zero, between any two adjacent sampled pixels of a curve, there is no more pixels and therefore there will be no need to use NURBS for approximating those pixels in between any two adjacent sampled pixels. There is a need to do more specific research on this particular problem.

After the search for curves is done, the sets of *valid sampled pixels* can model the curves via NURBS for further matching purpose. Although the line fitting process with the *pixel recursive traversal* algorithm is not perfect, it points out a direction to improve NURBS matching. Due to the limitations of the current line fitting algorithm as shown above, there is room left for future research in this area.

5.4 NURBS In-Betweening

With the result of line fitting, the features of NURBS modeled curves in the two images can be used for matching. Three steps need to be performed here for reasonable matching:

- define feature points of the segments;
- match the feature points;
- compute the displacement vectors of any two corresponding feature points in the matching segments.

In order to reasonably match the segments, feature points need to be defined, in our case corner points are used. The reason for defining feature points is the geometric information of the segments should be maintained during the NURBS in-betweening process, otherwise the generated in-between frames may be distorted. Feature points matching is a critical issue, which involves the search of the matching feature points and the criteria for choosing a reasonable corresponding feature point. After the feature points of a pair of segments are identified, the displacement vectors of any two corresponding feature points can be computed. The number of feature points in the matching segments may not be the same, therefore there would be three types of matching:

1. one-to-one matching;
2. many-to-one matching;
3. one-to-many matching.

The *one-to-one matching* would be straightforward, unless in trivial cases, it is unlikely to happen in real images. The *one-to-many matching* could be converted to the *many-to-one matching* by reversing the order of the matching portions, hence we have only *many-to-one matching* left. Unlike one-to-many matching, it is easier to identify a unique target pixel in the many-to-one matching. Having all the displacement vectors defined, the in-betweening could start by analyzing the displacement vectors.

In order to simplify the process for verification purpose, *interactive feature point extraction* method is used. With the interactive modeling [35, 42], users are granted more controls over the curves and shapes. If the result proves acceptable with interactive extraction, automatic feature point extraction could be future work to enhance what has been done here for the *interactive feature point extraction*.

5.5 Interactive Feature Point Extraction

In order to isolate the difficulty of automatic NURBS matching discussed in section 5.3.4, from the issue of NURBS in-betweening, we use interactive feature point extraction for NURBS matching. A graphical user interface program is built with *OpenGL User Interface (GLUI)* to facilitate the interactive in-betweening via NURBS curves. GLUI is a portable API for building OpenGL based user interface programs. The graphical user interface program is used for interactive processing purposes like image display, image thinning, interactive feature point extraction, NURBS curve manipulation, matching of NURBS curves and reviewing of in-betweening results.

Figure 5.18: Interactive Feature Point Extraction

Figure 5.18 shows the graphical interface program and the example of fitting a NURBS

curve on a thinned image. Each loaded image has a workspace associated with it, which stores the image related information, e.g. the NURBS curves of an image, whether the thinned image or the original one should be displayed etc. A user could interactively specify the feature points to be matched on an image and generate the NURBS curves based on the feature points. When the user switches between different images, the interactive work he/she has done will be stored, or restored and redisplayed.

Whenever a user specifies a feature point on a thinned image by clicking the mouse, an automatic search around the clicked point gives the nearest pixel coordinates x and y . In this way, the user does not need to worry whether the point he/she specifies is actually an on-curve pixel or not. The program accurately adjusts the clicked point coordinates and makes sure the specified feature point is on a particular curve. If any two points specified by a user are close enough, it is possible that they are adjusted to the same curve point. This can be further improved by adding image zooming feature to the interactive application.

5.6 Generation of In-Between Curve

The curves of the in-between segments can be modeled and generated via NURBS by obtaining the corresponding in-between feature points. The in-between feature points are acquired by partitioning the start-to-end displacement vectors of the matched feature points and adding the partitioned displacement vectors to the starting segment feature points. The linear in-betweening is represented by the formula as:

$$f(\mathbf{u}') = f(\mathbf{u} + \frac{\vec{\mathbf{d}}}{n}) \quad (5.1)$$

where $f(\mathbf{u}')$ is the in-between pixel gray level. $f(\mathbf{u})$ is the gray level of pixel on the starting segment. $\vec{\mathbf{d}}$ is the displacement vector of any two corresponding feature points in the starting and ending key frames. n is the number of in-between frames needed to be generated, $\mathbf{u}$ is the pixel coordinate on the starting image, $\mathbf{u}'$ is the corresponding in-between pixel coordinate.

Once having the displacement vectors determined for all the feature points involved in the in-betweening process, NURBS curves can be generated for the different sets of in-betweened feature points. As an example, Figure 5.19 shows one set of NURBS curves formed by in-betweened feature points. These NURBS curves become the in-betweened portions for key frames.

Figure 5.19: NURBS Curves and Feature Points

However, since 2D images are considered, after the cartoon characters on the starting and ending key frames are line-fitted with NURBS curves, the in-between feature point coordinates are computed as:

$$x' = x + \frac{\vec{x}}{n+1} * (i+1) \quad (5.2)$$

$$y' = y + \frac{\vec{y}}{n+1} * (i+1) \quad (5.3)$$

where $\vec{x}$ and $\vec{y}$ are the displacement vectors for any two matched feature points, x' and y' are the in-between feature point coordinates and i is in the range of $[0, n-1]$. The gray level function 5.1 is not considered because after thinning an image, all the pixels have a gray level of either 0 (black) or 255 (white) only. Therefore, the in-between pixel coordinate

equations 5.2 and 5.3 are used to obtain the in-between feature points' coordinates and the gray level of the in-between feature points is 0 (black), by default.

The curves to be matched on the original images should be removed when the in-between curves are to replace the original curves as in our implementation we re-use the image buffers. The *divide-and-conquer* approach is used for the removal of the original curves, i.e. the original curves are partitioned into interval areas formed by any two adjacent feature points. A straight line links any two adjacent feature points as shown in Figure 5.20.

Figure 5.20: Curve Removal

The reasoning is based on the assumption that enough feature points along the original curves are obtained such that the straight lines connecting all the feature points approximately fit into the original curves. Every pixel along the straight lines is considered and a small rectangle area around the pixel is searched for the original pixels and once found, the original pixels are removed. In this way, we remove the original curves and re-use image buffers. The above discussion explains our specific implementation for re-using image buffers in the interactive in-betweening process. The reason for re-using image buffers is to maintain the pixel values for the other portions of a cartoon character, which are not involved in the segmented in-betweening but still contribute to the cartoon character.

5.7 Integrated Solution for In-Betweening

Based on other research work [2, 9, 40] and the above experiments and findings, an integrated solution for in-betweening is proposed. Given a number of pairs of key frames,

- MFBA algorithm and the distribution model (Section 4.1.1) are applied. These key frames are processed in a distributed environment in order to achieve an overall processing speed-up compared to local MFBA-IB processing;
- Interactively segment the key frame portions that are observed having discouraging in-between motion via MFBA-IB;
- Do the interactive NURBS matching;
- Synthesize the final in-between results and generate a sequence with NURBS-improved in-between motion.

Figure 5.21 shows the work flow of this solution.

Figure 5.21: Integrated In-Betweening Process

A graphical user interface takes key frames as input and distributes the key frames to the clustering computers for MFBA in-betweening process. The users are allowed to browse the MFBA generated in-between frames through the user interface. With human's manipulation, interactive segmentation is performed on the matched portions that have

discouraging in-between motion as shown in Figure 5.22 by comparing the original key frames and the MFBA generated in-between frames.

Figure 5.22: Segmentation of Key Frames

At the moment, the NURBS matching can only be done restrictively in the interaction mode. Users have to interactively generate NURBS curves by fitting points on key frames as shown in Figure 6.3. NURBS based in-between results are synthesized into in-between frames when portions of key frames are matched, or inserted into the MFBA generated in-between sequence. A proper in-between sequence is generated by adding some post-processing techniques when required, e.g. color enhancement and line thickening.

5.8 Discussion

The NURBS-improved in-between matching solution explores the possibility to generate more acceptable in-between frames by combining computer graphics and image processing techniques. There are a number of steps to follow in our proposal. Although in some situations we achieve improved results, it is still far from being perfect. Further research is needed to explore the solutions to certain specific problems, e.g. large motion changes in key frames.

In our procedures for NURBS-improved in-between matching, we do line thinning at the first step. As a result, we need to thicken the in-betweened curves for consistency with the original key frames. In that sense, line thickening is a kind of post-processing technique to obtain similar curve characteristics compared to the original curves. There are a few methods to thicken in-between curves. One of them would be to compare the corresponding feature points of in-between curves with their starting and ending curves. The *region growing* [20] idea for image segmentation could be adopted here, i.e. feature points can be the centers for growing the width of curves. This is not tackled as our focus is on curve matching.

Automatic line fitting is not as easy as it appears to be. In our research context, line fitting is the procedure to model a full cartoon character in a suitable structure via NURBS. It needs to capture the essential characteristics of a cartoon character and store the available information for NURBS-improve in-between matching. Our effort of doing automatic line fitting is illustrated in section 5.3. Based on automatic line fitting, we automatically reconstruct key frames and generate in-between frames via NURBS modeling. Although certain progress has been made, automatic curve fitting and matching remains as a very difficult problem. As one of the solutions for this situation, we attempted interactive NURBS in-between matching by specifying feature points via human manipulation.

By combining our research results, we proposed an integrated solution for applying in-betweening steps from start to end. A research framework with graphical user interfaces and utility programs is developed to tackle real cartoon animation production. The next chapter shows the supporting experimental work that has been done, e.g. the graphical user interface program for distributed processing of MFBA-IB, GUI for NURBS-improved matching and its results.

Chapter 6

Experimental Results and Discussion

In comparison to the original local MFBA-IB processing, the overall processing throughput of hand-drawing cartoon frame in-between generation has been improved via our distributed and parallel communication model. Also, the specific in-between motion problem with MFBA-IB has been improved by the NURBS matching algorithm. However, the results show that there is still room to improve on the in-between matching as it is not an easy task and the in-between frame quality is measured subjectively by observation. Despite the limitations and constraints of proposed in-between matching via NURBS, the NURBS matching algorithm shows a good attempt to do in-betweening by combining computer graphics and image processing techniques. In tackling real cartoon animation production, a framework of user interface programs to utilize our APIs is developed. Experiments on the proposed algorithms with the developed GUI are conducted and the results are discussed in the following sections.

6.1 Graphical User Interface

In our research work, a *Graphical User Interface* (GUI) program has been built to schedule and monitor the distributed jobs running on clustering computers. The GUI makes use of the distributed communication API in *Section 4.1*. The graphical user interface program is designed and implemented via Qt and OpenGL as shown in Figure 6.1.

Figure 6.1: Scheduling GUI for Distributed MFBA-IB Processing

The graphical user interface program provides interactive functionalities such as browsing images, scheduling key frame pairs and monitoring the status of a computer cluster running MFBA-IB etc. The in-between frames after MFBA-IB processing will be shown in the file browser of the user interface program. The backend program is implemented as an integration of MFBA-IB and the distributed processing agent designed in *Chapter 4*. It runs on the clustering computers and does distributed MFBA-IB processing. It is also responsible for communication with the front-end graphical user interface program. When a pair of key frames is scheduled for running, it is distributed over to the backend program for MFBA-IB processing. Compare to the original local MFBA-IB processing, this achieves better overall throughput performance as discussed in *Chapter 4*.

6.2 Interactive NURBS In-Betweening

A graphical user interface program is designed for the interactive NURBS in-between matching as shown in *Figure 6.2*. It is implemented via GLUI API, which is a portable OpenGL graphical user interface API integrated with GLUT (Graphics Library Utility

Toolkit).

Figure 6.2: GUI for Interactive NURBS In-Between Matching

The program provides interactive assistance for animation artists to perform NURBS in-between matching, and basic interactive operations as well, such as loading and displaying key frames, storing and restoring working sessions. An animation artist can specify feature points interactively to define cartoon portions to match by drawing appropriate NURBS curves on the starting and ending key frames through this program. Then the NURBS curves are matched according to the principles that have been explained and discussed in Chapter 5.

Figure 6.3 shows an example of matching two key frames *faceA.tif* and *faceB.tif*. There are three curves on the key frames to be matched. These curves are drawn interactively through the user interface program. Notice that the pixels on all these three curves have disparate motion change directions. This is one of the situations we want to improve for MFBA-IB.

Figure 6.3: Interactive Segmentation and Curve Fitting On Two Key Frames

The results of the NURBS-improved matching are shown in Figure 6.4. We can see that the results are reasonably good. It indicates that the NURBS-improved matching algorithm has the potential for improving in-betweening on certain cases although it takes time to draw the curves interactively at the moment.

Figure 6.4: NURBS In-Between Frames

Interestingly, the results generated by MFBA-IB are shown in Figure 6.5. By comparing Figure 6.4 and Figure 6.5, we can see that the MFBA-IB in-between motion of the eyes and mouths is not as observable as the NURBS generated in-between motion.

Figure 6.5: MFBA In-Between Frames

However, the interactive NURBS in-between algorithm is still not perfect. Figure 6.6 shows another example of matching two cartoon key frames using the interactive

NURBS in-between matching algorithm. It can be noticed that the fox's tail in the in-between frames are not properly connected to the rest of the cartoon character. This is the limitation of interactive NURBS matching, which does not appear in MFBA-IB matching as we can see from Figure 6.7. The difference between Figure 6.6 and Figure 6.7 is that the interactive NURBS matching handles part of a cartoon character while MFBA-IB matching automatically takes care of the whole cartoon character. The solution to this at the moment would be to model every connected curve of the matching segments we are interested in. This will increase interactive work load for sure, therefore, until it is totally automatic, the NURBS-improved matching is more suitable for simple cartoon frames or isolated segments (no connection to the rest parts of a character) that are observed having discouraging in-between motion via MFBA-IB.

Figure 6.6: Interactive NURBS In-Between Frames

Figure 6.7: MFBA In-Between Frames

6.3 Summary

In this chapter, we presented our research achievements on improving MFBA-IB in two fields: distributed processing and NURBS-improved matching. In order to push our research results into user-applicable level instead of leaving them on paper, we demonstrated how we applied our research work in previous chapters on providing user-friendly tools for animation artists. Our backend research work is presented to animation artists by the front-end interactions through GUI, which covers the underlying complicated processing. These efforts help us to tie our research much closer to industrial applications.

The distributed MFBA-IB has been demonstrated to gain better throughput performance in comparison to the original local MFBA-IB. It makes processing a large number of key frame pairs into reality with the aid of our GUI program. Our distributed and parallel processing model API has also been integrated with MFBA-IB in the GUI program to provide user-friendly interaction between animation artists and MFBA-IB. More results and discussions on the distributed MFBA-IB can be found in Chapter 4.

In-between motion improvements are also shown for certain cases of cartoon frames via interactive NURBS in-between matching. However, some of the NURBS in-between results are not that satisfactory as expected due to the preservation of relationship to the original neighbouring curves is not good enough. The reasons are:

- the curves have inter-connected points, which are not matched accordingly;
- the in-betweening curves sometimes cannot connect properly to the rest of a cartoon character;
- the points specified by users sometimes are not equally spreaded, therefore the curve fitting result has larger disparity with the original curves.

Although the current results are not perfect, the NURBS-improved matching is believed to be a good attempt to make in-betweening more subjectively acceptable in future as the geometric information of matching curves are preserved by modeling via NURBS mathematically. With more effort put in, it may become a novel matching algorithm.

Chapter 7

Future work

As can be seen in this thesis, we have yielded encouraging results in improving MFBA in-between algorithm, which inspires further research to make MFBA-IB more efficient and applicable for a larger variety of images. Despite the limitations on performance throughput and in-between motion, the MFBA in-between algorithm is a great effort for reducing the labor for cartoon artists. In our research, a distributed algorithm and an interactive NURBS matching algorithm are proposed to improve MFBA-IB. They are not perfect but show encouraging improvements on the in-betweening process. Upon what has been done so far, there are a few more areas where the in-betweening process can be further refined and improved.

1. Several independent cartoon characters in a pair of key frames can be automatically and dynamically detected and processed via our distributed algorithm. We have shown an example in Figure 4.10 by interactively segmenting the two independent characters and distributed for MFBA-IB processing (Figure 4.11).
2. For NURBS-improved matching, the automatic curve detection on the original image and the automatic feature point acquisition can help to automate the interactive NURBS algorithm. Figure 7.1 shows a possible workflow. In this way, the entire in-betweening process can be automated with improved in-between motion and faster processing throughput for a large number of key frames with the aid of distributed processing. And a curve model can be set up to represent a whole cartoon character.

Figure 7.1: Auto-NURBS In-Between Matching

Once it is done, mathematical approach via NURBS can be applied on the model and the correspondence rules for matching [19] can be defined accordingly.

3. The distributed processing API can be further developed to enhance the system performance of any in-between algorithm that is integrated with the API. Due to the fact that automatic in-betweening is a very difficult artificial intelligence problem, some more research work can be done to make it more applicable in the cartoon production and ease the work load of animators to a greater extent.
4. Missing 3D information is a problem for in-betweening as the cartoon images are 2D renditions of 3D objects. This problem can be solved by extending 2D NURBS modeling into 3D modeling and re-constructing the 3D information based on a key frame sequence. There will be much more difficulty in doing this, but it is believed to produce satisfactory results once it is done.
5. Automatic NURBS-improved matching process can be developed to work with the distributed MFBA-IB such that the discouraging in-between motion problem can be further improved with less human interaction.

MFBA-IB has been improved in its processing throughput and in-between motion quality by introducing distributed processing and NURBS techniques in our research. As can be seen in the above discussions, in order to further develop and enhance MFBA-IB, there is still much room for the research in computer-assisted in-betweening. And it is believed that in the future MFBA-IB becomes one of the industrial tools for the in-between frame generation and other novel algorithms and systems will also be developed.

Chapter 8

Conclusions

As can be seen in the previous chapters in this thesis, we have achieved encouraging results in improving MFBA-IB in two fields: distributed processing and NURBS-improved matching. The performance throughput of MFBA-IB has been improved by our proposed distributed processing approach. The discouraging in-between motion problem of MFBA-IB has also been improved by our proposed NURBS-improved matching algorithm. The attempts to address the in-betweening problems that MFBA-IB faces have yielded encouraging progress and improvements, which inspires further research in computer-assisted in-betweening area.

As has been discussed, MFBA-IB computes the pixel level motion between two key frames in a hierarchical motion estimation approach. It is a computationally intensive process which makes it time-consuming to process a large number of key frame pairs. As the MFBA-IB algorithm does not consider the temporal information between pairs, the key frame pairs are processed in a sequential order. Distributed image processing technique is thus introduced to process key frame pairs in a parallel fashion to improve the overall throughput. With the assumption that different pairs of key frames have almost the same image complexity, the overall distributed processing time of MFBA-IB is expected to be similar to one pair of local MFBA-IB processing as the time to transfer a few megabytes of image content over network is negligible in comparison to a few minutes of local MFBA-IB processing time.

In order to utilize more computation power to gain better performance throughput, a distributed API has been designed in our research to ease the effort for deploying the MFBA-IB algorithm as well as other in-between algorithms over clustering computers. The API is designed in a flexible manner such that it can be used to integrate with other computationally intensive applications. It is also reliable since it is implemented via the socket communication based on TCP/IP in Linux. Due to its native Linux implementation, the API is robust and cost effective compared to a Microsoft windows implementation that is vulnerable to virus and expensive in OS license. MFBA-IB therefore has been improved in our research in terms of robustness and lower cost. A graphical user interface program is also designed and implemented to integrate with the distributed API and provide user manipulation abilities, e.g. image viewing, key frame pair scheduling and processing status monitoring etc. The result of such a distributed MFBA-IB processing mechanism shows an efficient way to process a large number of key frame pairs in a much shorter time.

MFBA-IB works well for general cartoon images, however, it has certain limitations, out of which the discouraging in-between motion problem dominates in the image portions that have disparate pixel motion change directions. In order to tackle this in-between motion problem inherent with MFBA-IB, an interactive NURBS-improved matching scheme is proposed to introduce computer graphics technique into in-betweening for improving MFBA-IB pixel motion. Users can model the curves on cartoon images via NURBS with the aid of our GUI program. And the NURBS curves on key frames are then matched with the user pre-defined feature points. In this way, the pixels with disparate motion change directions are matched more accurately. As we have seen in the previous chapters, MFBA-IB with the NURBS-improved matching shows an encouraging result, which proves the success of our effort in improving in-between pixel motion for MFBA-IB.

In order to automate the NURBS-improved matching scheme for MFBA-IB, an automatic curve detection algorithm called *pixel recursive traversal* is developed to automatically discover and model all the curves in a pair of cartoon key frames via NURBS and match these curves accordingly. This algorithm works well for simple cartoon frames, therefore it is the foundation of further development for complicated cartoon frames. This

algorithm also reduces human manipulation for the interactive NURBS-improved MFBA-IB. It is optimistic that a fully-automated NURBS matching algorithm comes in the near future and produces accurate in-between pixel motion.

Finally, although encouraging results have been achieved for our research in improving MFBA-IB, automatic in-betweening is not an easy task, which has been mentioned in the introduction chapter, therefore it has to be noticed that there are always certain drawbacks with various algorithms, so are with the distributed MFBA-IB scheme and the NURBS-improved MFBA-IB algorithm. Hence, there is still room to further enhance and improve MFBA-IB. However, in view of what we have done in this thesis, our effort has resulted in one step towards the goal of making a computer-assisted in-betweening system for real cartoon animation production.

Bibliography

- [1] Hock Soon Seah and Feng Tian. Computer-assisted coloring by matching line drawings. *The Visual Computer*, 16(5):289–304, June 2000.
- [2] Hock Soon Seah and Ji Lu. Computer-assisted in-betweening of line drawing: Image matching. In *7th International Conference on Computer-Aided Design and Computer Graphics*, pages 193–200, 2001.
- [3] Ju Jia Zou and Hong Yan. Vectorization of cartoon drawings. In *Pan-Sydney Area Workshop on Visual Information Processing 2000*, Sydney, Australia, 2000.
- [4] Jessica K. Hodgins and James F. O'Brien. Computer animation. College of Computing and Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA 30332-0280, 2002.
- [5] Quan Chen, Feng Tian, Hock Soon Seah, Jie Qiu, and K. Melikhow. Automatic inbetweening based on feature matching. In *International Workshop on Technology, IWAIT*, 2004.
- [6] Edwin Catmull. The problems of computer-assisted animation. In *Computer Graphics, SIGGRAPH*, Computer Graphics Lab. New York Institute of Technology, Old Westbury, New York 11568, 1978.
- [7] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from image sequences. Springer-Verlag, 1993.

Bibliography

- [8] Michael van de Panne. *Control Techniques for Physically-Based Animation*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, 1994.
- [9] Fabian Di Fiore, Philip Schaeken, Koen Elens, and Frank Van Reeth. Automatic in-betweening in computer assisted animation by exploiting 2.5d modelling techniques. In *Computer Animation 2001, IEEE Inc*, Expertise Centre for Digital Media - Limburg University Centre. Wetenschapspark 2, B-3590 Diepenbeek, Belgium, 2001.
- [10] Nestor Burtnyk and Marceli Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Communications of the ACM*, 19(10):564–569, October 1976.
- [11] William T. Reeves. In-betweening for computer animation utilizing moving point constraints. *Computer Graphics*, 15(3), August 1981.
- [12] Alexander Kort. Computer aided in-betweening. In *NPAR 2002*. Fraunhofer Institute for Applied Information Technology, 2002.
- [13] Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas, and Frédéric Taillefer. Tictactoon: A paperless system for professional 2d animation. In Robert Cook, editor, *Proceedings of ACM SIGGRAPH 95, Computer Graphics Proceedings, Annual Conference Series*, pages 79–90, New York, 1995.
- [14] Philippe Lavoie. An introduction to nurbs, January 1999. <http://libnurbs.sourceforge.net/nurbsintro.pdf>.
- [15] Philippe Lavioe. An introduction to nurbs++, April 1999. <http://libnurbs.sourceforge.net/user.pdf>.
- [16] C. Ramakrishnan. An introduction to nurbs and opengl. MAT 200C. Winter 2002.
- [17] David F. Rogers. *State of the Art in Computer Graphics - Visualization and Modeling*, pages 225–269. Springer-Verlag, New York, 1991.

Bibliography

- [18] Les Piegl. On nurbs: A survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, January 1991.
- [19] Ramesh Raskar. Edge-based correspondence using the viterbi algorithm. Master's thesis, Computer Science, the Graduate College of the University of Iowa, August 1995.
- [20] Nikos Paragios and Rachid Deriche. Coupled geodesic active regions for image segmentation. Technical report, Thème 3 – Interaction homme-machine, images, données, connaissances Projet Robotvis, Octobre 1999.
- [21] Benoit Gennart and Roger D. Hersch. Computer-aided synthesis of parallel image processing applications. In *Parallel and Distributed Methods for Image Processing III, SPIE International Symposium on Optical Science, Engineering and Instrumentation*, volume 3817, pages 48–61, Denver, Colorado, July 1999.
- [22] Jeffrey M. Squyres, Andrew Lumsdaine, Brian C. McCandless, and Robert L. Stevenson. Cluster-based parallel image processing. Technical report, Laboratory for Scientific Computing. Department of Computer Science and Engineering, University of Notre Dame, IN 46556, August 1996. Submitted to the Journal for Parallel and Distributed Computing.
- [23] Xiangjian He, Tom Hintz, and Qiang Wu. A skeleton algorithm on clusters for image edge detection. In *15th International Parallel&Distributed Processing Symposium, San Francisco*, pages 134–140, April 2001.
- [24] Jeffrey M. Squyres, Andrew Lumsdaine, Brian C. McCandless, and Robert L. Stevenson. A cluster-based parallel image processing toolkit. In *IS&T Conference on Image and Video Processing*, San Jose, CA, February 1995.
- [25] J. You, W. P. Zhu, H. A. Cohen, and E. Pissaloux. Fast object recognition by parallel image matching on a distributed system. In *7th IEEE Symposium on Parallel and Distributed Processing*. School of Computing & Information Science, South Australia University, October 1995.

Bibliography

- [26] Fabian Di Fiore and Frank Van Reeth. Employing approximate 3d models to enrich traditional computer assisted animation. In *Proceedings of the Computer Animation 2002 (CA 2002) 1087-4844/02 2002 IEEE*, Expertise Centre for Digital Media - Limburg University Centre, Wetenschapspark 2,B-3590 Diepenbeek, Belgium, 2002.
- [27] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18(3), July 1984.
- [28] Jamshed N. Patel, Ashfaq A. Khokhar, and Leah H. Jamieson. Implementation of parallel image processing algorithms in the cloner environment. In *IEEE Workshop on VLSI Signal Processing*, pages 83–92, 1994.
- [29] Cheolwhan Lee, Yuan-Fang Wang, and Tao Yang. Global optimization for mapping parallel image processing tasks on distributed memory machines. *Parallel and Distributed Computing*, 42:91–100, 1997.
- [30] D. Koelma and H. J. Sips. A software architecture for parallel image processing. In *Proceedings of the 3rd Annual Conference of the Advanced School for Computing and Imaging (ASCI'97)*, pages 34–40, Heijen, The Netherlands, 1997.
- [31] F. J. Seinstra and D. Koelma. Transparent parallel image processing by way of a familiar sequential api. In *Proceedings of the 15th International Conference on Pattern Recognition*, pages 824–827, Barcelona, Spain, September 2000.
- [32] Leah H. Jamieson, Gregory W. Cook, Edward J. Delp, Farooq Hameed, Susanne E. Hambrusch, Jamshed N. Patel, Ashfaq A. Khokhar, and Ke Shen. Parallel scalable libraries and algorithms for computer vision. In *International Conference on Pattern Recognition*, volume III, pages 223–228, 1994.
- [33] S. M. Smith. Edge thinning used in the susan edge detector. Technical report, Defense Research Agency, Chobham Lane, Chertsey, Surrey, UK, 1995.
- [34] Ravi Ramamoorthi, Cindy Ball, and Alan H. Barr. Dynamic splines with constraints for animation. Technical report, California Institute of Technology, 1997.

Bibliography

- [35] Hong Qin and Demetri Terzopoulos. Triangular nurbs and their dynamic generalizations. *Computer-Aided Geometric Design*, 14:325–347, 1996.
- [36] Demetri Terzopoulos and Hong Qin. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, April 1994. Special Issue on Interactive Sculpting.
- [37] M. Preston and W. T. Hewitt. Animation using nurbs. *Computer Graphics Forum*, 13(4):229–241, 1994.
- [38] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 313–324, August 1996.
- [39] David R. Forsey and Richard H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions on Computer Graphics*, 14:134–161, 1995.
- [40] Bert Juttler and Michael G. Wagner. Computer-aided design with spatial rational b-spline motions. *ASME Journal of Mechanical Design*, 118:193–201, 1996.
- [41] Subodh Kumar. *Interactive Rendering of Parametric Spline Surfaces*. PhD thesis, Department of Computer Science, University of North Carolina, Chapel Hill, 1996.
- [42] Frédéric W. B. Li, Rynson W. H. Lau, and Mark Green. Interactive rendering of deforming nurbs surfaces. In *Proceedings of Eurographics '97*, volume 16, pages 47–56, 1997.
- [43] F. Kahlesz and R. Klein A. Balázs. Nurbs rendering in opensg plus. In *Proceedings of OpenSG Symposium 2002*. University of Bonn, Institute of Computer Science II, Computer Graphics, Römerstrasse 164. 53117 Bonn, Germany, January 2002.
- [44] Marc Alexa and Wolfgang Müller. Representing animations by principal components. In *EUROGRAPHICS 2000/M. Gross and F.R.A. Hopgood*, volume 19, 2000.

Every day, in every way, we are getting better and better.

-Emile Coué

- End of Thesis -