SKETCH-BASED SKELETON-DRIVEN 2D ANIMATION AND MOTION CAPTURE

JUNJUN PAN

June 2009

National Centre for Computer Animation

Media School

Bournemouth University

SKETCH-BASED SKELETON-DRIVEN 2D ANIMATION AND MOTION CAPTURE

JUNJUN PAN

A thesis submitted in partial fulfilment of the requirements of the Media School of Bournemouth University for the degree of Doctor of Philosophy

June 2009

National Centre for Computer Animation

Media School

Bournemouth University

ABSTRACT

This research is concerned with the development of a set of novel sketch-based skeleton-driven 2D animation techniques, which allow the user to produce realistic 2D character animation efficiently. The technique consists of three parts: sketch-based skeleton-driven 2D animation production, 2D motion capture and a cartoon animation filter.

For 2D animation production, the traditional way is drawing the key-frames by experienced animators manually. It is a laborious and time-consuming process. With the proposed techniques, the user only inputs one image of a character and sketches a skeleton for each subsequent key-frame. The system then deforms the character according to the sketches and produces animation automatically. To perform 2D shape deformation, a variable-length needle model is developed, which divides the deformation into two stages: skeleton driven deformation and nonlinear deformation in joint areas. This approach preserves the local geometric features and global area during animation. Compared with existing 2D shape deformation algorithms, it reduces the computation complexity while still yielding plausible deformation results.

To capture the motion of a character from exiting 2D image sequences, a 2D motion capture technique is presented. Since this technique is skeleton-driven, the motion of a 2D character is captured by tracking the joint positions. Using both geometric and visual features, this problem can be solved by optimization, which prevents self-occlusion and feature disappearance. After tracking, the motion data are retargeted to a new character using the deformation algorithm proposed in the first part. This facilitates the reuse of the characteristics of motion contained in existing moving images, making the process of cartoon generation easy for artists and novices alike.

Subsequent to the 2D animation production and motion capture, a "Cartoon Animation Filter" is implemented and applied. Following the animation principles, this filter processes two types of cartoon input: a single frame of a cartoon character and motion capture data from an image sequence. It adds anticipation and follow-through to the motion with related squash and stretch effect.

ACKNOWLEDGMENT

First and foremost, I would like to thank my supervisors, Professor Jian J. Zhang, Dr. Xiaosong Yang and Dr Reza Sahandi. Professor Zhang and Dr. Xiaosong Yang have guided me to the wonderland of computer animations. They are both wonderful and kind people. It is joyful to discuss problems with them and the sparks of inspirations often come from such discussions. I shall never forget the abundant time and efforts Professor Zhang has put into my thesis and the directions from Dr. Xiaosong Yang in research. Dr Reza Sahandi has given me a lot of valuable comments on my thesis writing and helped me finalize the draft. I would not have finished the writing so quickly without their help.

I would like to thank Claudia Moore and her animation team for testing our prototype system and providing us with helpful suggestions. My sincere appreciation goes to Mrs. Jan Lewis and Mr. Dan Cox for their daily support and kind helps in non-academic matters.

My study and research here is financially supported by the bursary of Bournemouth University and the Overseas Research Students Award Scheme (ORSAS). I also would like to take this opportunity to acknowledge their support.

Finally, I dedicate this thesis to my family, for their love and encouragement on my journey of growth.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background

Character modelling and animation involves everlasting efforts and extensive research over many decades since the advent of the first computerized human models in the 1970s [Isaac 2003]. At present, their applications have spread to a great variety of fields, including media, biomedicine, military, education and entertainment. Human-like virtual characters which congest in computer games (e.g. “Warcraft”) and films (e.g. “Avatar”, “King Kong”, “Shrek”, etc) are playing a remarkable role in modern public media and entertainment.

Generally, character modelling and animation are regarded as a challenging and labour-intensive work that requires both professional skills and artistic expertise. To build and animate a vivid virtual character, three stages are usually needed in this process: modelling, motion and deformation. These animation production steps require extensive expertise (mesh modelling, dynamics, kinematics, physical simulation, etc), professional software (Maya, 3DS MAX, etc), special equipments (3D laser scanner, motion capture system, etc) and proficient computer skills [Fox 2004]. Therefore, ordinary users are hardly given the opportunity to produce convincing animation due to lack of appropriate skills, knowledge and resources.

In addition, for animators, animation production is a big systemic project which involves collaborations among various specialists and production teams [Fox 2004, Isaac 2003]. For example, a typical CG animation pipeline generally contains three stages: preproduction (scriptwriting, storyboarding, character design, etc), production (modelling, rigging, animation, scene layout, rendering, etc), and postproduction (compositing, video editing, etc). In the preproduction process, 2D artists in the creative team usually design character appearances and storyboards through freehand sketching. Next, these drawings and storyboards are implemented by 2D/3D modelers and animators in the production team through modelling, rigging, animation, and rendering. In the last stage, these semi-finished clips are sent to the technicians for the final video editing and compositing. Therefore, animation production is usually a big and time-consuming job, which is too complicated and overwhelming for a single artist or animator to undertake.

Sketch-based animation, as a useful tool for character modelling and animation, provides a good solution to animators and ordinary users to avoid the difficulties discussed above due to its intuitiveness and simplicity. Many papers [Davis et al. 2003, Thorne et al. 2004, Li 2006, Yuki et al. 2007] have been published and several techniques have been developed into commercial software, e.g. [Igarashi et al. 1999]. With the help of sketch-based techniques, animators can translate their 2D drawings into 3D models. Instead of handling the details step by step, the modeller/animator can visualize and evaluate the fast-prototyped models at an early stage, which can be further refined with other 3D tools to meet the artistic needs.

Nevertheless, compared with above progress in 3D animation, 2D animation (also can be called cartoon or cel animation) has not benefited as much from these achievement. Up until today, most professional cartoon studios still produce huge amounts of animation (key-frames and in-betweens) manually [Daniel 2006]. Largely overlooked by the computer graphics community, the 2D animation process remains laborious, time-consuming and predominately manual, in contrast with the contemporary 3D animation production process.

1.2 Research Objectives

This research is aimed at investigating and designing an efficient and easy-to-use technique for quick articulated cartoon production. And a functional system is expected to be developed to cope with the practical requirements in 2D animation. The research objectives are in the following aspects:

- Investigating and evaluating the traditional cartoon production process. Designing and developing a practical and easy-to-use sketch-based technique to generate cartoon animation.
- Investigating the existing 2D deformation and animation techniques for cartoon character. Developing a new real-time 2D shape deformation algorithm to be used with the sketch-based cartoon strategy.
- Designing and developing a useful motion capture and retargeting technique for 2D character animation.
- Developing and implementing an animation system with sketching interface and auxiliary functions for cartoon production.
- Implementing a "Cartoon Animation Filter" and applying it to the final cartoon output from the developed system.

1.3 Framework and Overview

The generation of key-frames and in-between frames are two of the most important and labour intensive steps in 2D animation production. For efficient use of the animators' time, the key-frames are drawn by skilful *key-framers*, while the in-between frames are created by those who are less experienced and skilful, known as the *in-betweeners*. Although some software tools, e.g. Animo, Toon Boom [2008], have been helpful in generating in-between frames, they often lack 'personality' in comparison with those created by an animator. Often, the in-betweens generated by software have to be tweaked by the animator to add 'personality' to the animation. In practice, many in-betweens remain created manually. On the same point, if the animator were able to produce a key-frame more easily and efficiently, then he/she could produce more densely spaced key-frames resulting in a much smaller number of in-betweens need to be generated. Even if the in-betweens were generated by software, it is reasonable to argue that they are unlikely to need further tweaking, because the number of in-betweens is so small and would have little effect on the quality of animation. Our techniques presented in this thesis are based on this argument.

Motivated by the skeleton-driven 3D animation techniques, sketch-based animation techniques, and recent progress in 2D deformations, e.g. [Igarashi 2005], in this thesis a novel technique is presented with the aims to improve the degree of automation for the production of 2D animation without sacrificing the quality. Our technique consists of three parts, Part 1: 2D animation sequence generation by sketch, Part 2: motion capture and retargeting and Part 3: application of "cartoon animation filter". Part 1 can be used independently to create an animation sequence. If it is combined with Part 2, one can easily reuse the 'motion' of an existing animation sequence and apply it to a different character. Part 3 is used to add exaggerated effects to the final result.

The most important issue concerned is how to handle the complex shape deformation of characters. In Part 1, for a character at a given orientation (for example, side view, front view or back view), the user first generate its skeleton by analyzing the geometry of the boundary curve. Similar to a 3D character, the skeleton acts as the driving structure and controls the deformation of the character. To deform a character, a *variable-length needle model* is introduced and an algorithm called *skeleton driven + nonlinear least squares optimization* is proposed. The idea is to divide the 2D shape deformation into two components. The first is skeleton driven deformation, which is controlled purely by the corresponding segment of the character skeleton; and the other is nonlinear least squares optimization, which is to compute the deformation in the joint areas which are associated with the skeletal joints. Our observation suggests most complex deformation occurs around the joint areas of a character during animation. For the interest of computational efficiency, the skeleton driven deformation is treated simply as a linear transformation. Only the deformation in the joint areas is solved by nonlinear least squares optimization. To ensure realistic deformation, properties such as boundary feature preservation, interior smoothness, the global area preservation are maximized during animation. The property of area preservation is also easily incorporated into the *variable-length needle model*. With our technique, once the first frame is given, the animator can easily create an animation sequence by drawing the skeleton for each subsequent key-frame. The system will produce the deformed character shape automatically, saving the animator from drawing the whole frame. Although the primary application of our technique is 2D animation production, it is also applicable to interactive graphical systems where the user can deform a 2D shape directly by moving /deforming its skeleton. Since it is very simple to use, it is anticipated that this approach is not only of interest to professional cartoon production studios, but also to novice 2D animators and artists for creating 2D moving graphics.

Motion capture and retargeting is another big issue in character animation. At present, most of the research work and academic papers in this field focuses on 3D animation. Although large amounts of video, cartoon and traditional 2D moving images exist, few effective approaches are available to make use of these abundant resources due to the special characteristics and principles of 2D animation [Williams 2001, Isaac 2003]. The main objective of Part 2 is to patch this obvious gap. Because our cartoon production technique is skeleton-based, the idea of Motion Capture from 3D animation is naturally borrowed to capture the 'motion' of a 2D animation sequence, and retarget the captured motion to a different character. However, instead of capturing the motion of a performer with a 3D motion capture device, the motion of a 2D character is captured from an image/cartoon/video sequence. Another obvious difference it has from the 3D case is that the skeleton length of a 3D performer is constant during animation, i.e. the skeleton segments are in essence rigid. To capture the motion of a performer, the 3D Motion Capture Officer embeds (fits) an animation skeleton into the 3D marker points during the calibration process, which helps the system compute the rotational angle of each skeletal segment relative to its parent (Autodesk Motion Builder [2008]). In 2D animation, changing length feature in the form of stretching and compression is one of the most powerful and expressive principles of animation [Williams 2001]. Fortunately, with our method, the 2D skeleton can be used to represent these important and expressive transformations.

Retargeting the captured motion to a different character has been extensively studied in 3D animation, e.g. [Gleicher 1998], and can be applied to 2D animation with our method. Here a feature region based tracking method is presented, commonly used in computer vision, to extract the motion of 2D objects in video or an image sequence. A mixed optimization coupled with template matching and Kalman prediction is applied. After the user has located all the joint regions of a character in the first frame, the system will track the motion of the joints automatically in the subsequent frames. The captured motion information is then retargeted to the predefined skeleton of a new 2D character to generate the deformation (animation). To avoid the problem of self-occlusion and feature disappearance during the tracking process, both visual and geometric features are taken into account.

In a sense, cartoon animation is the art of manipulating motion to emphasize the primary actions of character. Experienced animators can guide the viewers' perceptions of the motion and literally bring it to life. Isaac Kerlow [Isaac 2003] introduced twelve principles of animation which were created in the early 1930s by animators in Walt Disney Studios. These principles were used to guide cartoon design and animation production as well to train the art students. These twelve principles became one of the foundations of hand-drawn cartoon character animation, which are mostly about five things: directing the performance, acting the performance, representing reality (through modelling, drawing, and rendering), interpreting real world physics, and editing a sequence of actions [Williams 2001]. Some original principles, such as anticipation and follow-through to the motion with related squash and stretch, are still useful today because they help us to create more "animated" characters and motions. In 2006, Jue Wang et al. [2006] present a technique named "Cartoon Animation Filter". It is a simple filter that takes an arbitrary motion signal as input and modulates it in such a way that the output motion is more alive. The filter only needs the user to set the desired filtering strength as the parameters. It can cope with three different types of motion input including: hand drawn motion, video objects and 3D MoCap data. The superiority of this technique lies in its simplicity and generality, which provides a good reference for our work. So in the third part of our work, a modified "Cartoon Animation Filter" was implemented and applied to the cartoon output from the first or the second part.

1.4 Contributions

There are five key contributions in this thesis:

1. A sketch-based skeleton-driven 2D animation technique is presented. To produce new frames, the user only needs to input one image of the character and sketch skeletons for subsequent key-frames.
2. To handle 2D shape deformation, a *variable-length needle model* is developed and the *skeleton driven + nonlinear least squares optimization* algorithm is introduced. Compared with other approaches for a similar purpose, it is more efficient and able to produce plausible deformation results.
3. A novel skeleton-based 2D motion capture technique is presented. It can extract the motion from cartoon, video and rendered moving image sequences by tracking the motion of the skeleton joints. Using both geometric and visual features, it prevents self-occlusion and feature disappearance in moving images.
4. A modified "Cartoon Animation Filter" is implemented and applied to the two types of cartoon output from the developed system. It can add exaggerated effect, such as squash and stretch, to the final result and make the animation more vivid.
5. A prototype animation system is developed to implement the described techniques above. This system is tested and evaluated by several cartoon examples in experiments. Coupled with assistant functions, it can deal with the most practical requirements for character cartoon production.

During the period of the research, several related publications have produced to report the technical developments achieved, which are listed as follows:

- Pan, J., Yang, X., Xie, X., Willis, P., and Zhang, J. Automatic Rigging for Animation Characters with 3D Silhouette. *Computer Animation and Virtual Worlds*, 20(2-3): 121-131, Jun. 2009.
- Pan, J., Zhang, J., Zhang, Y., and Zhou, H. X-ray-Based Craniofacial Visualization and Surgery Simulation. In book: *Recent Advances in the 3D Physiological Human*. Springer-Verlag (LNCS), August 2009, pp.208 -224.
- Chang, J., Pan, J., and Zhang, J. Modeling Rod-like Flexible Biological Tissues for Medical Training. In book: *Modelling the Physiological Human*. Springer-Verlag (LNCS), Jan 2010, pp.51-61.
- Wang, M., Chang, J., Pan, J., Jian J. Zhang. Image-based bas-relief generation with gradient operation. Proceedings of the 11th IASTED International Conference Computer Graphics and Imaging, February, 2010, Austria.
- Pan, J. and Zhang, J. Sketch-based skeleton-driven 2D animation and motion capture. (submitted to Visual Computer)

1.5 Thesis Structure

The structure of the thesis is outlined in Figure 1.1.

Chapter 1 is a brief introduction of this thesis. Chapter 2 reviews and evaluates the traditional cartoon production process first, and points out places for improvements. Then the existing techniques in 2D shape deformation and 2D motion capture are investigated. In the end, the "Cartoon Animation Filter" technique is discussed. Chapter 3 introduces the sketch-based skeleton-driven 2D animation technique. A *variable-length needle model* and the *skeleton driven + nonlinear least squares optimization* algorithm for 2D shape deformation are particularly described. Chapter 4 introduces the skeleton-based 2D motion capture and retargeting technique. The core content is a joint tracking algorithm based on computer vision. Chapter 5 describes the implementation of the "Cartoon Animation Filter" and its application in the final output exaggeration. Chapter 6 introduces the structure and interface of this prototype 2D animation system. The animation system is also tested and evaluated with several practical cartoon examples in experiments. Chapter 7 concludes the thesis and identifies potential directions in future research and development.

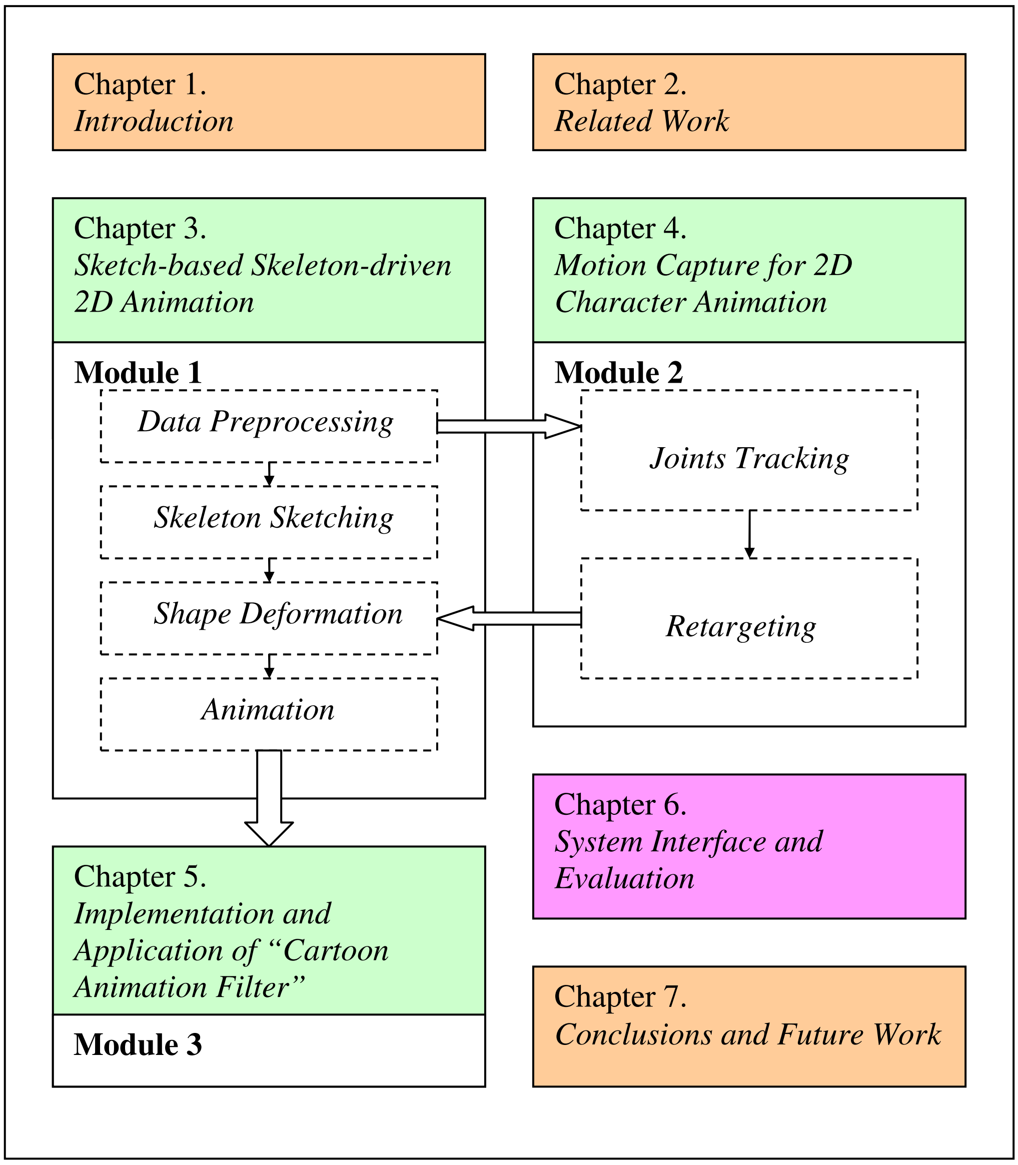


Figure 1.1 Structure of the thesis

CHAPTER 2

RELATED WORK

Research on “Sketch-based Skeleton-driven 2D Animation and Motion Capture” mainly involves two aspects: 2D animation generation and motion capture. There is a significant body of previous work concerned with these two research areas. First, the traditional cartoon production process is discussed. Then the most relevant developments are reviewed.

2.1 Traditional Cartoon Production

Traditional animation, which can also be called cel animation or hand-drawn animation, was first used for the animated films in the 20th century. The individual frame of a traditional animated film is photograph of drawings which firstly drawn by animators on a paper. To create the performance of movement, each drawing differs slightly from the previous one. The animators' drawings are traced or photocopied onto a transparent acetate sheet which is called cels. The cels are filled in with assigned colours or tones on the side opposite the line drawings. The completed character cels are photographed one by one into motion picture film in front of a painted background by a rostrum camera.

The traditional cel animation production became obsolete at the beginning of the 21st century. Nowadays, animators' drawings and the backgrounds are scanned into or drawn online directly on a computer. Various computer graphics softwares (such as “Toon boom”, “Animo”) are used to colour the drawings and simulate camera movement and effects. The final animated image sequence is output to several types of media, including traditional film and newer media such as digital video. The visual feeling of traditional cel animation is still preserved, and the style of animators' work has remained essentially the same as before. Some animation production studios used the term "tradigital" to describe the cel animation which makes extensive use of the computer graphics technology.

According to the survey by Robertson [1994] and Sykora [2006], there are two kinds of commercial computer graphics core systems: *Ink and Paint* and *Automated In-Betweening* are used to produce 2D animation in professional cartoon studios. The pipeline structure of these two systems is illustrated in Figure 2.1 (a) and (b) [Fekete et al. 1996]. The automatic work done by computers is marked with a dark background. In both of them, most steps have an *exposure sheet*, which lists all the frames in a scene. Each line includes the phoneme pronounced by characters and the position where the camera will shoot the cartoon figures. Each scene also has a set of stages, of which only the background can be painted in parallel with the animation stages. These stages include:

Story Board: Splits script into scenes with music and dialogue.

Sound Track: Records music and dialogue in prototype form.

Sound Detection: Fills the dialogue column of an exposure sheet.

Layout: Manages backgrounds and main character positions, with specifications for camera movement 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, and provides instructions for the in-betweeners.

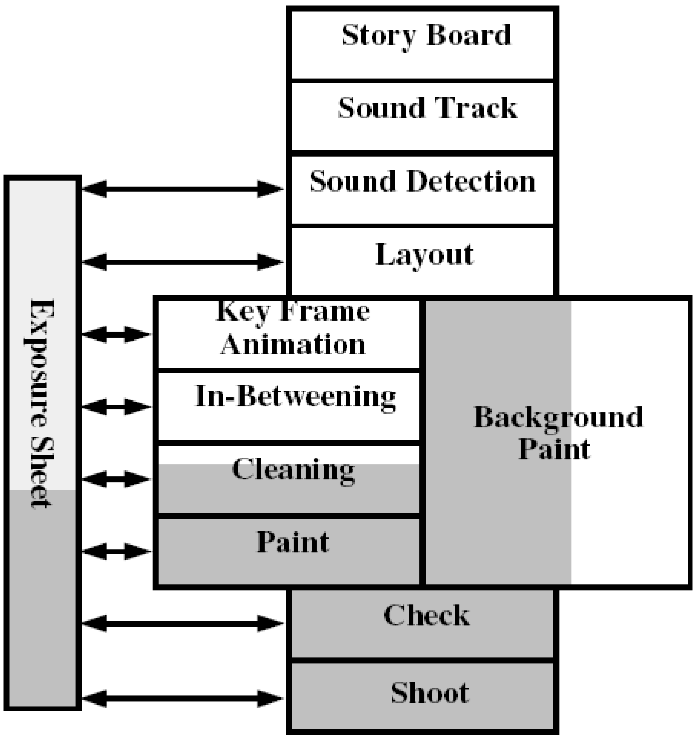
In-Betweening: Draws the missing frames according to the key-frame animator's instructions.

Cleaning: Cleans up the drawings to achieve final quality of the strokes.

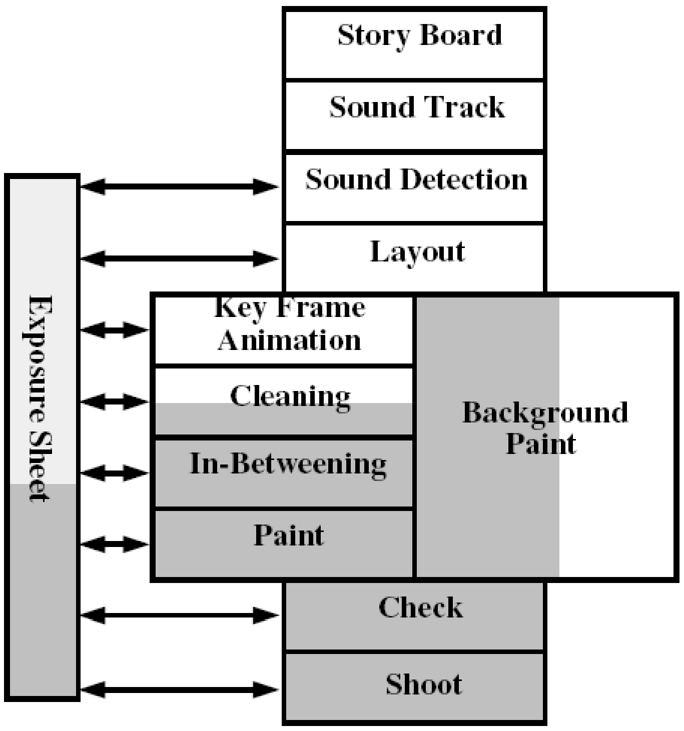
Paint: Photocopies the clean drawings onto acetate celluloid (cels) and paints zones with color.

Check: Verifies animation and backgrounds according to the layout and approves for shooting.

Record: Shoots frame-by-frame on video or film, using a rostrum.



(a)



(b)

Figure 2.1 Pipeline structure of two different systems in cartoon production [Fekete et al. 1996]. (a) Ink and Paint, (b) Automated In-Betweening

For both of *Ink and Paint* system and *Automated In-Betweening* system, all important artwork (key-fame animation) is still drawn manually, which usually needs 50~300 animators to draw thousands of characters in different poses with human hands, and later digitized and managed by the computer after the cleaning stage. Therefore, to significantly reduce the human labour input in the traditional cartoon making process, a novel sketch-based skeleton-driven cartoon technique is proposed in this thesis. In this technique, once a template image of a character is given, the animator can easily create an animation sequence by just drawing the skeleton for each subsequent key-frame. This saves the animator from drawing the whole frame.

2.2 Recent Progress in 2D Animation

Generally, 2D animation can be divided into two classes: one is the traditional 2D character animation. (It is also called: cartoon or cel animation) The other is direct manipulation on 2D image, such as physical model based simulation, which can be used to generate natural phenomena animation. The technique in our thesis belongs to the first category, but some techniques in the second category are also used. There is a significant body of previous work concerned with 2D animation. Here only the most relevant developments to ours are discussed.

Hsu et al. [1994] developed a system to create 2D drawing with "skeletal strokes". Its expressiveness as a general brush and efficiency for interactive use make it suitable as a basic drawing primitive in 2D modelling. Fekete et al. [1996] developed a system (TicTac Toon) for professional 2D animation studios that replaces the traditional paper-based production process. It was the first animation system to use vector-based paintings and graph. Kort [2002] presented an algorithm for computer aided in-betweening. The algorithm is layer-based which classify the drawing into strokes, chains of strokes and relations among them. Some constraints are introduced to match the different parts between drawings. He also designed a cost function to determine the correct matching. Agarwala et al. [2004] presented an approach to track the character contours in video sequence. The user first locates curves in two or more frames. Then these curves are used as key-frames by a computer vision based tracking algorithm. This method combines computer vision with user interaction and solves a space time optimization problem for time-varying curve shapes and user-specified constraints. From the developed system, a video can be transformed to nonphotorealistic animation automatically. However, one disadvantage of this approach is that it cannot cope with the occlusion or disappearance of contours. Chuang et al. [2005] presented a method to automatically synthesize the "stochastic motion texture" in still 2D images. It simulated natural phenomena relying on physical models. With this approach, they can animate the images containing passive elements with the motion driven by wind, like water, trees, and boats. Xu et al. [2007] inferred motion cycle of animals from snapshots with different individuals which captured in a still picture. Through searching the motion path in the graph connecting motion snapshots, they can infer the order of motion snapshots and construct the motion cycle. Then they animated a still picture of a moving animal group, such as birds and fish, by morphing among the ordered snapshots.

2.3 2D Shape Deformation

For the sketch-based cartoon generation, the most important problem concerned is how to manipulate complex shape deformation of objects with free hand drawings. One popular approach is the FFD (Free-Form Deformations) presented by [MacCracken and Joy 1996]. In this method, the user embeds an object into a lattice that can be deformed by a few control points. Then the shape of this object can be manipulated by moving these associated points. The disadvantage of this approach is that it needs FFD domains setting (building the lattice and setting deformation parameters), which is a tedious process, and the user must manipulate the control vertices laboriously to deform the object. Another popular method is using a predefined skeleton or control points [Lewis et al. 2000]. The user controls the skeleton/points and the shape of character can be adjusted according to the related skeleton/points. In our system, the skeleton is chosen as the sketch input because of its simplicity and intuitiveness for animators (The comments from animators can be seen in Section 6.4.3).

Most 2D deformation techniques researched are *control point* based. Although skeletons are incorporated into some commercial packages, the purpose is primarily to assist posing a character, not to deform or animate a character [Toon Boom 2008]. Igarashi et al. [2005] designed an “as-rigid-as-possible” animation system which allows the user to deform the shape of a 2D character by manipulating some control points. In this system, the shape is constructed by a triangular mesh and the user can alter the positions of some vertices by moving the control points. Through minimizing the distortion of each triangle, the positions of the remaining free vertices are computed. This is a non-linear problem and is typically expensive to solve. To reduce the computation cost, the authors presented a two step deformation algorithm, which simplifies it into two linear least-squares minimization problems. As it only approximates the original problem, it can cause implausible results in some cases due to its linear feature. Weng et al. [2006] presented a 2D shape deformation algorithm based on nonlinear least squares optimization. It used a non-quadratic energy function to represent this problem, which achieves more plausible deformation results. However, the iterative solution is computationally more costly than that by Igarashi et al. [2005]. For both methods, the user needed to define many control points on the object if the deformation is complex or for complex characters, which represents a disadvantage for cartoon production. Schaefer et al. [2006] proposed a 2D shape deformation algorithm based on linear moving least squares. To minimize the amount of local scaling and shear, it restricts the transformations of similarity and rigid-body with moving least squares, which can be derived from the closed-form formulation. It avoids the input image triangulation and performs globally smooth deformation. Later, they also extended this point-based deformation method to line segments. However, as the authors have admitted, this method deforms the entire image with no regard to the topology of the object. This weakness limits its use in 2D character animation. Wang et al. [2008] presented another 2D deformation technique based on the idea of rigid square matching. Instead of using triangular meshes, they use uniform quadrangular meshes as the control meshes. On the downside, because rotation transformation is the main deformation mechanism, it is difficult to preserve the area of the character during deformation. In addition, the obtained deformation is quite rigid, not a perfect fit for soft objects and characters.

All the methods above employ global optimization. One disadvantage of global optimization is that the shape of all triangles needs recomputing even if a small posture change happens. This is computationally expensive and is not necessary in many cases. In our algorithm, the shape deformation is divided into two components: skeleton driven deformation and nonlinear deformation of joint areas. The former can be treated as a linear transformation and the latter is solved by nonlinear least squares optimization, but only for local regions. This local optimization scheme reduces the computation costs and can still achieve plausible deformation results.

2.4 2D Motion Capture and Retargeting

Most research on motion capture and motion retargeting focuses on 3D animation

[Gleicher 1998; Favreau et al. 2004; Sand et al. 2003]. Many effective algorithms have been developed and benefited applications including computer games and film special effects. In contrast, little has been done for 2D animation. Bregler et al. [2002] presented a method to capture and retarget the non-rigid shape changes of a cartoon character using a combination of affine transformation and key-shape interpolation. Their work is in principle similar to that of extracting and mapping feature parameters (e.g. motion and deformation) [Pighin 1998]. It is effective in representing the qualitative characteristics (i.e. motion in this case). But it is difficult to be precise. Therefore, although it can be useful for cartoon retargeting, it is not easy for the animator to control the movement and deformation accurately. In contrast, a skeleton-driven approach gives the animator better control of the deformation during animation. Hornung et al. [2007] presents a method to animate photos of 2D characters using 3D motion capture data. Given a single image of a character, they retarget the motion of a 3D skeleton to the character's 2D shape in image space. To generate realistic movement, they use the "as-rigid-as-possible" deformation [Igarashi et al. 2005] and take projective shape distortion into account. In comparison, our method directly transfers the 2D motion data from an existing image sequence. It does not need the user manually specifying correspondences between 2D pose and 3D pose of character.

2D animation can be regarded as a consistent image sequence. Our approach, which is influenced by several video based approaches [Shi and Tomasi 1994, Cai and Aggarwal 1996, Chen et al. 2005, Aggarwal and Triggs 2006, Michoud et al. 2007], tracks the motion of the character's joints. Shi et al. [1994] present the famous KLT tracking algorithm in computer vision. Its basic principle is that a good feature is one that can be tracked well, so tracking should not be separated from feature extraction. If a feature is lost in a subsequent frame, the user can optionally ask the procedure to find another one to keep the number of features constant. So the algorithm divides the tracking into two stages: good feature extraction and feature matching frame-to-frame. Compared with KLT tracker, without good feature selection, our algorithm directly tracks the interested feature region (joints) which is located interactively for each frame. Cai and Aggarwal [1996] used Multivariate Gaussian model to search the most likely matches of human subjects between consecutive frames in multiple fixed cameras. Bregler [1997] presented a vision based motion capture technique to recover articulated human body configurations without markers in complex video sequences. They used an integration of exponential maps and twist motions to estimate the differential motion, which can track complex human motions with high accuracy. Chen et al. [2005] presented a markerless motion capture technique to extract motion parameters of a human figure from a single video stream. It used the silhouette as image features and model-based method to optimize the searching in pose space. With physical constraints and knowledge of the anatomy, a viable pose sequence can be reconstructed for many live-action cases. Aggarwal and Triggs [2006] described a learning based method for recovering 3D human pose from monocular image sequences. Using a mixture of regression method to return multiple solutions for each detected silhouette of character, the resulting system can track long sequences stably, and reconstruct 3D human pose from single images in ambiguous cases.

Most of the approaches above are model-based, which focus on the motion capture of human body. However, since our system needs dealing with a variety of characters with different shape and topology, such as humans, animals, plants, monsters and other articulated objects with regular topology structure, the model-based tracking methods are ineffective and cannot be used here directly. In our techniques, to extract the motion of a character, more general features: colour and geometry information (position, velocity) of the joints are selected.

2.5 "The Cartoon Animation Filter"

How to add cartoon principles, such as anticipation and follow-through to the motion with related squash and stretch of geometry, to the output animation automatically is another important research topic in computer animation. Isaac Kerlow [2003] introduced twelve principles of animation which were created in the early 1930s by animators in the Walt Disney Studios. They are mainly about five things: representing reality (through drawing, modelling, and rendering), acting the performance, directing the performance, editing a sequence of actions, and interpreting real world physics. These twelve principles became the foundations of hand-drawn cartoon character animation. Wang et al. [2006] presented the technique of “Cartoon Animation Filter”, a simple filter that inputs motion signal, and outputs the result more “animated” (vivid). In this filter, nearly all parameters are set up automatically. The user only needs to input the filtering strength and the output can be hand drawn motion, video objects and Mocap data. The advantage of this animation filter lies in its simplicity and generality, which set a good example for our research. For completeness, in this thesis, the function of this filter is implemented in the prototype system.

2.6 Summary

In this chapter, the traditional cartoon production process was firstly introduced. Then the typical techniques for 2D shape deformation and motion capture, especially those for cartoon animation purposes, were presented. A greater attention was paid to the control point based 2D shape deformation methods and video based tracking approaches for motion capture. In the end, principles for cartoon exaggerated effects and a “Cartoon Animation Filter” technique have been reviewed.

In the following chapters, the key original contributions work will be discussed. In Chapter 3, the sketch-based skeleton-driven 2D animation technique will be described. It includes a new algorithm for 2D shape deformation. In Chapter 4, the skeleton-based 2D motion capture and retargeting technique will be introduced. The core content is a computer vision based joint tracking method. Chapter 5 is the implementation of modified “Cartoon Animation Filter” and its application in our developed animation system.

CHAPTER 3

SKETCH-BASED SKELETON-DRIVEN 2D ANIMATION

In traditional cartoon production, animators need to draw the whole picture of the character manually for each key-frame, which is the most laborious and time-consuming work in animation making process. In this chapter, the sketch-based skeleton-driven 2D animation technique is discussed. With this technique, animators or ordinary users can produce fast cartoon animation by only sketching the skeleton of a 2D character.

3.1 Overview

Our technique consists of five steps. Given an input character image, known as the original template model, the algorithm proceeds as follows:

1. The silhouette of the original template model is detected with the marching squares algorithm [Lorensen and Cline 1987].
2. The curve skeleton (medial axis) of the model is identified using the Hilditch's thinning algorithm [Cornea, et al. 2006].

3. The animation skeleton guided by the extracted curve skeleton is generated. This process is called the skeletonization.
4. The mesh vertices are decomposed into regions, each of which is associated with a segment of the animation skeleton using a Euclidean metric. This process is called decomposition or skinning.
5. After the animation skeleton has been constructed, area and shape preserving deformation is achieved by using a combination of the variable length needle model and a non-linear optimization based on rotation and scale invariant (RSI) Laplacian coordinates, mean value coordinates and edge lengths. The character model can be animated/ deformed by sketching an animation skeleton for each key-frame using our prototype system.

In the following, the technical detail is discussed and a popular cartoon figure: mm (Figure 3.1 (a)) is used to illustrate the result.

3.2 Silhouette Detection and Triangulation

As the input to our prototype cartoon generation system, the user first imports a 2D character serving as the *original template model*, which can be represented by BMP/JPEG or vector graphics format. The requirement is that the boundary of the object should be represented by a closed polygon. For BMP/JPEG images, the background is currently removed manually. Its silhouette is detected with the marching squares algorithm detailed below, forming a closed polygon (Figure 3.1 (b)). Then the discrete uniform sampling is conducted inside the silhouette (Figure 3.1 (c)) to make the distance between neighbour vertices almost equal.

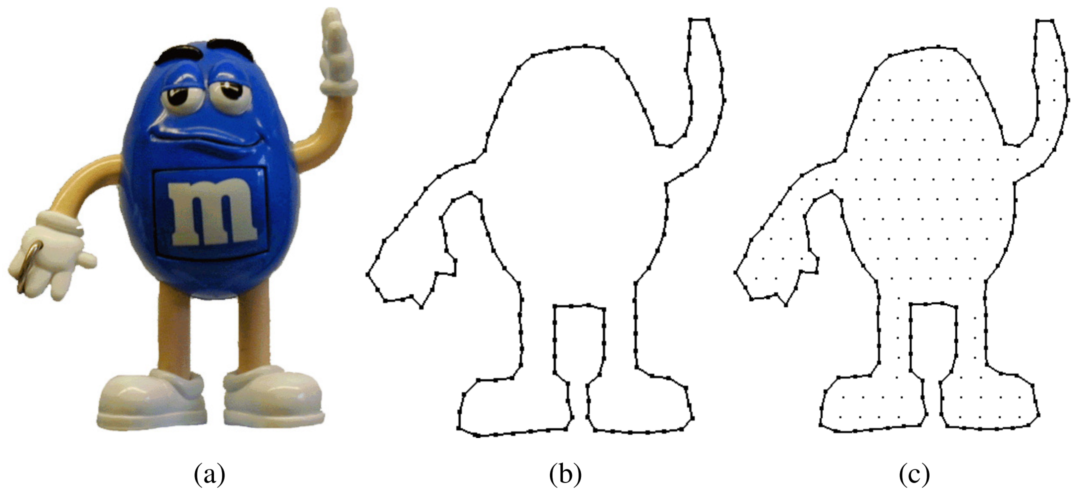


Figure 3.1 Experimental result of silhouette detection for a cartoon character. (a) Original image, (b) Vector graphic for silhouette, (c) Vertexes after discrete uniform sampling

The marching squares algorithm for silhouette tracing is similar to the marching cubes algorithm [Lorensen and Cline 1987]. It can be described as follows. In the binary segmented image, assuming that the object consists of a set of black pixels and the background is white, the algorithm examines four adjacent pixels each time in a 2 x 2 grid. The search is anticlockwise. For any given position in the image, the 2 x 2 grid is in one of 16 possible cases shown in Figure 3.2. For example, if the grid is in the object, all pixels are black. If it is in the background, they would all be white. For positions on the edge of the object, the 2 x 2 grid belongs to other 14 cases which can be used to determine which position in the image is examined next as indicated by the arrow. The process is repeated until the test grid reaches its starting position. Then the silhouette is traced and the loop terminates.

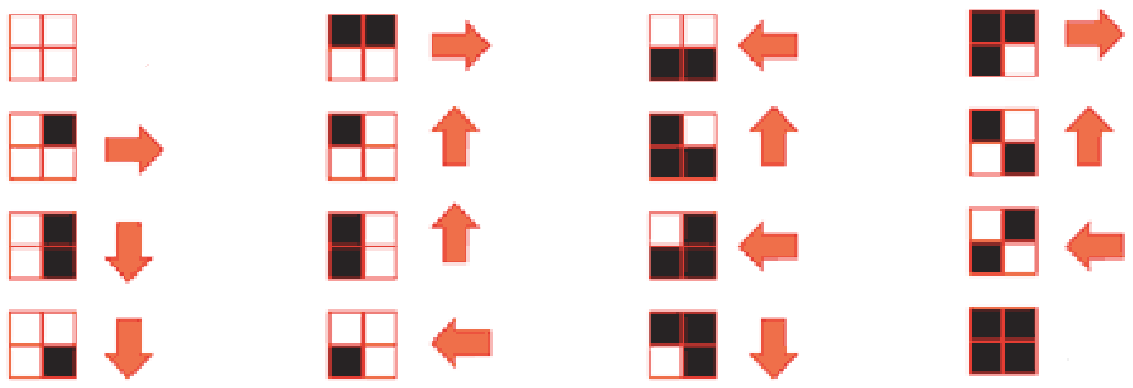


Figure 3.2 16 possible cases and direction of the moving 2 x 2 grid on the next iteration

Distributing discrete points allows the polygon to be triangulated. Many triangulation algorithms exist. Igarashi et al. [2005] used a particle based algorithm which performed better manipulation results by using near-equilateral triangles with similar sizes across the region. In our system, a simple but effective method is designed to achieve the same result. Starting with the standard Delaunay Triangulation [Shewchuk 2002] to connect all the vertices (including interior vertices and boundary vertices) (Figure 3.3 (a)), all the external triangles outside the boundary are classify and deleted to generate the final triangle meshes. This process achieves the same experimental result of the Constrained Delaunay Triangulation, which is shown in Figure 3.3 (b). The sampling density is adjustable at the user's will to form sparser or denser meshes depending on the requirements. To make sure a character shape is properly triangulated, the template model should be expanded or the limb occlusion is solved beforehand using image completion techniques [Drori et al. 2003, Sun et al. 2004].

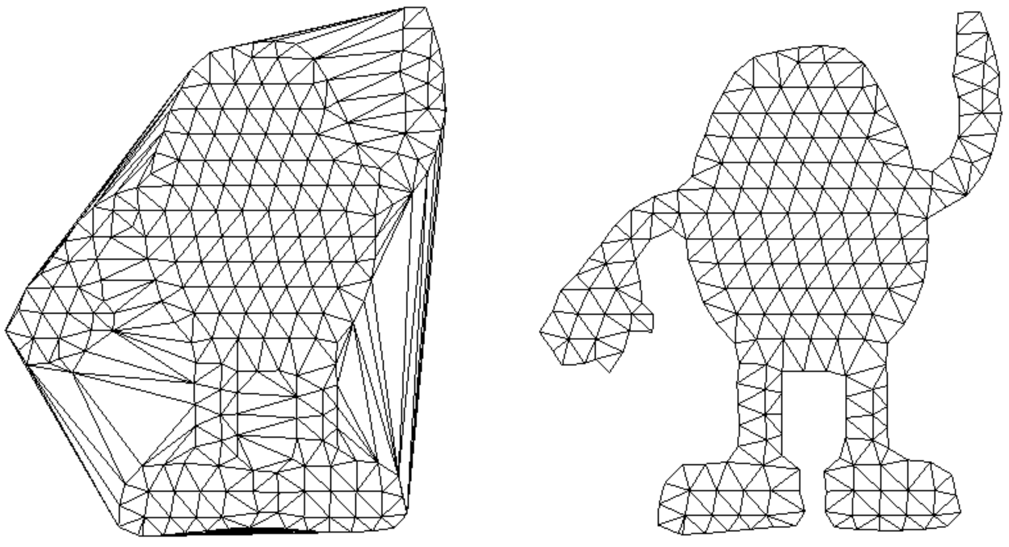


Figure 3.3 Process of our designed Constrained Delaunay Triangulation. (a) Standard Delaunay Triangulation, (b) Final result after deleting the external triangles

3.3 Skeletonization and Decomposition

After the silhouette detection and triangulation, to guide the animation skeleton production, the curve skeleton (medial axis) of the model is first identified using a 2D thinning algorithm. This process is called *skeletonization.* In our system, the *curve skeleton* is generated by the Hilditch's thinning algorithm [Cornea, et al. 2006].

Hilditch's thinning is an algorithm that can be used for a binary image. It can produce clean curve skeleton without unexpected needles or branches. It can be described as the following operations, which are based on the configurations of the neighbourhood for each pixel.

For a 3×3 pixel region which is shown in Figure 3.4, if $P_1 = 1$ (the centre pixel is black) and it is 8-neighbourhood of pixel P_1 , the following four conditions are defined:

1. $2 \le B(P_1) \le 6$

2. $A(P_1) = 1$
3. $P_2 \cdot P_4 \cdot P_8 = 0$ or $A(P_2) = 1$
4. $P_2 \cdot P_4 \cdot P_6 = 0$ or $A(P_4) = 1$

Where $A(P_1)$ is the number of 0,1 patterns in the sequence $P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_0, P_2$. $B(P_1)$ is the number of non-zero neighbours of P_1. If all the four conditions above are satisfied, the pixel P_1 will be removed (change black to white, $P_1 = 0$). This iteration is repeated until no more pixels can be removed in the binary image.

P_3	P_2	P_0
P_4	P_1	P_8
P_5	P_6	P_7

Figure 3.4 3×3 pixel region in a binary image

To produce an *animation skeleton*, the user locates the joints either on the curve skeleton or the mesh vertices. An example of the curve skeleton is shown in Figure 3.5 (a). Some end points of the curve skeleton branches (red points in Figure 3.5 (a)) can be used as skeletal joints directly. After skeletonization, every vertex is attached to its nearest skeleton segment. This is called the *decomposition* or *skinning*, which classifies the vertices into different *parts*. Our algorithm is similar to the KNN (K-Nearest-Neighbour) classification [Dasarathy 1991]. One important difference is that in our method when computing the minimum distance between vertices and skeleton segments, it should not be intersected with silhouette. This can be expressed as:

$$l(v_i) = \min\{\underset{\text{non-intersected}}{distance}(v_i, segment_j)\}, (j \in [1, k]) \qquad (3.1)$$

where $\underset{\text{non-intersected}}{distance}(v_i, segment_j)$ denotes the Euclidean distance between vertex v_i and the skeleton segment: $segment_j$, and it should not be intersected with silhouette of object; ***k*** represents the total number of skeleton segments. $l(v_i)$ gives the minimum among all the Euclidean distances between vertex v_i and the skeleton segments: $segment_j$ $j \in [1, k]$. Suppose $l(v_i)$ is the distance between vertex v_i and skeleton segment: $segment_m$, vertex v_i will be attached to $segment_m$ and its new coordinates will be determined by the position of this skeleton segment during animation. The decomposition result for the example cartoon character is shown in Figure 3.5 (b). In this figure, there are 16 skeleton segments, which have been colour-coded to represent the associated *vertex regions*. As our method is based on the shortest distance classification, the decomposition algorithm may occasionally classify some vertices in a wrong region. The user can refine the final decomposition result through interactive adjustment (Section 6.3).

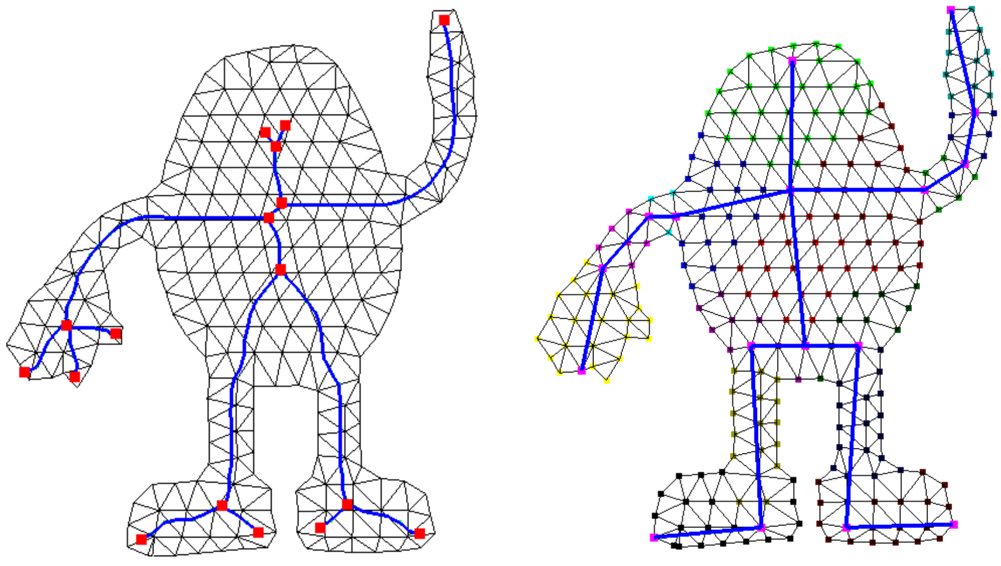

(a) (b)

Figure 3.5 Result of skeletonization and decomposition. (a) Curve skeleton, (b) Skeleton and decomposition

Based on the classification of all the vertices, triangles can now be classified into two types, *interior triangles* and *joint triangles*. If the three vertices of a triangle are of the same colour, i.e. they are all associated with one skeleton segment, the triangle is an interior triangle. Otherwise the triangle is a joint triangle. Both types of triangles are shown in Figure 3.6. Meanwhile, the vertices can also be sorted into three categories, *silhouette vertices, interior vertices* and *joint vertices* illustrated in Figure 3.6. Silhouette vertices form the contour of the object. Except for silhouette vertices, if all the neighbouring triangles of a vertex are interior triangles, this vertex is an *interior vertex*; otherwise it is a *joint vertex*.

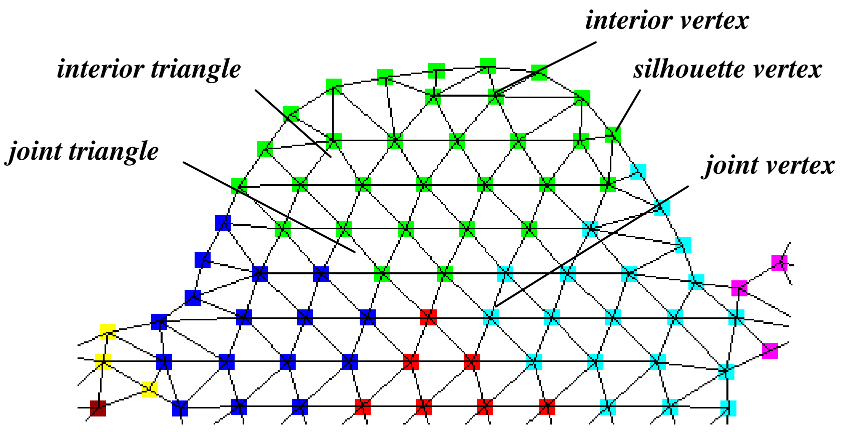


Figure 3.6 Illustration of our definition for different types of vertices and triangles

3.4 2D Shape Deformation

Shape deformation is crucial to the quality of animation and it is an essential step in our technique. The main objective of our algorithm design is both to minimize the boundary change, interior shape distortion and computational overheads. A 2D character is deformed in two stages: skeleton driven deformation for each vertex region (Stage 1) and nonlinear deformation for the joint areas (Stage 2). For Stage 1, since the computation involves simple transformations, it incurs only a small overhead. Stage 2 minimizes implausible deformations. Although the computation is more complex, it involves only a small portion of the vertices.

3.4.1 The Variable-Length Needles Model

The variable-length needles model represents the geometry of the deformable object using a collection of variable-length needles. Each needle links a vertex to its attached skeleton segment. Each needle originates from the skeleton and extends outward in a fixed angle. The vertex is at the end point of a needle. The length of a needle $l(v_i)$ is the Euclidean distance between the vertex and the corresponding skeleton segment. The computation formula is given in (3.1). Figure 3.7 illustrates an example of the variable-length needles model.

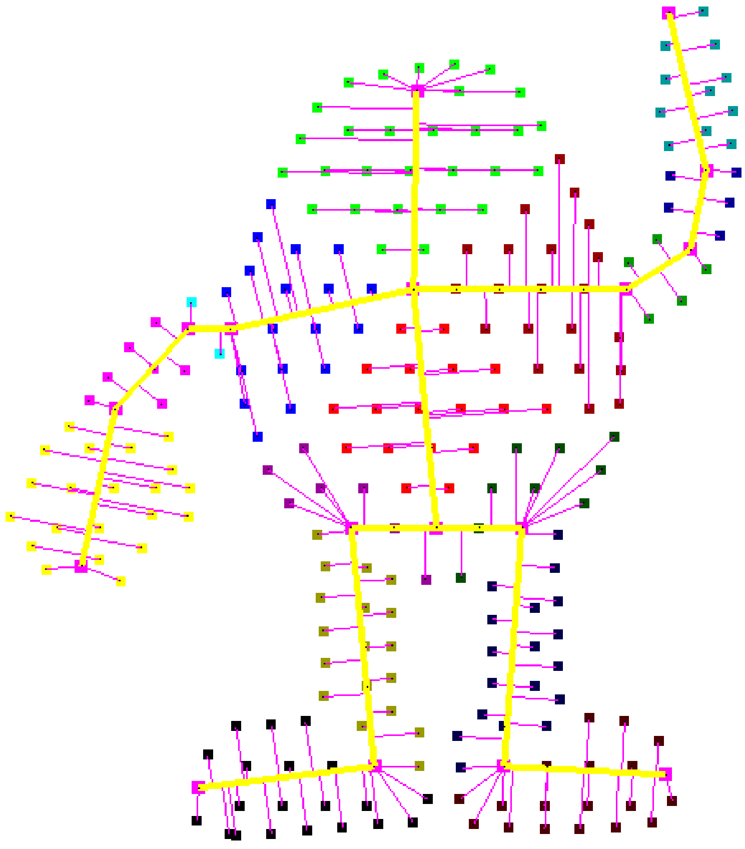


Figure 3.7 Variable-length needles model for a cartoon character

3.4.2 Stage One: Skeleton Driven Deformation

In skeleton driven deformation, the geometry of all vertices is determined only by the position of the corresponding skeleton segment. Because the points are close to the skeletal segment, it is reasonable to regard the needles to be subject to the affine transformations of the skeleton segment during animation. Rotation and scaling are legitimate transformations here. During transformation, the length and direction of the needles relative to the skeleton segment are unchanged if the length of the skeleton segment is constant, leading to fast computation of the new coordinates of the mesh vertices.

Cartoon characters often exhibit significant squashing and stretching deformations. An advantage of using our needle model is that the area enclosed by the boundary can be maintained mostly by ensuring the change of the length of a needle to be reciprocal of the change of the linked skeletal segment length. Because the needles cover the character's surface, this simple method nearly preserve the global area of the character. Figure 3.8 demonstrate the effect of this global area preservation. One skeletal segment is used to deform the bottle. Our developed animation system supply two deformation options for users: One is with area preservation (middle). Another is without area preservation (right).

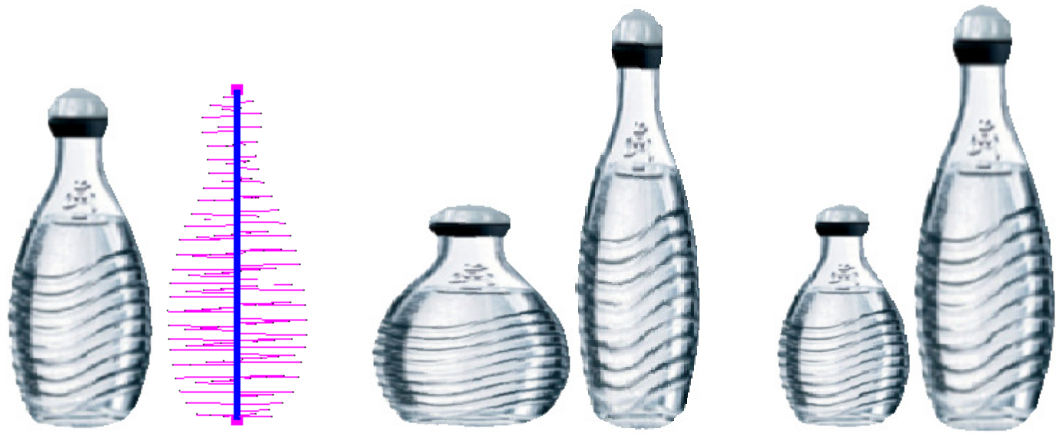


Figure 3.8 Deformation with (middle) and without (right) global area preservation. The original object and variable-length needle model are shown on the left

Figure 3.9 illustrates this deformation process. As can be seen in Figure 3.9 (e), the deformation is quite realistic. However, the texture and contour curve in some joint areas is not sufficiently smooth, and some joint triangles even overlap. This suggests that to minimize shape distortion, the joint areas need to be concentrated and ensured the deformation conforms to the original model. Here some important geometric properties of the 2D model shape, such as the local shape features of the contour, the interior shape smoothness and local area need to be preserved. This process is described in the next section.

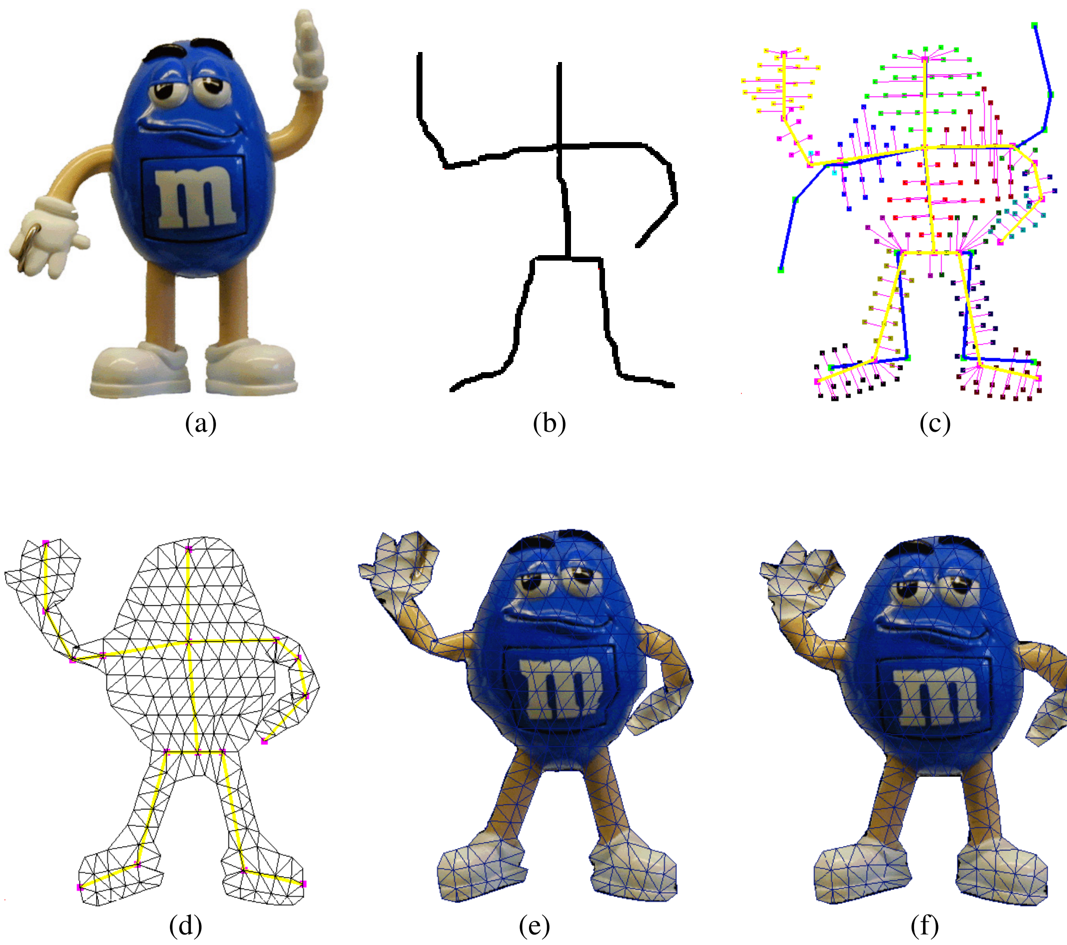


Figure 3.9 Deformation process. (a) Original character, (b) Sketched skeleton, (c) Deformed character displayed as a variable-length needle model. The blue lines represent the skeleton of the original template model before deformation, (d) Mesh and skeleton after the deformation of Stage one, (e) Character after the deformation of Stage one, (f) Character after the deformation of Stage two

3.4.3 Stage Two: Nonlinear Deformation in Joint Areas

As mentioned in Related Work, there are quite few approaches on shape deformation, such as FFD (Free Form Deformation), physically-based method. Here, as it is without parameter setting and fast in convergence, the gradient domain techniques are applied to solve deformation as an energy minimization problem.

The energy function usually contains a term for a position constraint and a term for a detail-preserving constraint. The detail-preserving constraint is usually nonlinear because it involves both the differentials for local details and the local transformations which are position dependent. Three geometric constraints are employed to prevent shape distortion. They are: rotation and scale invariant (RSI) Laplacian coordinates [Sorkine 2004]; mean value coordinates [Floater 2003] and edge length of the triangular mesh. The first constraint preserves the local shape feature of the contour curve; the second constraint preserves the interior smoothness; the third are used to achieve local area preservation [Weng et al. 2006]. Our algorithm can be viewed as a combination of three linear and nonlinear optimizations.

Let (V , E) be the 2D graph of a character's mesh model, where V and E are the sets of vertices and edges respectively. V can be divided into three subsets: silhouette vertices: V_s , joint vertices: V_p and joint vertices: V_q . Assumes that the quantity of vertices is n , V_s contains k silhouette vertices, V_p contains m joint vertices, and V_q contains $n-m-k$ interior vertices. In Stage two, the coordinates of all the interior vertices are fixed. As discussed earlier, the RSI Laplacian coordinates are used to minimize the silhouette distortion in the joint areas. The mean value coordinates are used to minimize the shape distortion of the joint areas. And the edge length of triangular mesh is used to minimize the local area change.

a. RSI Laplacian Coordinates

The curve Laplacian coordinate of vertex v_i is computed using the equation (3.2) below, which formulates the difference with the average of its neighbouring vertices.

$$\delta_i = Lp(v_i) = v_i - (v_{i-1} + v_{i+1})/2 \tag{3.2}$$

Lp is called the Laplace operator of the curve. v_{i-1} and v_{i+1} are the vertices adjacent to v_i on the curve. As the ordinary Laplacian coordinates do not account for rotation and scaling of the curve, here rotation and scale invariant (RSI) Laplacian coordinates [Sorkine 2004] is used to handle the deformation of silhouette. Given that the joint areas where visible distortions occur are mainly interested, so only the silhouette vertices in the joint areas, denoted by $V_{s'}$ are need to constrain. To preserve the local features of the curve concerned, following objective function is minimized:

$$\sum_{v_i \in V_{s'}} \| T(v_i) - T(\tilde{v}_i) \|^2 \tag{3.3}$$

where $T(v_i)$ stands for the RSI Laplacian coordinates of v_i before deformation and $T(\tilde{v}_i)$ stands for the RSI Laplacian coordinates of v_i after deformation. v_i is the vertex *i* before deformation and $\tilde{v}_i$ is the vertex *i* after deformation.

b. Mean Value Coordinates

Mean value coordinates can be regarded as a generalization of barycentric coordinates to k-sided polygons [Floater 2003]. Because they are based on the Mean Value Theorem for harmonic functions, mean value coordinates can be used to reinforce the relative position of each vertex v_i in V_p with its neighbouring vertices v_j in the nonlinear deformation area. Formula (3.4) is used to compute the mean value coordinates of a vertex in the polygon by its neighbouring vertices [Floater 2003].

$$w_{i,j} = \frac{\tan(\alpha_j / 2) + \tan(\alpha_{j+1} / 2)}{| v_i - v_j |} \tag{3.4}$$

where α_j is the angle between vectors $\overrightarrow{v_j v_i}$ and $\overrightarrow{v_{j+1} v_i}$. Each weight $w_{i,j}$ is then normalized by the sum of all weights equal to 1, to form the mean value coordinates of v_i with its neighbouring vertices. According to the property of mean value coordinates, for all joint vertices $v_i \in V_p$, there is:

$$v_i - \sum_{v_i v_j \in E} w_{i,j} v_j = 0 \tag{3.5}$$

To preserve the mean value coordinates in the joint area, following objective function is minimized:

$$\| \tilde{v}_i - \sum_{v_i v_j \in E} w_{i,j} v_j \|^2 \tag{3.6}$$

where $v_i v_j$ is the edge between v_i and v_j.

c. Edge Lengths

The following energy function is used to penalize the edge length variance for the joint triangles:

$$\sum_{v_i, v_j \in V_p, v_i v_j \in E} \| | v_i - v_j | - | \tilde{v}_i - \tilde{v}_j | \|^2 \tag{3.7}$$

where $| v_i - v_j |$ is the edge length of $v_i v_j$ before deformation, and $| \tilde{v}_i - \tilde{v}_j |$ is the edge length of $v_i v_j$ after deformation.

Combining (3.3), (3.6) and (3.7), our overall objective function can be rewritten in the following matrix form:

$$w_1 \| \mathbf{T}\mathbf{V}_{\mathbf{s}'} - \mathbf{T}\tilde{\mathbf{V}}_{\mathbf{s}'} \|^2 + w_2 \| \mathbf{M}_{\mathbf{p}} \tilde{\mathbf{V}}_{\mathbf{p}} \|^2 + w_3 \| \mathbf{H}\mathbf{V}_{\mathbf{p}} - \mathbf{H}\tilde{\mathbf{V}}_{\mathbf{p}} \|^2 \tag{3.8}$$

$\mathbf{V}_{\mathbf{s}'}$ represents the coordinates of k' vertices in $V_{s'}$. $\mathbf{T}$ is a $k' \times k'$ RSI Laplacian coordinates matrix. $\mathbf{M}_{\mathbf{p}}$ is a $m \times m$ mean value coordinates matrix which can be pre-computed before deformation. E_p is the edges set of all joint vertices. Supposing its size is r, $\mathbf{H}$ is a $r \times m$ edge matrix which is used to compute the edge vectors of joint triangles. The sum of weights: w_1, w_2 and w_3 are normalized to 1 and in the experiments, equal weighting is used for each terms. However, it is useful that the user is given the freedom to adjust the weighting to emphasize certain geometric property.

Here to solve this nonlinear optimization problem efficiently, an iterative Gauss-Newton method mentioned in [Press et al. 2007] is adopted. Since only the joint areas with a small number of mesh vertices are processed, using this approach does not significantly impact on the overall performance. This algorithm can be described with following formula:

$$\min_{\mathbf{V}^{k+1}} \| \mathbf{A}\mathbf{V}^{k+1} - \mathbf{b}\mathbf{V}^{k} \|^2 \tag{3.9}$$

In (3.9), $\mathbf{A} = (\mathbf{T}\mathbf{M}_p\mathbf{H})^T$, $\mathbf{b} = (\mathbf{T}\mathbf{V}_{s'}\mathbf{O}\mathbf{H}\mathbf{V}_p)^T$. $\mathbf{V}^k$ is the point position solved in the k-th iteration and $\mathbf{V}^{k+1}$ is the point positions to be computed at *k+1*-th iteration. Since $\mathbf{b}\mathbf{V}^k$ is already known at the current iteration, (3.9) can be solved through a linear least squares system:

$$\mathbf{V}^{k+1} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}\mathbf{V}^k \tag{3.10}$$

Where $\mathbf{A}$ is dependent only on the geometry before deformation. With a feedback substitution in each iteration, this optimization problem can be solved automatically.

The result is shown in Figure 3.9 (f) where both the silhouette and the texture inside the object are smoothly deformed compared with the result of Stage one. For this particular example, the computation converges with 36 iterations. The number of iterations varies with many factors including the shape of the model, the number of vertices and the magnitude of the deformation. In our experiment, the average number of iterations across all the examples is around 35.

3.5 Depth Adjustment and Fine Tuning

Collision detection is a practical problem for the deformation of cartoon characters. When different parts of a character overlap, if the depths are not assigned appropriately, the overlapping parts may interpenetrate (Figure 3.10a). Moreover, assigning static depth values for vertices [Igarashi et al. 2005] does not work in all possible situations. In our system, the dynamic depth adjustment is allowed through interaction. Upon the generation of a new deformed model, user monitors the mesh for self-intersection and set an appropriate depth order to the overlapping parts. When the user clicks any vertex in an overlapping part, all the vertices in this decomposed region will have the same depth value as the clicked one. Figures 3.10 (b) and (c) give an example of two different depth adjustment results.

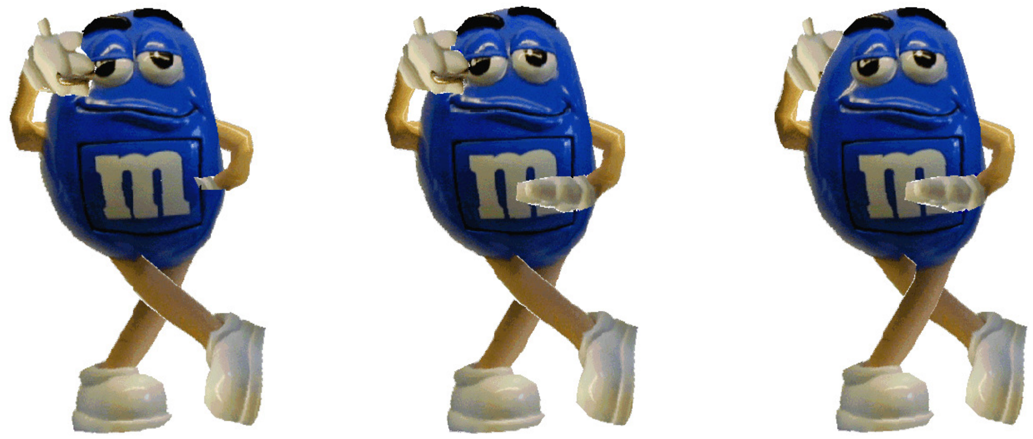

(a) (b) (c)

Figure 3.10 Depth adjustment. (a) Deformed result before depth adjustment, (b) Deformed result after depth adjustment, (c) Depth adjustment result with a different depth order

Our system also allows the user to fine tune the local geometric details of the model in two ways: sketch curves and point dragging. The sketch curves are used to fine tune the silhouette of an object. Similar to the nearest neighbour method, the start and end points of the silhouette segment along the object contour (the shortest Euclidean distances from the start and end points respectively to the sketch curve) are searched. For each vertex on the silhouette segment of the variable-length needle model, the angle between the needle and the skeleton segment, and change the length of the needle to move its end point to the new position on the sketch curve. This process is illustrated in Figure 3.11. In Figure 3.11, S_0 and S_n are the start and end vertices of silhouette segment S_0S_n (blue line). C_0 and C_n are the start and end points of sketch curve C_0C_n (red line). S_i is one vertex in the silhouette segment S_0S_n and S_i' is its new position in the sketch curve after it is deformed. C_j and C_{j+1} are two nearest neighbour points of S_i' on the sketch curve. The coordinates of S_i' can be interpolated by the position of C_j and C_{j+1}.

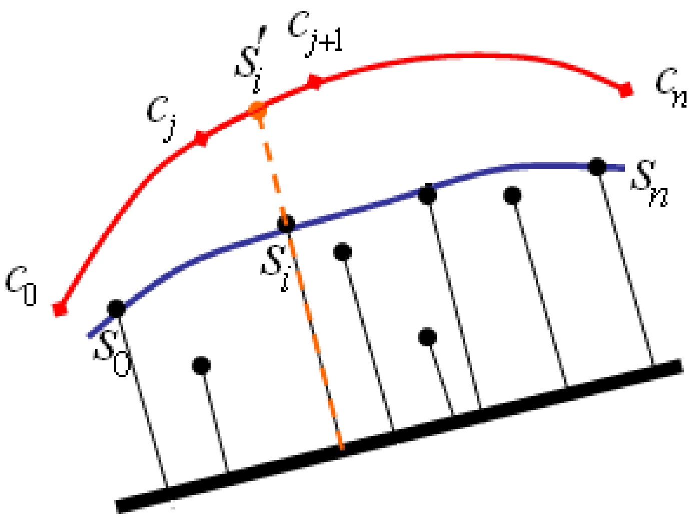
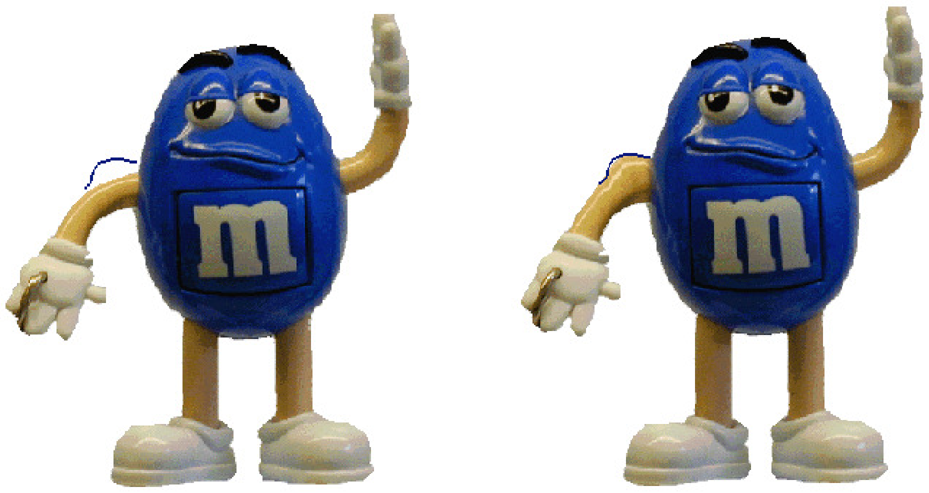
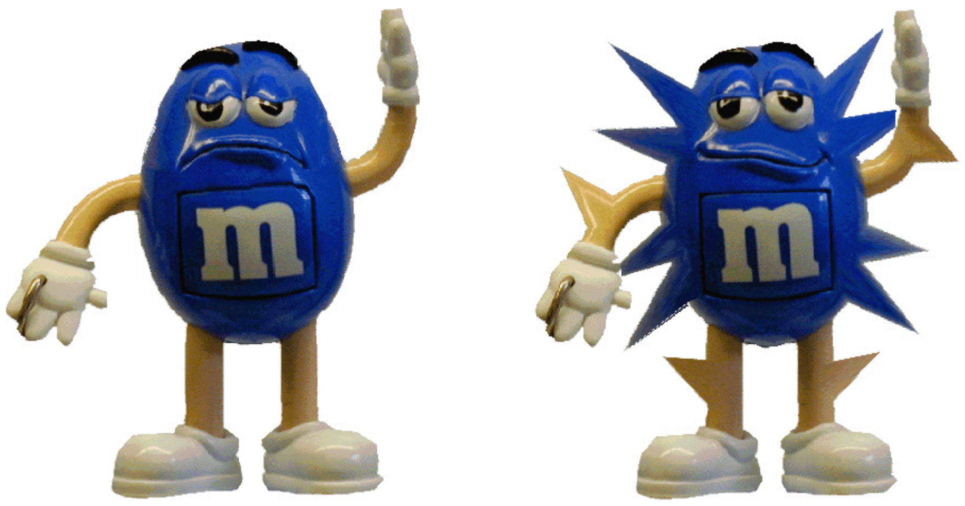


Figure 3.11 Sketch map of the fine tune process with sketch curve.

An example is given in Figure 3.12 (a) where the profile of the right arm is altered with a sketch curve. Point dragging is more straightforward. The user can pick and drag any points to reshape the character. It is very useful to edit or generate detailed shape changes after the main skeleton-driven deformation is complete, such as facial expressions. Figure 3.12 (b) shows two examples. The left one changes the face expression and the right one creates a hedgehog hair style.



(a)



(b)

Figure 3.12 Fine tuning local geometric detail. (a) Sketch curve fine tuning, (b) Deformation through point dragging

3.6 In-betweens

In-between frames are generated by interpolating the deformation produced from the two stages discussed above, skeleton-driven deformation (Stage 1) and non-linear deformation in the joint areas (Stage 2). Many interpolation techniques can be used. In this Section, the generation of in-betweens by three key-frames is explained. Suppose the animation time *t* is from 0 to 1, f_{start}, $f_{skeleton\text{-}driven}$ and f_{end} represent the shape of the initial frame before deformation, the shape generated with the skeleton-driven deformation only and the shape of the end frame, respectively. The computation of each in-between frame ***f(t)*** consists of two elements. The first describes the skeleton-driven deformation which is solved by spherical linear interpolation (slerp). The second element represents the non-linear deformation which can be computed by the linear interpolation of the geometry displacement between $f_{skeleton\text{-}driven}$ and f_{end} . The algorithm expression can be described as (3.11).

$$\underset{t\in[0,1]}{f(t)} = \text{slerp}[f_{start}\times(1-t)+f_{skeleton-driven}\times t]+(f_{end}-f_{skeleton-driven})\times t \quad (3.11)$$

Figure 3.13 gives three in-between frames produced with our system.

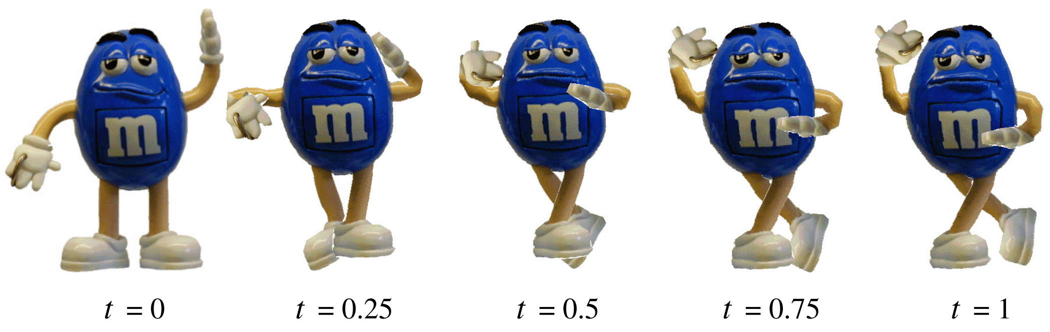


Figure 3.13 Generation of in-between frames

3.7 Experiments and Comparison

Firstly, two experiments are made to compare our algorithm with two previous methods: [Igarashi et al. 2005] and [Weng et al. 2006], which are relevant to 2D animation and represent important references for our research. The first experiment is to deform a long thin elastic object appeared in the work of both [Igarashi et al. 2005] and [Weng et al. 2006]. The skeleton consists of two segments (Figure 3.14 (b)) and the result is given in Figure 3.14 (f). Figure 3.14 (d) and (e) are the results from [Igarashi et al. 2005] and [Weng et al. 2006] respectively. Figure 3.14 (c) is the target to achieve. The result achieved by our algorithm is the closet to the target.

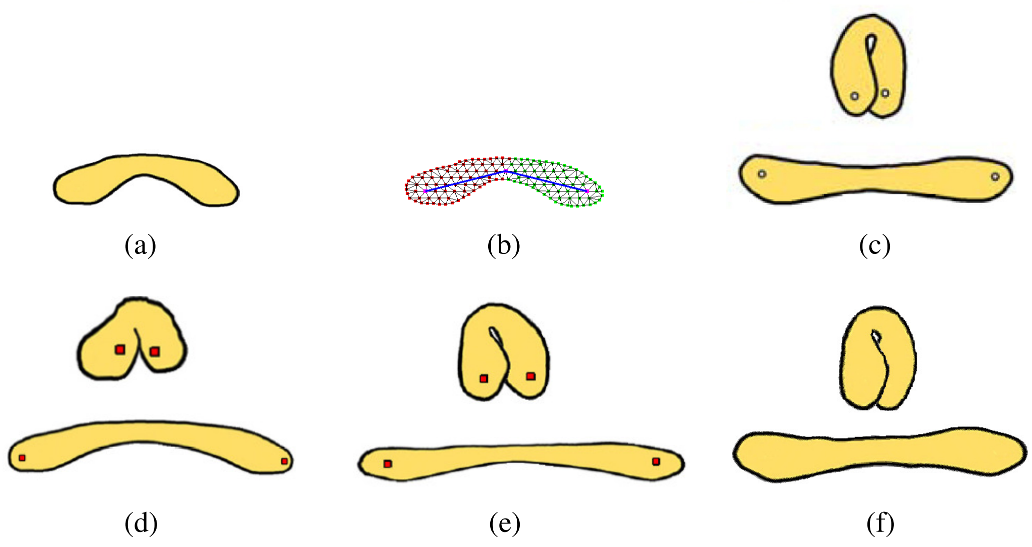
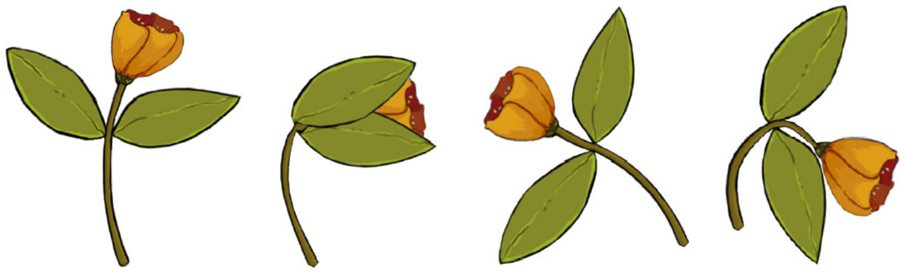
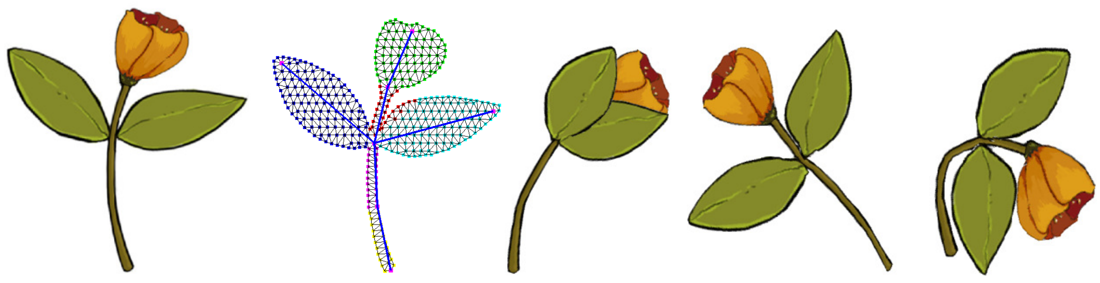


Figure 3.14 Comparing our algorithm with the approaches in [Igarashi et al. 2005] and [Weng et al. 2006]. (a) Original object, (b) Decomposition result, (c) Target result, (d) Deformation result by [Igarashi et al. 2005], (e) Deformation result by [Weng et al. 2006], (f) Deformation result with our algorithm

The second experiment is to comparatively study the computational complexity of our algorithm. In this experiment, a 2D flower model is deformed by our algorithm into four similar postures to those in [Weng et al. 2006]. The deformation results are shown in Figure 3.15. From left to right, the first figure is the original model before deformation and the other three figures are the deformed one. Our deformation algorithm is tested on a 3.2GHz Pentium 4 workstation with 1GB memory. Table 3.1 gives the comparison results. Since our deformation algorithm does not perform nonlinear shape deformation for all triangles, our method takes about a quarter of the time in iteration. This is especially significant when performing larger and more complex animations.



(a)



(b)

Figure 3.15 Flower model deformed by our algorithm and [Weng et al. 2006]. (a) Deformation results with [Weng et al. 2006], (b) Deformation results with our algorithm (from left to right, original template, decomposition result and deformed figures)

Table 3.1 Comparison of data statistics and timing

Cartoon model: Flower	Method in [Weng et al. 2006]	Our deformation algorithm
Boundary vertices	114	123
Interior vertices	256	27 (Joint vertices)
Precomputing time	22ms	9ms
Iteration time	0.589ms	0.143ms

Our third experiment is to investigate the relation between the deformation visual performance and the quantity of vertexes. The following figures illustrate the results of deforming a cartoon character to a same pose with different sampling density. In Figures 3.16 (b), (c) and (d), the object contains 186, 244, 498 vertexes respectively. From the experimental results, the deformation, especially in the area of left arm, becomes smoother when the vertex number increases. But the shape distortion in joint area becomes more obvious as the nonlinear deformation concentrates in joints area. The shape distortion is comparatively enlarged when the size of triangle become smaller. And the computation complexity increases too. So to achieve a balance in appearance and computation efficiency, a medium vertexes number is generally used in discrete sampling. Currently, the uniform sampling method is used in triangulation. One way to improve the problem above is to use the non-uniform sampling to generate more vertices in character's joints area. It can perform better deformation result in stage two.

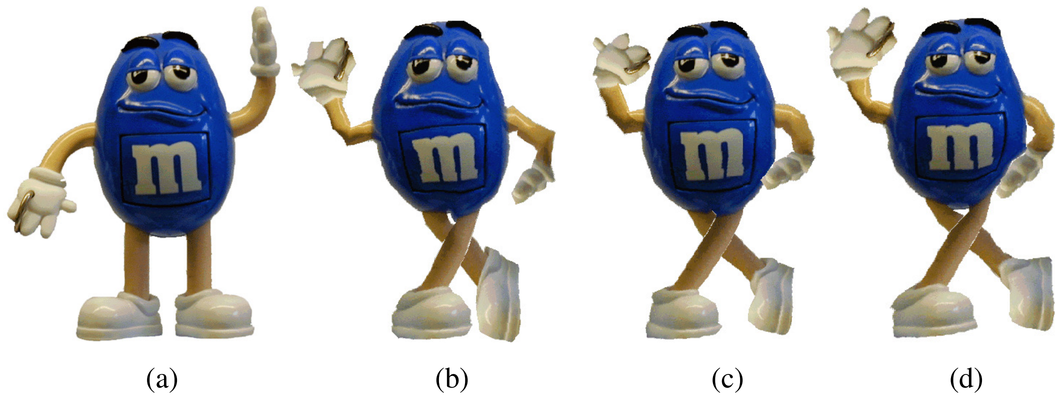


Figure 3.16 Deformed results with different sampling density of cartoon character. (a) original model, (b) deformed figure with 186 vertexes, (c) deformed figure with 244 vertexes, (d) deformed figure with 498 vertexes

3.8 Summary

A sketch-based skeleton-driven 2D animation technique is described in this chapter, which is concerned with the fast production of 2D character animation by sketching the skeletons only. This technique consists of five steps: silhouette detection and triangulation, skeletonization and decomposition, 2D shape deformation, Depth adjustment and fine tuning, and In-betweening.

Given an original image of a character and the sketched skeleton sequence, our prototype system will generate a deformed character with different poses automatically. In comparison with the conventional practice where all key-frames are hand-produced, our method requires much less user input without depriving the animator of controlling the artistic quality. It is much faster and less labour-intensive to create a sequence of 2D animation. The preliminary user tests with our prototype system support this assertion (seen Section 6.4).

Our theoretical contribution in this respect includes a variable-length needle model, which can nearly preserve the area of a character during animation. This is an essential property for exaggerative effect in 2D animation; and the skeleton driven + nonlinear least squares optimization algorithm. The key idea is to deform linearly the areas clearly associated with each skeletal segment and to deform non-linearly the areas around a skeletal joint by preserving the local contour features, the interior smoothness and local area. Experiments demonstrate that our method is much faster than the previous methods and also performs plausible nonlinear shape deformations successfully.

CHAPTER 4

MOTION CAPTURE FOR 2D CHARACTER ANIMATION

Motion capture and retargeting is an important research field in computer animation. However, most research on motion capture and motion retargeting focuses on 3D animation. Although large amounts of video, cartoon and other traditional types of moving images exist, few effective approaches are available to make use of these abundant resources due to the special characteristics and principles of 2D animation [Williams 2001, Isaac 2003]. The objective of this Chapter is to fill this obvious gap. Since our cartoon production technique is skeleton-based, the idea of motion capture from 3D animation can be naturally borrowed to capture the 'motion' of a 2D moving images sequence, and retargeted the captured motion to a different character. On the basis of the sketch-based skeleton-driven 2D animation discussed in Chapter 3, a 2D motion capture and retargeting technique is proposed. With this technique, animators can establish a 2D motion database by extracting the motion characteristics from existing images sequence including video, cartoon and other types of moving images, allowing this enormous asset to be archived and reused.

4.1 Overview

Given an initial frame of an image sequence containing an animated character, users are able to capture the motion by tracking the skeleton joints of each frame. The user identifies the skeleton joints of the initial frame. To avoid the problem caused by temporary loss of texture information mainly due to occlusion of articulated body parts, our tracking algorithm makes use of both geometric and visual features. The displacements of these joints can then be easily retargeted onto a model using the skeletal deformation technique described in Section 3.4.

4.2 Preconditions

Once the first frame is identified from a moving image sequence, the curve skeleton is automatically extracted in the same way as was described in Section 3.3. Based on this curve skeleton, the animator marks the joints on the image. To capture the motion from the subsequent frames/images, the key step is to track the positions of the joints. Since users are concerned with 2D images/frames, it is reasonable to assume that the texture change of the joints is small between any two consecutive frames. Our design therefore is to track the joint positions using texture as the visual cue. Our technique will capture the motion of an *original character* and retarget it to the *target character*. To ensure it works effectively, the image sequences and the target character should satisfy the following preconditions:

1. The image sequence is consistent, i.e. the change between two consecutive frames is relatively small.
2. The target character has the same topology and a similar pose to the original character in the first frame of sequence.
3. There is no occlusion for all the feature joints in the first frame.

What needs pointing out is that our motion capture method is not limited to cartoon sequences only. It can motion capture a cartoon sequence, a video sequence and a rendered 3D animation sequence.

4.3 Initialization

For a given image sequence or video as the input, the system first subtracts the background for each frame using the Interactive Video Cutout system in [Wang et al. 2005]. Figure 4.1 (b) shows the result. The user then locates all the joints by marking small rectangles on the original character to indicate the joint positions, using the automatically generated curve skeleton as a guide (Figure 4.1 (c)). The size of the rectangle for each joint is variable and can be controlled by the user. Here the Hilditch's thinning algorithm mentioned in Section 3.3 is still adopted to generate the curve skeleton.

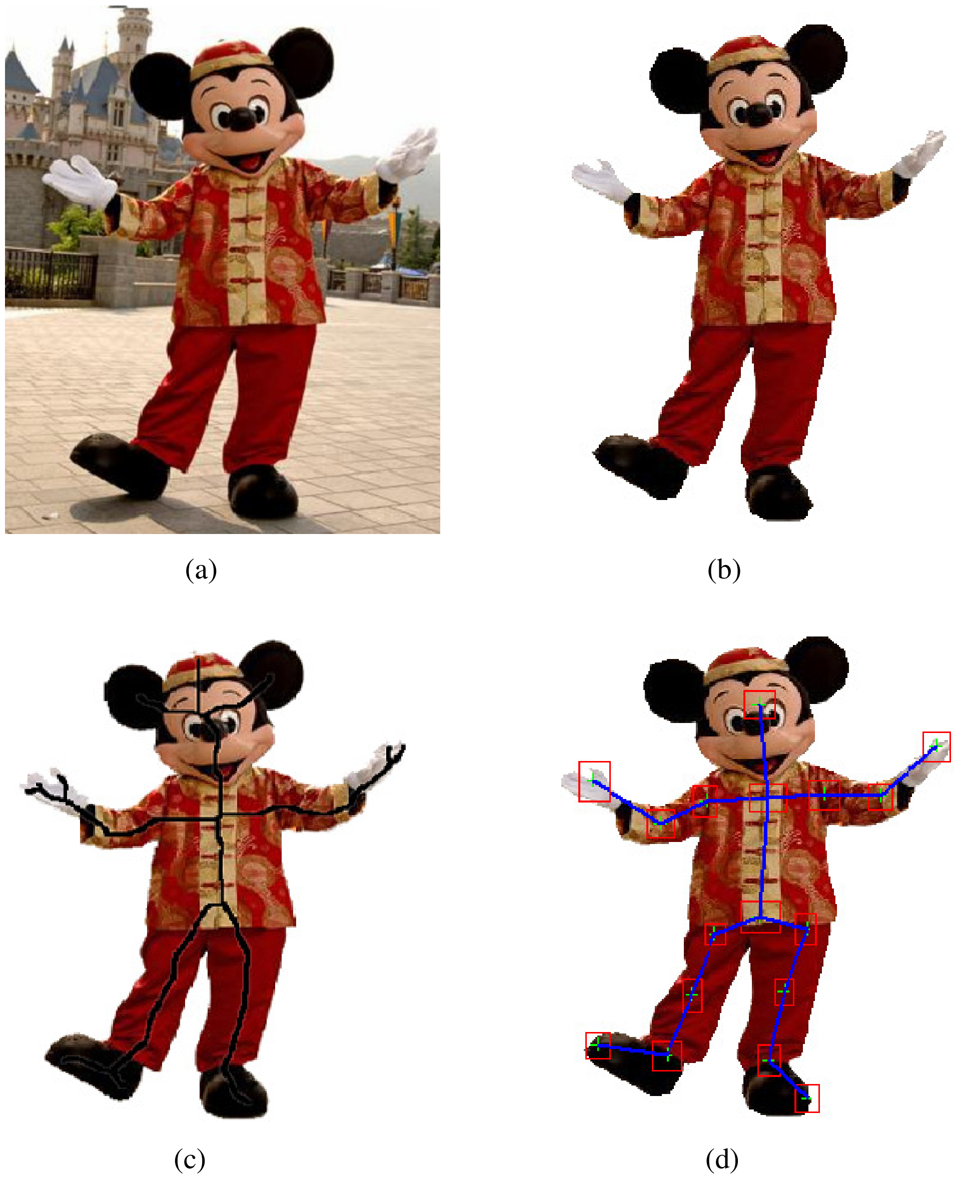


Figure 4.1 Initial setup for motion capture. (a) Original video in the first frame, (b) Background subtraction result, (c) Curve skeleton generation, (d)Original character in the first frame and located joints (Copyright: Disney)

Figure 4.1 (d) shows the joints location of *original character* to be tracked. The red rectangles represent the located joint regions. Connecting all the joints leads to the generation of an animation skeleton. To identify the skeleton in the subsequent frames, our idea is to track the joint positions in these frames. To map the captured motion to a target character, the target character is required to have a similar topology and pose to those of the original character. Figure 4.2 shows an example of target cartoon character. And the skeletonization and decomposition are also made.

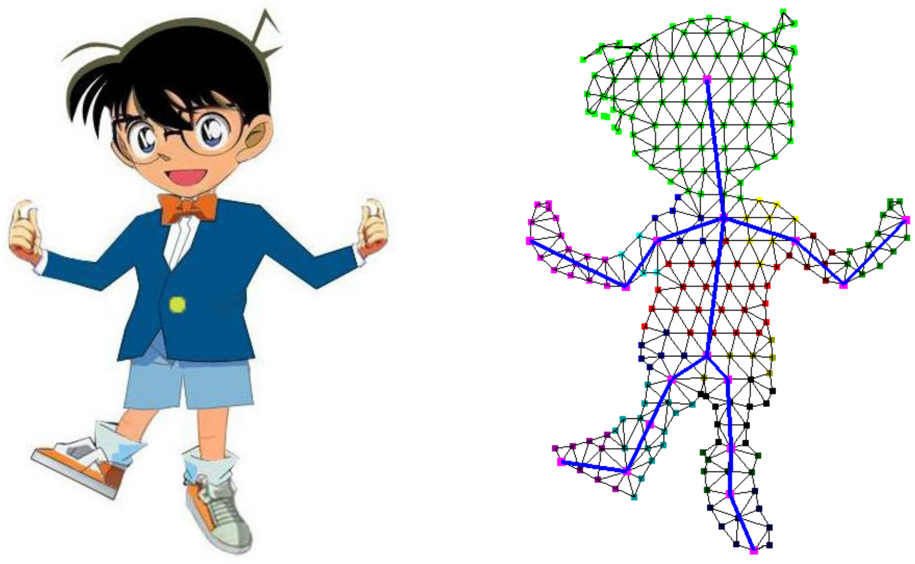

(a) (b)

Figure 4.2 Target character. (a) A cartoon character (b) Skeleton and decomposition result (Copyright: SEGA)

4.4 Tracking

Moving images of static objects can be relatively easy to track with colour information alone. But it is not sufficient for articulated characters. This is because parts of a character may overlap from time to time (e.g. Figure 4.7 frame 6, 12, 18) where colour information disappears. In order to get around this issue, two types of features: visual and geometry are used. The visual feature can be regarded as the template matching. The geometric feature allows the joint positions to be predicted in a next frame by estimating the velocity of the joints.

4.4.1 Visual Feature: Template Matching

For the visual feature, an *n* dimensional feature vector $\mathbf{C}_t = [\mathbf{c}_{1t}, \mathbf{c}_{2t}, \ldots, \mathbf{c}_{mt}, \ldots \mathbf{c}_{nt}]^{\mathrm{T}}$ is used, where $\mathbf{c}_{mt}$ is the texture matrix of the *m*-th rectangle region at frame *t*. The joint (the centre of the corresponding rectangle) is tracked between consecutive frames by searching the closest match of the joint region in the previous frame *t-1*. Therefore (4.1) can be used to describe the searching.

$$\min F(\mathbf{c}_{mt}, \mathbf{c}_{mt-1}) \tag{4.1}$$

s. t. $F(\mathbf{c}_{mt}, \mathbf{c}_{mt-1}) = RedDiff(\mathbf{c}_{mt}) + GreenDiff(\mathbf{c}_{mt}) + BlueDiff(\mathbf{c}_{mt})$

$$RedDiff(\mathbf{c}_{mt}) = \sum_{i=0}^{u}\sum_{j=0}^{v}(pixel_t[i][j].red - \mathbf{c}_{mt-1}[i][j].red)^2$$

$$GreenDiff(\mathbf{c}_{mt}) = \sum_{i=0}^{u}\sum_{j=0}^{v}(pixel_t[i][j].green - \mathbf{c}_{mt-1}[i][j].green)^2$$

$$BlueDiff(\mathbf{c}_{mt}) = \sum_{i=0}^{u}\sum_{j=0}^{v}(pixel_t[i][j].blue - \mathbf{c}_{mt-1}[i][j].blue)^2$$

Where $pixel_t[i][j]$ represents each pixel in the moving rectangle at frame *t*. $F(\mathbf{c}_{mt}, \mathbf{c}_{mt-1})$ represents the colour difference of moving rectangle $\mathbf{c}_m$ between frame *t* and frame *t-1*. The length and height of moving rectangle $\mathbf{c}_m$ are fixed for each frame. This process is similar to the template matching in image processing. Figure 4.3 gives an example of tracking the left knee of character to illustrate this colour feature matching process. The moving feature is the blue rectangle in the images.

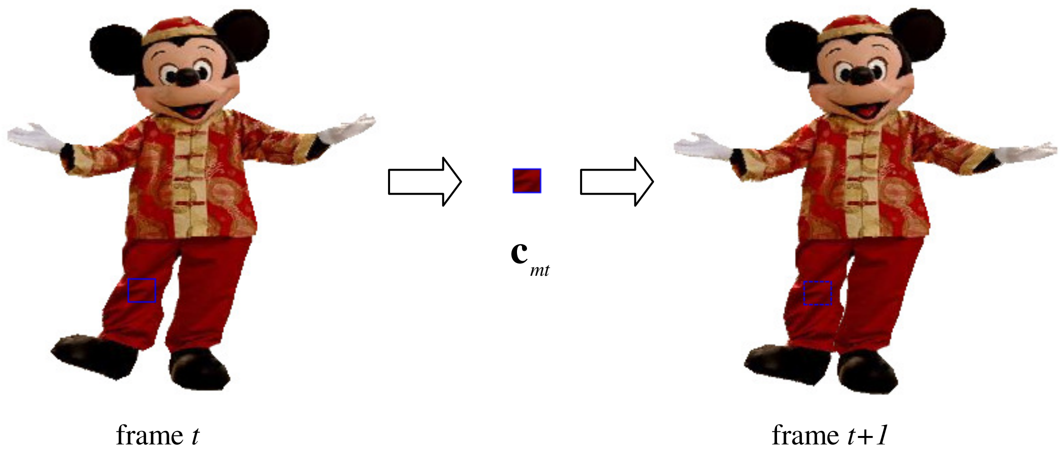


Figure 4.3 Colour feature matching of joint region $\mathbf{c}_{mt}$ from frame t to frame $t+1$

During colour feature matching, the start point is the predicted position of the moving rectangle centre in the current frame (See 4.4.2). Then the 8 neighbourhood pixels of centre point will be searched in anticlockwise. The searching route is illustrated in Figure 4.4.

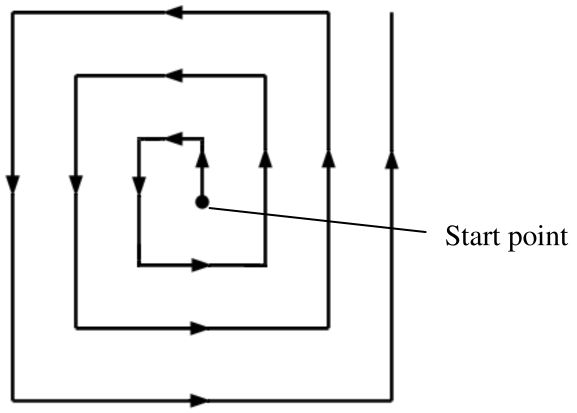


Figure 4.4 Searching route in colour feature matching

One problem needs solving in colour feature matching is the rotation of moving rectangle, especially for the joints of character limbs. Figure 4.5 illustrates the rotation of moving rectangles for a limb joints.

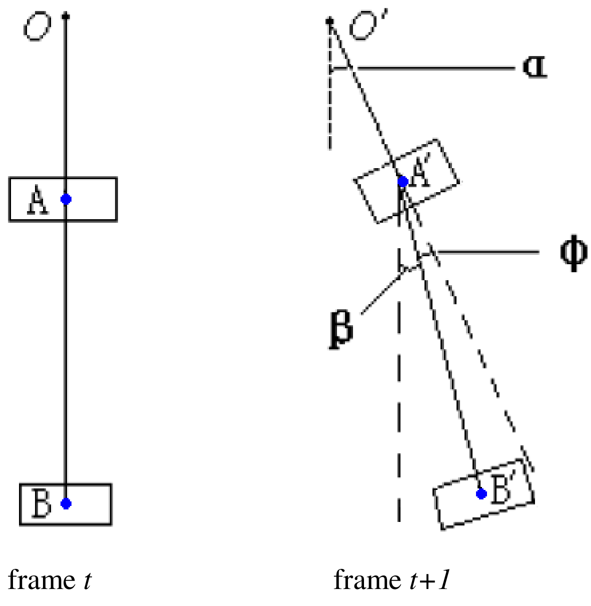


Figure 4.5 Rotation of moving rectangles for a limb joints

In Figure 4.5, O, A, B are the moving rectangles for a limb at frame *t*. O', A', B' are the corresponding moving rectangles at frame *t+1*. α and ϕ are the rotation angle of A' and B'. Then (4.2) is used to compute α and ϕ.

$$\Delta x = (x_{t+1}(A') - x_{t+1}(O')) - (x_t(A) - x_t(O)), \Delta y = (y_{t+1}(A') - y_{t+1}(O')) - (y_t(A) - y_t(O))$$

$$\alpha = 2arctg(\Delta y / \Delta x) \tag{4.2}$$

$$\Delta x' = (x_{t+1}(B') - x_{t+1}(A')) - (x_t(B) - x_t(A)), \Delta y' = (y_{t+1}(B') - y_{t+1}(A')) - (y_t(B) - y_t(A))$$

$$\beta = 2arctg(\Delta y' / \Delta x')$$

$$\phi = \alpha - \beta$$

In (4.2), $x_t(P)$ and $y_t(P)$ are the centre coordinates of moving rectangles *P* in X and Y orientation at frame *t*. $x_{t+1}(P)$ and $y_{t+1}(P)$ are the predicted centre coordinates of moving rectangles *P* at frame *t+1*. For sub-pixel problem in the image rotation, the re-sampling method [Jain 1989] is used to solve it.

4.4.2 Geometric Feature: Position Prediction

Geometry is another useful feature for the tracking due to its consistency during the motion. Here the position of joint is used as the geometric feature. One effective way to track the object with the position information is Kalman filter [Kalman 1960], which is widely used in computer vision. Firstly, a brief introduction of its working principle is given.

a. The Kalman filter

The Kalman filter can be regarded as a recursive estimator, which is based on linear dynamical systems discretized in the time domain. They are modelled on a Markov chain built on linear operators perturbed by Gaussian noise [Grewal and Angus 2001]. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. Kalman filter is purely used in the time domain. This feature is different from many other filters (for example, a low-pass filter), which are formulated in the frequency domain and then transformed back to the time domain for implementation. In the following, the notation $\widehat{\mathbf{x}}_{n|m}$ represents the estimate of $\mathbf{x}$ at time n given observations up to time m. The state of the filter is represented by two variables:

$\widehat{\mathbf{x}}_{k|k}$ is the estimate of the state at time k given observations up to and including time k

$\mathbf{P}_{k|k}$ is the error covariance matrix

The Kalman filter has two distinct phases: Predict and Update. The predict phase uses the state estimate from the previous time step to produce an estimate of the state at the current time step. In the update phase, measurement information at the current time step is used to refine this prediction to arrive at a new, more accurate state estimate, again for the current time step. Figure 4.6 illustrates a simple work flow of the Kalman filter.

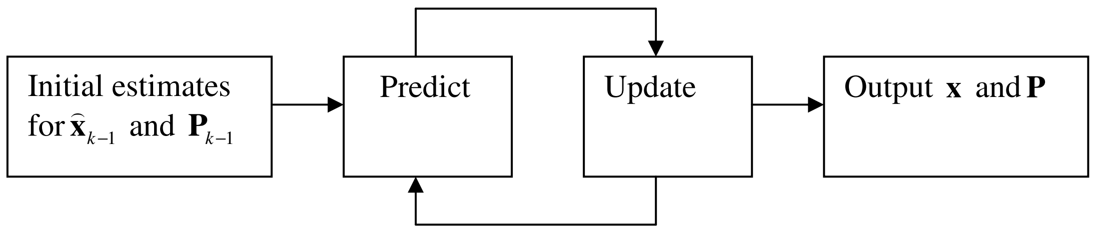

Initial estimates for $\hat{\mathbf{x}}_{k-1}$ and $\mathbf{P}_{k-1}$
Predict
Update
Output $\mathbf{x}$ and $\mathbf{P}$

Figure 4.6 Operation of the Kalman filter

b. Simplified Prediction Model

The 2D moving image sequence can be treated as a dynamical system. The position of interested tracking point is $\mathbf{P} = (x, y)^{\mathrm{T}}$, and can be described as $\mathbf{P} = \mathbf{P}_0 + \eta$. $\mathbf{P}_0$ is the ideal position of the interested point. $\mathbf{P}$ is the tracked position of this point by the Kalman filter. η is Gaussian noise. Covariance matrix is $\mathbf{R}$. The state vector $\mathbf{S} = (\mathbf{P}, \mathbf{P}', \mathbf{P}'')$, $\mathbf{P}' = (x', y')^{\mathrm{T}}$, x' and y' are the velocity of tracking points in X and Y direction. $\mathbf{P}'' = (x'', y'')^{\mathrm{T}}$, x'' and y'' are acceleration of tracking points. Then the state function can be expressed as:

$$\mathbf{S}_{t+1} = \mathbf{F}\mathbf{S}_t + \mathbf{Q}\mathbf{n}_t \tag{4.3}$$

where $\mathbf{F} = \begin{bmatrix} I_2 & I_2\tau & \frac{1}{2}I_2\tau^2 \\ 0 & I_2 & I_2\tau \\ 0 & 0 & I_2 \end{bmatrix}$, $\mathbf{Q} = \begin{bmatrix} \frac{1}{2}I_2\tau^2 \\ I_2\tau \\ I_2 \end{bmatrix}$. t is the frame sequence number. I_2 is the 2×2 identity matrix. τ is the time interval between consecutive frames. $\mathbf{n}_t = \begin{bmatrix} \mathbf{n}_{xt} \\ \mathbf{n}_{yt} \end{bmatrix}$ expresses the Gaussian noise of acceleration in X and Y direction and with a mean value equal to 0. So the tracking point is in a uniformly variable rectilinear motion in an interval.

Assuming that there are n joints (interested point) to be tracked in each frame, the

positions of the rectangle centres at frame *t* form a geometric feature vector:
$\mathbf{G}_t = [\mathbf{g}_{1t}, \mathbf{g}_{2t}, \ldots, \mathbf{g}_{mt}, \ldots \mathbf{g}_{nt}]^{\mathrm{T}} = [(x_1, y_1)_t, (x_2, y_2)_t, \ldots, (x_m, y_m)_t, \ldots (x_n, y_n)_t]^{\mathrm{T}}$.
Because the interval between frames is small, the following simplified prediction model is used to compute the centres of the joint regions $\mathbf{G}_t(\overline{x}_m, \overline{y}_m)$

$$\mathbf{G}_t(\overline{x}_m, \overline{y}_m) = \mathbf{G}_{t-1}(x_m, y_m) + \mathbf{V}_{m,t-1}\tau + \mathbf{A}_{m,t\text{-}1}\tau^2 / 2 \tag{4.4}$$

$$\mathbf{V}_{m,t-1} = [\mathbf{G}_{t-1}(x_m, y_m) - \mathbf{G}_{t-2}(x_m, y_m)] / \tau$$

$$\mathbf{A}_{m,t-1} = [\mathbf{V}_{t-1}(x_m, y_m) - \mathbf{V}_{t-2}(x_m, y_m)] / \tau$$

where $\mathbf{G}_{t-1}(x_m, y_m)$, $\mathbf{V}_{m,t-1}, \mathbf{A}_{m,t-1}$ are the tracked centre position, velocity and acceleration of the *m*-th joint at frame *t*-1. $\mathbf{G}_t(\overline{x}_m, \overline{y}_m)$ is the predicted centre position of the *m*-th joint at frame *t*. τ is the time interval between adjacent frames.

4.4.3 Mixed Optimization

Considering both visual and geometric feature described above, a joint (the centre of the corresponding rectangle) is tracked between consecutive frames by searching the closest match in the previous frame. Using Bayes' rule with a uniform *a priori* distribution case [Jain 1989], the matching is equivalent to finding the maximum of $P(\mathbf{F}_t \mid \mathbf{\Theta})$, where $\mathbf{F}_t$ denotes a feature vector of the character at frame *t*. $\mathbf{\Theta}$ denotes the feature parameters corresponding to the tracked result at frame *t*-1. Here the whole feature space is divided into two sub-spaces: visual and geometric spaces as follows:

$$P(\mathbf{F}_t \mid \mathbf{\Theta}) = P_c(\mathbf{C}_t \mid \mathbf{\Theta}_c) P_g(\mathbf{G}_t \mid \mathbf{\Theta}_g) \tag{4.5}$$

where $P_c(\mathbf{C}_t \mid \mathbf{\Theta}_c)$ and $P_g(\mathbf{G}_t \mid \mathbf{\Theta}_g)$ are PDFs (Probability Density Functions) corresponding to the visual and geometric features respectively. Maximizing $P(\mathbf{F}_t \mid \mathbf{\Theta})$ can be described as the following mixed optimization problem, which is to minimize the sum of Mahalanobis distances in the sub-spaces, i.e.

$$\min \sum_{m=1}^{n} D_{mt} \tag{4.6}$$

s. t. $D_{mt} = w_{\alpha} D_{c,mt} + w_{\beta} D_{g,mt}$

$$D_{c,mt} = F(\mathbf{c}_{mt}, \mathbf{c}_{mt-1})$$

$$D_{g,mt} = (x_{mt} - \overline{x}_{mt})^2 + (y_{mt} - \overline{y}_{mt})^2$$

Where $D_{c,mt}$ and $D_{g,mt}$ are the Mahalanobis distances of visual and geometric features of the *m*-th joint region. w_{α} and w_{β} are the weights used to normalize the corresponding distances. In our work, w_{α} is $1/(255)^2$ and w_{β} is $(1/r)^2$, where r is the radius of the search range. $D_{c,mt}$ represents the colour difference in RGB space. $D_{g,mt}$ represents the Euclidean distance between the moving rectangle centre and the predicted joint region centre.

Figure 4.7b illustrates the tracking result of character in Figure 4.1. Four tracked frames are selected in an 18 frame image sequence. As can be seen from frames 6, 12, 18, the problem of self-occlusion is effectively solved due to the position prediction. However, this tracking method is not without limitations. Since the frame-by-frame tracking is inherently subject to error accumulation, the accuracy is limited to a small number of frames. One effective way to solve this problem is to divide a large image sequence into a number of segments which consist of fewer frames, and the user correct the tracking error manually for the first frame in each segment. Therefore, our system also allows the animators interactively adjust the tracking result at any frame if appropriate.

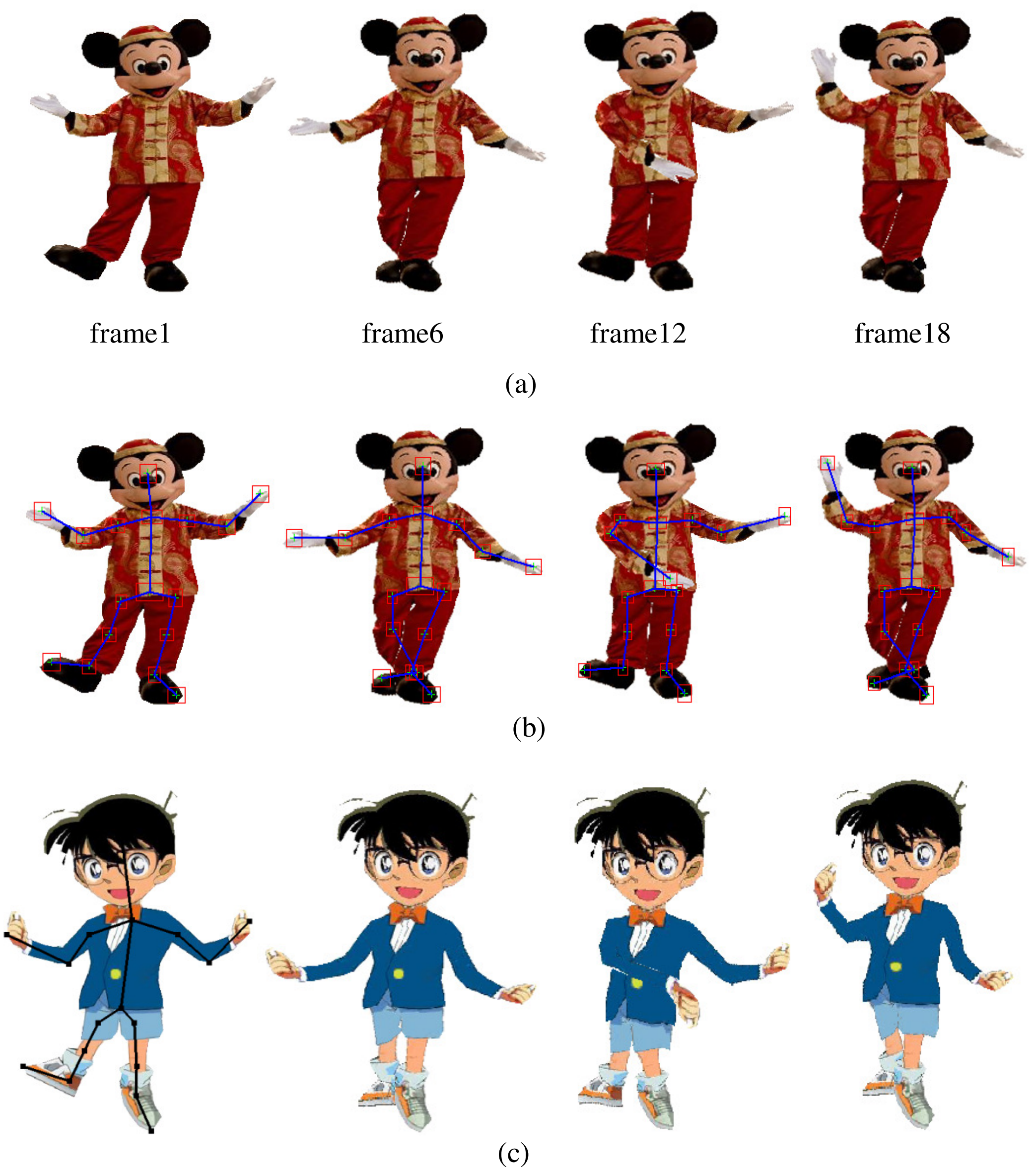


Figure 4.7 Tracking and retargeting. (a) Original image sequence, (b) Joint tracking result, (c) Deformed target character

4.5 Retargeting

The target character should be of the same topology and in a similar pose to that of the original character at the first frame. To retarget a captured motion to the new character, a skeleton is first generated as described in Chapter 3. There is a lot of existing work in retargeting on 3D animation, such as [Gleicher 1998], which is directly applicable to our case. In this thesis, a simple method is implemented to demonstrate the retargeting process. For a 2D character, a skeleton can have both linear (length) and angular (orientation) displacements during motion, i.e. a skeleton segment can stretch / squash and rotate. The basic idea of our simple motion retargeting is to map the captured increments of both length and orientation angle of a skeletal segment, which are computed by:

$$\Delta l_{m,t} = l_{m,t} / l_{m,t-1} \qquad \Delta \alpha_{m,t} = \alpha_{m,t} - \alpha_{m,t-1} \tag{4.7}$$

where $l_{m,t}$, $\alpha_{m,t}$ represent the length and orientation angle of the *m*-th skeleton segment at frame *t*. For the target model, the length $l'_{m,t}$ and orientation angle $\alpha'_{m,t}$ of the *m*-th skeleton segment at frame *t* can be trivially computed by:

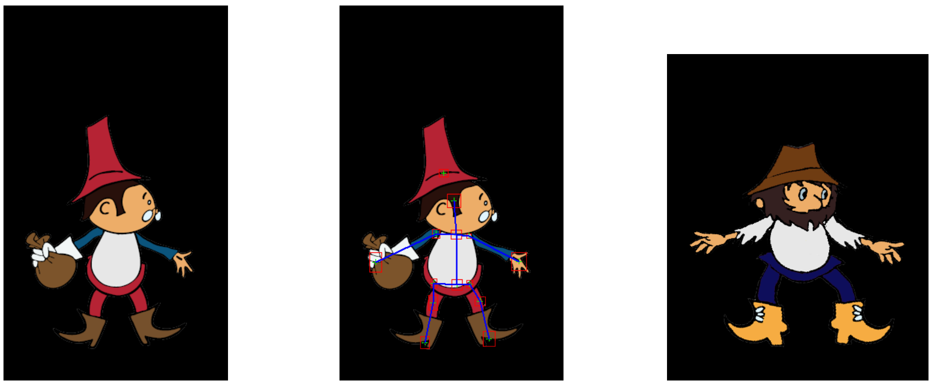
$$l'_{m,t} = l'_{m,t-1} \Delta l_{m,t} \qquad \alpha'_{m,t} = \alpha'_{m,t-1} + \Delta \alpha_{m,t} \tag{4.8}$$

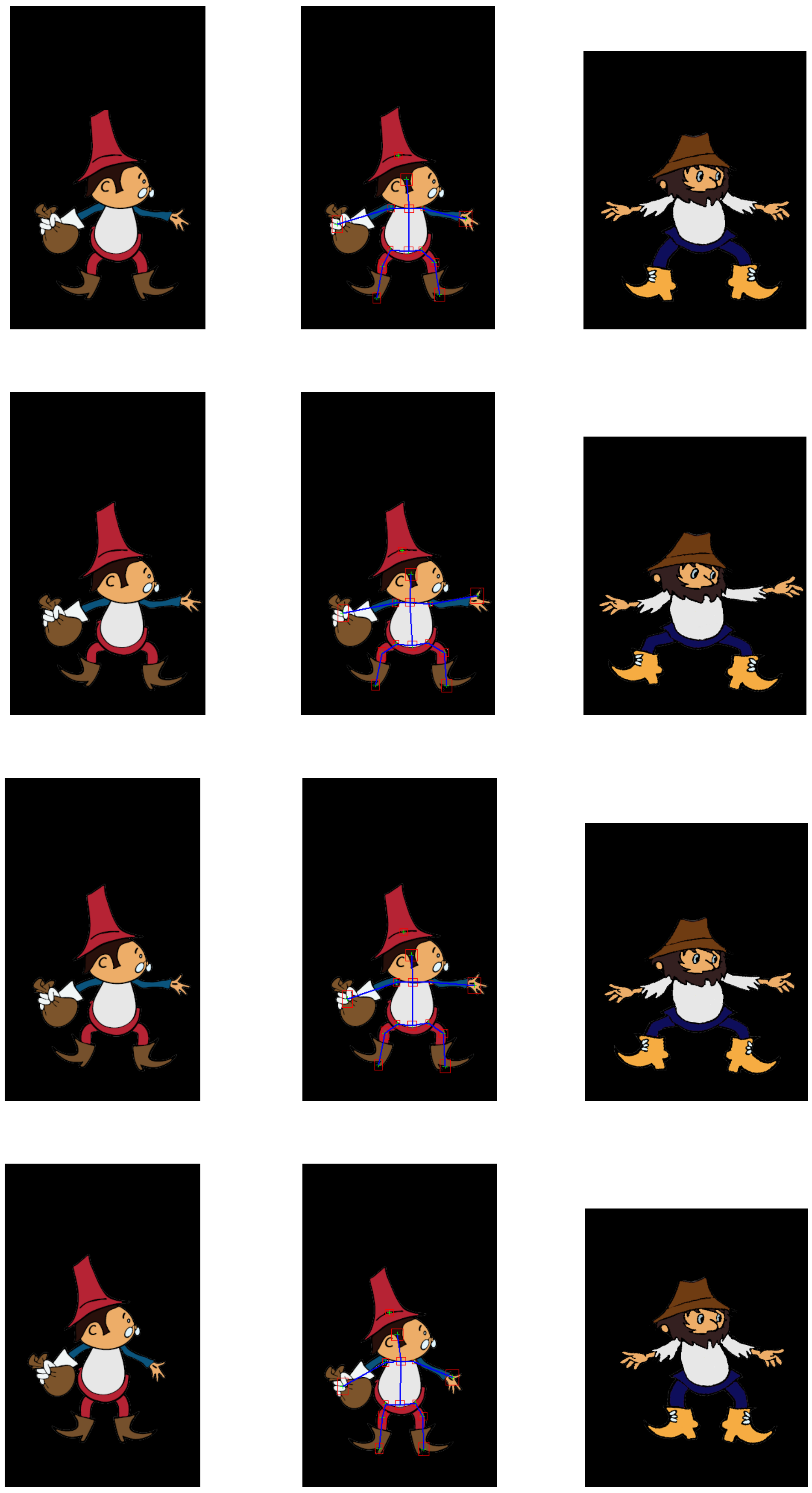
Figure 4.7c illustrates the retargeting result for the target character in Figure 4.2.

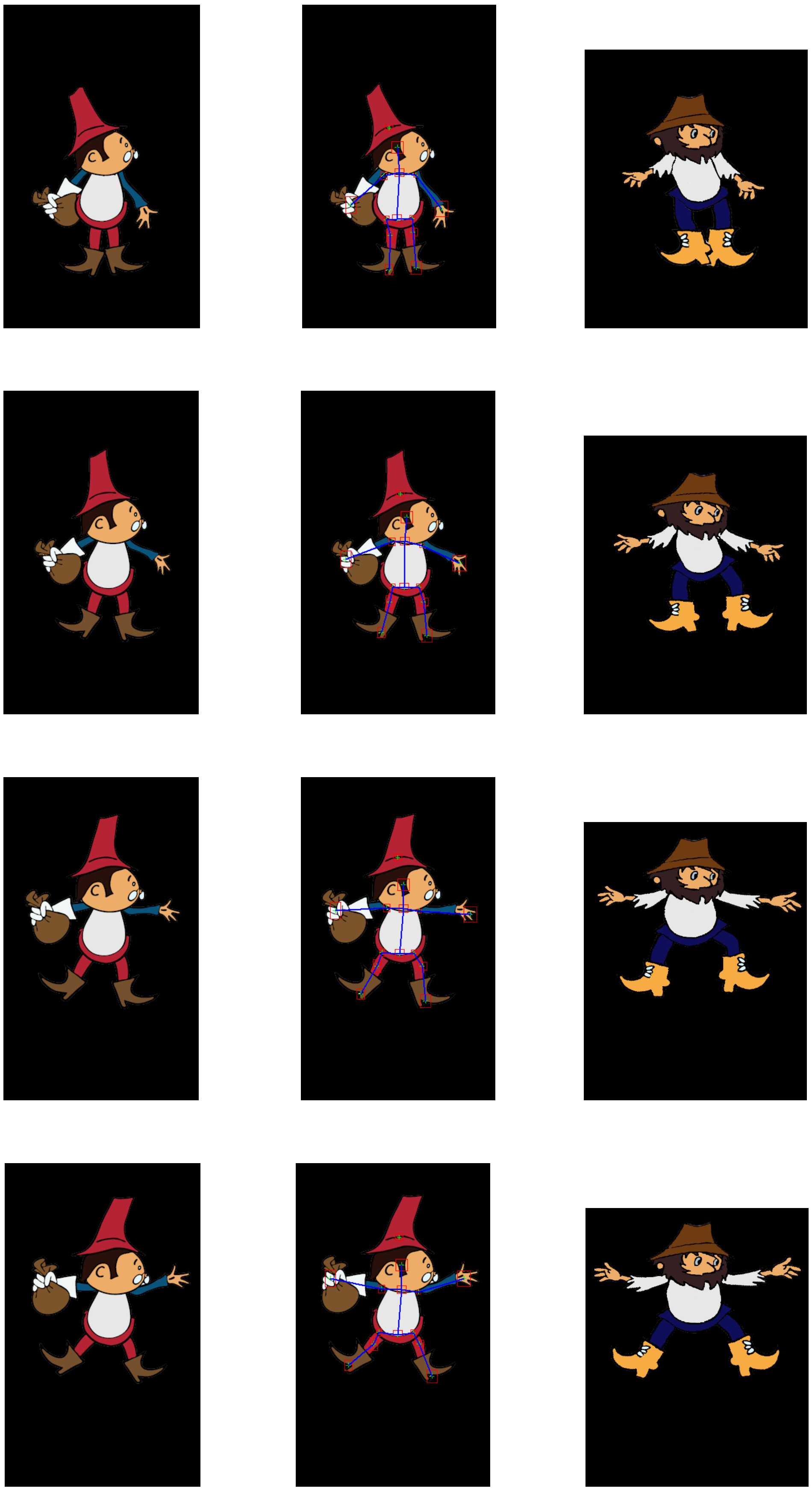
4.6 Experiments

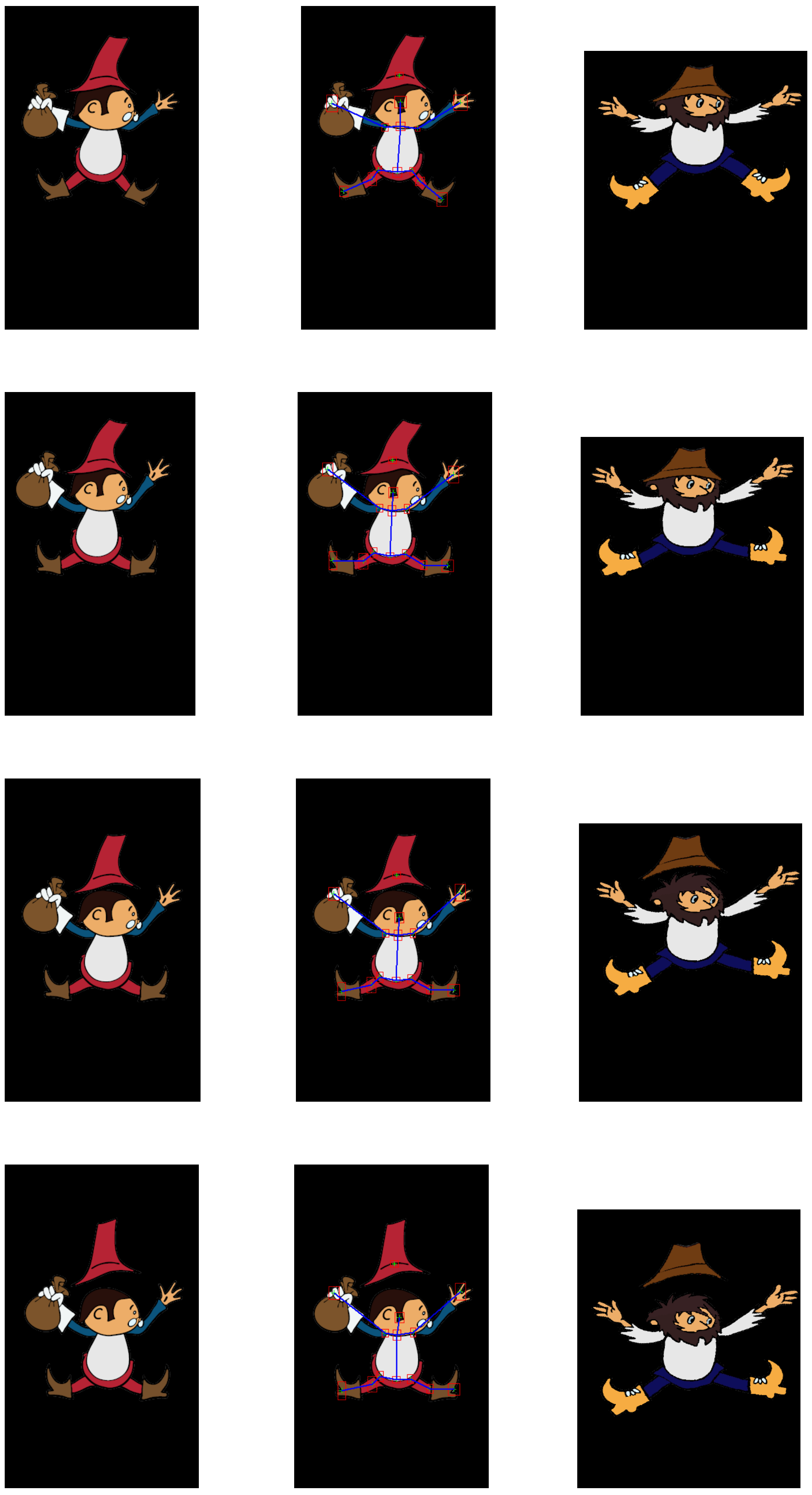
Our motion capture method is not limited to cartoon sequences only. It can motion capture video sequence and rendered 3D animation image sequence as well. Figure 4.7 illustrate the example for a video sequence. The following two experiments are used to test our motion capture method in cartoon sequence and rendered 3D animation image sequence.

The first experiment is to track the joints of a jumping cartoon man and retarget the motion to a new character. The hat and the man are treated as two objects, and tracked separately. Figure 4.8 illustrates the result. There are 16 frames in this sequence.









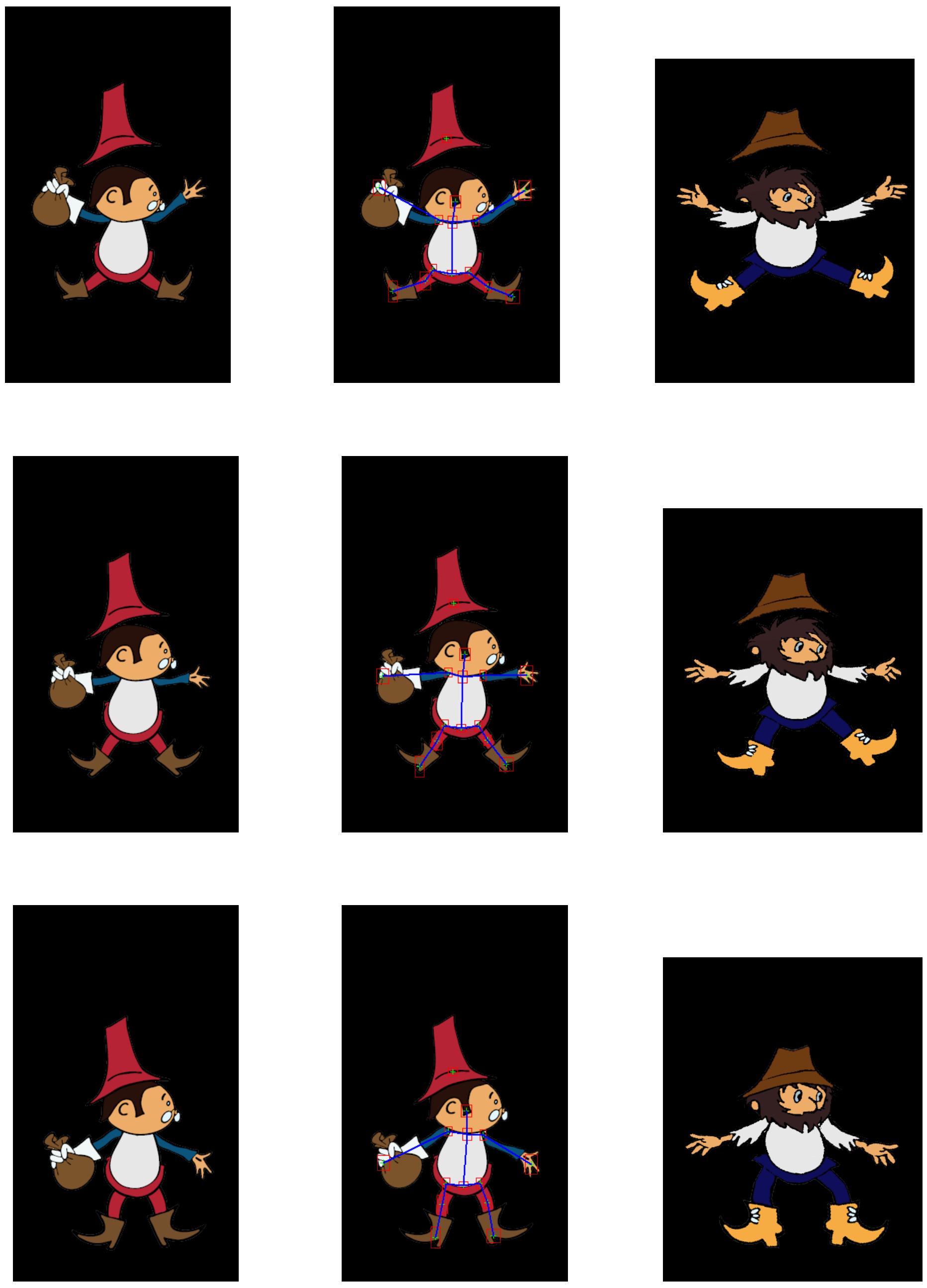
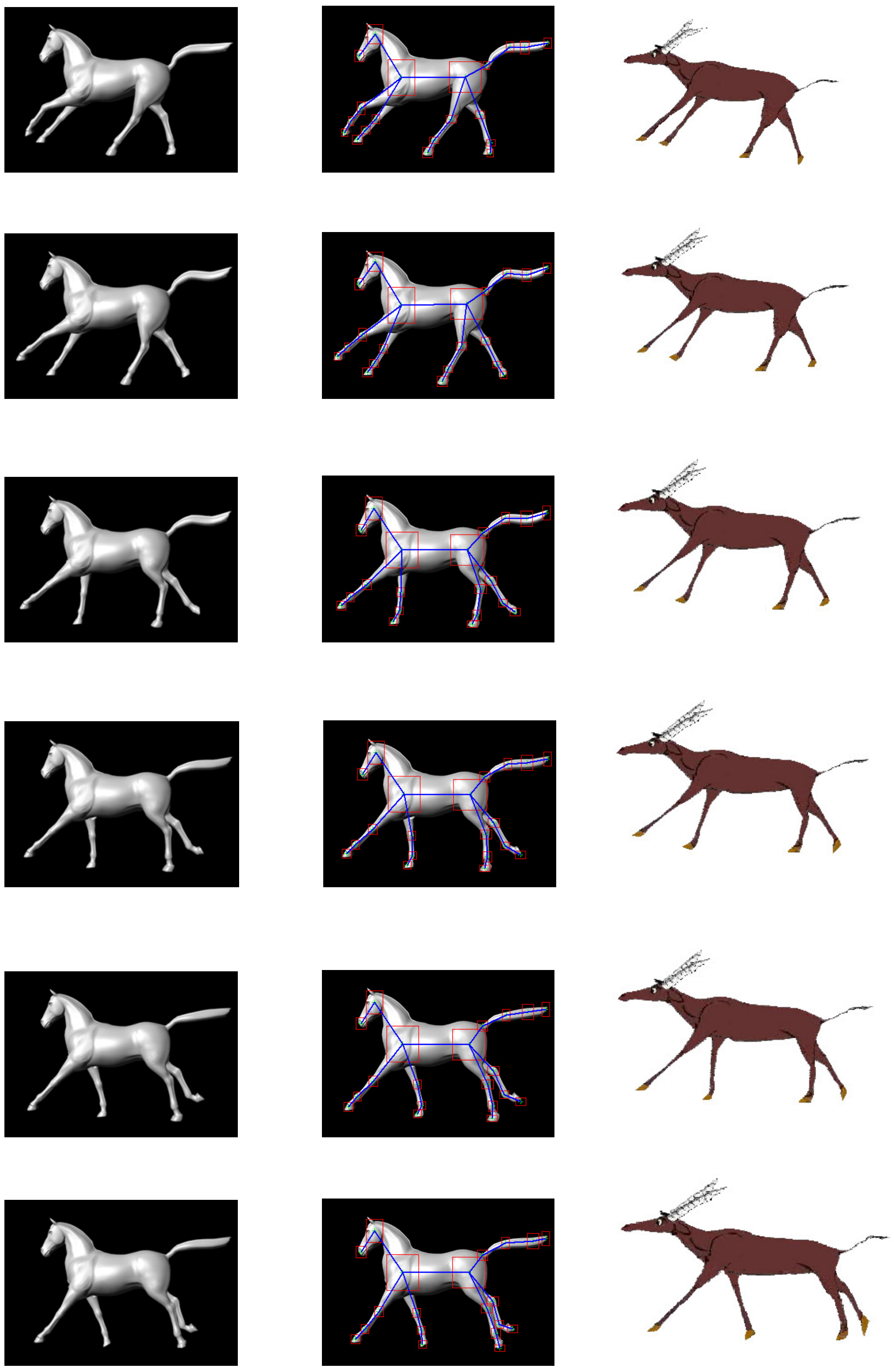
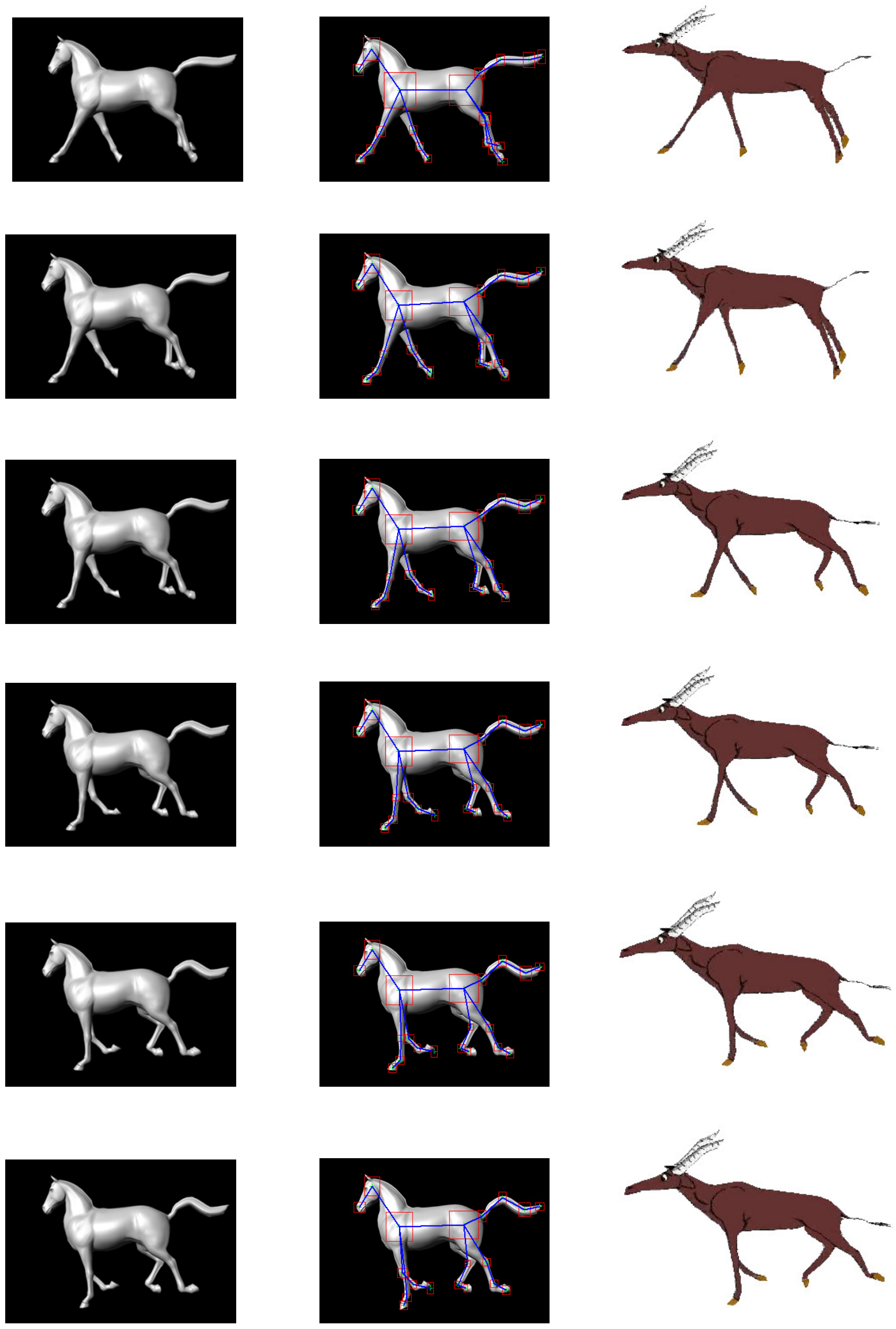
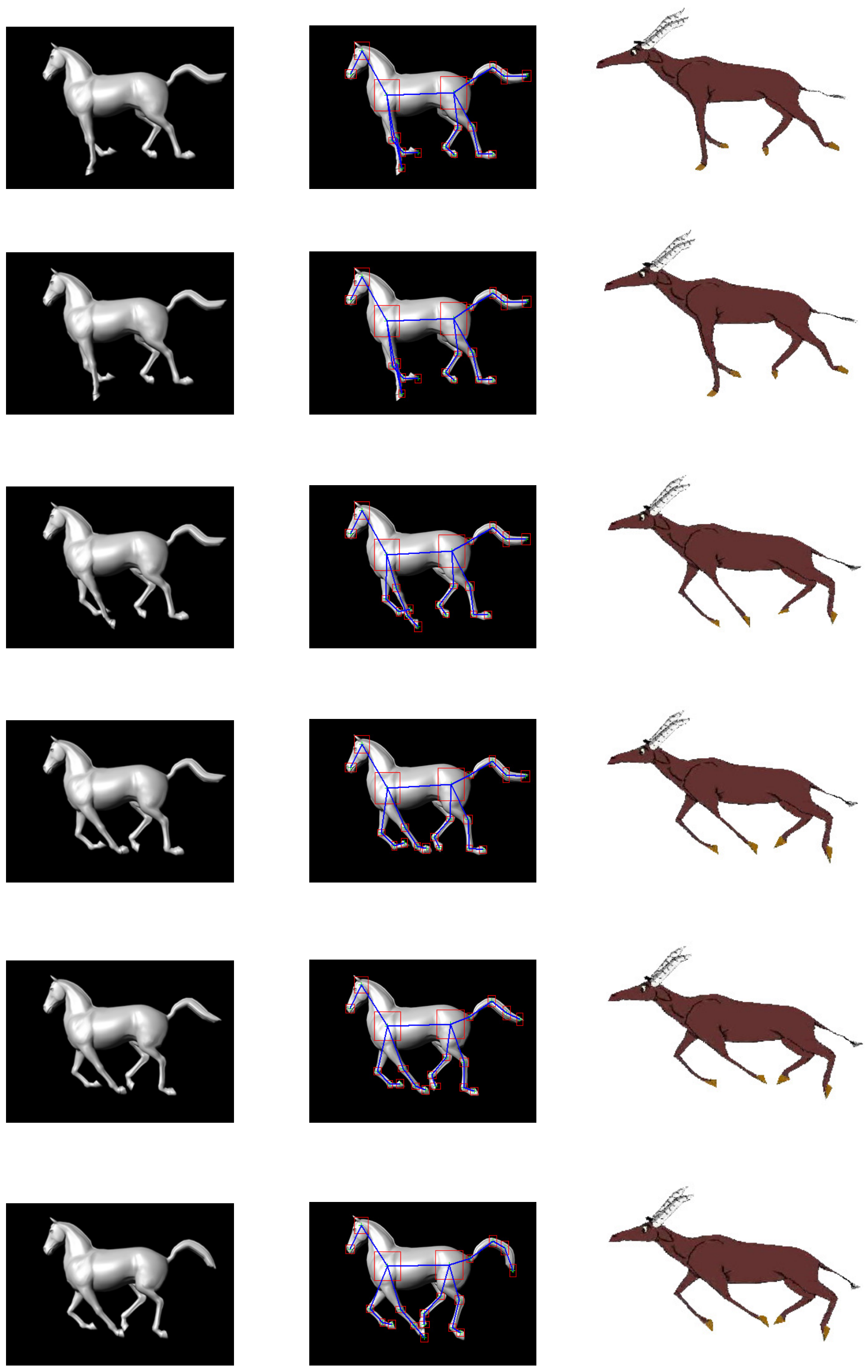


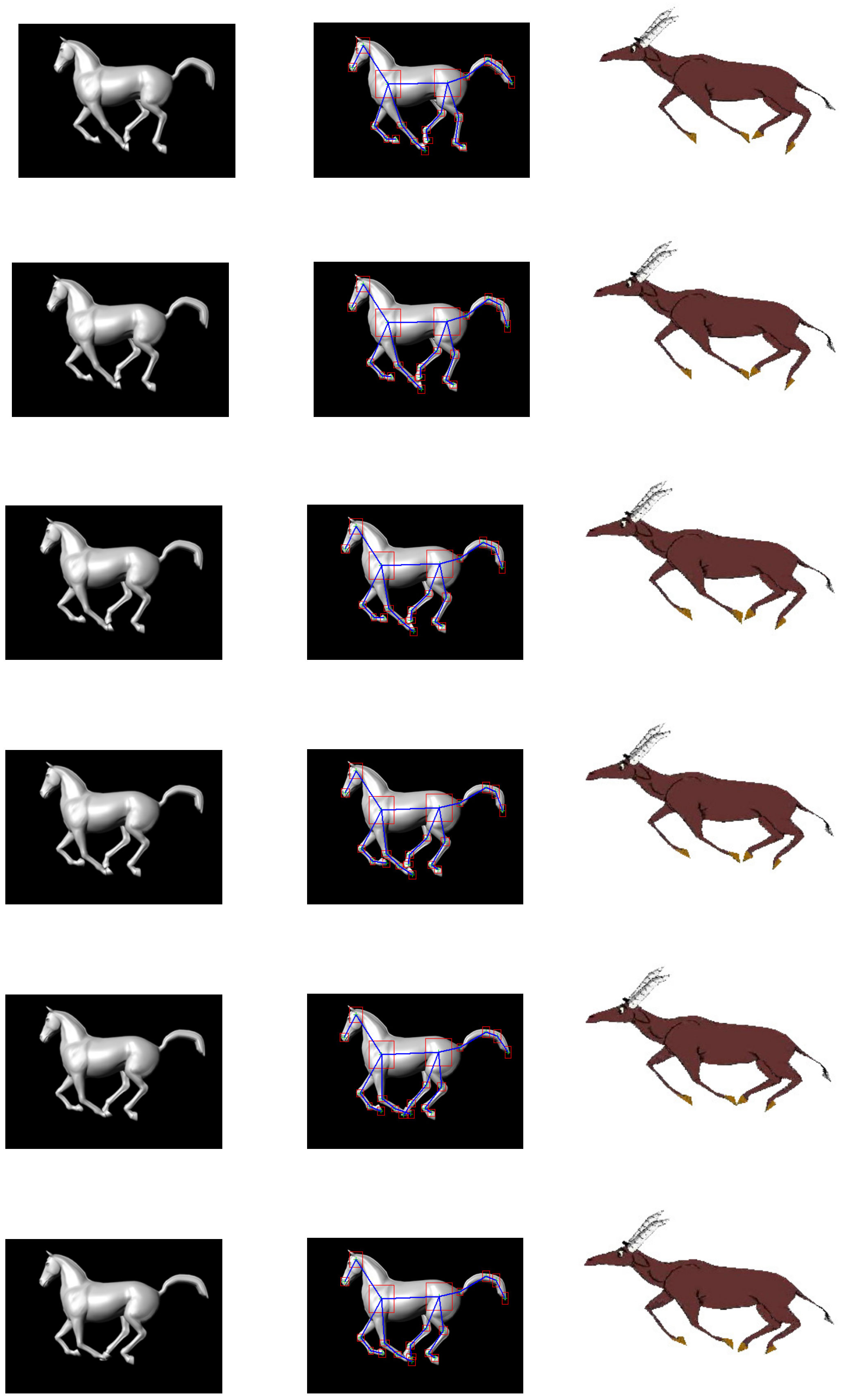
Figure 4.8: Motion of a jumping cartoon man retargeted to a new character. (Left: original image sequence; Middle: joints tracking result; Right: retargeted new character)

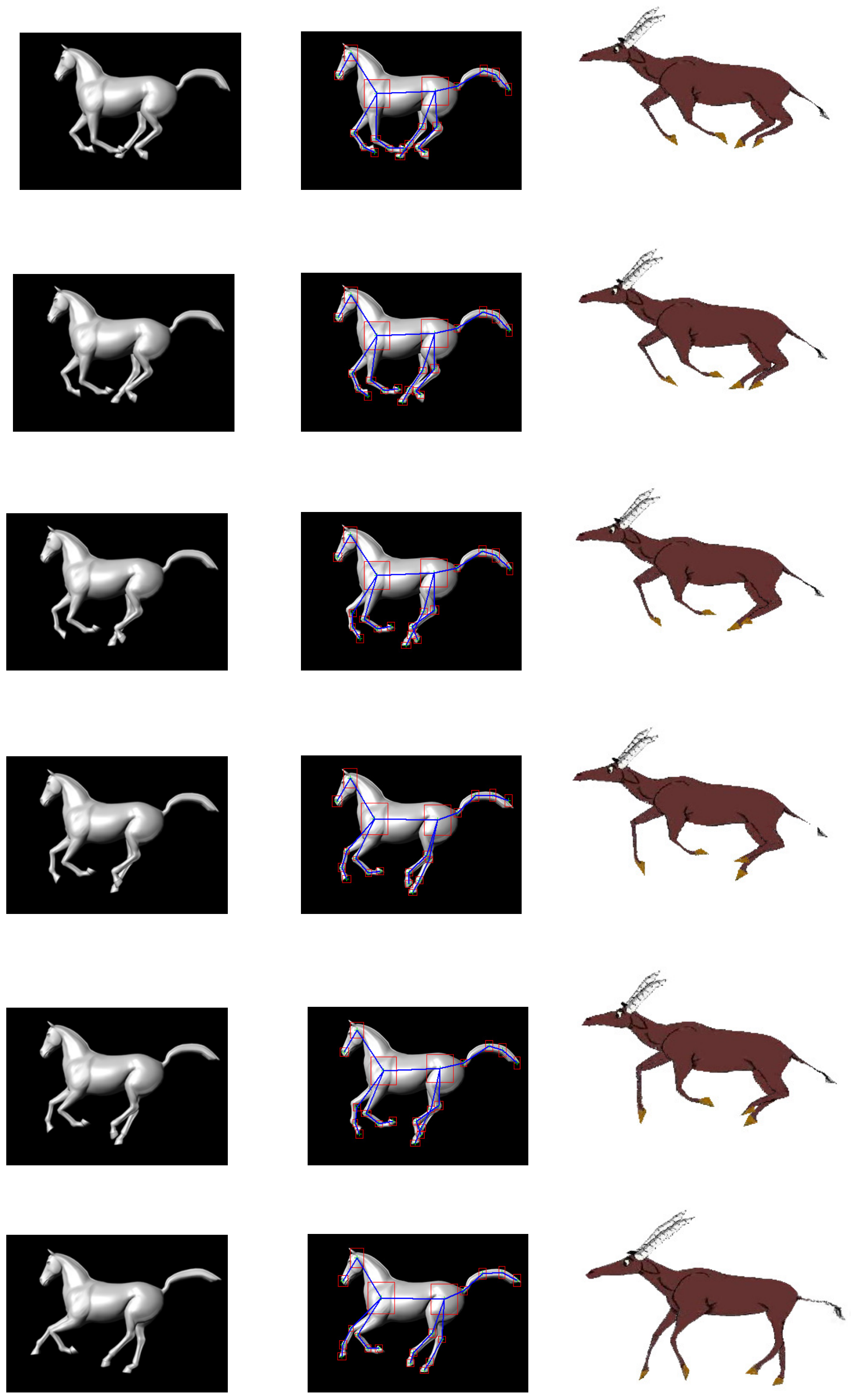
The second experiment is to track the joints of a 3D running horse (rendered as a 2D image sequence using Maya) and retarget it into a cartoon gazelle. Figure 4.9 illustrates the result. There are 32 frames in this sequence.











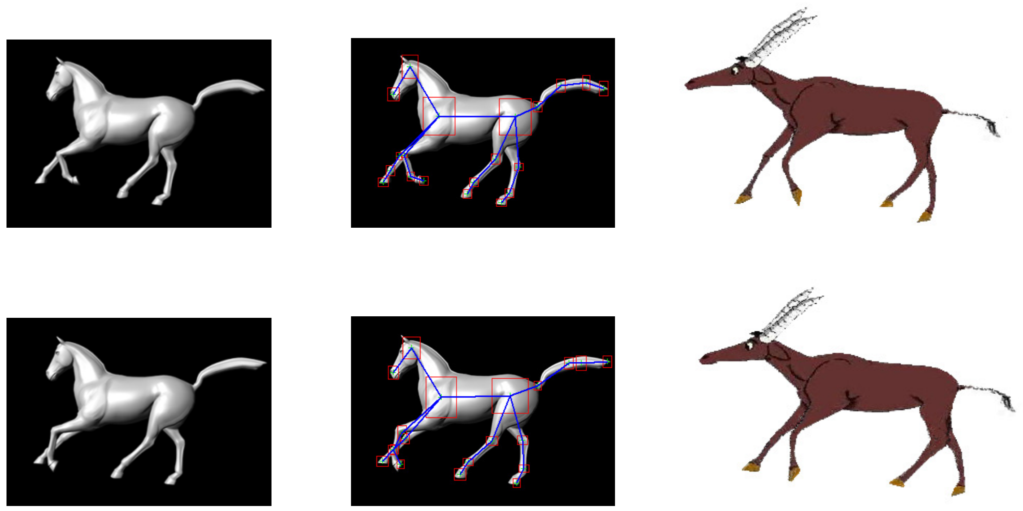


Figure 4.9: Joint tracking of a running horse in a rendered 3D animation image sequence and retargeting to a cartoon gazelle. (Left: original image sequence; Middle: joints tracking result; Right: retargeted new character)

From the results, the problem of self-occlusion is effectively solved. However, this experiment also shows a drawback of our technique. Since only a simple retargeting method is implemented and it hardly considered the problem of scaling and foot-ground constraints, the running cartoon gazelle does not look natural in some frames compared with the original running horse sequence. It reveals the direction to improve further in the future.

4.7 Summary

A skeleton-based 2D motion capture technique was described in this chapter. Once a skeleton is established in the first frame of a moving image sequence, all the joint positions from each subsequent image are tracked considering both visual and geometric features of the object. This process can be regarded as a mixed optimization with a template matching and the position prediction by Kalman filter.

Our 2D motion capture technique can be applied to various types of moving images, including 2D cartoon animation, videos and image sequences of rendered 3D animations. In the experiment, examples of all three types of image sequences have been demonstrated. Using the 2D shape deformation algorithm discussed in the previous chapter, retargeting a capture motion to a new 2D character is applied once the skeleton of the target character is obtained. Currently our implementation incorporates a simple retargeting of joint angles and skeleton length changes to the new character.

CHAPTER 5

IMPLEMENTATION AND APPLICATION OF “CARTOON ANIMATION FILTER”

Cartoon animation is the art of manipulating motion to emphasize the primary actions of character. Experienced animators can guide the viewers’ perceptions of the motion and literally bring it to life through a set of animation principles [Williams 2001, Isaac 2003]. These original principles, such as anticipation and follow-through to the motion with related squash and stretch, are still useful today since they can help animators to create more believable characters and motions. In 2005, Wang et al. [2005] presented a technique named “Cartoon Animation Filter”. It is a simple filter that takes an arbitrary motion signal as input and modulates it in such a way that the output motion is more ”alive”. The superiority of this technique lies in its simplicity and generality, which provides a good reference for our work. In this Chapter, our aim is to implement this “Cartoon Animation Filter” and applied it to the cartoon output from our system. It can add exaggerated effect to the final result and make the animation more vivid. According to the category of animation output, two different strategies are used in implementation. First of all, the working principle of “Cartoon Animation Filter” is briefly introduced.

5.1 Working Principle

In [Wang et al. 2005], the filter is described by the following mathematical formula:

$$x^*(t) = x(t) - \overline{x''}(t) \tag{5.1}$$

where $x(t)$ is the signal to be filtered, and $\overline{x''}(t)$ is a smoothed version of the second derivative of $x(t)$. This approach is equivalent to convolving the motion signal with an inverted Laplacian of a Gaussian (LoG) filter [Wang et al. 2005]:

$$x^*(t) = x(t) \otimes (-LoG) \tag{5.2}$$

Figure 5.1 is the graph of this inverted LoG filter. Because of the negative lobes that precede and follow the main positive lobe which is in the center of curve, the filter can add the effect of anticipation and follow-through to the final signal output [Wang et al. 2005].

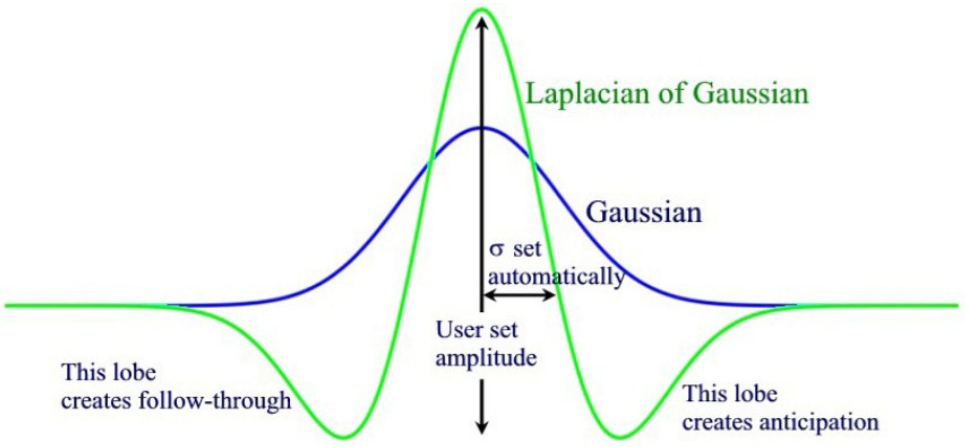


Figure 5.1 The cartoon animation filter [Wang et al. 2005]

In (5.1), the second derivative of the signal $x(t)$ is convolved with a Gaussian and then subtracted from the original motion signal:

$$\overline{x''}(t) = x''(t) \otimes Ae^{(-((t/\sigma)\pm\Delta t)^2)} \tag{5.3}$$

In (5.3), A is the strength of the filter and σ is the Gaussian standard deviation, the width of the smoothing kernel. The time shift Δt can be used to create the stretch and squash effects by applying the filter shifted forward or backward in time domain.

Figure 5.2 shows an example of applying the filter to a translational motion. In this example, a ball stands still, then moves with a constant speed to the right, then stops.

By applying the cartoon animation filter to the ball centroid without time shift Δt, the ball will move slightly to the left at the starting point to accumulate more acceleration, and overshoot at the stop point to release its inertia. So the anticipation and follow-through are added to the motion.

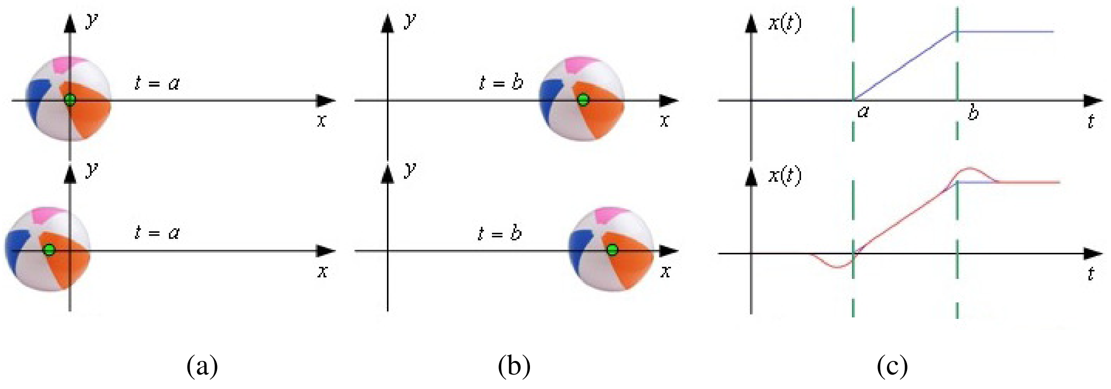


Figure 5.2 Anticipation and follow-through effect after filtering [Wang et al. 2005]. Up: original 1D translational motion on a ball. Down: Filtered motion on the ball centroid with follow-through and anticipation effects. (a) Starting point, (b) Stop point, (c) Graph of $x(t)$

5.2 Implementation and Application

In [Wang et al. 2005], they use (5.3) as the second derivative of the signal $x(t)$. The time shift Δt is used to create stretch and squash effects by applying the filter shifted forward or backward in time domain. However, in our system, the designed 2D shape deformation algorithm can naturally preserves the global area, which can exhibit squashing and stretching effects directly (Section 3.4.2). So a simplified version of (5.3) as the cartoon animation filter is designed. It can be expressed as (5.4).

$$\overline{x''}(t) = x''(t) \otimes Ae^{(-(t/\sigma)^2)} \qquad (5.4)$$

In (5.4), A is the strength of the filter and σ is the Gaussian standard deviation. In our system, σ is a fixed parameter and A is controlled by the user.

There are two kinds of animation output in our system. The first is the single frame of deformed character generated by sketches (Chapter 3). The second is the new 2D animation frames captured and retargeted from an existing image sequence (Chapter

4). According to the category of animation output, two different strategies are used to implement this cartoon animation filter.

5.2.1 Filtering Single Frame of Cartoon Character

For the single frame of deformed character generated by sketches, an interactive way is used to filter the output. Figure 5.3 illustrates this process.

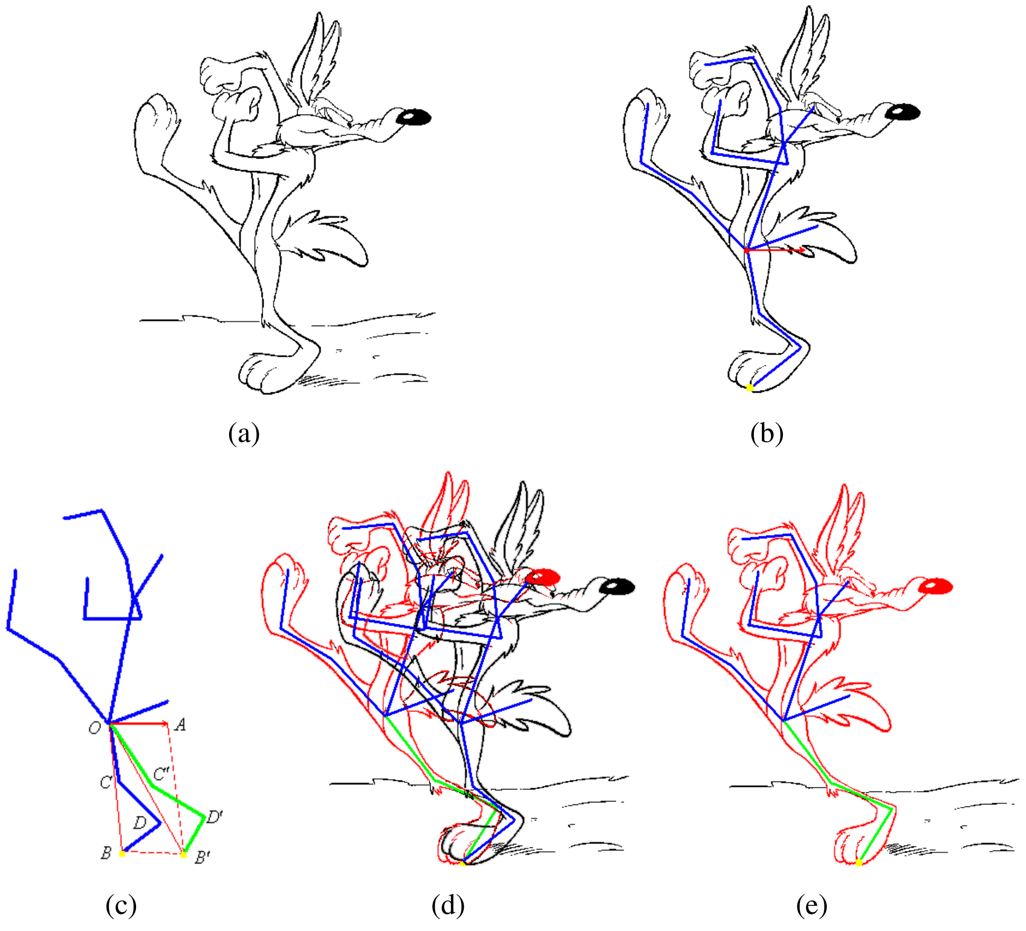


Figure 5.3 Filtering single frame of cartoon character. (a) Original image, (b) Original skeleton with acceleration and fixed vertex setting, (c) The deformation of skeleton**,** (d) The comparison of original image (black one) and filtered image (red one), (e) Filtered image

This example is a cartoon character at the starting point to run towards right. The cartoon animation filter effects to the skeleton root node of the character (centroid). The user draws an arrow-headed line (acceleration vector) on this root vertex to express the expected acceleration. The direction of vector indicates the acceleration orientation. The length indicates the magnitude of acceleration. Then the user locates the fixed vertex according to the constraints. In Figure 5.3(b), because of the ground-constraint, the fixed vertex is on the right foot (the yellow point).

Figure 5.3(c) illustrates the deformation of skeleton during filtering. Here, in this approach, only the skeleton segments connected with fixed vertex is deformed. So in this example, the deformed limb is the right leg, which consists of three skeleton segments: OC , CD and DB , from root node to fixed vertex. These skeleton segments are treated as a whole group and deformed together. These are:

$$\begin{aligned} &\overrightarrow{OA}+\overrightarrow{OB}=\overrightarrow{OB'} \\ &\overrightarrow{OB}=\overrightarrow{OC}+\overrightarrow{CD}+\overrightarrow{DB} \\ &\overrightarrow{OB'}=\overrightarrow{OC'}+\overrightarrow{CD'}+\overrightarrow{DB'} \end{aligned} \tag{5.5}$$

$\overrightarrow{OA}$ is the acceleration. The transformation of $\overrightarrow{OB}$ ($\overrightarrow{OC}+\overrightarrow{CD}+\overrightarrow{DB}$) can be split to two stages. The first stage is to rotate $\overrightarrow{OB}$ to the orientation of $\overrightarrow{OB'}$. The second stage is to scale $\overrightarrow{OB}$ along the orientation of $\overrightarrow{OB'}$. After $\overrightarrow{OB}$ is transformed, the coordinates of C', D' are also determined. Finally, since B is the fixed point, the deformed character should be translated to make B' return to the position of B (Figure 5.3(d)). All steps above are automatic except the acceleration and fixed vertex setting. During filtering, though the length of skeleton segments may be changed, the global area is still preserved. Figure 5.3(e) gives the result after filtering. It can be seen that the right leg is stretched and the global area is almost invariable.

Figure 5.4 gives another example of filtering a single frame cartoon character. It is a cartoon man starting to jump. The expected acceleration is put on the root node of skeleton. There are two fixed points in this character. Both left and right feet are the constraints with the ground. From the filtering result in Figure 5.4(c), it can be seen that the legs are benter and squashed which can accumulate more energy.

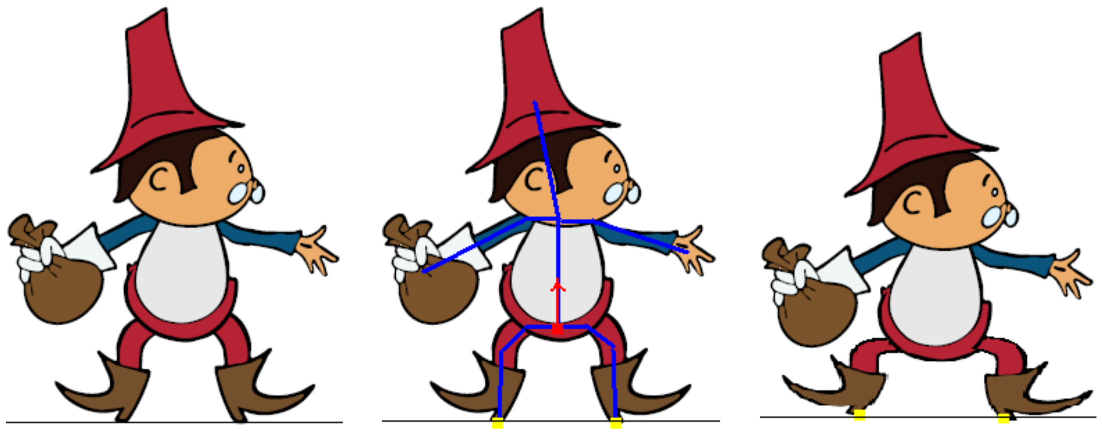


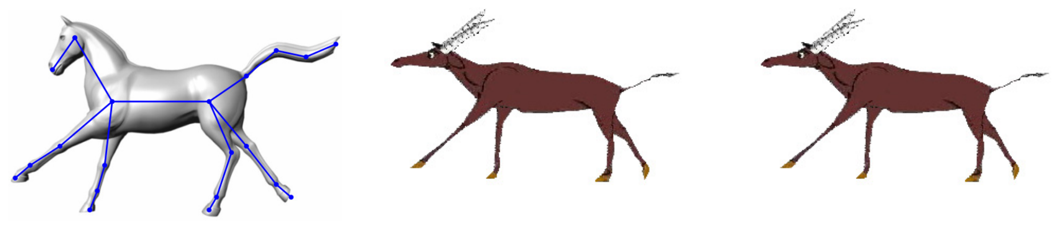
Figure 5.4 Filtering a jumping cartoon man. (a) Original image, (b) Original skeleton with acceleration and fixed vertex setting, (c) Filtered image

5.2.2 Filtering Motion Capture Data

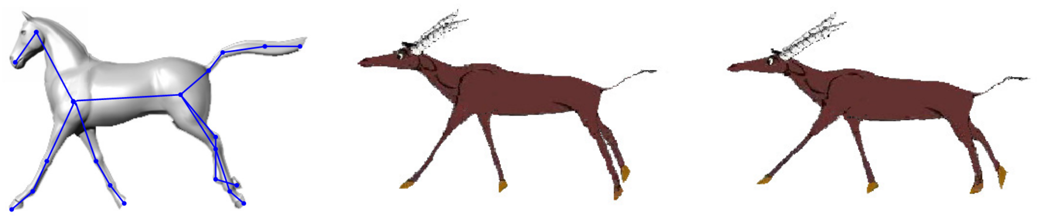
For the motion capture data from an input image sequence, (5.1) and (5.4) are used to filter the motion data for each joint of the character. According to the descriptions in Chapter 4, there are two types of transformation for a skeleton segment: rotation and scaling. So two motion parameters: the length and orientation angle of a skeletal segment are processed separately as input signals during filtering. Following formula is used to compute the second derivative of them:

$$l'_{m,t} = (l_{m,t} - l_{m,t-1}) / \tau \qquad l''_{m,t} = (l'_{m,t} - l'_{m,t-1}) / \tau \tag{5.6}$$
$$\alpha'_{m,t} = (\alpha_{m,t} - \alpha_{m,t-1}) / \tau \qquad \alpha''_{m,t} = (\alpha'_{m,t} - \alpha'_{m,t-1}) / \tau$$

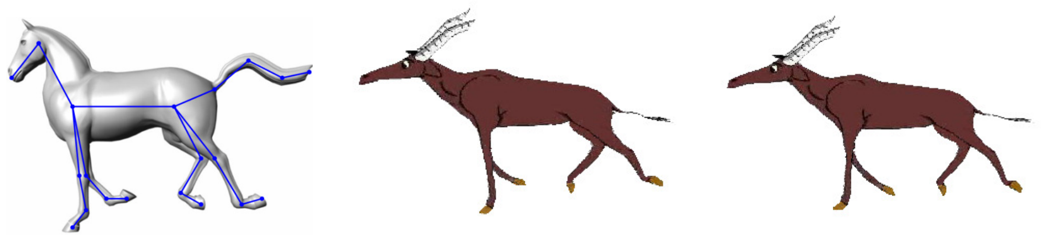
Where $l_{m,t}$, $\alpha_{m,t}$ represent the length and orientation angle of the *m*-th skeleton segment at frame *t*. τ is the time interval between adjacent frames. Generally, as there is no time interval for the first frame, the filtering is started from the second frame.



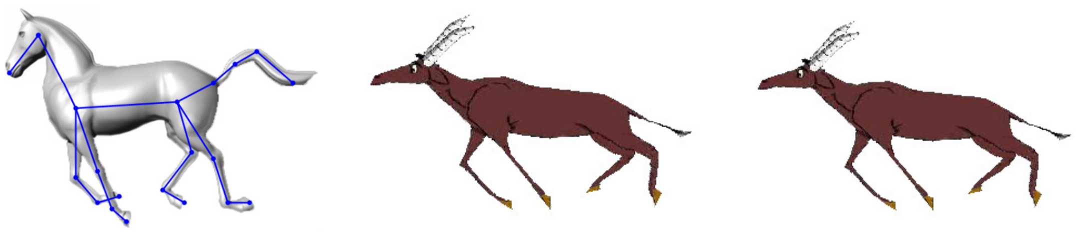
frame 2



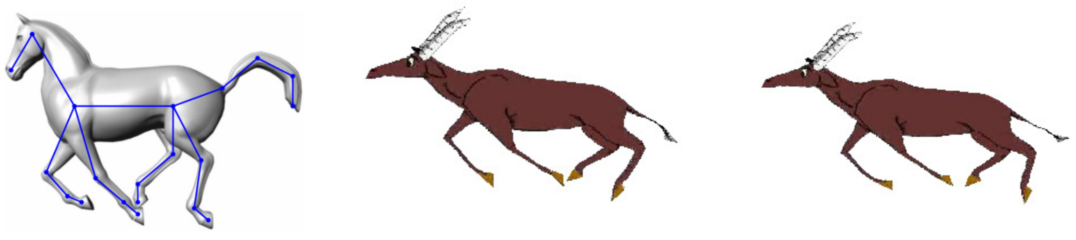
frame 6



frame 10



frame 14



frame 18

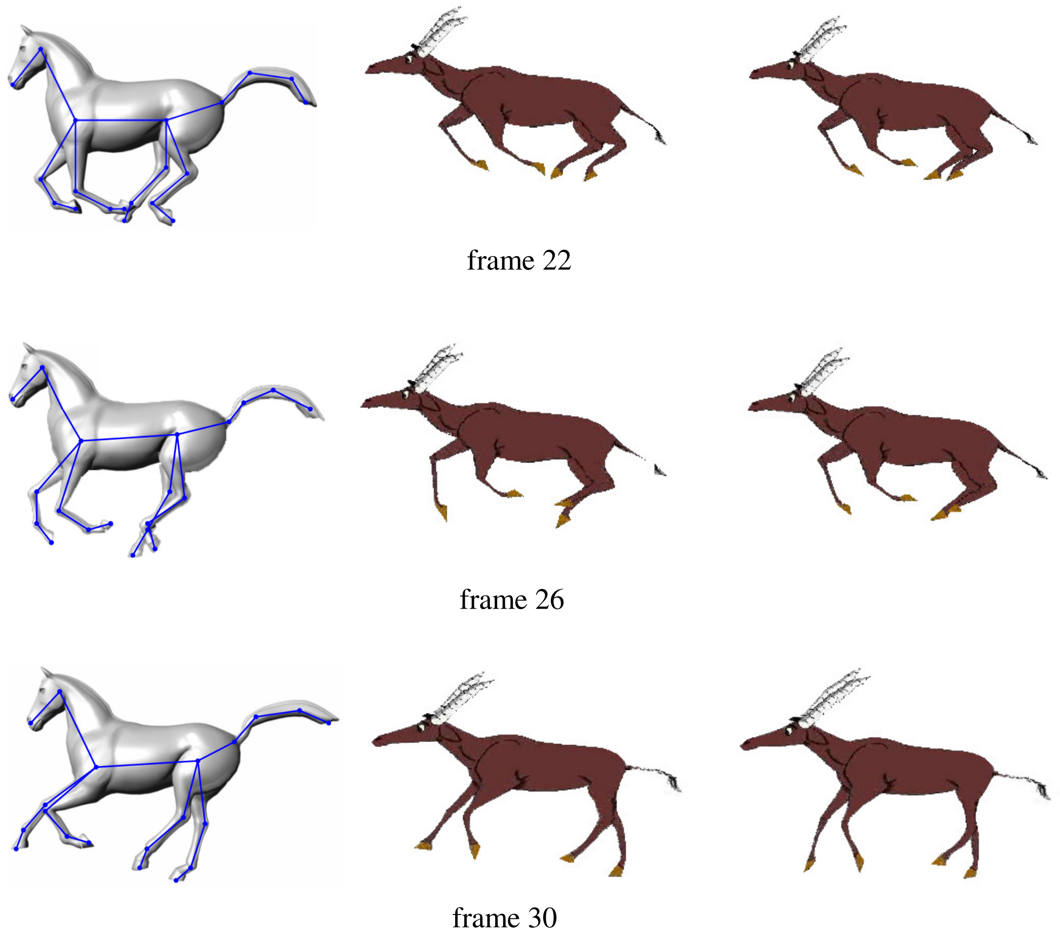


Figure 5.5 Filtering motion capture data from image sequence. (Left: original image sequence of running horse; Middle: retargeted running cartoon gazelle without filtering; Right: retargeting result after filtering)

Figure 5.5 illustrates an example of applying the cartoon animation filter to a motion capture data. It is based on the second experiment in Chapter 4 (Section 4.6). To show the difference of retargeting result before filtering and after filtering, 8 frames are chosen from the sequence with 32 frames. The strength of the filter (A) is 1.5. Though the change of skeleton segment bend angle and length are relatively small, the squash effect in frame 22 and 26 can still be seen after filtering.

5.3 Summary

A technique named “Cartoon Animation Filter” is implemented and applied in this chapter. It is a simple filter that takes an arbitrary motion signal as input and make the output motion more ”alive”. Its working principle is briefly introduced first. Then, according to the practical requirement of our animation system, a simplified version of its mathematic expression is designed and applied to two different types of input: single frame of cartoon character and motion capture data from image sequence. The experiments show that the filter can add anticipation and follow-through to the cartoon output with related squash and stretch effect.

CHAPTER 6

SYSTEM INTERFACE AND EVALUATION

From Chapter 3 to Chapter 5, the sketch-based skeleton driven 2D animation, motion capture and the application of "Cartoon Animation Filter" are introduced respectively**.** To implement these techniques, a complete and easy-to-use 2D animation system has been developed. In this chapter, the structure and interface of the system will be introduced, including the interactive operations and auxiliary functions. In the end, through practical animators testing, the system will be also evaluated and discussed.

6.1 System Structure

There are three modules in this prototype system: Module 1 is sketch-based skeleton driven 2D animation. Module 2 is the motion capture and retargeting. Module 3 is the cartoon animation filter. Figure 6.1 illustrates its pipeline structure.

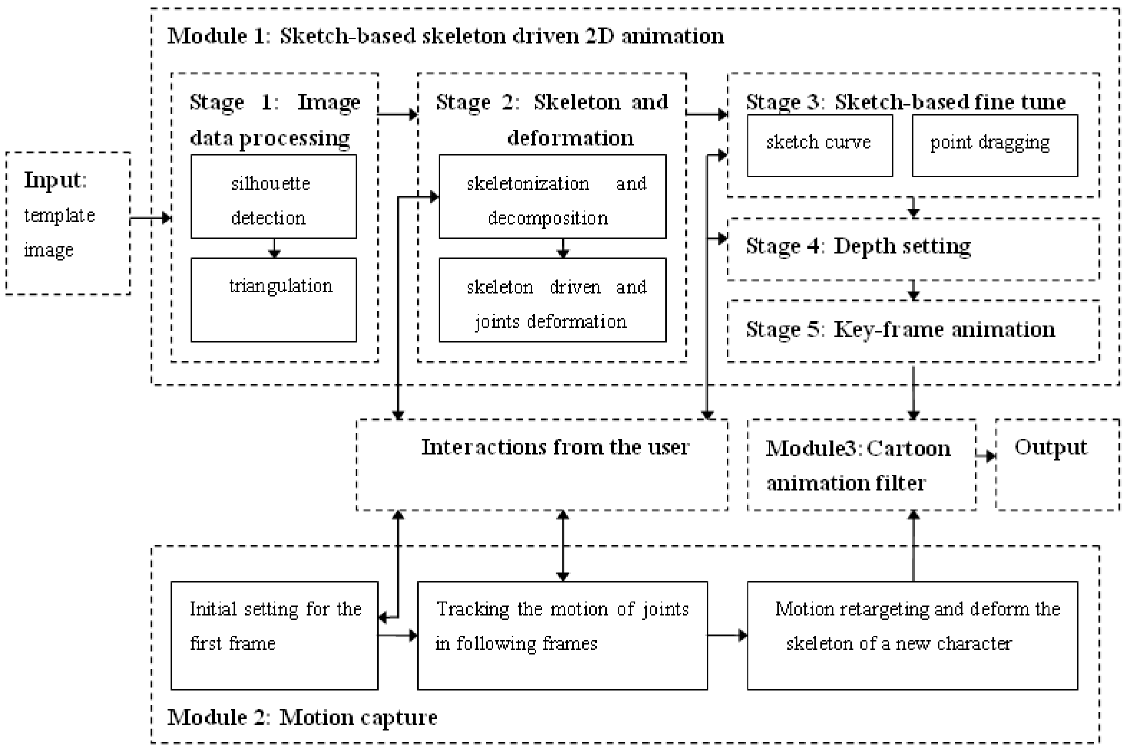

Module 1: Sketch-based skeleton driven 2D animation
Input: template image
Stage 1: Image data processing
silhouette detection
triangulation
Stage 2: Skeleton and deformation
skeletonization and decomposition
skeleton driven and joints deformation
Stage 3: Sketch-based fine tune
sketch curve
point dragging
Stage 4: Depth setting
Stage 5: Key-frame animation
Interactions from the user
Module3: Cartoon animation filter
Output
Initial setting for the first frame
Tracking the motion of joints in following frames
Motion retargeting and deform the skeleton of a new character
Module 2: Motion capture

Figure 6.1 Pipeline structure of system

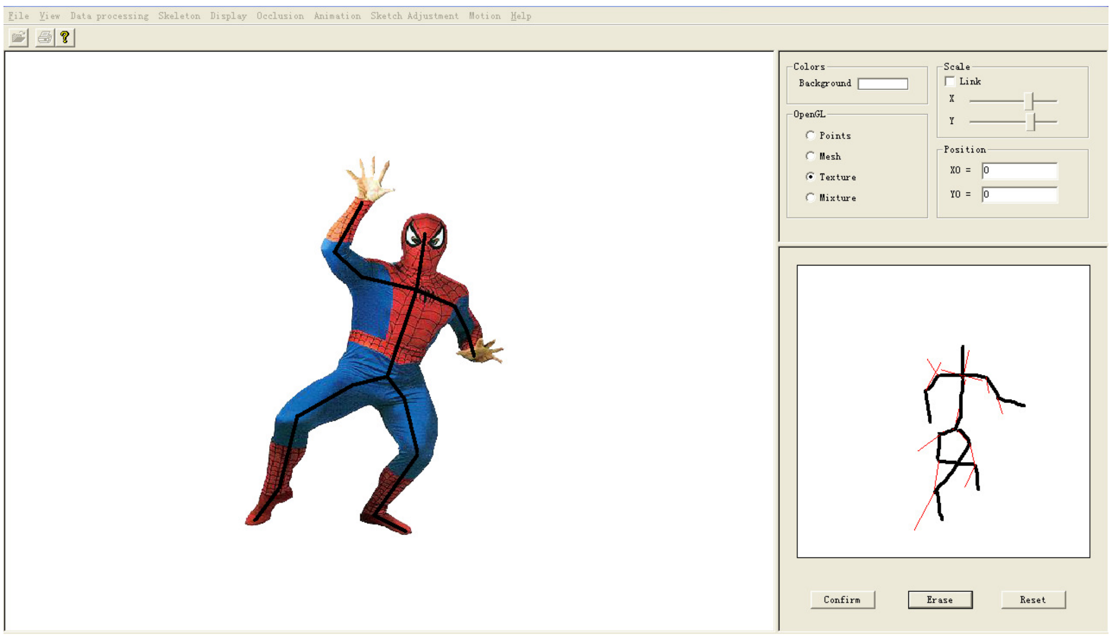
In module 1, the user first imports a 2D character as a template model, which can be represented by a BMP/JPEG image or vector graphics. For BMP/JPEG images, the backgrounds are removed manually and the silhouette is automatically detected with the marching squares algorithm. According to the automatically generated curve skeleton (Section 3.2), after discrete sampling inside the silhouette and triangulation in Stage 1 (Section 3.2), the user places skeletal joints at mesh vertices to build a 2D skeleton. Then the system will attach all the vertices to corresponding skeleton segments through decomposition (Section 3.3). In Stage 2, the user draws the sketch skeleton with different poses in the interface. Then the system will generate a new deformed figure automatically according to the sketched skeleton (Section 3.4). When an overlap happens in the deformation, a dynamic depth setting is needed in Stage 4 (Section 3.5). In the end, the user selects three sub key frames. The system will generate the final key-frame animation through a mixed interpolation (Section 3.6).

For motion capture, according to the steps of stage 1 and stage 2 in module 1, the user first processes the skeletonization and decomposition of the target character. For a given image sequence or video, the user locates all the joint regions of the original character in the first frame (Section 4.3). Then the motion of the joints will be tracked automatically in the following frames (Section 4.4). Finally the motion is retargeted and the predefined skeleton of target character is deformed to generate cartoon animation (Section 4.5).

In module 3, the cartoon animation filter is applied to add exaggerated effects such as anticipation and follow-through to the motion with related squash and stretch. It can process two types of input. One is the single frame of cartoon character from module 1 (Section 5.2.1), the other is the motion capture data from module 2 (Section 5.2.2).

6.2 Interface and Interaction

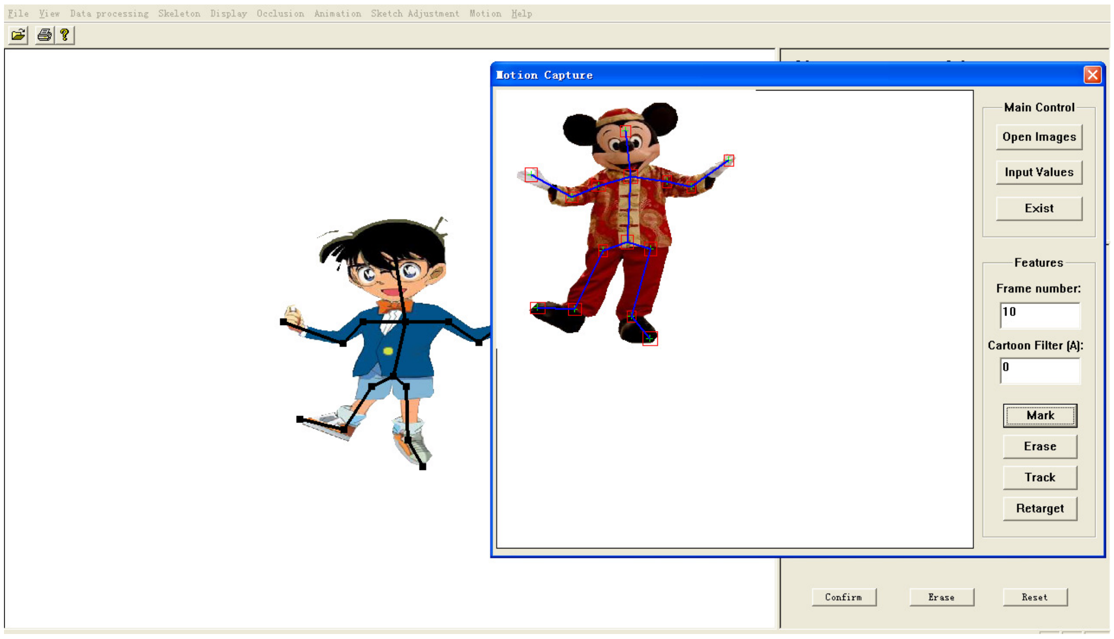
Figure 6.2 gives a screenshot of our prototype system interface from module 1 to module 3 respectively.


File View Data processing Skeleton Display Occlusion Animation Sketch Adjustment Motion Help
Colors
Background
OpenGL
Points
Mesh
Texture
Mixture
Scale
Link
X
Y
Position
Confirm
Erase
Reset

OpenGL: ( ) Points ( ) Mesh [✓ Texture] ( ) Mixture

Position: X0 = 0, Y0 = 0

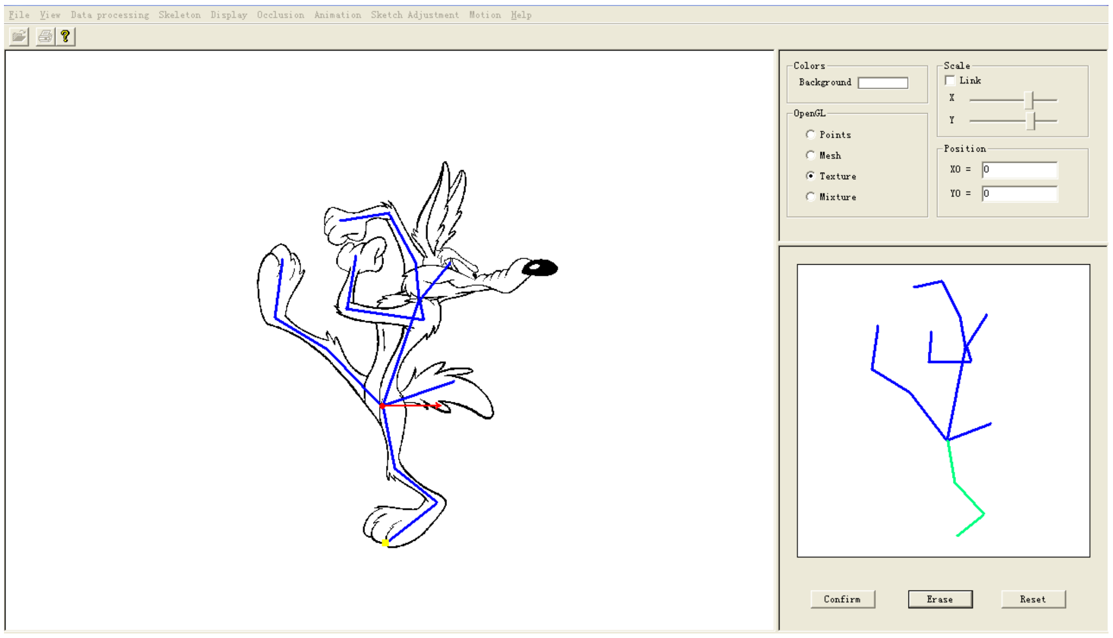
(a)



Frame number: 10

Cartoon Filter (A): 0

(b)


File View Data processing Skeleton Display Occlusion Animation Sketch Adjustment Motion Help
Colors
Background
Scale
Link
X
Y
OpenGL
Points
Mesh
Texture
Mixture
Position
Confirm
Erase
Reset

OpenGL: Points Mesh [✓ Texture] Mixture

X0 = 0

Y0 = 0

(c)

Figure 6.2 Interface of the prototype system. (a) Module 1: sketch-based skeleton driven 2D animation, (b) Module 2: motion capture for 2D character, (c) Module 3: cartoon animation filter

Four main interactions are manipulated through this interface, as discussed in the following:

a. Skeleton sketching: To generate new poses for a character, the user sketches the skeleton at the bottom right window as shown in Figure 6.2(a). The black lines denote the drawn skeleton segments and red lines indicate the direction and length of the corresponding original skeleton segments of the template model. The differences of length and direction between the pairing black and red lines indicate the amount of rotation and scaling in the new pose.

b. Skeleton generation: The user can place joints guided by the generated curve skeleton to construct the animation skeleton for the template model.

c. Initial setup for motion capture: When tracking the motion of a character from an image sequence, the user needs locating all the joints in the first frame according to the extracted curve skeleton. Using the mouse the user specifies the size of rectangle regions whose centres represent the joint locations (Figure 6.2(b)).

d. Acceleration and fixed vertex setting: When processing single frame of cartoon character with cartoon animation filter, the user first draws a red arrowheaded line (vector) on the skeleton root to express the expected acceleration (Figure 6.2(c)). The direction of vector indicates the acceleration orientation. The length indicates the magnitude of acceleration. Then the user locates the fixed vertex according to the constraints. The skeleton of 2D character is displayed at the bottom right window as shown in Figure 6.2(c). The green lines illustrate the skeleton segments which will be deformed. The blue lines illustrate the fixed skeleton segments.

6.3 Auxiliary Functions

To cope with practical problems in cartoon production, our system also contains some important auxiliary functions for better interaction

a. Decomposition correction

In some cases, our automatic decomposition algorithm may map some vertexes to wrong skeleton segments since it is based on the shortest distance classification. So to fine tune the final decomposition result, interactive manipulations from animators are also needed. Figure 6.3 illustrates such a case of decomposition correction.

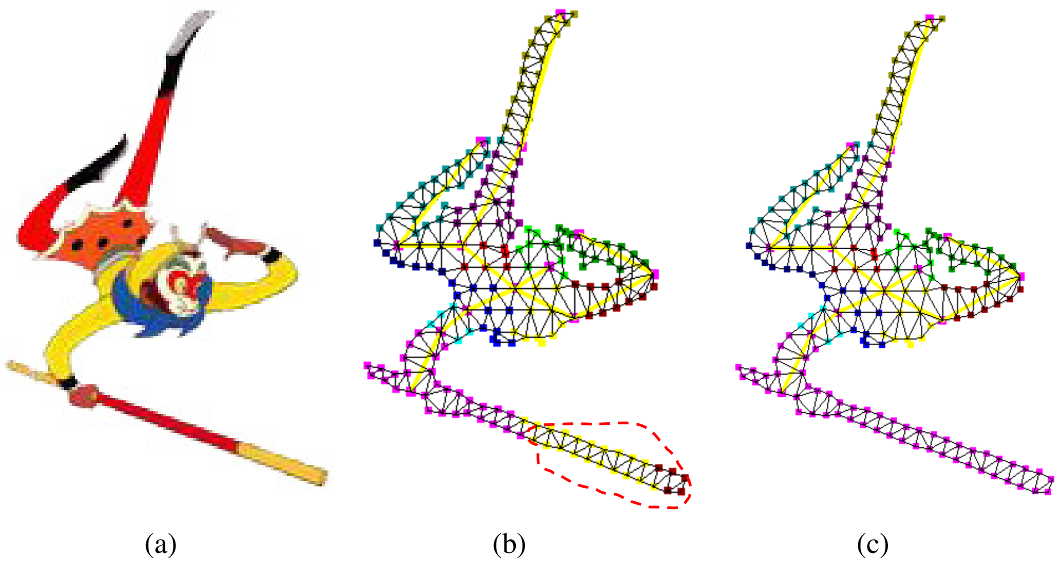


Figure 6.3 Decomposition correction. (a) Original image, (b) Original decomposition result before correction, (c) Decomposition result after correction

In Figure 6.3(b), our decomposition algorithm classifies the vertexes on the stick in left hand into three classes. In fact, the stick is totally rigid during deformation and should be all mapped to the skeleton segment of the left hand. In correction, the user draws a closed curve (the red dashed line) by mouse to enclose the vertices which need adjustment. Then the user selects a skeleton segment to which all the enclosed vertices will be attached. Figure 6.3(c) shows the result after decomposition correction.

b. Depth adjustment

When self-intersection occurs during deformation/animation, the user needs to set the depth values to avoid this problem. For example, the hand of a character can be either at the front or back of the body, but it should not be of the same depth as the body. During deformation, the system monitors the mesh for self-intersection and set an appropriate depth order to the overlapping parts. When the user clicks any vertex in an overlapping part, all the vertices in this decomposed region will have the same depth value as the clicked one.

c. Fine tuning by sketches

In Module 1, the system also allows the user to fine tune the final deformed mesh by sketch curve and point dragging to perform some special deformations as described in Section 3.5.

d. Key-frame setting

During in-betweens generation, the user can set the total number of frames and choose any of them as the key pose.

e. Joint tracking correction

In Module 2, the frame-by-frame tracking method is used to capture the motion of character joints. However, since it is inherently subject to error accumulation, the accuracy is limited to a small number of frames. One effective way to solve this problem is to divide a large image sequence into a number of segments which consist of fewer frames, and correct the tracking error for the first frame in each segment. Therefore, our system allows the user interactively adjust the tracking result at any frame if appropriate.

6.4 Evaluation and Discussion

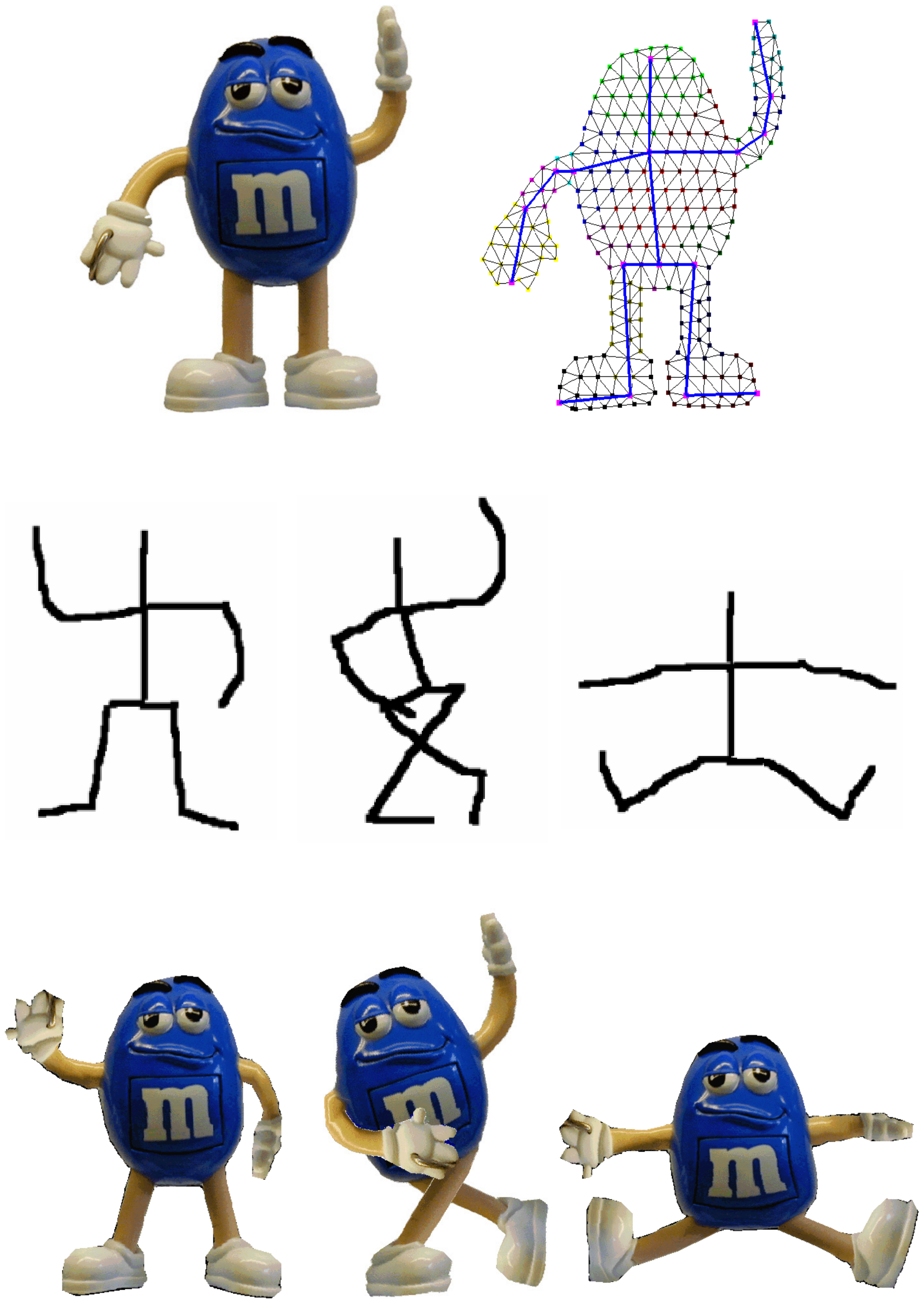
A group of three animators is invited to test our prototype system. This test focus on the Module 1 in our system: sketch-based 2D animation, which is most related to the animator’s work. The aim is to evaluate the visual quality and efficiency of our system in cartoon animation production.

6.4.1 Testing Procedures

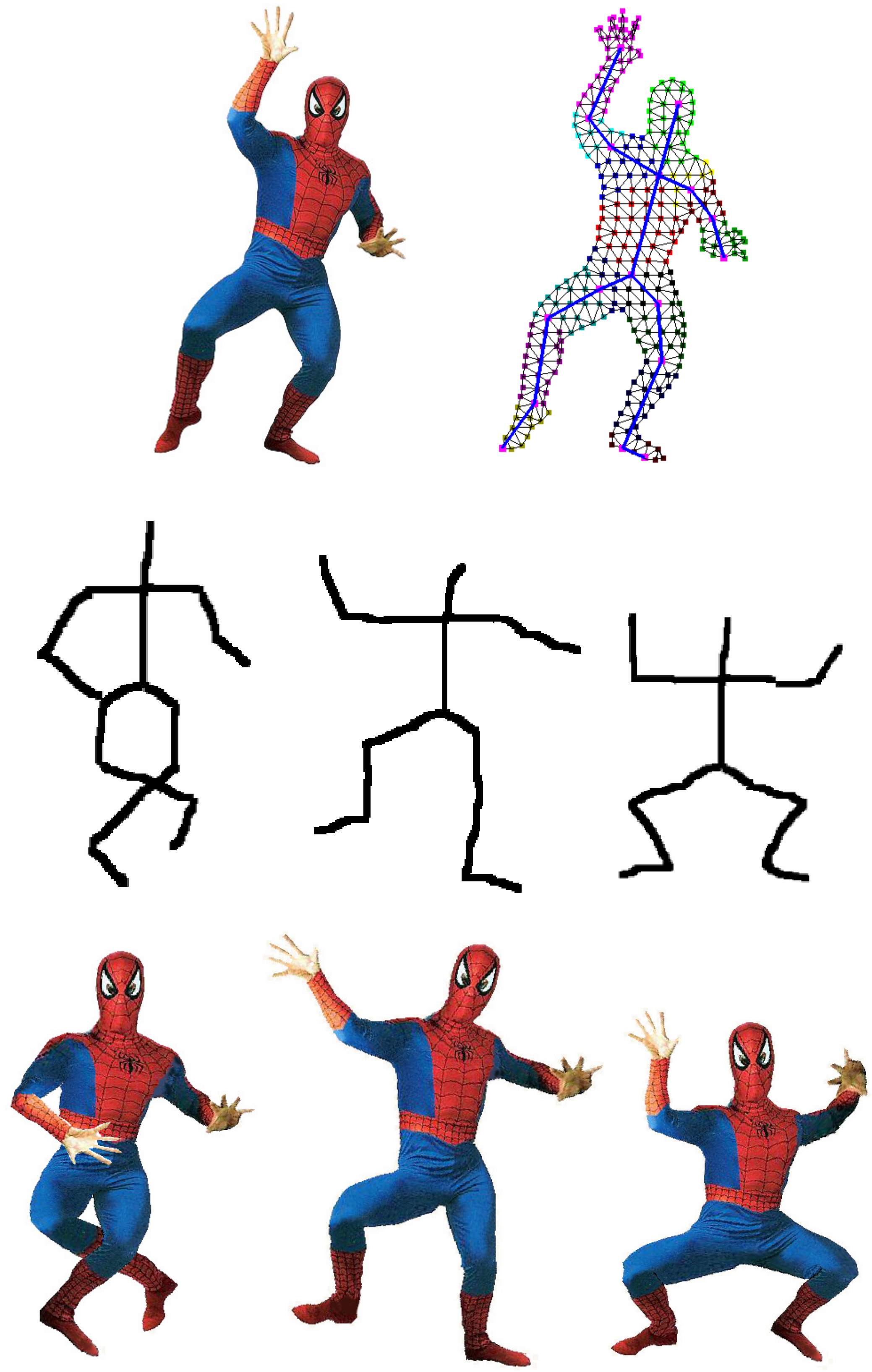
The testing comprised four steps: *introduction, system demonstration, performance testing*, and *user feedback*. First, animators are given a brief introduction (5 minutes) about the research object and the main functions of this animation production system. Then, a demonstration (15 minutes) was provided on how to use this sketching interface to draw sketch skeletons and generate key-frame animations. During the demonstration, the animators were told how to manipulate the skeleton location, depth adjustment, sketch-based fine tuning and key-frame setting. After this step, the animators are invited to run the software for no more than 15 minutes to be familiar with it. Then, the animators were requested to generate short animations for two cartoon characters by sketches. All the original cartoon images were acquired from the Internet. Animators just need to sketch three key postures, and the system will deform the character automatically. Then animators finish the depth setting and fine tuning. The step of performance testing is usually no more than 30 minutes. Animators can ask questions anytime if they need help. Finally, after the performance testing, animators were asked to give some comments, suggestions and fill in a questionnaire (Appendix A) to answer some routine questions.

6.4.2 Testing Result

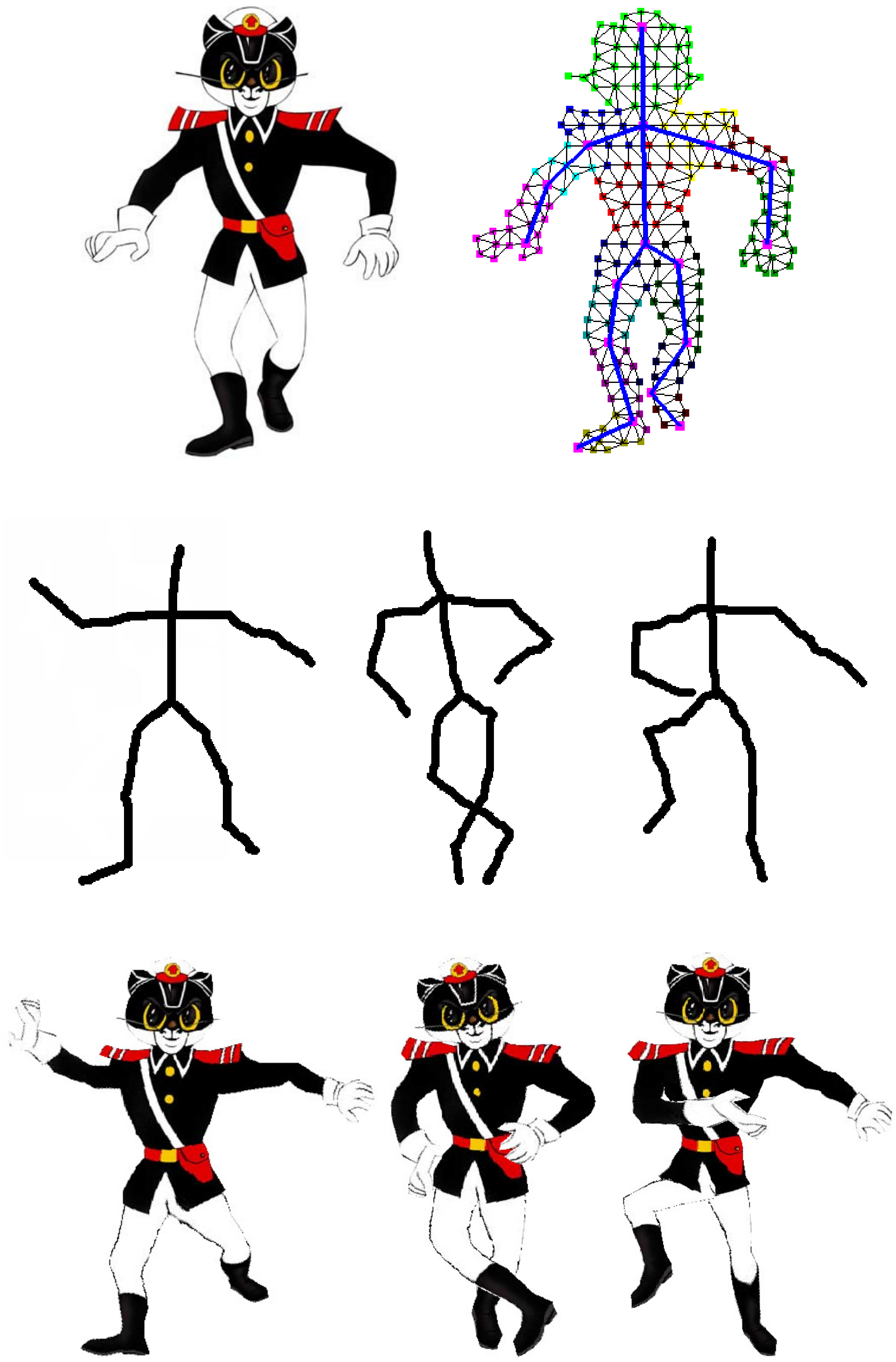
Figure 6.4 gives the sketch skeleton and the corresponding deformed object as the key frames for six cartoon characters. All the original figures are the popular cartoon character which can be downloaded from Internet. The pose and sketch are provided by animators. For each figure, the first row is the original template model and the decomposition results with a skeleton. The next is the sketch skeleton drawn by the animators and the corresponding deformed character as key frames.



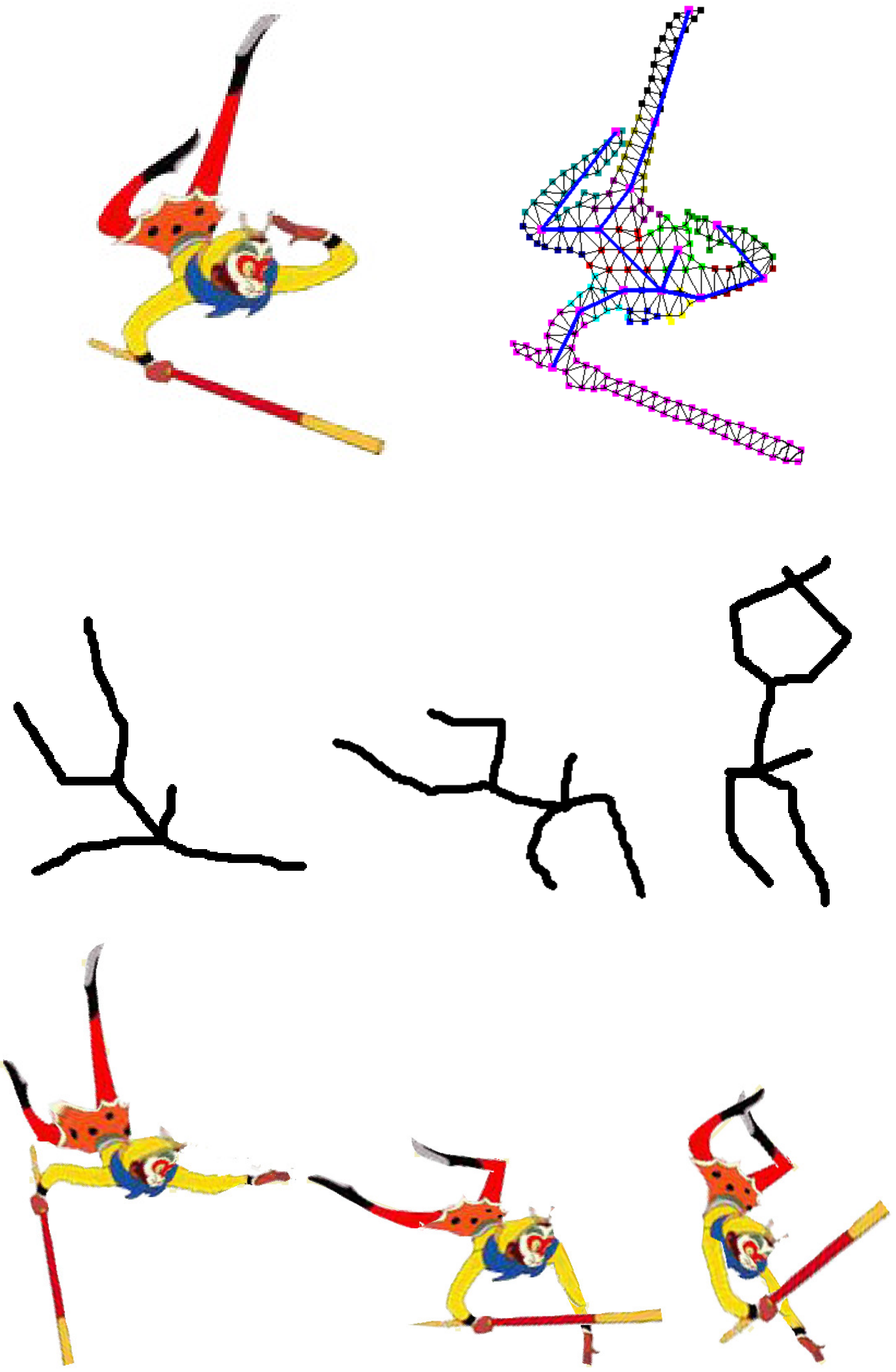
(a)

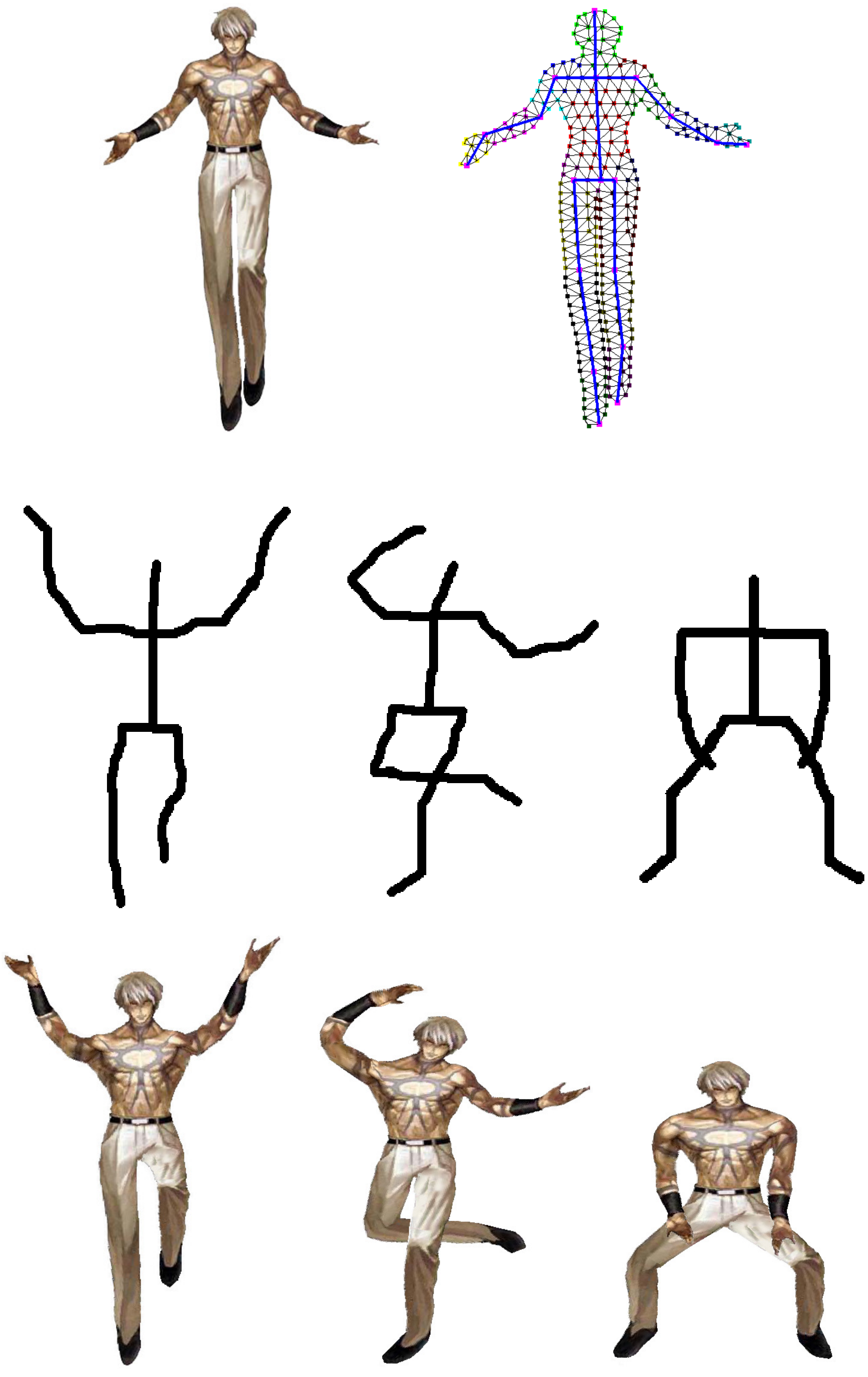


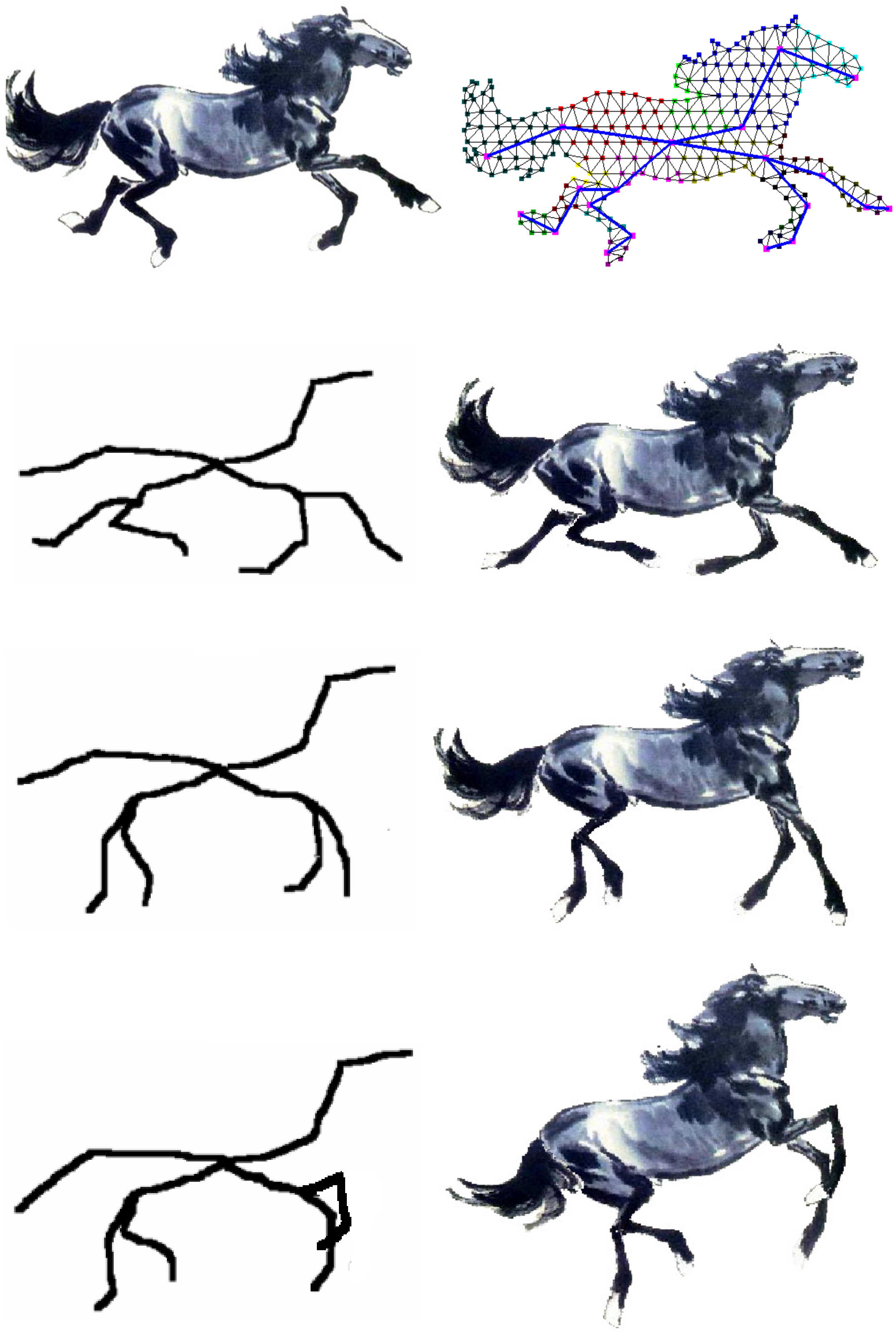
(b)



(c)


(d)


(e)



(f)

Figure 6.4 Six groups of cartoon characters animated by our system, from top to bottom: original template model, decomposition results with skeleton, sketch skeleton and corresponding deformed figures. (a) Mm, (b) Spiderman (www.gzmiqi.com/cp.php?nowmenuid=87533), (c) Black cat sergeant (www.cctv.com/kids/special/C15750/04/index.shtml), (d) Monkey king (www.js. xinhuanet.com/10/content_4256139.htm), (e) Sasuki (http://bbs.qlwb.com.cn/ uploadfile/2005-3/200532518920291.jpg), (Copyright: SEGA) (f) Running horse (http://www.cjwy.net/Photo/UploadPhotos/200605/20060518115116155.jpg)

6.4.3 Feedback

In the user feedback, the animators were first asked to give their overall comments on our sketch-based skeleton driven 2D animation technique, followed by some routine questions (Appendix A) regarding the interface design and system functionalities. In general, the consensus among the animators was that the sketching-based cartoon process is easy and fun. They found our technique to be much more efficient than the current practice adopted in many commercial cartoon production houses, as sketching a skeleton is much faster than drawing a whole frame, and that our design is consistent with their animation practice. To create a key-frame, often the animator would first sketch a stick figure (i.e. the skeleton) and then overlay the body shape on top guided by the stick figure. This process is called the *deep structure*. Sketching the skeleton alone relieves them from the time-consuming part, i.e. to draw the whole character body. In terms of the interface design, animators enjoy its clean and graphical input. Considering the further developments, one animator suggested that it is better to add a time line in the interface, and use a sketch pad (tablet) as an input device so that the user can sketch directly with a pen.

6.5 Summary

In this chapter, our developed animation system is described, including the pipeline structure, the interface and the auxiliary function. There are three modules in our prototype system. They are: sketch-based skeleton driven 2D animation, motion capture and retargeting, and the cartoon animation filter. There are five interactive operations through the interface, which consists of: skeleton sketching, skeleton generation, depth adjustment and fine tuning, initial setup for motion capture, and acceleration and fixed vertex setting. In the end, through practical system testing and evaluation from animators, our system is found to be more efficient. The possible areas of further development in future are also pointed out.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Sketch-based animation is one of the most popular research fields which have attracted a great deal of attention from the computer graphics research community. Many papers have been published in this area and some techniques have even been developed into commercial software. However, compared with sketch-based 3D animation, the area of 2D animation has not profited much from these achievements.

2D animation is a long sought research topic both in art and computer graphics. The generation of key-frames and in-between frames are the two most important and labour intensive steps in 2D animation production. Although some software tools, e.g. Animo, Toon Boom [2008], have been helpful in generating in-between frames, they often lack of 'personality' in comparison with those created by a human in-betweener. Often, the software-generated in-betweens have to be tweaked by the animator to give back the 'personality' to the animation. So in practice, up until now, most professional cartoon studios still produce huge amounts of animation (key-frames and in-betweens) manually.

In this thesis, motivated by the skeleton-driven 3D animation techniques, sketch-based animation techniques, and recent progress in 2D deformations, a novel technique is presented with the aim to improving the degree of automation for the production of 2D animation without sacrificing the quality. Our technique consists of three parts, Part 1: sketch-based 2D animation generation (Chapter 3); Part 2: motion capture and retargeting (Chapter 4) and Part 3: cartoon animation filter (Chapter 5). Part 1 can be used independently to create an animation sequence. If it is combined with Part 2, one can easily reuse the ‘motion’ of an existing animation sequence and apply it to a different character. Part 3 is used to add exaggerated effect to the final animation output.

The first part is concerned with the fast production of 2D character animation by sketching only the skeletons. Given an original image of a character and a sketched skeleton sequence, our prototype system generated deformed characters with different poses automatically. In comparison with the conventional practice where all key-frames are hand-produced, our method requires much less user input without depriving the animator of controlling of the artistic quality. It is much faster and less labour-intensive to create a sequence of 2D animation. The preliminary user tests with our prototype system support this assertion (seen Section 6.4.3).

Our first theoretical contribution in this part is a variable-length needle model, which can mostly preserves the global area of a character during animation. This is an essential property for quality 2D animation. Another contribution is the skeleton driven + nonlinear least squares optimization algorithm. The key idea is to divide the 2D shape deformation into two components: skeleton driven deformation and nonlinear deformation in the joint areas. For the interest of computational efficiency, the former is treated simply as a linear transformation, where the latter is solved by nonlinear least squares optimization. To ensure realistic deformation, properties such as boundary feature, local area preservation and interior smoothness are maximized during animation. Compared with other approaches, our algorithm is more efficient and able to produce plausible deformation results.

The second part is concerned with the development of a skeleton-based 2D motion capture technique. As our cartoon production technique is skeleton-based, naturally the idea of motion capture is borrowed from 3D animation to capture the ‘motion’ of a 2D animation sequence, and retarget the captured motion to a different character with the deformation algorithm described in Part 1. In theory, a feature region based tracking method is presented, commonly used in computer vision, to extract the motion of 2D objects from an image sequence. Once a skeleton is established in the first frame of a moving image sequence, all the joint positions are tracked in each subsequent image considering both geometric and visual features of the images. This 2D motion capture technique can be applied to various types of moving images, including 2D cartoon animation, videos and image sequences of rendered 3D animations. In Chapter 4, examples of all three types of image sequences in experiments have been illustrated. Retargeting a capture motion to a new 2D character is applied once the skeleton of the target character is obtained.

The third part of our work is the implementation and application of the “cartoon animation filter”, which is a simple filter that takes an arbitrary motion signal as input and makes the output motion more ”alive”. According to the requirement of our animation system, a simplified version of this filter is used and interactive operation is added to it. The modified filter can be applied to two different types of animation output in our system: single frame of cartoon character and motion capture data from image sequence. From the experimental results, it can add anticipation and follow-through to the motion with appropriate squash and stretch effect.

In the end, the developed software package is introduced. It is an easy-to-use prototype 2D animation system which implemented all the algorithms and functions mentioned above. Through the interface, the user can conduct quick cartoon production by sketching or from existing image sequence. Through a practical test and evaluation from animators, the efficiency and advantages of this system are proved.

7.2 Limitations and Future Work

Despite the advantages of the current work, our research experience also reveals several limitations which need further technical improvement in the future.

(1) The core problem relates to the texture information of the template image. Because there is no 3D information of a 2D character, large pose change can result in loss of correct texture for subsequent frames. Although some research in matting, image completion and texture synthesis [Efros and Leung 1999, Sun et al. 2004, Drori et al. 2003] has attempted to solve this problem, it is still an open problem for all 2D deformation techniques. I plan to use image merging techniques [Irani and Peleg 1991] to solve this problem in the future. Another alternative is to use multi-template images, one for each typical pose. The user then selects the most similar image to deform.

(2) Another limitation is the error accumulation in tracking. Currently the tracking error is corrected at the first frame of each sequence segment. I plan to use more robust tracking approach or tracking feature such as Hidden Markov Model [Cappé et al. 2005] to solve this problem.

(3) Our decomposition method also will be improved. Attaching each vertex to only one skeleton segment is computationally efficient, but can be a little too rigid. Multi-weight skinning may help reduce shape distortion in animation.

(4) In our technique, the triangulation method is based on uniform sampling. Non-uniform sampling will generate more vertices in character's joints area than elsewhere. This will achieve more optimal deformation results in stage two.

(5) There is a large space for further research in retargeting. Our current simple approach is only to demonstrate our motion capture method. It would be desirable to incorporate a 3D animation technique (e.g. [Gleicher 1998]) to treat retargeting as a space-time optimization problem. The motion editing techniques [Gleicher 2001] developed for 3D motions are also relevant to our future work.

(6) In system development, the animators who tested our system also gave some good comments. One is the time line. Currently, our system does not provide the time line. Another suggestion is to use a sketch pad (tablet) as an input device so that the user can sketch directly with a pen. These functions will be added in future.

APPENDIX A

QUESTIONNAIRE

QUESTIONNAIRE

This questionnaire is designed to seek the requirements and comments for developing a “sketch-based skeleton driven 2D animation system”, which can transfer the sketch skeleton from animators or novices into deformed key-frames and generate the animation automatically.

SECTION 1: General Questions about Sketching Cartoon Character

1. What is/are the most significant point(s) you should bear in your mind to create a “right” cartoon character sketching? (Please rank them according to their significance)

A) Skeleton structure	☐
B) Balance	☐
C) Proportion	☐
D) Exaggerated effect	☐
E) Others (Please specify)	☐

2. What object would you prefer to sketch when you drawing a cartoon character at the first stage?

A) Skeleton ☐

B) Contour ☐

C) Special parts of the character ☐

D) Others (Please specify) ☐

3. To draw a key-frame of cartoon character, please describe your sketching procedures in order, e.g. Reference line→ spine→ skeleton→ joints→ profiles, etc.

SECTION 2: Specific Questions about Sketching Cartoon Character

4. How do you normally maintain the proportion of character body during drawing? Give your answers for different scenarios by selecting one or more of the following choices:

A) Use some rules of thumb for an "average" body proportion (e.g. using the size of head as the main reference, so 1 body equals 7 heads height approximately, the shoulder is usually 3 heads wide, etc.)

B) Measuring by some drawing devices (e.g. pencil)

C) Following the intuition

D) Others (Please specify)

(i) When you draw a cartoon figure with images as references

A ☐ B ☐ C ☐ D ☐ ______________________________

(ii) When you draw a cartoon figure by memory

A ☐ B ☐ C ☐ D ☐ ______________________________

(iii) When you draw a cartoon figure in creative imagination

A ☐ B ☐ C ☐ D ☐ ______________________________

5. To draw a cartoon character, how do you normally manage the projection foreshortening of different body parts?

6. What is your solution for drawing overlapped components?

A) Draw the visible one (the one in front) first	☐
B) Draw the blocked one (the one behind) first	☐
C) Ignore the blocked one	☐
D) Others (Please specify)	☐

7. How do you normally express the exaggerated effects such as squash and stretch in drawing a cartoon character? Is there any rule (e.g. the global area is preserved)?

8. How do you express the relative position among several symmetrical body components (e.g. left/right arm, left/right leg) when sketching the skeleton? (Please tick multiple if applicable)

A) Draw one thicker, the other one relatively thinner	☐
B) Draw one longer, the other one relatively shorter	☐
C) In a special order (e.g. first left, then right)	☐
D) Others (Please specify)	☐

SECTION 3: Questions and Comments about the system and interface

9. Compared with other commercial cartoon software you used, do you feel it is more efficient and convenient to draw the sketch skeleton of cartoon character first, then the system generate the deformed character automatically? How about its accuracy?

10. For the reference skeleton segments (red lines) provided by the system, do you think it is helpful when you sketching the skeleton to deform the character?

A) Yes, it is very useful ☐

B) Not bad, a little useful ☐

C) No, it is unnatural ☐

D) Others comments ______________________________

11. For the fine tuning by sketches provided by the system (including through curves and point dragging), do you think this function is helpful?

A) Yes, it is very useful ☐

B) Not bad, a little useful ☐

C) No, it is unnatural ☐

D) It doesn't matter ☐

E) Others comments ______________________________

12. Do you think it is helpful for other auxiliary functions of our system, such as decomposition correction, depth adjustment, fine tuning by sketches? Please give the comments.

13. Please list the advantages of this system in your opinion compared with traditional cartoon making process.

14. Please list the disadvantages of this system.

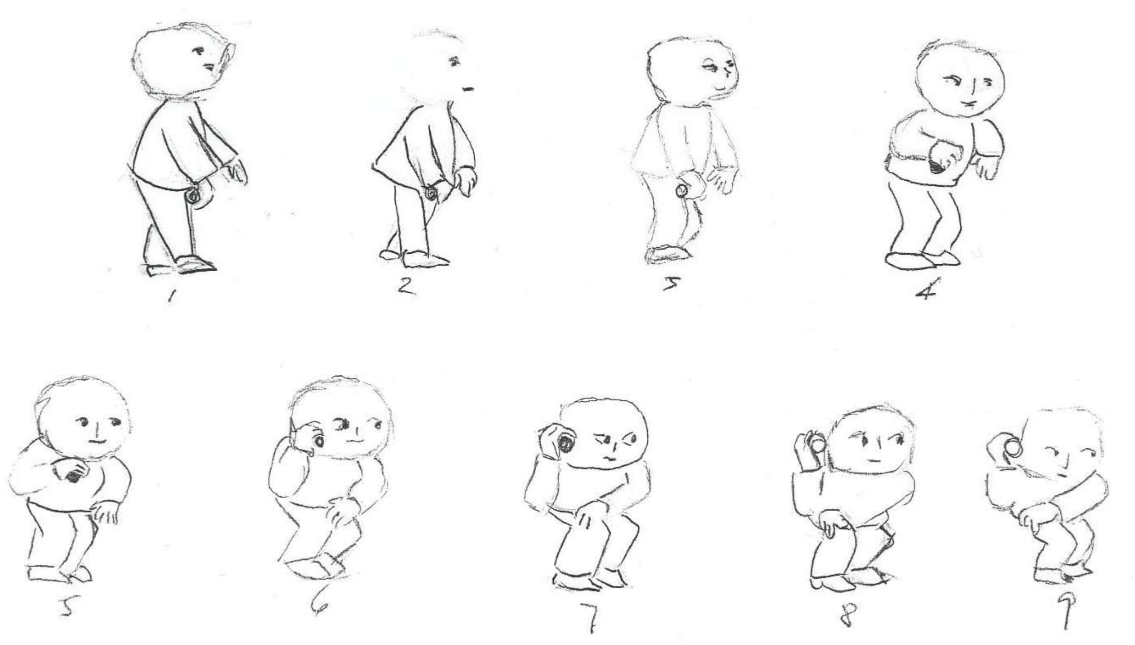
15. Please give your suggestions and comments about this system and interface for our further development.

APPENDIX B

SOME SKETCHES PROVIDED BY ANIMATORS

Following figures are some sketches used to guide and direct cartoon making in system test and evaluation (Section 6.4).

B.1 Sketches Provided by Animator 1



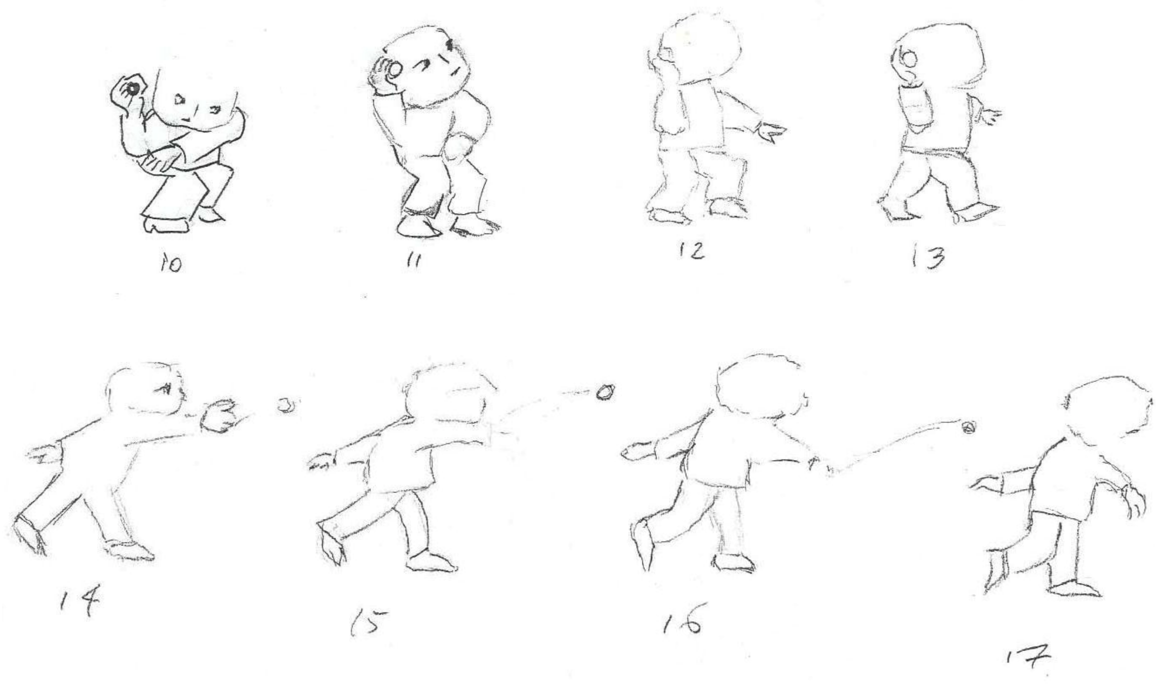
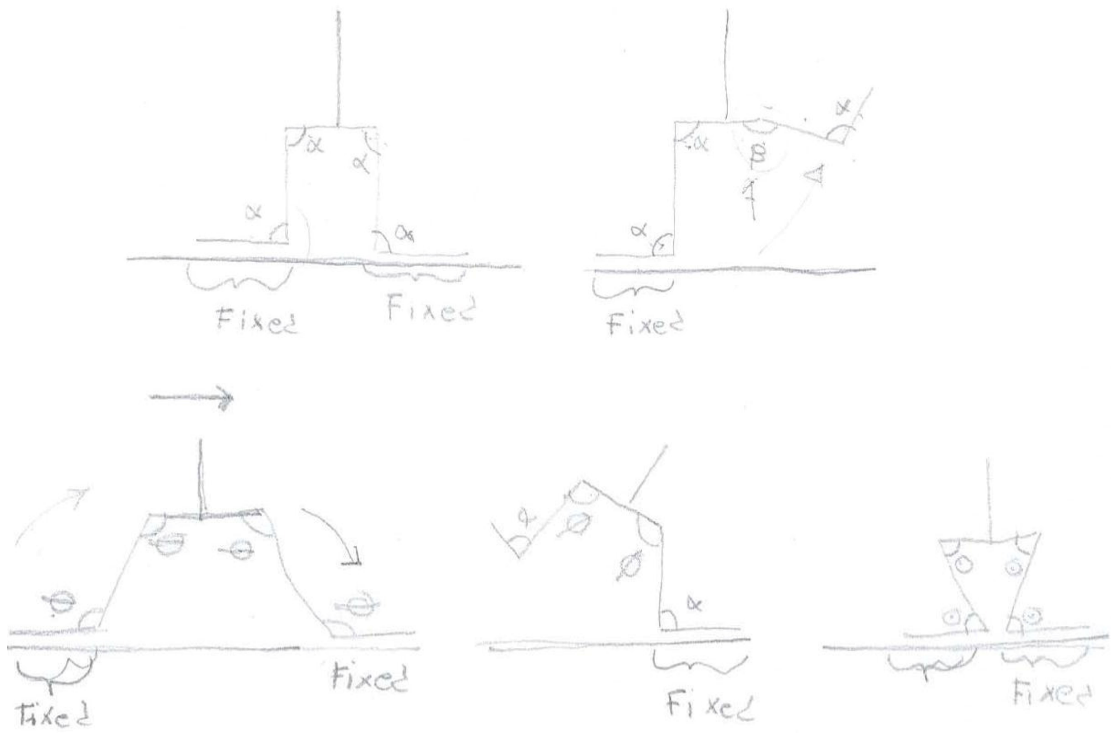
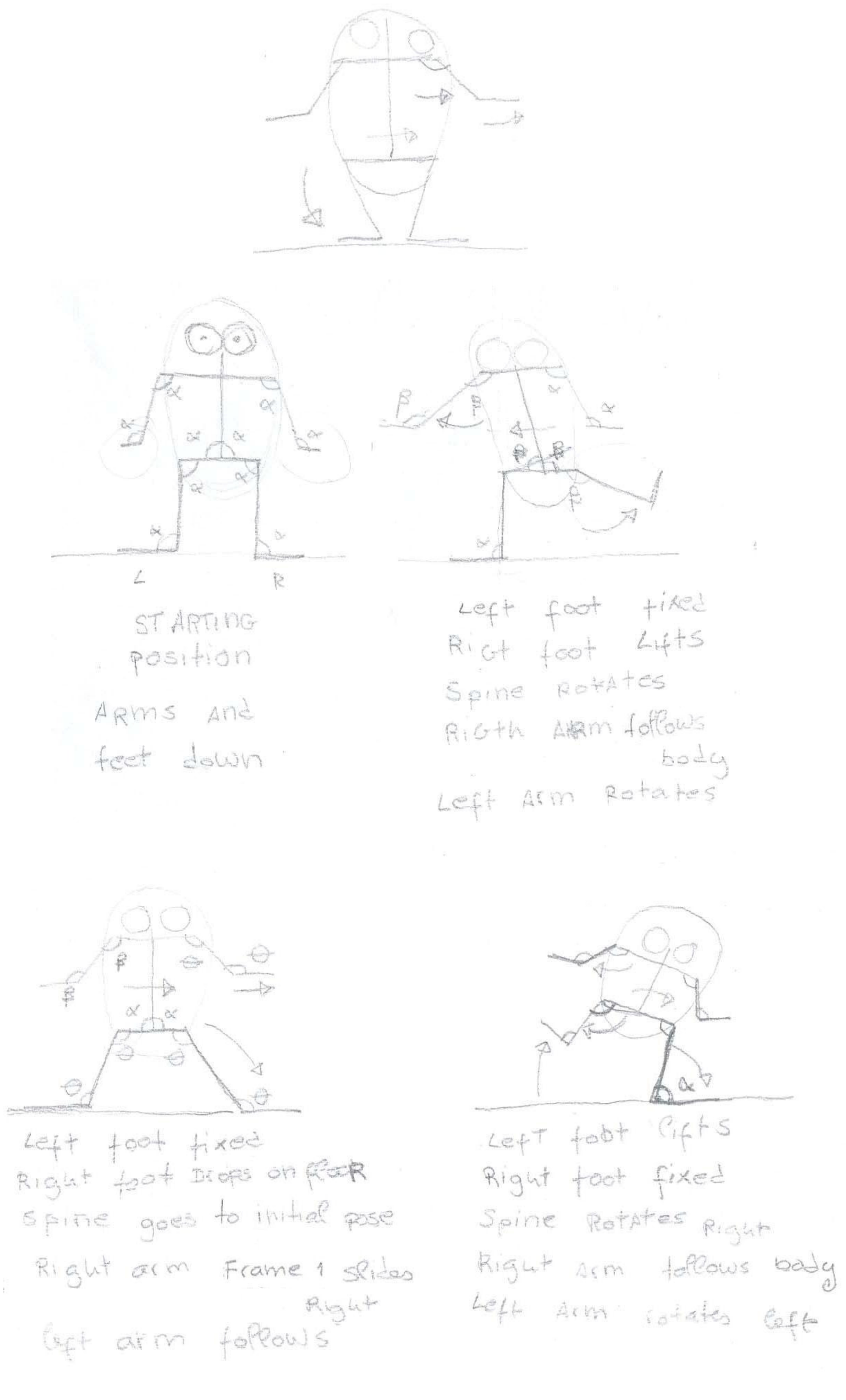


Figure B.1 Key-frames of the throwing.

B.2 Sketches Provided by Animator 2



(a)



(b)

Figure B.2 Key-frames of the robot walking.

REFERENCES

Agarwala, A., Hertzmann, A., Salesin, D. H., and Seitz, S. M., 2004. Keyframe-based tracking for rotoscoping and animation. *ACM SIGGRAPH 2004*. ACM Press, New York, NY, 584-591.

Aggarwal, K. and Triggs, B., 2006. Recovering 3D human pose from monocular images. *IEEE Transaction of Pattern Recognition, Mach*. 28(1), 44–58.

Alexa, M., Cohen-or, D. and Levin, D., 2000. As-rigid-as-possible shape interpolation, *Proceedings of ACM SIGGRAPH 2000*, ACM Press/Addison-Wesley Publishing Co, 157–164.

Alexe, A., Gaildrat, V. and Barthe, L., 2004. Modeling from sketches using spherical implicit functions. *Proceedings of 3rdinternational conference on computer graphics, virtual reality, visualization and interaction in Africa*, 25-34.

Barrett, W. A. and Cheney, A. S., 2002. Object-Based image editing. *Proceedings of the SIGGRAPH Conference*, ACM Press/Addison-Wesley Publishing Co, 777–784.

Bregler, C., Loeb, L., Erika, Chuang, E. and Deshpande, H., 2002. Turning to the masters: motion capturing cartoons. *SIGGRAPH 2002 Conference Proceedings*, ACM Press/Addison-Wesley Publishing Co, 121-129.

Bregler, C., and Malik, J, 1998. Tracking People with Twists and Exponential Maps. *Proceedings of IEEE CVPR,* 8-15.

Bruderlin, A. and Williams, L., 1995. Motion signal processing. *Proceedings of SIGGRAPH 95*, ACM Press/Addison-Wesley Publishing Co, 97–104.

Cai, Q., and Aggarwal, K., 1996. Tracking human motion using multiple cameras, *Proceedings of Conference on Pattern Recognition*, Vienna, Austria, August, 68-72.

Cappé, O., Moulines, E., and Rydén, T., 2005. Inference in Hidden Markov Models. Springer.

Chen, Y., Lee, J., Parent, R. and Machiraju., 2005. Markerless monocular motion capture using image features and physical constraints. *Proceedings of the Computer Graphics International 2005*, 36-43.

Chuang, Y.Y., Goldman, D. B., Zheng, K. C., Curless, B., Salesin, D. H. and Szeliski, R., 2005. Animating pictures with stochastic motion textures. *SIGGRAPH 2005 Conference Proceedings*, ACM Press/Addison-Wesley Publishing Co, 853-860.

Cornea, D., Silver, D. and Min, P., 2006. Curve-skeleton properties, applications and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 6(3), 81-91.

Correa,W. T., Jensen, R. J., Thayer, C. E. and Finkelstein, A. 1998. Texture mapping for cel animation. *Proceedings of the SIGGRAPH 98 Conference*, ACM Press, 435-446.

Dasarathy, B., 1991. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, ISBN 0-8186-8930-7

Davis, J., Agrawala, M., Chuang, E. and Slesin, D., 2003. A sketching interface for articulated figure animation. *Proc. Eurographics/SIGGRAPH Symposium on Computer Animation*, 320-328.

Drori, I., Cohen-or, D. and Yeshurun, H., 2003. Fragment-based Image Completion. *ACM Trans. Graphics*, 22(3), 303–312.

Efors, A. A. and Leung, T. K. 1999. Texture synthesis by nonparametric sampling. *Proceedings of the 7th International Conference on Computer Vision*,1033–1038.

Favreau, L., Reveret, L., Depraz, C. and Cani, M.P., 2004. Animal gaits from video. *Eurographics/ACMSIGGRAPH Symposium on Computer Animation*, 277-286.

Fekete, J.D., Bizouarn, E., Cournarie, E., Galas, T. and Taillefer, F., 1996. TicTacToon: A paperless system for professional 2D animation. *Proceedings of ACM SIGGRAPH 1996 Conference,* ACM Press/Addison-Wesley Publishing Co, 79–90.

Floater, M.S., 2003. Mean Value Coordinates. *Computer Aided Geometry Design,* 20(1), 19–27.

Fox, B., 2004. *3DS MAX 6 animation: CG Filming from Concept to Completion.* McGraw-Hill Osborne, California, US.

Gleicher, M., 1998. Retargeting motion to new characters. *Proceedings of ACM SIGGRAPH 1998,* ACM Press/Addison-Wesley Publishing Co, 33-42.

Gleicher, M., 2001. Motion Path Editing. *Proceedings 2001 Symposium on Interactive 3D Graphics*, 195-202.

Grewal, S., and Angus, A., 2001. *Kalman Filtering: Theory and Practice Using Matlab, Second Edition,* John Wiley & Sons Inc.

Hornung, A., Dekkers, E., and Kobbelt, L., 2007. Character animation from 2d pictures and 3d motion data. *ACM Trans. Graphics*, 26, 1, 1-9.

Hoshino, J. and Hoshino, Y., 2001. Intelligent storyboard for prototyping animation, *Proc. IEEE Conference on Multimedia and Expo*, 165-171

Hsu, S. C. and Lee, I. H., 1994. Drawing and animation using skeletal strokes. *Proceedings of ACM SIGGRAPH 94,* ACM Press/Addison-Wesley Publishing Co, 109-118.

Igarashi, T., Matsuoka, S., and Tanaka, H. 1999. Teddy: a sketching interface for 3D freeform design. *Proceedings of ACM SIGGRAPH 99,* ACM Press/Addison-Wesley Publishing Co, 79–89

Igarashi, T., and Hughes, J., 2003. Clothing Manipulation, *15th Annual Symposium on User Interface Software and Technology, ACM UIST'02, Paris, France, October 27-30, 2002,* 91-100.

Igarashi, T., Moscovich, T., Hughes, J.F., 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graphics,* 24(3), 1134–1141.

Irani, M., and Peleg, S., 1991. Improving Resolution by Image Registration. *Graphical Models and Image Processing,* May.

Isaac, K., 2003. Applying the 12 Principles to 3D Computer Animation**,** presentation at the *3D Festival in Copenhagen, and from the 3rd Edition of his book: The Art of 3D Computer Animation and Effects.*

Jain, A., 1989. *Fundamental of Digital Image Processing*. Prentice-Hall International, Inc.

James, D. L. and Twigg, C. D., 2005. Skinning Mesh Animations. *Proceedings of ACM SIGGRAPH 05*, ACM Press/Addison-Wesley Publishing Co, p. 399-407.

Kalman, R. E., 1960. A New Approach to Linear Filtering and Prediction Problems, *Transaction of the ASME—Journal of Basic Engineering, March,* 35-45.

Kong, Y. and Rosenfield, A., 1989. Digital topology: Introduction and survey. *Comp. Vision, Graphics and Image Proc. 48, 3*, 357-393.

Kort, A., 2002. Computer aided in-betweening. *Proceedings of the 2nd international Symposium on Non-Photorealistic Animation and Rendering (Annecy, France, June 03 - 05, 2002).* NPAR '02. ACM, New York, NY, 125-132.

Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. 2002. Interactive control of avatars animated with human motion data. *Proceedings of the SIGGRAPH Conference,* ACM Press/Addison-Wesley Publishing Co, 491–500.

Lewis, J. P., Cordner, M. and Fong, N., 2000. Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation. *SIGGRAPH 2000 Conference Proceedings*, ACM Press/Addison-Wesley Publishing Co, 165-172.

Li, Y., 2006. 3D Character Animation Synthesis From 2D Sketches*, Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia,* 81-90.

Liu, C., Torralba, A., Freeman, W. T., Durand, F. and Adelson, E. H., 2005. Motion magnification. *Proceedings of SIGGRAPH 2005*, ACM Press, 519–526.

Lorensen, W. and Cline, Harvey., 1987. Marching Cubes: A high resolution 3D surface construction algorithm. *Proceedings of SIGGRAPH 1987*, Vol. 21, 4, 163-169.

Maccracken, R., and Joy, K.I., 1996. Free-form Deformations with Lattices of Arbitrary Topology. *Proceedings of ACM SIGGRAPH 1996,* ACM Press/Addison-Wesley Publishing Co, 81-188.

Marksion, L., Cohen, J.M., Crulli, T. and Hughes, J.F., 1999. Skin: a Constructive Approach to Modeling Free-form Shapes. *Proceedings of ACM SIGGRAPH 1999*, ACM Press/Addison-Wesley Publishing Co, 393-400.

Mealing, S., 2002. *Computers and Art*. 2_{nd} Edition, Bristol: Intellect Ltd.

Michoud, B., Guillou, E., Briceno, H. and Boaz, S., 2007. Real-time marker-free motion capture from multiple cameras. *Proc. 11th International Conference on Computer Vision,* 125-134.

Moeslund, T. B. and Granum, E., 2001. A survey of computer vision based human motion capture. *Computer Vision Image Understanding, 81, 3,* 231–268.
Autodesk Motion Builder: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=6837710.

Pan, J., Yang, X., Xie, X., Willis, P. and Zhang, J. J., 2009. Automatic Rigging for Animation Characters with 3D Silhouette. *Computer Animation and Virtual Worlds*, 20(2-3), 121-131.

Pan, J., Zhang, J. J., Zhang, Y., and Zhou, H., 2009. X-ray Based Craniofacial Visualization and Surgery Simulation. *Recent advances in the 3D Physiological Human*, Springer-Verlag, New York. ISBN: 1848825641, August 2009, 208-224.

Pavic, D., Schonefeld, V., and Kobbelt, L. 2006. Interactive image completion with perspective correction. *The Visual Computer,* 22, 9, 671–681.

Pighin, F., Hecker, J., Lischinski, D., and Szeliski, R., 1998. Synthesizing realistic facial expressions from photographs. *Proceedings of ACM SIGGRAPH 1996,* ACM

Press/Addison-Wesley Publishing Co, 75–84.

Press, W. H., Flannery, B., Teukolsky, S., and Vetterling. W. 2007. Numerical Recipes: The Art of Scientific Computing. Third Edition, Cambridge University Press.

Robertson, B., 1994. Digital Toons. *Computer Graphics World*, Jul, 40–46.

Sand, P., McMillan, L. and Popovic, J., 2003. Continuous capture of skin deformation. *ACM Transactions on Graphics, 22(3),* 578-586

Schaefer, S., Mcphail, T. and Warren, J., 2006. Image deformation using moving least squares. *SIGGRAPH 2006 Conference Proceedings,* ACM Press/Addison-Wesley Publishing Co, 255-262.

Sheffer, A. and Kraovoy, V., 2004. Pyramid coordinates for morphing and deformation. *Proceedings of 3DPVT*, 125-134.

Shewchuk, J. R., 1996. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. *First Workshop on Applied Comp. Geometry Proc.*, 124-133.

Shewchuk, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation, computational geometry: Theory and applications. *Computational Geometry: Theory and Applications*, 22, 1, 3, 21–74.

Shi, J., and Tomasi, C., 1994. Good features to track. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR94), Seattle*. 185-192.

Shinya, M., Aoki M., Tsutsuguchi, K., and Kotani, N., 1999. Dynamic texture: physically based 2D animation. *Proc. ACM SIGGRAPH, 1999,* 239-246.

Sminchisescu, C., 2006. 3D human motion analysis in monocular video techniques and challenges. *Proc. the IEEE International Conference on Video and Signal Based Surveillance*, 76-85.

Sminchisescu, C., and Triggs, B., 2003. Kinematics Jump Processes For Monocular 3D Human Tracking. *Proc. Conf. Computer Vision and Pattern Recognition*, 69-76.

Sorkine, O., Cohen-or, D. and Toledo, S. 2003. High-pass quantization for mesh encoding. *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing,* 42–51.

Sorkine, O., Lipman, Y., Cohen-or, D., Alex, M., Rossel, C. and Sediel, H. P., 2004. Laplacian surface editing. *Symposium on Geometry Processing. ACM SIGGRAPH/Eurographics,* 179–188.

Sun, J., Jia, J., Tang, C. K. and Shum, H. Y., 2004. Poisson matting. *ACM Trans. Graph. 23, 3*, 315–321.

Sun, J., Yuan, L., Jia, J. and Shum, H. Y. 2005. Image completion with structure propagation. *ACM Trans. Graph. 24, 3*, 861–868.

Sykora, D., 2006. Computer Assisted Analysis of Classical Cartoon Animations. *Doctoral Thesis*. Department of Computer Science and Engineering, Czech Technical University in Prague.

Thorne, M., Burke, D. and Panne, M., 2004. Motion Doodles: An interface for sketching character motion. *SIGGRAPH 2004 Conference Proceedings*, ACM Press/Addison-Wesley Publishing Co, 424-431.

ToonBoom: http://www.toonboom.com/products/digitalpro/eLearning/tipsTricks/ 2008

Van, W., Di, F., and Van, F., 2005. Uniting cartoon textures with computer assisted animation. *Proceedings of the 3rd international Conference on Computer Graphics and interactive Techniques in Australasia and South East Asia (Dunedin, New Zealand, November 29 - December 02, 2005). GRAPHITE '05*, 245-253.

Wang, J., Bhat, P., Colburn, A., Agrawala, M., and Cohen, M., 2004. Interactive Video Cut out. *SIGGRAPH 2004 Conference Proceedings, ACM Press,* 124-131.

Wang, J., Drucker, M. S., Agrawala, M. and Cohen, F.M., 2006. The Cartoon Animation Filter. *SIGGRAPH 2006 Conference Proceedings,* ACM Press/Addison-Wesley Publishing Co, 155-162.

Wang, Y., Xu, K., Xiong, Y. and Cheng, Z. 2008. 2D shape deformation based on rigid square matching. *Computer Animation and Virtual Worlds*, 19(3-4), 411-420.

Wang, Y. and Lee, Y., 2008. Curve skeleton extraction using iterative least squares optimization. *IEEE Transactions on Visualization and Computer Graphics,* 5, 196-209.

Wang, C. and Yuen M., 2003. Freeform extrusion by sketched input. *Computers and Graphics*, 27(2), April, 255-263.

Weng, Y., Xu, W., Wu, Y., Zhou, K. and Guo, B., 2006. 2D Shape Deformation Using Nonlinear Least Squares Optimization. *The Visual Computer,* 22(9), 11, 653-660

Wiedermann, J., 2002. *500 3D-objects*, Taschen.

Wilhelms, J., 1997. Animals with Anatomy, *IEEE Computer Graphics and Applications,* v.17, May 1997, 22-30

Williams, R., 2001. *The Animator's Survival Kit*, London and New York: Faber &

Faber.

Wyvill, B., 1997. *Animation and Special Effects.* Morgan Kaufmann, Chapter. 8, p. 242–269.

Xu, X., Wan, L., Liu, X., Wong, T., Wang, L. and Leung, C., 2008. Animating animal motion from still. *ACM SIGGRAPH Asia 2008*, *ACM Press,* 1-8.

Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B. and Shum, H. Y., 2004. Mesh editing with poisson-based gradient field manipulation. *Proceedings of ACM SIGGRAPH 2004*, 644-651.

Yuki, M. and Igarashi, T., 2007. Plushie: an interactive design system for plush toys. *SIGGRAPH 2007 Conference Proceedings,* ACM Press, 389-397