ANALYSIS, SYNTHESIS, AND RETARGETING OF FACIAL EXPRESSIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Erika S. Chuang
March 2004

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christoph Bregler
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Patrick M. Hanrahan

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Eugene J. Alexander

Approved for the University Committee on Graduate Studies.

Abstract

Computer animated characters have recently gained popularity in many applications, including web pages, computer games, movies, and various human computer interface designs. In order to make these animated characters lively and convincing, they require sophisticated facial expressions and motions. Traditionally, these animations are produced entirely by skilled artists. Although the quality of manually produced animation remains the best, this process is slow and costly. Motion capture performance of actors and actresses is one technique that attempts to speed up this process. One problem with this technique is that the captured motion data can not be edited easily. In recent years, statistical techniques have been used to address this problem by learning the mapping between audio speech and facial motion. New facial motion can be synthesized for novel audio data by reusing the motion capture data. However, since facial expressions are not modeled in these approaches, the resulting facial animation is realistic, yet expressionless.

This thesis takes an expressionless talking face and creates an expressive facial animation. This process consists of three parts: expression synthesis, blendshape retargeting, and head motion synthesis. Expression synthesis uses a factorization model to describe the interaction between facial expression and speech content underlying each particular facial appearance. A new facial expression can be applied to novel input video, while retaining the same speech content. Blendshape retargeting maps facial expressions onto a 3D face model using the framework of blendshape interpolation. Three methods of sampling the keyshapes, or the prototype shapes, from data are evaluated. In addition, the generality of blendshape retargeting is demonstrated in three different domains. Head motion synthesis uses audio pitch contours to derive

new head motion. The global and local statistics of the pitch and the coherency of head motion are utilized to determine the optimal motion trajectory. Finally, expression synthesis, blendshape retargeting, and head motion synthesis are combined into a prototype system and demonstrated through an example.

Preface

This thesis is about animation. It is impossible to evaluate the result without seeing the animated sequences. The videos of animation are included in the attached CD. They are named with the same numbers as shown in the figures, which are labeled with (video figure) in the list of figures.

Acknowledgments

This thesis was made possible by the support from many people whom I have come across over the course of graduate school. I would like to thank:

My advisor Chris Bregler, who introduced me to computer animation, which is an exciting research area that straddles across multiple disciplines, including computer vision, machine learning and computer graphics. He is a great source of inspiration for obtaining insights on how to pick interesting problems, and developing your own style of research. This thesis would not have been possible without his patience, support, and encouragement.

The rest of my thesis committee: Pat, who inspired me to look at a problem in new ways. Gene, who introduced me to bio-mechanical engineering where the studying of motion has great impact on medical research. Lance, who undoubtedly is the world's leading expert on facial animation, asked many good questions and gave great suggestions.

Jutta, Hoa, Ada and Heather, for making life more pleasant in Gates Building.

Lorie, who introduced me to the beauty of traditional animation, which plays an important role in my research. Kathy, whose research influenced my thinking in many ways. Her passion for dance is always inspiring.

The Stanford graphics lab. Not only it is one of the most dynamic places to be in terms of learning and brainstorming on research ideas, it is also a very friendly and supportive environment.

The people in the NYU graphics lab, who made me feel at home in the busy city of Manhattan during my several visits and helped me inside and outside the lab.

The people in USC graphics lab, for their friendships during my stay in L.A. J.P.,

who was always very approachable and helpful during research discussions.

My officemates Pedrito, Ursula, Lorrenzo, Hrishi, Natasha, Dan, Brad, who not only were the greatest people to learn things from, but also made graduate school fun.

My friends from various stages of graduate school for their friendships and advices on grad school survival. Allen, Yung-hsiang, Li-Yi, Fu-Tai, Doug, Yi-Chang, Cindy. Members of TND, Afra, Niloy, Natasha, Kostatis, Daniel, Brad, for wasting all my Thursday nights for the last year and half "with other geeks, instead of meeting and socializing with normal people.", and "spending the money I should've been saving for the new gadgets I've been wanting." (original quote from the 50th TND). Nevertheless, it's good to know that I have friends on Thursday nights.

My actresses, Eva, Alicia, Kate, and Rachel. I learned so many things from your expressive acting, some of those you will probably never find out yourself.

Friends outside of graduate school, who reminded me from time to time that life can be normal.

James, not only for his love and support, but also for the effort he spent on fixing my 'Chinglish' writing and helping me with video equipment during those late nights before deadlines.

Poncho, because it is important to mention your dog.

My family, for their unconditional love and support.

I am sure I have left out some people, but it probably means that they are just as important to me.

Contents

List of Figures

Chapter 1

Introduction

Computer animated characters are now necessary components of many applications, including computer games, movies, web pages, human computer interface designs, and communication and psychological studies. They are used in the place of actual human performers for many reasons, including freedom to change the appearance of the characters, simulating a crowd scene, and issues with privacy. Figure 1.1 shows an example of an animated character.

In order to make these animated characters convincing, they require sophisticated facial expressions and motions. Traditionally, facial animation has been produced largely by skilled artists manually. Although manually animated characters ensures the best quality, this process is slow and costly. While large studios or production houses can afford to hire hundreds of people to make feature films and movies, it is not feasible for low budget or interactive applications.

Over the last 30 years, a great deal of research effort has been dedicated to techniques to assist and automate the process of character animation. As part of that

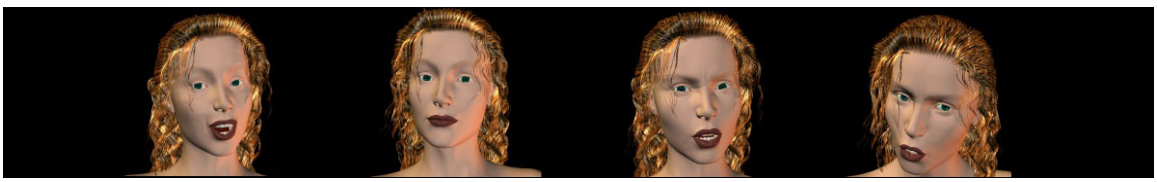


Figure 1.1: A computer animated character

effort, motion capture based, or sometimes called performance driven methods, have been investigated to produce facial animation more efficiently. To this day, motion capture data is often applied to only a single animation and never used again. Each new animation requires a new capture session, resulting in a large amount of effort. In some situations, re-capturing is simply not an option, such as the use of old captured footage, where the actor or actress is no longer available. Ideally, an artist could reuse motion data and freely edit the speech content, the emotional style, or the visual appearance of a character, while retaining the essence of the captured performance. However, editing motion capture data is still quite difficult, and no completely satisfying solution yet exists.

Most current research has focused on either data driven speech synthesis which changes the speech content and lip motion of a video sequence, or character animation which focuses on visual appearance and methods for retargeting geometric deformations from one face onto another. Relatively few methods exist for editing and retaining the expressive style of facial animation. While existing methods can produce photorealistic results, they focus on changing the content of "what" the character does, but not the style of "how" they do it, leaving the expression of captured data untouched. This thesis addresses the need for expressive facial animation by introducing methods for editing the emotional style of the data, retargeting the data while retaining the style, and synthesizing head motion in a way that explicitly incorporates emotion.

Methods for editing content, style and appearance have been largely separated, with few attempts to build a complete systems for editing and retargeting all aspects of a facial animation. One of the keys to this problem lies in the choice of data representation. Content editors seek to preserve the identity of the original actor or actress, so they choose a representation that preserve the appearance of the data, often including complex high-dimensional texture models. Retargeting methods seek to preserve speech content while editing appearance, so they choose a representation such as marker positions that is general and adaptable to multiple face models. Unfortunately sparse models of 3D markers may not contain sufficient information to encode emotion. A complete system will need to choose a data representation that is

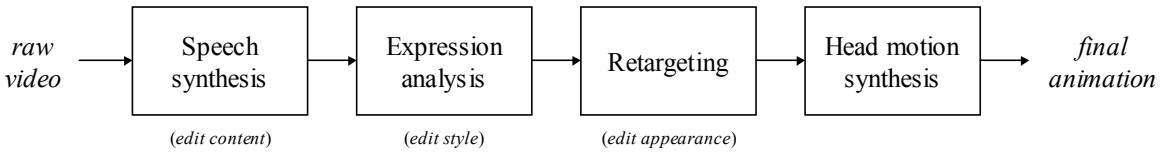


Figure 1.2: Editing pipeline for expressive speech animation. This work focuses on the last three modules, editing and retargeting expressive animation onto new characters.

compatible with these often contradictory goals. To address this need, and highlight the importance of designing algorithms that are compatible with the whole process, this thesis introduces a framework for integrating content, style and appearance into a single editing pipeline.

This thesis attempts to resolve the problem of data representation by choosing the middle ground. We neither choose a dense texture model, nor a simple sparse marker model. We instead analyze video texture information and extract concise higher level information that encodes facial expression. This expression information is used to augment a sparse geometric model when retargeting. In addition, our method allows artists to retain control over the process of interpreting various facial expressions. Input geometry is mapped implicitly to the output face models, rather than preserving explicit geometric similarity.

Figure 1.2 illustrates the editing pipeline proposed in this work. The input performance is a video sequence of a talking face. The speech content of this sequence could be edited using any of several existing techniques. The resulting video is then analyzed using a statistical model, to factor emotional style and speech content into two components. This model essentially provides an expression shader with which we can modify the emotional style of the video. We next retarget the video sequence onto a 3D character. By using the extracted emotion vector to augment traditional shape information, our retargeting method preserves emotional style while allowing the characters appearance to be freely edited. Finally, since head motion is an important aspect of expression, we present a data driven synthesis technique that matches the characters emotional state. Since this work emphasizes the importance of expression, we focus on the last three modules, simply passing the input video's speech content directly to the expression synthesis module.

The following sections review the techniques involved in creating a computer animated face. Since the main focus of this thesis is animating facial expressions using motion capture data, I will also review some related work on editing and reusing facial motion capture data, and adding head motion to facial animation.

1.1 Techniques for computer facial animation

Computer animation of faces has been a topic of interests since the pioneering work in Frederic Parke's theses, "Computer Generated Animation of Faces" [70], and "A Parametric Model for Human Faces" [71], published in 1970's. Since then, much work has been published in computer facial animation. Like most subjects, the nature of facial animation is multidisciplinary and many areas of research are involved. To organize these areas for the convenience of discussion, they are roughly categorized into the following tasks.

Creating facial models The 3D (or 2D) face model used for facial animation. There are many ways to obtain such a model. It can be created manually with a software modeling tool; captured from a 3D clay model using a digitizing pen, which is mounted on a mechanical arm that calculates the position of the tip of the pen; or captured from a real human using cameras or other scanning technology. This 3D data is then parameterized into a mathematical representation such as polygonal models, splines, or implicit surfaces, so that they can be manipulated on the computer.

Modeling static facial expressions Methods for manipulating the geometry of the face models to change the appearance. These methods focus on ways to parameterize the face model, and map the parameters to observable changes in the facial appearance, such as dimensional variations among different individuals, or facial expressions that reflect an emotional state.

Animating dynamic facial expressions Methods to specify motion and dynamics for the face model to generate a sequence of static facial expressions. The

motion may include lip sync, which maps content of the speech to the shape of the mouth, and changes in facial expressions, which map emotional states to the appearance.

Rendering Methods for synthesizing the actual facial images displayed on the screen. Depending on the goal of the animation, the style of rendering may range from photo-realistic to pencil sketches.

Since the focus of this thesis is on animating facial expressions, we will only cover techniques for modeling and animating facial expressions (items shown in *italics* above) in the following sections. Further information on creating face models or rendering facial images can be found elsewhere [73].

1.1.1 Modeling static facial expressions

The task of modeling static facial expressions involves mapping a set of values, or parameters, to the appearance of the face model. Modeling static facial expressions requires two tasks for creating a parameterized face model: development of the appropriate parameter set, and methods for mapping the numerical values for the parameters to the observable changes in the appearance of the facial model [72].

Developing facial parameter set Face parameters can be defined in many levels of details. They range from detailed descriptions of the appearance of the face, to the underlying structure and material property of the facial physiology. Figure 1.3 shows a continuum of space of the facial parameters.

In one extreme, the parameters can describe the appearance of the face, such as how much the eye brow is raised, how many wrinkles are on the forehead, and whether there are sideburns. Since the space of the facial appearance is very large, the number of parameters to describe all the variations in details often becomes intractable. Various statistical techniques have been developed for compressing the large amount of information to some manageable size. For instance, EigenFace [92] is a linearization of facial images onto a set of orthogonal basis. These parameters are

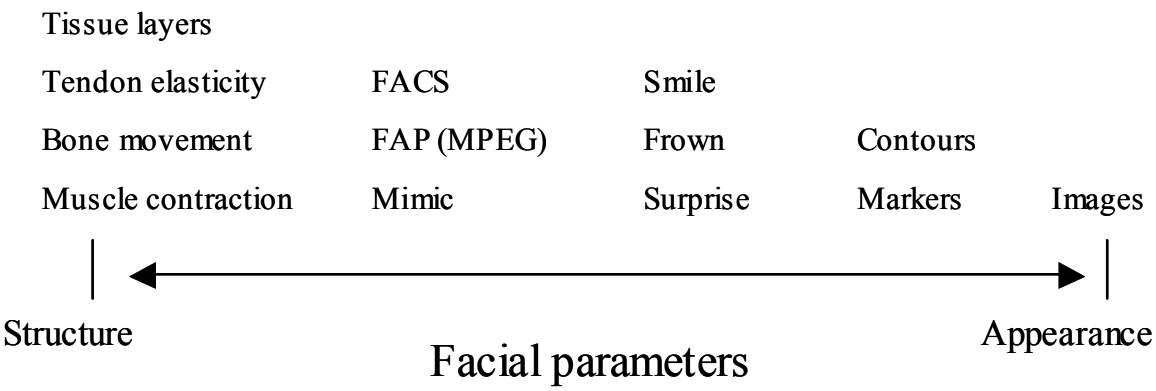


Figure 1.3: A continuum of facial parameters. In one extreme, the parameters can describe the appearance of the face. In the other extreme, the parameters can describe the underlying structure and material property of the human face.

very useful for applications like compression and facial recognition, however, they are difficult to control and manipulate for the purpose of animation.

In the other extreme, the parameters can describe the underlying structure and material property of the human face, such as bones, muscles, tendons, and tissues, etc. Ideally, if the physical property of these parameters can be simulated accurately, we can produce any kind of facial appearance using physical laws. However, physical simulation of biological materials is still an active research areas, and it is not yet mature enough for generating long sequence of complex facial animation.

Various types of parameterization combine the approaches from these two extremes, supplementing structure-based parameters with parameters based on appearance. For instance, the most popular methods for encoding facial expressions developed by Ekman, *Facial Action Coding System* (FACS) [32], is based on activity of facial muscles. Nevertheless, one can be trained to annotate with this parameterization by looking images of faces. Parke [72] described a hybrid method that divides the facial parameters into conformation parameters, which distinguish one individual from another, and expression parameters, that are more universal to all human faces. The Facial Action Parameters (FAPs) used in MPEG4 is another example of this type of approach. It is designed to compress facial animation, but was originally inspired by FACS [68, 44].

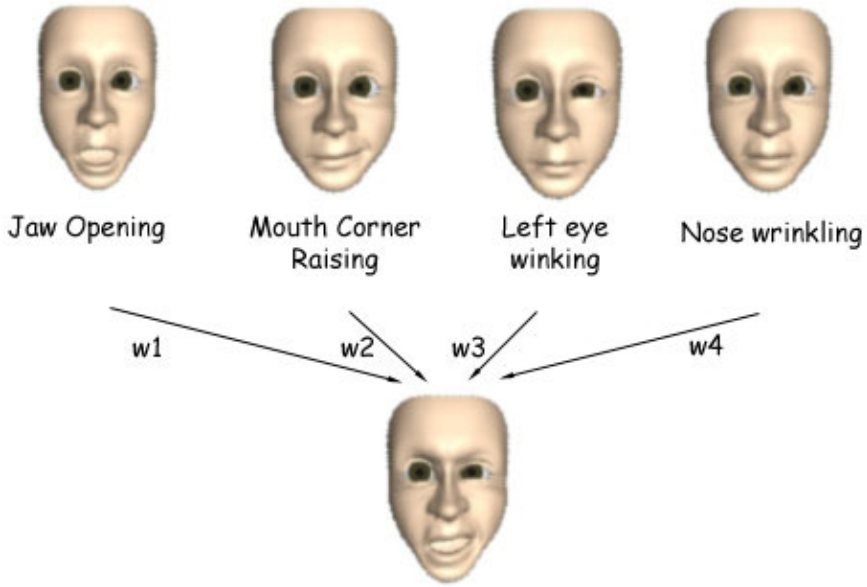


Figure 1.4: Illustration of blendshape interpolation. The four images on the top are examples of changing one single parameter value, e.g. jaw opening, mouth corner rasing, eye winking, nose wrinkling. The bottom image is an example of weighted combination of these four basic parameters.

Regardless of the parameters used, there is always a need to describe more complex facial expressions based on the basic predefined parameters, such as a smile combined with a wink. *Blendshape interpolation* is a powerful technique that combines and mixes basic facial expressions using a weighted combination of underlying parameters. Figure 1.4 shows a simple example of blendshape interpolation. The four images on the top are examples of changing one single parameter value, e.g. jaw opening, mouth corner rasing, eye winking, nose wrinkling. The bottom image is an example of weighted combination of these four basic parameters. By using a small set of predefined parameters combinatorially, animators can to create additional facial expressions and increase the variety of facial appearance. In chapter 3 we will come back to this technique and describe how it is used in the framework of motion retargeting.

Mapping parameters to appearance Given a set of facial parameters and a face model, we need to map the numerical values of the parameters to observable changes in the face model. This mapping is highly dependent on how the parameters are defined. If the parameters are defined near the end of the spectrum for describing the appearance of the face, the mapping tends to be more direct. However, if the parameters are defined near the other end of the spectrum for structural description, the mapping often involves layers of complex modeling. Regardless of the parameterization, somewhere in the pipeline, there needs to be some kind of geometrical deformation that changes the shape of the face to reflect the differences.

Most geometric deformation techniques are based on a number of control primitives, such as control points, edges, triangle patches, and curves, placed upon the surfaces of the model, or regular grids for dividing the volumes. To describe how the rest of the face deforms with respect to these control primitives, ones solves a scatter data interpolation problem by applying a function that describes the weighted contribution for the deformation in the region of influence [5, 63, 42, 67].

For parameters that are structure-based, more complex mappings are required for simulating the physical property of the underlying structures. Local deformation functions are embedded in higher level descriptors such as muscle and skin layers, and the mechanical properties such as muscle spring constant, zone of muscle influence, elasticity of the tissues, etc. are translated into the local deformation that causes the model surface to deform and create various facial expressions [78, 95, 60, 55, 54].

The choice of parameters and the mechanism for the mapping is often the most important part of facial animation. However, there is not a universally correct method, since the decisions is dependent on the specific application. For instance, in medical applications, the physiological correctness in the facial parameters would be far more important than in entertainment and social applications.

1.1.2 Animating dynamic facial expressions

Animating facial expressions involves specifying motion and dynamics to generate a sequence of static facial expressions. In other words, we need to specify the values of

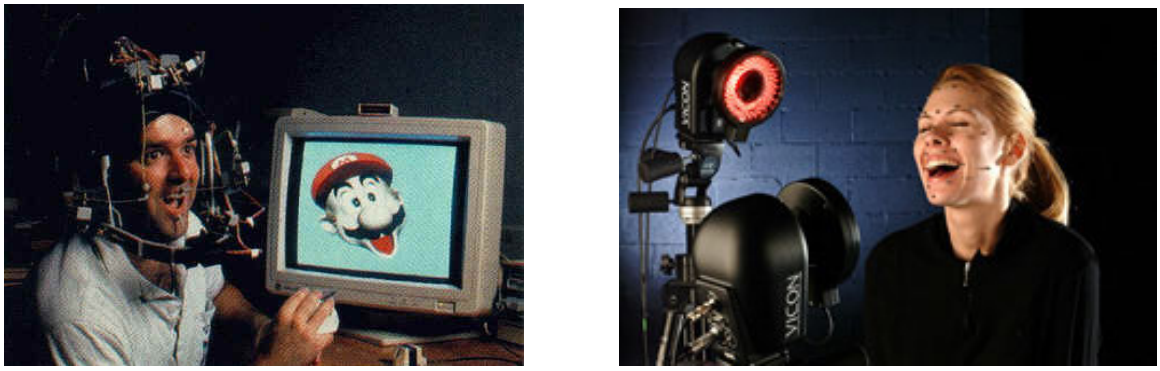
the facial parameters over time. Existing techniques include the following:

Keyframe Artists specify the parameters for each frame manually throughout the sequence. This ensures high quality facial animation. However, it is labor intensive and not very efficient for long animation sequence. To reduce the amount of work, given two keyframes, we can create the in between facial expressions using interpolation functions. This interpolation takes place in the time domain, as opposed to a spatial one in the case of blendshape interpolation described in section 1.1.1. Frames that are 'less important'(do not exhibit characteristic movement) are automatically filled in by the process. Common functions used for interpolation include linear and spline, or perhaps other higher order functions.

Rule-based Researchers in psychology, communications and behavioral science study the link between speech, facial expression, and gestures [85]. These studies result in sets of rules that are very useful for animating believable characters. For instance, facial expressions can be categorized based on their communicative meanings, e.g. a facial shrug indicating "I don't know" is often displayed as eyebrow flashes and mouth corners pull down and back [87]. Many other rules are derived from various functions that facial motion serve. For instance, some motions are tied to the intonation of the voice, e.g. eyebrow raising is often linked to accented words; other motions serve biological needs of the face, e.g. wetting the lips, and blinking; some motion reflect emotional state, e.g. frowning for worry or wrinkling of the nose for disgust [17, 18]. Multi-layer approaches allows one to specify the synchronous effects among the motions [53]. The advantage of a rule-based approach is that the facial animations are compatible with social behaviors and the semantics of the spoken words. However, the rules can only direct facial animation at the higher level. Much of the detailed expressions and motion still require manual specifications.

Performance driven Sometimes called *motion capture based animation*, or *digital puppeteering*. The motion is first recorded from the face or body of a human performer and then used for driving the motion of the animated face. In the early days,

the methods for capturing the movement were mostly mechanical. For example, the first performance-driven computer animated face was "Mike, the Talking Head" [83]. Mike was driven by a specially built controller that allowed a puppeteer to control many parameters of the character's face, including mouth, eyes, expression, and head position. Mechanical devices allow real-time performance, however, they are often bulky and uncomfortable to wear, as figure 1.5a suggests. The more recent technolo-



(a) (b)

Figure 1.5: Examples of facial motion capture systems. (a) Mike, the talking head. The controller is built that allows the puppeteer to control the character's face, including mouth, eyes, expressions, and head position. (b) Vicon [86], a commercial motion capture system.

gies for tracking facial motion are based on optical techniques. Vision-based capture techniques are less intrusive than mechanical systems, allowing the performers to move around freely. Commercial systems often use retro-reflective markers, which shine very brightly when illuminated, and make tracking easier for real-time application (figure 1.5b). However, fundamental problems such as noise and missing markers caused by occlusion often limit the real-time capability. There are also techniques for marker-less tracking, however until very recently they were not robust enough for real-time performance [33, 30, 25].

In recent years, performance driven facial animation has gained popularity due to better technology in tracking and an increasing number of commercialized motion capture systems. There are several advantages of this approach, including shortened

production time, and the possibility of capturing the essence and personality of a real person and apply it to an animated character.

Mapping the motion of the marker positions to 3D face models entails a number geometric techniques as described in section 1.1.1.

Analysis-based This technique consists of tracking facial expressions from a live video sequence, extracting the facial parameters, and giving the parameters as input to a facial animation system [90, 35, 33, 77]. The parameters are either in terms of muscle contractions, FACS units, Facial Animation Parameters (PAR) as specified in the MPEG4 compression standard, or weights for the blendshapes. However, estimating facial parameters from video sequence is difficult due to the subtlety and complexity of facial deformation. Often a complex model of the face is required to achieve the level of details in a live video. It is still an active research topic in the computer vision community.

The dividing line between performance driven and analysis-based techniques has blurred over the last several years. From a high level point of view, both sets of techniques capture live performance and use it as input to animation system. The differences arise from the format of the input data and the facial parameterizations. In performance driven techniques, the data is captured and represented as 3D marker positions, and is used directly for facial animation. In analysis-based techniques, the data is captured as images, and parameterized into a more structural representation. The structural representation is then used for the animation in reverse. Recently, techniques have been introduced to mix and match different data representation and facial parameterization to combine advantages in both approaches. For instance, in photo-realistic, performance driven facial animations, a combination of dense laser-scanned mesh, and optical flow and photogrammetry techniques are often used to capture the details on the face to achieve the level of quality required for films, adding sophisticated analysis steps to performance driven techniques [11]. Marker positions are sometimes used to assist the estimation of muscle contractions, simplifying the analysis-based techniques to achieve real-time performance. All of these variations

can be described as part of a forward/backward process by considering the spectrum of facial parameters shown in figure 1.3. This process starts with selecting a input data representation on the right side of the spectrum (e.g. marker positions) and solving for the desired facial parameters on the left (e.g. muscle contraction), and then mapping them back to the right for animation. The input data can be 2D or 3D features, dense texture or coarse points, markers or without markers. The facial parameters to estimate can be appearance-based, geometric-based, or structure-based. Finally, the techniques for the animation depend on the choice of facial parameters. The general belief is that the structural parameters are more universal and powerful to represent the complexity of the face. However, to estimate structural facial parameters from input data requires more sophisticated models, while the direct mapping method can bypass much of the complicated steps and achieve good result.

1.2 Editing facial motion data

The discussion at the end of the previous section broadens the definition of motion capture based facial animation to include the two schools of techniques on performance-driven and analysis-based methods which are traditionally separated. We will use this broadened definition of motion data for the discussions of reusing motion data and editing motion data in this section.

Even though there are a lot of advances in motion capture technology, processing motion data is still a demanding task. Therefore techniques for reusing motion data are imperative due to the time and budget constraints in many applications. Furthermore, the ability to edit the motion data is essential for increasing the versatility of the existing motion data. The difficulty in editing motion capture data is manifold, including the lack of unified methods for segmenting, classifying, controlling and modifying the motion data. In addition, facial animation is used for many different applications, and the type of editing that best fits in the context of the usage is dependent on the application, making it even more difficult to standardize the approaches. Finally, the success of the animation depend on our ability to retarget the motion to other characters without losing the essence and personality of the original performer.

As shown in figure 1.2, the motion editing pipeline can be broken down into three stages: editing the content, the style, and the appearance of the motion data. Most existing work is in the area of content editing and appearance editing.

Content editing, often called lip sync, is a much studied topic as it is essential for applications that generate new facial animations for new speech, such as a virtual conversational agent. The phonetic units of speech provide an intuitive starting point for motion segmentation, which leads to well structured methodologies for lip sync. Cohen and Massaro stressed the importance of co-articulation for natural lip sync [20]. To learn this effect from motion data, statistical techniques has been applied to learn models of speech content in order to derive novel motion based on existing data. For instance, Video Rewrite [13] applied machine learning techniques to synthesize a talking face. To ensure a smooth transition between words, a triphone model is used for each viseme. The resulting system enables the same person to say things that she has never said before. Ezzat et al. [37] improved upon Video Rewrite by learning a set of prototype mouth shapes from the training data. This minimizes the storage required to keep all the video in the database. Other techniques are introduced to estimate the pose of face to more accurately model local image deformations caused by small head motion [22]. These work are examples of the image-based approach, where the data is parameterized based on the appearance, and analyzed to various degrees for editing. The focus of these work is the realism of the synthesized face, or the naturalness of the audio-to-visual mapping for the speech animation. Voice Puppetry [12] learns a probability distribution of facial motion by applying a Hidden Markov Model (HHM) to both the audio signal and the facial motion. Given novel audio input, the algorithm predicts the most likely trajectory of the facial motion. The input data to the system is the 2D positions of the facial markers. The synthesized motion is then retargeted to a different face by image warping. Over the last couple of years, extensions of these techniques to 3D were introduced due to advances in vision technology [52, 50].

Style editing refers to the controlling of the expressive aspect of motion data. Research effort in data-driven facial synthesis has largely ignored emotion, and the resulting speech retains the same neutral expression as the input data. As the topic

is related to the content of chapter 2, the discussion of related work is deferred to that chapter.

Appearance editing refers to using motion data captured from one individual and retargeting it onto a different character. This is well studied in the context of performance-driven facial animation. As the topic is directly related to the content of chapter 3, the discussion of related work is deferred to that chapter.

Ultimately, we wish to create facial motion that captures the idiosyncrasy of a particular person, and yet retain full artistic control for in creating the style and appearance of the character. The higher level goal of this thesis is to tie together the work of the analysis and learning based facial synthesis techniques with the other areas of computer facial animation, such as modeling and rendering, to generate better facial animation more efficiently and more realistically.

1.3 Animating facial gestures

In character animation, creating motion for the face is only a small part of the effort. What makes a character come alive is all the other non-verbal aspects of facial gestures that help to convey the personality and mood. These facial gestures include head motion, postures, blinking, eye gaze. Without these gestures, facial animation looks very rigid and unnatural. In fact, traditional animators often start animating the head motion and body gestures, and fill in the details of the facial expressions and lip sync later. The final part of this thesis will tackle one of these facial gestures, head motion, by introducing a technique for generating head gesture using motion capture data.

Perlin [75] proposed generating random noise for character movement. In addition to head motion, this random noise is injected to different layers of animation, from low level signals such as eye blinking, to higher level signals that suggest certain attitude. These motions are then blended together to create improvisational virtual characters. The random motion is not correlated to anything in the animation, therefore after staring at the character for a long time, one often gets the impression that the character is not very engaging. Nevertheless, it is surprisingly effective given how simple

this approach is. This further confirms our belief that head motion is an imperative part of facial animation.

Another category of approach to generate head motion is behaviorally based. Head-motion procedures that model speaker turns and interaction with a user have been proposed by Cassell [17], Nagao and Takeuchi [87], and Poggi and Pelachaud [79]. Most recently, DeCarlo et al. [26] proposed a new procedural system that enhances non-verbal aspects of speech with head motions. They also carefully studied non-verbal cues in TV newscast videos, and noticed that very often head-motions (brief nodding, side turns) are used to highlight the intended interpretation of the utterance. These rule-based techniques for generating head motion share the same advantage of being behaviorally correct as the rule-based facial animation technique. However, they also suffer the same issues in that large amounts of manual specifications are required to fill in the details.

This thesis introduces a data driven method for synthesizing head motion that is correlated with the emotional content of the facial animation, and contains the fine details that are necessary for interesting animation. We draw inspiration from several recent articulated motion synthesis systems that have explored similar data driven methods [2, 80, 57, 61].

1.4 Contribution of the thesis

The contribution of this thesis includes the following:

- Designed an algorithm for editing facial expressions of live performances using a bilinear model derived from motion data. The bilinear model allowed factoring the expressions from the visual content of the speech.

- Formulated blendshape retargeting as a constraint least-square problem, and evaluated three methods for selecting keyshapes from the source motion data.

- Designed an algorithm for synthesizing head motion from audio input signal using existing motion capture data. Incorporated expressiveness in the synthesis

algorithm through selected audio features that manifest the emotional states of the speaker.

- Proposed an editing pipeline for expressive facial speech animation and validated the method through a prototype system built with the techniques described above.

1.5 Organization of the thesis

The remainder of this thesis is organized as follows.

Chapter 2, 3 and 4 expand module 2, 3 and 4 shown in figure 1.2 respectively. Chapter 2 introduces a statistical model for analysis and synthesis of facial expressions from video input. Chapter 3 discusses the issues of facial motion retargeting in the framework of blendshape interpolation. Chapter 4 describes the synthesis of head motion from audio input.

Each chapter, if necessary, begins with a more detailed background for the particular subarea. It then proceeds to describe the methodology, experiment, results, and a summary of the chapter and a discussion of possible extensions.

Chapter 5 put the techniques discussed in the previous 3 chapters together into animation examples to demonstrate the editing pipeline proposed by figure 1.2. Finally, chapter 6 puts the three submodules together into a prototype system that implements the pipeline proposed in figure 1.2 and concludes with discussions of potential future work.

Chapter 2

Facial Expression Analysis and Synthesis

The goal of this chapter is to investigate methods for editing the style of facial motion capture data. Most existing work on reusing facial motion capture data emphasizes editing the content (the audio-to-visual mapping) and the realism of the synthesized face. Not much attention has been paid to the problem of controlling and modifying the facial expression itself. If the actor/actress was originally recorded in a happy mood, all output animations are generated with the same happy expressions. Ultimately, life-like and realistic facial animation needs to cover the entire range of human expression and mood change.

The combination of speech and expression presents a new challenge for analysis methods. A realistic character needs to talk and convey emotions at the same time, instead of talking and then stopping intermittently just to make expressive faces. However, the level of emotion expressed throughout the speech is typically not constant, but varies throughout time, i.e. a happy person does not speak with a constant smile, instead a smile is apparent only at certain moments in a sequence. As one tries to make a certain mouth shape necessary for speaking, the dynamics of the entire facial configuration change depending on the viseme. This variation is crucial to robust expression recognition as well as to facial animation. Most research effort in the area of analyzing facial expressions from video data has been focused on methods for

facial tracking, or recognizing single facial expressions that belong to some predefined category. Little has been done to analyze the *interaction* between facial expression and speech.

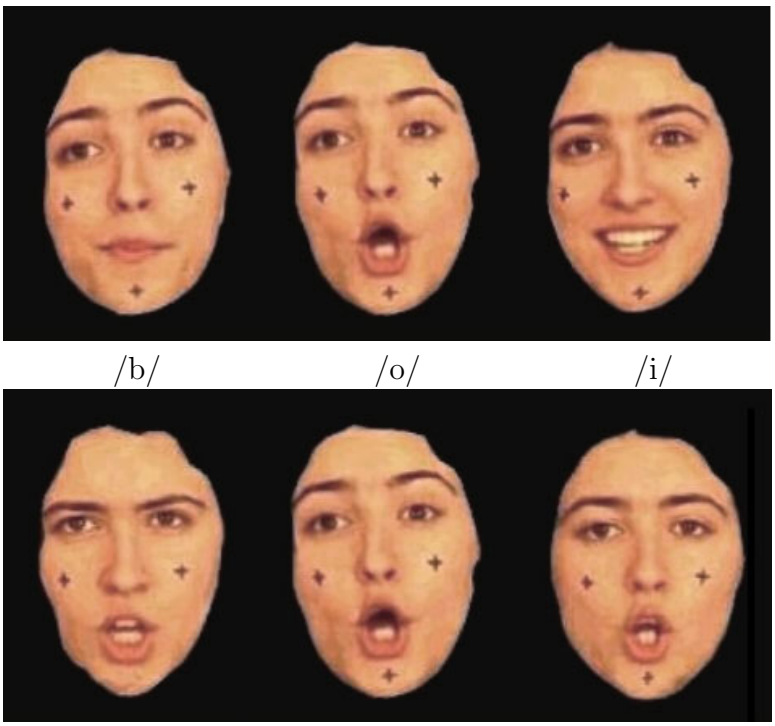


Figure 2.1: Two-factor analysis for expressive facial speech. The top row shows different visemes with the same happy expression. The bottom row shows the same viseme with different expressions. The two factors interact to create complex and subtle facial expressions

We introduce a model that describes such variations by factorizing facial expressions from speech content. Figure 2.1 shows an example of these variations. In the images in the top row, the actress is pronouncing different phonemes, while the facial expression remains happy. In the images in the bottom row, the actress is pronouncing the same phoneme with angry, happy and neutral expressions. While the variation caused by the different phonemes is noticeable from shape information alone, the variation caused by the facial expressions is very subtle and much of it is in the fine details around the eyes and eyebrows. With the factorization model, we are able to

derive the facial expressions from the images and induce different expressions onto existing images.

This chapter is organized as follows. Section 2.1 describes the related work for analyzing facial expressions. Section 2.2 introduces the models we use to parameterize the video images, and separating the factors of content and style in expressive visual speech. Section 2.3 describes the experiment, including tracking, preprocessing, analysis, and synthesis. Finally, the chapter concludes with discussion and future work.

2.1 Background

This section is broken down into two subsections. Section 2.1.1 summarizes related work on parameterizing and analyzing facial expressions. Section 2.1.2 describes statistical models for data analysis.

2.1.1 Facial expression analysis

A great deal of previous research involves the study of appropriate models of facial expression. Most of the effort has been with regard to tracking and recognition of facial expressions. These methods primarily focus on the static or short-term dynamics of facial expressions - the timing of a single facial action, such as a smile or a frown, where the facial configuration starts at a somewhat neutral position and undergo some changes to reach a final state. Facial expressions are often learned and recognized in relation to Facial Action Coding System (FACS) [32]. FACS encodes visually distinguishable facial movements in 46 small units called action units (AUs), which describe qualitative measures such as pulling lip corners, or furrow deepening. A facial expression, such as a smile, is described as a combination of these units. Facial expression recognition typically involve visual tracking of movement of facial features such as corners and contours using various optical flow techniques [66] or transient features such as furrows and wrinkles by observing changes in the texture [91]. In some cases, facial features are parameterized into physical (muscle) mode describing the

underlying facial structure [36, 33, 34]. Sometimes facial images are transformed into different subspaces either for compression or for better classification (e.g. PCA, ICA, LFA, FLD, etc.) [28, 62]. Recognition is achieved by categorizing the parameterized facial features into a set of predetermined motions as described in FACS [66, 97, 84, 9]. Six major facial expressions are commonly used for recognition: happy, anger, sad, disgust, fear, and surprise.

FACS is a great tool for studying facial expression, but there are some known shortcomings when it is used for animating facial speech, including the lack of subtle descriptions around the mouth for speech, and timing information associated with muscle contractions [74]. Even though various extensions of FACS describe the temporal characteristics of facial expressions by templates, or model them explicitly with multi-state representations [19, 62], they are still limited to analyzing single facial expressions such as a smile, or a frown, and are insufficient for animating facial expressions accompanied by continuous speech.

Furthermore, when the estimated facial parameters are applied to animation, the subtle details of the face are often lost. One of the problems is that the goal of facial expression recognition is to distinguish one expression from another. The characteristic details that are unique to individuals are reduced, since they are noise to the classifier and not important for the task of recognition. Unfortunately, these details are very important for animation. Unless sophisticated face models are available for the forward mapping (mapping AU to facial appearance), controlling these parameters for facial animation is often insufficient to create very realistic facial expressions.

Image-based techniques have become more popular recently because they side step this problem. Instead of trying to estimate all these parameters from the data, the image or texture is used directly to recreate the character. Lots of data is collected and animation is generated by re-ordering clips of the original data according to the speech [13, 12, 37, 22]. Following this philosophy, to generate expressive speech animation, the simplest solution would be to collect all possible speech elements spoken in every possible mood, and then apply existing techniques. However, this requires a very large amount of data and does not scale very well. We focus our attention on the effect of facial expression on the talking face and model the interaction between

speech and the expressions *explicitly*, thus reducing the amount of data required and adding controllability to the system.

2.1.2 Statistical models for multi-factor problems

As we can see in figure 2.1, speech and expression interact with each other and affect the facial configuration in a complex manner. Changes in the facial expression can alter the facial features in ways that are often associated with certain speech components. The reverse confusion is also common. The goal of this work is to untangle these two underlying factors.

The task of separating multiple contributing factors in data is often called the blind source separation problem in the machine learning literature. Most existing unsupervised learning techniques for solving this problem consist of the following two categories: the additive factor models [65, 6, 48, 39], and the hierarchical factor models [46, 24, 47]. These models are either insufficiently rich to capture the complex interaction between the factors (additive models), or do not have efficient training algorithms (hierarchical models). For example, Principle Component Analysis (PCA), an additive model, is known to minimize mean square errors in the reconstructions. However, PCA assumes that the factors are uncorrelated, which is unlikely in the case of speech and expression. As a result, multiple components are often equally important for the same cause. In that situation, a common technique is to first apply PCA to subsets of the training data that are known to contribute to the factor of interest, subtract the contribution from the primary component from the data, and iteratively apply the model to the residual data to obtain bases for the remaining components. For example, Reveret et al. [82] applied PCA to different subsets of point positions on the lower face during speech motion to differentiate jaw opening from lip motion like lip rounding and lip closure. A similar technique is used for analyzing facial motion capture data by applying Independent Component Analysis (ICA), a generalized case of PCA. ICA assumes a set of basis vectors that are independent from each other, which is unlikely for speech and expression, as discovered by Cao et al. [16]. Cao applied ICA to expressive speech motion by using the point positions

around the mouth explicitly for calculating the contribution of the speech content. Provided with sufficient domain knowledge, this method can identify the components that are most responsible for the emotions and speech respectively. But since there are multiple factors involved, and the relationship between the factors are not further characterized, it is difficult to generalize the result to new data.

We propose using a bilinear model to describe the interaction between the factors. Bilinear models are two-factor models with the mathematical property of separability, and the outputs are linear in either factor when the other is held constant. Together, the two factors modulate each other's contributions multiplicatively, which allows rich interactions between them. It has been shown that bilinear model successfully separate content and style in several different applications, such as face recognition in different head poses, extrapolating digits for new font styles, etc. [88]. In addition, a bilinear model provides the following properties that are desirable for our task: (1) generalizable, such that it can fit new unseen data and help to minimize the amount of training data required, (2) efficient to train, so that the learning task is simple and tractable, and (3) easily controllable, such that we can modify the parameters in a predicted manner for different applications. The details of the bilinear model will be described in section 2.2.2.

There are challenges in applying a bilinear model to edit facial expressions in existing video data. Since the expression and speech content are sampled from continuous functions, we can not effectively classify the style and content. For each new video frame, content extrapolation is required. As Tenenbaum et al. pointed out [88], extrapolation is more difficult than classification because the model must be judged by visual appearance, rather than solely based upon an abstract error metric. In addition, the solution needs to be close to the model in an absolute sense, as opposed to classification for which a relative measure is sufficient. In other words, the output data needs to lie in the correct subspace for the facial image to be valid. As a result, the choice of data representation and model fitting strategy are very important, and specific to each problem domain. The remainder of this chapter will discuss a facial image model that behaves well in a linear vector space, and the constrained model fitting that guarantees a solution in the correct subspace.

Finally, to use a bilinear model, we must make a simplistic assumption, that at each time instant, the facial configuration is influenced *only by two underlying factors*: a visual speech component and an expression component. This assumption is somewhat limiting. However, it helps us to concentrate on the issue without being distracted. To scale the problem to multiple factors, readers should review the general class of multilinear model, which has been applied successfully to various recognition tasks [64, 93].

2.2 Modeling

The process of modeling consists of two steps. The first step is appearance modeling. Even though an image from a video sequence consists of thousands of pixels, not all possible combinations of pixel values result in a valid image of a face. Therefore, the concept of a facial subspace is very important when working with face images. By representing the image data with an appearance model that describes the facial subspace, it reduces the dimensionality and the data storage. The second step of modeling describes the relationship between expression and the content that contribute to the facial appearance. As mentioned earlier, a bilinear model is used for this task.

2.2.1 Appearance modeling

We represent images of a talking face as a combination of shape and texture in a similar spirit as in Active Appearance Model (AAM) [21]. AAM is similar to Eigenfaces, however, in comparison to EigenFace [92], a representation based in PCA applied only to the pixel intensity of the images, AAM combines the shape and pixel intensity information into one representation. As a result, AAM is more robust in dealing with shape changes. Although earlier we have discussed that PCA is insufficient for modeling factor interaction, it is nevertheless a very effective technique for dimensionality reduction, or compression. Many applications, such as facial recognition, have been quite successful with PCA. For the purpose of synthesis, however, Eigenface is not an ideal representation, since the reconstructed face images may be blurry, or have

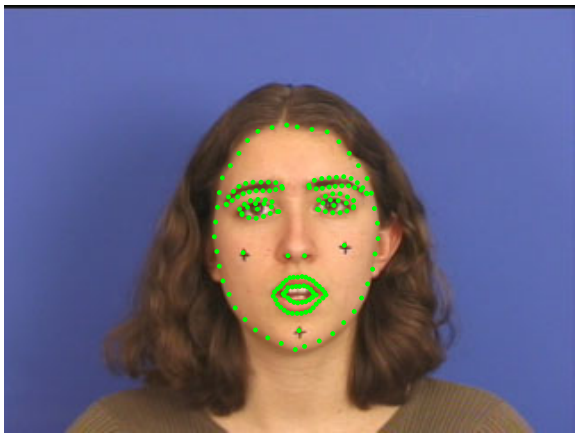


Figure 2.2: Facial features. The x,y location of the facial features, shown in green, dots are used to describe the shape of the face.

ghosting on the edges.

In a shape-texture combined appearance model, shape is represented by a vector of feature locations on the face as shown in green dots in figure 2.2, and texture by the color values of the pixels inside the boundary of the face in the image. The feature points, $[f_1, .., f_n]$, on the face are represented as the vector $\vec{x}$, where $\vec{x} = [f_{x1}, f_{y1}, f_{xn}, f_{yn}]'$. To reduce the dimensionality, Principal Component Analysis (PCA) is applied to the data from all the frames in the video. Each example shape can now be approximated using:

$$\vec{x} = \vec{x_0} + \mathbf{P}_s \cdot \vec{b_s} \tag{2.1}$$

where $\vec{x_0}$ is the mean shape vector, $\mathbf{P}_s$ is the set of orthogonal modes of variation derived from the PCA, and $\vec{b_s}$ is a set of shape parameters.

The dimensionality of the texture data is reduced via a similar method. To build a statistical model of color appearance, we warp the images to be aligned with the mean shape using a triangulation-based image warping technique, and then sample the color values from the shape-normalized image to form a vector $\vec{g}$. Similarly to the shape model, we apply PCA to $\vec{g}$, where

$$\vec{g} = \vec{g_0} + \mathbf{P}_g \cdot \vec{b_g} \tag{2.2}$$

where $\vec{g_0}$ is the mean gray-level vector, $\mathbf{P}_g$ is a set of orthogonal modes of variation and $\vec{b_g}$ is a set of color parameters.

The shape and appearance of the face can thus be summarized by the vectors $\vec{b_s}$ and $\vec{b_g}$. The facial configuration vector at any given frame is defined as

$$\vec{y} = \begin{bmatrix} B_s \cdot \vec{b_s} \\ \vec{b_g} \end{bmatrix} \tag{2.3}$$

B_s is a diagonal matrix with weights for the shape parameters, allowing for a difference in units between shape and gray values. Details for computing the weights can be found in [21].

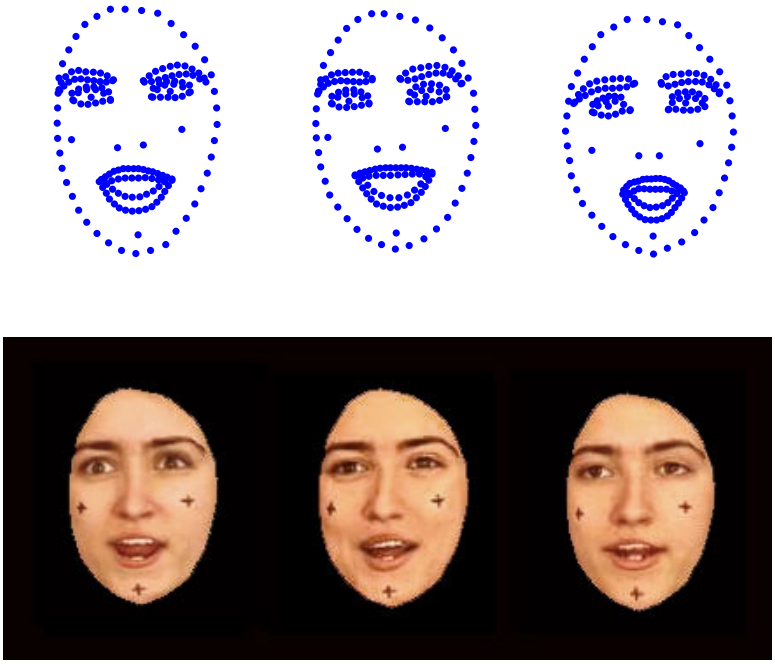


Figure 2.3: Representing facial appearance: shape vectors (top) and shape normalized texture (bottom) for different facial expressions. From left to right: angry, happy, and neutral. Notice the subtle differences in the texture around the eyes and cheeks convey a lot of information.

Figure 2.3 shows the shape vectors and shape normalized texture for three different facial expressions: angry, happy, and neutral. Notice that although the shape

alone conveys some differences between the facial expressions, much of the information is differentiated more robustly by human observers in the subtle details of the texture. This simple illustration shows that both shape and texture are important in representing complex facial expressions.

2.2.2 Bilinear model for style and content

Bilinear models are two-factor models with the mathematical property of separability. They can be described as

$$y_{k,s} = \vec{e_s}' \cdot \mathbf{W}_k \cdot \vec{c} \tag{2.4}$$

where $y_{k,s}$ is the k^{th} component of the facial vector, as shown in equation 2.3, with a particular style s. $\vec{e_s}$ is the expression vector of style s, $\vec{c}$ is the content vector, and $\mathbf{W}_k$ is the basis matrix that describes the multiplicative relationship between the style and the content. For K facial components, $\vec{y}_s = [y_{1,s}, ..., y_{K,s}]'$, we will need $\mathbf{W} = [W_1, ..., W_K]$. In our example, the style is the facial expression, and the content is the visual content of the speech, or the viseme.

Equation 2.4 is called a *symmetric* bilinear model, where the content and the style are described separatively and free to vary, while the basis matrix $\mathbf{W}_k$ stays fixed. In some applications, however, it is useful to have *asymmetric* bilinear model, described as

$$\vec{y}_s = \mathbf{B}_s \cdot \vec{c} \tag{2.5}$$

Here $\vec{y}_s$ is the facial vector show in equation 2.3 (instead of one of the components in the vector as in the symmetric case). $\mathbf{B}_s$ is a style, or expression, specific basis matrix, and $\vec{c}$ is the content vector. Conceptually, the relationship between the symmetric and asymmetric model can be described as $\mathbf{B}_s \simeq [\vec{e_s}' \cdot \mathbf{W}_1, ..., \vec{e_s}' \cdot \mathbf{W}_K]'$. In practice, this equality depends on the number of dimensions retained for the style and content vectors, as will become clear in the details of model training in section 2.3.2.

The symmetric model is very general and flexible, however, more difficult to train and apply. The asymmetric model is less flexible, but very easy to compute, and thus useful in applications where one of the factors is known in advance.

2.3 Using the model

After Institution Review Board approval and informed consent, we recorded video sequences of an actress speaking with three different expressions: happy, angry, and neutral. The text contains four phrases, each takes about 3 to 4 seconds to complete, and covers the major groups of visemes in English [69]. The actress is advised to keep her head steady, as the appearance model we use is inherently a 2D method. The out-of-plane motion will be considered noise in this representation. Another additional source of noise that causes problems is blinking. Since it occurs at random intervals and has little correlation to either expressions or speech, we remove the frames that contain blinking from the training set. To obtain the facial features shown in figure 2.2 for the appearance model for an entire sequence of video without painstakingly labelling them, a good tracking method is required. The first step in the experiment is tracking, then followed by some data processing and the model training. Finally, the model can be used to analyze new video data, and to modify the expression for the new data.

2.3.1 Facial feature tracking

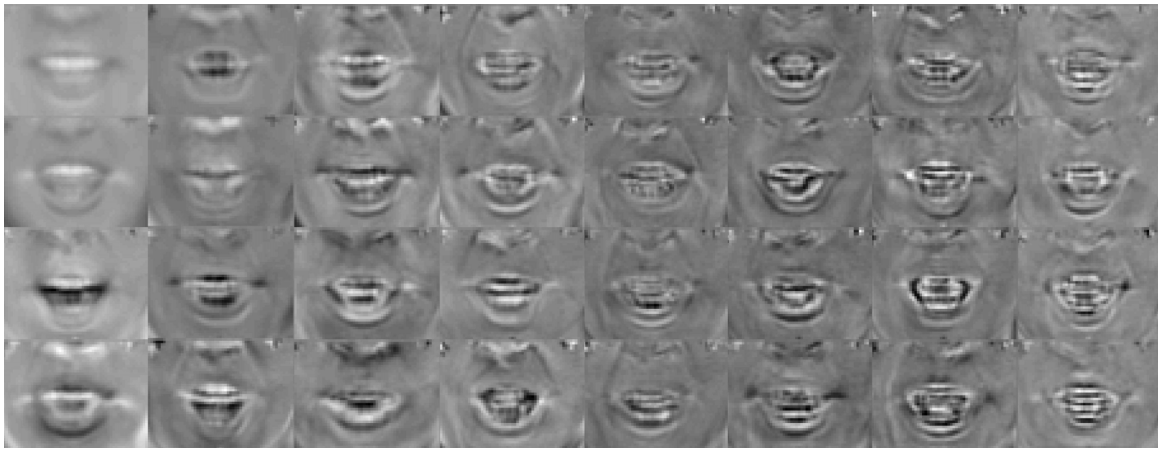


Figure 2.4: Eigenlips: orthonormal basis for the lip images

We implemented an appearance model based tracker related to EigenTracking [8]

for tracking facial features as shown in figure 2.2. The algorithm can be summarized as follows. First we assume the appearance of a nonrigid deformable object can be represented by a linear combination of a number of basis images, reprented by a set of orthonormal vectors. Figure 2.4 shows an example of the basis for the lip area. The object in motion undergoes both affine deformation (translation, rotation and shear) and nonrigid deformation, which can be represented by the six affine parameters and coefficients for the orthonormal vectors. Tracking the object is formulated as an optimization problem that searches for the affine parameters and basis coefficients that best fit the appearance of the object in the new frame. The error function is the square difference between the target image and the updated warped image. Figure 2.5 shows an example of lip tracking using this method. The image on the left is the affine warped version of the original image. The center image shows the reconstructed image using the basis, and the image on the right shows the lip contour points tracked.

To obtain the image database to train the linear model used for tracking, a handful of images from the sequence are labeled with contour points. These images are chosen to cover a wide variety of nonrigid object deformations, e.g. extreme lip opening and closure. A typical number of images to label ranges from 20-30 for the lip area. Then, each labelled image is morphed with every other labeled image to enlarge the training set of the eigenbase.

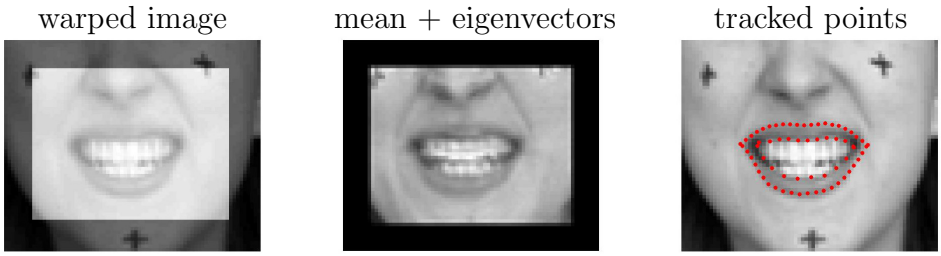


Figure 2.5: Facial feature tracking. Left: original image affine warped. The light gray area in the middle shows the mask for the area of interests. Center: reconstructed image of the lip region using the eigen basis shown in figure 2.4. Right: resulting lip contour marked in red dots.

2.3.2 Model Training

Given training data that consists of neutral, happy, and angry expressions, and frames 1 through T, we first represent the facial data as facial configuration described in equation 2.3. The number of PCA coefficients used retains 99% accuracy for the shape, and 80% for each of the three color channels. The input vectors are stacked in the vertical direction for the different expressions, and horizontally for the different frames (content) to form a single matrix Y.

$$\mathbf{Y} = \begin{bmatrix} \vec{y}_n(1) & ... & \vec{y}_n(T) \\ \vec{y}_h(1) & ... & \vec{y}_h(T) \\ \vec{y}_a(1) & ... & \vec{y}_a(T) \end{bmatrix} \cong [\mathbf{W}^{VT} \cdot \mathbf{E}]^{VT} \cdot \mathbf{C} \tag{2.6}$$

This matrix formulation requires common content vectors from different expressions to be in the same columns. However, people generally speak at different tempos for different facial expressions, thus correspondence cannot be directly established in the raw video frames. We solve for this correspondence by temporally aligning the audio speech signal to match the video frames, since visual speech components are likely to be correlated with the audio speech. The speech signal is processed using Relative Spectral Perceptual Linear Prediction (Rasta-PLP) [45]. Dynamic time warping, commonly used in speech recognition [81], is then applied to align all recordings of the same speech content, captured with different facial expressions. After warping the sequences, correspondence can be established between frames in different expressions. Unfortunately, even though the audio and video signals are highly correlated, the mapping is not exactly one-to-one and there can still be some timing discrepancies after the warping. This problem is especially severe before the onset of the plosive consonants, where the exact timing of the lip closing is not clear according to the audio signal, thus manual correction is sometimes needed at these boundaries. We used the neutral speech as the reference and align happy and angry speech to the neutral. After the alignment, we are left with 386 frames, or a little over 12 seconds of video, for the training set.

The bilinear model decomposes this matrix into three sub-matrices.

$$\mathbf{Y} \cong [\mathbf{W}^{VT} \cdot \mathbf{E}]^{VT} \cdot \mathbf{C} \tag{2.7}$$

for the symmetric bilinear model.

where $\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \vdots \\ \mathbf{W}_k \end{bmatrix}$, $\mathbf{E} = [\vec{e}_n, \vec{e}_h, \vec{e}_a]$, $\mathbf{C} = [\vec{c}(1), \cdots, \vec{c}(t)]$.

Following the notation in equation 2.4, $\mathbf{W}$ is the basis matrix that describes the relationship between the style and the content for each component of the K dimensional facial vector. $\mathbf{E}$ is the style matrix containing the expression vectors for neutral, happy and angry. And $\mathbf{C}$ contains the content vectors for the training data for frames 1 through T.

For the asymmetric bilinear model, the decomposition is

$$\mathbf{Y} \cong \mathbf{B} \cdot \mathbf{C} = \begin{bmatrix} \mathbf{B}_n \\ \mathbf{B}_h \\ \mathbf{B}_a \end{bmatrix} \cdot \mathbf{C} \tag{2.8}$$

where the $\mathbf{B}_n$,$\mathbf{B}_h$, and $\mathbf{B}_a$ are style specific basis for neutral, happy and angry.

The $\cong$ sign is used in both decompositions instead of the equal since there is dimensionality reduction during the training steps. $\mathbf{X}^{VT}$ is the *vector transpose* of $\mathbf{X}$. A vector transpose is a way to stack the matrix where the two factors involved switch roles. For instance, the vector transpose for $\mathbf{Y}$ would be

$$\mathbf{Y}^{VT} = \begin{bmatrix} \vec{y}_n(1) & \vec{y}_h(1) & \vec{y}_a(T) \\ \vdots & \vdots & \vdots \\ \vec{y}_n(t) & \vec{y}_h(t) & \vec{y}_a(T) \end{bmatrix}.$$

The training algorithm minimizes the total squared error for the reconstruction of the training set. For asymmetric bilinear model, training can be accomplished by

simply performing a singular value decomposition (SVD) on matrix $\mathbf{Y}$.

$$\mathbf{Y} = [\mathbf{U} \cdot \mathbf{S}] \cdot \mathbf{V}' \cong \mathbf{B} \cdot \mathbf{C} = \begin{bmatrix} \mathbf{B}_n \\ \mathbf{B}_h \\ \mathbf{B}_a \end{bmatrix} \cdot \mathbf{C} \tag{2.9}$$

The style-specific basis matrices $\mathbf{B}_n$, $\mathbf{B}_h$, and $\mathbf{B}_a$ have dimension K by J, obtained by taking the first J columns of $[\mathbf{U} \cdot \mathbf{S}]$. The content matrix $\mathbf{C}$ is the first J row of $\mathbf{V}'$ contains the content vectors for time 1 through T. The number J is the intrinsic dimensionality of the contents. Ideally, the size of J can be judged by the magnitude of the singular values on the diagonal entries of $\mathbf{S}$. The J-th entry where the magnitude drops drastically tells the true dimensionality of the data. Unfortunately, in reality this cutoff is rarely easy to find due to noise in the data, and the magnitude of the singular values usually decreases more gradually. Therefore, in practice the value J is often found experimentally.

For the symmetric model, we need to solve the content vectors, $\mathbf{C}$, the style vectors, $\mathbf{E}$, and the basis matrix, $\mathbf{W}$, (shown in equation 2.7) simultaneously. This is done by solving the content and style vectors iteratively in a similar fashion shown in the training of the asymmetric model, by fixing one and solving for the other, and alternating until the vectors converge. The SVD shown in equation 2.9 serves as the initialization for the content vectors. Following some matrix manipulation, we can solve for the expression vectors by performing another SVD. For the details on matrix manipulation, readers should refer to [41].

The result of the training include an I by 3 expression matrix $\mathbf{E} = [\vec{e}_n, \vec{e}_h, \vec{e}_a]$, a J by T content matrix $\mathbf{C} = [\vec{c}_1, ..., \vec{c}_t]$, and a set of K basis $\mathbf{W}_i$ for each of the component in the face vector, each with dimension I by J. In equation 2.9, we reduce the dimensionality for the contents sub-matrices to J. Similarly, we reduce the dimensionality of the expression sub-matrices to I. Again, the value I and J are theoretically the intrinsic dimensionalities and can be found by the cutoff in the singular values, but in practice they are often found experimentally. In our experiment, I is usually 3, and J is typically between 8 and 10.

2.3.3 Modifying facial expression

Given a new test image, represented by the facial configuration vector $\vec{y}_{test}$, we would like to analyze it and perform some modifications. Three examples are shown:

1. **Known expression** : Test image with known facial expression and unknown content. Change the facial expression while keeping the speech content.

2. **Unknown expression** : Test image with unknown facial expression and content, but the facial expression is represented in the training data. Change the facial expression while keeping the speech content.

3. **Untrained expression** : Test image with unknown facial expression and content. The facial expression is not represented in the training data. Change the facial expression while keeping the same content, and then extrapolate different content with the new expression.

Known expression In the first example, we assume we know the facial expression for the input image, and we would like to obtain the content parameters $\vec{c}$ and change the expression. This can be done via least square fit to the asymmetric bilinear model by using the appropriate expression-specific basis matrix $\mathbf{B}_s$ found from the training set.

$$\vec{c}(t) = (\mathbf{B}'_{known} \cdot \mathbf{B}_{known})^{-1} \cdot \mathbf{B}'_{known} \cdot \vec{y}(t)_{test} \tag{2.10}$$

To extrapolate the new expression while keeping the same content, simply multiply the content vector with the basis matrix for the new expression.

$$\vec{y}(t)_{new} = \mathbf{B}_{new} \cdot \vec{c}(t) \tag{2.11}$$

The resulting vector is then re-projected onto the shape and texture PCA, re-warped, to obtain the final image. Figure 2.6 shows the result. On the left are the input images in the happy and angry expression. On the right are the reconstruction of the input image and the extrapolated expressions. As mentioned in the previous section, the dimensionality of the content vectors, J, is determined experimentally. When J is

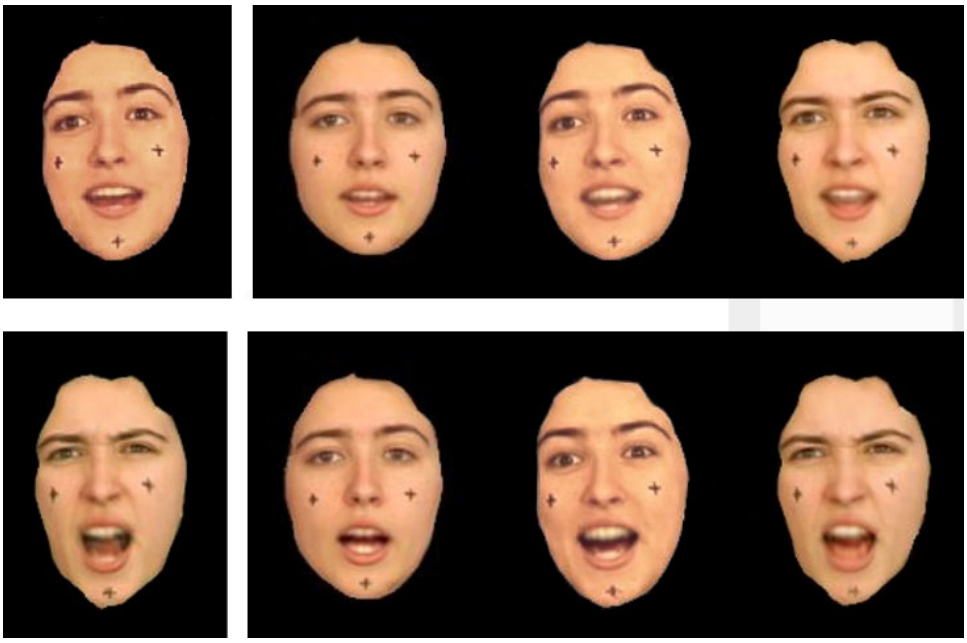
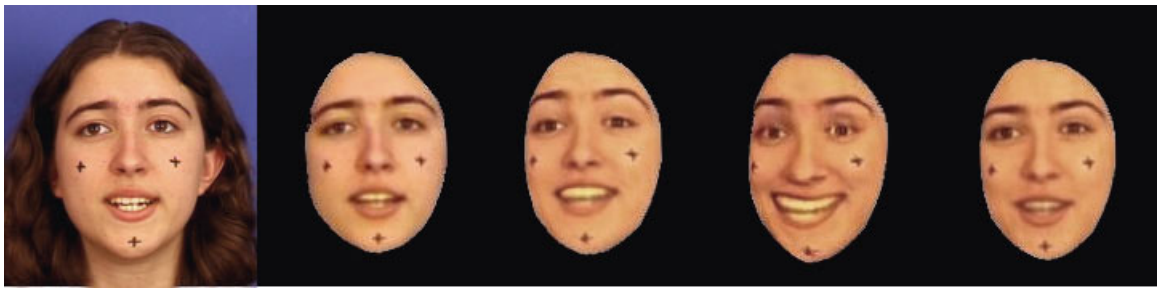


Figure 2.6: Changing facial expression using asymmetric bilinear model. On the left are the test images in happy and angry expression. On the right are the reconstructions of the input expressions and the extrapolated expressions using the bilinear model. See video figure 2.6 for the entire sequence.

too large, the least square solution of equation 2.10 is very sensitive to noise. While the reconstruction of the original test image looks good, the noise gets amplified when extrapolated to a new expression, and the result often does not even look like a normal face. When J is too small, it can not represent the richness of the data and the result converges to the average content. Figure 2.7 shows the examples where the dimensions are not chosen properly.

Unknown expression The second example is to take a face image and extract both the expression and the content. This is more difficult since the two factors interact with each other in a complex manner. For this task we use the symmetric bilinear model. We follow the iterative scheme by solving for one factor, while keeping the other factor fixed. Given the input data $\vec{y}_{test}$ and an initial guess for the content



(a) (b) (c) (d) (e)

Figure 2.7: Dimensionality for the content vector (J). (a) original test image (b) reconstructed test image (c) modified expression to happy (d) when J is too large. (e) when J is too small, the content converges to the average.

vector $\vec{c}_0$ from equation 2.4, the expression vector is

$$\vec{e_s}' = [[\mathbf{W} \cdot \vec{c}_0]^{VT}]^{-1} \cdot \vec{y}_{test} \tag{2.12}$$

Similarly, the content vector is

$$\vec{c}_0 = [[\mathbf{W}^{VT} \cdot \vec{e_s}']^{VT}]^{-1} \cdot \vec{y}_{test} \tag{2.13}$$

where $\mathbf{X}^{-1}$ is the pseudo inverse of $\mathbf{X}$. Analyzing images using the symmetric model is more difficult. Since the problem is under-constrained, if the new image is noisy, the alternating scheme in equation 2.12 and 2.13 sometimes does not converge to the correct local minimum. Additional constraints can be applied to ensure the convergence to the correct local minimum. This can be achieved by observing the fact that the first element of the content vector, derived from the training data, stays within a range over time. We can enforce that for the first few iterations by re-initializing the first component of the content vector to be the mean of the training data, to ensure convergence in the right direction. The result for this task is shown in figure 2.8. Figure 2.10 shows the first 5 content coefficients for 30 frames after 2,4,6,8, and 10 iterations. Figure 2.11 shows the style vectors and the square error for these 30 frames for the first 50 iterations. Notice that they style and content quickly converge after about 12 iterations.

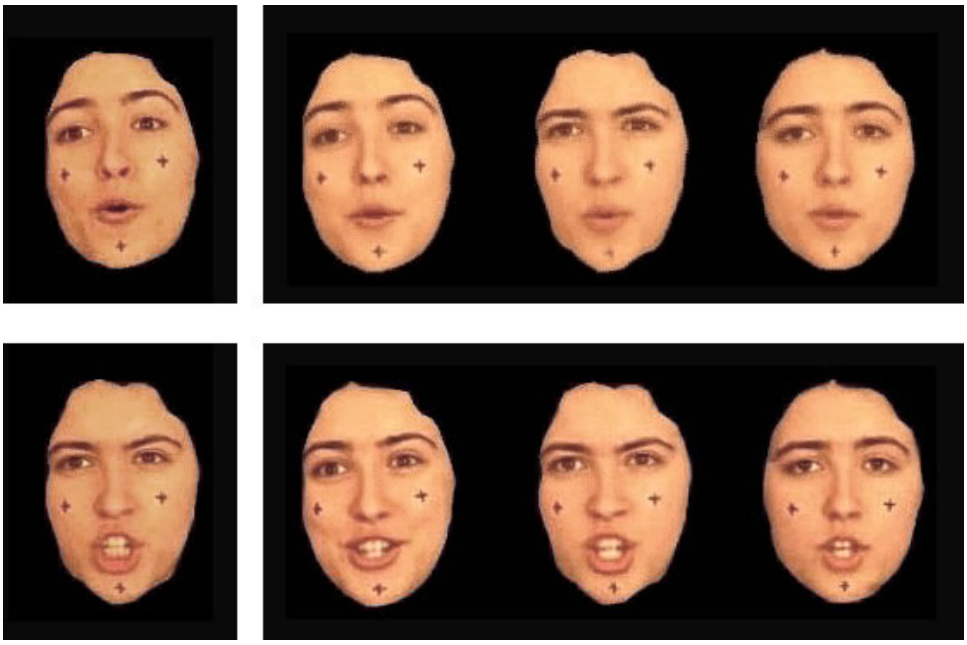


Figure 2.8: Changing facial expression using symmetric bilinear model. On the left are the test images in happy and angry expression. On the right are the reconstructions of the input expressions and the extrapolated expressions using the bilinear model. See video figure 2.8 for the whole sequence.

Untrained expression The final example is to take a face image of completely unknown expression, and extract both the expression and the content components. This task is called translation, and is more challenging because it has to extrapolate in both directions. The image in the top left corner of figure 2.9 shows an image of a sad face. The left column shows the unknown speech content translated to a previously known expression, while the top row shows the new expression translated to other contents. The middle images fill in the results for the respective categories.

The data used in the bilinear model is already in a compressed PCA form. A test image that goes through this compression already has error, which is later accumulated after fitting to the bilinear model. To determine the extent of influence that PCA compression and bilinear fitting have on the result, we compared using only the training images for training of the PCA models, and using both the training and test images for training of the PCA models. The difference is small, indicating that the

Figure 2.9: Untrained facial expression as input. The top left corner is the novel image with unknown style and content. The top row shows the new style translated to known content. The left column shows the new content translated into known style.

PCA model is sufficient to cover the variations in the data.

2.3.4 Adding temporal constraints

So far, the algorithm has not consider the temporal factor. In fact, since the frames from 1 through T are treated equally, it makes no difference if all the frames are shuffled, either in the model training stage, or in the analysis and synthesis stage. Since there is no temporal coherency in the least square solution, the resulting motion is often very jerky. A few simple solutions can help to address this issue. The first solution is applicable in the case that neither expression nor content are known (using symmetric model). Instead of solving for the content and expression parameters in the test sequence frame by frame as show in equation 2.12 and 2.13, we can solve for N frames at a time by assuming the expression vector does not change within these N frames. This is done by simply replacing $\vec{c}_0$ in equation 2.12 with $C = [\vec{c}_0, \vec{c}_1, ..., \vec{c}_N]$. This is valid if we make the assumption that facial expression changes at a slower rate than the frame rate.

Another simple way to enforce temporal constraints is to explicitly treat n frames as a bundle. For example, take n=3, the input configuration vector in equation 2.3 becomes

$$\vec{y}(t)' = \begin{bmatrix} \vec{y}(t-1) \\ \vec{y}(t) \\ \vec{y}(t+1) \end{bmatrix} \tag{2.14}$$

When analyzing a new sequence, a sliding window of n frames is fitted to the model. To construct the output sequence, only the middle part of the configuration vector $\vec{y}(t)'$ is used to produce the image. The resulting animation from this modification appears to be a lot smoother. Figure 2.12 shows an example where the input sequence is happy expression. Figure 2.12(a) shows an example when there is no frame-to-frame constraint for the content vectors. The left column is the reconstruction for the input expression. The middle column shows the extrapolated angry expression, and the column on the right shows the extrapolated neutral expression. Note that in the 3rd

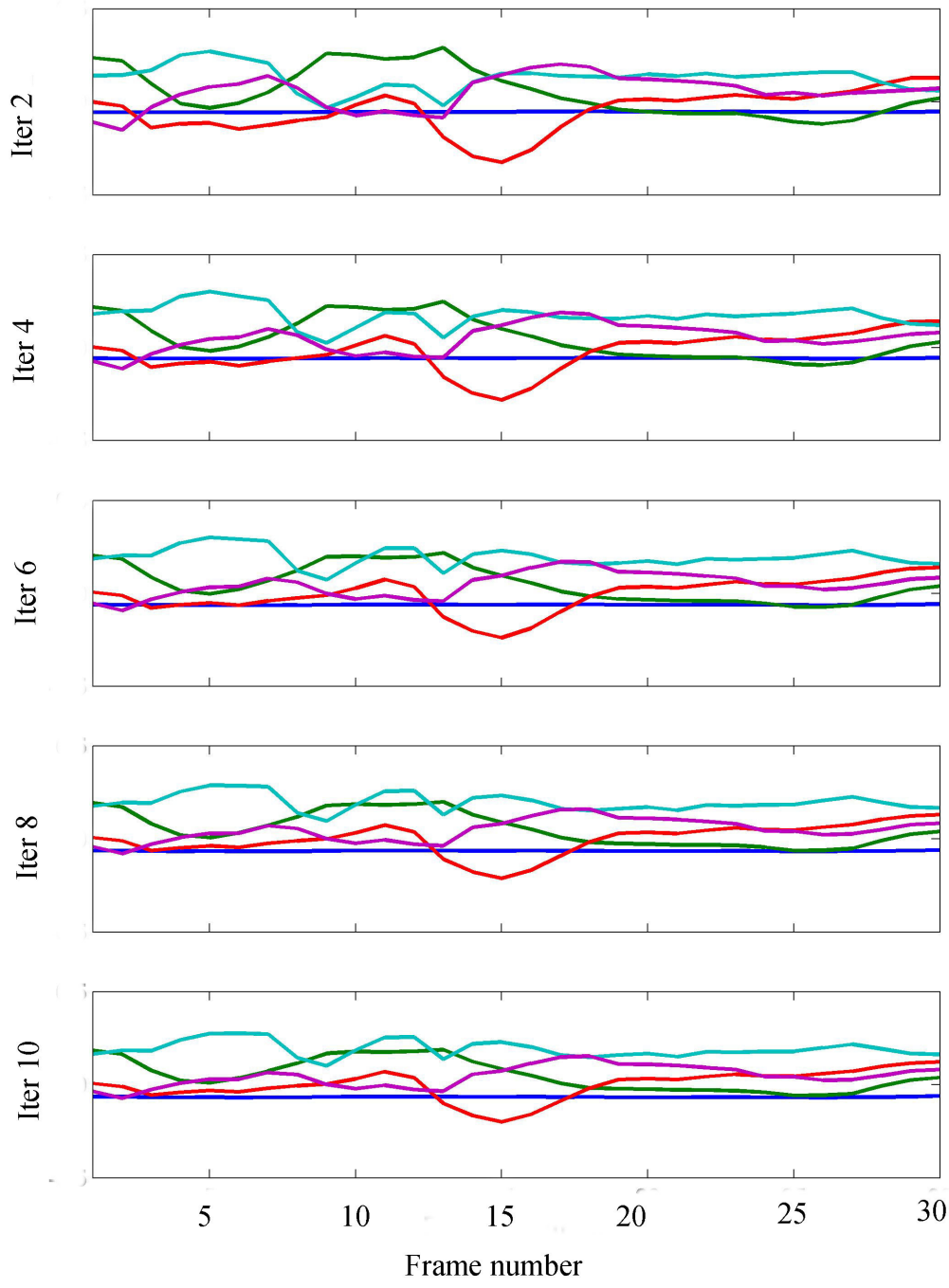


Figure 2.10: Convergence for the first 5 coefficients of content vectors after 2, 4, 6, 8, and 10 iterations, as shown from top to bottom.

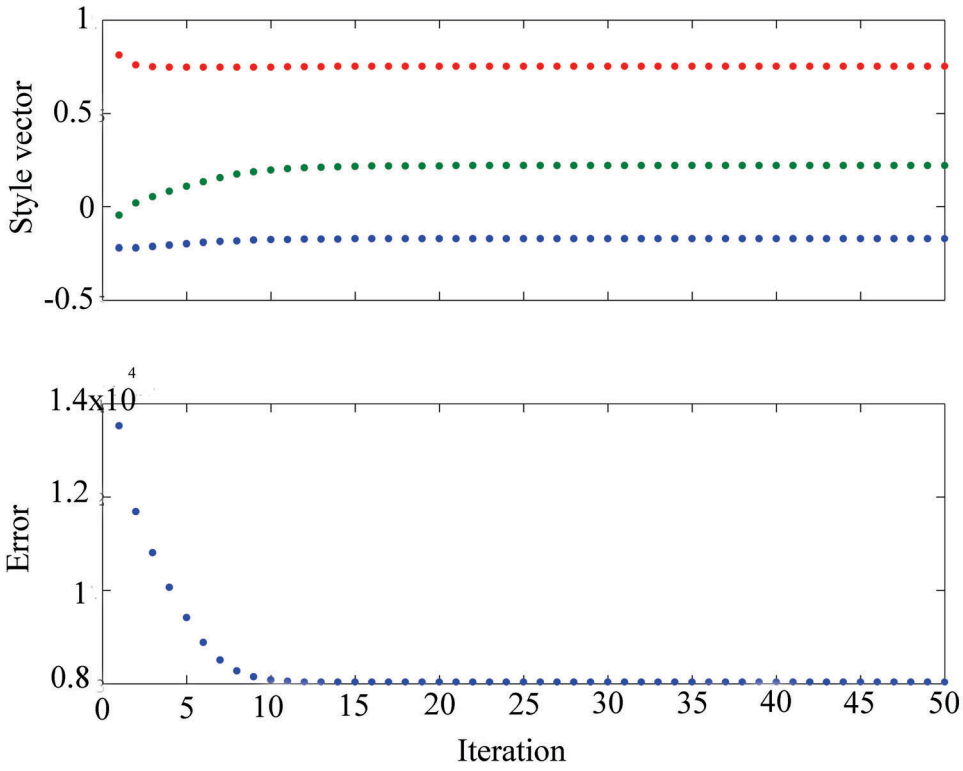


Figure 2.11: Convergence for the style vectors. Top: convergence of the style vector for the first 50 iterations. Bottom: squared reconstruction error. The convergence is reached after 12 iterations.

frame from the top (highlighted in the yellow box), the facial configuration changes drastically from the neighboring frames above and below. Figure 2.12(b) shows the result with 3 frames constraints. The corresponding frames are regularized such that the frame-to-frame motion appears to be smoother.

There is a tradeoff for the number of frames n. The larger n is, the smoother the motion is. However, the essence of the motion, the extremes like the pops and the jerks, are lost. The worst case is around the plosive consonant, where the lips no longer close tight. This artifact is actually quite common in all of the image and learning based facial animation methods. In our examples, we typically do use a 3-frame constraint.

(a) no temporal constraint (b) with temporal constraint

Figure 2.12: Adding temporal constraint. (a) The facial configuration changes drastically from the neighboring frames above and below for the frame highlighted in yellow box. (b) Three frames are grouped together as in equation 2.14. The corresponding frames are regularized such that the frame-to-frame motion is smooth. See video figure 2.12 for the entire sequence.

Continuous space of facial expressions The space of facial expression is continuous, and the in-between expressions are very important. Using the bilinear model, the in-between expressions can be represented very efficiently. Simply apply a weight, α, specifying the amount of each target expression,

$$\vec{y}_i = (\alpha \cdot \vec{e}_{s1} + (1 - \alpha) \cdot \vec{e}_{s2}) \cdot \mathbf{W}_i \cdot \vec{c} \tag{2.15}$$

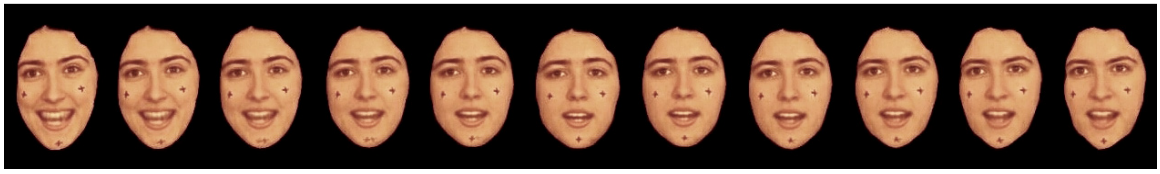


Figure 2.13: Interpolation between the expressions. The leftmost, center, and the rightmost image are the original happy, neutral and angry expressions, while the rest are interpolated.

Figure 2.13 shows the interpolation from happy to neutral, and from neutral to angry. If α is negative or greater than 1, then the facial expression is effectively exaggerated. For example, figure 2.14 shows an exaggerated angry expression, where $\alpha = 1.5$ for the angry vector. It is important that the magnitude of α remains close to 0 and 1. If the magnitude is too large, a non-valid face image can be produced since the re-projected face is no longer in the correct facial subspace.

Figure 2.14: A exaggerated expression by extrapolation.

2.4 Summary and future work

In this chapter, we show that the relationship of facial expressions and visual speech can be described by a bilinear model. The bilinear model encodes sufficient expressiveness to capture the range of variations and is able to extrapolate new styles and contents from limited observations, thereby requiring relatively little training data compared to other models in a similar class. Two formulations of bilinear models are fit to facial expression data. The asymmetric model derives style specific parameters, and is used to analyze a novel video sequence of known expression in order to recover the content. Together with the style-specific parameters for other expressions, new sequences of different expressions can be extrapolated. The symmetric bilinear model derives a set of parameters independent of content and style. To apply the bilinear model to facial expression data, care needs to be taken to ensure that the output results in valid facial images. This is achieved by representing input images with an Active Appearance Model (AAM), and applying spatial constraints in the iterative solution. Finally, a temporal constraint is applied to ensure smooth transitions between the resulting frames.

There are some possible extensions that can be added to this framework. For example, in this work, the content and expression factors are not applied symmetrically, i.e. we specify the expression at a higher level since we know the specific expressions used in the training data. Similarly, if we pre-analyze the phonetic content of the training data, we can also directly control the content for the speech. This is useful for combining content and expression editing into a single framework.

The techniques presented here are 2D. Given the advances in 3D capture and modeling technology, it may be possible to extend this framework to 3D data. One of the limitation to the 2D approach is that the actor or actresses can not turn their heads while speaking. This restricts natural acting. We could lift this restriction if 3D data were used.

There are limitations to the technique presented in this chapter, however. First, the expressiveness of the bilinear model can serve as a double-edged sword. While it captures the styles in the data, it can also capture unintended factors or noise, and

extrapolate that noise in the new style. For example, if the raw video data has a different color balance for each expression, the color variation can be interpreted as part of the expressive style, and exaggerated when a new facial expression is derived. This creates a strict requirement for carefully controlled data collection to ensure that no undesirable factors are present in the aligned data. Furthermore, the bilinear model has limited ability to represent the temporal characteristic of facial expressions, i.e. the timing of all sequences has been normalized. As the tempo of speech is an important indicator of mood, we need to investigate methods for including expression specific timing information in the synthesis model.

Chapter 3

Facial Expression Retargeting

The goal of this chapter is to investigate methods for using motion capture data to create facial animation. This is sometimes called facial motion retargeting. The term 'motion retargeting' was first described by Michael Gleicher [40] for articulated motion to imply that the source data and the target model may not have the same physical properties, such as size, proportion, or degrees of freedom. We do not restrict the definition of motion capture to be just the data obtained from markers, but broadly refer to all motion recorded from live action. Figure 3.1 shows an example illustrating retargeting a facial expression to a 3D face model.

Figure 3.1: Facial motion retargeting

There are a wide variety of existing methods for retargeting motion capture data which vary with respect to the format of the raw motion data, how much the motion

data is analyzed, and what type of facial parameters are obtained. These methods can be categorized according to whether they parameterize motion explicitly or implicitly.

Retargeting with *explicit parametrization* is by far the most common, and it seems every conceivable parametrization has been used for retargeting. These methods use a representation that falls along the continuum introduced in figure 1.3, from appearance based to structural. The core method in all cases relies on the fact that the chosen parametrization is in meaningful correspondence in the source and target models, so that translation from source to target can occur. Since raw motion capture data is nearly always at the appearance end of the spectrum, these methods often include sophisticated analysis modules for translating data from appearance based to structural before retargeting.

Many methods make no attempt to analyze the appearance-based motion data but directly apply the motion capture data to the corresponding features on the target model. The motion data typically consists of marker positions, either coarsely positioned on important muscle locations on the face, or densely covered the entire face [42]. Three major steps are involved in the retargeting process. First, the correspondences between the source and the target are established. Next, the displacement vectors from the source data are normalized both in the orientation and in the magnitude to match the shape and proportion of the target model. Finally, a scatter data interpolation method is used to compute the displacement of the remaining vertices, since the target model often has more vertices then the number of markers available. Various techniques have been proposed that use different warping functions to produce the smoothest result for the intermediate vertices [63], Noh et al. introduced a method to automatically find dense correspondences of feature points from a given set of coarse correspondences [67].

Other methods analyze the motion and parameterize it explicitly somewhere between appearance and structure. These parameters usually have values associated with physical appearance of the face, such as the amount of eye opening, eyebrow raising, mouth opening, etc. [15, 72, 68, 44]. The target face must have a corresponding set of parameters and methods for deforming the mesh for any given set of parameter values. Since the facial proportion of the source face may be different from

that of the target model, the values for the parameters are re-normalized during the retargeting.

Lastly, it is possible to analyze the motion data to obtain parameters for the underlying structure in the face. A wide range of facial parameters of varying degrees of complexity can be used. For instance, one may decide to use muscle-based parameters, but build an additional layer of mapping, such as those described by FACS [32], to relate the muscle geometries to the motion of the skin surface. Another approach is to build a complex physically-based model that resembles the performer, and estimate the muscle contractions in order to deform the mesh to match the motion data. To retarget, the target model typically has matching complexity such that given the muscle contraction parameters from the source data, the target model can simulate the same motion [51, 89].

Techniques which rely on *implicit parametrization* analyze the motion data in terms of the relation between a number of basic shapes, often called morph targets, or keyshapes. Each frame in the source data is a blendshape interpolation (shown in figure 1.4), which is a weighted combination of the morph targets. The target model must have a *corresponding set of keyshapes*. Given the relative keyshape weights for a given frame in the source data, it is assumed that these weights can be applied directly to the corresponding keyshapes in the target model to produce an analogous facial pose [77, 56].

Key challenges for blendshape retargeting include methods for tracking in a form compatible with blendshapes, ensuring that source and target models are indeed in correspondence, and choosing an appropriate set of keyshapes.

Pighin et al. [76] introduced a method for building a photo-realistic 3D model of a person from a number of photographs of basic expressions. This model produced very convincing facial expressions using a linear combination of these hand chosen basic expressions. Keyshapes for additional individuals were constructed using the same method and underlying parameterization, thus ensuring the needed correspondence. Further, the choice of a photo-realistic model allows tracking to be performed directly in the space of keyshapes by finding the best fit between the rendered model and an observed video sequence of the same individual [77]. Building a photo-realistic

3D model requires a great deal of labor, and this method unfortunately requires a corresponding model be built for each individual from whom source data is captured, as well as for each desired target appearance.

Kouadio et al. [56] proposed a method for producing real-time facial animation by mapping motion captured data to a bank of predefined facial expressions. Their method employs a two part retargeting procedure. Correspondences are established between points on the motion source and a sparse set of vertices on the target model. Retargeting from the space of motion capture to the desired model is achieved using this explicit parameterization. Rather than use some form of scattered data interpolation to determine the positions of the remaining mesh vertices, as is done in most explicit parameterization retargeting methods, they follow with an implicit retargeting stage. This implicit stage treats the sparse vertices as the source motion, and retargets onto the complete set of dense mesh vertices. Since the implicit retargeting stage occurs between sparse and dense versions of the same models, correspondence between source and target keyshapes is ensured.

Implicit retargeting provides a great deal more flexibility that does explicit retargeting. In principle, this technique can be used even when source and target have very different topology, something that is difficult to achieve with explicit parameterization. For example, consider retargeting a face onto a sack of flour. Explicit parameter correspondence would be difficult to achieve, but an artist may nevertheless be able to implicitly specify a corresponding set of keyshapes.

Additionally, implicit retargeting is very intuitive for animators since it coincides with the most popular way to animate a face, i.e. setting the weights which specify an interpolation between a set of facial blendshapes [73]. Most commercial computer animation tools provide for this method of animation. *Blendshape retargeting* is the reverse process in that instead of having the artists specify the weights for each basic expression frame by frame, the weights for the animation are derived automatically from the motion data.

This thesis builds on existing implicit retargeting methods, resolving a number of previous difficulties. We desire a method that is flexible enough to deal with different motion capture sources. Previous methods either required building a complex

model for every actor and actress, or resorted to a stage of explicit parameterization retargeting. In either case it is not possible to easily make use of motion data that was obtained from some unexpected external source. This thesis shows that existing methods can in fact be used to retarget from any motion source to any target model. In addition, previous methods rely on the user to choose an appropriate set of keyshapes. This thesis presents an comparison of several methods for automatically choosing a set of keyshapes given an example sequence of motion data.

Although most previous retargeting methods are designed to be entirely automatic, this thesis intentionally keeps the artists in the loop by having them create corresponding target keyshapes, instead of relying on some pre-defined mapping. To make characters more interesting artists often exaggerate certain characteristics, and we do not wish to deny them this flexibility. Exaggeration is commonly used in the art of caricature. As a result, a caricature is often more recognizable then a photo-realistic drawing of a person [14]. This principle applies equally to animation and artist can exaggerate or modify appearance freely using the method presented here.

The remainder of this chapter is organized as follows. Section 3.1 begins by introducing the problem of blendshape retargeting. Methods for selecting the set of keyshapes from the source motion data are discussed, followed by descriptions of decomposing the source motion data into a weighted combination of the keyshapes. Finally, the weights are used for retargeting the motion to a target model. Section 3.2 shows a number of examples that illustrate the method. Finally, section 3.3 concludes the chapter with discussion and future work.

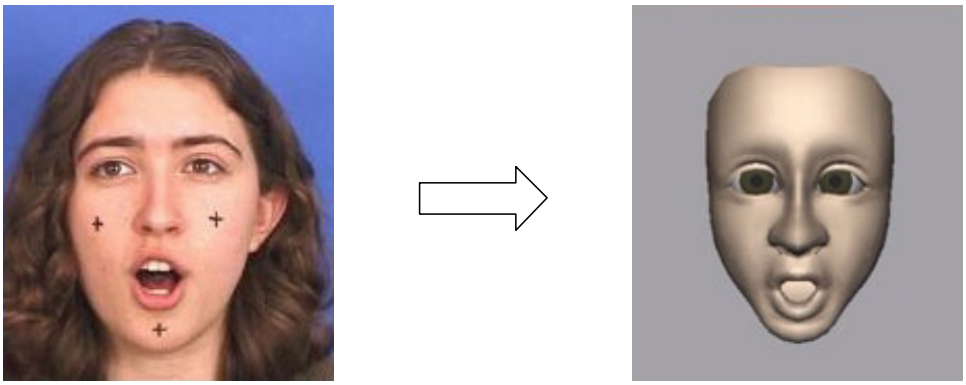
3.1 Blendshape retargeting

The problem of blendshape retargeting can be described as follows. Given a sequence of input motion data $\mathbf{S} = [\vec{S}(1), \vec{S}(2), ..., \vec{S}(t)]$, each frame is a function of a set of source keyshapes and a set of weights associated with the keyshapes for that frame. That is, $S(t) = F(\mathbf{B}_{source}, \Omega(t))$, where $\mathbf{B}_{source} = \{\vec{B}_{S1}, \vec{B}_{S2}, ..., \vec{B}_{Sk}\}$, and $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$. Given the input data $\mathbf{S}$, the function $F(\cdot)$ and the source keyshapes $\mathbf{B}_{source}$, blendshape retargeting involves solving the weights $\Omega(t)$ for each

frame in the source data. In the target domain, we need an equivalent, or analogous set of keyshapes for the target model, $\mathbf{B}_{target} = \{\vec{B}_{T1}, \vec{B}_{T2}, ..., \vec{B}_{Tk}\}$.. The definition of analogy is open for interpretation, and is up to the artists to decide. For retargeting, we simply take the weights Ω, and carry them to the target domain and solve the sequence $\mathbf{T} = [\vec{T}(1), \vec{T}(2), ..., \vec{T}(t)]$, where $\vec{T}(t) = H(\mathbf{B}_{target}, \Omega(t))$. The function $F(\cdot)$ and $H(\cdot)$ are not necessarily the same, and are often dependent on the parameterization of the keyshapes. For instance, if the keyshapes are represented as a set of control vertices that define a volume, then a desirable function may be a nonlinear morphing function that preserves the volume when two keyshapes are interpolated. We use a linear function in our examples, as solving for the weights is a simple least square problem.

$$\vec{S}(t) = \sum_{i=1}^{k} \omega_i(t) \cdot \vec{B}_{Si} \tag{3.1}$$

The same linear function is used for the target domain. We will show that for most of our examples, a linear function is sufficient given a carefully selected parameterization. Figure 3.2 illustrates the concept of blendshape retargeting using a linear function.



$\mathbf{S}(\mathrm{t}) = \sum_{i=1}^{k} \omega_i(t) \cdot \mathbf{B}_{Si}$ $\Omega(t)$ $\mathbf{T}(\mathrm{t}) = \sum_{i=1}^{k} \omega_i(t) \cdot \mathbf{B}_{Ti}$

Figure 3.2: Blendshape retargeting with a linear function

3.1.1 Blendshape decomposition

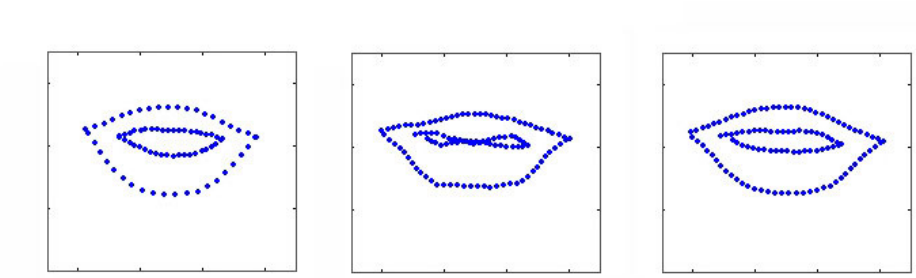
Facial decomposition takes the facial features from the captured sequence and decomposes them into a weighted combination of the keyshapes. The weights derived during this process will be used to specify the animated motion of the target model. According to equation 3.1, the least square fit seems to be a simple optimal solution. However, we do not really care about a perfect reconstruction of the input data, instead we are interested in the quality of the motion in the target animation. Our goal is to find the optimal weights describing the process of creating each frame in the target domain. However since we do not yet have the target motion to use as ground truth, there is not an objective error function.

Although we can not characterize the desired error function analytically, we can evaluate different methods empirically. The least-squares solution turns out to be a poor choice due to overfitting. Consider the following example. Given a set of keyshapes for the source data represented as the point positions of the lip contours as shown in figure 3.3(a), we resample the contours and use denser samples as keyshapes for the target model. Given a lip shape from a frame in the input sequence, we show that the least-square solution for equation 3.1 does not guarantee a valid retargeting when applied to the target keyshapes. The result is shown in figure 3.3(b). The resulting lip shape is unrealistically distorted. This is because by strictly minimizing the least square error it is possible to overfit the data. This overfitting often manifests itself as large positive and negative weights that counteract and compensate for one another. While these large weights minimize the error for the input shapes, they amplify noise when applied to the target model, even in this simple case where the target keyshapes are only a re-sampled version of the original.

A solution to this problem is to add a positive constraint to the least-square equation.

$$\vec{S}(t) = \sum_{i=1}^{k} \omega_i(t) \cdot \vec{B}_{Si}, \qquad \omega_i \geq 0 \tag{3.2}$$

This restriction forces the weights to be relatively small. It also makes most of the weights close to zero so that only a few keyshapes are used for any given input. Figure 3.3(c) shows the same example implemented with the positive constraint. The



(a) (b) (c)

Figure 3.3: Blendshape retargeting with positive weight constraint

constrained least-square problem can be solved using linear programming [58].

3.1.2 Key shapes selection

The goal of choosing the keyshapes is to cover the space of the facial expressions in the source motion data, and to minimize the number of shapes required in order to minimize the amount of work for the artists in modelings. To minimize the mean square error in equation 3.1, the optimal solution would be to use Principal Component Analysis (PCA), to determine an orthonormal basis set that represents the directions of greatest variance. Unfortunately, even though PCA is mathematically elegant, it does not have physical meaning for the purpose of modeling. Another approach is to use a set of meaningful predefined facial poses, such as jaw opening and squinting, and ask the actors or actresses to perform them. These basic poses can be captured first and used as the keyshapes, as in the approach taken by Pighin et al. [76]. However, unless the actor and actress has great control over each individual facial muscle, sometimes it is difficult to perform these basic poses in an isolated manner, leading to several rounds of trials to capture a sufficient set of facial poses. Furthermore, if we want to use an old video recording as source data, the original actors/actresses may not be available. Our approach is to use actual frames from the motion data as keyshapes, since they are available to us, and they can be shown to the retargeting artists as a meaningful reference while they create the corresponding shapes for the target model. Given a sequence $\mathbf{S} = [S(1), S(2), ..., S(t)]$, we need to

choose a subset that covers the variation in the sequence. We evaluated and compared several methods for automatically choosing this subset.

To provide insight into choosing keyframes, we asked an animator to look through an entire sequence of video, and manually select the frames that would be considered as 'keyframes' for animation. Figure 3.4 shows the sequence with each frame projected onto the first two principle directions, and then arranged in a scatter plot. The images of the lip area for the keyframes are superimposed on the graph to show their locations in relation to the other frames. Notice that with the exception of a few frames, most keyshapes chosen by the artists happen to lie at extreme points, near the edges of this 2D representation of the lip motion data. This observation prompted us to consider three methods for automatically choosing keyframes: using the convex hull of the 2D projection, selecting extreme positions along each principle axis, and clustering.

Convex hull

First we project the entire sequence of the source motion data onto a low dimensional PCA basis. The data points that lie on the convex hull of this space are chosen as keyshapes. It turns out that for dimensions greater than two, nearly every frame in the sequences lies on the convex hull. Therefore we only consider the first two coefficients. Since every other data point must lie within the convex hull, it is expected that all other shapes can be obtained by interpolating between the chosen key shapes. Figure 3.5 shows a similar plot as the one in figure 3.4, except that the images are replaced by the points on the convex hull.

This method selects a set that seems to share a large number of shapes with the manually selected set. However, there are some shortcomings. First, there is a fixed number of points on the convex hull, and some of them are very similar from each other, such as the two images on the far right in figure 3.5. If one were to use fewer keyshapes, it is not clear which ones should be prioritized, and which ones would be considered less important.

PCA extrema

Similar to the previous method, we first project the entire sequence of the source motion data onto a PCA basis. The data points that have the most extreme values when projected onto the principle axes are selected as keyshapes. Starting from the first principle axis where the eigenvalue is the largest, the two frames with the maximum and the minimum projection onto each axis are chosen as keyshapes. Therefore, to pick k key shapes, we will use the first $k/2$ axes. The logic behind this algorithm is that for each mode of variation in the sequence, we want to choose the poses that represent the extrema of this mode. All other shapes should be derivable as an interpolation between these two extrema. Sometimes the extremum of one dimension may coincide with the extremum of a different dimension, in which case only one keyshape is necessary to represent the two modes of variation. Sometimes two points may be very similar even though they are selected from different modes. We calculate the Euclidian distance between every pair of keyshapes, and if the distance is below a threshold, the point that represent the extremum for the less dominant dimension is eliminated.

Clustering

This method is an obvious choice as it is a popular algorithm among various machine learning applications, including compression, pattern recognition, data mining, and natural language processing. The intuition here is that by clustering the input data and choosing a key shape in each cluster, coverage of the entire range of motion is ensured. Again, we project the data onto a PCA basis to produce a low dimensional vector. This is commonly done in clustering for high dimensional data [29, 38]. The low dimensional data is clustered into k groups using the K-mean algorithm [43]. The center of mass of each cluster is chosen as a keyshape. Clustering is performed using both two and three dimensional PCA coefficients.

All of the above methods derive sets of keyshapes that provide reasonable solutions to our problem. Since there is not an objective measurement for the quality of the

retargeted animation, we can only evaluate these methods using the mean square reconstruction error of equation 3.2 for the source data. Figure 3.6 shows the mean square reconstruction error of the points around the lips as the number of keyshapes increases. The keyshapes are chosen by the three methods described above, while the weights are produced by least square with constraints as shown in equation 3.2. The solid blue line shows the error from using the maximum spread along the principle components. The dashed green and magenta lines show the errors from using the key shapes chosen by the clustering method. The green line is the result from using 2 principle components, and the magenta line is the result from using 3 principle components. Finally, the red dot shows the error from using the convex hull method. Since the number of data points on the convex hull is fixed, only one data point is present. Note that using the maximum spread along the principle components results in the smallest reconstruction error among the three methods. In addition, the error decreases monotonically as the number of key shapes increases, suggesting a graceful degradation when fewer key shapes are used. The errors produced by clustering also decrease with an increasing number of key shapes. However, the curves are somewhat noisy and not strictly monotonic. This is partially caused by the fact that the non-deterministic behavior of the clustering algorithm. Furthermore, we noticed that using more dimensions for clustering does not improve the result. The convex hull method seems to perform as well as choosing the maximum along principle components. However, as mentioned earlier, since the number of chosen key shapes is fixed, this technique is less flexible.

From the above analysis, we concluded that the second method, which picks the extreme data points along principle components axes, selects the keyshapes from the input sequence that best suit our requirement. This method also has the property that quality degrades gradually as the number of key shapes is reduced. Figure 3.7 shows some of the key shapes picked by this algorithm. Note that the set of key shapes consists of a variety of eye and mouth shapes, matching our intuitive notion of covering the space of possible poses.

3.2 Examples

We began with a video sequence of an actress speaking in various facial expressions. The facial features were tracked using the technique described in section 2.3.1. The input to the keyshape selection algorithm is the location of the feature points as shown in green dots in figure 2.2. To minimize the number of keyshapes, the face is divided into the top and bottom region. Examples of keyshapes chosen are shown in figure 3.7. These shapes cover the space of variations that exist in the recorded motion.

For the target model, we started with a NURB model created in the software Maya, which provides a blendshape tools for creating animation. The model is already parameterized with a predefined sets of basic shapes, such as jaw opening and squinting, as shown in figure 3.8. Sliders are used to control the amount of each component. The selected keyshapes shown in figure 3.7 are used as the references for creating keyshapes for this model. The user can adjust the slider values until the final blendshape reflects what the user sees. The whole process takes about one hour for a novice user, which includes posing for 8 key shapes for the eye area, and about 20 key shapes for the mouth. To combine the blendshape for the top and bottom regions, the positions for the control vertices in the intermediate region between the eyes and mouth are computed by interpolation with linear weights. The resulting animation is in figure 3.10, where selected frames are shown. This particular sequence has 386 frames, where the actress has a scared and suspicious facial expression. Please see the corresponding video for the entire animation. Figure 3.9 shows another example sequence where the actress looks angry.

One advantage of this retargeting technique, is that it decouples the motion capture technique from the retargeting method. It does not require a complex model that matches the appearance of the performer, nor does it require explicit correspondences between the source and the target. As a result, this method is very general and is not limited to any specific type of input data or output media. In the previous example, we used the output of the appearance model tracker, and retargeted the motion onto a 3D face model. Figure 3.11 shows that the input data can be retargeted onto a 2D

drawing of a cartoon face. Furthermore, we can alternately use marker, or feature-based tracking as input data. For instance, the left side of figure 3.12 shows a setup marker-based motion capture system where we used 7 cameras from a Vicon system. The right side shows the tracked markers on the actress' face. Figure 3.13 shows that marker data can be retargeted to the same 3D face as the first example.

This technique is not restricted to faces, and it does not require the same topology for the source and target. In figure 3.14, a cartoon bear is tracked with a technique very similar to the appearance-based model tracking used in section 2.3.1. The motion is used to animate a 2D drawing of a flower. The bear is a bipedal figure, whereas the flower does not have legs. The artist interprets the kicking and stretching of the leg with bending of the flower stem, which successfully capture the style of the bear dance.

In the final example, we show retargeting a 2D articulated figure to an articulated figure in 3D. On the top of figure 3.15 shows a 2D drawing of a man walking on the left, and the goal is to retarget his motion to a cartoon 3D character that has arms, legs, short tail, and long bangs, as shown on the right. This example has two additional challenges. The first is that 2D articulated motion is used to infer 3D motion, and the second is that unlike the other examples shown so far, linearly interpolating the control vertices will cause the arms and leg to shrink and collapse. Fortunately, by decoupling the motion capture and the retargeting, these two problems are both easy to resolve. Since the mapping is implicit, all the 3D interpretations of the 2D poses that can not be described by translation and rotation, are embedded implicitly in the 3D keyshapes. Furthermore, since the source and target parameterizations are decoupled, we use the joint angles defined by the skeleton underneath the 3D character for the interpolation, even though the source data is parameterized with shape morphing. On the bottom of figure 3.15 shows the retargeting result. The top row consists of the side by side comparison of the original drawing and the retargeted skeleton for the 3D model, and the bottom row shows the rendered version.

The formulation in our method does not enforce any temporal constraint. In some cases, jerky motion can result from various sources of error. The first source of error arises from the fact that although the solution to equation 3.2 ensures that only a few

keyshapes are used for each frame, some small weights can still remain in the solution. These small weights may have negligible contribution to the source data, but their effects can be 'amplified' in the target domain if the particular keyshape that the weight corresponds to happens to be significant. This can be resolved by filtering the weights temporally before retargeting. The resulting animation looks smoother, however some artists feel that the essence of the motion is altered by smoothing. In order to accommodate this artistic preference we allow the animator to freely adjust the level of smoothing.

The second source of error, however, is more fundamental to the implicit parameterization. The artist is relied upon to maintain consistency in the mapping. For instance, if eyebrow raising in one source keyshape is mapped to ear enlarging in the target keyshape, then eyebrow raising in another keyshape should also map to ear enlarging in that keyshape. If this consistency is violated, it may result in jerky motion. We have found that animators are capable of adequately preserving consistency, and automatic checking for errors of this kind is left as a topic for future research.

3.3 Summary and future work

This chapter described a method for creating facial animation using a combination of motion capture and blendshape interpolation. This method uses implicit mapping of the parameters, and relies on extracting a compact set of keyshapes from the source data, and thus does not require building a complex model for each individual to be captured. Three methods for extracting the keyshapes were evaluated. Facial tracking data is then decomposed into a weighted combination of key shapes. These key weights implicitly encode the mouth shape, eye movements, motion, and style present in the original sequence. By transferring these weights to a new set of key shapes modeled by the artist, this characteristic motion is preserved. The method described here is very intuitive and straightforward, and it does not restrict the types of input and output data. We show several examples of applications, with input ranging from marker positions of facial motion capture, to a dancing bear in cartoon, and output ranging from 2D drawing to 3D characters.

There are other possible extensions of the work in this chapter. First, the assumption of a linear interpolation function is limiting, and generalization to more complex functions would allow this framework to be applied to a wider range of facial analysis methods. Better methods for extracting keyshapes and determining key weights may yet be discovered. For instance, in our examples, we sometimes partition the face into multiple regions to minimize the scope of the retargeting. We do this because our least squares fit is global and eyebrow shape can conflict with mouth shape. A more local representation of the input data would be desirable in this context. Finally, learning the dynamics of facial expressions is also an interesting future direction. The technique described in this chapter allows artistic interpretation in the spatial domain. The next interesting step would be to allow artistic editing for the time domain.

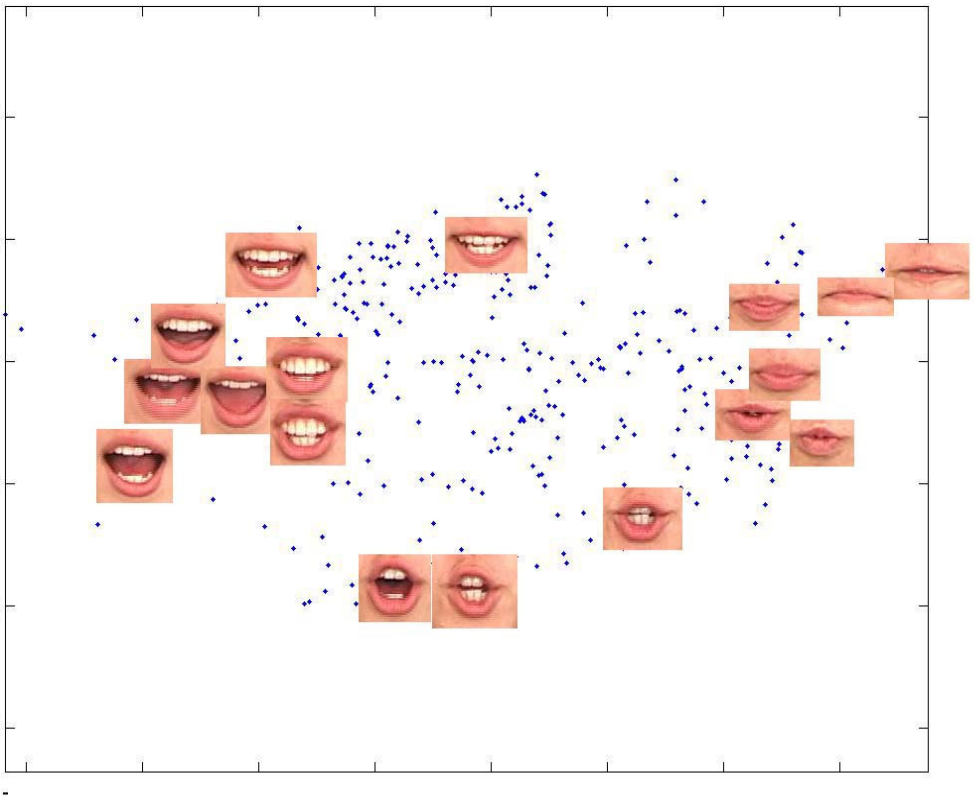


Figure 3.4: Manually selected keyshapes and the PCA coefficients. The keyshapes are plotted against the first two coefficients of the PCA. Note that most of the shapes are near the convex hull with a few exceptions.

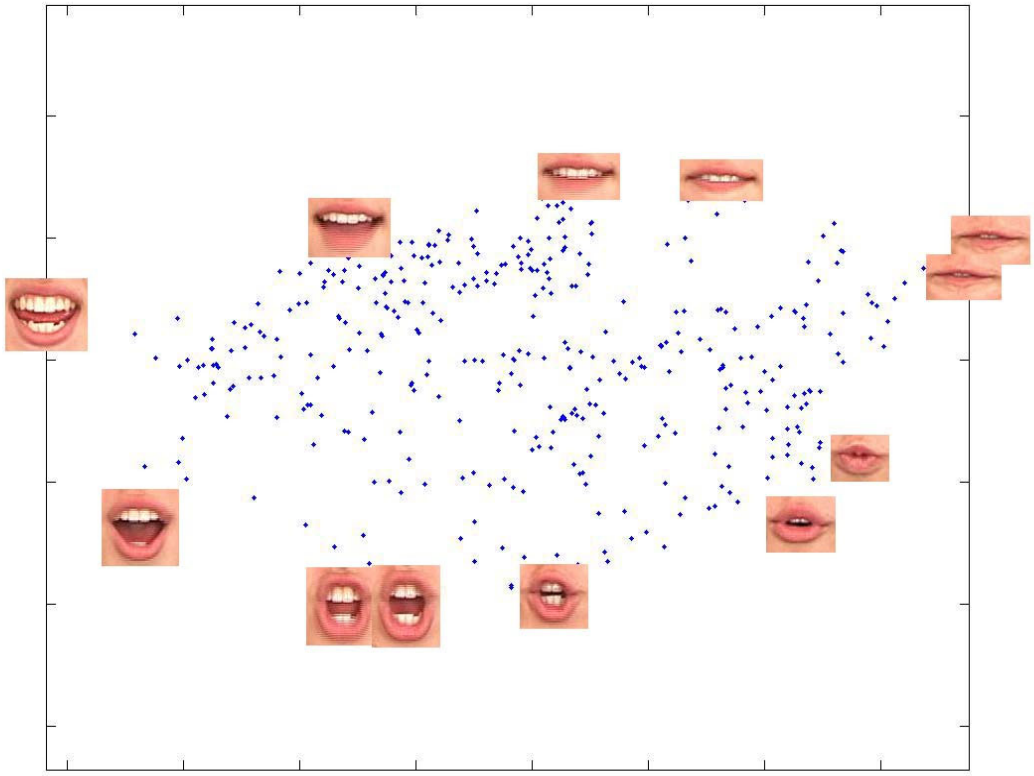


Figure 3.5: Keyshapes on the convex hull of the PCA space. The keyshapes are plotted against the first two coefficients of the PCA.

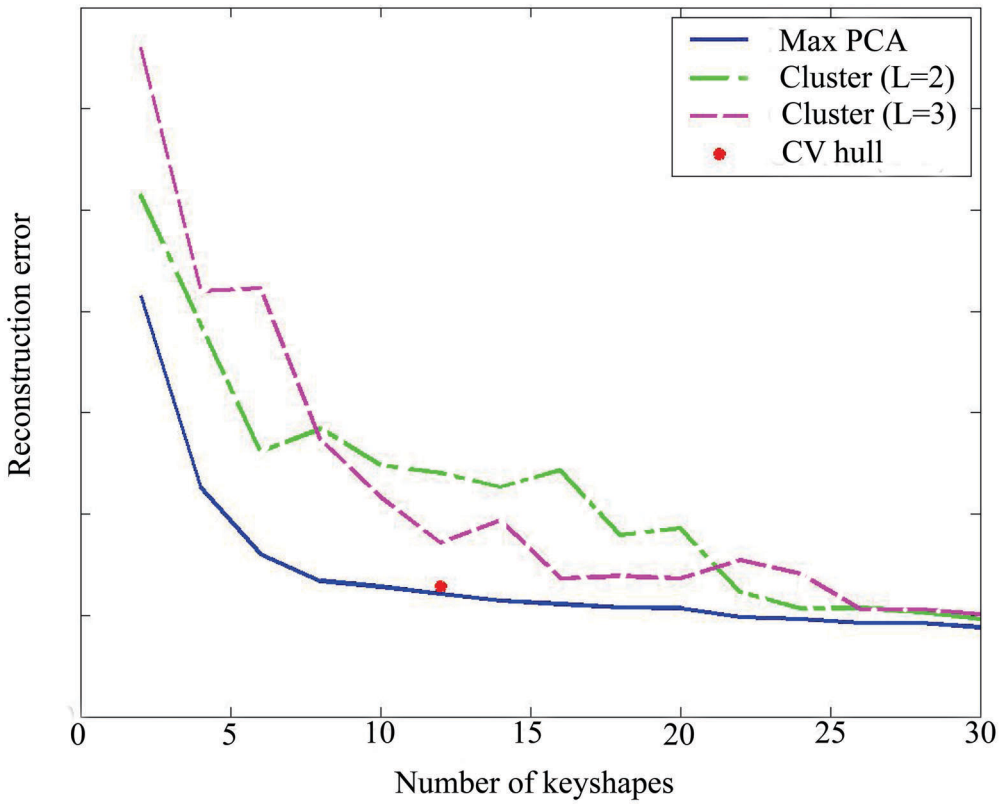


Figure 3.6: Square reconstruction error by using keyshapes selected by the three proposed methods. The solid blue line shows the error from using the maximum spread along the principle components. The dashed green and magenta lines show the errors from using the key shapes chosen by the clustering method. Two and three principle components were used in these cases (L=2,3). The red dot shows the error from using the convex hull method.

Figure 3.7: Selected keyshapes for the eye and mouth region

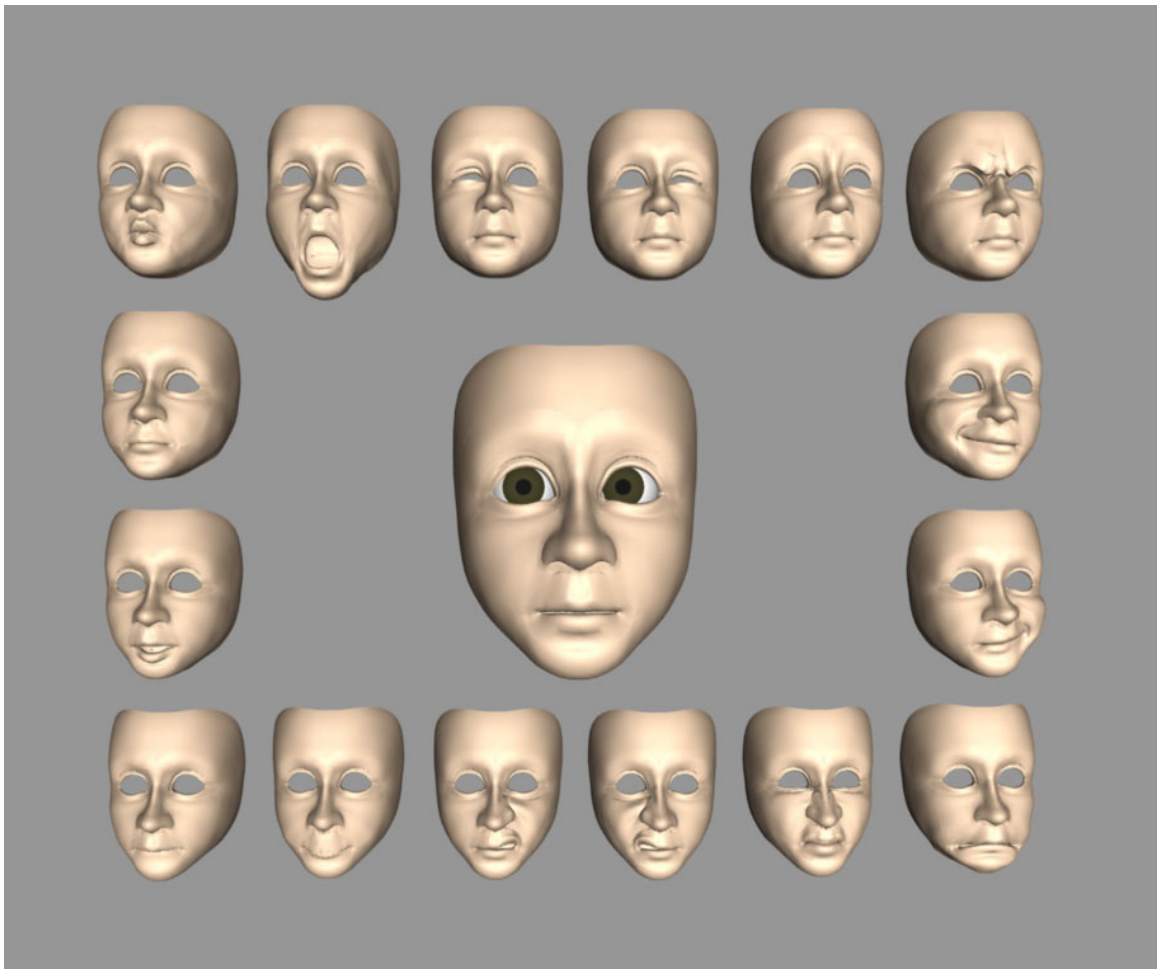


Figure 3.8: Target face model in Maya. The center shape is the blendshape, which is a weighted combination of the other shapes shown in the figure. The model is constructed by a number of NURBS patches.

Figure 3.9: Retarget to a 3D model: angry. See video figure 3.9 for the entire sequence.

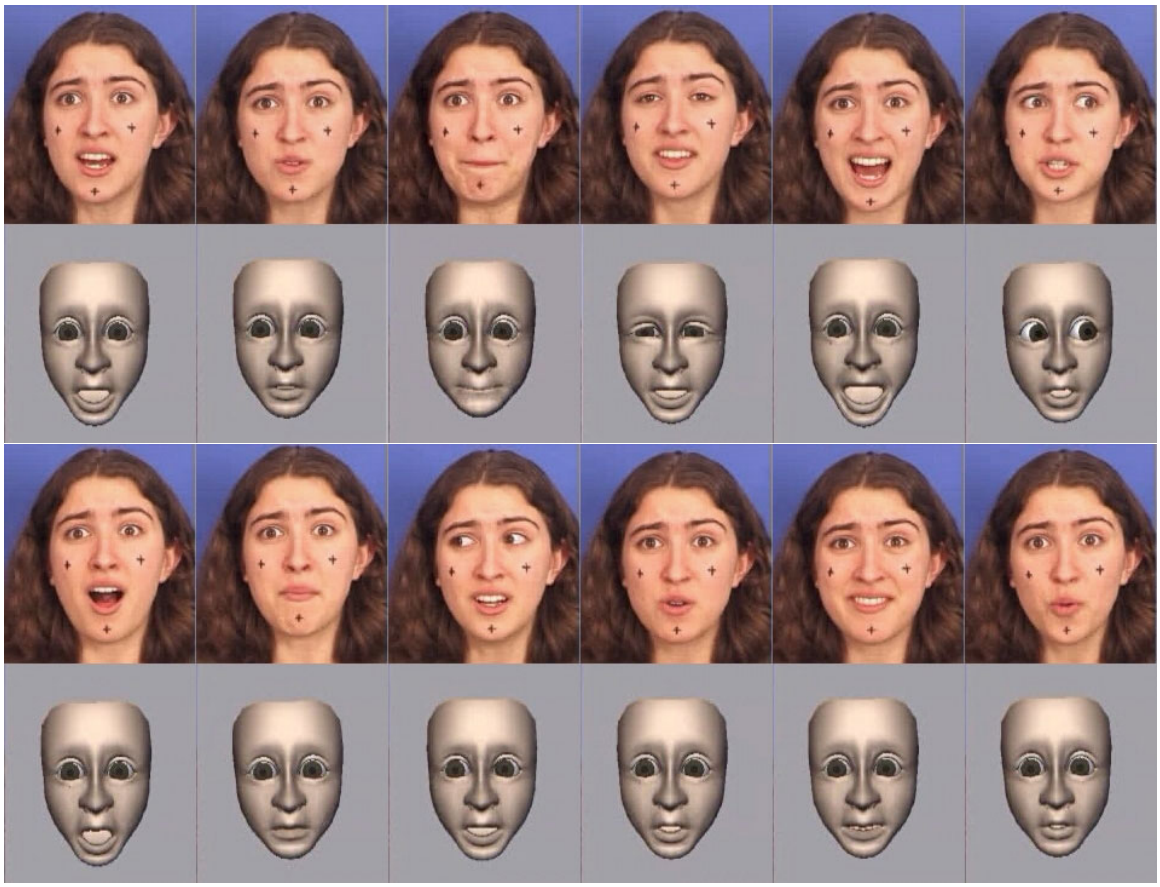


Figure 3.10: Retarget to a 3D model: fear. See video figure 3.10 for the entire sequence.

Figure 3.11: Retarget to a 2D drawing. See video figure 3.11 for the whole sequence.

Figure 3.12: Marker-based motion capture. On the left: 7 cameras set up with the Vicon system. On the right: the facial markers.

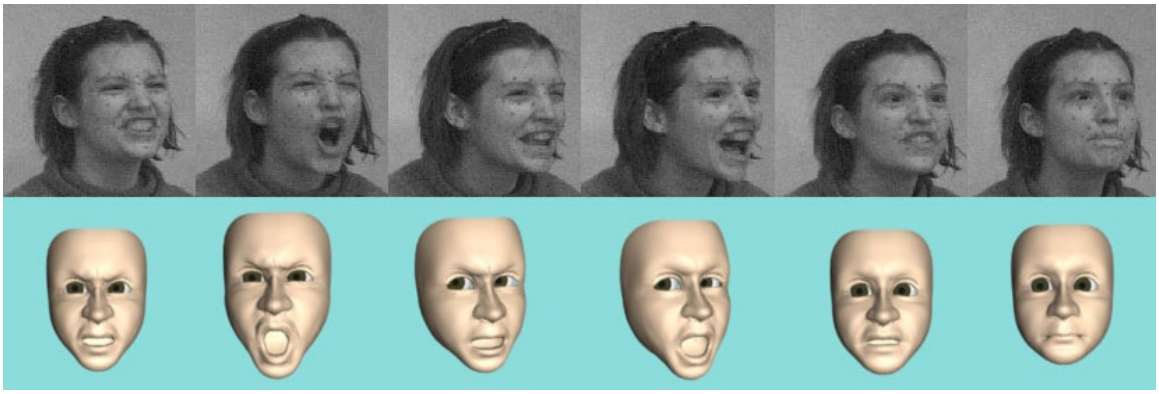


Figure 3.13: Retarget marker data to a 3D model. (the eye-gaze and the head motion are not generated by blendshape retargeting.) See video figure 3.13 for the entire sequence.

Figure 3.14: Retarget a cartoon figure to 2D drawing. See video figure 3.14 for the entire sequence.

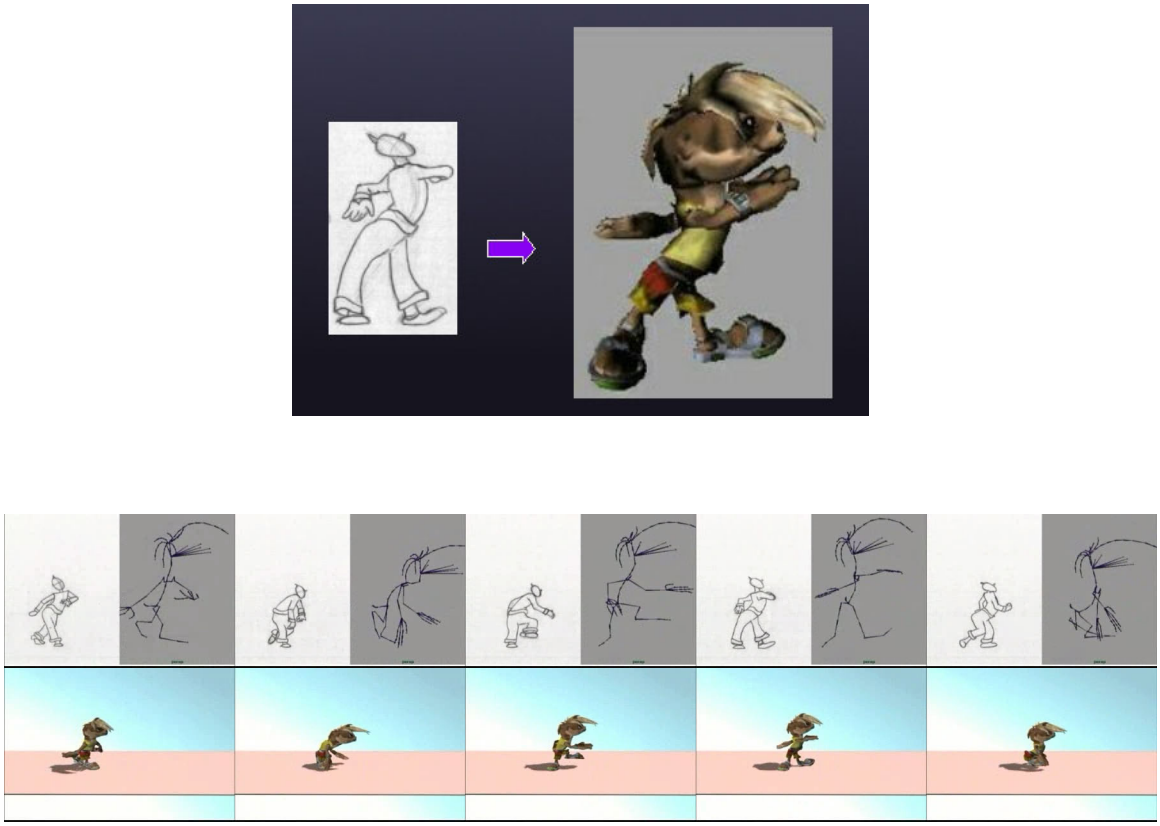


Figure 3.15: Retarget a cartoon figure to a 3D articulated model. The top row consists of the side by side comparison of the original drawing and the retargeted skeleton for the 3D model, and the bottom row shows the rendered version. See video figure 3.15 for the entire sequence.

Chapter 4

Facial Gesture: Head Motion Synthesis

When people engage in a conversation or story telling, they use nonverbal cues to emphasize their intention and emotion, and to clarify situations that are subject to multiple possible interpretations. These nonverbal cues are very important for animating a realistic character. They include eye gaze, eye blinking, head motion, and various body and hand gestures. This chapter takes head motion as an example and investigates methods for using motion capture data to create head motion animation. Head motion is arguably one of the most important aspects of character animation. In fact, in traditional animation, animators often start out creating the head motion first, and then fill in "less important" details, like lip-sync and other expressions. Many important aspects of facial speech are expressed or further emphasized by head motions. This chapter will illustrate how to synthesize head motion given new audio data using existing motion capture data.

Traditionally, head motion for animated faces are produced manually, randomly [75], or by a set of rules derived from communication studies [17, 87, 79, 26]. These techniques are either tedious for a long animation sequence, or do not capture the fine details of motion that constitute a realistic character. In recent years, performance driven facial animation has become more popular, since it shortens the animation production cycle, and more importantly, it captures the essence and personality of a

real person. Techniques to edit speech content have recently received a great deal of attention [13, 12, 22]. However, these techniques often neglect to generate realistic head motions. The generated animations have either no head motion, or the head motion has little correlation with the rest of the face. Although the lip motions and other facial features are animated very realistically in many systems, the lack of the "right" head motions diminishes the quality of the facial animation.

Head motion synthesis presents a different set of challenges to facial speech animation. The coupling between phonemes and visemes (the visual appearance of the mouth) is much stronger than the coupling found between the audio signal and head motion. Many statistical techniques for facial animation rely on strong correlations among the different parameters. However there is no deterministic relationship or obvious correlation between head motion and other communication cues. Given the same sentence and expression, many different head motions are possible. Nevertheless, a correlation does exist, and randomly generated head motion will not convey the expressiveness desired.

Fortunately, there is evidence that it is possible to use audio information as simple as the pitch contour to generate head motion. Recent study in the phonetics and linguistic community suggests that there is an anatomical coupling between head motion and pitch [49]. An empirical study by Yehia et al. [98] also concluded that the pitch contour in the audio signal is highly correlated to the head motion. In fact, using the head motion, they were able to predict pitch contour reliably. However, the prediction in the reverse direction was not successful. The author suggested that predicting head motion from pitch is a one to many mapping problem, where one input can produce many outputs, thus additional information is required.

Most existing methods in speech driven facial animation have recognized the phenomenon of many-to-many mapping and have emphasized techniques for resolving it. Unfortunately these techniques are not designed for extremely loose couplings such as the ones encountered here. We need a method that will describe this loosely coupled information between audio properties and the head motion.

This work makes the assumption that even though there are many valid choices for motion locally, due to the one-to-many problem, the head motion has to be consistent

globally. Since our goal is to synthesize new motion, not reconstruct the original motion precisely, we can select any of the many valid choices. The characteristics of human voice often manifest the emotional state of the person, i.e. it is easy to tell from the voice whether a person is sad or happy. The method presented in this chapter explicitly incorporates emotion in the speech to help generate expressive head motion.

The rest of this chapter is organized as follows. Section 4.1 introduces the head motion synthesis problem and explains its main differences from the facial speech animation problem that leads to a set of different approaches. Section 4.2 discusses the animation methodology and the stages involved in this process. Section 4.3 describes the actual experiment. Finally, the chapter concludes with discussion and future research direction.

4.1 Problem description

Given a sequence of audio data, the goal of synthesis is to find the best head motion trajectory accompanying the audio data.

A typical approach for this kind of problem is to make the assumption that the audio information and the head motion are governed by some process, which can be represented by a discrete number of states, and the relationships between the process, the audio, and the head motion are modeled by a set of probability distribution functions, which are governed by the Bayes rule. According to this formulation, the goal of synthesis requires solving two sub-problems. The first is to determine the state sequence $\mathbf{Q} = \{q_1, ..., q_T\}$, based on a sequence of observations, $\mathbf{O} = \{o_1, ..., o_T\}$ (i.e. input audio signal). The second subproblem is that given this state sequence, find the best trajectory for the head motion.

A Hidden Markov Model (HMM) is a logical choice for modeling problems with this kind of temporal characteristic. Here we will discuss the fact that even though this is a common approach for many facial speech animation systems, it is not suitable for synthesizing head motion.

A Hidden Markov Model is a probabilistic model of the joint probability of random variables $\{O_1, ..., O_T, Q_1, ..., Q_T\}$, where the O_t's are the observations throughout time, and Q_t's are the discrete hidden states corresponding to the process governing the outcome. Normally, this problem is intractable, but under the Markov assumptions it is quite doable. The assumptions are: (1) the conditional probability of the state variable, given the previous state, is independent from all the other previous variables. In other words, each state is only dependent on the immediate previous state, i.e.

$$P(Q_t|Q_{t-1}, O_{t-1}, ..., Q_1, O_1) = P(Q_t|Q_{t-1}) \tag{4.1}$$

(2) the observation at t-th time, given the t-th state, is independent of all other variables, i.e.

$$P(O_t|Q_1, O_1, ..., Q_T, O_T) = P(O_t|Q_t) \tag{4.2}$$

The solution to HMM has well known algorithms documented in speech recognition[81] and other pattern classification applications [7].

So far, we have mentioned how the state sequence relates to the observation sequence, but not about how the two possible observations, audio signal and head motion, relate to each other. One method is to assume that since they are governed by the same process, it is sufficient to use HMM trained from one or the other for both. This assumption works pretty well for facial speech animation. For example, Bregler et al. [13] trained a HMM on the audio signal for recognizing speech at the triphone level, and used the triphones to sample the facial animation using a non-parametric method. Brand [12] trained a HMM model based on the facial configuration and used the same transitional probability and state distributions for obtaining the emission probabilities for the audio signal. Given new input audio, the state sequence is solved, and another maximum likelihood procedure is used to sample the facial configuration based on the state sequence. Both of these approaches make an assumption of strong correlation between the audio signal and the motion data. However, the correlation between the audio signal and head motion is a lot weaker, and will likely have a many-to-many mapping. The assumption that they are governed by the same structure is very unlikely to be true. In addition, the methods

above are more appropriate when a large database of motion exists. Our goal is to use a small amount of data, but still be able to generate head motions that are interesting and plausible for the given input audio.

Our method is to look directly at the correlation between the head motion and the audio data. Due to the fact that the relationship between them is many-to-many, and there is not a good model for the density distribution, we chose to use a non-parametric technique which samples the distribution based on the data itself. This technique is commonly known as *synthesis by example* in computer graphics, and is used for texture synthesis[31, 96, 4, 4]. In addition, to take into account that there is a large amount of temporal variation in both the audio data and the head motion data, yet the motion has to be consistent globally, we model our problem at two scales. At the smaller scale, instead of considering the data at every time sample, as proposed by Yehia et al. [98], we divide the data into segments, and consider a segment as an inseparable unit. At the larger scale, we focus on the relation between segments, and consider the cost through an entire path of segments to achieve a globally optimal solution.

Recent work in articulated motion synthesis shares similar philosophies with this segment-based approach [80, 2, 57, 61]. Our method is most similar to Pullen and Bregler [80] in spirit in that our methods are designed to work with a small amount of data. However the domain is quite different, in that their input is a keyframed motion sequence, whereas we deal with both audio and motion data.

4.2 Synthesis by example

The synthesis process can be summarized as follows. We begin by building a database of examples which relate audio pitch to motion. Synchronized motion and audio streams are captured and then segmented and stored in the database. A new audio stream can be matched against segments in the database. A smooth path is found through the matching segments and stitched together to create synthetic head motion. Figure 4.1 shows the overview of the approach. The task consists of four major steps: segmentation, pitch matching, path searching, and motion blending.

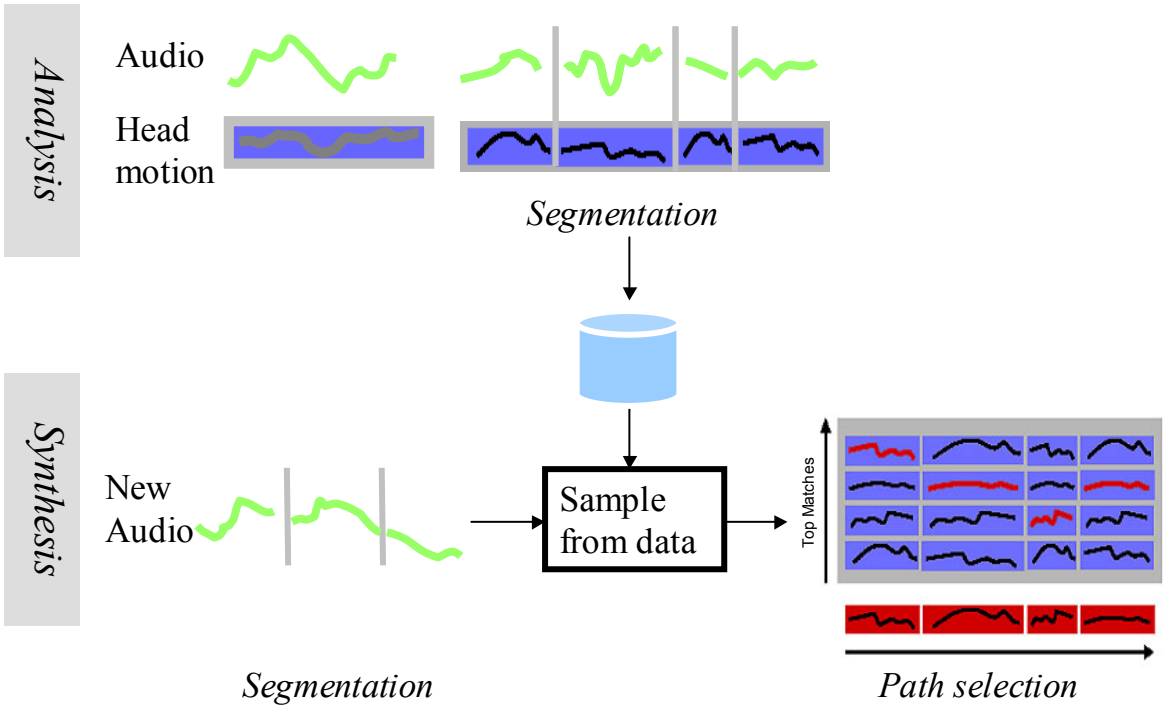


Figure 4.1: Overview of head motion synthesis

4.2.1 Segmentation

We segment the input audio track using pitch contour information, instead of using head motion for the segmentation. The same segment boundary is then used for slicing the head motion tracks. This choice is driven by the fact that audio is the only information we have available when synthesizing. In addition, we observe empirically that most of the large head motions happen right after pitch onset, suggesting that this is indeed a good segment boundary.

We used a program called Praat [10] to process the audio data and extract pitch. Unfortunately this processing is relatively noisy. The top of figure 4.2 shows a raw pitch signal. We filter the pitch values to consider only those that lie within the normal human range. In addition we interpolate neighboring values to fill in very short periods of missing data. After cleaning the pitch curve, we use voiceless regions (regions with no pitch), to divide the curve into segments. The bottom part of figure 4.2 shows the same pitch contours now partitioned and cleaned up after the

processing. Since head motion continues regardless of whether the audio contains a voiced and voiceless region, a segment is defined as starting from the onset of one pitch segment and ending on the onset of the next pitch segment.

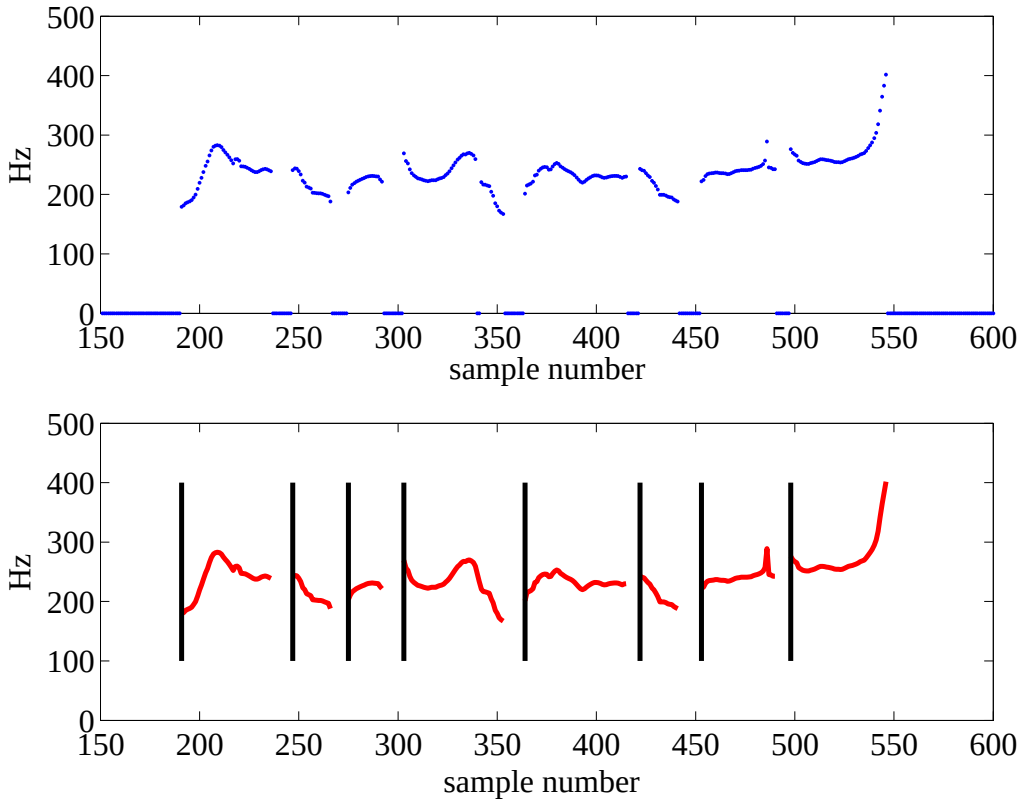


Figure 4.2: Pitch segmentation. Top: noisy pitch data. Bottom: cleaned up pitch data and segmented for use in the matching process

4.2.2 Pitch matching

For a given sequence of new audio pitch segments, we need to find a matching sequence of segments from the database. We define the matching distance in a two-stage process:

In the first stage, we compare pitch for an entire phrase, not just a single segment. This is important because the emotional, idiosyncratic content of speech is often

conveyed at a sentence level through pitch phrasing. By matching first at the sentence or phrase level we use only those sentences in our database with expressive style similar to our new audio sequence. Sentences are compared by matching feature vectors derived from the audio data. These features include statistics related to the speech rhythm; the speaking rate, and the average length of the voiced and voiceless regions, as well as simple statistics on the pitch signal, including the minimum, maximum, mean, and standard deviation of pitch values [27]. These statistics are normalized and used as a feature vector. Euclidian distance is used for calculating the top M sentences that best match the test input. These M sentences represent a subset of the database which has the desired expressive style.

In the second stage, we compare individual pitch segment in the test phrase against the pitch segments in the database subset. Each pitch segment is resampled to match the average length of all pitch segments. Then root-mean-square difference is used to compute a distance metric to every other pitch segment.

$$P_d = RMS(P_{test} - P_{template}) \tag{4.3}$$

P_{test} and $P_{template}$ are length normalized pitch contour. To avoid over stretching or shortening of the segments, a second criteria is used to penalize the cases where the difference in lengths of the original segments are too large.

$$L_d = \frac{|length(P_{template}) - length(P_{test})|}{length(P_{test})} \tag{4.4}$$

A combined distance metric is defined as a weighted sum of these two metrics.

$$D_{total} = c \cdot P_d + (1 - c) \cdot L_d \tag{4.5}$$

Finally, the list of matching segments for each input segment is pruned to retain only the top K choices. These segments are the possible matches which will be stitched together during path searching.

4.2.3 Path searching

Given N pitch segments in the input audio, and the top K matches for each test segment, each matching segment forms a node in a Trellis graph, as shown in figure 4.3. We need to find a path through the graph that produces a good head motion

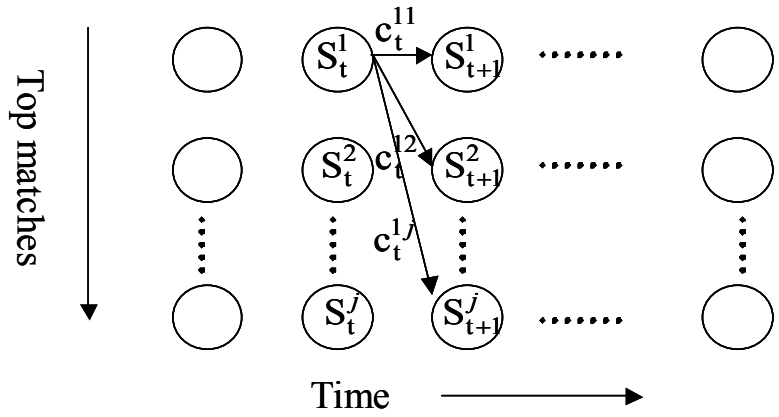


Figure 4.3: Trellis graph formed by the matching segments

trajectory. Here we use the head motion data accompanying each matching segment and compute the cost of transitions from one segment to the next. A good motion path is determined if the segments meet the following criteria:

1. Similar pitch segments, as computed in equation 4.5.
2. Similar length in the voiceless region following the voiced region.
3. Compatible pair-wise matching boundaries for the head motions between successive segments, i.e. position, velocity and acceleration.
4. Consecutive segments in the original database are highly encouraged, since it results in the most natural motion.
5. The same segment is discouraged from appearing twice, since it produces repetitive motion.

The first two items define the cost of each segment, while the last three terms define the transitional cost from one segment to the next. Each of them is weighted

and summed to give a final cost for each segment and transition path. The relative weighting of these factors was determined experimentally. We then apply the Viterbi algorithm to find the best path through the segments [94].

4.2.4 Motion blending

The motion data associated with each segment is joined together to produce the final motion. To achieve this, each motion segment is re-sampled to match the length of the test segment. Although this alters the frequency content of the motion data, the length of the segment is part of the matching criteria, thus the selected segments should have length similar to the test segment. Next, the starting position of each motion segment is moved to match the ending position of the previous segment. This could potentially lead to drifting in long sentences. When this occurs, we break long sequences into smaller parts, and perform path search for each sub-sequence. Each sub-sequence is linearly warped to join the next sub-sequence to correct for the drifting. Finally, the end positions where the segments are connected are smoothed to minimize high frequency artifact caused by mismatch in the velocity.

4.3 Experiments and result

After Institution Review Board approval and informed consent, we recorded a collection of 67 phrases performed by an actress. Each phrase consists of 2 or 3 sentences. The actress was free to move from waist up, and was informed that highly expressive motions are desired, therefore the footage consists of a wide variety of motion.

A marker-based motion-capture system from Vicon [86] was used for motion acquisition. We employed 7 cameras with 120 Hz visual sampling frequency. 42 markers are placed at important facial position to obtain the facial expression. In addition, 8 markers were placed on a plastic hair-band, and 2 markers are on each of the ear lobes. In this study, we only use the position of the 10 markers on the hair-band and the ear lobes to get accurate head motions, due to the rigidity of the points relative to each other. The Vicon software computes the 3D positions of the marker points.

We computed 3D rotations and translations for the hair band and ear marker positions using a technique based on singular value decomposition (SVD) [3]. In order to increase the amount of data, we add to the head motion data by mirror imaging the head motion.

The test audio is analyzed with techniques described in section 4.2.1, and used to synthesize new head motion. A ball-shaped head is to used to visualize the result. Figure 4.4 shows sub-sampled frames from the test data.

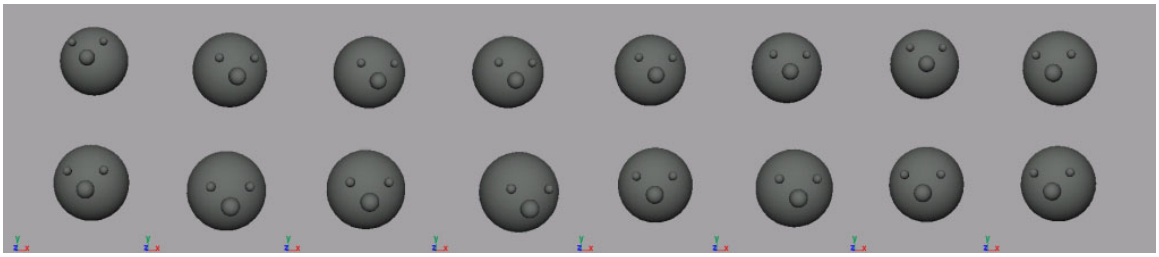


Figure 4.4: A ball-head animated with synthesized head motion. See video figure 4.4 for the entire sequence

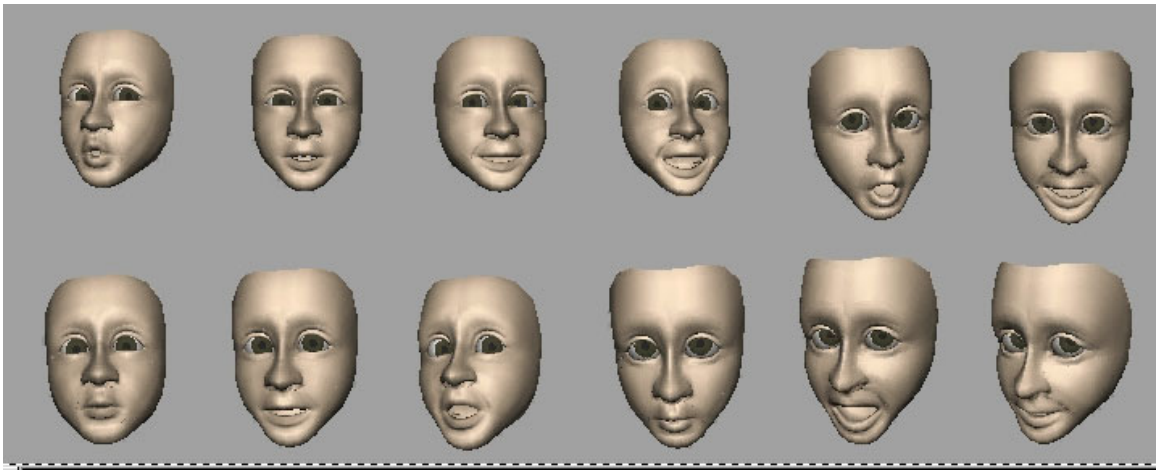


Figure 4.5: Facial animation with synthesized head motion. See video figure 4.5 for the entire sequence.

The result is reasonable, however, we found that it is difficult to judge the quality of the synthesized result, if we animate only head-motions, but do not render and animate facial expressions. Most people do not perceptually isolate head motion from

the rest of the face. Therefore, we also apply the head motion to an animated 3D head model containing facial expressions. We used the retargeting technique described in chapter 3 to generate the facial animation. The test audio was not captured from the same actress that we used for training the head motion. However, since we used pitch contour as the audio input, and both actresses have similar pitch range, the synthesis works even though the voice is from a different person. We found that by integrating the lip motion, the facial expression, and the head motion together, the face really comes alive. In comparison with head motion synthesized with random noise, the data-driven head motion seems to convey much more personality. Figure 4.5 shows some sub-sampled frames of the 3D facial animation with synthesized head motion.

4.4 Summary and future work

This chapter demonstrated a new technique that is able to generate head-motion for facial animation using motion capture data. Given input audio pitch, the algorithm searches for the best head motion path from the motion database, and incorporates emotional state explicitly in this process. In deriving these head-motions from example motion capture data, we are able to convey specific style and idiosyncrasies. This increases the realism of the animation. This is one of the first systems that generates head-motion in a data-driven way. We found, that it is in fact possible to create new head-motions from only the audio-pitch information.

Several aspects of head-motion depend on the higher level semantics. Many non-verbal cues are used by a speaker to highlight the intended interpretation of the utterance. Many of those cues co-occur in pitch and head-motions, and are thus reconstructed by our technique. However, several cues are exhibited in head-motion only. For example, if the speaker says: “and this on the right and that on the left”, she most certainly will also move her head to the right and to the left. We can not infer this kind of head-motion from the pitch alone.

There are many possible extensions for the work in this chapter. Many aspects of head-motions are not derivable from the audio information. Those cues should be an

additional input of our technique. Ultimately this technique could be used to augment a rule-based technique that follows semantic cues, adding subtle idiosyncratic features.

Chapter 5

Animation system

The final goal of this thesis is to combine the techniques introduced from chapter 2 to chapter 4 together to demonstrate the concept shown in the editing pipeline for motion capture based facial animation as proposed in figure 1.2. This chapter will show examples where the input is a video sequence of a talking face. The speech content of this sequence could be edited with existing techniques [13, 22, 37]. Since this thesis emphasizes the importance of expression, we focus on the last three modules, simply passing the input videos speech content directly to the expression synthesis module. The input video is analyzed using a bilinear model, to factor emotional style and speech content into separate components. The video sequence is then retargeted onto 3D characters using the blendshape retargeting method. By augmenting shape information with the extracted emotion vector, the retargeting method preserves emotional style while allowing the artists to freely design output models that bring out the characteristics of the key facial expressions. Finally, since head motion is an imperative part of expressive facial animation, we use data driven synthesis technique to create head motion that matches the character's emotional state.

5.1 Training and modeling

The training data is drawn from three short video sequences including three basic expressions: neutral, angry and happy. Each sequence is around 12 seconds long, but

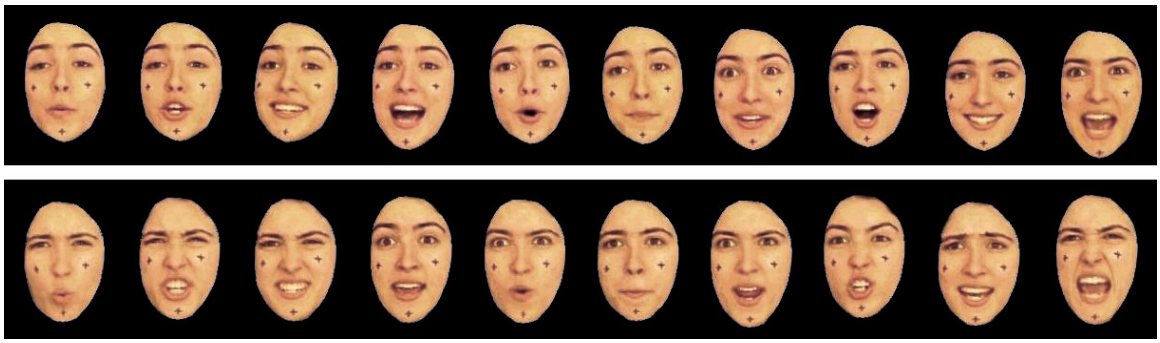
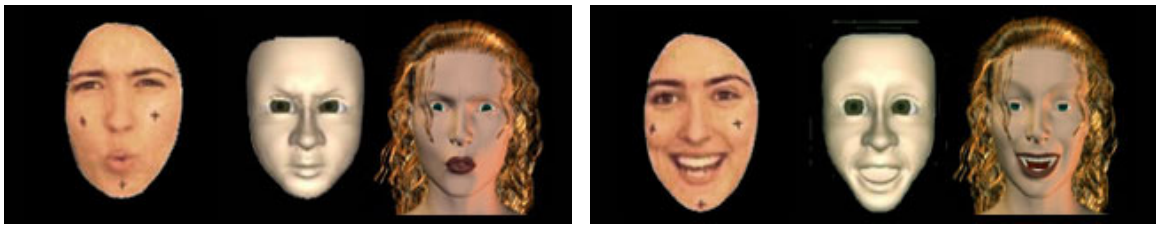


Figure 5.1: Examples of selected keyshapes for happy and angry expressions

chosen so that the words cover most viseme groups. The facial features are tracked, and the resulting data is compressed and represented as facial vectors as shown in equation 2.3. A symmetric bilinear model is then fitted to this data as described in section 2.2.2.

Subsets of keyshapes for different facial expressions are selected from the training set using the method described in chapter 3. In the examples shown in this chapter, there are approximately 20 keyshapes in each expression set. The entire face is treated as a whole, as opposed to being separated into top and bottom regions as described in chapter 3. Figure 5.1 shows some of these keyshapes. These keyshapes will be used for decomposing and retargeting facial expressions.



(a) (b)

Figure 5.2: Source keyshapes and the matching target models, (a) angry (b) happy. The model in the middle is posed in Maya, and the one the right is posed in Poser

For the output character, we used commercial models that have built-in morph targets, such as mouth opening, eye brow raising, and sliders for manipulating the blendshape with the morph targets. Target keyshapes that match the chosen keyshapes

are constructed to correspond to each source keyshape. Depending on the model, the user's skill, and inclination for perfectionism, this modeling process can take anywhere from minutes to hours. Figure 5.2(a) and (b) shows some examples of these keyshapes for the target models. On the left in each of these images is a keyshape selected from the video source. In the middle is a face model built in Maya [1], and on the right is a female head model built in Poser [23].

5.2 Animating

With the bilinear model and the set of output keyshapes, we are ready to generate new facial animations. To show the process of making a new expressive animation, we videotaped an input test sequence about 13 seconds long which consists of the actress saying a completely new set of sentences with neutral facial expression. The test sequence is analyzed using the bilinear model with techniques shown in section 2.3.3, and sequences with different facial expressions are synthesized. We then decompose the synthesized sequences into a weighted combination of keyshapes for each expression set. The resulting weights are used for retargeting.

Since we do not have a method for synthesizing expressive audio, we recorded the actress speaking the same text as the input test sequence but with different emotions. The audio is used as input to the head motion synthesis. Finally, the timing of the final animation is warped to match the new speech. Figure 5.3 shows a sequence of facial animation with the angry facial expression using the Maya model. Figure 5.4 shows the same sequence retargeted onto the Poser model. The images in the top shows an angry sequence, and the images in the bottom shows a happy sequence.

To create an animation with arbitrary or changing facial expression, we blend the retargeted facial animation for each basic expression together.

$$\vec{H}(t) = \sum_{j=1}^{N} \alpha_j(t) \cdot \sum_{i=1}^{K} w_i^{(j)}(t) \vec{G}_i^{(j)} \tag{5.1}$$

where $N = 3$ is the number of basic expressions, the α_j's are the barycentric coordinates of the desired expression in terms of the basic expression vectors found during

training, and K is the number of keyshapes used for that particular expression. With the equation, we can build sliders in the animation tool for changing the α values from frame to frame, such that the animators can use this tool to focus on animating the facial expressions without worrying about the lip sync. Figure 5.5 shows a snapshot of this particular tool setting inside Maya [1]. Notice there are two sliders on the right for adjusting anger and happiness. Figure 5.6 shows two instances of this example, where the facial expression changes from happy to angry from left to right, and the visemes are different in the top and bottom rows. Notice that the way the facial expression changes around the eyes, the cheek, and the mouth regions varies slightly for the two different visemes. This will be hard to achieve with a traditional method where the lip sync and the facial expressions are specified separately and simply superimposed.

The input test video used in the example above only consists of the neutral facial expression. However, other time-varying facial expressions can be used, as the bilinear model can extract the facial expression. This knowledge can be used to enhance performance driven facial animation. In addition to the shape information that traditional methods rely on, the extracted facial expression serves to disambiguate details that shape alone may not be sufficient to handle, and helps to improve the quality of the retargeted animation.

Although specific methods and data types were used for the modules described in this example, the methodology is much more general, and we wish to reiterate our emphasis on building a complete process for creating expressive animation. Changes could be made to almost any module while preserving the pipeline, and we hope to inspire future work by suggesting a couple of possibilities here. We currently rely on 2D video as a source for deriving input shape and emotion vectors. The additional data present in video-rate 3D range images, would presumably result in better models. It would also be interesting to explore physical simulation for creating the morph targets. In this case, instead of blending the locations of vertices, the output space would be a linear combination of physical parameters. Linking expression to physical parameters may lead to a level of performance beyond what is now possible.

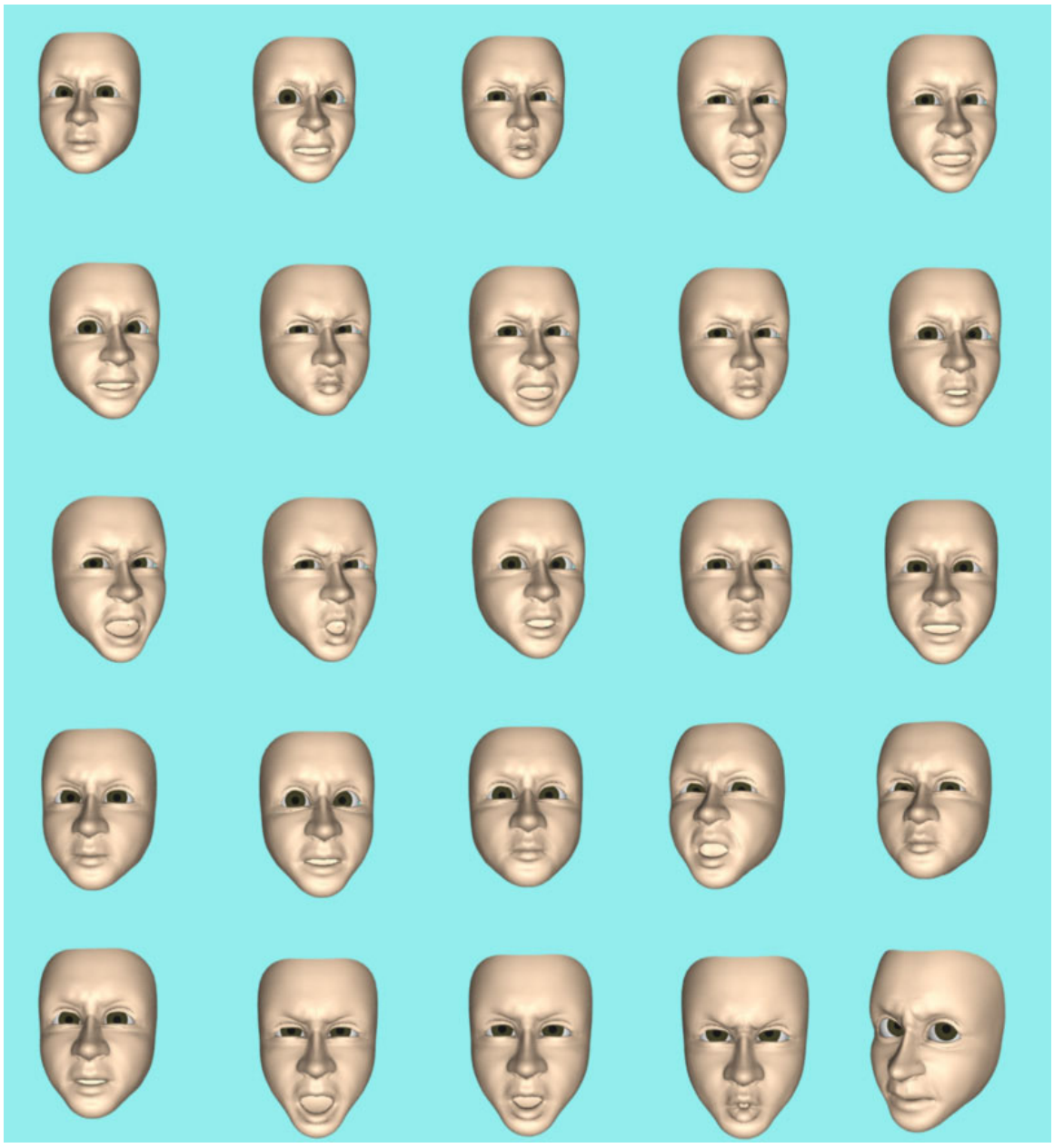


Figure 5.3: Sample frames of the test sequence with angry expression retargeted to the Maya model. See video figure 5.3 for the entire sequence.

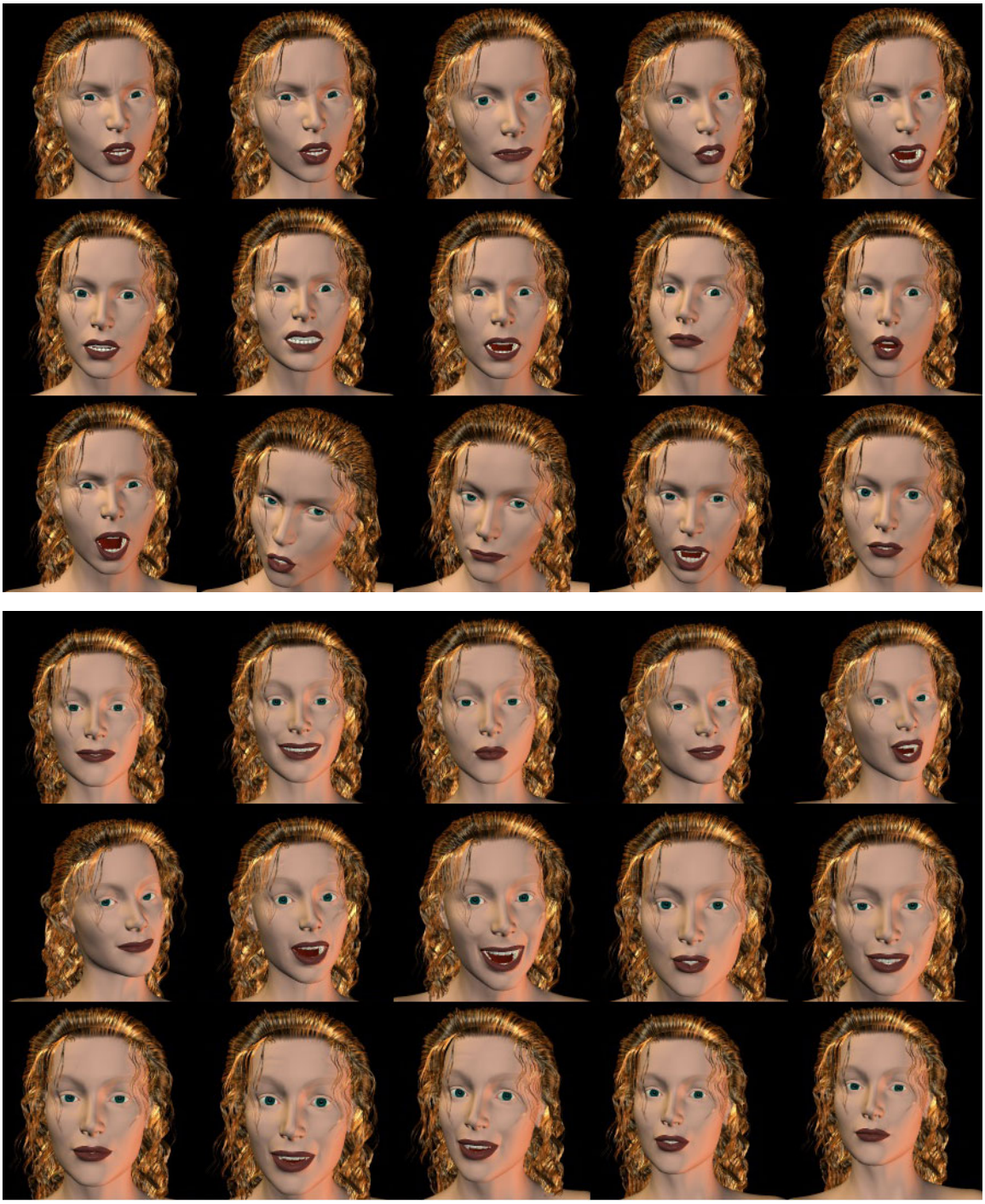


Figure 5.4: Sample frames of another retargeted sequence. Top: sample frames of the test sequence retargeted with angry expression. Bottom: same sequence retargeted with happy expression. See video figure 5.4 for the entire animation

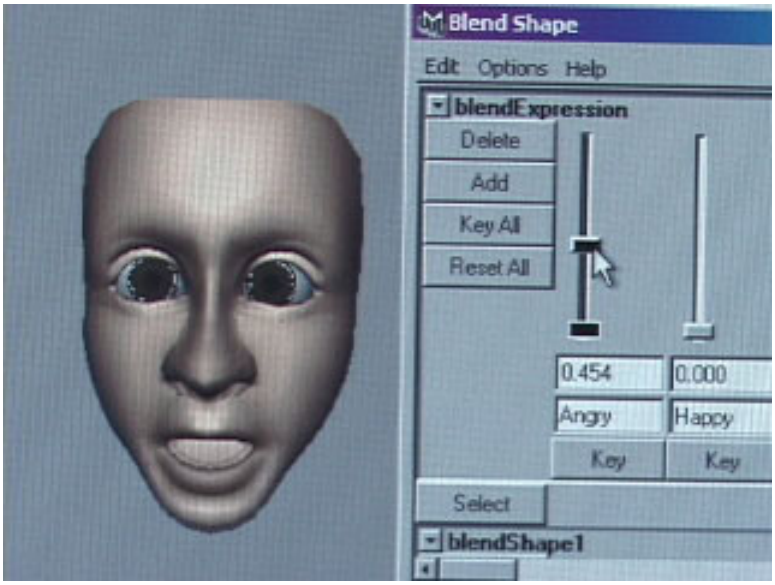


Figure 5.5: Snapshot of the expression slider. The animation for different expressions are blended so that the user can control the facial expressions without changing the lip sync

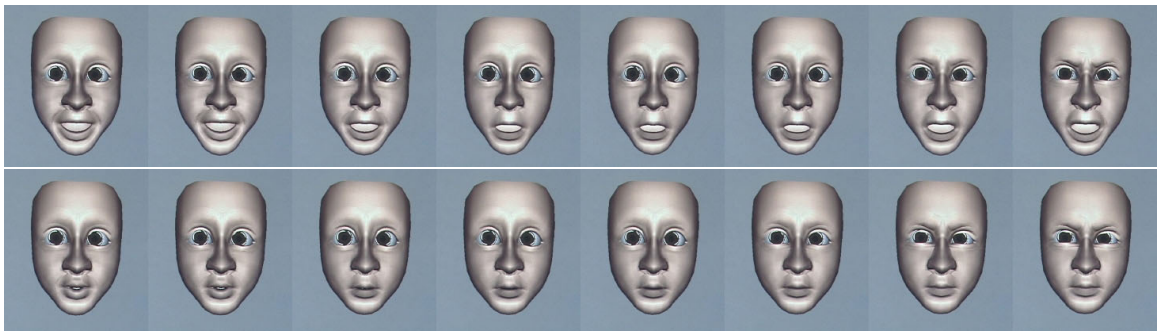


Figure 5.6: The expression changes with the slider value

Chapter 6

Conclusions

6.1 Summary

This thesis proposed a method for creating expressive speech animation using motion capture data. First, it extends the definition of motion capture to include all motion recorded from live performance, as opposed to just data from point markers. A motion editing pipeline is proposed, where the motion data goes through three stages of editing: speech content, facial expression, and the appearance of the character. Finally, facial gestures are added as the last stage to enhance the facial animation. This thesis assumes that the techniques for the content editing are available, and focuses on the remaining three parts of the editing pipeline. Chapter 2 introduces a method of editing the facial expressions by first factoring the motion data into content and style using a bilinear model. While keeping the speech content unchanged, the model allows us to change the input facial expressions. Chapter 3 introduces a method for retargeting the facial expressions to a different model with an implicit parameterization. Implicit parameterization allows the input and output characters to have different parameterizations and topologies. This retargeting method requires a set of keyshapes to be chosen from the input data. Three methods for selecting the keyshapes are presented and evaluated in this chapter. Chapter 4 introduces a method for synthesizing head motion for the animation by using motion capture data and audio pitch information. The algorithm also uses audio features that are related

to emotional state to help to direct the expressions in the animation. Finally, the three techniques described in chapter 2 to chapter 4 are combined into an animation system in chapter 5.

Finally, the goal for the framework described in this thesis is meant to encourage people to think about performance driven facial animation in a more flexible and modularized way, instead of being constrained by specific technologies.

6.2 Future work

The technology for automatic character animation is still at its infancy. Human motion is so complex. If we want to model the entire chain of commands from the brain to all observable behavior and actions, we will probably never run out of topics. There are many possible directions for future work that are all important to creating an expressive animated character. Unfortunately this thesis has a limited scope. I can only list a few possible areas that are most relevant to the current work:

Data capturing The success of performance driven facial animation is based on how much information can be captured and how much of it can be preserved in the animation process. Much of the subtleties of facial expressions reside in the wrinkles and shadows cast by changing geometry at the micro scale. Techniques for capturing 3D motion and shape deformation at this fine scale still do not exist.

Modeling The more details that we are able to capture, the more important a model is for better understanding the large amount of data available to us. A bilinear model model only takes care of two factors that interact to produce the appearance of the face. Many other factors are also related to each other in manners across different levels of details. A Taxonomy of how these factors interact with each other would be very useful in applying suitable models to incorporate them together.

Editing The current methods for editing animation are still very primitive and require a lot of manual specification. With improved capture technique and behaviorial

modeling, better techniques are needed to automate much of this process without sacrificing the quality of the animation.

Timing The current work only maps the relationship between different facial expressions in the spacial domain. However, people express their emotion through the tempo of speech and timing of their action. Currently, we rely on the speech to derive the timing of various facial expression. However, it would be interesting study the relationship between the two and how to effectively model the temporal characteristics of facial expressions.

Head motion and expression Currently we rely on very primitive features in the voice to help determine the emotion carried in the voice and how it can affect the head motion. We would like to be able to control that information and manipulate the head motion according to the expression more effectively.

Eye gaze and expression Eye gaze is known to be very important in animated characters. There have been studies of the fast movement of the eye (the saccade), and its application to animating more natural looking characters [59]. However, little work has been done on the emotional aspects of eye movement.

Multi-disciplinary integration It is important to realize that creating believable animated characters rely on knowledge from multiple disciplines, including computer graphics, computer vision, natural language processing, speech processing, and many sub-fields of psychology. Ultimately, more effort in integrating these disciplines is imperative in making expressive characters.

Bibliography

[1] Alias. Maya. http://www.alias.com/.

[2] O. Arikan and D. Forsyth. Interactive motion generation from examples. In *Proceedings of ACM SIGGRAPH*, pages 483–490, 2002.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Leastsquares fitting of two 3-d point sets. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(5):698–700, Septempber 1987.

[4] M. Ashikhmin. Synthesizing natural textures. In *Proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[5] T. Beier and S. Neely. Feature-based image metamorphosis. *Proceedings of ACM SIGGRAPH*, pages 35–42, July 1992.

[6] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

[8] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proceedings of the European Conference in Computer Vision*, pages 329–342, 1996.

[9] M. J. Black and Y. Yacoob. Tracking and recognizing facial expressions in image sequences, using local parameterized models of image motion. Technical Report CAR-TR-756, Center of Automation Research, University of Maryland, 1995.

[10] P. Boersma and D. Weenink. Pratt: doing phonetics by computer. http://www.fon.hum.uva.nl/praat/.

[11] G. Borshukov, D. Diponi, O. Larsen, J. P. Lewis, and C. Tempelaar-Lietz. Universal capture - image-based facial animation for "the matrix reloaded". In *Proceedings of ACM SIGGRAPH : Sketches and applications*. ESC Entertainment, ACM Press, 2003.

[12] M. Brand. Voice puppetry. In *Proceedings of ACM SIGGRAPH*, pages 21–28. ACM Press/Addison-Wesley Publishing Co., 1999.

[13] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of ACM SIGGRAPH*, pages 353–360. ACM Press/Addison-Wesley Publishing Co., 1997.

[14] S. Brennan. Caricature generator. Master's thesis, MIT, 1982.

[15] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *Proceedings of Non-Photorealistic Animation and Rendering*. ACM Press, 2000.

[16] Y. Cao, P. Faloutsos, and F. Pighin. Unsupervised learning for speech motion editing. In *Proceedings of the Symposium on Computer Animation*, pages 225–231. Eurographics Association, 2003.

[17] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial epression, gesture and spoken intonation, for multiple conversation agents. In *Proceedings of ACM SIGGRAPH*, pages 413–420. ACM Press, 1994.

[18] J. Cassell, H. Vilhjlmsson, and T. Bickmore. Beat: the behavior expression animation toolkit. In *Proceedings of SIGGRAPH*, pages 477–486. ACM Press, 2001.

[19] I. Cohen, A. Garg, and T. S. Huang. Emotion recognition from facial expressions using multilevel hmm. In *Neural Information Processing Systems, Workshop on Affective Computing*, 2000.

[20] M. Cohen and D. Massaro. Modeling coarticulation in synthetic visual speech. In *Computer Animation*. Springer-Verlag, 1993.

[21] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(6):681–685, 2001.

[22] E. Cossato. *Sample-Based Talking-Head Synthesis*. PhD thesis, Swiss Federal Institute of Techology, Switzerland, October 2002.

[23] CuriousLabs. Poser. http://www.curiouslabs.com/.

[24] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.

[25] D. DeCarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, 2000.

[26] D. DeCarlo, C. Revilla, M. Stone, and J. Venditti. Making discourse visible: Coding and animating conversational facial displays. In *Computer Animation*, pages 11–16. IEEE Computer Society, 2002.

[27] F. Dellaert, T. Polzin, and A. Waibel. Recognizing emotions in speech. In *Proceedings of International Conference on Spoken Language Processing*, volume 3, pages 1970–1973, Philadelphia, PA, 1996.

[28] G. Donato, M.S. Bartlett, J.C. Hager, and P. Ekman. Classifying facial actions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.

[29] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[30] G. J. Edwards, C. J. Taylor, and T. F. Cootes. Learning to identify and track faces in images sequences. In *Internation Conference in Computer Vision*. IEEE Computer Society, 1997.

[31] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference in Computer Vision*, volume 2, pages 1033–1038. IEEE Computer Society, 1999.

[32] P. Ekman and W. V. Friesen. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, 1978.

[33] I. Essa and S. Basu. Modeling, tracking and interactive animation of facial expressions and head movements using input from video. In *Proceedings of Computer Animation*, pages 68–69, 1996.

[34] I. Essa and A. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):757–763, 1997.

[35] I. A. Essa, T. Darrell, and A. Pentland. Tracking facial motion. In *Proceedings of IEEE Workshop on Nonrigid and Articulated Motion*, 1994.

[36] I. A. Essa and A. P. pentland. Facial expression recognition using a dynamic model and motion energy. In *Internation Conference in Computer Vision (ICCV)*, pages 36–42. IEEE, 1995.

[37] T. Ezzat, G. Geiger, and T. Poggio. Trainable videorealistic speech animation. In *Proceedings of ACM SIGGRAPH*, pages 388–398. ACM Press, 2002.

[38] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[39] Z. Ghahramani. Factorial learning and the EM algorithm. In *Advances in Neural Information Processing Systems*, pages 472–478. MIT Press, Cambridge, MA, 1995.

[40] M. Gleicher. Retargeting motion to new characters. In *Proceedings of ACM SIGGRAPH*, pages 33–42. ACM Press, July 1998.

[41] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[42] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proceedings of ACM SIGGRAPH*, pages 55–66. ACM Press, 1998.

[43] J. Hartigan and M. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[44] F. Hartung, P. Eisert, and B. Girod. Digital watermarking of MPEG-4 facial animation parameters. *Computers and Graphics*, 22(4):425–435, 1998.

[45] H. Hermansky, N. Morgan, A. Bayya, and P.Kohn. RASTA-PLP speech analysis technique. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 1, pages 121–124. IEEE, 1992.

[46] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, pages 1158–1161, 1995.

[47] G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions Royal Society B*, 352(1177–1190), 1997.

[48] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *Advances in Neural Information Processing Systems*, pages 3–10. Morgan Kaufmann Publishers, Inc., 1994.

[49] K. Honda. Interactions between vowel articulation and F0 control. In B. D. Joseph O. Fujimura and B. Palek, editors, *Proceedings of Linguistics and Phonetics: Item Order in Language and Speech (LP'98)*, pages 517–527. Hungary: Kakrolinum Press, Prague, 2000.

[50] P. Hong, Z. Wen, and T.S. Huang. Real-time speech driven expressive synthetic talking faces using neural networks. *IEEE Transaction on Neutral Networks*, 13(4):916–927, July 2002.

[51] T. Ishikawa, H. Sera, S. Morishima, and D. Terzopoulos. Facial image reconstruction by estimated muscle parameters. In *Proc. Third International Conference on Automatic Face and Gesture Recognition*, pages 342–347. IEEE Computer Society, 1998.

[52] G. A. Kalberer and L. Van Gool. Face animation based on observed 3D speech dynamics. In *Computer Animation*, pages 20–27. IEEE Computer Society, 2001.

[53] P. Kalra, A. Mangili, N. Magnenat-Thalmann, and D. Thalmann. *Smile: A multilayered facial animation system.* Springer-Verlag, 1991.

[54] K. Khler, J. Haber, and H-P. Siedel. Geometry-based muscle modeling for facial animation. In *Proceedings Graphics Interface*, pages 37–46, 2001.

[55] R. M. Koch, M. H. Gross, and A. A. Bosshard. Emotion editing using finite elements. *Computer Graphics Forum*, 17(3):295–302, 1998.

[56] C. Kouadio, P. Poulin, and P. Lachapelle. Real-time facial animation based upon a bank of 3D facial expressions. In *Computer Animation*, pages 128–136. IEEE, 1999.

[57] L. Kovar and M. Gleicher. Motion graphs. In *Proceedings of ACM SIGGRAPH*, pages 473–482. ACM Press, 2002.

[58] C. L. Lawson and R. J. Hanson. *Solving Least Square Problems*, chapter 23, page 161. Prentice-Hall, 1974.

[59] S. P. Lee, J. B. Badler, and N. I. Badler. Eyes alive. In *Proceedings of ACM SIGGRAPH*, pages 637–644. ACM Press, 2002.

[60] Y. Lee, D. Terzopoulos, and K. Walters. Realistic modeling for facial animation. In *Proceedings of ACM SIGGRAPH*, pages 55–62. ACM Press, 1995.

[61] Y. Li, T. Wang, and H. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of ACM SIGGRAPH*, pages 465–472. ACM Press, 2002.

[62] J. J. Lien, T. Kanade, J. F. Cohn, and C-C Li. Automated facial expression recognition based FACS action units. In *Proc. Third IEEE Int. Conf. Automatic Face and Gesture Recognition*, pages 390–395. IEEE, 1998.

[63] P. Litwinowicz and L. Williams. Animating images with drawings. In *Proceedings of ACM SIGGRAPH*, pages 409–412. ACM Press, 1994.

[64] D. Terzopoulos M. A. O. Vasilescu and. Multilinear analysis of image ensembles: Tensorfaces. In *Proc. 7th European Conference on Computer Vision*, pages 447–460, 2002.

[65] K. V. Mardia, J. T. Kent, and S. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[66] K. Mase. Recognition of facial expressions for optical flow. *IEICE Transactions*, E74(10):3474–3483, 1991.

[67] J. Noh and U. Neumann. Expression cloning. In *Proceedings of ACM SIGGRAPH*, pages 277–288. ACM Press, 2001.

[68] J. Ostermann. Animation of synthesis faces in MPEG4. In *Computer Animation*, pages 49–55. IEEE, 1998.

[69] E. Owens and B. Blazek. Visemes observed by hearing-impaired and normal hearing adult viewers. *Journal of Speech and Hearing Research*, 28:381–393, September 1985.

[70] F. I. Parke. Computer generated animation of faces. In *Proceedings of ACM National Conference*, pages 451–457. ACM, 1972.

[71] F. I. Parke. *A Parametric Model for Human Faces*. PhD thesis, University of Utah, 1974.

[72] F. I. Parke. Parameterized models for facial animation. *IEEE Computer Graphics and Applications*, 2(9):61–68, 1982.

[73] F. I. Parke and K. Waters. *Computer Facial Animation*. A K Peters, 1996.

[74] C. Pelachaud, N. I. Badler, and M. L. Viaud. Final report to NSF of the standards for facial animation workshop. Technical report, University of Pennsylvania, 1994.

[75] K. Perlin. Layered compositing of facial expression. Proceedings of SIGGRAPH : Sketches and Applications, 1997.

[76] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of ACM SIGGRAPH*, pages 75–84. ACM Press, 1998.

[77] F. H. Pighin, R. Szeliski, and D. Salesin. Resynthesizing facial animation through 3D model-based tracking. In *International Conference on Computer Vision*, pages 143–150. IEEE, 1999.

[78] S. M. Platt and N. I. Badler. Animating facial expressions. In *Proceedings of ACM SIGGRAPH*, pages 245–252. ACM Press, 1981.

[79] C. Poggi, F. Pelachaud, and F. de Rosis. Eye communication in a conversational 3D synthetic agent. *Special issue on Behavior Planning for Life-Like Characters and Avatars of AI Communications*, 2000.

[80] K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of ACM SIGGRAPH*, pages 501–508, 2002.

[81] L. Rabiner and B-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[82] L. Reveret and I. Essa. Visual coding and tracking of speech related facial motion. In *IEEE CVPR International Workshop on Cues in Communication*, 2001.

[83] B. Robertson. Mike, the talking head. *Computer Graphics World*, pages 15–17, July 1988.

[84] M. Rosenblum, Y. Yacoob, and L. Davis. Human emotion recognition from motion using a radial basis function network architecture. In *Workshop on Motion of Nonrigid Articulated Objects*, pages 43–49. IEEE Computer Society, 1994.

[85] K. R. Scherer. The functions of nonverbal signs in conversation. *The Social and Physhological Contexts of Language*, pages 225–243, 1980.

[86] Vicon Motion Systems. http://www.vicon.com/.

[87] A. Takeuchi and K. Nagao. Communicative facial displays as a new conversational modality. In *ACM/IFIP INTERCHI*, pages 187–193. ACM Press, 1993.

[88] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

[89] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. In *Proc. Third International Conf. on Computer Vision*, pages 727–732, 1990.

[90] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):569–579, 1993.

[91] Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):97–115, 2001.

[92] M. Turk and A. Pentland. Face recognition using eigenfaces. *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[93] M. A. O. Vasilescu and D. Terzopoulos. Multilinear subspace analysis for image ensembles. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 93–99, 2003.

[94] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Informat. Theory*, 13:260–269, 1967.

[95] K. Waters. A muscle model for animating three-dimensional facial expression. In Maureen C. Stone, editor, *Proceedings of ACM SIGGRAPH*, pages 17–24, July 1987.

[96] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH*, pages 479–488. ACM Press, 2000.

[97] Y. Yacoob and L. S. Davis. Computing spatio-temporal representations of human faces. In *Proceedings of Computer Vision and Pattern Recognition*, pages 70–75. IEEE, 1994.

[98] H. Yehia, T. Kuratate, and E. Vatikiotis-Bateson. Facial animation and head motion driven by speech acoustics. P. Hoole (ed). 5th Seminar on Speech Production: Models and Data, Kloster Seeon, Germany, May 1-4 2000.