Model-based Video Coding Techniques

by

Manson Siu

A dissertation submitted in partial fulfilment of

the requirements for the degree of

Master of Philosophy

Department of Electronic and Information Engineering

The Hong Kong Polytechnic University

January 2001

Dedication

To my parents

Abstract

This thesis is primarily concerned with how to integrate a model-based coding system. A large portion of this thesis is devoted to our proposed algorithms, which address the solutions for the difficulties encountered in a model-based coding system for real world applications. Apart from that, this thesis also covers our proposed model-based coding system.

In this thesis, we first discuss some existing model-based coding systems, and pinpoint the areas that limit the practicality of the application of model-based coding techniques. Typical approaches used in existing systems employ dedicated graphic models for different objects of interest. The objective of our work is to investigate and tackle the limitations that commonly encountered in practical applications of model-based coding. Accordingly, a robust model generation technique and a universal texture extraction technique are developed. Integrated with the aforementioned techniques, our system is capable of generating object models by deforming a graphic model in a model library according to the given image source. The deformation results are object models that well describe the objects of interest. This eliminates the necessity to install a specific model for each object of interest. Whereas, the proposed universal texture extraction technique provides a solution to obtain the texture information of an object without limits of the object's orientation and size.

We also investigated some facial animation techniques during the integration of our system. There are two advantages by implementing these facial animation techniques in our system. First, it results in a further reduction in the channel bandwidth required. This is achieved by the concise expression of the animation parameters for describing the change of a graphic model. Second, the possibility that extends the system to a synthetic coding system is preserved.

Finally, we provide some solutions to realize the critical parts of a system. These critical part significantly affect the practicality of a model-based coding technique. In a practical system, minimal knowledge requirement is necessary. Unlike most of the existing model-based coding systems, tailor-shaped head models are not required in our system. It is not necessary to provide a specific head model for a particular user. Furthermore, our system is designed to be scalable in alleviating the heavy computation overhead, and therefore it is favourable to real world applications.

Acknowledgements

I would like to give my sincere gratitude to whom has provided me guidance and encouragement throughout this study. Firstly, I must give my heartfelt gratitude to my sincere supervisor Dr. Y. H. Chan for his continuous support and providing direction of this research. I am thankful for his invaluable advises and suggestions for the study, as well as in writing up my thesis.

Special thanks are given to various bodies from the Hong Kong Polytechnic University, where I have the opportunity to work and to make discussions over my research topics. Thanks to my colleagues Dr. Y.L. Chan, Dr. S. O. Choy, Dr. S. T. Choy, Dr. Bonnie Law, Dr. C. T. Leung, Dr. W.C. Poon, Dr. Wilson Poon, Mr. W. L. Hui, Mr. K. W. Kwok, and Mr. K. K. Yiu, for their interesting discussions and their encouragement. I would also like to give my appreciation to all staff members of the department of Electronic and Information Engineering, especially, Professor W. C. Siu, Professor C. K. Tse, Dr. P. K. Lun, Dr. Charles Surya, Mr. W. H. Wong and clerical staffs of the General Office. For their supportive advises and provide me pleasant working environment over the years. It is my pleasure to acknowledge the Research & Postgraduate Studies Office of the Hong Kong Polytechnic University for her generous support.

Thanks are credited to my truly friends, thanks for the kindness to their supports whenever I were stressed, and share my happiness.

Finally, I must express my grateful thanks to my family members. Without their love and support throughout the years, this study would not have the chance to be completed.

Table of Contents

List of Publications

The following technical papers have been published or submitted for publication based on the result generated from this work.

1. M. Siu and Y.H. Chan, "A robust universal texture extraction technique for model-based coding", Proceedings, The International Symposium on Optical Science, Engineering, and Instrumentation, SPIE, Vision Geometry VIII, vol. 3811, pp.329-336, Denver, Colorado USA, July 1999.

2. M. Siu and Y.H. Chan, "An adaptive model generation technique for model-based video coding", IEEE 3rd Workshop on Multimedia Signal Processing, Copenhagen, Denmark, pp.357-362, September 1999.

3. M. Siu and Y.H. Chan, "A robust model generation technique for model-based coding," Proceedings, IEEE International Conference on Image Processing, pp.202-206, Kobe, Japan, 1999

4. M. Siu, Y.H. Chan, and W.C. Siu, "A robust model generation technique for model-based video coding," Accepted, to be published in IEEE Transactions on CAS for Video Technology.

List of Figures

List of Tables

List of Abbreviations

2D	:	Two-dimensional
3D	:	Three-dimensional
API	:	Application Program Interface
AU	:	Action Unit
DCT	:	Discrete Cosine Transform
FACS	:	Facial Action Coding System
FEP	:	Facial Expression Parameter
MPEG	:	Moving Picture Experts Group
PCS	:	The International Picture Coding Symposium
PSTN	:	Public Switched Telephone Network
SNHC	:	Synthetic/Natural Hybrid Coding
TTS	:	Text-to-Speech

List of Notations

X, Y, Z	:	X-axis, Y-axis, and Z-axis of a coordinate system
x, y, z	:	x, y, and z coordinates of a particular point in a coordinate system
T	:	Transformation matrix of Translation
R	:	Transformation matrix of Rotation
S	:	Transformation matrix of Scaling
$C_{i,j,k}$	:	Confidence level that describes the significance of a geometric coordinate of vertex j in view i, *where* $k \in \{x,y,z\}$ specifies the coordinate
γ	:	Angle between an emitted ray and X-Z plane
β	:	Angle between an emitted ray's projection on X-Z plane and Z-axis
$C_i(\gamma, \beta)$	:	Confidence level that describes the significance of an emitted ray in view i
$\psi_i(\gamma, \beta)$	:	Pixel value of an intermediate texture map associated with view i
$\psi(\gamma, \beta)$	:	Pixel value of a resultant texture map
$E_{current,k}$	:	Current extension of an Action Unit indexed by k
$E_{destiny,k}$	:	Destiny extension of an Action Unit indexed by k

$V_{k,effect}$	:	Geometrical coordinate of an effect vertex of an Action Unit, where $k \in \{x,y,z\}$ specifies the coordinate
$V_{k,attach}$	:	Geometrical coordinate that specifies the static end of a vertex of an Action Unit is attached, where $k \in \{x,y,z\}$ specifies the coordinate
D_k	:	Displacement of an attached vertex of an Action Unit along a particular coordinate axis, where $k \in \{x,y,z\}$ specifies the coordinate axis
D_{scale}	:	Scale of the displacement of an attached vertex of an Action Unit

Chapter 1

Introduction

1.1 General Introduction

Large bandwidth requirement is always a problem to video coding systems in multimedia communication. There are various kinds of coding schemes for digital video compression and communication. Typical examples include H.263, MPEG-1, MPEG-2, and MPEG-4. Recently, a new type of video coding method called model-based coding has attracted much attention as a potential candidate for very low bit-rate visual communication.

The applications of the aforementioned video coding standards stress on different aspects. H.263 is for video coding over a channel of very narrow bandwidth, such as Public Switched Telephone Network (PSTN). MPEG-1 and MPEG-2 are standards for digital video media storage and communication for different picture quality, whereas MPEG-4 is designed for multimedia applications and communication over a very low bit-rate channel.

MPEG-4, which is a newly launched video coding standard approved in October 1999, is a tailor-designed video coding scheme for multimedia applications. One of the most important concepts brought about is the object-based coding concept. When encoding a video sequence by means of object-based techniques, portions of objects that appear in the coding sequence will be identified and encoded separately. In the case where computer graphic models are used in the object description, this is known as a model-based coding technique.

In a model-based coding system, a source image undergoes certain processes to obtain a coded bit stream. Essential processes in a novel model-based coding system include object orientation extraction, geometrical transformation, dynamic feature extraction, and rendering. In order to suit a model-based coding system for communication, a generic model library is used in both the transmitter and the receiver of the system. A generic model library is defined as a set of predefined graphic models. With the application of the generic model library, better compression performance can be resulted by referencing the existing models.

Owing to the growing importance of applications using facial animation, model-based coding takes the most important role in MPEG-4 facial image coding area. Due to its favorable nature in low bit-rate communication, various research on adopting model-based coding, especially in video conferencing system, have been carried out. In realizing a video conferencing system, the human head is the most important part to be coded. The Synthetic/Natural Hybrid Coding (SNHC) in MPEG-4 addresses this application in facial expression animation, by characterizing facial gestures with a set of action parameters.

Inheriting the advantages of object-based coding and the growing importance of synthetic coding for multimedia applications, model-based coding becomes one of the hottest research topics in digital image processing. One of the significant facts is that model-based coding is the best way to encode video sequences in a less compact form. In our research, model-based coding systems for general practical applications are investigated. In this thesis, our proposed model-based coding scheme will be introduced. Its implementation and its advantages in real applications will also be described.

1.2 Objectives

Model-based video coding is the hottest member in object-based coding systems. Although there are a lot of approaches in its implementation, the three basic elements commonly involved in various systems are models, texture maps, and motion parameters of the objects involved. Relevant treatments for obtaining these elements such as model synthesizing, texture extraction, and motion tracking are computation-intensive and hence they are not favorable for real time applications.

In this project, our investigation starts with evaluating existing model-based coding systems. Throughout the evaluation, we found that, in most of the existing systems, graphic models and texture maps are predefined according to range scanning results of specific objects of interest. This leads to a significant shortcoming for practical model-based coding applications.

Our mission is to reduce the limits of current model-based coding. A robust model generation technique and a universal texture extraction technique are proposed to achieve this. The robust model generation technique is capable of synthesizing a model of a particular object based on a generic model. The universal texture extraction technique is capable of extracting textures from objects of any orientation and size. Our objective is to realize a system with minimal application overhead and make it applicable to real world applications. Owing to these motivations, our proposed scheme should cope well with minimal knowledge, and favorable to real world applications. Accordingly, the infrastructure of our algorithms should be optimized for effective computation.

In extension to the multimedia applications proposed in MPEG-4, our system should be extendable to the animation scheme proposed in MPEG-4. This means that the system should also be able to synthesize video sequence by animating objects with dedicated animation parameters.

1.3 Organization of Thesis

Following the introduction, the background of recent research in video coding will be given in chapter 2. In this chapter, both major video coding schemes and standards will be reviewed. Throughout the reviews, the revolution of model-based coding is pinpointed. The classification of coding techniques, as well as the conceptual ideas of model-based coding techniques, and the importance of model-based coding are followed.

In chapter 3, our proposed model-based coding scheme is raised for discussions. In this chapter, revisions of some computer graphics arithmetic, which are widely applied in existing model-based coding systems are given. In addition, comparison of our proposed scheme and traditional model-based coding scheme is raised. Moreover, details of functional components in our system are described.

In chapter 4, details of one of the proposed techniques, namely, a robust model generation technique is investigated. This technique addresses a common problem in model-based coding systems, where tailor shaped graphic models of objects of interest are always fundamental prerequisites. Details of the motivations of the introduction of this technique, modes of operations, assumptions, and procedures will be given. A series of simulation results for performance evaluations is followed.

In chapter 5, details of another proposed technique, namely, a universal texture extraction technique is raised. This technique addressses the applications of spherical texture maps. Dedicated texture extraction and update techniques will be given. The motivations of the introduction of this technique, as well as the detail procedures will also be given. A series of simulation results for performance evaluations is followed.

Chapter 6 is devoted to the integration of a synthetic coder of our system. This chapter illustrates the procedures for applying facial action units to the animation of a human head object model integrated in our system. Animation results are generated according to several action parameters for evaluations.

In chapter 7, conclusions of our works are given. Suggestions for future developments in our system, and the field of model-based video coding are provided.

Chapter 2

Background on Model-based Video Coding

In this chapter, we first discuss the historical background of Model-based Coding. We bring out the differences between traditional coding techniques and model-based coding techniques. The flow of a novel model-based coding system, its important features, and recent research will be discussed in this chapter. In addition, fundamental background for connecting chapters will also be discussed.

2.1 History and development of Model-based Coding

2.1.1 Precursors

In early 1980's, Model-based coding was first proposed when new thinking emerged to achieve very low bit rate video coding. Tracing back to the ascendant of Model-based coding, the earliest precursors was in 1851, a "comic electric telegraph", in which a flexible model of a face was demonstrated at the Great Exhibition in London by G.R. Smith [1]. The demonstration was carried out by distorting the face using magnets, which could be operated electrically at a distance. After a century, in 1961, Gabor and Hill [2] proposed an encoding process. The common objects were recognized and after recognition, a standard form of object was substituted in the encoder, which we call codebook coding nowadays. The author of "Signals, Systems and Noise", J. R. Pierce [3], wrote an imagination in his book. He imagined a model of human face was stored in the receiver. The transmitter would transmit the movements of eyes, lips and jaws of a real face to the receiver side. This was the prelusive concept of a model-based coding system.

During the 1970's there was groundbreaking work in the fields of computer graphics, computer vision, and psychology. A parameterized model of human face was developed by Parke [4] in 1975. The model used polygonal facets to construct a *wireframe* representation of the face. The composing facets were of varying size and the surface of model was rendered by Phong shading [5], which gave the appearance of smooth though plastic like skin. Parke also speculated that his parameterized models might be useful in medical field and in the data compression of image sequences. However, he did not suggest ways to extract the parameters from images of real faces.

Another important evolution of model-based video coding is traceable to the field of social psychology. There was research in understanding the relationship between the emotion and facial expression. "The Mechanisms of Human Facial Expression", a publication of the pioneering neurophysiologist called C. B. Duchenne du Boulogne [6], was first published in French in 1862. His fascinating photographs and commentary provided the foundations for experimentation in the perception and communication of human facial affects. Figure 2.1 shows the principal photographic subject, "The Old Man", in Duchenne's experiments.

Figure 2.1 Facial photographs from the book
"The Mechanism of Human Facial Expression"

In 1970's and 80's, there were numerous methods measure facial movements resulting from the action of muscles. Reviewing among such techniques [7], the Facial Action Coding System (FACS) [8] is the most comprehensive, widely used and versatile system. FACS is a scoring system for measuring facial expressions, developed by Ekman and Friesen in1978. FACS provides over 50 different facial actions, which can be combined to give various facial expressions.

2.1.2 Early Proposals of Model-based Coding

In 1981, Lippman [9] described a system called "Speechmaker" at the International Picture Coding Symposium (PCS) in Montrael, Canada. The Speechmaker was designed for the use in videoconferencing. There were 16 different primitive lip positions pre-stored in a videodisc. These selections were driven by speech, and thus composite video with no additional bits. Later on, Forechheimer and Fahlander [10] presented a system for low bit rates coding by making use of animation at PCS 1983 in Davis, CA. This paper describes most of the elements currently associated with MBC, such as the analysis of an input image sequences, the synthesis of a three-dimensional (3D) model of the object at the encoder side, the transmission of information to the receiver side for the reconstruction of the 3D model, and the composition of the decoded image sequence by projecting the reconstructed 3D model onto an image plane. Furthermore, Forechheimer's group subsequently developed a wireframe model of human head known as CANDIDE [11], which is widely used in recent model-based coding research. Figure 2.2 shows the model CANDIDE.

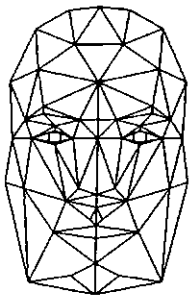


Figure 2.2 CANDIDE

2.1.3 Realistic Modeling

Upon the growing interest of model-based coding research, realistic modeling of an object becomes one of the hottest investigations. Triangular mesh with topology similar to the well-known CANDIDE model is employed to build the generic 3D models presented in a model-based coding system. Texture is then mapped onto the model to obtain photo-realistic appearance.

Texture mapping, which is originated in the field of computer graphics [12], states a strand evolution in the realism of synthesized models. To each vertex of a model, a fixed texture coordinate is assigned by projecting the vertex position onto the texture coordinate space. Common texture mapping procedures project images from selected frames in a video sequence onto a wireframe model. Aizawa, Harashima, and Saito [13][14][15], succeeded in constructing 3D model of a real face. The eye and mouth portions were copied and pasted from the input images. Their works demonstrated the remarkable power of texture mapping in producing realistic synthetic model.

The quality of a synthesized model is closely related to the precision of a wireframe model. A precise 3D model can be obtained by using a 3D shape acquisition system, such as a range scanner [16]. Cyberware [17], which was incorporated in 1982, was the pioneer in 3D

digitizing technology. Their 3030RGB/PS scanner and WB4 scanner are, respectively, used for scanning a human head and an entire body, which can be used to construct a 3D model at high resolution. Figure 2.3 shows the setup of Cyberware performing a whole-body scan.

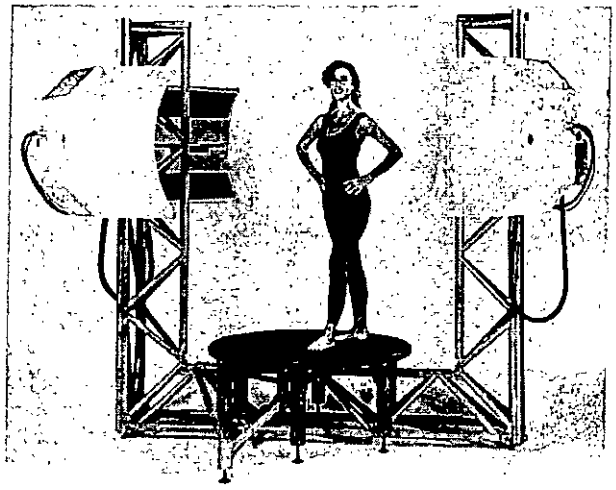


Figure 2.3 WB4 whole-body scanner by Cyberware

The WB4 whole-body scanning system scans the entire human body in one pass and creates high-resolution 3D models. Figure 2.4 shows some 3D graphic models with texture constructed by the scanned results of WB4 scanner.

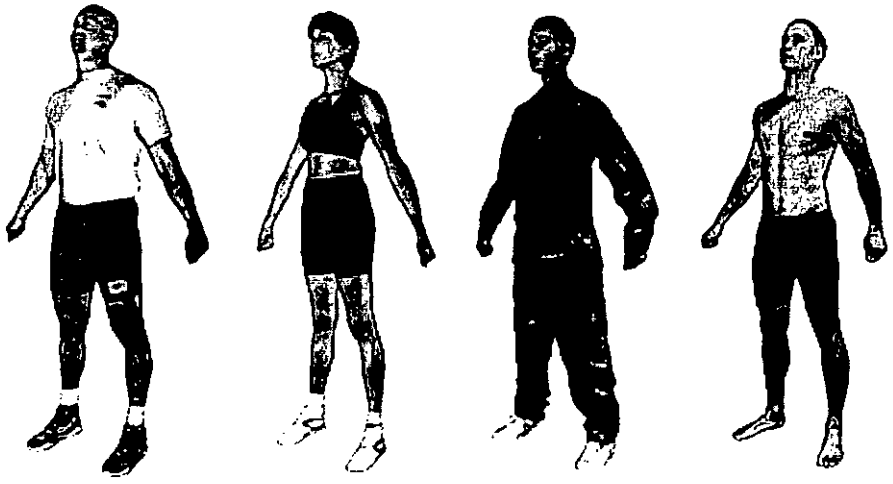
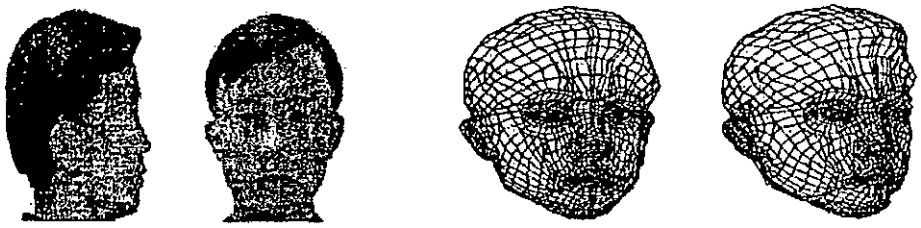


Figure 2.4 Reconstructed graphic models of Cyberware WB4 scanned results

Though Cyberware's solution produces accurate, high-resolution graphic models with textures, the equipment cost and its scanning procedures are not suitable for model-based communication systems.

A typical model-based communication system consists of a set of prototype graphic models in both the transmitting and receiving sides. One solution to model a specific object is to employ a strategy, which adjusts the prototype model to the object of interest. In the work of Lijun and Anup [18], a specific 3D human head graphic model is generated by fitting the prototype model in the generic model library to the front and side views of a person's head. Figure 2.5 shows the generated facial model by Lijun and Anup.



(a) Original images (b) Generated model

Figure 2.5 Generated individual facial model by Lijun and Anup [D4]

2.1.4 Classification

There have been several attempts in the classification of model-based coding. Harashima, Aizawa, and Saito [19] have located model-based coding in a generation table of coding developments. Table 2.1 shows the classification.

Generation of coding		Expected bit rate	Example of coding
0^{th} Generation	Direct Waveform Coding	10^{7} - 10^{8}	PCM
1^{st} Generation	Statistical Redundancy Reduction Coding	10^{5} – 10^{7}	Predictive coding, Transform coding
2^{nd} Generation	Structure/Feature Extraction Coding	10^{4} – 10^{5}	Contour coding, Segmentation coding
3^{rd} Generation	Analysis Synthesis Coding	10^{3} – 10^{4}	Model-based synthetic coding, Parameter coding
4^{th} Generation	Recognition Reconstruction Coding	10^{2} – 10^{3}	Knowledge-based synthetic coding, Command coding
5^{th} Generation	Intelligent Coding	10 – 10^{2}	Semantic coding

Table 2.1 Generation Classification of Coding by Harashima, Aizawa and Saito [19]

In the zeroth generation of the classification, the coding method assumed that the video is neither of any structure. The first generation method models statistical correlation between pixels. The second generation method segments object features from image, such as motion, contours and textures. Model-based coding is classified as the third generation technique. In this generation, objects are analyzed and synthesized. The fourth generation method recognizes, locates and segments objects from image, in which symbolic data are transmitted for the reconstruction of the objects. In the fifth generation method, speech and video are coded into meaning, intention or motives. According to the generation classification, increasing amount of intelligence is evident in each generation of coding system.

There was another classification proposed by Musmann in April 1994 [20]. His classification is based on the model used to represent the video source. Table 2.2 shows the classification.

Class	Source Model	Example of Coding
1	Pixel	PCM
2	Statistically dependent pixel, Block of pixels and colors	Predictive Coding, Transform Coding
3	Motion vectors	Motion compensated hybrid coding
4	Moving unknown object, such as shape, motion and color	Analysis-synthesis coding
5	Predefined object models	Knowledge-based coding
6	Symbolic parameters, such as action units for facial expressions	Semantic Coding

Table 2.2 Classification of Coding by Musmann [20]

Months later, Li, Lundmark, and Forchheimer [21] proposed another classification of very-low-bit-rate coding. They divided very-low-bit-rate coding into waveform-based coding and model-based coding, in which model-based coding was further divided into *semantic coding*, and *object-oriented coding*. Semantic coding uses specific object models, whereas object-oriented coding does not.

Owing to the variations in terminology used in different classifications, the term *Model-based coding* will be defined as a generic term. It covers all coding systems in which a model of an object is used in either coding or decoding process, as shown in figure 2.6.

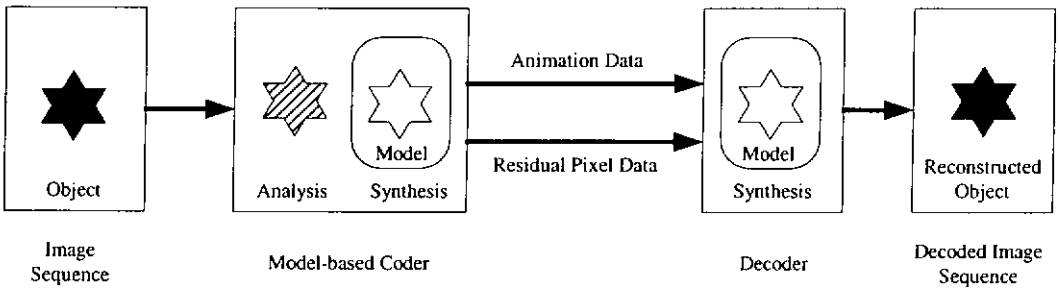


Figure 2.6 Novel System of Model-based Coding

2.2 Important Features

Model-based coding is a new image coding strategy, which is currently widely investigated. It signifies the parameter coding nature and the suitability in very-low-bit-rate coding. In a typical model-based coding scheme, prior knowledge about the scene in terms of models of object is used. Thus by using general object models, the actual scene can be represented with much less data as compared with statistical-based coding, which results in much higher compression ratios.

The coding philosophy of model-based coders is to analyze the input images by making use of models and generate encoded parameters to describe the actual scene. The encoded parameters are made up of the description of an actual scene instead of the correlation between image pixel contents. Therefore, the quality of a model-based coding system depends on how well the wireframe model can represent the actual object in a scene.

Basically, model-based coding has a systematic architecture, which can be divided into four correlated stages as shown in Figure 2.7. There are four major components in the transmitting side. Specifically, they are image analysis, image synthesis, model update and parameter encoding. As in the receiving side, parameter decoding, model update, and image synthesis are the major components.

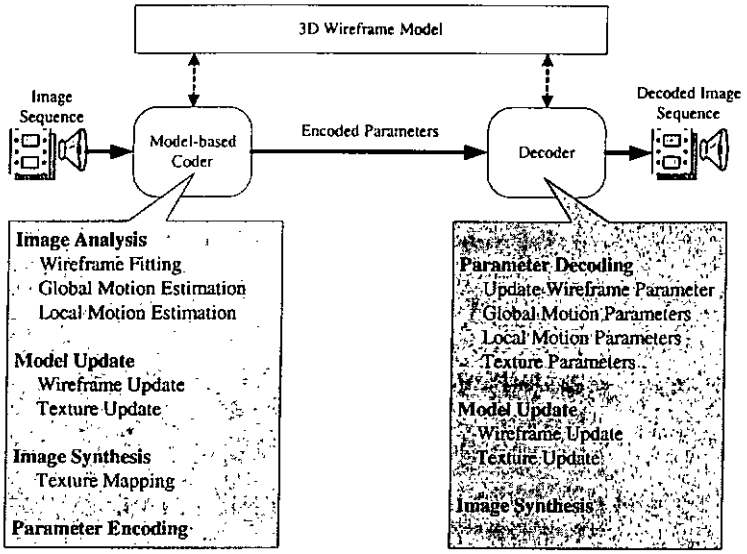


Figure 2.7 Features

2.2.1 Image Analysis

Image analysis is the first of all procedures in model-based coding. In analyzing an input image to the system, information such as the description of an object, the position of an object, the estimated global motion and estimated local motion should be obtained.

Early investigations of model-based coding scale and position a wireframe model by fitting the orthographic projection of the wireframe model to a frontal view of the actual human head object by using a set of manually selected feature points. Aziwa used four feature points to fit a human head to a model [22]. These four feature points are located at the tip of the chin, temples, and the point located in the midway between left and right eyebrows, as shown in figure 2.8.

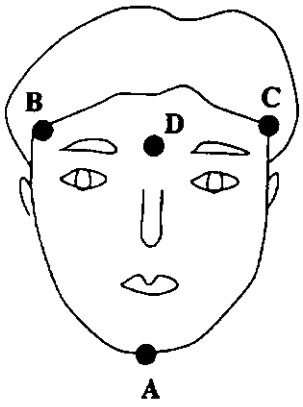


Figure 2.8 Feature points for fitting the wireframe used by Aizawa [22]

An alternative approach is to use snakes to model the boundary of the object. Choi used adaptive snakes to find the boundary of a head (figure 2.9) [23], which is claimed to be a robust method. However, neither methods can provide depth information of the head, since these methods are all based on a single frame.

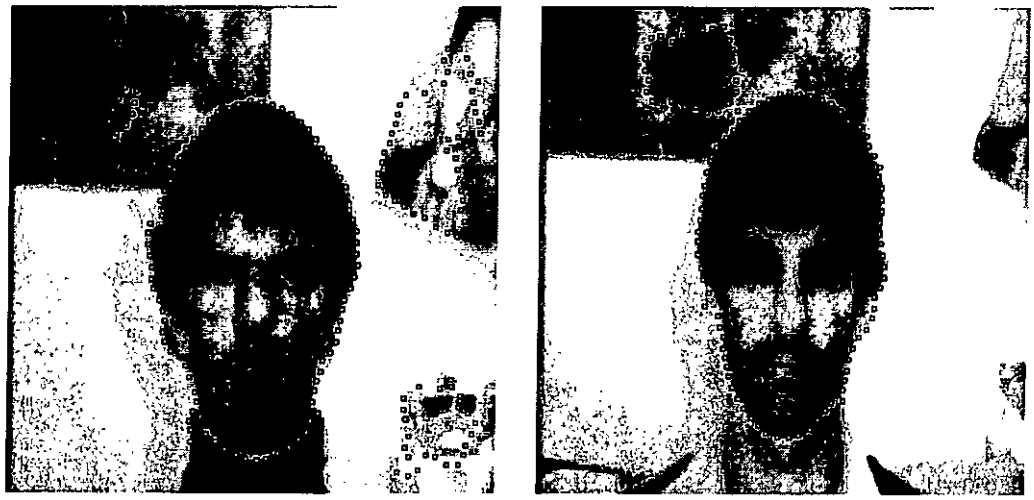


Figure 2.9 Adaptive snakes method to model the boundary of human head by Choi [23]

Apart from fitting an object to a model, the motion of an object should also be analyzed. The motion of a head can be decomposed into two components, say, the global motion of the head and the local motion caused by facial expressions such as motion of the mouth, eyebrows, and eyes. The global motion of a head can be characterized by six rigid motion

parameters of an affine transform given in equation 2.1, where (x_1, x_2) and (x'_1, x'_2) are the geometrical coordinates and transformed geometrical coordinates respectively. Estimation of these parameters can be done by using point correspondence or optical flow based approaches [24]. After compensating for the global motion of the head, Action Units are activated to best fit the residual motion field and the local motion is then estimated.

$$\begin{cases} x'_1 = ax_1 + bx_2 + c \\ x'_2 = dx_1 + ex_2 + f \end{cases} \tag{2.1}$$

2.2.2 Model Update

A model-based coding system involves a transformation of a wireframe model according to global and local motion parameters. The transformed model will be applied to obtain a texture image by using given texture mapping to obtain a texture image. Position and orientation parameters that describe the transformation are estimated through an image analysis on the given texture images. In the receiving side of a model-based coding system, one image is synthesized for each image frame. The frequency of model update is proportional to the frame rate of the coding system. Further discussion on our proposed model update approach will be brought up in chapter 4.

2.2.3 Image Synthesis

In order to produce realistic synthesized result, surface color information of the object of interest is acquired in the input images to compose the resultant model. This information is termed as "texture". In a model-based coding system, image synthesis is involved in both transmitting and receiving sides in different aspects. At the transmitting side, texture data of an object in input images is extracted and synthesized into a texture map. At the receiving

side, the texture map received is mapped onto the surface of a wireframe model. The texture-mapped model is then projected onto an image plane to synthesize an output image. The image synthesis manipulates texture extraction and texture mapping in the transmitting side and the receiving side respectively. Details of our proposed texture extraction approach will be discussed in chapter 5.

2.2.4 Parameter Coding

According to the generation classification described in [19], model-based coding is classified as a 3rd generation coding technique. Being a member of Analysis Synthesis coding, model-based coding codes image content into parameters, with the help of an analysis. Examples of coding parameters include objects' orientation, objects' motion, as well as facial expression parameters. Based on these parameters, the receiver combines wireframe information and texture information to reconstruct image scene. In chapter 6, we will further discuss our investigations in synthesizing facial expressions by a linear combination of FACS action units.

2.3 Applications of Model-based Coding

Model-based coding is becoming an important technique in image compression because it can reduce more redundancy than traditional block-based coding techniques. This advantage can be achieved by applying a synthesized graphic model in the image coding process, where the generation of the model is based on a set of action parameters. One of the major applications of model-based coding is in videophone, where scenes are generally restricted to head-and-shoulder movement. A predefined graphic human head model is commonly adopted in existing model-based coding systems for this application. Input images are encoded into an identification number of the speaker through object recognition,

degrees of transformations of the speaker model through motion estimation and a set of relative deformation rules between synthesized frames. As a result, the resultant encoded bit stream is obviously shorter than that of traditional block-based approaches.

Integration with Text-to-Speech (TTS) Synthesizing is another important application of model-based coding techniques. In a TTS Synthesizer, a speech synthesizer identifies the input text of a user, transforms it into a sequence of phoneme and reproduces audio output. Model-based coding techniques can generate corresponding video sequence accordingly for visual presentation.

Other applications of model-based coding such as game interfacing and facial interaction interfacing are of growing importance. Integration in these areas result in more precise encoded data and thus leads to better utilization of limited communication bandwidth.

2.4 Present Research

Anticipating the rapid convergence of the telecommunications industry, the computer industry and the entertainment industry, the Moving Picture Experts Group (MPEG) officially initiated a new MPEG-4 standardization phase in 1994. The goal of the MPEG-4 standard [25] addresses the need for coding natural and synthetic data, and high compression efficiency. Synthetic and Natural Hybrid Coding (SNHC) is a part of the MPEG-4 standard and becomes one of the hottest research topics in model-based coding.

SNHC allows the transmission of a 3D model and can be combined with 2D video streams. An example of combining traditional hybrid video coding methods with model-based coding has been proposed by Chowdhury et al. in 1994 [26]. In Chowdhury's work, a

switched model-based coder was introduced to switch between an H.261 coder and a 3D model-based coder to generate encoded output frame.

Besides SNHC, tracking facial features is another hot topic in model-based coding research [27][28]. A lot of previous works have been paid in tracking facial features from images. Among all these works, tracking with deformable templates is one of those methods often brought along with model-based coding [28]. A deformable template is a graphic model whose shape can be defined by modifying the parameters that describe the geometry of the template. Figure 2.10 shows an example of an eye deformable template developed by Yuille [29]. By shaping the template and analysing the image content, the template can be located and further information such as orientation information can be obtained.

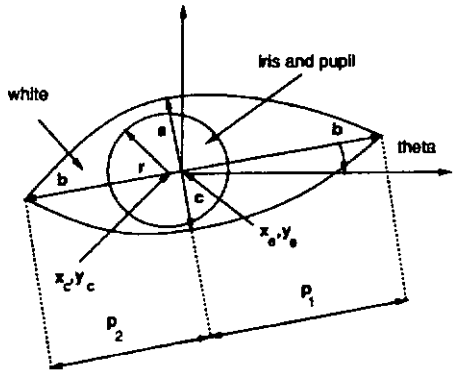


Figure 2.10 A deformable template of human eye by Yuille [29]

Research in reconstructing images from model-based coded results, such as model generation, texture extraction, merging extracted textures, and muscle animation are of equal importance. Due to the computation complexity involved, existing model-based coding systems are often unfavourable to practical real applications. In the forthcoming chapters, our solutions in tackling these limitations and their integration to our model-based coding system will be presented.

Chapter 3

Proposed Model-based Coding Scheme

3.1 Introduction

A novel model-based coding system involves an image analyser, an image synthesizer, a model deformation processor and a parameter coder. As we described in the previous chapter, model-based coding is an object-based analysis-synthesis coding technique, where computer graphics theory is involved in the image reconstruction procedures.

In this chapter, we first give a brief revision on computer graphics arithmetic, such as coordinate systems and geometrical transformations. These kinds of arithmetic are widely applied in all model-based coding systems. Secondly, procedural discussion on our proposed model-based coding scheme is given and the advantages of our proposed system are brought forth.

3.2 Revision on Computer Graphics Arithmetic

Traditional approach for rendering three-dimensional objects is to build a basic renderer, and then add various enhancements on to the system. A local reflection model such as Phong model is usually incorporated with a basic renderer. Most of these renderers work with objects that are represented by a set of polygons.

3.2.1 Coordinate Systems

One view of the geometric part of the rendering process is that it consists of a series of coordinate transformations of an object in an object library. The coordinate systems involved are common to all rendering systems and are fundamental to the understanding of three-dimensional computer graphics.

Local transformation and modeling could be easier if vertices of an object are stored with respect to a reference point, which is located in or near the object itself as shown in figure 3.1. This coordinate system is known as local coordinate system.

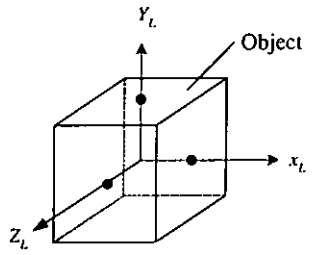


Figure 3.1 Local coordinate system

Once an object has been modeled, it is placed in the scene that we wish to render. All objects in the scene have their separate local coordination systems. The global coordinate system, which is commonly referenced by all objects in a scene, is known as the world coordinate system (figure 3.2). The definition of a scene can be thought of bonding of various local coordinates systems. All the objects have to be transformed into this common scene space in order that their relative relationships can be defined.

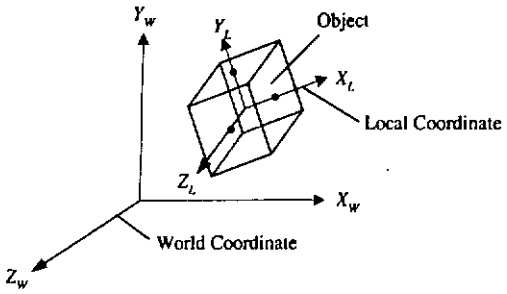


Figure 3.2 World coordinate system

Lastly, there is a coordinate system known as eye or camera coordinate system, which is used to establish viewing parameters and a view volume. A virtual camera is often used as a conceptual aid in computer graphics. This virtual camera can be positioned anywhere in the world coordinate space and point in any orientation. Figure 3.3 shows the positioning of a virtual camera which follows the requirements summarized by Pixar's RenderMan [30]. The viewing direction in the virtual camera obeys a left-handed coordinate system as shown in figure 3.4.

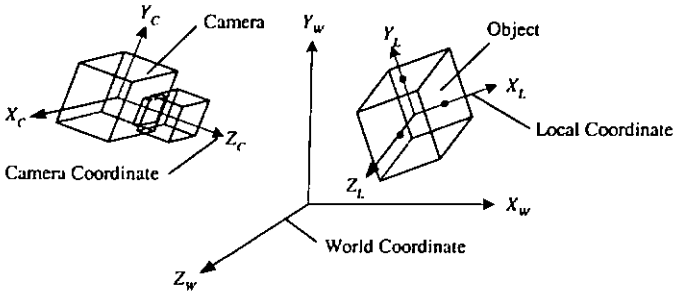
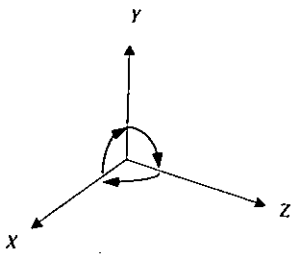


Figure 3.3 Camera coordinate system



Axis of Rotation	Direction of Positive Rotation
X	Y to Z
Y	Z to X
Z	X to Y

Figure 3.4 Left-handed coordinate system

3.2.2 Screen Space

Screen space is defined to be the space that acts within a closed volume called the viewing spectrum, which delineates the volume of space to be rendered. The fundamental transformation that takes us into screen space is the perspective transformation, which takes a point in the scene and projects it onto a view plane positioned at distance D away from the view point, and oriented normal to the viewing direction. As shown in figure 3.5, the screen coordinate (x_S, y_S) of the point P', which is defined to be the projection of the point P, are given in equations 3.1.

$$\begin{cases} x_S = D\dfrac{x_E}{z_E} \\ y_S = D\dfrac{y_E}{z_E} \end{cases} \qquad (3.1)$$

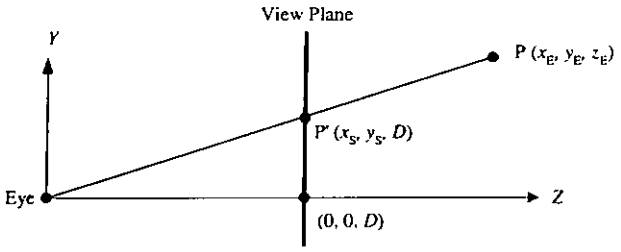


Figure 3.5 Perspective transformation

This implies that objects lying outside the viewing spectrum are not rendered. Suppose we have a square view plane window of size $2h$ x $2h$ and it is located symmetrically about the viewing direction as shown in figure 3.6. The four planes of the view frustum are then defined by:

$$\begin{cases} x_E = \pm\dfrac{hz_E}{D} \\ y_E = \pm\dfrac{hz_E}{D} \end{cases} \qquad (3.2)$$

There are two additional planes called the near and far clipping planes respectively. The clipping planes lie perpendicular to the viewing direction. In figure 3.6, they are defined as $z_E = D$ and $z_E = F$.

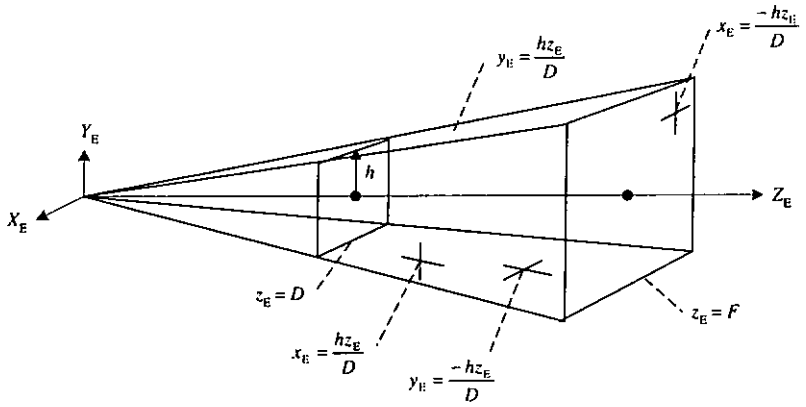


Figure 3.6 Definition of view frustum

Now, let us consider the third component of the screen space, namely z_S. It has been shown in equation 3.3 that the transformation of z takes the form:

$$z_S = A + \frac{B}{z_E} \qquad \text{where } A \text{ and } B \text{ are constants} \qquad (3.3)$$

By normalizing the range of z_S values, the range $z_E \in [D, F]$ maps into the range $z_S \in [0,1]$. The full perspective transformation is then given in equation 3.4.

$$\begin{cases} x_S = D\dfrac{x_E}{hz_E} \\ y_S = D\dfrac{y_E}{hz_E} \\ z_S = F\dfrac{1 - D/z_E}{F - D} \end{cases} \qquad (3.4)$$

Let us consider the relationship of z_E and z_S a little more closely. Both of them provide a measure of the depth of a point. Interpolating along a line aligning with the viewing direction in the eye space is not the same as interpolating in the screen space.

3.2.3 Homogeneous Geometrical Transformation

In a computer graphics system, geometrical transformation is used to formulate the change of an object's position, orientation, and size. This will be discussed in this section.

Homogeneous coordinates were first developed in geometry [31][32], and have been applied in graphics [33][34]. In homogeneous coordinates, an additional coordinate is added to a point. Instead of being represented by a pair of numbers (x, y), each point is represented as (x, y, w) with w not equal to zero.

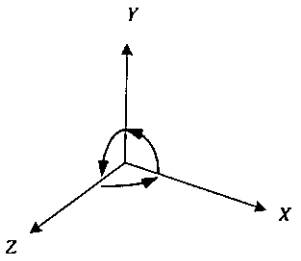
The translation, scaling, and rotation of a point from $P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ to $P' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ are respectively formulates as,

$$P' = T \times P \quad \text{where} T = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (3.14)$$

$$P' = S \times P \quad \text{where } S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$P' = R \times P \quad \text{where } R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

As described earlier, it is often necessary to move an object from one space to another space in a computer graphics system. Therefore, the notation used to describe the transformation in such a 3D renderer has been firmly established, which is known as the 4×4 homogeneous transformation matrix. A three-dimensional transformation is represented by a 4×4 matrix, where the representation of a point is of the form (x, y, z, w).



Axis of Rotation	Direction of Positive Rotation
X	Y to Z
Y	Z to X
Z	X to Y

Figure 3.8 Right-handed coordinate system

Figure 3.8 shows a right-handed coordinate system. Translation and scaling in a three-dimensional space is a simple extent to that of two-dimensional space. Let (d_x, d_y, d_z) and (s_x, s_y, s_z) be the translation and scaling amount along the X, Y, and Z axis respectively. The translation and scaling matrices are then respectively given as,

$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

As for the rotation along X, Y, and Z axis, they are respectively represented by,

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

and

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

3.2.4 Rendering Pipeline

The geometric transformation introduced in the previous section is only one of the processes introduced in a rendering pipeline shown in figure 3.9. The remaining processes are rasterization, hidden surface removal and shading. They are carried out concurrently in the screen space.

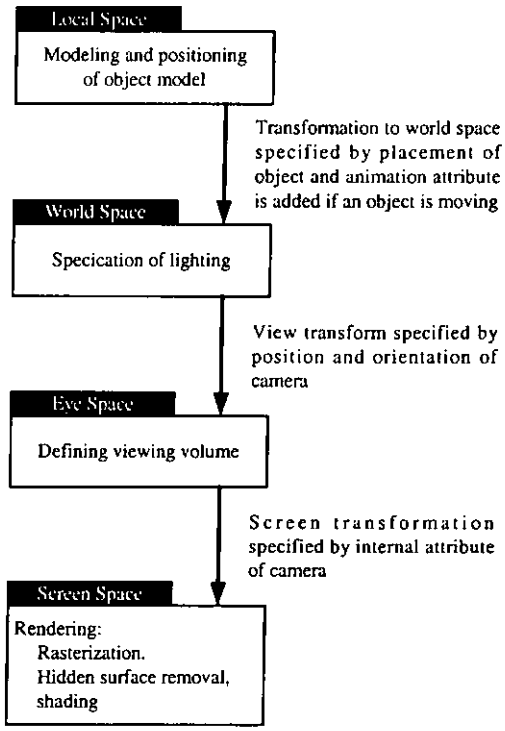


Figure 3.9 The rendering pipeline

Rasterization, hidden surface removal and shading are the three core processes carried out in the renderer. These processes are operated at polygonal level. One polygon is operated at a time. We can consider the three processes as three two-dimensional linear interpolation processes. Rasterization interpolates area between vertices to find the coordinates that define the limit of a span, and obtain the pixels between the span limits. Hidden surface removal interpolates z-coordinates along the viewing direction, which provides a depth value for each pixel from depth values of vertices. Shading implies interpolation on vertex intensities for finding out the intensity of each pixel.

To perform an interpolation, information such as vertex coordinates for rasterization, vertex depth values for hidden surface removal, or vertex intensities for shading are required.

In general, the overall procedural structure of the rendering process can be summarized as shown in figure 3.10:

```
for (each polygon) {
      perform geometric transformations into screen space
      for (each scan line within a polygon) {
            find spans by interpolation and rasterize span
            for (each pixel within the span) {
                  perform hidden surface removal and shade
            }
      }
}
```

Figure 3.10 Procedural structure of rendering process

3.2.5 Hidden Surface Removal

Many hidden surface removal algorithms have been developed. Most of them are described and categorized in a classic paper written by Sutherland *et al* [35]. An excellent and comprehensive algorithm for hidden surface removal is known as *Z*-buffer algorithm, which can to be found in [36].

The *Z*-buffer algorithm operates in a three-dimensional screen space. Each pixel is associated with a two-dimensional screen coordinate (x_S, y_S), together with a depth value. The *Z*-buffer is initialized to the depth of the far clipping plane. For each pixel, we compare its interpolated depth with the depth already stored in the *Z*-buffer (x_S, y_S). If the value is less than the stored value, then the corresponding pixel is closer to the viewer than the stored one. The shading value of the corresponding pixel is written to the screen memory, and the current depth is placed in the *Z*-buffer.

A Z-buffer occupies more memory than a corresponding frame buffer. Perspective projections further increase the memory requirement as objects at a different depth may transform into the same z-value after perspective divide.

3.3 Proposed Model-based Coding System

The model-based coding system is the hottest member in object-based coding systems. There has been vast amount of research carried out on relevant topics such as feature extraction [37], motion tracking [38] and model-synthesizing techniques [18][39]. A typical system model of video communication systems that adopt conventional model-based coding schemes is shown in figure 3.11. In this system model, a generic model library is required. The generic model library is a collection of predefined graphic models, which can be used for image analysis, feature extraction, and parameter coding. The usage of predefined models can reduce bit rate further by saving the transmission overhead, which carries the model information. However, these predefined models are usually tailor-shaped for specific objects, and cannot adapt dynamically to the new information provided in the course of application. Thus, the usage of tailor-shaped graphic models limits the practicality of a model-based coding system.

In our investigation, automatic model generation and texture extraction are essential parts of a practical model-based coding system. In a practical system, a graphic model from the generic model library should be able to deform itself into a number of models of different but similar shapes to describe different objects of similar shapes rather than remain static to represent a single object.

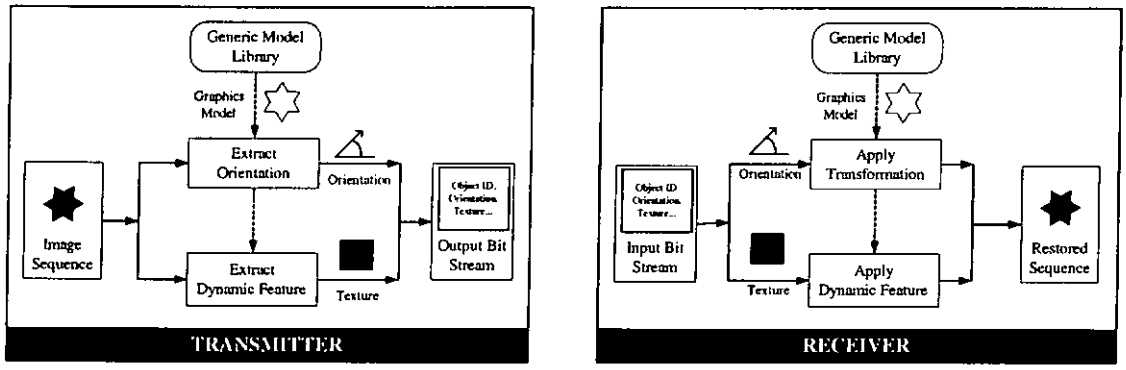


Figure 3.11 Video Communication Model of a Basic Model-based Coding System

Although there are a lot of approaches in implementing a model-based coding system, three basic elements, namely, models, texture maps, and motion vectors of the objects are commonly involved in a system. At the decoder side of a system, corresponding models are adjusted with the motion vector received, and video frames are reproduced by rendering the adjusted models with their texture maps. Texture mapping is a process that maps an image on to the model surfaces. The image involved is called texture map. The usage of texture mapping can result in highly realistic rendered results. Therefore, texture mapping is vitally important to the systems. However, only little research effort has been paid to texture extraction.

The objective of our research is to improve the practicality of model-based video coding techniques. We propose a model-based video coding system where automatic model generation and texture extraction are integrated. An animation scheme, which is compatible to future extension of MPEG-7 is also developed.

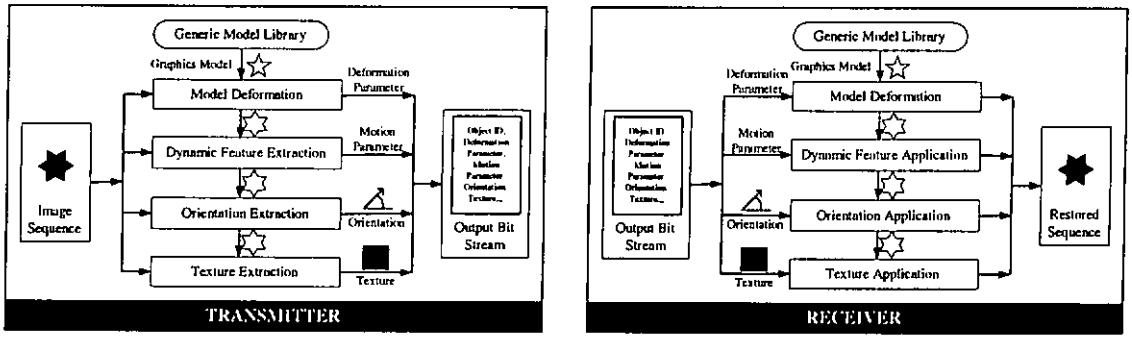


Figure 3.12 Proposed Model-based Coding Scheme

Figure 3.12 shows the flow of our proposed model-based coding scheme. In our proposed scheme, there are four procedures in both transmitting and receiving sides. These procedures can be generalized into four stages, namely, orientation manipulation, dynamic feature manipulations, model deformation, and texture manipulation.

Before going into details of the aforementioned procedures, a discussion on pre-processing techniques is given. This addresses some assumptions and minimal knowledge required to understand how our system operates.

3.4 Pre-processing Procedures

3.4.1 Model Validation

A validation of an object model in generic model library is necessary before any modification or update is made to the model. One of the possible errors that may exist in the model is caused by the existence of information redundancy. The unnecessary information existed in a model may lead to discontinuity of model surface. Figure 3.13 shows a typical example of the potential problems caused by redundant vertex information.

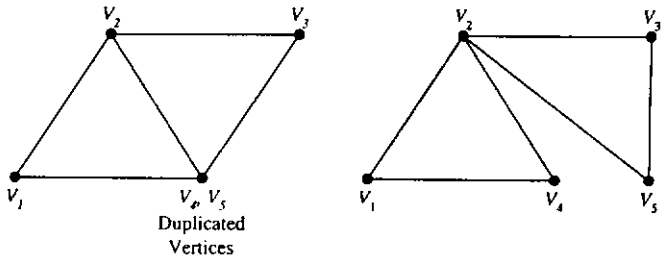


Figure 3.13 Faces discontinuity caused by redundancy vertex information.

The discontinuity due to duplicated vertices will lead to unexpected error in weighting algorithm. The duplicated vertices may not have an edge connected and hence duplicated vertices may not response correctly to the changes. In the example, vertices V_1, V_2 and V_3 are supposed to be the connected vertices of V_5. However, there is a duplicated vertex of V_5, say V_4, and V_1 is connected to V_4 instead of V_5. Due to this duplicated information provided, the change in V_5 will not be able to propagate to V_1 as illustrated in the example.

There are two ways to remove this potential risk. First, one may detect and remove the duplicated vertex and update the face information. Second, a new edge of length zero is added to connect the duplicated vertices. The former approach results in a more precise model, procedural description is given in Figure 3.14.

```
for (i = 1 to number_of_vertices) {
      if (vertex(i) == redundancy)  {
            remove vertex(i)
            for (j=1 to number_of_faces)  {
                  if (vertices_of_face(j) == vertex(i))
                        vertices_of_face(j) = duplicated_vertex
            }
      }
}
```

Figure 3.14 Procedures in removing vertices redundancy

3.5 Orientation Manipulation

In order to extract matching information for generating the object's model with the generic model on hand, the orientation of the object appeared in the captured image or video frames must be known. The orientation information is a global parameter and this piece of information can be obtained via an analysis of the image context. Various researches for this purpose have been done. Typical methods are based on template matching [40], face feature matching [37], point matching [38], or matching using eigenfaces.

Template matching uses the Karhunen-Loeve transformation of a set of object pictures for matching [41][42]. However, in real world applications, the orientations of a head cannot be reliably recognized by template matching due to the fact that physical transformation of an object and illumination may cause the appearance of the object to vary substantially. A more reliable result can be obtained by extending the eigenface technique presented by Turk and Pentland [43] to determine the orientation of an object. In [44], Moghaddam and Pentland make use of eigenfaces to detect the orientation of a head. Accuracy up to 92% can be achieved on average.

3.6 Dynamic Feature Manipulation

The dynamic feature information is represented by the local motion information over the surface of an object. Consider a human head as the object of interest. Facial expressions are considered as the dynamic feature information to be manipulated. There is a significant amount of research in describing facial expression information. A distinguish system called Facial Action Coding System describes facial expression by enumerating action units of a face. There have been several attempts to extract dynamic features of a face. Terzpoulos and Waters [45] developed a method to trace linear facial features by making use of active

contour models. Mase [46][47] has developed a method to track action units by using optical flow. Li, Roivainen, and Forchheimer [48] described another approach in which a control feedback loop between computer graphics and computer vision processes is used.

In our proposed system, dynamic features are represented in action units, where the action units are based on the reference of the Facial Action Coding System presented by Ekman and Friesen [49]. The receiver collects action unit parameters and modifies action units of the graphic model accordingly to generate movements as well as gestures. Finally, the updated model is put into a rendering pipeline and render result is then obtained.

3.7 Model Deformation

In most of the existing model-based coding systems, dedicated models for specific objects are used. These models are well defined for the object to be coded, and are applied in the analysis-synthesis procedures directly. Common methods in generating these object models are based on the range data of the object. The range data can be estimated from the disparity map in stereo graphics [50] or obtained from the depth measurement in laser scanning [17]. However, the equipment cost in laser scanning and the limitation that specific views are required in stereo graphics make these aforementioned methods infeasible in practical applications.

A new technique that generates a particular model by deforming the graphic models in the model library is proposed in our system. The deformation results give graphic models that well describe the objects of interest without necessary to install a different model for each object of interest. Furthermore, the proposed algorithm also breaks through some limitations

found in the existing model generation approaches such as that the orientation and the size of the object must not deviate from those of the model in the library.

3.8 Texture Manipulation

The application of texture takes part in every model-based coding system so as to produce photo-grade results. Existing model-based coding systems map predefined textures to graphic models. These textures are obtained through direct projection of source images to the model or through canonical projection in a laser scanning process. These methods have weaknesses. The generated texture is only valid for a specific view in the former case and for a specific rotation axis in the latter case.

The texture extraction technique implemented in our proposed system extracts texture information without orientation limitation. Each input image of specific orientations provides particular texture information. They are combined together to generate a single resultant texture map. Furthermore, the resultant texture can be updated gradually. By doing so, it is not necessary to generate multiple textures for a single object of interest.

3.9 Summary

In this chapter, we have revised some computer graphics arithmetic and introduced our proposed model-based coding system. The revised topics provide some essential computer graphics knowledge to understand how a model-based coding system works.

In most of the existing model-based coding systems, the models and the texture information used in the coding system are dedicated for particular objects. However, in our investigations, models and texture that are generated dynamically based on the input object

are far more useful than predefined specific information in practical applications. This is the motivation of our research work. Therefore, in our proposed system, a model deformation stage and a texture manipulation stage are implemented. The model deformation stage generates a particular model by deforming an available graphic model in the model library, while the texture manipulation stage extracts texture information to generate a single global texture for an object.

Computer graphics arithmetic such as homogenous geometrical transformation is presented in this chapter. The description of a rendering pipeline is also given. A preprocessing technique called model validation technique is also discussed. This discussion addresses some potential risks and provides solutions to these problems.

In next chapter, we will explore the realization of a dynamic model generation technique and its applications. Some simulation results will also be presented for evaluating its performance.

Chapter 4

Robust Model Generation Technique

4.1 Introduction

Vast amount of research has been carried out on relevant topics such as feature extraction [37], motion tracking [38] and model-synthesizing techniques [39][40], but model generation is still a time consuming procedure that is not attractive for real time applications.

In conventional model-based coding schemes, pre-defined models are generally used. These models are static and can neither be adaptive to fit the real features of the object of interest nor be updated dynamically in the course of application. Accordingly, a very specific model is necessary for a particular object and one cannot use a generic model to generate models for similar objects whenever it is necessary so as to save memory space. Therefore, for applications involving real-time video coding, a simple model generation method that can easily construct a model or modify it dynamically according to the scenes processed is necessary.

There are methods for generating an object model with stereo graphics [50] and laser scanning [17]. Laser scanning can provide a very accurate model for an object, but the size and the cost of the equipment involved makes it infeasible for the aforementioned applications. As for methods based on stereo graphics, they generally work only on specific

views. Because of these reasons, practical model synthesis technique becomes an important investigation in our model-based coding research.

In this chapter, we propose a new technique to generate a model for a particular object with a generic model. The proposed approach has several advantages over the conventional methods using stereo graphics. First of all, no specific view with known orientation of the object is required. Second, updating the generic model with information extracted from multiple views is supported. Third, feature extraction processes for different views can be done independently and hence parallel processing is allowed. Accordingly, our method is comparatively more efficient and practical for actual applications.

4.2 Robust Model Generation

In the proposed approach, a generic model for objects of similar shapes is used in the model-based encoding and decoding processes. The usage of generic model can achieve very low bit rate coding that benefits communication applications. The idea of using generic model is to use a standard collection of graphic models to represent standard objects. However, a single object may have various shapes that a standard model may not well represent them. Our proposed algorithm can solve this problem easily as it can make a generic model dynamically adapt to the shape of the object of interest whenever new information is available.

Note the proposed algorithm can actually operate in two modes for different applications. In its first operation mode, a target model can be generated as a pre-defined model for later applications based on different views of the object of interest. In its second operation mode,

the target model can be gradually built or updated with new information extracted from new video frames being processed. Parallel processing is allowed in both circumstances.

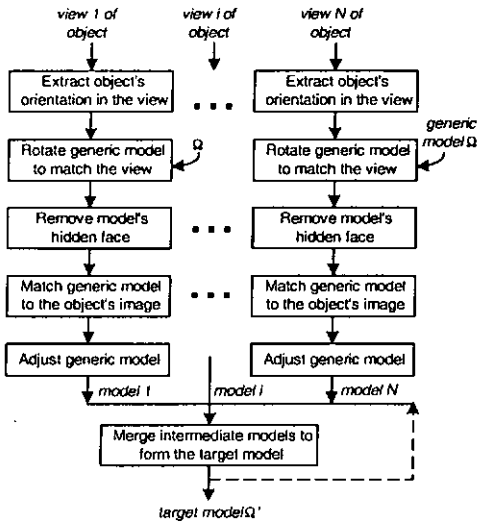
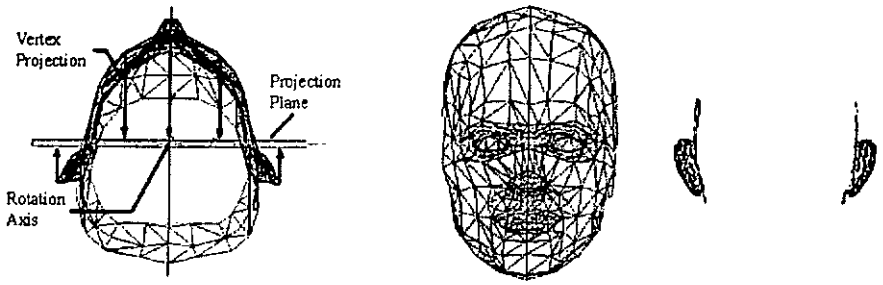


Figure 4.1 Model Synthesis Process Flow chart

Figure 4.1 shows the process flow of the proposed model generation scheme. In the following part of this section, we shall discuss the processes involved in details.

4.2.1 Hidden surface removal

The first process to come with is the orientation extraction process. After identified the orientation of the object, the generic model is rotated accordingly to match the orientation. The hidden surface of the model is then removed with a modified Z-buffering algorithm. In order to increase the efficiency and reduce the computation effort, the vertices are projected onto a projection plane that passes the center of the model and is parallel to the view plane, instead of the horizontal plane that passes through the origin of the world coordinates. Figure 4.2 shows the seen face of a head model of orientation (0,0,0) and how it is obtained. The left portion of figure 4.2b shows the surface closer while the right one shows the surface farther than the projection plane to the viewpoint.



(a) Vertex Projection (b) Seen Surface

Figure 4.2 Hidden surface removal

4.2.2 Matching a generic model to an object's image

The processed model will then undergo a model matching procedure. First of all, a projected view of the processed model is obtained by projecting the model onto the view plane with orthographic projection. Then, the boundary information of this projected view and the object's image is extracted. After translating the generic model to make the center of its projection coincident with the center of the object's image, the model is adjusted such that the boundary of the projected view of the model matches that of the object's image.

Besides the boundary vertices, remarkable feature points like eye corners, mouth corners, and ears are also modified in this process. The corresponding feature points on the generic model will be matched to that of the object's image by making use of feature matching techniques. The feature matching technique proposed by Tang and Thomas in [39] is adopted in our system to match face features. Figure 4.3 shows the intermediate result of the process.

(a) (b) (c)

Figure 4.3 Model Matching: (a)Projected view of a model (b)Corresponding positions of the vertices appeared at the boundary of the model's projected view (c)Adjusted model

4.2.3 Modifying generic model

Modifying the position of a vertex in an object model will result in a change of the positions of its connected vertices. Their change is not uniform. The closer a vertex from the adjusted vertex, the more significant the change should be. The modification algorithm we adopted here is an iterative algorithm.

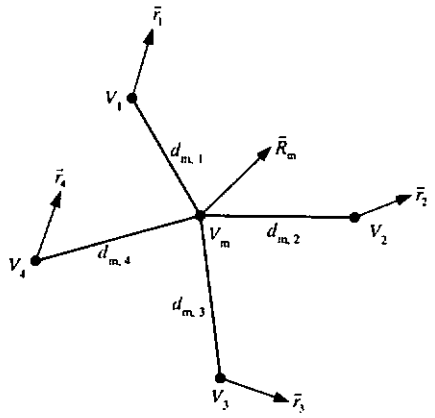


Figure 4.4 Corresponding resultant displacement and its connected neighbours

Figure 4.4 shows a particular vertex V_m of the model and its connected neighbors. In general, the resultant displacement of V_m due to its neighboring vertices' displacements, say $\bar{R}_m$, is given as

$$\vec{R}_m = \sum_{V_n \in \Lambda_m} \vec{r}_n \times \frac{w_{m,n}}{w_m} \tag{4.1}$$

where $\vec{r}_n$ is the radial displacement of vertex V_n, Λ_m is the set of V_m's connected neighboring vertices. The weighting parameters $w_{m,n}$ and w_m are, respectively, defined as

$$w_{m,n} = \frac{\sum_{V_k \in \Lambda_m} d_{m,k}}{N \cdot d_{m,n}} \tag{4.2}$$

and

$$w_m = \sum_{V_n \in \Lambda_m} w_{m,n} \tag{4.3}$$

where N is the total number of elements in Λ_m and $d_{m,n}$ is the distance between vertices V_m and V_n.

The vertices are updated one by one until all of them are processed. These procedures are then repeated again and again until there is no more change of the resultant displacement for every vertex in the model.

4.2.4 Updating resultant model

For a particular view of the object, a new model can be obtained by adjusting the generic model with the aforementioned procedures. For the sake of reference, we refer to the model associated with view orientation i as Ω'_i. This intermediate model will then be used to update the target model Ω' in order to produce a natural model.

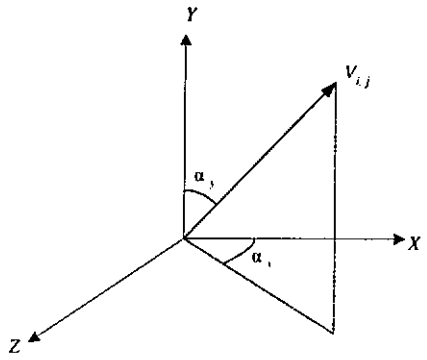


Figure 4.5 Representation of a corresponding vertex direction i

Let the vertices of model Ω'_i be $\{V_{i,j} : j=1,2...K\}$, where K is the total number of vertices of the model. The line connecting the origin to a particular vertex $V_{i,j}$ defines a vector $(\alpha_{i,j,x}, \alpha_{i,j,y})$ as shown in figure 4.5. This vector is used as an input parameter to calculate $C_{i,j,x}$, $C_{i,j,y}$ and $C_{i,j,z}$, the confidence levels of the three components of $V_{i,j}$'s coordinates with respect to its counterparts $V_{k,j}$'s, where $k \neq j$. Specifically, $C_{i,j,x}$, $C_{i,j,y}$ and $C_{i,j,z}$ are given by

$$
\begin{aligned}
C_{i,j,x} &= \left|(\cos\alpha_{i,j,x} \times \sin\alpha_{i,j,y}) \times \cos\theta_{i,y} - (\cos\alpha_{i,j,x} \times \cos\alpha_{i,j,y}) \times \sin\theta_{i,x} \times \sin\theta_{i,y}\right| \\
C_{i,j,y} &= \left|(\cos\alpha_{i,j,x} \times \cos\alpha_{i,j,y}) \times \cos\theta_{i,x}\right| \\
C_{i,j,z} &= \left|(\cos\alpha_{i,j,x} \times \sin\alpha_{i,j,y}) \times \sin\theta_{i,y} - (\cos\alpha_{i,j,x} \times \cos\alpha_{i,j,y}) \times \sin\theta_{i,x} \times \cos\theta_{i,y}\right|
\end{aligned}
\tag{4.4}
$$

where $\theta_{i,x}$, $\theta_{i,y}$ and $\theta_{i,z}$ is the degree of orientation rotated perpendicular to *X*-axis, *Y*-axis, and Z-axis respectively.

Hereafter, this set of equations is referred to as *significance equations* as it tells the significance of a particular vertex component of Ω'_i in updating Ω'.

The coordinate of the corresponding vertex in Ω', say $(V'_{j,x}, V'_{j,y}, V'_{j,z})$, is then updated with the following set of rules accordingly.

$$V'_{j,x} = \begin{cases} V_{i,j,x} & \text{if } C_{j,x} < C_{i,j,x} \\ V'_{j,x} & \text{else} \end{cases}$$

$$V'_{j,y} = \begin{cases} V_{i,j,y} & \text{if } C_{j,y} < C_{i,j,y} \\ V'_{j,y} & \text{else} \end{cases} \qquad (4.5)$$

$$V'_{j,z} = \begin{cases} V_{i,j,z} & \text{if } C_{j,z} < C_{i,j,z} \\ V'_{j,z} & \text{else} \end{cases}$$

Thresholds $C_{j,x}$, $C_{j,y}$ and $C_{j,z}$ are, respectively, the maximum values of $C_{k,j,x}$'s, $C_{k,j,y}$'s and $C_{k,j,z}$'s for all processed views k's . They are all initialized to be zero at the very beginning. Since the computation of $C_{i,j,x}$, $C_{i,j,y}$ and $C_{i,j,z}$ is independent of each other, the model can be updated with respect to the 3 coordinate components in parallel. This increases its efficiency.

In our approach, the target model is initialized to be the generic model. The updating rule (4.5) guarantees that the target model can eventually converge into a stable state after a certain number of iterations on different views.

4.3 Simulation Results

Simulation has been carried out to evaluate the performance of the proposed algorithm. In our simulation, based on the generic model shown in figure 4.6, a single spherical texture map is first generated with 3 different images shown in figure 4.7. The generic model is then fitted to the three images shown in figure 4.7 according to the object's orientations in

the images. Three corresponding intermediate models are generated accordingly. Figure 4.8, 4.9 and 4.10 show the generated results.

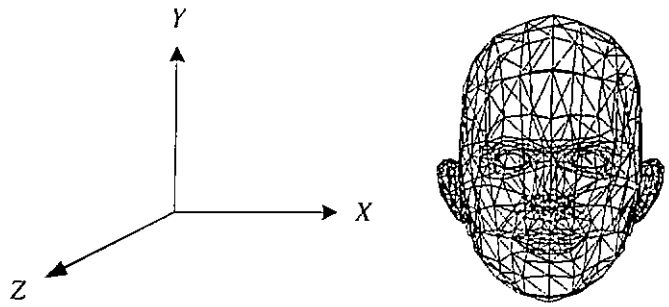
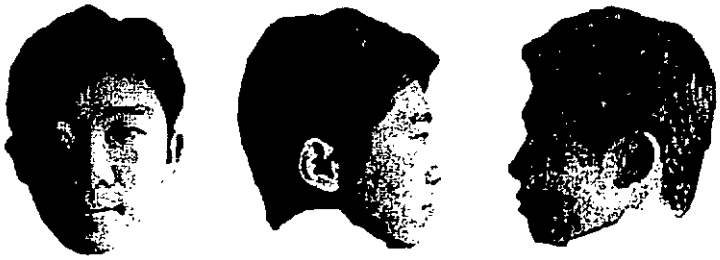
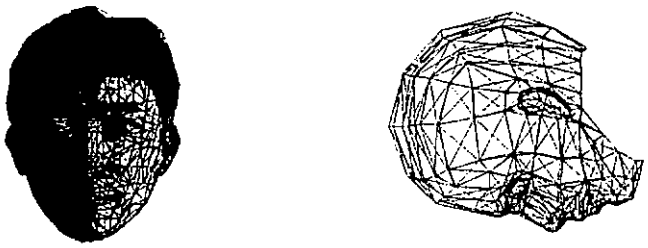


Figure 4.6 Coordinate system and generic model used in the simulation



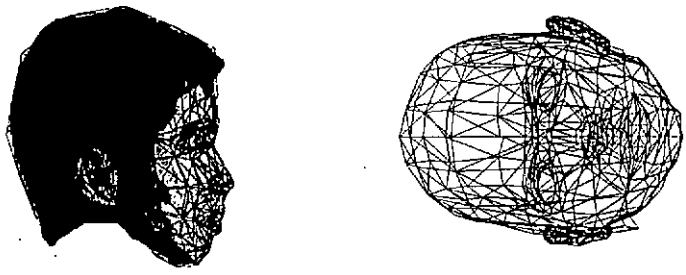
(a) (b) (c)

Figure 4.7 Three test input images in different orientations



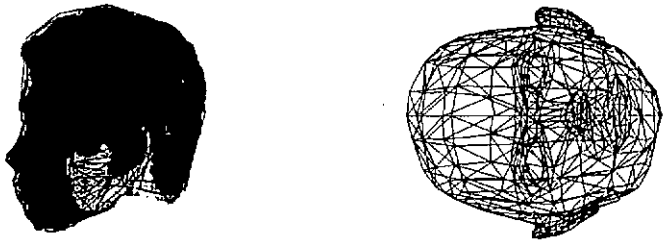
(a) Fitted Model (b) Another View

Figure 4.8 The intermediate model obtained with view 4.7a



(a) Fitted Model (b) Another View

Figure 4.9 The intermediate model obtained with view 4.7b



(a) Fitted Model (b) Another View

Figure 4.10 The intermediate model obtained with view 4.7c

The intermediate model generated in each view is then used to update the target model. The extraction procedures in obtaining the update model can be optimized with parallel computing. The confidence levels of the vertices of the intermediate models are calculated and the target model is modified with a reference to these parameters. Figure 4.11 shows three views of the target model.

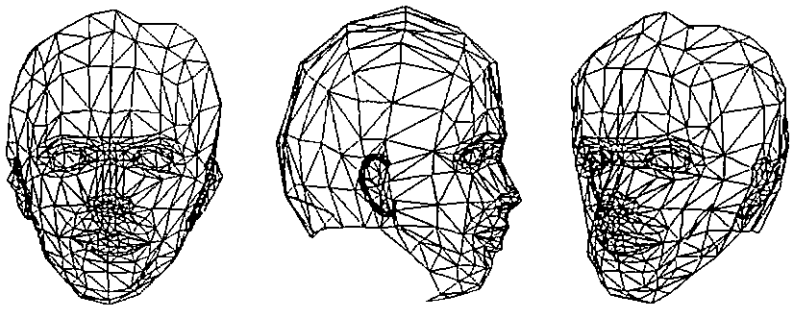


Figure 4.11 Resultant model in the simulation

Once the target model is obtained, one can map the spherical texture to the target model to construct different views of the object. Figure 4.12 shows some texture-mapped results obtained with the target model.

Figure 4.12 Three views of the rendered graphic model

4.4 Summary

In our work on model synthesis, the practicality is concerned. With the help of our proposed model generation technique, several advantages can be achieved:

1. It is robust to the object's orientation and size in the view.
2. The resultant model can be gradually built or updated with model update parameters extracted from different views under the guidance of a set of significance functions.
3. The extraction of update parameters from different views can be carried out in parallel without interference to each other.

Synthesizing a graphical model is always a time tolerating procedure in model-based coding systems. With the help of our approach, a specific human head model can be obtained more efficiently as compared with other conventional approaches, which is therefore more attractive to the practical use of model-based coding techniques in real applications.

Chapter 5

Robust Universal Texture Extraction Technique

5.1 Introduction

In the field of model-based coding, there are a lot of approaches in its implementation. Models, texture maps and motion vectors of the objects concerned are three basic elements commonly involved. Vast amount of research has been carried out on relevant topics of these elements. However, little effort has been paid to texture extraction.

Common texture extraction methods used in video coding nowadays include plane projection [51] and cylindrical texture projection such as [52]. However, both methods have their own limitations in practical applications.

In parallel projection methods such as [51], the texture pixels are obtained directly from the projection of the object of interest on a plane perpendicular to the camera's viewing direction. One of its significant weaknesses is that the generated texture map is limited for that particular view. Accordingly, the rendered result will not be favourable if the orientation of the object being rendered is not the one when the texture is extracted.

Cylindrical texture coordinates used in canonical model approach [53] can support multiple views in its texture generation process. However, this method is dedicated to the applications where the object of interest solely rotates about a single axis. All texture pixels

must be extracted from views the viewing directions of which are perpendicular to the axis of rotation. If this condition is violated, uncertainty will exist and error will be unavoidable when the object's image is reproduced with the map generated. Besides, the distortion of the texture pixels extracted and the texture map's sensitivity to projection error are position-dependent.

In short, when plane projection or cylindrical projection is adopted, more than one texture maps are required for a single object in order to reproduce different views of the object as the texture maps generated are orientation-oriented. These sorts of problems can be resolved with a spherical texture map [54].

In this chapter, a generalized texture extraction technique, which adopts spherical projection is discussed. This technique is robust to the orientation and the size of the object that appears in the view and it can be efficiently implemented with parallel processing technique. The texture map generated is robust in a way that it can be easily adjusted to work with its associated model to reproduce a view of the corresponding object even though the object is scaled, rotated and even slightly deformed. In such a case, only one single generic texture is required for a particular object. Besides, the distortion of the texture pixels extracted and the texture map's sensitivity to projection error are nearly not position-dependent as the texture pixels are uniformly distributed over the surface of the model. As a result, the proposed technique is very attractive in the practical use of model-based video coding techniques in real applications.

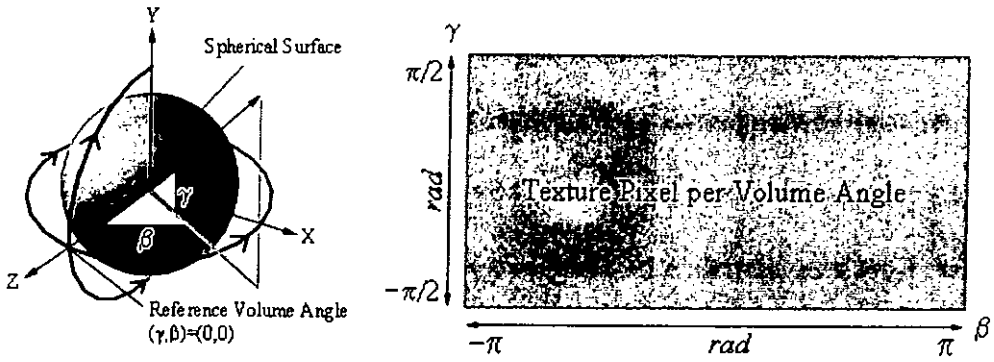
5.2 Robust Texture Extraction

Similar to the proposed model generation technique, the proposed texture extraction algorithm can again operate in two modes for different applications. The first operation mode extracts texture based on different views of the object of interest. The second operation mode composes texture map gradually by using the new information extracted from new video frames being processed. Again, parallel processing is allowed in both circumstances.

In the algorithm, pre-processing procedures have to be carried out before texture extraction is performed. Required pre-processing procedures include object orientation detection, visible surface determination, and model matching techniques. They have been discussed in the previous chapters.

5.2.1 Texture Grabbing

The key idea in extracting the texture in our spherical approach is to trace a ray from the center of the modeled object to its valid visible faces with ray tracing technique. Figure 5.1 shows an associated spherical coordinate system of the object and its connection to the texture map to be generated. The center of the object is at the origin of the coordinate system.



(a) Spherical Coordinates (b) Generic Texture Mapping Coordinates

Figure 5.1 Relationship between spherical coordinates and pixels of the generic texture map

The direction of a particular emitted ray from the center of the object can be described in vector form as ER(γ ,β)=(cosγ sinβ, sinγ, cosγ cosβ), where $-\pi/2 < \gamma < \pi/2$ and $-\pi < \beta < \pi$. If the ray pierces the plane containing a triangular face whose vertices are (x_0,y_0,z_0), (x_1,y_1,z_1) and (x_2,y_2,z_2) as shown in figure 5, the intersecting point (x,y,z) will be given by

$$\begin{cases} x = t\cos\gamma\sin\beta \\ y = t\sin\gamma \\ z = t\cos\gamma\cos\beta \end{cases} \tag{5.1}$$

where $$t = \frac{a_4}{a_1\cos\gamma\sin\beta + a_2\sin\gamma + a_3\cos\gamma\cos\beta},$$

and a_1, a_2, a_3, and a_4 are the coefficients of the plane equation

In particular, we have

$$\begin{vmatrix} x-x_0 & y-y_0 & z-z_0 \\ x_1-x_0 & y_1-y_0 & z_1-z_0 \\ x_2-x_0 & y_2-y_0 & z_2-z_0 \end{vmatrix} \equiv a_1x + a_2y + a_3z - a_4 = 0 \tag{5.2}$$

The validness of the point (x,y,z) is checked against if it lies on the specific triangular face. If it is valid, it will then be projected to the view plane with orthographic projection. The position to which projected on the view plane is recorded and the pixel of the corresponding position in the processing video frame is considered as the texture value of the map position (γ, β). The confidence level of the texture value obtained is defined as $C(\gamma,\beta) \equiv (\vec{n} \cdot \vec{v}_i)^2$, where $\vec{v}_i$ is a unit vector in the viewing direction and $\vec{n}$ is the unit normal vector of the triangular face. Note some positions of the texture map cannot be defined with the aforementioned procedures as some parts of the object are hidden in the view. For these positions, the confidence levels of their texture values are set to be zero, which implies the corresponding texture pixels are undefined.

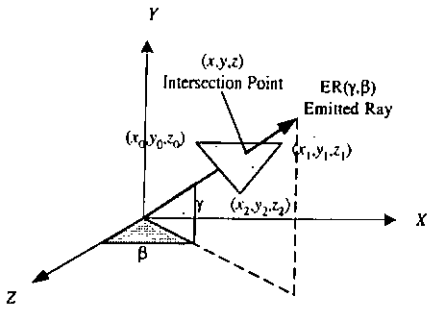


Figure 5.2 Emitted ray and its coincidence face

5.2.2 Integration of partial maps

For a particular view of the object, a corresponding texture map can be obtained with the aforementioned procedures. For the sake of reference, we refer the map associated with view orientation i to Ψ_i. Maps Ψ_i's can be integrated to form a single composite texture map Ψ. In particular, the textual pixel of the composite map at position (γ, β) is given by

$$\Psi(\gamma,\beta) = \frac{\sum_{i=1}^{N} C_i(\gamma,\beta)\Psi_i(\gamma,\beta)}{\sum_{i=1}^{N} C_i(\gamma,\beta)} \quad \text{if } \sum_{i=1}^{N} C_i(\gamma,\beta) \neq 0 \qquad (5.3)$$

where $\Psi_i(\gamma,\beta)$ denotes the texture value of Ψ_i at position (γ,β) and $C_i(\gamma,\beta)$ is its corresponding confidence level. N is the total number of views involved.

When the algorithm operates at its second operation mode, the composite texture map is gradually updated with new available views. Consider the case in videophone applications. A video sequence consists of a number of video frames. At the very beginning, the composite texture map is initialized to be a blank map. As the video runs, we get more video frames and hence more views of the object of interest. For each view i, a corresponding texture map Ψ_i can be obtained. This texture map is then merged with the current composite texture map to form a new composite texture map. The updating process is simple. Consider we have already updated the composite texture map with k different views. When the $(k+1)^{th}$ view comes, the composite texture map and its corresponding confidence levels are, respectively, updated by

$$\Psi_{(k+1)}(\gamma,\beta)=\frac{C_{(k)}(\gamma,\beta)\Psi_{(k)}(\gamma,\beta)+C_{k+1}(\gamma,\beta)\Psi_{k+1}(\gamma,\beta)}{C_{k+1}(\gamma,\beta)+C_{(k)}(\gamma,\beta)} \tag{5.4}$$

and $$C_{(k+1)}(\gamma,\beta)=C_{k+1}(\gamma,\beta)+C_{(k)}(\gamma,\beta) \qquad \text{for all } \gamma,\beta \tag{5.5}$$

where $\Psi_{(k)}(\gamma,\beta)$ and $C_{(k)}(\gamma,\beta)$ are, respectively, the pixel value of the composite texture map obtained with the first k different views and its corresponding confidence level at position (γ,β). Both $\Psi_{(0)}(\gamma,\beta)$ and $C_{(0)}(\gamma,\beta)$ are initialized to be zero at the very beginning. Views that come more than once are taken into account once only such that they

cannot contribute twice to the updating process in order not to bias the texture map to views that appear frequently. The updating procedure is repeated until the whole texture map is completely constructed.

5.3 Simulation Results

Simulation has been carried out to investigate the performance of the proposed scheme. In our simulation, a partial composite texture map is generated with three images of different views identified by $\vec{\theta} = (\theta x, \theta y, \theta z)$, where $\theta x, \theta y$ and θz are, respectively, the rotational angles along X, Y and Z axis. The coordinate system and the generic model used in the simulation are, respectively, shown in figures 5.3a and 5.3b. In particular, Figure 6b shows the projected view of the model of orientation $\vec{\theta} = (0,0,0)$.

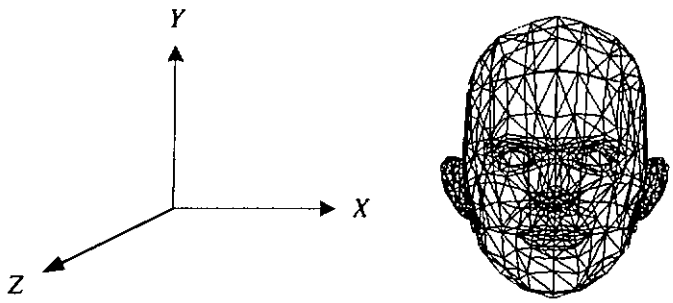
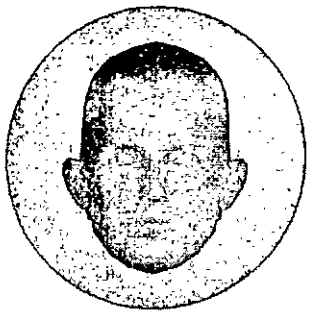


Figure 5.3 Coordinate system and generic model used in the simulation

For each view, the image of the head is first segmented from the input image. The orientation of the head appeared in the image is then determined with the *Point Matching* technique [38]. Next, the texture information of the head in the view is extracted by ray-tracing the emitted ray from the corresponding model's center to the surface of the model, which is already discussed in section 5.2.1.

Figures 5.4, 5.5 and 5.6 show, respectively, the local texture maps obtained from views $\bar{\theta}$=(0,0,0), $(0, \pi / 2, 0)$ and $(0, 3\pi / 2, 0)$.

(a) Segmented Image

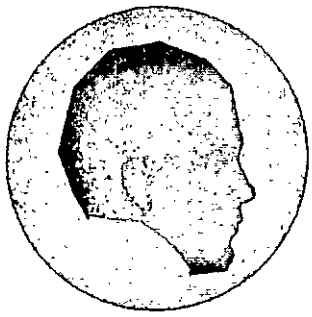


(b) Orientation of the model

(c) Extracted Texture

Figure 5.4 A local texture map obtained with a view of the head of orientation $\bar{\theta}$ = (0.0.0)

(a) Segmented image

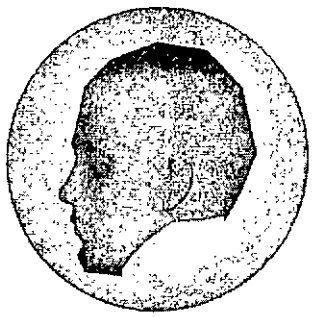


(b) Orientation of the model

(c) Extracted Texture

Figure 5.5 A local texture map obtained with a view of the head of orientation $\vec{\theta} = (0, \pi/2, 0)$.

(a) Segmented image



(b) Orientation of the model

(c) Extracted Texture

Figure 5.6 A local texture map obtained with a view of the head of orientation $\bar{\theta} = (0, 3\pi/2, 0)$.

The three local maps shown in figures 5.4c, 5.5c and 5.6c are then merged together to form a composite texture map shown in figure 5.7.

Figure 5.7 Resultant texture map obtained with figures 5.4c, 5.5c and 5.6c.

Once the composite texture map is obtained, it is applied to the corresponding graphic model to construct different views of the head. Figure 5.8 shows some rendering results of the model.

Figure 5.8 Views of the model rendered with the composite texture map generated

5.4 Summary

A robust texture extraction technique is important in model-based coding techniques. With the help of our texture extraction technique, several achievements over other techniques can be concluded:

1. It is robust to the object's orientation and size in the view.
2. The resultant map can be gradually built or updated with local texture maps extracted from different views under the guidance of confidence levels.
3. The extraction of update parameters from different views can be carried out in parallel without interference to each other.
4. There is only one single generic texture for a particular generic model.

Chapter 6

Synthetic Coding Using Model-based Coding Techniques

6.1 Introduction

Synthetic coding is a coding technique that synthesizes image or video contents according to a set of control parameters. The synthetic coding plays a very important role in modern image coding techniques, due to its advantages in saving more communication bandwidth as compared with traditional coding techniques. In this chapter, we will discuss our works on developing a synthetic coding system with model-based coding techniques.

In a synthetic coding system, a transformation engine, an animation engine, and a rendering engine are usually required in composing an image. These components work on models, texture maps, and motion vectors for synthesizing results. One of the motivations of our works is to adopt model-based coding in practical synthetic coding applications.

6.2 System Schematic

A model-based synthetic coding system, which adopts model-based coding scheme consists of a generic model library for image synthesis. As described in chapter 3, the core of our proposed system is divided into four functional stages, namely, an orientation manipulation stage, a dynamic feature manipulation stage, a model deformation stage, and a texture manipulation stage. These functional stages can be categorized into three abstract procedures, which are implemented with three separate engines as shown in figure 6.1.

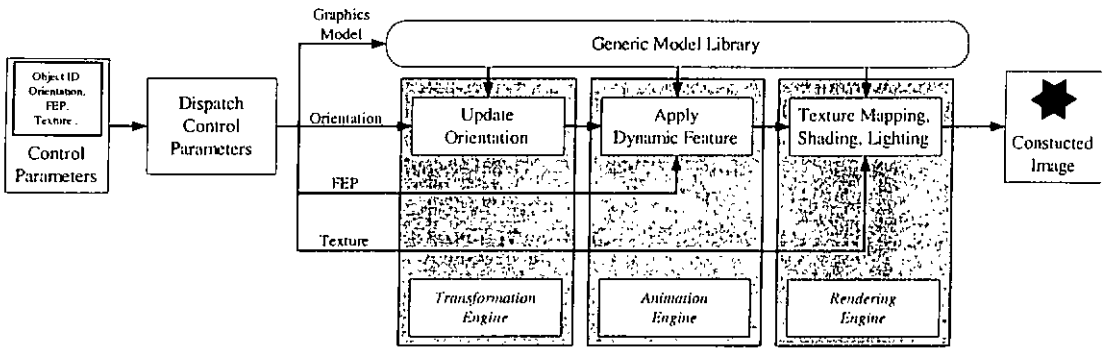


Figure 6.1 Schematic of synthetic coder

When control parameters arrive at the system, they are dispatched into transformation information, animation information, and rendering information such as texture information and lighting information. They are passed into the transformation engine, the animation engine, and the rendering engine respectively according to their functional purposes. The transformation engine manipulates global motion parameters such as scaling, translation, and rotation of an object. The animation engine manipulates local motion parameters such as surface deformation, action unit, and jaw motion of a human head object. The rendering engine renders instant model with texture maps and lighting information.

6.3 Transformation Engine

The various components making up the scene to be rendered may be defined in different coordinate systems. Coordinate system transformations are used to move models from one coordinate system into another coordinate system. The transformations used include operations such as translation, rotation, and scaling [55]. Homogeneous coordinates and coordinate transformations are used in describing and manipulating models among coordinate systems. Applications of homogenous coordinates in the transformations provide advantages in describing complex transformations into a combination of multiple transformations.

The transformation engine transforms models from its associate coordinate system into another coordinate system for animation or rendering processes. The transformations involved include modeling transformations and viewing transformations. Modeling transformation includes scaling of model size, rotation of model orientation, and translation of a model location. Viewing transformation manipulates viewing coordinates, which are specified by establishing the viewing position and the viewing direction in the world coordinate system. In addition, perspective transformation is also involved in viewing transformation to give perspective effect.

The modeling transformation in the world coordinate space is also known as global transformation. Figure 6.2 shows the user interface in our system for defining the rotation of a model. A rotation is declared by defining the initial degree and the end degree of the rotation about each axis in the world coordinate system. The information will be put in a global motion queue for animation procedures.

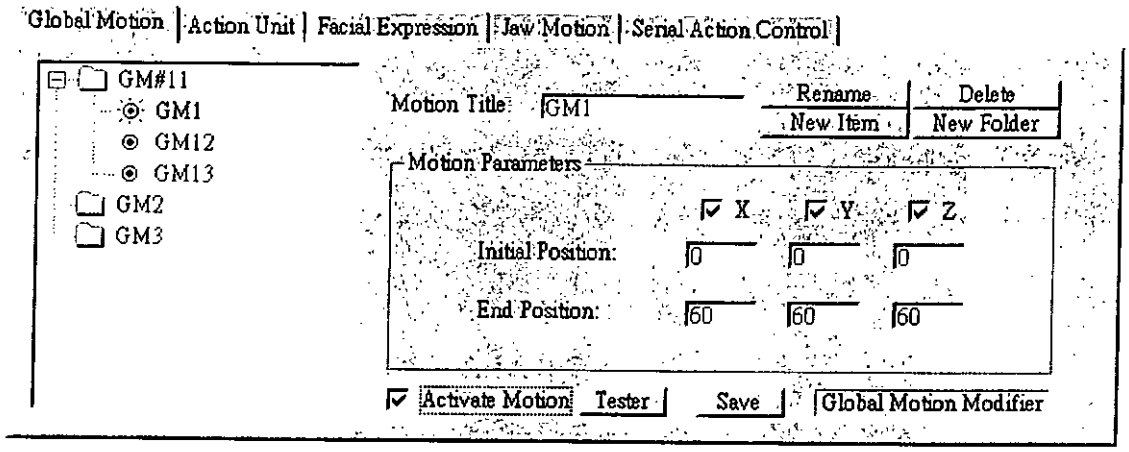


Figure 6.2 Global motion input interface

6.4 Animation Engine

The global motion of a model is handled by the transformation engine. In the animation engine, local motions such as facial action and jaw motion will be handled.

There are many approaches to animate facial action. For example, a key-pose method was proposed by Parke, in which desired shapes are specified at two time frames [56]. An interpolation algorithm is used to generate the necessary poses for animation within the two key frames. If this approach is used, the key poses in the key frames must be well specified. However, developing the required three-dimensional key-pose description is difficult. Therefore this approach becomes impractical when large amount of key poses are required. Owing to the difficulty of key-pose interpolation, Parke is motivated and developed parameterised face models [57]. Parameterised models could generate a wide range of faces and facial expression based on a small set of input control parameters. One of the advantages in using parameterised models is that limited amount of information is manipulated in producing an animation.

Since the eyes and the mouth are the most important organs in a facial expression, most of the expression parameters are related to these areas. Examples of useful expression parameters for the mouth are the degree of extension of a jaw, the position of upper lip, and the position of the corners of the mouth.

6.4.1 Implementation of Animation Engine

In our model-based human head coding system, a graphic model with 494 vertices and 966 polygons shown in figure 6.3 is adopted as the human head object model. This model resides in our generic model library.

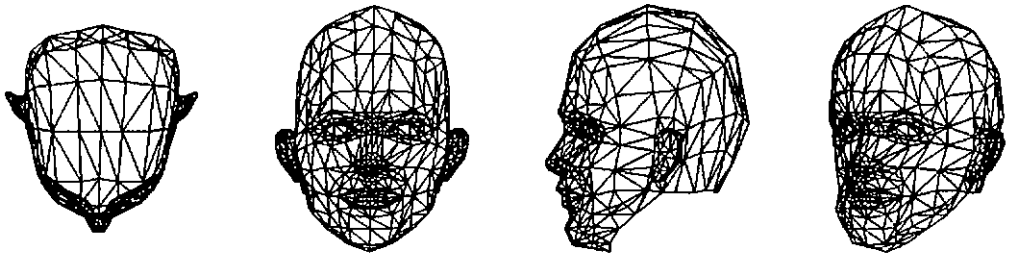


Figure 6.3 Generic human head model

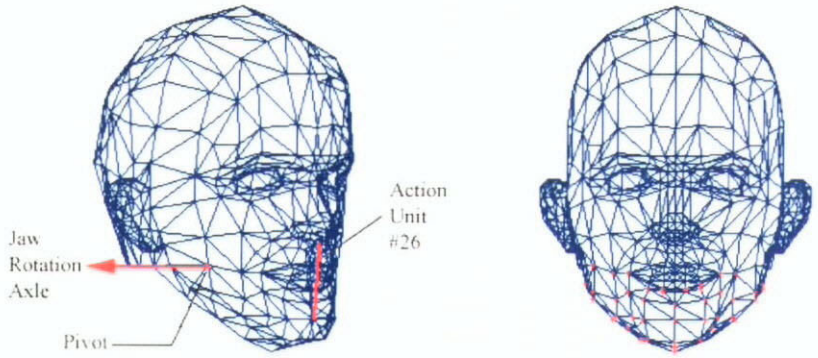
One of the earliest attempts to systematise the muscular activities that create the diverse facial expression is known as the Mimic Language, developed by Hjortsjo [58]. Nowadays, there is a similar and widely used notation of facial parameters. This notation was developed by Ekman and Friesen and is called the Facial Action Coding System (FACS) [49]. The FACS system includes about 46 independent facial actions, where the definitions of these Action Units (AU) are based on the activity of facial muscles. Examples of these action units are the inner brow raiser, the outer brow raiser, and the lip tightener.

The primary goal of the Facial Action Coding System was to develop a comprehensive system, which could reliably describe all possible visually distinguishable facial expressions. Facial actions involved by muscles that can be controlled voluntarily are parameterised, and any muscle not under voluntary control is not included in the approach. There are altogether 46 action units defined in the Facial Action Coding System. A single facial expression could be provided by composing as many as 20 action units. On the other hand, a facial expression may only involve one action unit. Not all action units can exist in a single composition. This is not only because some action units involve opposite actions, but because some of the actions can conceal the presence of others as well. A list of single facial action unit is given in Appendix A. The FACS provides a set of parameters that their

combinations describe a number of facial expressions. However, the implementation of FACS into computer graphics facial animation and synthetic coding remains abstract.

6.4.2 Jaw Motion

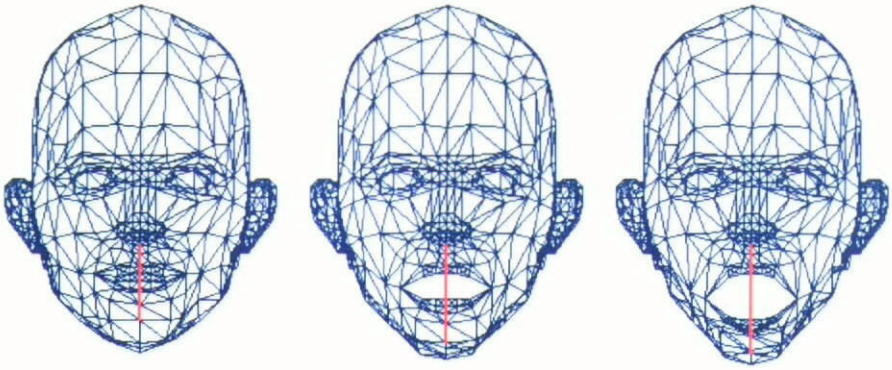
The mouth is opened by rotating the vertices of the lower part of a face about a jaw pivot axis. Jaw motion is necessary for the mouth to represent various speech and expression posture. Figure 6.4 illustrates the jaw rotation axis in our model and vertices affected by the rotation.



(a) Jaw rotation axle (b) Vertices affected by jaw rotation

Figure 6.4 Jaw motion

In figure 6.4, the vertices marked with red crosses are vertices that will be affected by the jaw rotation. The rotation axis of the jaw passes through the indicated jaw pivot point. The lower lip and the corner of the mouth rotate with the jaw. Positive jaw motion is shown in figure 6.5. In this figure, the jaw motion is controlled by an attached action unit labelled 26. Figure 6.5 shows the mouth in jaw opening action.



(a) AU#26 = 0 (b) AU#26 = 0.5 (c) AU#26 = 1.0

Figure 6.5 Jaw opening action

6.4.3 Facial Action

The surface of a human head model undergoes certain facial muscle actions to provide expressions. These facial surface changes are governed by facial actions. Typical handles of facial actions include brow raiser, lip corner puller, and chin raiser. In this engine, facial actions are realized with action units of Facial Action Coding System [49]. As an example, the brow raiser is controlled by AU#4, and the chin raiser is controlled by AU#17.

Action Units, in which their definitions are based on the activity of facial muscles, appear as motion vectors on the model surface. However, Facial Action Coding System concerns the description of facial motions only. It does not care hat the motions mean. In the implementation, several control parameters are added to give more favorable descriptions of the action units. The control parameters are listed as follow:

- identification of attach vertex
- identification of effect vertex
- stationary amplitude of action unit
- maximum amplitude of action unit
- minimum amplitude of action unit

Figure 6.6 illustrates the user interface implemented in our system for defining action units of the model. The previously mentioned parameters are defined through the interface.

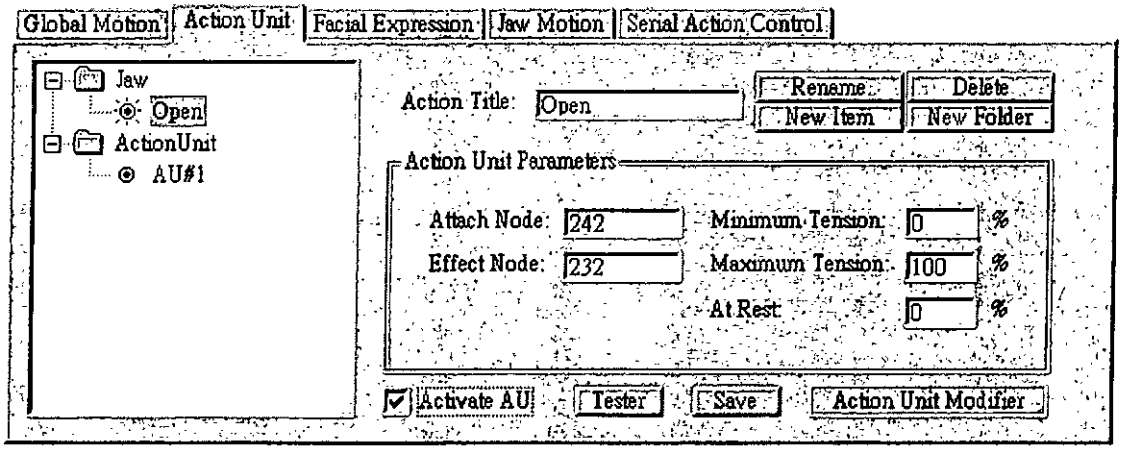


Figure 6.6 Action unit input interface

Table 6.1 shows the action units used in our system and their associated parameters.

Action Unit Name	Minimum Amplitude (%)	Maximum Amplitude (%)	Stationary Amplitude (%)
AU#1L	50	120	100
AU#1R	50	120	100
AU#2L	70	110	100
AU#2R	70	110	100
AU#3L	80	110	100
AU#3R	80	110	100
AU#4L	50	110	100
AU#4R	50	110	100
AU#9L	70	110	100
AU#9R	70	110	100
AU#10C	50	100	100
AU#10L	70	100	100
AU#10R	70	100	100
AU#12L	60	150	100
AU#12R	60	150	100
AU#16L	70	120	100
AU#16R	70	120	100
AU#17C	100	130	100

Table 6.1 Action Unit Parameters

Action Unit Name	Minimum Amplitude (%)	Maximum Amplitude (%)	Stationary Amplitude (%)
AU#18L	70	150	100
AU#18R	70	150	100
AU#20	50	150	100
AU#25	100	200	100
AU#26	0	100	0
AU#27L	90	130	100
AU#27R	90	130	100
AU#45L	20	100	100
AU#45R	20	100	100

Table 6.1 Action Unit Parameters (continue)

Figure 6.7 shows the geometrical locations of action unit parameters of our human head graphic model.

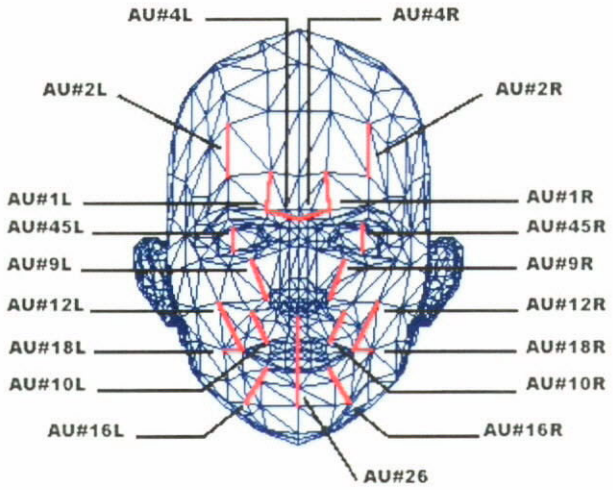


Figure 6.7 Locations of implemented Action Unit Parameters

When an action unit extends, geometrical locations of corresponding vertices of the graphic model are altered to fit the change. Let the destiny extension of an action unit AU_i be $E_{destiny,i}$. At a particular instance t, the extension of the action unit $E_{current,i}$ is given as

$$E_{current,i} = E_{current,i} + \sigma \times (E_{destiny,i} - E_{current,i}) \tag{6.1}$$

$$\sigma = \frac{1}{(t_{finish} - t) \times f} \tag{6.2}$$

where t_{finish} is the expected completion time for the AU extension, f is the frame rate, and $E_{current,i}$ is initialized with the at rest extension $E_{rest,i}$ at $t = 0$.

The geometrical location of an effect vertex of an action unit, say ($V_{x,effect}$, $V_{y,effect}$, $V_{z,effect}$) is given as

$$\begin{aligned} V_{x,effect} &= D_x \times D_{scale} + V_{x,attach} \\ V_{y,effect} &= D_y \times D_{scale} + V_{y,attach} \\ V_{z,effect} &= D_z \times D_{scale} + V_{z,attach} \end{aligned} \tag{6.3}$$

where ($V_{x,attach}$, $V_{y,attach}$, $V_{z,attach}$) is the geometrical location of an attach vertex of the action unit. The displacement parameters D_x, D_y, D_z, and displacement scale D_{scale} , are respectively defined as

$$\begin{aligned} D_x &= V_{x,effect} - V_{x,attach} \\ D_y &= V_{y,effect} - V_{y,attach} \\ D_z &= V_{z,effect} - V_{z,attach} \end{aligned} \tag{6.4}$$

$$D_{scale} = \sigma \times \frac{(E_{destiny,i} - E_{current,i}) + E_{current,i}}{100} \tag{6.5}$$

In order to provide natural deformation of the model surface, the change of the position of a vertex should affect its connected neighbors. In our system, the displacement of the affected vertex is stored and affects its neighbor vertices through the interpolation technique as discussed in section 4.2.3.

6.4.4 Facial Expression Parameters

Up to this point, global motion parameters, action units, and other aforementioned parameters act as input to produce corresponding human head image. However, a single facial expression may include up to twenty action units, which results in a complicated description for a facial gesture with the involved parameters. Therefore Facial Expression Parameters (FEP) are applied to interface all engines of our system. The FEP is a set of defined action units that are categorized by expressions. Each expression parameter holds the following facial action information:

- identification of action unit
- destiny amplitude of particular action unit in corresponding expression

In advance to the application of facial expression parameters, gesture is represented by three parameters instead of a lump sum of action unit parameters. The Facial Expression Parameters are defined by a particular status of a predefined combination of action units. Table 6.2 shows one of the FEP in the system and its tie to its correspondent action units. The complete set of implemented FEP is given in Appendix B.

Figure 6.8 shows the user interface in our system for defining Facial Expression Parameters. The information will be put into an action queue for animation procedures.

Action Unit Name	Amplitude in Expression (%)
AU#1L	120
AU#1R	120
AU#2L	110
AU#2R	110
AU#3L	110
AU#3R	110
AU#4L	70
AU#4R	70
AU#12L	140
AU#12R	140
AU#16L	110
AU#16R	110
AU#18L	85
AU#18R	85

Table 6.2 An example of Facial Expression Parameter

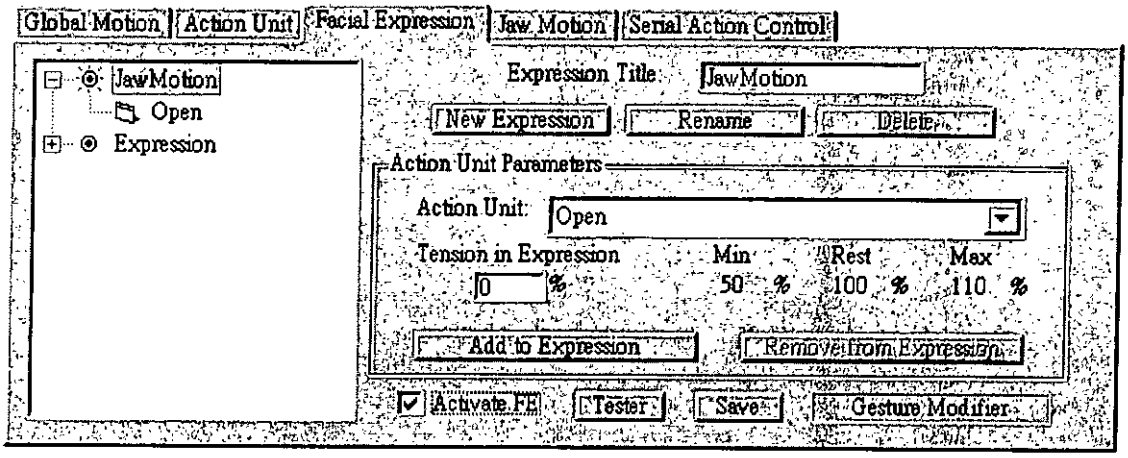


Figure 6.8 Facial expression input interface

6.5 Rendering Engine

The rendering engine is responsible for the tasks to be carried out in the final stage of the system. In our rendering engine, texture mapping and rendering are performed. As we have mentioned in chapter 5, our texture extraction technique produces spherical resultant texture. In modern polygonal model renderer, spherical mapping is widely adopted in texture mapping. One of the advantages in using spherical mapping in rendering is that

spherical texture gives more even texture resolution over the model surface as compared with cylindrical and plane mapping. Another superiority of our rendering engine as compared with other conventional implementations of model-based coding systems is that a real time three-dimensional graphic model renderer is employed in our rendering engine.

The implementation of our system builds on Microsoft Windows™ platform with the aid of DirectX Application Program Interfaces (API). The DirectX API provides a direct interface to the hardware installed in the host machines of our software. Direct3D, which is one of the interface of DirectX API enables direct low-level access to 3D graphics hardware and provides a very fast software based rendering of the complete 3D rendering pipeline. Figure 6.9 illustrates the relationships of DirectX, Windows, and a host machine.

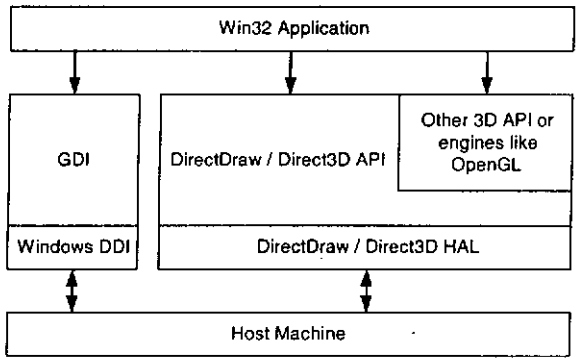


Figure 6.9 Relationships of DirectX, Windows, and computer

6.6 Serialization of Parameters

Action parameters can be put into a sequence to perform continuous animation of gesture with the three engines implemented in our system. Figure 6.10 shows the user input interface of our system for inputting and organizing action parameters.

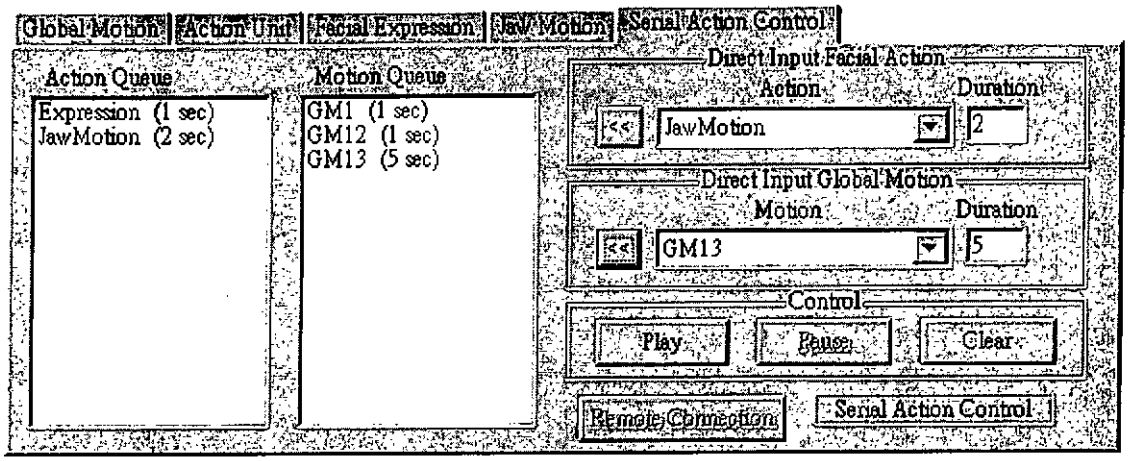


Figure 6.10 Serial action control interface

In the panel of the interface, two sets of control parameters, say the global motion parameter defined in section 6.3 and the facial action parameter defined in section 6.4, can be inputted. Their associated time duration stamp can also be inputted. The parameters are put into queues when they are inputted from the panel. There are two action queues implemented in the system. They are responsible for the facial action and global motion respectively. The animation engine iterates interpolations from the current amplitudes of action units to the maximum amplitude of the next entry in the queue. When the action queue reaches the end, the amplitudes of all action units are restored with stationary values.

6.7 Simulations

Simulations have been carried out to evaluate the performance of the implementation. In the simulations, Facial Expression Parameters and global motion parameters are inputted from the control panel as shown in figure 6.10. The FEP is defined by combination of action units that can be inputted from the interface shown in figure 6.6 and 6.8. The global motion parameters can be defined through the interface shown in figure 6.2.

Research in facial expression has concluded that there are six universal categories of facial expressions recognized across culture [59]. These categories are sadness, anger, joy, fear, disgust, and surprise. Within these categories there are wide range of variations in expression details. For different expression, a different set of action units are activated. These six universal expressions are shown in figure 6.11.

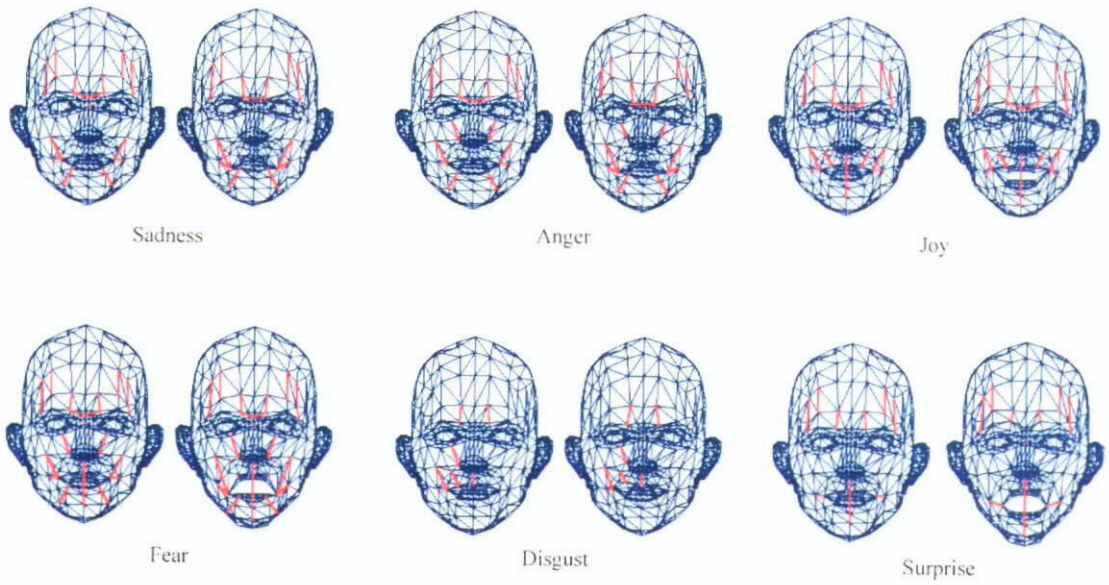


Figure 6.11 Implementations of universal expressions

In the figure, the red lines indicate the activated action units in each of the expressions. The figures on the left show the action units at normal extension, the figures on the right show the changes applied to particular action units. The rendered results are shown in figure 6.12.

Figure 6.12 Rendered results of universal expressions

In another simulation, we add global motion information to the human head model to produce more meaningful facial expressions. Figure 6.13 shows the rendered results of a human head model with facial expressions in various orientations. In the figure, four rendered results of facial expressions are shown. Corresponding orientation parameters are given in table 6.3.

Figure 6.13 Rendered results of expressions with orientations.

Location in the figure	Facial Expression	Orientation about X-axis	Orientation about Y-axis	Orientation about Z-axis
Upper Left	Surprise	-10	-20	5
Upper Right	Sadness	-10	0	0
Lower Left	Anger	10	0	0
Lower Right	Joy	5	-5	5

Table 6.3 Model parameters of figure 6.13

6.8 Summary

There are three basic components in composing the proposed model-based synthetic coding system. They are, namely, a transformation engine, an animation engine, and a rendering engine. In this chapter, the implementations of these three basic components are illustrated. In the transformation engine, transformations such as, translation, rotation and scaling are performed to graphic models. Homogeneous transformation is considered in the system to change the orientation of a model. These transformations respond to the global motion of the object of interest. For the animation engine, local motion is handled. In our system, a Facial Action Coding System is integrated in the animation engine. Jaw motion and facial action are two distinct types of local motions handled with the engine. These actions are represented in terms of action units defined in FACS and handled in the animation engine to emulate the actual human head movements. In order to refine facial actions with action units, Facial Expression Parameters are proposed. FEP describes facial gesture based on a set of action units. Lastly, the processed human head model is put into the rendering engine. Texture mapping is applied to the graphic model. The model is shaded with a real-time three-dimensional rendering engine. Direct3D is employed in developing our real time rendering engine. In our simulation, the rendering process can deliver up to 25 frames per

second. These results are very positive to support the use of our system for realising model-based human head gesture reconstruction.

Chapter 7

Conclusions

7.1 Summary of the work

In this work, we suggested some solutions to realize model-based coding in real world applications. These solutions include a robust model generation technique, a universal texture extraction technique, and a scheme to implement facial animation. In this section, we summarize the contributions we have made in these aspects.

In chapter 3, our proposed model-based coding system is presented. In our proposed system, several ideas have been integrated. The critical modifications we made with respect to conventional systems are in model generation and texture extraction. In order to make model-based coding practical for general applications, these two modifications are essential. As we have mentioned before, in most of the existing model-based coding systems, static predefined models and texture maps are used. For a particular object, its model and texture information is predefined according to the a priori knowledge of the object. This a priori knowledge is generally extracted with stereo graphics techniques or laser scanning. Therefore, these conventional systems are not practical in real-time applications due to the heavy computation effort required and the expensive equipments involved. Our proposed techniques solve this problem as they can update the model and build its texture dynamically. Because of these reasons, our system is capable of generating object models by deforming a single generic model. It is also capable of extracting texture maps for later applications.

The robust model generation technique described in chapter 4 was integrated into the proposed system due to our concern of the practicality of model-based coding techniques. This technique can be operated in two modes of operations. In the first operation mode, a target model is generated based on some given images of the object concerned prior to encoding. This pre-generated model is then applicable for later applications. In the second mode of operation, the target model can be gradually built or updated with new information extracted from new video frames being processed during encoding.

With the proposed model generation technique, the orientation and the size of the object in the view provided are no longer constraints in the generation of an object model. The resultant model can be gradually built or updated with parameters extracted from different views under the guidance of a set of significance functions. Furthermore, the extraction of parameters from different views can be carried out in parallel without interference from each other. Synthesizing a graphic model is no longer a time tolerating procedure when it is realized with our approach as compared with other conventional approaches, which is therefore more attractive for the practical use of model-based coding techniques in real applications.

In chapter 5, a universal texture extraction technique is presented. Similar to the aforementioned robust model generation technique, the presented texture extraction technique can be operated in two operation modes as well. In most of the existing systems, there are constraints in texture extraction. For instance, the orientation of the object whose texture is being extracted is limited by the physical constraints of the equipment involved and multiple texture maps are required for a particular object. In order to tackle these

limitations and make it favourable to real time applications, we proposed a texture extraction technique. With this technique, a single resultant texture map can be gradually built and updated with local texture maps extracted from different views according to the confidence level of each local texture map. The orientation and the size of the object in the view provided are no longer constraints in the texture extraction. Only one single generic texture is required for a particular model. The extraction of the update parameters can be carried out in parallel without interference from each other, so as to be favourable to real time applications.

Lastly, in chapter 6, a model-based synthetic coding system is presented. In our work on synthetic coding, model-based coding techniques are applied. Images are synthesized according to the input control parameters inputted to the system. These parameters carry animation control information and rendering information. In our work, a set of animation control parameters called Facial Expression Parameters are defined based on Action Units of Facial Animation Coding System. Making use of FEP results in an abstract description of facial gesture and a reduction in communication bandwidth as compared with other coding methods. Simulations have been carried out to use FEP to emulate muscle movement and different gestures successfully.

7.2 Future Developments

Some techniques for realizing a practical model-based coding system have been presented. In this section, we will discuss some possible direction of the future developments in this field of research.

7.2.1 Effective motion tracking techniques

Apart from the areas addressed in this thesis, there are some other areas in a model-based coding system, which should be realized in a more effective way. One of the common areas that tolerate the system performance is motion tracking. Recently, motion tracking is carried out by feature tracking [27]. In most cases, a feature tracking technique requires a set of predefined features, yet the tracking result is not good in term of accuracy. Again, as we have described in previous chapters, the necessity of a huge amount of predefined information is not favorable to general applications. Moreover, the accuracy of an object's orientation estimated during encoding directly affects the quality of the reconstructed video sequence. As a result, effective motion tracking technique based on minimal a priori knowledge is necessary. A possible solution for this problem may be to make use of stereo graphics. Accurate orientation information can be obtained from the disparity information presented in stereo images. We believe that motion tracking technique based on stereo graphics is one of the most effective substitutions to the existing motion tracking approaches [50].

7.2.2 Integration with MPEG-7

There is a new member of the MPEG family, called MPEG-7. This member introduces "Multimedia Content Description Interface". MPEG-7 specifies a standard set of descriptors that can be used to describe multimedia information. When MPEG-4 works with MPEG-7, model-based coding can take an important role in reconstructing multimedia content from MPEG-7 descriptions. In the future, studies of synthesizing techniques through description language will be of growing importance. Since MPEG-7 standardizes lower level abstraction of multimedia content, synthesizing techniques will be another important extension of model-based coding applications.

Appendix A

Single Facial Action Units

AU Number	FACS Name	Muscular Basis
1	Inner Brow Raiser	Frontalis, Pars Medialis
2	Outer Brow Raiser	Frontalis, Pars Lateralis
4	Brow Lowerer	Depressor Glabellae; Depressor Supercilli; Corrugator
5	Upper Lid Raiser	Levator Palebrae Superioris
6	Cheek Raiser	Orbicularis Oculi, Pars Orbitalis
7	Lid Tightener	Orbicularis Oculi, Pars Palebralis
8	Lips Toward Each Other	Orbicularis Oris
9	Nose Wrinkler	Levator Labii Superioris, Alaeque Nasi
10	Upper Lip Raiser	Levator Labii Superioris, Caput Infraorbitalis
11	Nasolabial Furrow Deepener	Zygomatic Minor
12	Lip Corner Puller	Zygomatic Major
13	Cheek puffer	Caninus
14	Dimpler	Buccinnator
15	Lip Corner Depressor	Triangularis
16	Lower Lip Depressor	Depressor Labii
17	Chin Raiser	Mentalis
18	Lip Puckerer	Incisivii Labii Superioris; Incisivii Labii Inferioris
19	Tongue Out	-
20	Lip Stretcher	Risorius
21	Neck Tightener	-
22	Lip Funneler	Orbicularis Oris
23	Lip Tightner	Orbicularis Oris
24	Lip Pressor	Orbicularis Oris
25	Lips Part	Depressor Labii, or Relaxation of Mentalis or Orbicularis Oris
26	Jaw Drop	Masetter; Temporal and Internal Pterygoid
27	Mouth Stretch	Ptergoids; Digastric
28	Lip suck	Orbicularis Oris
29	Jaw Thrust	-
30	Jaw Sideways	-

Table A.1 Single Facial Action Units

AU Number	FACS Name	Muscular Basis
31	Jaw Chencher	-
32	Lip Bite	-
33	Cheek Blow	-
34	Cheek Puff	-
35	Cheek Suck	-
36	Tongue Bulge	-
37	Lip Wipe	-
38	Nostril Dilator	Nasalis, Pars Alaris
39	Nostril Compressor	Nasalis, Pars Transversa and Depressor Septi Nasi
41	Lid Droop	Relaxation of Levator Palpebrae Superioris
42	Slit	Orbicularis Oculi
43	Eyes Closed	Relaxation of Levator Palpebrae Superioris
44	Squint	Orbicularis Oculi, Pars Palpebralis
45	Blink	Relaxation of Levator Palpebrae and Contraction o Orbicularis oculi, Pars Palpebralis
46	Wink	Orbicularis Oculi

Table A.1 Single Facial Action Units (Continue)

Appendix B

Implemented Facial Expression Parameters

Action Unit Name	Extension in Expression (%)
AU#1L	120
AU#1R	120
AU#2L	110
AU#2R	110
AU#3L	110
AU#3R	110
AU#4L	70
AU#4R	70
AU#12L	140
AU#12R	140
AU#16L	110
AU#16R	110
AU#18L	85
AU#18R	85

Table B.1 Sadness

Action Unit Name	Extension in Expression (%)
AU#1L	120
AU#1R	120
AU#2L	70
AU#2R	70
AU#3L	80
AU#3R	80
AU#4L	70
AU#4R	70
AU#9L	70
AU#9R	70
AU#12L	130
AU#12R	130
AU#16L	110
AU#16R	110
AU#18L	85
AU#18R	85

Table B.2 Anger

Action Unit Name	Extension in Expression (%)
AU#1L	90
AU#1R	90
AU#2L	90
AU#2R	90
AU#3L	120
AU#3R	120
AU#4L	105
AU#4R	105
AU#10C	110
AU#12L	70
AU#12R	70
AU#18L	110
AU#18R	110
AU#26	30

Table B.3 Joy

Action Unit Name	Extension in Expression (%)
AU#1L	90
AU#1R	90
AU#2L	110
AU#2R	110
AU#3L	110
AU#3R	110
AU#4L	110
AU#4R	80
AU#9L	80
AU#9R	110
AU#12L	140
AU#12R	140
AU#16L	120
AU#16R	120
AU#18L	80
AU#18R	80
AU#26	50

Table B.4 Fear

Action Unit Name	Extension in Expression (%)
AU#1L	110
AU#1R	110
AU#9L	80
AU#10C	90
AU#10L	80
AU#18L	80

Table B.5 Disgust

Action Unit Name	Extension in Expression (%)
AU#1L	60
AU#1R	60
AU#2L	90
AU#2R	90
AU#3L	105
AU#3R	105
AU#18L	150
AU#18R	150
AU#26	60

Table B.6 Surprise

References

[1] Musmann H.G., Personal Communication, University of Hannover, Germany, 1989.

[2] Garbor D. and Hill P.C.J., "Television band compression by contour interpolation", *Proc. IEE*, vol. 108, part B, no. 39, pp. 303-313, 1961.

[3] Pierce J.R., *Signals, Systems, and Noise*, pp. 139-140, Hutchison, London, 1962.

[4] Parke F.I., "A model for human faces that allows speech synchronized animation," *Computers & Graphics,* vol. 1, no. 1, pp.1-4, 1975.

[5] Phong B.T., "Illumiation for computer generated pictures", *Communications of the ACM*, vol. 18, no.6, pp. 311-316, 1975.

[6] Duchenne de Boulogne C.B., *The Mechanism of Human Facial Expression*, Paris: Jules Renard, 1862.

[7] Ekman P., "Methods for measuring facial action", *Handbook of methods in Nonverbal Behavior Research*, pp 45- 90, Cambridge University Press, 1982.

[8] Ekman P. and Friesen W.V., *Facial Action Coding System*, Consulting Psychologists Press, Palo Alto CA, 1977.

[9] Lippman A., "Semantic bandwidth compression: Speechmaker", *Proc. Int. Picture Coding Symp.*, pp. 29-30, Montreal, Canada, 1981.

[10] Forchheimer R. and Fahlander O., "Low bit-rate coding through animation", *Proc. Int. Picture Coding Symp.*, pp. 113-114, Davis CA, 1983,

[11] Rydfalk M., *CANDIDE, a parameterised face*, Dept. Elect. Eng., Linkoping University, Sweden, October 1987.

[12] Blinn J.F. and Newell M.E., "Texture and reflection in computer generated images". *Communications of the ACM*, vol. 19, no. 10, pp. 542-547, 1976

[13] Aizawa K., Harashima H., and Saito T., "Model-based image coding system-modeling a person's face and synthesis on facial expressions", *IEEE/IECE Global Telecommunications Conference*, pp. 45, Tokyo, 1987

[14] Aizawa K., Harashima H., and Saito T., "Model-based analysis-synthesis image coding system for very low-rate image transmission", *Proc. Int. Picture Coding Symp.* , paper 4.3, Turin, Italy, 1988.

[15] Aizawa K., Harashima H., and Saito T., "Model-based analysis-synthesis image coding (MBASIC) for a persons' face", *Signal Processing: Image Communication*, vol.1 no. 2, pp. 139-152, 1989.

[16] Suenaga Y. and Watanabe Y., "A method for the synchronized acquisition of cylindrical range and color data", *IEICE Transactions*, vol. E74, pp. 3407-3416, 1991.

[17] Cyberware, http://www.cyberware.com/

[18] Lijun Yin, and Basu A., "MPEG4 face modeling using fiducial points", *Proceedings on International Conference on Image Processing*, vol. 1, pp.109-112, 1997.

[19] Harashima H., Aizawa K., and Saito T., "Model-based analysis-synthesis coding of videotelephone images-conception and basic study of intelligent image coding", *IEICE Transactions*, vol. E72, no. 5, pp. 452-458, 1989.

[20] Musmann H.G., "Source models for image sequence coding," keynote address presented at the *International Workshop on Coding Technology for Very Low Bit-rate Video,* University of Essex, UK, April 1994.

[21] Haibo Li, Lundmark A., and Forchheimer R., "Image sequence coding at very low bit rates: A review", *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 589-609, September 1994.

[22] Aizawa K., Choi C.S., Harashima H., and Huang T.S., "Human facial motion analysis and synthesis with application to model-based coding", *Motion Analysis and Image Sequence Processing*, pp. 317-348, Norwell, Kluwer MA, 1993.

[23] Choi W.P., Lam K.M., and Siu W.C., "An adaptive active contour model for highly irregular boundaries", *IEEE Transactions on Pattern Recognition*, vol. 34, pp. 323-331, 1999.

[24] Bozdagi G., Tekalp A.M., and Onural L., "3-D motion estimation and wireframe adaptation including photometric effects for model-based coding of facial image sequences", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 246-256, June 1994.

[25] ISO/IEC FDIS 14496-2, Generic Coding of audio-visual objects: (MPEG-4 video), Final Draft International Standard, Document N2502, 1999.

[26] Chowdhury M.F., Clark A.F., Downton A.C., Morimatsu E., and Pearson D.E., "A switched model-based coder for video signals", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 216-227, June 1994.

[27] Colmenarez A., Frey B., Huang T.S., "Detection and tracking of faces and facial features", *IEEE International Conference on Image Processing*, vol. 1, pp. 657 – 661, October 1999.

[28] Ngan K.N., Rudianto R.L., "Automatic face location detection and tracking for model-based video coding", *IEEE International Conference on Signal Processing*, vol. 2, pp. 1098 – 1101, October 1996.

[29] Yuille A. L., Cohen D. S., and Hallinan P. W., "Feature extraction from faces using deformable templates", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 104-109, June 1989.

[30] Pixar Corporation, *The RenderMan Interface, Version 3.0*, Pixar Corporation, San Rafael CA, May 1988.

[31] Maxwell E.A., *Methods of plane projection geometry based on the use of general homogeneous coordinates*, Cambridge University Press, Cambridge, England, 1946.

[32] Maxwell E.A., *General homogeneous coordinates in space of three dimensions*, Cambridge University Press, Cambridge, England, 1951.

[33] Blinn J.F., "A homogeneous formulation of lines in 3-spaces", *SIGGRAPH 77*, pp. 237-241, 1977.

[34] Blinn J.F., and Newell M.E., "Clipping using homogeneous coordinates", *SIGGRAPH 78*, pp. 245-251, 1978.

[35] Sutherland I.E., Schumacker R.A., and Sproull R.F., "A characterization of ten hidden surface algorithms", *ACM Computing Surveys*, vol. 6, no. 1, pp. 1-55, 1974

[36] Foley J.D., van Dam A., Feiner S.K. and Hughes J.F., *Computer Graphics – Principles and Practice*, Addision-Wesley, Reading MA, 1989.

[37] Kokuer M. and Clark A. F., "Feature and model tracking for model-based coding", *International Conference on Image Processing and its Applications*, no. 354, pp. 135-138, 1992.

[38] Kin-Man Lam and Hong Yan, "Face recognition using point-matching based on a single front view", *RWC'97*, pp. 418-423, Japan, 1997.

[39] Li-an Tang and Huang T.S., "Automatic construction of 3D human face models based on 2D images", *IEEE ICIP*, vol. 3, pp. 467-470, 1996.

[40] Brunelli R., and Poggio T., "Face recognition: features versus templates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042-1052, October 1993.

[41] Turk M., and Pentland A., "Face processing: models for recognition", *SPIE Intelligent Robots and Computer Vision*, vol. 1192, pp. 22-32, 1989.

[42] Kirby M., and Sirovich L., "Application of the Karhunen-Loeve procedure for the characterization of human faces", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, January 1990.

[43] Turk, M. and Pentland A., "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1986.

[44] Moghaddam B., and Pentland A., "Face recognition using view-based and modular eigenspaces", *SPIE Automatic Systems for the identification and Inspection of Humans*, vol. 2277, pp. 12-18, July 1994.

[45] Terzopoulus D., and Waters K., "Analysis and synthesis of facial image sequences using physical and anatomical models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 569-579, June 1993.

[46] Mase K., "Recognition of facial expressions from optical flow", *IEICE Transactions*, Special Issue on Computer Vision and its Applications, vol. E74, no. 10, pp. 3474-3483, 1991.

[47] Mase K., and Pentland A., "Automatic lipreading by optical-flow analysis", *Systems and Computers*, vol. 22, no. 6, pp. 67-76, 1991.

[48] Li H., Roivainen P., and Forchheimer R., "3-D motion estimation in model-based facial image coding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 545-555, June 1993.

[49] Ekman P. and Friesen W.V., *Manual for the Facial Action Coding System*, Consulting Psychologists Press Inc., Palo Alto CA, 1978.

[50] Liang-Hua Chen and Wei-Chung Lin, "Visual surface segmentation from stereo", *Image and Vision Computing*, vol. 15, pp. 95-106, 1997.

[51] Eric A. Bier and Kenneth R. Sloan, "Two-part texture mappings", *IEEE Computer Graphics and Applications*, pp. 40-53, September 1986.

[52] Kurihara T. and Arai K., "A Transformation method for modeling and animation of the human face from photographs", *Computer Animation '91*, pp. 45-58, Springer Verlag, Tokyo, 1991.

[53] Alan Watt and Mark Watt, *Advanced Animation and Rendering Techniques – Theory and Practice*, pp. 177-182, Addison-Wesley, 1992.

[54] Parke F.I., and Keith Waters, *Computer Facial Animation*, pp.98-100, A K Peters Ltd.

[55] Foley J., can Dam A., Feiner S., and Hughes J., *Computer graphics: principles and practice, second edition*, Addison-Wesley, Reading MA, 1990.

[56] Parke F. I., "Computer generated animation of faces", Master's thesis, UTEC-CSc-72-120, University of Utah, Salt Lake City UT, June 1972

[57] Parke F. I., "A parameteric model for human faces", PhD thesis, UTEC-CSc-75-047, University of Utah, Salt Lake City UT, December 1974.

[58] Hjortsjo C. H., "Man's face and mimic language", Studentliterature, Lund, Sweden, 1970.

[59] Ekman P., "The argument and evidence about universals in facial expressions of emotion", *Handbook of Social Psychophysiology*, pp. 143-146, John Wiley, Chichester, 1989.